

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

SHOPIFY INC.
Petitioner

v.

DKR CONSULTING LLC
Patent Owner

IPR2025-00131
U.S. Patent No. 10,846,785

**DECLARATION OF TODD MOWRY, PH.D.
U.S. PATENT NO. 10,846,785
CHALLENGING CLAIMS 1–20**

TABLE OF CONTENTS

	Page
I. INTRODUCTION	1
II. QUALIFICATIONS AND PROFESSIONAL EXPERIENCE	1
III. MATERIALS CONSIDERED	5
IV. LEGAL PRINCIPLES.....	7
A. Claim Construction.....	7
B. Anticipation.....	8
C. Obviousness.....	8
V. LEVEL OF ORDINARY SKILL	11
VI. TECHNOLOGY BACKGROUND.....	12
A. Electronic commerce.....	12
B. Electronic Commerce in Social Networks	13
C. Widgets for Product Listings.....	13
VII. OVERVIEW OF THE '785 PATENT	15
A. Alleged Invention	15
B. Prosecution History	17
VIII. OVERVIEW OF THE PRIOR ART REFERENCES	20
A. Dierks	20
1. The “payment server 114,” “payment server 204,” and “payment processing system 308”	23
B. Johnston.....	25
1. Johnston is Entitled to its Provisional Filing Date.....	30
IX. SPECIFIC GROUNDS FOR UNPATENTABILITY	32
A. Ground 1: Claims 1–20 are Obvious over Dierks in view of Johnston	32
1. Independent Claim 1	32
2. Dependent Claim 2.....	80
3. Dependent Claim 3.....	83
4. Dependent Claim 4.....	86

5.	Dependent Claim 5.....	89
6.	Dependent Claim 6.....	90
7.	Independent Claim 7	90
8.	Dependent Claim 8.....	102
9.	Dependent Claim 9.....	103
10.	Dependent Claim 10	105
11.	Dependent Claim 11	108
12.	Dependent Claim 12	108
13.	Dependent Claim 13	108
14.	Independent Claim 14	109
15.	Dependent Claim 15	115
16.	Dependent Claim 16	115
17.	Dependent Claim 17	117
18.	Dependent Claim 18	118
19.	Dependent Claim 19	118
20.	Dependent Claim 20	118
X.	CONCLUSION.....	122
XI.	AVAILABILITY FOR CROSS EXAMINATION.....	122
XII.	RIGHT TO SUPPLEMENT	122
XIII.	JURAT	123

I, Todd Mowry, declare as follows:

I. INTRODUCTION

1. My name is Todd Mowry.

2. I have been retained as an expert in this proceeding by counsel for Shopify Inc. (“Shopify”) to provide my expert conclusions regarding the validity of claims 1–20 of U.S. Patent No. 10,846,785 (the “’785 patent”). For the reasons set forth below, it is my conclusion that claims 1–20 of the ’785 patent are invalid.

II. QUALIFICATIONS AND PROFESSIONAL EXPERIENCE

3. My qualifications are stated more fully in my curriculum vitae, which is attached as Exhibit A. Below is a summary of my education, work experience, and other qualifications.

4. I am a Professor in the Department of Computer Science at Carnegie Mellon University. I also have a courtesy appointment in the Department of Electrical and Computer Engineering. I have served on the faculty of Carnegie Mellon University for 27 years starting in 1997 through the present (2024).

5. I also served on the faculty of the University of Toronto for four years between 1993 and 1997, in the Department of Electrical and Computer Engineering and a courtesy appointment in the Department of Computer Science. Prior to that appointment, I served as a Graduate Research Assistant in the Department of Electrical Engineering at Stanford University for four years between 1989 and 1993.

6. As a faculty member, I have taught and continue to teach courses and directed research in computer systems and software, operating systems, distributed and network systems, parallel processing, and compiler optimization.

7. I received a B.S. degree in Electrical Engineering with Highest Distinction from the University of Virginia in May 1988. I received an M.S. in Electrical Engineering from Stanford University in June 1989, and a Ph.D. in Electrical Engineering from Stanford University in March 1994.

8. I have worked in the computer industry in various capacities. I was a part-time Computer Architect and then Computer Architecture Consultant at Silicon Graphics, Inc., in Mountain View, California (formerly MIPS Computer Systems in Sunnyvale, California) from 1989 to 1993 and 1993 to 1996, respectively. I was a Visiting Scientist at IBM in Toronto from 1996 to 2004. During that same time period (1996 to 2004), I was also a Member of the Technical Advisory Board of SandCraft, Inc. in Santa Clara, California. I was the Director of the Intel Research Pittsburgh Lab at Intel Corporation in Pittsburgh, Pennsylvania from 2004 to 2007.

9. I have authored 21 journal articles and 71 conference papers. I am also an inventor on five patents.

10. I have published a number of papers in the top research conferences in the fields of operating systems and data storage (including a paper that won the Best Paper Award at the USENIX 2nd Symposium on Operating Systems Design and

Implementation (OSDI '96)). *See, e.g.,* Olatunji Ruwase, Michael A. Kozuch, Phillip B. Gibbons, and Todd C. Mowry, *Guardrail: High Fidelity Correctness Checking of Device Drivers for Safeguarding I/O Operations*, in Proceedings of the Nineteenth International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS) (2014); Andrew Pavlo, Gustavo Angulo, Joy Arulraj, Haibin Lin, Jiexi Lin, Lin Ma, Prashanth Menon, Todd Mowry, Matthew Perron, Ian Quah, Siddharth Santurkar, Anthony Tomasic, Skye Toor, Dana Van Aken, Ziqi Wang, Yingjun Wu, Ran Xian, and Tieying Zhang, *Self-Driving Database Management Systems*, in Proceedings of the 2017 Conference on Innovative Data Systems Research (CIDR) (2017); and Amit Manjhi, Charles Garrod, Bruce M. Maggs, Todd C. Mowry, Anthony Tomasic, *Holistic Query Transformations for Dynamic Web Applications*, in 2009 IEEE 25th International Conference on Data Engineering (ICDE) (2009).

11. I have also been an active member of the Parallel Data Lab at Carnegie Mellon University since 1997, which is often described as “academia’s premiere storage systems research center.”

12. I am the recipient of several honors and awards: the Arthur Samuel Thesis Award (awarded by the Stanford Computer Science department to the top two Ph.D. theses in a given year), several IBM Faculty Development Awards (1996, 1997, 1998, 2000, 2001, 2002, and 2003), several Best Paper Awards (the Second

Symposium on Operating Systems Design and Implementation in 1996; the 20th International Conference on Data Engineering (ICDE) in 2004), the Alfred P. Sloan Research Fellowship (awarded to researchers in recognition of distinguished performance and a unique potential to make substantial contributions to their field), the Most Thought-Provoking Idea Award in 2004 (awarded by the Architectural Support for Programming Languages and Operating Systems (ASPLOS)), the TR100 Award in 1999 (awarded by MIT's Technology Review magazine to the top 100 most promising young innovators in science and technology), and I became an ACM Fellow in 2016.

13. I am a member of the Institute of Electrical and Electronics Engineers (IEEE) and the Association of Computing Machinery (ACM). I was the Editor-in-Chief of ACM Transactions on Computer Systems (the premier journal for computer systems research) from 2013–2018. I was an Associate Editor for the journal prior to that, since 2001. I was the Program Chair of the International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS) in 2010. I was the Co-Program Chair of the International Conference on Parallel Architectures and Compilation Techniques (PACT) in 2001. I have been on the programming committee in various years for ASPLOS, the International Symposium on Computer Architecture (ISCA), the International Symposium on

Microarchitectures, and the Workshop on Architectural and System Support for Improving Format.

14. Overall, I have over 30 years of experience in the field of computer science and, specifically, computer architecture, operating systems, and document storage and retrieval.

15. In light of the foregoing, I consider myself to be an expert in the field of computer science, and believe that I am qualified to provide an opinion as to what a person of ordinary skill in the art would have understood, known, or concluded regarding the subject matter of the '785 patent at the time of its alleged invention.

16. I am being compensated for my time at my ordinary hourly rate of \$650. My compensation is not dependent on the outcome of these proceedings or the content of my opinions. To the best of my knowledge, I have no financial interest in Shopify. I have no financial interest in the '785 patent.

III. MATERIALS CONSIDERED

17. In preparing this declaration, I have reviewed the claims, specification, and file history of the '785 patent.

18. I have reviewed and understand the references and exhibits cited in this declaration, including the following:

Exhibit	Description
1001	U.S. Patent No. 10,846,785

Exhibit	Description
1003	U.S. Patent No. 6,868,403 (“Wiser”)
1004	U.S. Patent Application Pub. No. 2008/0183593 (“Dierks”)
1005	U.S. Patent Application Pub. No. 2010/0114739 (“Johnston”)
1006	U.S. Patent Application Pub. No. 2010/0250382 (“Babaria”)
1008	U.S. Patent Application Pub. No. 2008/0255962 (“Chang”)
1009	’785 Patent File History
1010	U.S. Provisional Application No. 61/093,917 (“Johnston-Provisional”)
1011	U.S. Patent No. 9,324,098 (“Agrawal”)
1012	U.S. Patent No. 7,565,308 (“Bollay”)
1013	U.S. Patent Application Pub. No. 2008/0228595 (“Hill”)
1019	U.S. Patent Application Pub. No. US 2007/0045405 (“Rothschild”)
1020	U.S. Patent No. 8,046,288 (“LeRoux”)
1022	U.S. Patent No. 9,203,971 (“Narain”)
1023	U.S. Patent No. 8,069,088 (“Foulser”)
1024	D. Kossmann et al., “An evaluation of alternative architectures for transaction processing in the cloud,” <i>Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data</i> , June 2010
1032	ACM Digital Library for Kossmann

IV. LEGAL PRINCIPLES

19. I am not an attorney. For purposes of this declaration, I have been informed by Shopify’s counsel about certain aspects of the law that are relevant to my analysis and opinions, as stated in the following paragraphs. I have applied these legal principles in arriving at my conclusions expressed in this declaration.

A. Claim Construction

20. I understand that “claim construction” is the process of determining a patent claim’s meaning. I have been informed that claim construction is a matter of law and that in this proceeding, claim construction will be determined by the Patent Trial and Appeal Board (“the Board”). I understand that, during IPR, claims are construed according to the “*Phillips* standard.” I understand that the Board need only construe the claims when necessary to resolve the underlying controversy. I also understand that, given the close correlation between the asserted prior art and the challenged claims of the ’785 patent, the Board need not construe any terms of the challenged claims to resolve the underlying controversy, as any reasonable interpretation of those terms consistent with their plain meaning (as would have been understood by a person of ordinary skill in the art (“POSITA”) at the time of the invention, having taken into consideration the language of the claims, the specification, and the prosecution history of record) reads on the prior art.

B. Anticipation

21. I have been informed and understand that a claim is invalid if it was anticipated or not novel. I also have been informed and understand that a claim is anticipated if each and every element as set forth in the claim is found, either expressly or inherently described, in a single prior art reference.

C. Obviousness

22. I have been informed and understand that a patent claim is invalid if it would have been obvious to a POSITA at the time of the alleged invention. This means that, even if all of the requirements of a claim are not found in a single prior art reference, the claim is not patentable if the differences between the subject matter in the prior art and the subject matter in the claim would have been obvious to a POSITA at the time of the alleged invention.

23. I have been informed and understand that a determination of whether a claim was obvious should be based upon several factors, including, among others: (1) the level of ordinary skill in the art at the time of the alleged invention; (2) the scope and content of the prior art; (3) the differences, if any, that existed between the claimed invention and the prior art; and (4) any objective considerations of nonobviousness.

24. I have been informed and understand that when determining any differences between the invention covered by the patent claims and the prior art, one

should not look at the individual differences between the patent claims and the prior art in isolation, but rather consider the claimed invention as a whole and determine whether or not it would have been obvious in light of all of the prior art.

25. I have been informed and understand that a single reference can render a patent claim obvious if any differences between that reference and the claims would have been obvious to a POSITA. Alternatively, the teachings of two or more references may be combined in the same way as disclosed in the claims. I have been informed and understand that when deciding whether to combine the various items described in the prior art, the relevant question is whether the prior art combination would have been obvious to a person with ordinary skill in the art at the time of the invention. I have been further informed that it can be important to identify a teaching, suggestion, or motivation, in either the prior art or the knowledge of persons skilled in the art, that would have prompted a person of ordinary skill in the relevant field to combine the elements in the prior art in the way the claimed invention does.

26. In determining whether a combination based on either a single reference or multiple references would have been obvious, it is appropriate to consider, among other factors:

- whether the teachings of the prior art references disclose known concepts combined in familiar ways, and when combined, would yield predictable results;
- whether a POSITA would implement a predictable variation, and would see the benefit of doing so;
- whether the claimed elements represent one of a limited number of known design choices, and would have a reasonable expectation of success by a POSITA;
- whether a POSITA would have recognized a reason to combine known elements in the manner described in the claim;
- whether there is some teaching or suggestion in the prior art to make the modification or combination of elements claimed in the patent; and
- whether the innovation applies a known technique that had been used to improve a similar device or method in a similar way.

27. I have been informed and understand that a POSITA has ordinary creativity, and is not an automaton. I have been informed and understand that in considering obviousness, it is important not to determine obviousness using the benefit of hindsight derived from the patent being considered. I have been informed and understand that a patent composed of several elements is not proved obvious

merely by demonstrating that each of its elements was independently known in the prior art.

28. I have been informed and understand that to show a motivation to combine, one is not required to show how the references would be physically combined to render a claim obvious. The analysis is not focused on whether the features of one reference can be physically incorporated into the structure of another reference, but rather whether a POSITA would have been motivated to combine the teachings of the references to achieve the claimed invention. For example, a POSITA would understand how predictable variations of known concepts could be combined and would recognize that where multiple tools were available in a toolbox, any could be chosen.

V. LEVEL OF ORDINARY SKILL

29. I have considered the '785 patent from the perspective of a POSITA at the time of the alleged invention. A POSITA for the '785 patent would have had at least the equivalent of a bachelor's degree in computer science, computer engineering, or a related subject and two or more years of experience in a related field of distributed systems, transaction processing, and web-based user interfaces. Less work experience may be compensated by a higher level of education, such as a master's degree, and vice versa. At the time of the claimed invention, I would have qualified as a POSITA under this standard.

VI. TECHNOLOGY BACKGROUND

A. Electronic commerce

30. Electronic commerce—the trading of goods and services over the Internet—was well-known more than a decade before the '785 patent's priority date. For example, Wiser, claiming priority to a 1998 provisional application, explained that “[t]he rapid development of the **Internet** and the World Wide Web has primarily focused on these technologies as vehicles for **online commerce** for the distribution of their products.” Exhibit 1003, 1:15–17; *see also* Exhibit 1006, ¶ 4 (“Merchants, service providers, and/or private individuals, may offer items for sale on a web page appearing on a website accessible by a browser via the internet.”); Exhibit 1011, 1:15–16 (“Consumers routinely shop for and **purchase products and services from merchant web sites.**”); Exhibit 1012, 1:15–17 (“**Internet electronic commerce** allows businesses to conduct sales transactions over the Internet between customers (buyers) and vendors **supplying goods and services.**”); Exhibit 1013, ¶ 4 (“**Electronic commerce** sites allowing users to post and sell their own items ... have also become popular.”).¹ The '785 patent itself acknowledges that “[e]lectronic commerce on the Internet has become commonplace.” Exhibit 1001, 1:25–26.

¹ Emphasis and annotations added throughout unless stated otherwise.

B. Electronic Commerce in Social Networks

31. Performing electronic commerce in social networks was also well-known before the '785 patent's priority date. For example, USP Application Publication No. 2008/0228595 to Hill, filed in March 2007, disclosed a "system for supporting electronic commerce in social networks." Exhibit 1013, Title; *see also id.*, ¶ 8 ("The present invention provides users the ability to add commerce capability (e.g., buy and sell goods and services) and syndicate product information to numerous social network third-party websites."); *see also* Exhibit 1010 (Johnston-Provisional), specification at 7 ("The seller can be selling their content on Facebook, Bebo, and MySpace in a few minutes. ... The Storefront can then be placed on social networking sites such as Facebook, MySpace, Bebo...."). Similarly, Chang, filed in 2007, disclosed a distributed commerce system for distributing an "item page" that includes an item for sale to "social networking servers reflecting a plurality of social networking sites" such as "FACEBOOK™, MYSPACE™, LINKED-IN™." Exhibit 1008, Abstract, ¶ 41.

C. Widgets for Product Listings

32. Given the proliferation of electronic commerce over the Internet, it became common to use "widgets" or widget code to allow sellers to more easily and freely advertise (i.e., "list") their products for sale. A "widget" is simply a portable piece of code that can be installed and executed within an environment using the

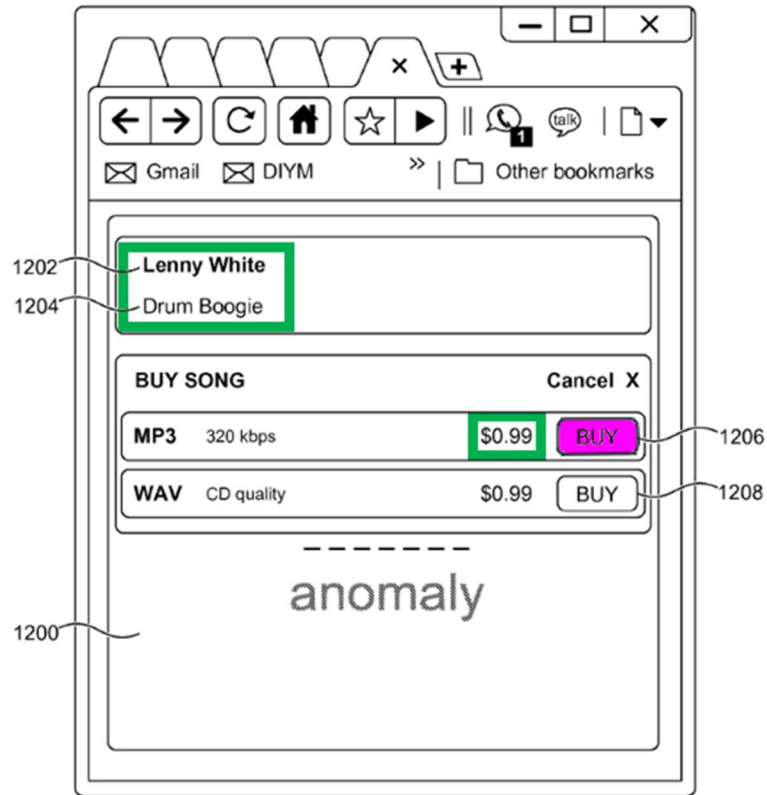
same or a different programming language. *See, e.g.*, Exhibit 1008, ¶ 31. For example, Chang disclosed widget servers that generate widget code for item/web pages related to items for sale that could be distributed to social networking sites. *See, e.g., id.* ¶¶ 37, 41, Figure 1. Notably, systems were known to gather pertinent information from sellers and generate such widget code. For instance, Dierks in 2008 disclosed a user interface that “presents a user with an easy, step-by-step process for entering relevant information and obtaining... code” that could be “cut-and-pasted into the user’s web page.” Exhibit 1004, ¶¶ 71, 74. Likewise, Johnston disclosed a server that provides user interfaces that allow sellers to generate “Embed code” and publish their product listings to the sellers’ social networking sites and “any HTML web site.” Exhibit 1005, Figures 3A–3G, ¶¶ 225–231. Adding functionality through the HTML code, such as a button that a purchaser presses to buy an item, was also well-known. Exhibit 1004, ¶ 54 (“The mark-up code may include HTML, XHTML code, or other forms of code for producing the display of a purchasing control, such as an ‘**add to cart**’ button.”). The widget code in the prior art comports with the ’785 patent’s description. *See* Exhibit 1001, 1:62–65 (“A web widget is portable software that can be installed and executed within a hypertext-markup-language web page by an end user that does not require additional compilation.”).

VII. OVERVIEW OF THE '785 PATENT

A. Alleged Invention

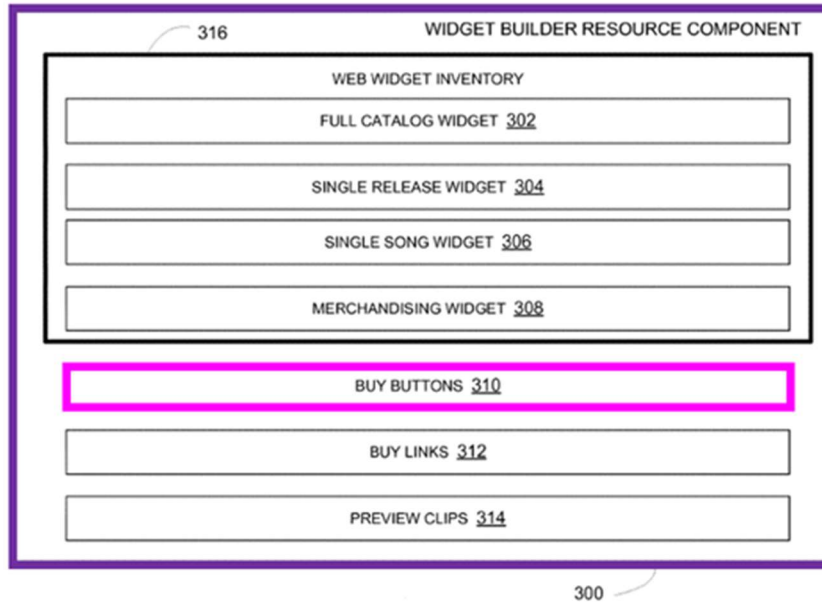
33. The claims of the '785 patent describe a system and method for generating a buy button and embedding the buy button into a web site. Exhibit 1001, claims 1, 7, and 14. The claims also describe an electronic store that is configured to facilitate payment processing related to the purchase of an item. Independent claims 1, 7, and 14 are similar. To the extent claims 7 and 14 include features that differ from claim 1, these features are taught by the prior art, as I discuss below in the grounds. *Id.*

34. Figure 12 of the '785 patent below illustrates an embedded **buy button 1206** that is used to purchase a song 1204 called “Drum Boogie.” *Id.*, 17:42–49. Figure 12 includes **information about the song**, including the name of song, the artist, and the price. *Id.* After clicking the **buy button 1206**, the user is brought to a checkout screen. *Id.*, Fig. 13, 17:50–61.



EX1001, FIG. 12

35. The '785 patent discloses that **buy buttons** are generated using a “**widget builder resource component 300**.” *Id.*, 8:40-43. As represented by Figure 3 below, the '785 patent indicates that the “**widget builder resource component 300** is used to create an inventory of ‘web widgets’ 316, **buy buttons 310**, buy links 312 and preview clips 314.” *Id.* 7:67–8:2.



EX1001, FIG. 3

36. As the ground herein demonstrates, generating and using buy buttons were well-known in the art before the '785 patent's priority date.

37. Also, an embeddable buy button and an electronic store that facilitates payment processing related to the purchase of an item were well-known in the art before the '785 patent's priority date.

B. Prosecution History

38. The '785 patent issued from USP Application No. 16/863,229, which was filed on April 30, 2020. The '785 patent is a continuation of U.S. Patent No. 11,455,678, which is a continuation of U.S. Patent No. 10,395,304, which is a continuation of U.S. Patent No. 9,679,298, which is a continuation-in-part of U.S. Application No. 12/852,090, now abandoned.

39. The examiner issued a sole Office Action on June 11, 2020, rejecting all claims 1–20 under (1) pre-AIA 35 U.S.C. 103(a) as being unpatentable over U.S. Patent Number 7,526,545 (“Jerome”), and (2) provisional double patenting as being unpatentable over claims 8–27 of USP Application 16/512,018. Exhibit 1009, 121–127.

40. The applicant filed a response on July 1, 2020, amending independent claims 1, 7 and 14. *Id.*, 161–175. Amended claim 1 is shown below.

1. (Currently Amended) A method to generate an embeddable buy button for an electronic commerce store, comprising:

receiving, by ~~an embeddable buy button builder executing on a first server operated by an electronic commerce provider~~, a selection of an item from an item inventory ~~and a representation of the item, wherein the item is to be listed for sale on the electronic commerce store~~;

generating, by the first server, the embeddable buy button, wherein the embeddable buy button is configured to display an image ~~the representation~~ of the item, and a hyperlink; and

embedding, by the first server, the embeddable buy button into a web page site, ~~wherein the web site is executed by a second server not operated by the electronic commerce provider~~,

wherein ~~the electronic commerce store~~ a checkout screen is displayed ~~within the web site~~ upon selection of the hyperlink, the ~~electronic commerce store~~ checkout screen configured to facilitate ~~payment processing related to a purchase of the item, and wherein the electronic commerce store is operated by the electronic commerce provider~~.

Id., 162.

41. Applicant equated the claimed “provider” to a “multimedia content distribution system 200” where the provider “executes both the embeddable buy button builder 204, as well as a transaction payment processing service 214.” *Id.*, 171. As I discuss herein, the payment processing system 308 of Dierks provides these same button-building and transaction payment processing services.

42. Applicant also argued that “*Jerome* does not disclose a system having a first server operated by an electronic commerce provider that allows the creation of embeddable buy buttons **and** where the same electronic commerce provider facilitates payment processing.” *Id.*, 172 (emphasis in original). Dierks likewise discloses these features, as I discuss below in detail.

43. A Notice of Allowance was issued on August 14, 2020. In the Notice of Allowance, the examiner stated that the prior art references did not disclose “‘embedding, by the first server, the embeddable buy button into a web site, wherein the web site is executed by a second server not operated by the electronic commerce provider’, and ‘wherein the electronic commerce store is displayed within the web site upon selection of the hyperlink, the electronic commerce store configured to facilitate payment processing related to a purchase of the item, and wherein the electronic commerce store is operated by the electronic commerce provider.’” *Id.*, 198. As the grounds below demonstrate, the prior art teaches these features.

VIII. OVERVIEW OF THE PRIOR ART REFERENCES

A. Dierks

44. Dierks discloses systems and methods that “permit relatively unsophisticated web page developers to integrate their web pages with an on-line payment system.” Exhibit 1004, ¶ 6. Similar to the ’785 patent, Dierks discloses techniques for embedding information into a web page, including a **description of an item** for sale, an **image representation** of the item, and an “**add to cart**” button that is used to buy the item, as shown below in Figure 1B. *Id.*, ¶¶ 36, 37, 73.

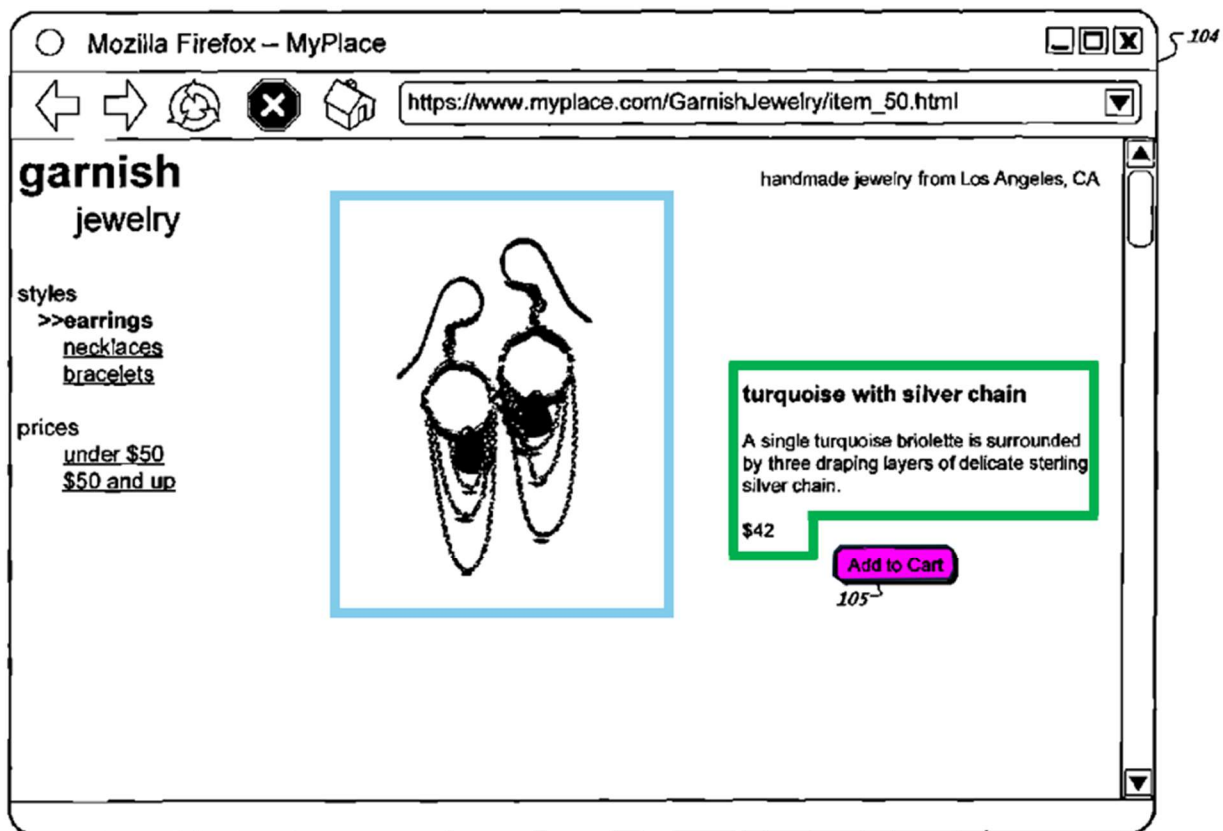


Exhibit 1004, FIG. 1B (annotated)

45. To create an “add to cart” button, Figure 4 of Dierks below discloses an interface for the user to enter **descriptive information** about the item for sale. *Id.*, ¶73. The user then clicks the “**Create Button Code.**” *Id.*, ¶74.

Mozilla Firefox - MyCheckout

https://www.mycheckout.com/Merchant/8675309/CreateItemCode/

1. Choose a button type.

☒ Basic Button

402 Add to Cart

404 Button with Options
Select this button type to offer different item options, such as sizes or colors.

2. Choose a button style.

My web page has a White 406 background.

3. Describe your item

Item Name: turquoise earrings 408 Price: 50.00 410 USD

Item Description: A single turquoise briolette is surrounded by three draping layers of delicate sterling silver chain. 412

414 Create Button Code

Insert this code for shopping cart:

```
<script type="text/javascript" src="https://www.mycheckout.com/shoppingcart.js?sellerId=8675309"></script><span id="mycheckout_status_bar"></span>
```

416

Insert this code for add to cart button:

```
<form action=https://www.mycheckout.com/Merchant/8675309/add onSubmit="return false;" method="post"><input type="hidden" name="item" value="PD94bWwgdMvyc2hvbj0iMS4wliBibmNvZGluZz0iVVRGLTgiPz4NCjxldGVlIHh0bG5zPSJodHRwOi8vY2h1Y2tvdXQuZ29vZ2ZxLmNvbS9zY2h0bWEvMmItDQogIDx0dWFudG0eT4xPC9xdWFudG0eT4NCiAgPHVuaXQ
```

418

46. Figure 3 of Dierks, reproduced below, discloses a **payment processing system 308** including an **HTML Generator 312** that “is configured to receive parameters regarding items to be sold by vendors and to generate code” that can be embedded into the web page to display the **description of the item** and the “**add to cart**” button shown above in Figure 1B. *Id.*, ¶ 62, Figure 3; *see also id.*, ¶¶ 36, 37, Fig. 4.

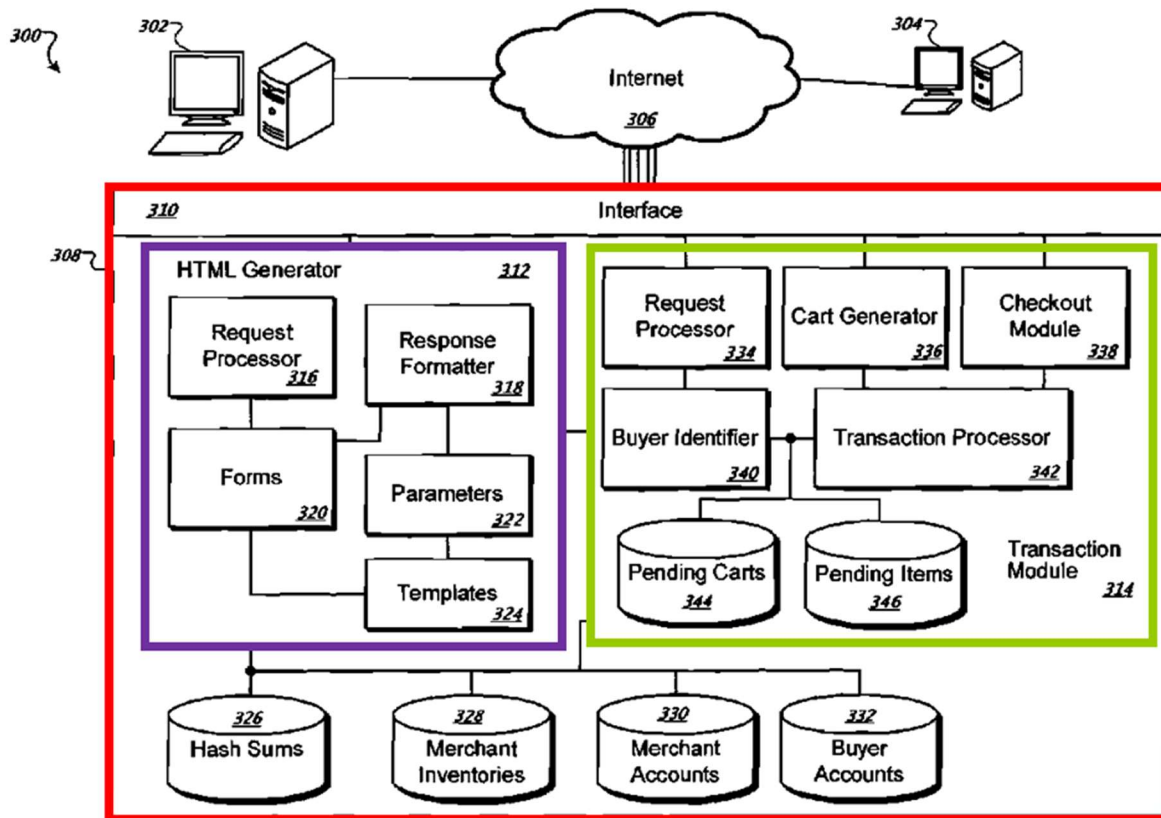


Exhibit 1004, FIG. 3 (annotated)

47. Figure 3 of Dierks discloses a **Transaction Module 314** that “receives requests from users who are seeking to buy items” and “provides information back to vendor sites to permit information about the transactions and other transactions to be reflected at the vendor sites.” *Id.*, ¶ 62. The **Transaction Module 314** shown in Figure 3 further includes a “Cart Generator 336” and a “Checkout Module 338.” *Id.*, Fig. 3, ¶ 70.

**1. The “payment server 114,” “payment server 204,” and
“payment processing system 308”**

48. Dierks discloses features with respect to a “payment server 114” and a “payment server 204.” Dierks also discloses features with respect to the “payment processing system 308.” A POSITA would have understood that the teachings regarding the payment server 114 and the payment server 204 apply to the payment processing system 308.

49. The payment servers 114, 204 and the payment processing system 308 are not disclosed as being separate embodiments. Instead, a POSITA would have understood that the patent application drafter decided to use different reference numerals for the servers when describing (1) a “*graphical conceptual flow* diagram” in Figure 1 that shows the *payment server 114* (*id.*, ¶ 19), (2) a “*conceptual flow* diagram” in Figure 2 that shows the *payment server 204* (*id.*, ¶ 21), and (3) a “*block* diagram” in Figure 3 that shows the *payment processing system 308*. (*id.*, ¶ 22). A POSITA would have understood that the block diagram approach used for Figure 3 allows for a more detailed disclosure of individual components. *Id.*, Fig. 3.

50. Dierks directly associates the payment server 114 in Figure 1 with the payment server 204 in Figure 2. *Id.*, ¶ 53 (“The seller system 202 may create a web page and, as shown by Arrow A, register the seller with a **payment server 204** such as **payment server 114**.”). Thus, a POSITA would have understood that the teachings in Dierks regarding the payment servers 114 and 204 are interchangeable.

51. Dierks further equates the payment server 204 in Figure 2 to the payment processing system 308 in Figure 3. For example, the payment server 204 in Figure 2 generates mark-up code used for providing the “add to cart” button. *Id.*, ¶ 54 (“As part of the registration process or at a later time, **the payment server 204 may generate and provide mark-up code** to the seller (Arrow A) so that the seller can cut-and-paste the code into the seller’s web page. The mark-up code may include HTML, XHTML code, or other forms of code for producing the display of a purchasing control, such as an ‘**add to cart**’ **button** that causes payment server 204 to record that a visitor to the web site wants to add a particular item associated with the control to his or her cart.”). The payment processing system 308 in Figure 3 also provides mark-up code to generate the “add to cart” button. *Id.*, ¶ 66, (“With the vendor web page complete and posted, prospective purchasers may visit the page and decide to click on a control, such as an ‘**add to cart**’ **control**, added by the **HTML generator 312** [i.e., part of the payment processing system 308].”).

52. Moreover, both the payment server 204 and payment processing system 308 are described as providing payment features, as evidenced by the word “payment” in their names. And the payment server 204 is described as providing a “shopping cart” feature just like the payment processing system 308. *See, e.g., id.*, ¶ 58 (“Once the **payment server 204** has updated a **shopping cart** for the consumer system 206 in its own records, it may pass information reflecting the updated

shopping cart back to the consumer system 206 (e.g., number of items in the cart and total dollar value of items in the cart), such as via XML messaging, so that the consumer system 206 can display the cart to the user.”); *id.*, ¶ 70 (“**Checkout module 338** [part of the **payment processing system 308**] may be accessed when a purchaser decides to finish a transaction and buy or edit items in his or her **shopping cart.**”).

53. Accordingly, a POSITA would have understood that the teachings in Dierks regarding payment server 114, payment server 204, and payment processing system 308 are related, such that the teachings made with reference to the payment servers 114, 204 are applicable to the payment processing system 308, and vice versa.

B. Johnston

54. Johnston relates to techniques that allow “a seller to establish at a hosted service a store front that is distributable across a plurality of third party hosts.” Exhibit 1005, Abstract. In particular, Johnston discloses, and illustrates in Figure 1A (reproduced below), a system including a **platform server 110** that communicates, over a network 104', such as the Internet, with devices that display a plurality of **Storefronts 102a–102c** that allows buyers to purchase content, products, and services from sellers. *See, e.g., id.*, ¶ 114.

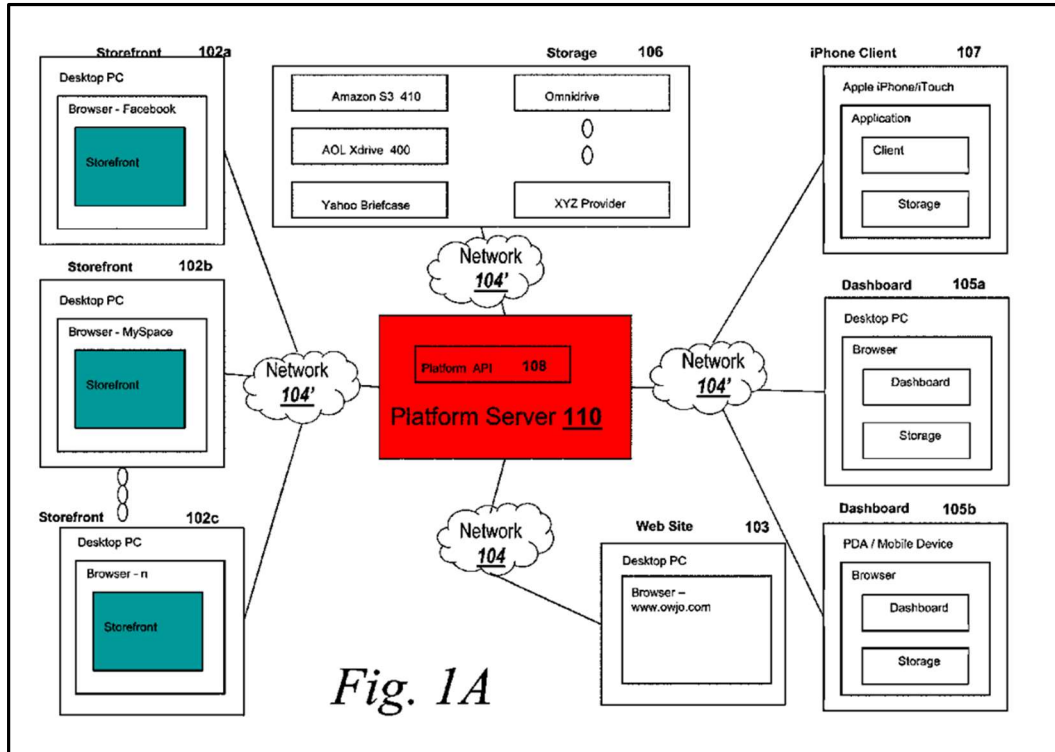


Exhibit 1005, FIG. 1A (annotated)

55. Similar to Dierks, Johnston's **platform server 110** includes a **Billing module 158** that processes financial transactions. *Id.*, ¶ 184, Figure 1C (reproduced below). The Billing module “handles, processes, manages and/or facilitates any of the financial related functions of the platform.” *Id.*

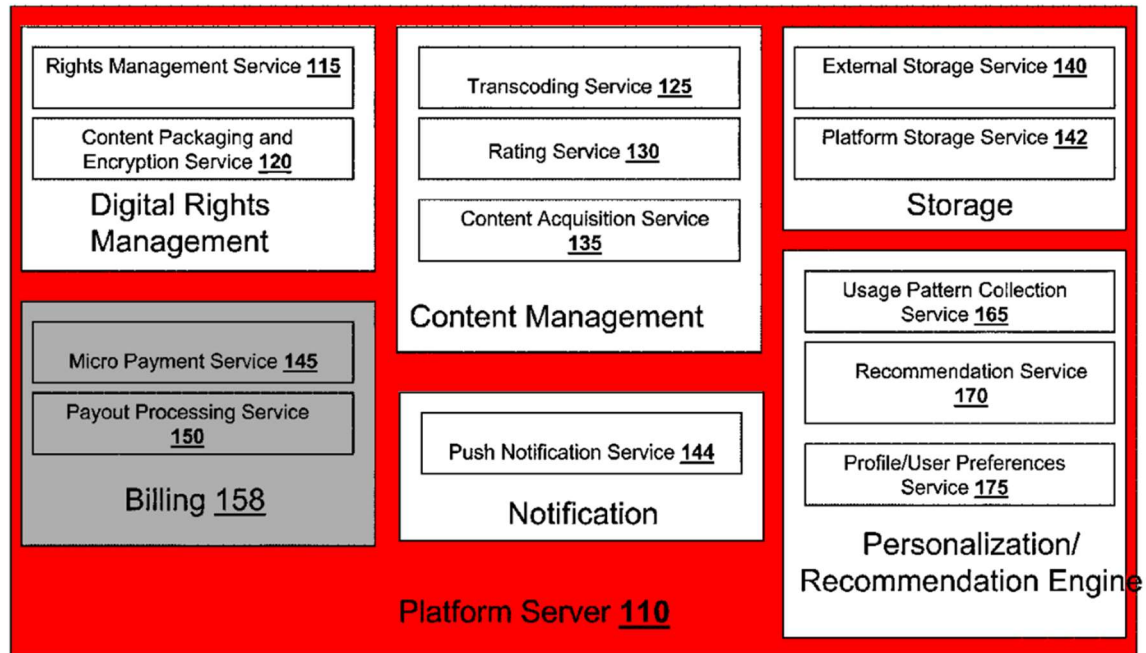


Exhibit 1005, FIG. 1C (annotated)

56. Also similar to Dierks, the **platform server 110** in Johnston provides user interfaces that allow sellers to create vendor Storefronts. *Id.*, Figs. 3A–3F, ¶¶ 225–230. Referring to Figures 3A–3G, Johnston discloses a seller “creating an OWJO Storefront via a user interface of the OWJO platform.” *Id.*, ¶ 225. For example, in Figure 3B (reproduced below), the user interface allows the seller to provide a **description of an item** (e.g., music) to be sold on the seller’s Storefront, including the price for the music. *Id.*, ¶ 226. The seller can do the same for videos, images, podcasts, and vodcasts. *Id.*, ¶¶ 227–230, Figs. 3C–3F.

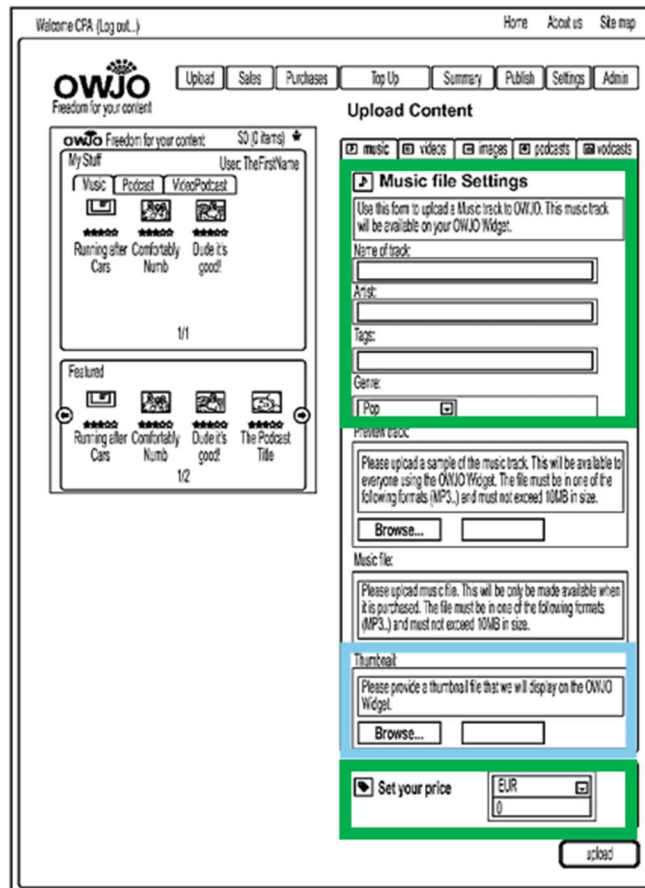


Exhibit 1005, FIG. 3B (annotated)

57. The interfaces of Johnston prompt the seller to select an image that represents the item for sale. For example, as shown above in Figure 3B, the interface prompts the seller to “**provide a thumbnail file that we will display on the OWJO Widget.**” *Id.*, Fig. 3B; *see also id.*, ¶ 226 (“Using a web browser ... a seller can create a music track by **specifying ... a thumbnail image that can be displayed in the OWJO Storefront.**”).

58. Also similar to Dierks, the Platform server 110 allows sellers to publish their Storefronts to “any HTML web site.” *Id.*, ¶ 231, Fig. 3G. Referring to Figures

4A–4C, Johnston describes an exemplary **Storefront** used to sell items. As can be seen in Figure 4B below, the **Storefront** includes (i) information about a music track for sale such as “**Artist**” and “**Genre**,” (ii) the “**Price: EUR \$ 25**” of the music track, and (iii) an “**Add to Cart**” button. This is similar to the product listing that the ’785 patent illustrates in Figure 12 (reproduced below alongside Johnston’s Figure 4B).

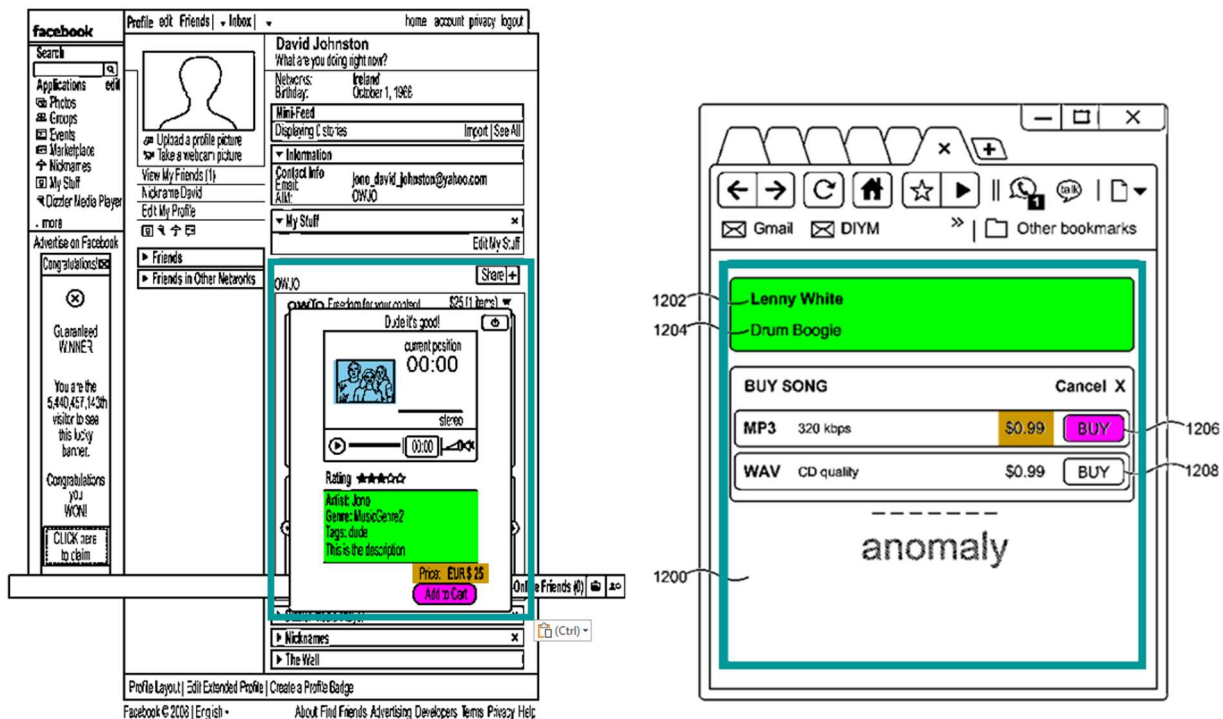


Exhibit 1005, FIG. 4B (annotated)

Exhibit 1001, FIG. 12 (annotated)

59. As shown in Figure 4B, the **Storefront** displays the **image representation** of the item for sale. Exhibit 1005, ¶ 222 (“The OWJO Client/OWJO Dashboard can **display thumbnails** representing the content that buyer A has rights to access or has otherwise purchased.”); *id.*, ¶ 289 (“**content** may be **displayed with thumbnails**, links or some other type or **form of representation** of the

content.”). The **image representation** of the item for sale in Johnston is analogous to the image representation of the item for sale, e.g., earrings, discussed above in Dierks, since they both provide an image representation of an item.

1. Johnston is Entitled to its Provisional Filing Date

60. Johnston is entitled to the September 3, 2008 filing date of provisional application No. 61/093,917 (“Johnston-Provisional”) to which it claims priority.

61. I have been informed that, for purposes of § 102(e), a published patent application is entitled to use the filing date of a beneficially claimed provisional application as an effective prior art date, provided that 1) the subject matter relied upon in the application is described in the provisional application; and 2) at least one of the claims in the application is supported by the written description of the provisional application.

62. As the following exemplary citations show, the subject matter relied upon in Johnston is described in Johnston-Provisional.

Johnston (Exhibit 1005)	Johnston-Provisional (Exhibit 1010)
Abstract	pp. 4–5, 7; FIG. 1
FIG. 1A	Fig. 1A
FIGS. 3A–3G, 4A–4C	FIGS. 3A–3G, 4A–4C
¶ 2	pp. 4–5 ²

² Page cites refer to the specification.

¶ 4	p. 1
¶ 21	pp. 4–5
¶ 68	p. 9
¶ 114	pp. 11-12, 16–17
¶ 118	pp. 12–13
¶ 178	p. 23
¶ 184	p. 24
¶ 204	p. 29
¶ 222	p. 34
¶¶ 225–231	pp. 35–38
¶ 272	pp. 42–43
¶ 289	p. 34
¶ 300	pp. 4-5, 6, 19, 22, 27

63. In addition, the exemplary citations below demonstrate that at least claim 1 of Johnston is supported by the written description of Johnston-Provisional.

Johnston (Exhibit 1005), claim 1	Johnston-Provisional (Exhibit 1010)
1. A method of self-service by a seller to establish at a hosted service a store front that is distributable across a plurality of third party hosts, the method comprising:	Figs. 1A, 3H; pp. 11–12, 38–39 ³

³ Page cites refer to the specification.

a) receiving, by a hosted service operating on one or more servers, specification of one or more items to be displayed for sale in a store front for a seller;	Figs. 1A, 3H; pp. 11–12, 38–39
b) generating, by the hosted service responsive to the specification, a portable widget for the store front identifying the one or more items for sale;	Figs. 1A, 3H; pp. 11–12, 38
c) receiving, by the hosted service from the seller, identification of one or more third party hosts to which to publish the portable widget of the store front; and	Figs. 1A, 3H; pp. 11–12, 39
d) initiating, by the hosted service, publishing of the portable widget of the store front to the one or more third party hosts.	Figs. 1A, 3H; pp. 11–12, 39

64. Accordingly, for purposes of 35 U.S.C. § 102(e), Johnston is entitled to an effective prior art date of September 3, 2008 filing date of Johnston-Provisional.

IX. SPECIFIC GROUNDS FOR UNPATENTABILITY

65. In my opinion, for the reasons noted below, claims 1–20 of the '785 patent would have been obvious in light of the prior art.

A. Ground 1: Claims 1–20 are Obvious over Dierks in view of Johnston

1. Independent Claim 1

a) 1[preamble]: “A method for embedding a buy button for an electronic commerce store, comprising:”

66. In my opinion, regardless of whether the preamble is limiting, Dierks discloses it. Figure 1B of Dierks shows a web site (“an electronic store”) for selling a product using an embedded “**add to cart**” button 105 (“a buy button”). Exhibit 1004, ¶ 36 (“[D]isplay 104 shows a selectable control in the form of an ‘**add to cart**’

button 105 that a purchaser may select **to purchase** a unit of the product—here a pair of earrings.”). Dierks also discloses that the electronic store includes a checkout screen that is accessed after the “Proceed to Checkout” button 118 in Figure 1B is pressed, as I discuss below for element 1[g].

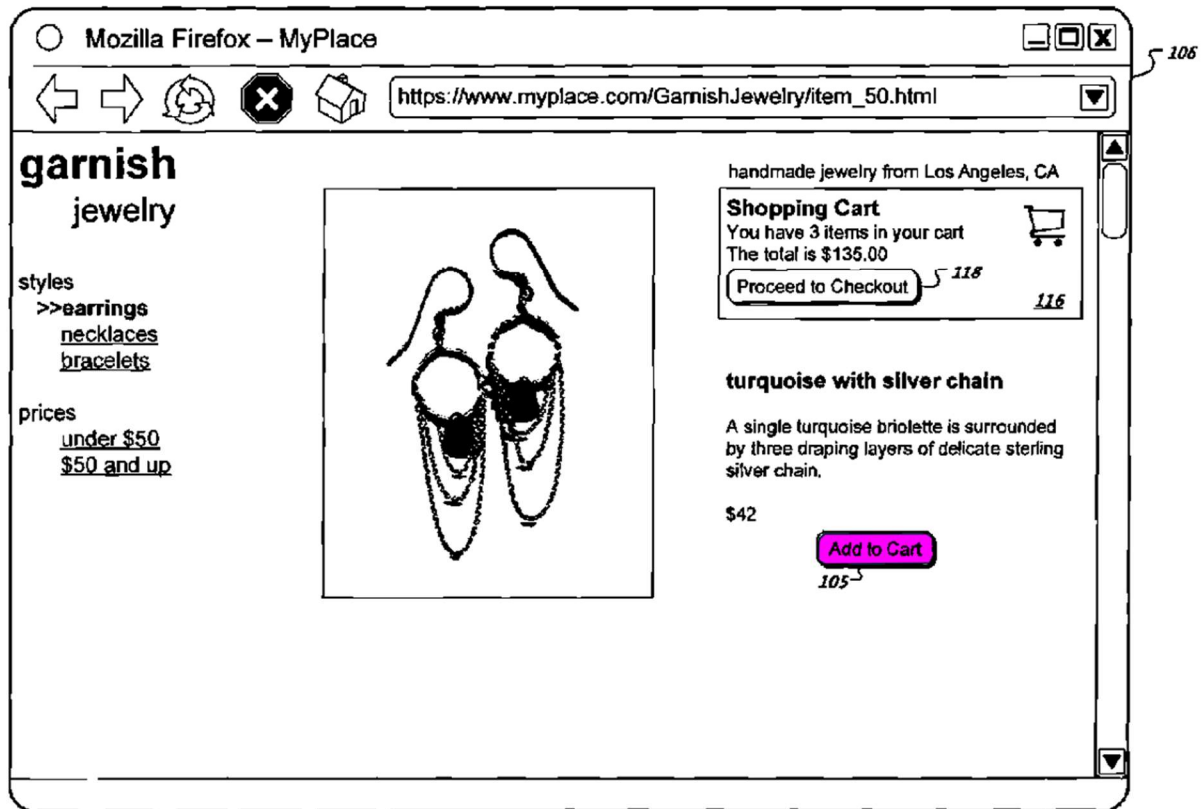


Exhibit 1004, FIG. 1B (annotated)

67. Dierks discloses a method for embedding the “**add to cart**” button 105. *Id.*, ¶ 8 (“In one implementation, a **computer-implemented method** is disclosed. ... Also, the code for the merchant web page can include **buy button** code generated by a service associated with the remote open payment system.”).

68. The “**add to cart**” button 105 of Dierks is analogous to the “buy button” of the ’785 patent. For both, pressing the button leads the user to a display showing the total cost of the item. The user then presses another button that brings the user to a further web page for entering payment-related information.

69. In particular, after the user presses the “add to cart” button 105 in Dierks, the user is shown a shopping cart that displays the total cost of the item, as I discuss below in more detail for element 1[f]. The user then presses a “proceed to checkout” button 118 that brings the user to a screen for providing payment-related information, as I discuss below for element 1[g].

70. A similar process occurs in the ’785 patent. Figure 12 of the ’785 patent shows a song 1204 called “Drum Boogie” that is for sale. Exhibit 1001, 17:57–58 (“the name of the selection (i.e., Drum Boogie”). The user presses the buy button 1206 of Figure 12 to indicate an intent to purchase the album. *Id.*, 17:16-21 (“after a consumer selects the buy button 1206 to purchase the MP3 format file”). The user is then brought to a screen shown in Figure 13 of the ’785 patent that displays the total cost of the item. *Id.*, 17:50-57 (“FIG. 13 is an **illustration of the screen shown in a web widget after clicking on a buy button**. In this embodiment of a web widget 1300, after a consumer **selects the buy button 1206** to purchase the MP3 format file, a checkout screen appears within the web widget 1300 as shown in this figure. The checkout screen of this web widget 1300 includes the **total cost of the content file**

and the number of files purchased, which items are shown in region 1304.”). Figure 13 is also described as providing “purchase request confirmation.” *Id.*, 2:66-67.

71. Figure 13 of the ’785 patent illustrates a “pay now with amazon.com” button 1306. A POSITA would have understood that the “pay now with amazon.com” button 1306 brings the user to the screen of Figure 14 so that the user can enter payment-related information. *Id.*, 17:62-18:1 (“FIG. 14 is an illustration of a web page for a transaction processing service 112 in one embodiment. In the illustrated embodiment, the transaction page for the Amazon Flexible Payment Service (i.e., the “Amazon FPS”) is shown. In this embodiment, this web page 1400 is the location where the consumer would enter credit card, debit card or bank account information.”).

72. Thus, the “add to cart” button in Dierks is analogous to the buy button 1206 in the ’785 patent since, when pressed, both lead to displaying the total cost of the item and providing an additional button that the user presses to access another web page where the user enters payment-related information.

73. Therefore, Dierks discloses the preamble of claim 1.

b) 1[a]: “receiving, by an embeddable buy button builder executing on a first server operated by a buy button provider, a selection of an item from an item inventory and a representation of the item, wherein the item is to be listed for sale on the electronic commerce store; and”

74. The combination of Dierks and Johnston renders obvious element 1[a].

(1) “an embeddable buy button builder executing on a first server operated by a buy button provider”

75. Dierks discloses “an embeddable buy button builder executing on a first server operated by a buy button provider.”

76. Dierks discloses a **payment server 204** (“a first server”) that receives information from a **seller system 202**. Exhibit 1004, Fig. 2, ¶ 53 (“The **seller system 202** may create a web page and, as shown by Arrow A, register the seller with a **payment server 204** such as payment server 114⁴.”).

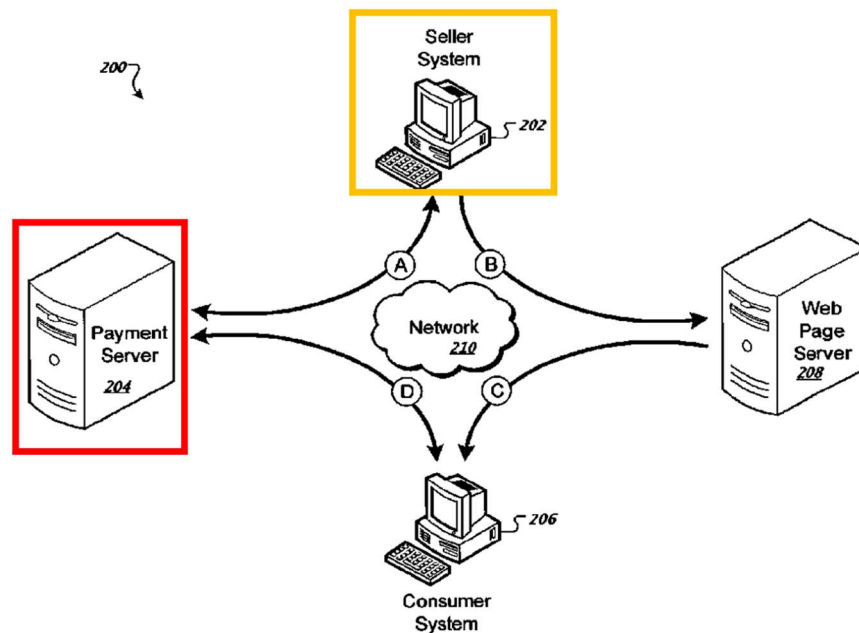


Exhibit 1004, FIG. 2 (annotated)

⁴ As I discuss in Section VIII.A.1, disclosure regarding the payment server 114 applies to the payment processing system 308.

77. Figure 3 of Dierks, below, discloses additional details of the payment server 204 with reference to the **payment processing system 308**. Like the payment server 204, Dierks discloses that the payment processing system 308 comprises a “server.” *Id.*, ¶ 62 (“The payment processing system 308 communicates with the outside world using an interface 310, which may take the form of **one** or more **web servers** and other such hardware, or may take other appropriate forms.”). *Id.*, ¶ 62. A POSITA would have understood that Dierks’s reference to “**one** or more **web servers**” is pertaining to the payment processing system 308 in general and not the “interface” specifically. A network interface is a point where pieces of equipment or protocol layers connect, which may be in the form of a Network Interface Card. Accordingly, a POSITA would have been taught that the payment processing system 308 of Dierks comprises at least one server (“a first server”) that includes the **HTML Generator 312** and **Transaction Module 314**, shown below.

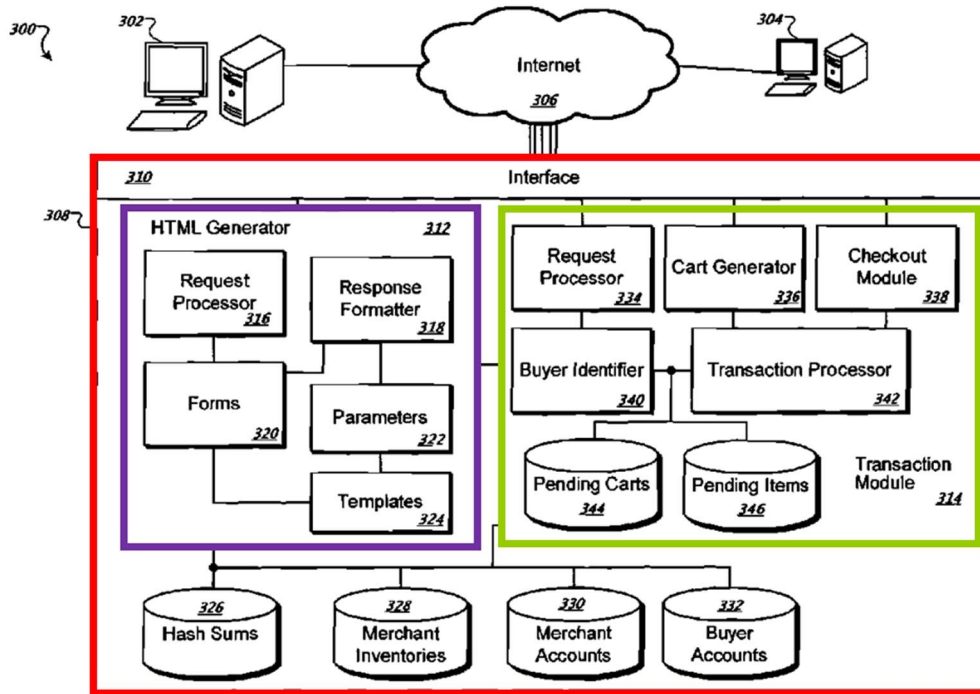


Exhibit 1004, FIG. 3 (annotated)

78. The **HTML Generator 312** discloses the claimed “*embeddable buy button builder*.” As I discuss below in more detail for element 1[b], the HTML Generator 312 generates an “add to cart” button (“*embeddable buy button*”) and therefore discloses the claimed embeddable buy button builder. The HTML Generator 312 is “*executing on*” the server, as claimed, because the payment processing system 308 includes the HTML Generator 312.

79. The server on which the **HTML Generator 312** (“*the embeddable buy button builder*”) executes is operated by the **payment processing system 308** (“*a buy button provider*.”). As I discuss in Section VII.B, applicant equated a “provider” to a “multimedia content distribution system 200” where the provider “executes both

the embeddable buy button builder 204, as well as a transaction payment processing service 214.” Exhibit 1009, 171. As I discuss herein, the payment processing system 308 of Dierks provides these same button-building and transaction payment processing services.

80. Accordingly, Dierks discloses an **HTML Generator** (“*embeddable buy button builder*”) executing on a first server operated by the payment processing system 308 (“*buy button provider*”).

(2) “receiving, by the embeddable buy button builder...a selection of an item from an item inventory”

81. Dierks discloses “receiving, by the embeddable buy button builder a selection of an item from an item inventory.”

82. Figure 1A of Dierks discloses that the vendor sells multiple items, i.e., pieces of jewelry. Exhibit 1004, ¶ 35. (“Display 102 shows a general page from a **vendor’s web site**. The display 102, for a vendor called Garnish Jewelry, shows a **number of** high-quality, handcrafted **jewelry pieces** (see FIG. 1A.”).

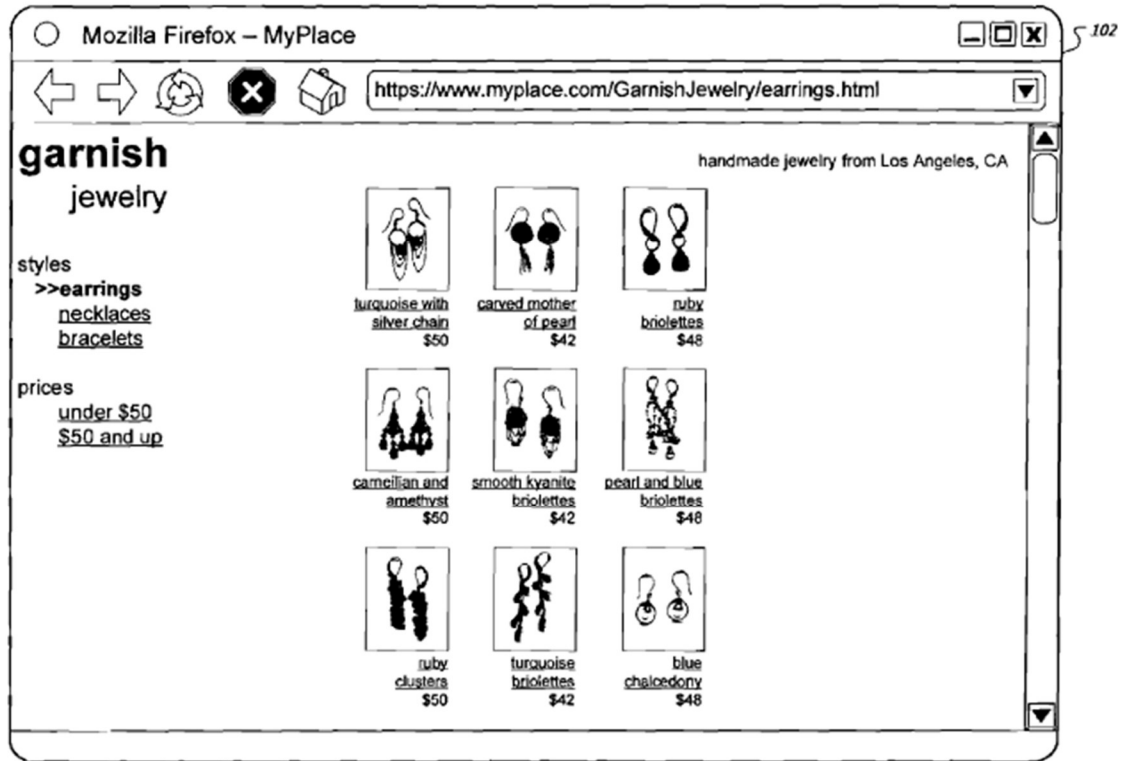


Exhibit 1004, FIG. 1A

83. Dierks further discloses that the vendor has an “*item inventory*,” stored in a database, that includes names and descriptions of items to be sold. *Id.*, ¶ 53 (“[T]he seller may optionally also open and populate an **inventory database** for goods to be offered for sale by the seller. The seller in particular may identify **names and descriptions for items to be sold** (which may include goods or services), along with prices for the items.”). Dierks discloses that a “Google Base” database can be used for storing the inventory. *Id.*, ¶ 41 (“[T]he **vendor stores** information relating

to the **vendor's inventory** using a system associated with payment server 114⁵, such as GOOGLE BASE, the payment server 114 can also check such a **database** for availability of the selected goods.”); *id.*, ¶ 86 (“an **inventory of the seller** is available to the payment processing system (e.g., through Google Base).”).

84. Dierks discloses that the **payment processing system 308** communicates with a database called “**Merchant Inventories 328**,” which might be implemented using Google Base. Exhibit 1004, Fig. 3.

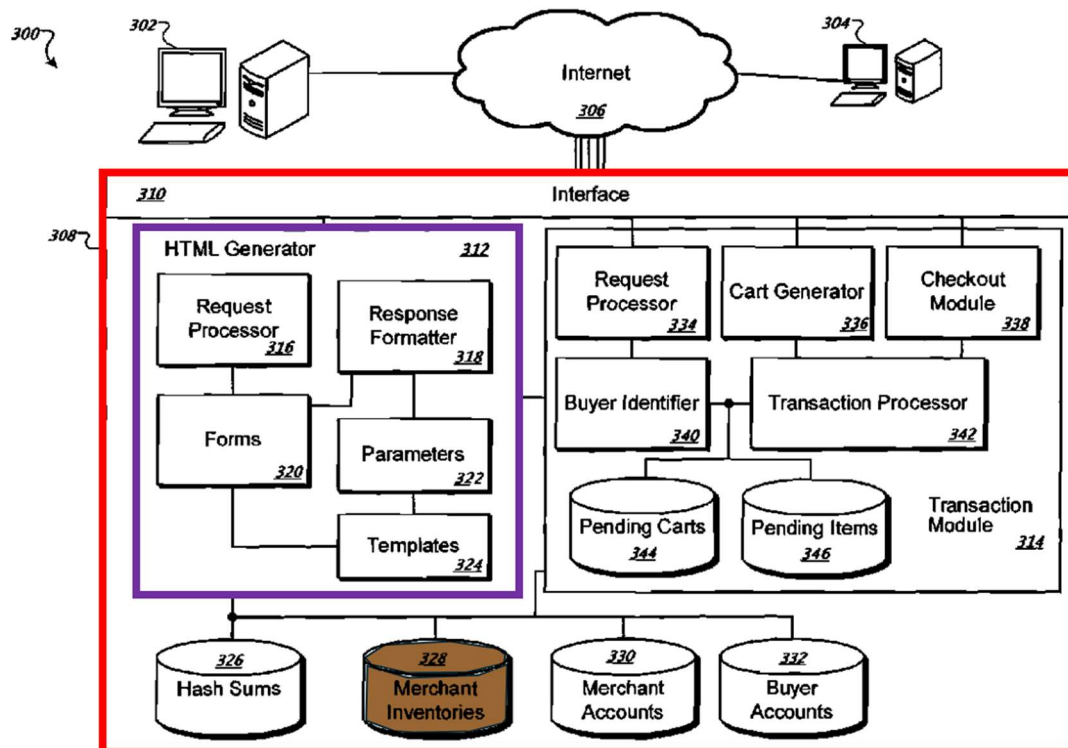


Exhibit 1004, FIG. 3 (annotated)

⁵ As I discuss in Section VIII.A.1, disclosure regarding the payment server 114 applies to the payment processing system 308.

85. A POSITA would have known that the Merchant Inventories 328 in Figure 3 represents a database because of the cylindrical structure that is used for 328. For example, reference numerals 326, 330, 332, 344, and 346 are shown with the same type of cylindrical structure and are each described respectively as being a “database.” Exhibit 1004, ¶ 65 (“hash sums *database* 326”); *id.*, ¶ 66 (“merchant accounts *database* 330 and buyer accounts *database* 332”); *id.*, ¶ 67 (“pending carts *database* 344”); *id.*, ¶ 68 (“pending items *database* 346”); *see also id.*, ¶ 41 (“The payment server 114⁶ may also place a ‘hold’ flag on the goods in the *inventory database*, or conduct others steps, so that another purchaser does not buy the last of the inventory while the first purchaser is finishing his or her shopping.”).

86. A POSITA would have understood that the Merchant Inventories database 328 could have been implemented with Google Base, because they are both disclosed as being inventory databases for vendors. Accordingly, Dierks discloses that the payment processing system 308 has access to an “*item inventory*” stored in database 328 or another database implemented with Google Base.

87. Dierks discloses that the HTML Generator 312 receives “*a selection of an item*” from the item inventory. In particular, Dierks discloses that the HTML

⁶ As I discuss in Section VIII.A.1, disclosure regarding the payment server 114 applies to the payment processing system 308.

generator “is configured to **receive parameters** regarding items to be sold by vendors and to generate code.” Exhibit 1004, ¶ 62; *see also id.*, ¶ 63 (“The **HTML generator 312** includes a number of structures for **receiving from web developers parameters** for web shopping controls and for generating HTML or other code that will produce such controls for the web developer. Request processor 316 receives communications from web developers through interface 310, and makes determinations about the actions to be taken on the requests. For example, request processor may receive an HTTP request to obtain mark-up language code for a web page, and may access forms module 320 to serve a form to the requester. The form may include areas in which the requester can provide information about an item the requester hopes to offer for sale, such as a description and price for the item. The **requester may enter such parameters** into the delivered form, and may submit the parameters back to payment processing system 308.”); ¶ 64 (“HTML generator 312 may then store the parameters in parameters module 322, and may access generic HTML code from templates module 324.”). These parameters include the name of the item. *Id.* ¶ 73 (“Text entry box 408 receives entry of a **product name**. ...”).

88. The seller provides the claimed selection of the item when entering the **item name**, as shown below in Figure 4, and pressing the “**create button code**” button 414. *Id.*, ¶ 74 (“A **create button code** button 414 causes an HTTP request to be sent to a remote server, with the information provided in the display 400 being

sent with the request. The remote server may then use the submitted information to generate code for managing a shopping cart, along with code for **generating an add to cart button.**"); *id.*, ¶ 73 ("Text entry box 408 receives entry of a **product name.** ..."); *id.*, ¶ 71 ("FIG. 4 shows an example screen shot for **obtaining web code for a purchasing button.** In the screen shot, a display 400 showing a browser interface presents a user with an easy, step-by-step process for **entering relevant information** and obtaining the code."). Thus, the seller "selects" an item from the item inventory when entering an item name that appears on the inventory list, and that selection is received by the HTML generator 312 (*"the embeddable buy button builder"*).

Mozilla Firefox - MyCheckout

https://www.mycheckout.com/Merchant/8675309/CreateItemCode/

1. Choose a button type.

☒ Basic Button

☐ Button with Options

402 Add to Cart

404 Select this button type to offer different item options, such as sizes or colors.

2. Choose a button style.

My web page has a White 406 background.

3. Describe your item.

Item Name: turquoise earrings 408

Price: 50.00 410 USD

Item Description: A single turquoise briolette is surrounded by three draping layers of delicate sterling silver chain. 412

414 Create Button Code

Insert this code for shopping cart:

```
<script type="text/javascript" src="https://www.mycheckout.com/shoppingcart.js?sellerId=8675309"></script><span id="mycheckout_status_bar"></span>
```

416

Insert this code for add to cart button:

```
<form action="https://www.mycheckout.com/Merchant/8675309/add" onsubmit="return false;" method="post"><input type="hidden" name="item" value="PD94bVWgdmVyc2lvbj0iMS4wliBlbmNvZGlhZz0iVVRGLTgIPz4NCjxJdGVtIHhtbG5zPSJodHRwOi8vY2hlY2tvdXQuZ29vZ2x1LnNvbS9zY2h1bWEvMmItDQogIDx0dWFudG10eT4xPC9xdWFudG10eT4NCiAgPHVuaXQ
```

418

Exhibit 1004, FIG. 4 (annotated)

89. Dierks teaches that the item name entered in text box 408 (“**turquoise earrings**”) corresponds to a preexisting item name in either the “Google Base” inventory, as I describe above, or the Merchant Inventories 328 shown in Figure 3. This is because Dierks discloses that the seller may populate an inventory database for goods “**to be offered for sale**” (i.e., *not for sale yet*) during a registration process when opening an account. *Id.*, ¶ 53 (“The seller system 202 may ... register the seller with a payment server 204 such as payment server 114⁷. The seller may identify himself or herself, and may **open an account** into which payments from buyers may be placed. In addition, **the seller may** optionally also **open and populate an inventory database** for goods **to be offered for sale** by the seller.”); *see also id.*, ¶ 54. Since Dierks uses the future tense of “to be offered for sale,” Dierks teaches that the inventory list exists *before* the vendor begins the process of entering the **item name** in text box 408. The vendor *selects* an item that is listed on the item inventory because the inventory contains the items that the user has for sale.

90. Moreover, Dierks discloses monitoring and updating services that require use of the item inventory. For example, the payment processing system 308 checks the availability of items using the inventory to confirm that items are

⁷ As I discuss in Section VIII.A.1, disclosure regarding the payment servers 204 and 114 applies to the payment processing system 308.

available. *Id.*, ¶ 41 (“In addition, where the **vendor stores** information relating to the **vendor’s inventory** using a system associated with payment server 114, such as GOOGLE BASE, the payment server 114 can also **check such a database for availability** of the selected goods. The payment server 114 may also place a “hold” flag on the goods in the inventory database, or conduct others steps, so that another purchaser does not buy the last of the inventory while the first purchaser is finishing his or her shopping.”). Dierks also discloses that the number of items in the inventory is decreased after an item is purchased. *Id.*, ¶ 86. (“In addition, where an *inventory of the seller* is available to *the payment processing system* (e.g., through Google Base), the *inventory may be decreased* for each of the *purchased items*”).

91. A vendor needs to select an item from the inventory to avail itself of the monitoring and updating services. If the vendor enters an **item name** that is *not from the item inventory*, the payment processing system 308 would not be able to check the availability of that item using the inventory nor update the inventory when that item is sold.

92. Accordingly, Dierks teaches receiving, by the HTML Generator 312 (“*an embeddable buy button builder*”) a selection of an item from the Merchant Inventories database 328 and/or Google Base (i.e., “*an item inventory*”).

(3) “receiving, by an embeddable buy button builder ... a representation of the item, wherein the item is to be listed for sale on the electronic commerce store”

93. Dierks combined with Johnston teaches “receiving, by an embeddable buy button builder ... a representation of the item, wherein the item is to be listed for sale on the electronic commerce store.”

94. Dierks does not explicitly disclose that the payment processing system 308 (“*the embedded buy button builder*”) receives a “selection of ... a representation of the item.” However, in view of Johnston’s teachings, a POSITA would have found this obvious. As shown below in Figure 1B of Dierks, the vendor’s web page (part of “*the electronic commerce store*”) includes the **item name** (e.g., “turquoise with silver chain”), an **item description** (e.g., “single turquoise briolette is surrounded by three draping layers of delicate sterling silver chain”), a **price** (i.e., \$42), and an **image of the product for sale**. Exhibit 1004, ¶ 36.

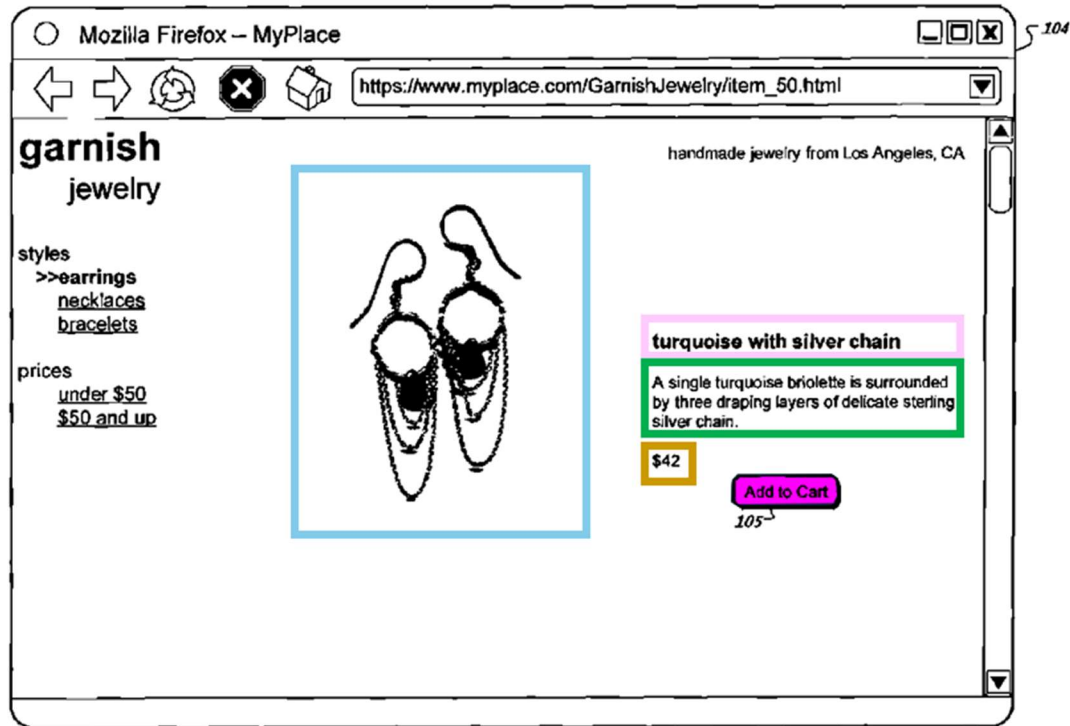


FIG. 1B

Exhibit 1004, FIG. 1B (annotated)

95. Dierks teaches that the **item name** is selected, as I describe above, and discloses that the **item description** and the **price** are entered through the screen shown below in FIG 4. *Id.*, ¶ 73 (“Various **text entry areas** 408-412 permit the user to **enter parameters** for the product to be sold.”).

Mozilla Firefox - MyCheckout

https://www.mycheckout.com/Merchant/8675309/CreateItemCode/

1. Choose a button type.

☒ Basic Button 402 ☐ Button with Options 404

Button with Options: Select this button type to offer different item options, such as sizes or colors.

\$21.00 - T-Shirt (small)

2. Choose a button style.

My web page has a 406 background.

3. Describe your item.

Item Name: 408 Price: 410

Item Description: 412

414

Insert this code for shopping cart:

```
<script type="text/javascript" src="https://www.mycheckout.com/shoppingcart.js?sellerId=8675309"></script><span id="mycheckout_status_bar"></span>
```

Insert this code for add to cart button:

```
<form action="https://www.mycheckout.com/Merchant/8675309/add" onsubmit="return false;" method="post"><input type="hidden" name="item" value="PD94bWwgdmlVyc2lvbj0iMS4wliBibmNvZGluZz0iVVRGLTglPz4NCjxJdGVtIHhtbG5zPSJodHRwOi8vY2hiY2tvdXQuZ29vZ2xiLmNvbS9zY2hibWEvMil+DQogIDx0dWFudGluT4xPC9xdWFudGluT4NCiAgPHVuaXQ
```

Exhibit 1004, FIG. 4 (annotated)

96. The payment processing system 308 then generates HTML that is used to display the item name, description, and cost—of the item to be listed for sale on the electronic commerce site—based on the selected parameters. *Id.*, ¶ 15 (“The system also includes an HTML generator configured to receive parameters relating to **an item offered for sale** and to **generate HTML code**.”).

97. Dierks is silent regarding how the **image of the item**, shown above in Figure 1B, is provided. A POSITA would have found it obvious for the image in Dierks to be provided to the payment processing system 308 by the vendor. The

image would be received in a manner similar to how the above-noted parameters are received through the screen of Figure 4. Doing so would save vendors the trouble of inserting images into web pages themselves or spending time to have someone else insert the images, which aligns with stated goals of Dierks, e.g., to assist users that have “limited computer skills” and limited “time.” *Id.*, ¶ 35 (“In the example, the **vendor** is an individual who **works full-time at another job** and is very intelligent but **has limited computer skills**. She enjoys crafting jewelry and sharing her work with others, but she **does not have the time to learn** how to set up a full-blown e-commerce site, and she certainly does not have the money to do so.”).

98. Thus, a POSITA would have found it obvious to configure the features in Figure 4 of Dierks to include a prompt telling the vendor to upload the image. Doing so was well-known before the priority date of the ’785 patent, as taught by Johnston and as I discuss next.

(4) Combination of Dierks and Johnston

99. The combination of Dierks and Johnston renders obvious the “receiving, by embedded buy button builder” a “selection of...a representation of the item,” as recited in element 1[a].

100. **First**, Johnston discloses and illustrates in Figures 3A–3G, a seller “creating an OWJO Storefront via a user interface of the OWJO platform.” Exhibit 1005, ¶ 225. For example, in Figure 3B (reproduced below), the user interface

prompts the seller to enter information associated with an item to be listed for sale.

Id., ¶ 226 (“seller can create a music track by specifying 1) the name of the track, 2) the name of the artist, 3) one or more tags in the form of a keyword or term associated with the music track, 4) the music genre, 5) a preview of the music track, 6) the actual music track that is for sale, 7) a thumbnail image that can be displayed in the OWJO Storefront and 8) the price that the user would like to sell his or her music track for.”). The item for sale in Figure 3B is music. The seller can do the same for other items for sale, such as videos, images, podcasts, and vodcasts. *See id.*, ¶¶ 227–230, Figs. 3C–3F.

The screenshot shows a web interface for OWJO, titled 'Freedom for your content'. The top navigation bar includes links for 'Welcome CPA (Log out...)', 'Home', 'About us', and 'Site map'. Below this is a secondary navigation bar with buttons for 'Upload', 'Sales', 'Purchases', 'Top Up', 'Summary', 'Publish', 'Settings', and 'Admin'. The main content area is titled 'Upload Content' and has tabs for 'music', 'videos', 'images', 'podcasts', and 'votcasts'. The 'music' tab is selected. The 'Music file Settings' section contains the following fields: 'Name of track:', 'Artist:', 'Tags:', 'Genre:' (with a dropdown menu showing 'Pop'), and 'Preview track:'. The 'Preview track' section has a text box with instructions: 'Please upload a sample of the music track. This will be available to everyone using the OWJO Widget. The file must be in one of the following formats (MP3...) and must not exceed 10MB in size.' Below this is a 'Browse...' button. The 'Music file:' section has a text box with instructions: 'Please upload music file. This will be only be made available when it is purchased. The file must be in one of the following formats (MP3...) and must not exceed 10MB in size.' Below this is another 'Browse...' button. The 'Thumbnail' section has a text box with instructions: 'Please provide a thumbnail file that we will display on the OWJO Widget.' Below this is a 'Browse...' button. At the bottom, there is a 'Set your price' section with a dropdown menu showing 'EUR' and a text box with '0'. An 'upload' button is located at the bottom right of the form.

Exhibit 1005, FIG. 3B (annotated)

101. The interface of Figure 3B allows the seller to select an image that represents the item for sale (“*a representation of the item*”). As shown above, the interface prompts the seller to “**provide a thumbnail file that we will display on the OWJO Widget**” by using a “browse” button. *Id.*, Fig. 3B.

102. Johnston discloses that the image is displayed on the Storefront as a “*representation of the item*” being sold. *Id.*, ¶ 226 (“Using a web browser ... a seller can create a music track by **specifying** ... a thumbnail **image** that can be **displayed**

in the OWJO Storefront.”); *id.*, ¶ 222 (“The OWJO Client/OWJO Dashboard can display thumbnails representing the content that buyer A has rights to access or has otherwise purchased.”); *id.*, ¶ 289 (“content may be displayed with thumbnails, links or some other type or form of representation of the content.”).

103. Figures 4A and 4B of Johnston show the user’s Storefront with the representative image. For example, the Storefront in Figure 4B shows an **image that represents** the song for sale. *Id.*, Fig. 4B, ¶ 68 (“FIG. 4B is an embodiment of the user interface of a Storefront to preview content.”); *cf.* Exhibit 1001, 13:6–9 (“In one embodiment of a web widget used for distribution of a musical catalogue, the multimedia display files include visual representations of the **cover art** used on each **album** in the **musical catalogue**.”).

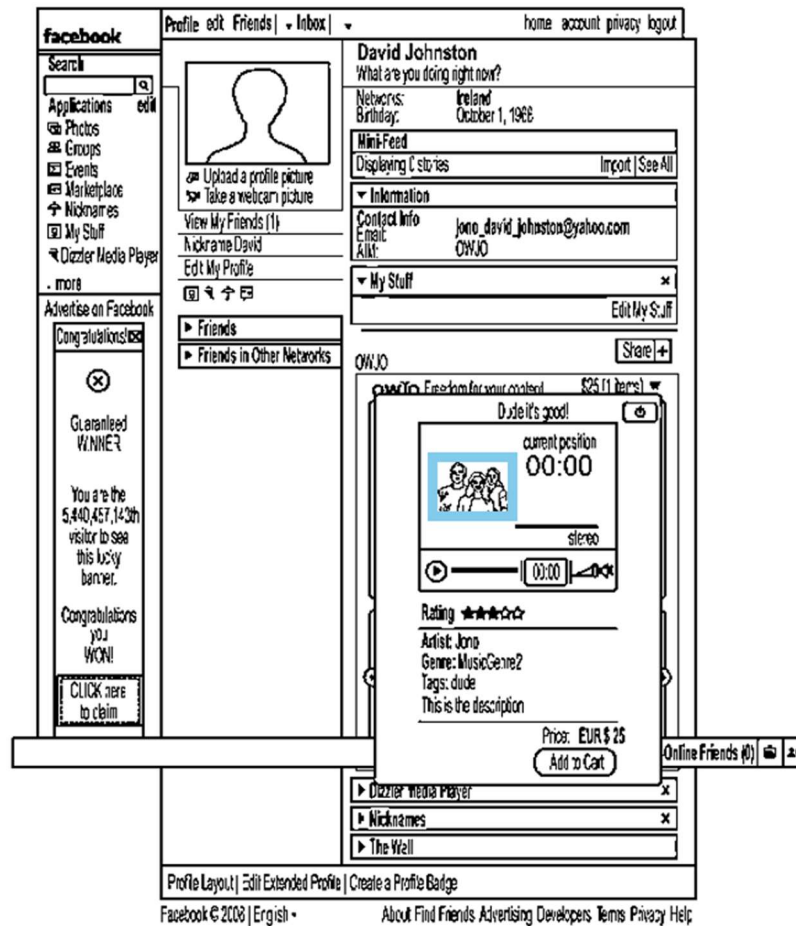


Exhibit 1005, FIG. 4B (annotated)

104. Johnston discloses that the seller selects the image by loading the image from the user's computer or from anywhere on the internet. Exhibit 1005, ¶ 226 (“The music track preview, actual music track and **thumbnail** can either be **uploaded from the machine** that is executing the web browser they are using or **loaded from a Uniform Resource Identifier** anywhere on the internet.”); *see also id.*, ¶¶ 227–230, 272.

105. Johnston discloses that the image is “received” by the system, as recited in element 1[a], and discloses that the system stores the image. For example,

Johnston discloses a storage 106 that “is a set of one or more storage services or locations **to store and retrieve content by the platform 110**. ... An artist or seller may provide and/or maintain his/her own storage 106.” *Id.*, ¶ 118. Johnston also discloses a content acquisition service 135 that is “constructed to ingest, **receive**, register or obtain **any type and form of predetermined formats or content types, such any format used for any type and form of video, audio, documents, music, movies, etc.**” *Id.*, ¶ 204. Johnston also discloses a storage module that “may manage any type and form of storage and may include storage internal to the platform 110 or external to the platform 110.” *Id.*, ¶ 178. A POSITA would have understood that the image uploaded by the user would be stored in at least one of the storage areas disclosed in Johnston. Regardless of where the image is stored, a “*selection*” of the image is received by Johnston’s system so that the image can be used for the Storefront.

(5) Motivation to Combine Dierks and Johnston

106. A POSITA would have been motivated, according to and informed by Dierks’s and Johnston’s teachings, to combine Dierks and Johnston so that the vendor in Dierks can upload an image to the payment processing system 308 for use on the vendor’s web page to represent the item for sale.

107. **First**, Dierks and Johnston are from the same field of endeavor. Both references are directed to electronic commerce. Dierks discloses a “computer-

implemented method [for] generating on a user terminal a **merchant web site** associated with a merchant with code for a merchant web page.” Exhibit 1004, Abstract; *see also id.*, ¶ 21 (“The computers include a seller system 202, such as that operated by a vendor who sets up and operates an **on-line ecommerce web site.**”). Similarly, Johnston discloses “a platform for the delivery, management and payment of **content between sellers and buyers.** In particular, the present application relates to systems and methods of self-service by a seller to **establish a store front** at one or more hosted services.” Exhibit 1005, ¶ 2, *see also id.* ¶ 21 (“**Sell on any website** (e.g., social networking site)–The Storefront is a widget or component, such as a Macromedia Flash component, that can be embedded on any website or on the artist's own social networking profile.”). Thus, both references are directed to electronic commerce.

108. Johnston, similar to Dierks, also acknowledges the investment needed to “create a website that has full ecommerce facilities” and offers a simple solution to people that want to sell items online. *Id.*, ¶ 4 (“If an artist wants to generate income from their creative content they would have to 1) **create a website that has full ecommerce facilities** and 2) promote their site by advertising and posting teasers or previews on media sharing sites such as YouTube or Flickr with the hope of encouraging visitors to their website to purchase content. **This is a significant investment** and would **deter most content creators.**”).

109. Johnston, similar to Dierks, provides a “self-service” way for a seller to establish an e-commerce store front. *Id.*, ¶ 2 (“[T]he present application relates to systems and methods **of self-service by a seller to establish a store front** at one or more hosted services.”). Johnston does this by allowing a user to simply provide the system with information about an item for sale, including an image representation of the item, so that the system handles the work of preparing the e-commerce store.

110. Thus, both references are directed to electronic commerce and to assisting users create web sites used for electronic commerce. Accordingly, a POSITA would have understood that Johnston’s teachings were directly applicable to Dierks’s system.

111. *Second*, a POSITA would have combined the elements taught by Dierks and Johnston—namely, a system that provides a product listing that is generated based on information provided from the user, as taught by Dierks, and the function of allowing a user to select an image representation of the product, as taught by Johnston—using known and routine computer programming principles that allow the payment processing system 308 in Dierks to receive the image and generate HTML used to display the image on the vendor’s web site. While Dierks discloses a “thumbnail” image, a POSITA would have understood that a larger file size image could be used, if the image needs to be a particular size or resolution. A POSITA would have understood that a “thumbnail” image is simply a scaled-down version

of the original image. This smaller image size is typically chosen so that it fits appropriately within the user interface when an image is not needed to occupy an entire screen, and it has the advantage of allowing the browser to display its page more quickly (since less image data needs to be downloaded.) The system of Dierks would store the image for the vendor, as I discuss above for Johnston (*Id.*, ¶¶ 118, 178, 204), and generate HTML that embeds the image along with the HTML generated for the item name, description, and price, as I discuss below for elements 1[b], 1[c]. The image could be embedded with known techniques taught by Dierks, such as by using the “src” attribute containing a URL pointing to the image to embed. Exhibit 1004, ¶ 38 (“src=‘https://checkout.google.com/**img**/addtoCart.gif’ alt=‘Add to Cart’ type=‘**image**’”); *id.*, ¶ 43 (“src=‘https://checkout.google.com/**img**/gcb2.gif’ alt= ‘checkout’ type=‘**image**’ id= ‘gcb_img’”). When a user’s browser parses the HTML from the vendor’s web site, the image would be fetched and displayed. This result would have been readily predictable and recognized by a POSITA.

112. **Third**, configuring Dierks based on Johnston’s teachings is simply an application of a known technique (uploading an image for inclusion on a web site) to a known system ready for improvement (Dierks’s system that displays an image but does not disclose where the image comes from) with predictable results. Application of this known technique would have been readily achievable by a POSITA via known and routine computer programming principles to implement

Johnston's image uploading and displaying features in the system in Dierks. This configuration of Dierks would have yielded the expected, predictable result of displaying an image "*representation of the item*" for sale, while not requiring sellers to possess the technical skills needed to add such an image on their own.

113. **Fourth**, a POSITA would have been motivated to combine the teachings of Dierks and Johnston because allowing a user to upload an image representation of an item for sale on an e-commerce web site would have been obvious to try. There were only two possible known and predictable choices for generating a product listing that includes an image of the item for sale—(1) requiring users to generate the code themselves for displaying images and (2) preparing the code for the users, so that the users do not have to learn how to prepare the code themselves. A POSITA could have pursued these two potential solutions with a reasonable expectation of success because the analysis would have simply involved trying each one of these two solutions. Thus, a POSITA would have found it obvious to configure Dierks by incorporating Johnston's teaching of embedding an image, representing a product, for the seller.

114. **Fifth**, the configuration of Dierks's system to include the functionality of receiving an image from a user and generating HTML that displays the image would have been readily implemented by augmenting the screen shown in Figure 4 of Dierks to prompt a user to browse for and select an image to display and having

the HTML Generator in Dierks generate HTML that displays the image. The HTML generator in Dierks already generates HTML that displays the item name, description, and price of the item, as I discuss above. Exhibit 1004, ¶¶ 63–64, 73–74. Thus, the payment processing system 308 would simply be configured to store the image, and the HTML generator would simply be configured to generate the HTML that displays the image, along with the item name, description, and price. A POSITA would have implemented this configuration using basic computer programming principles to configure Dierks’s payment processing system 308 to receive and store the image so that the generated HTML, which is used for the vendor’s web site, displays the image when the web site is accessed. Making this configuration would have been well within the knowledge and skillset of a POSITA.

(6) Reasonable Expectation of Success in Combining Dierks and Johnston

115. A POSITA would also have had a reasonable expectation of success. The combination of Dierks and Johnston represents a straightforward implementation of adding functionality to Dierks’s system that allows uploading of an image and preparing HTML that displays the image of an item for sale. Dierks already discloses an HTML Generator that generates HTML to display the name, description, and price of the item, and a POSITA would have been familiar with and able to implement the functionality to do the same for an uploaded image. The reasons a POSITA would have expected success parallel those that provide

motivation for this combination—including because both Dierks and Johnston are directed to electronic commerce, and both disclose product listings. *See, e.g.*, Sections VIII, X.A.1.b.4-5.

116. Accordingly, Dierks in view of Johnston renders obvious element 1[a] with respect to “receiving, by embedded buy button builder” a “selection of...a representation of the item.” A POSITA would have been motivated to combine the teachings of these references and would have had a reasonable expectation of success. For these same reasons, a POSITA would have been motivated to combine the teachings of these references for all subsequently discussed claims and limitations in this ground.

117. Further, Dierks discloses that the item is sold on the vendor’s web page and, therefore, discloses that “the item is to be listed for sale on the electronic commerce store,” as claimed. Exhibit 1004, ¶ 62 (“The HTML generator, in this example, is configured to receive parameters regarding *items to be sold by vendors* and to generate code, such as HTML or XHTML, for the vendors to *include in their web pages*. ...”).

118. Accordingly, Dierks and Johnston teach receiving, by the HTML Generator 312 (“*the embeddable buy button builder*”) executing on a first server operated by a buy button provider, the name of an item in inventory (“*selection of an item from an item inventory*”) and an item description (“*representation of the*

item”), wherein the item is to be listed for sale on a vendor’s web page (“*the electronic commerce store.*”).

119. Therefore, Dierks and Johnston render obvious element 1[a].

c) 1[b]: “generating, by the first server, the embeddable buy button”

120. As I discuss for element 1[a], Dierks discloses that the **payment processing system 308** (“*the first server*”) includes the **HTML Generator 312**, shown below, that generates the code for the “add to cart” button 105 (“*the embeddable buy button*”). Exhibit 1004, ¶ 66 (“With the vendor web page complete and posted, prospective purchasers may visit the page and decide to click on a control, such as an ‘**add to cart**’ control, **added by the HTML generator 312.**”).

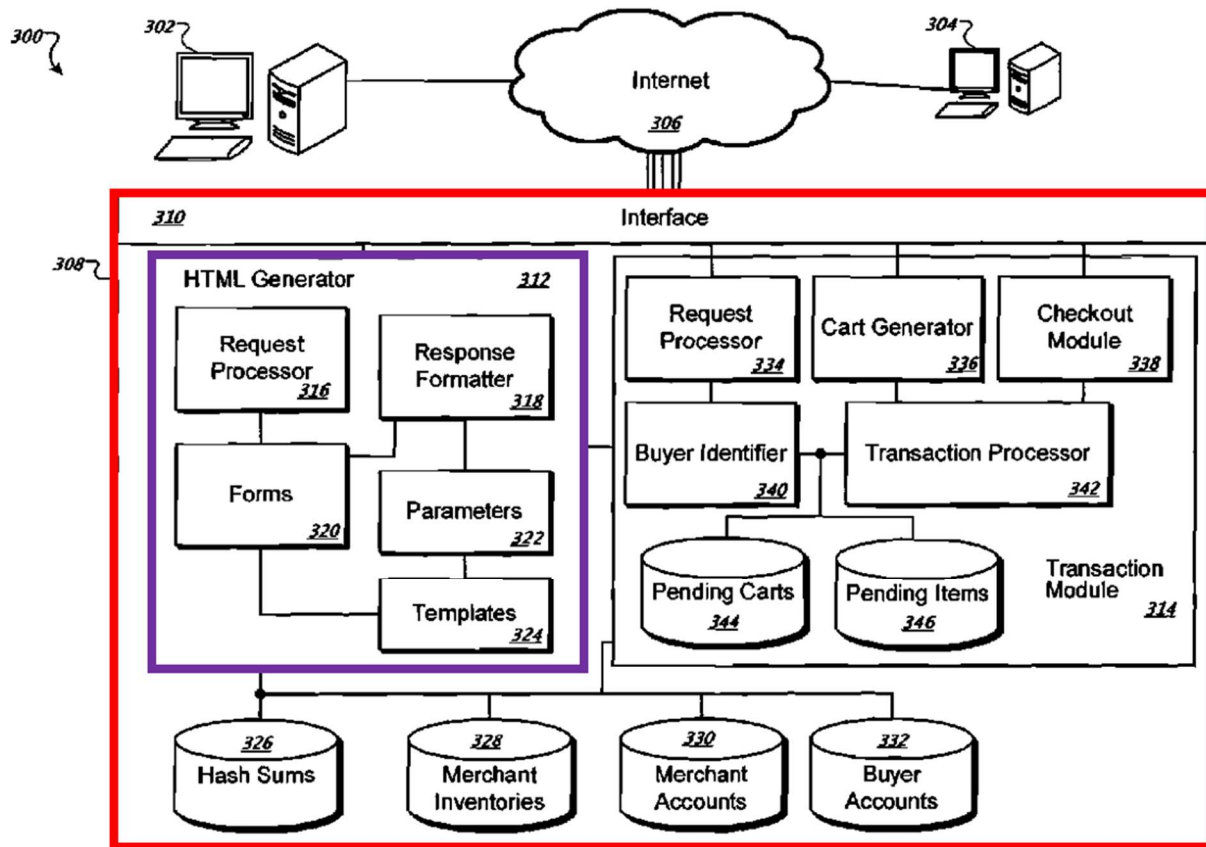


Exhibit 1004, FIG. 3 (annotated)

121. As I discuss for element 1[a], the **HTML Generator 312** receives parameters that the seller enters into the screen shown below in Figure 4.

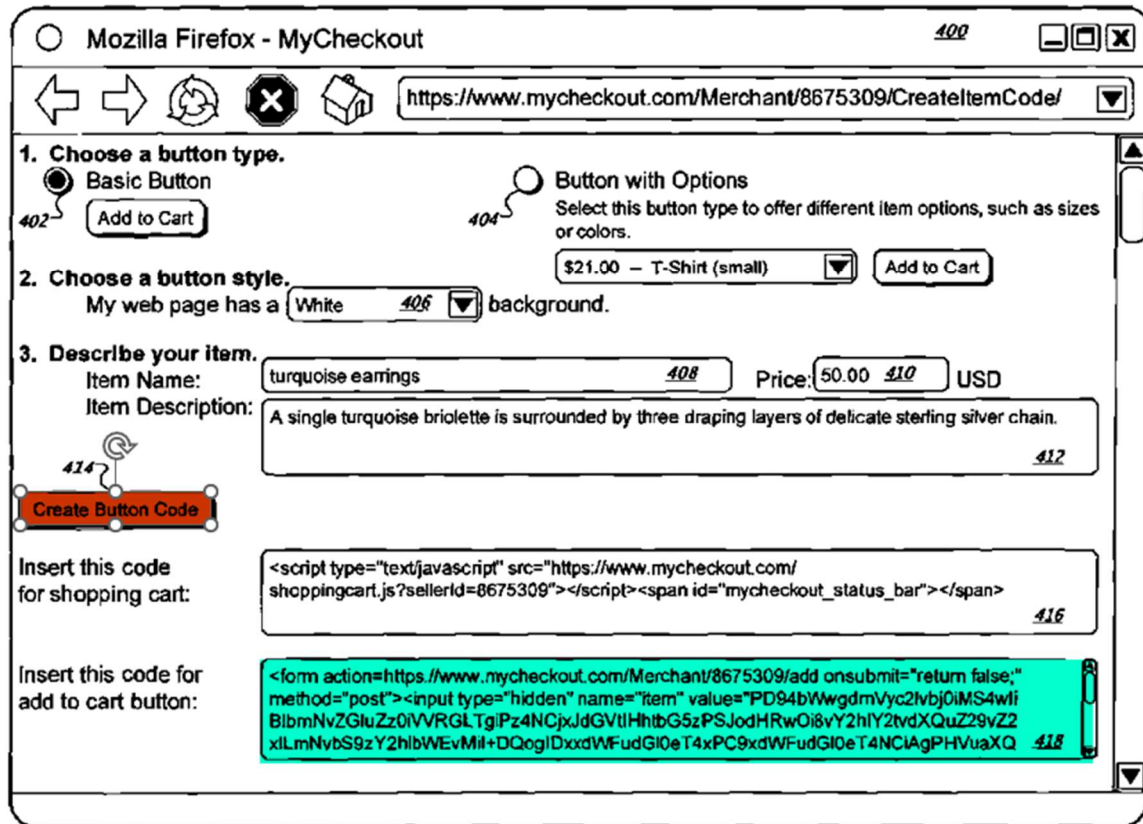


Exhibit 1004, FIG. 4 (annotated)

122. After the parameters are entered, the seller selects the “**Create Button Code**” button 414 that causes the HTML Generator 312 to generate the “add to cart” button. *Id.*, ¶ 74 (“A **create button code button 414** causes an HTTP request to be sent to a remote server, with the information provided in the display 400 being sent with the request. The remote server may then use the submitted information to generate code for managing a shopping cart, along with code for **generating an add to cart button.**”); *see also id.*, ¶ 62 (“The **HTML generator**, in this example, is configured to receive parameters regarding items to be sold by vendors and to **generate code**, such as HTML or XHTML, for the vendors to include in their web

pages, **to generate controls** for making ordering and selection of the items easier for visitors to the sites.”); Fig. 4. The code generated for the “add to cart” button is shown above in **area 418**. *Id.*, ¶ 74 (“The code for generating an add to cart button, displayed in **area 418**, may take the form of HTML code as shown in more detail above.”).

123. Accordingly, Dierks discloses “*generating*,” by the HTML Generator 312, which is part of “*the first server*,” the “add to cart” button (“*the embeddable buy button*”).

124. Therefore, Dierks discloses element 1[b].

d) 1[c] “wherein the embeddable buy button is configured to display the representation of the item, and a hyperlink;”

125. As I discuss for element 1[a], the combination of Dierks and Johnston teaches that the HTML Generator 312 (“*the embeddable buy builder*”) receives an image representation of the item.

126. The code generated for the “add to cart” button displays more than the depiction of the button 105. The code also displays the item name, description, and price, along with the depiction of the button. In particular, Dierks discloses that the parameters entered into the screen of Figure 4 are “coded along with the button.” Exhibit 1004, ¶ 73 (“Various text entry areas 408-412 permit the user to enter parameters for the product to be sold using the generated button. Text entry box 408 receives entry of a **product name**, while text entry box 410 receives a **product**

price, and text entry box 412 receives a **product description**. Such entries may be coded along with the button, so as to be displayed on the vendor's web page with the button. ...”).

127. Thus, the item name, description, and price are coded with the button, i.e., the generated code displays the depiction of the “**add to cart**” button, the item name (“**turquoise with silver chain**”), the **description**, and the **price**. These items are placed next to the **image of the item**, as shown below in Figure 1B.

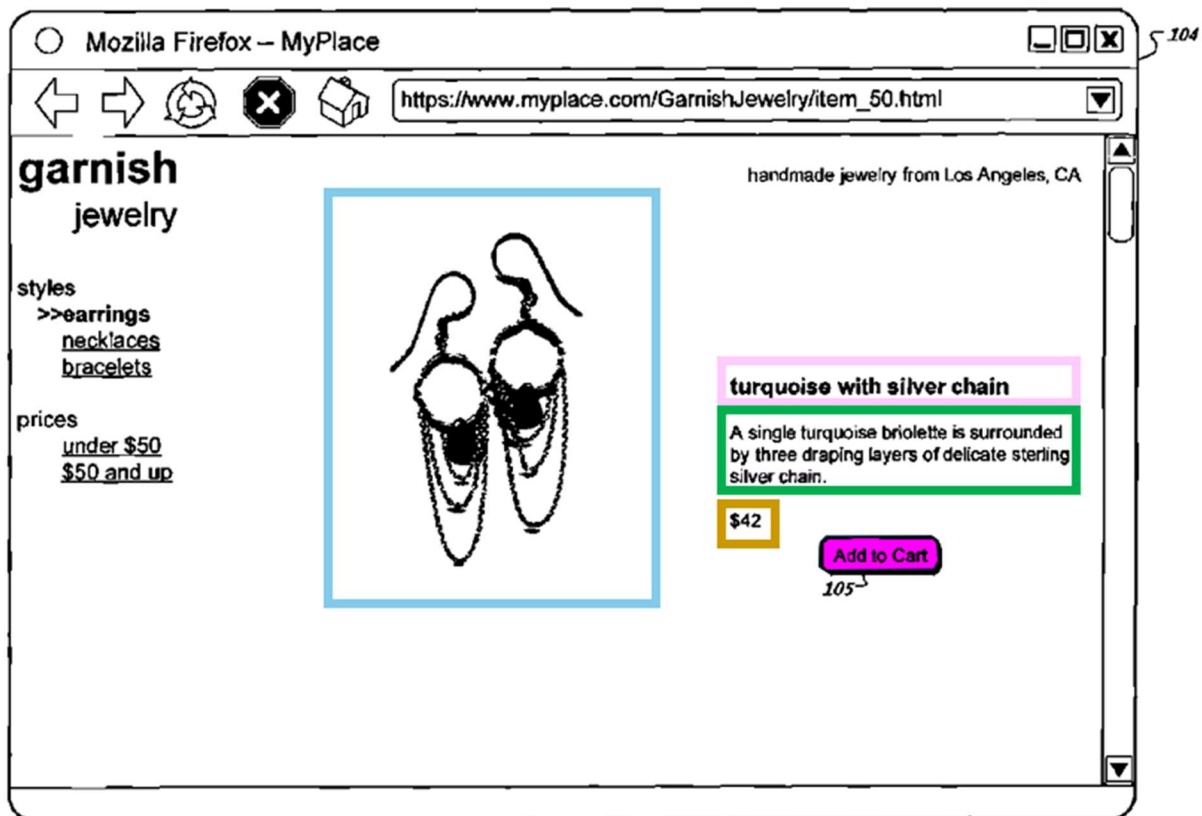


Exhibit 1004, FIG. 1B (annotated)

128. A POSITA would have understood that the code for the item name, description, and price are included with the code for the depiction of the “**add to**

cart” button because Figure 4, shown below, has two areas, 416 and 418, that include code. Area 416 is used for a “shopping cart” feature that I discuss further below. This leaves area 418 to include the code for the depiction of the button 105 and the item name, description, and price.

Mozilla Firefox - MyCheckout 400

https://www.mycheckout.com/Merchant/8675309/CreateItemCode/

1. Choose a button type.

☒ Basic Button 402 ☐ Button with Options 404

Select this button type to offer different item options, such as sizes or colors.

2. Choose a button style.

My web page has a 406 background.

3. Describe your item.

Item Name: 408 Price: 410 USD

Item Description: 412

414

Insert this code for shopping cart:

```
<script type="text/javascript" src="https://www.mycheckout.com/shoppingcart.js?sellerId=8675309"></script><span id="mycheckout_status_bar"></span>
```

 416

Insert this code for add to cart button:

```
<form action="https://www.mycheckout.com/Merchant/8675309/add" onsubmit="return false;" method="post"><input type="hidden" name="item" value="PD94bWwgdmVyc2lvbj0iMS4wIjBibmNvZGluc290IHRGLTglPz4NCjxJdGVtIHhtbG5zPSJodHRwOi8vY2h1Y2tvdXQuZ29vZ2x1LmNvbS9zY2hibWEvMil+DQogIDxldWFudGI0eT4xPC9xdWFudGI0eT4NCiAgPHVuaXQ
```

 418

Exhibit 1004, FIG. 4 (annotated)

129. When Dierks is configured in view of Johnston, to receive the uploaded image representation of the item, as I discuss for element 1[a], it would have been obvious to include the code for *displaying the image*, along with the code for *displaying the name, description, and price*. This is because (1) the image

representation of the item is another form of a product description, (2) the image is disclosed by Dierks as being positioned next to the name, description, and price of the item, as shown above in Figure 1B, and (3) Dierks discloses that the code for the “add to cart” button is already configured to display more than just the depiction of the button 105.

130. A POSITA would have been able to include the code needed to display the image using known and routine computer programming principles. Therefore, the combination of Dierks and Johnston teaches that the “add to cart” button (“*the embeddable buy button*”) is configured to display the image representation of the item.

131. Further, when the “add to cart” button 105 is pressed, it causes an HTTP request and, therefore, includes a hyperlink⁸, such that the code for the “add to cart” button is configured to display a hyperlink, i.e., display the depiction of the button that functions as a hyperlink. *Id.*, ¶ 38 (“[s]election of the **button 105 may cause an HTTP request**, in the form of an XMLHttpRequest (XHR), to be made of payment

⁸ The term “hyperlink” is not used in the ’785 patent specification outside of the claims.

server 114⁹, as indicated by Arrow A.”). A POSITA would have understood that a hyperlink can be presented in different forms, like an image, icon, text, or any type of visible element that, when clicked, sends an HTTP request. *See, e.g.*, Exhibit 1022, 5:26-28 (“The web page of the reseller includes a hyperlink, which is indicated by a graphic display, such a button, highlighted text, or the like.”); *id.*, 2:47-53 (“Instead of entering the URL, the user employs the mouse to click on a highlighted portion of text (known as “hypertext”) or graphics (together known as “hyperlinks”), to retrieve associated information. The browser interprets the hypertext formatted as hypertext markup language (HTML) and transmitted using hypertext transfer protocol (HTTP).”). In the case of Dierks, the displayed hyperlink is in the form of the “add to cart” button 105 that is visible to the user and causes the HTTP request when pressed. Accordingly, the combination of Dierks and Johnston teaches that the “add to cart” button, including its underlying code, (“*the embeddable buy button*”) is configured to display the image of the item (“*the representation of the item*”), and the depiction of the button itself that is the hyperlink.

e) 1[d]: “embedding, by the buy button provider, the embeddable buy button into a web site,

132. Dierks discloses element 1[d].

⁹ As I discuss in Section VIII.A.1, disclosure regarding the payment server 114 applies to the payment processing system 308.

133. Dierks discloses that the payment processing system 308 (“*the buy button provider*”) embeds the “add to cart” button (“*the embeddable buy button*”) into a web site. Dierks discloses that the payment processing system 308 embeds the code for the “add to cart” button into a web site using multiple techniques. For example, the payment processing system 308 can embed the code “automatically, such as by *pushing the code* to particular spots in an *open mark-up language document* on the seller’s client device, *e.g.*, at particular *pre-determined tags inserted in the page.*” Exhibit 1004, ¶ 78. Dierks also discloses that the code can be embedded by inserting “pointers” in the web page that point to the code. *Id.*, ¶ 78 (“code may also be stored remotely of the client device, and *pointers to the code may be placed in the seller’s web page.* In addition, the code incorporated into the web page may itself point to other code, such as Javascript code.”).

134. A POSITA would have understood that Dierks’s reference to “page” and “web page” means a “web site.” For example, Figure 1B of Dierks shows the following domain address, which is a web site: <https://www.myplace.com>. The specific web page within the web site that has the “**add to cart**” button is https://www.myplace.com/GarnishJewelry/item_50.html. *Id.*, Fig. 1B.

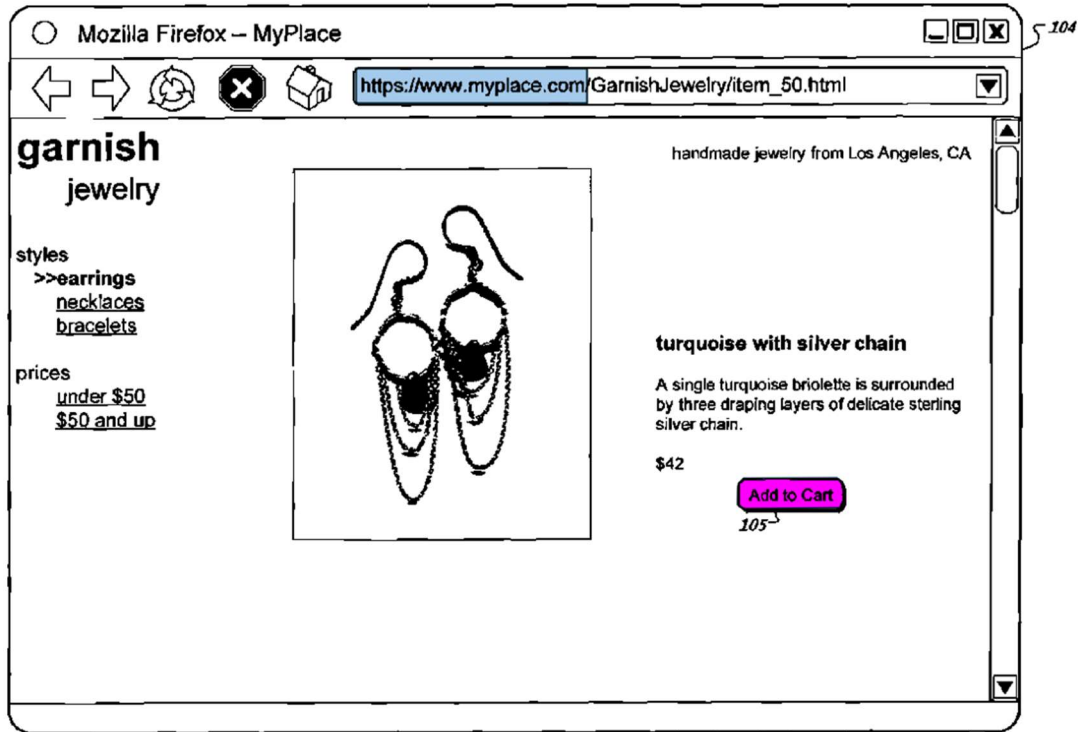


Exhibit 1004, FIG. 1B (annotated)

135. Accordingly, the payment processing system 308 (*“the buy button provider”*) embeds the code for the “add to cart” button by inserting the code, either directly or by using pointers, into the vendor’s web site.

136. Therefore, Dierks discloses element 1[d].

f) 1[e] wherein the web site is executed by a second server not operated by the buy button provider,”

137. Dierks discloses element 1[e].

138. Figure 3 of Dierks discloses **server 302** that provides access to the web site. Exhibit 1004, ¶ 61 (*“server 302 ... may represent a web server **providing access** to one or more **web sites** that are offering goods or services for sale.”*). A POSITA would have understood that these “web sites” that offer goods for sale

correspond to the web site that I discuss for element 1[d] that includes the embedded “add to cart” button.

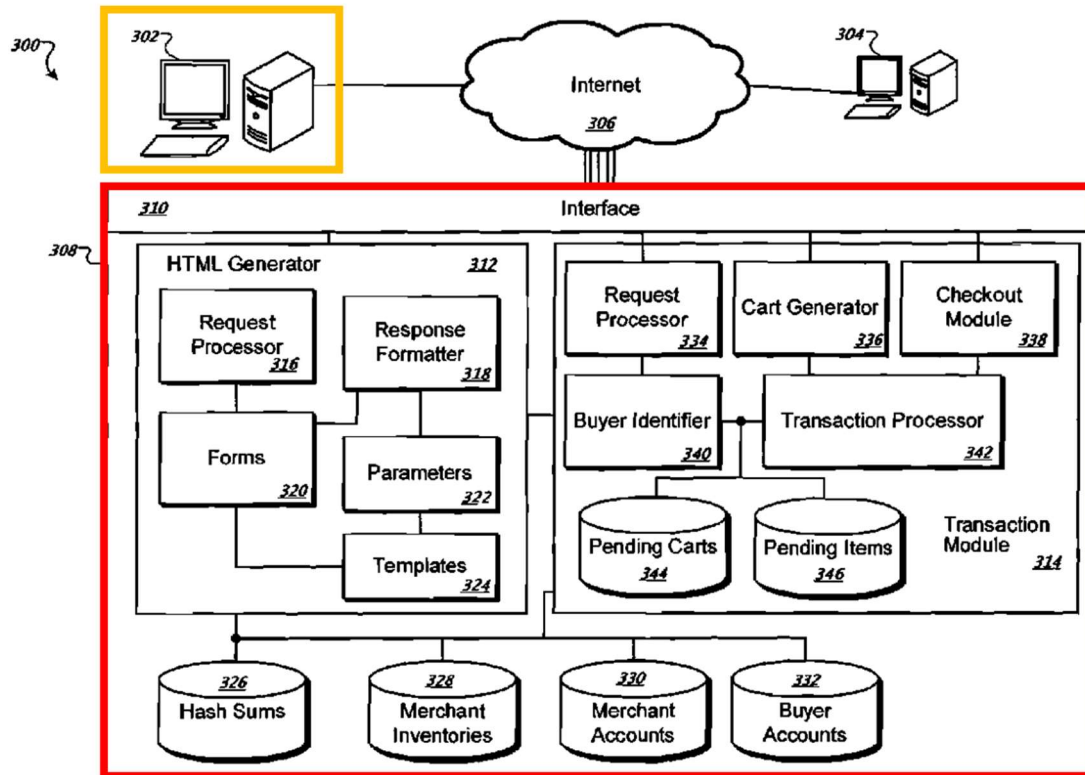


Exhibit 1004, FIG. 3 (annotated)

139. Dierks does not disclose that **server 302** is operated by the **payment processing system 308** (“*buy button provider*”). Further, a POSITA would not have expected the server 302 to be operated by the payment processing system 308 since they provide separate services and are not disclosed as being related. Accordingly, Dierks discloses that the web site is executed by the server 302 (“*a second server*”) and is *not* operated by the payment processing system 308 (“*the buy button provider*”).

140. Therefore, Dierks discloses element 1[e].

g) 1[f]: “wherein the electronic commerce store is displayed within the web site upon selection of the hyperlink,

141. Dierks Discloses element 1[f].

142. Dierks discloses a “**shopping cart**” 116 feature that forms part of the electronic commerce store because it facilitates the purchase of an item, as shown below in Figure 1C. Exhibit 1004, ¶ 9 (“the **shopping cart** information includes displaying a **shopping cart** total that includes shipping and tax charges.”); *id.*, ¶ 51 (“When the purchaser chooses to complete the purchase, their account is debited by the payment server 114¹⁰ in an amount **equal to the price of all the goods in the shopping cart** plus tax, shipping/handling, and other costs, such as a transaction charge for the payment processor.”).

¹⁰ As I discuss in Section VIII.A.1, disclosure regarding the payment server 114 applies to the payment processing system 308.

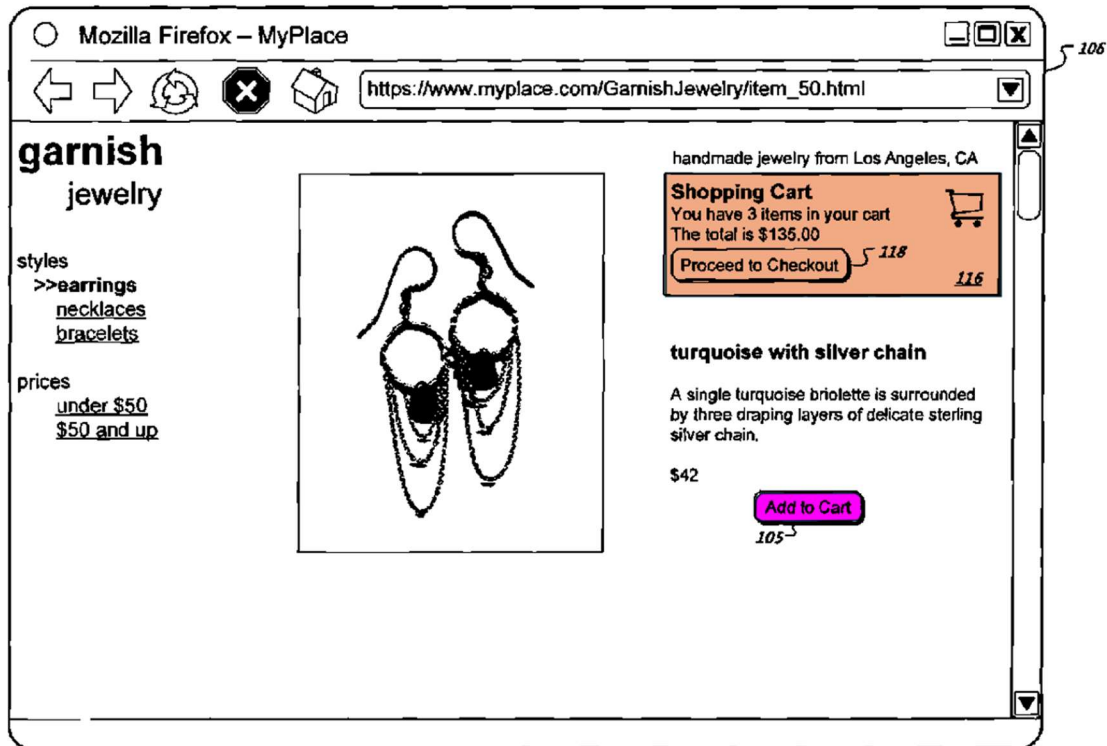


Exhibit 1004, FIG. 1C (annotated)

143. The **shopping cart 116** is displayed within the web site upon selection of the “**add to cart**” 105 button (“*the hyperlink*”). Pressing the “add to cart” button 100 activates the hyperlink, as I describe for element 1[c]. *Id.*, ¶ 43 (“Display 106 shows the result of a **user’s selection of button 105** (see FIG. 1C). Specifically, a **shopping cart object** shown as a graphical element such as an **iframe 116** has been created on display 106.”).

144. Accordingly, Dierks discloses that the shopping cart (part of “*the electronic commerce store*”) is displayed within the web site upon selection of the “add to cart” button (“*the hyperlink*”).

145. Therefore, Dierks discloses element 1[f].

h) 1[g] the electronic commerce store configured to facilitate payment processing related to a purchase of the item, and”

146. Dierks discloses element 1[g].

147. Dierks discloses a secure checkout (another part of “*the electronic commerce store*”) that facilitates payment processing related to a purchase of the item. The secure checkout is accessed when a user selects the “proceed to checkout” button 118 shown in the shopping cart 116 that I discuss for element 1[f]. Exhibit 1004, ¶ 44 (“The purchaser may also choose to **go to a checkout to pay** or otherwise edit entries in their shopping cart, by **selecting the ‘proceed to checkout’ control button 116.**”); *id.*, ¶ 48 (“At display 112, the user/purchaser has selected a ‘**proceed to checkout**’ button, and is redirected to a web page operated by the payment server 114.¹¹...”).

148. The **secure checkout** is shown below as having a **payment mechanism 124** that facilitates payment processing related to the purchase of an item.

¹¹ As I discuss in Section VIII.A.1, disclosure regarding the payment server 114 applies to the payment processing system 308.

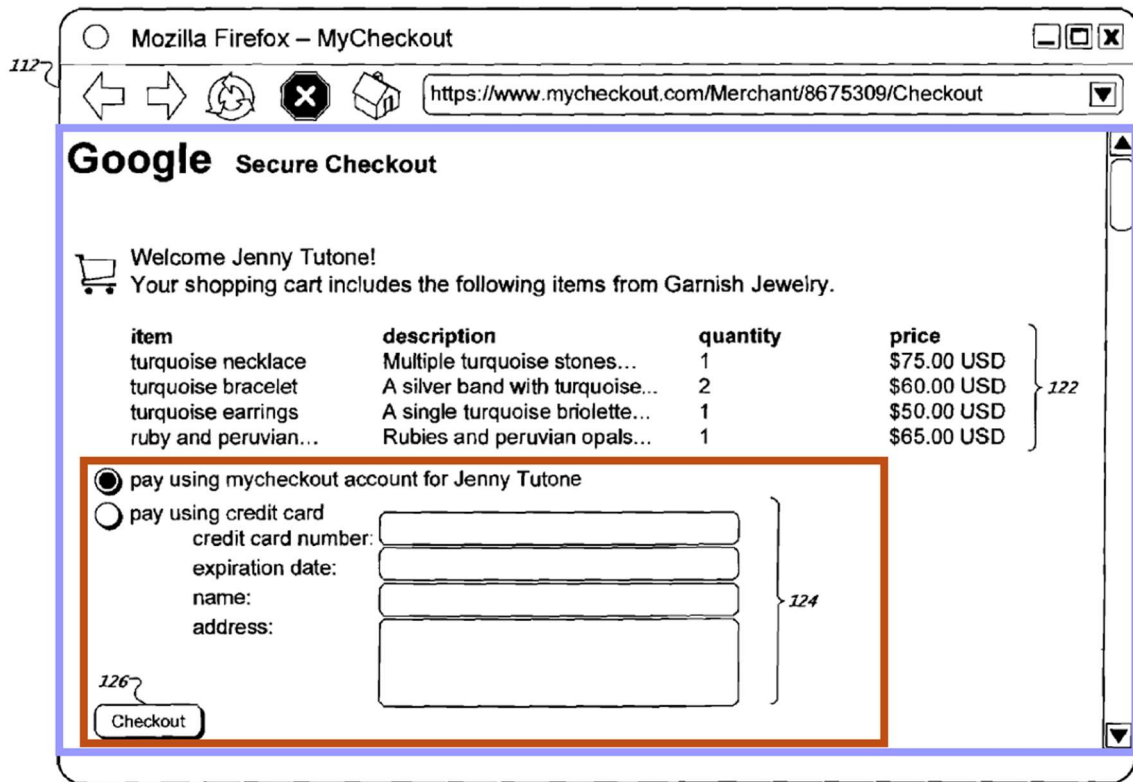


Exhibit 1004, FIG. 1E (annotated)

149. The **payment mechanism 124** facilitates payment processing by providing the user with options to pay for the item. *Id.*, ¶ 48 (“the user may be shown a list of **items** and their prices 122 that have been selected by the user....The user may also be permitted to **select a payment method**, such as through a GOOGLE CHECKOUT account or a debit or credit card or other payment mechanism 124.... Finally, the user may confirm that the displayed information is accurate and may cause a **closing of the transaction** by selecting checkout button 120.”); *id.*, ¶ 70 (“The checkout module may **manage closing out of a transaction**, such as by

computing a total cost, any transaction fees, taxes, and shipping/handling, in a conventional manner.”).

150. Dierks also discloses that the user’s account is debited for the transaction. *Id.*, ¶ 51 (“When the purchaser chooses to complete the purchase, their **account is debited** by the payment server 114¹² in an amount equal to the price of all the goods in the shopping cart plus tax, shipping/handling, and other costs, such as a transaction charge for the payment processor”).

151. While Figure 1E above shows a domain address of <https://www.mycheckout.com>, Dierks discloses that the secure checkout can be *displayed with the vendor’s web site*, similar to the shopping cart 116 (part of the “*electronic commerce store*”). *Id.*, ¶ 44 (“the **checkout** may also be **performed in the same content window** of a browser **as the vendor’s web site**, by mechanisms similar to those described above”).

152. Dierks discloses to display the checkout in the same window as the vendor’s web site “to **avoid** taking a purchaser on a ‘**round trip**’ to the payment processor’s web page and back to the vendor’s page, or opening another window or tab in a browser.” *Id.*, ¶ 34. Dierks teaches to not put the checkout in a new window

¹² As I discuss in Section VIII.A.1, disclosure regarding the payment server 114 applies to the payment processing system 308.

because the new window “would likely have a different interface than does the web site of the vendor, and *could thus be confusing to the purchaser*, or at least take away from the immersive aspect of the shopping transaction that is more readily available *when one company handles both the merchandise and payment processing*.” *Id.*, ¶ 34.

153. Therefore, Dierks discloses element 1[g].

i) 1[h]: “wherein the electronic commerce store is operated by the buy button provider.”

154. Dierks discloses that the shopping cart and secure checkout (part of “*the electronic commerce store*”), that I discuss above for elements 1[f] and 1[g], is operated by the payment processing system 308 (“*the buy button provider*”).

155. Dierks discloses that the **payment processing system 308**, shown below in Figure 3, includes a **Transaction Module 314** having a “Transaction Processor 342,” “Pending Carts [database] 344,” “Cart Generator 336,” and a “Checkout Module 338.” Exhibit 1004, Fig. 3.

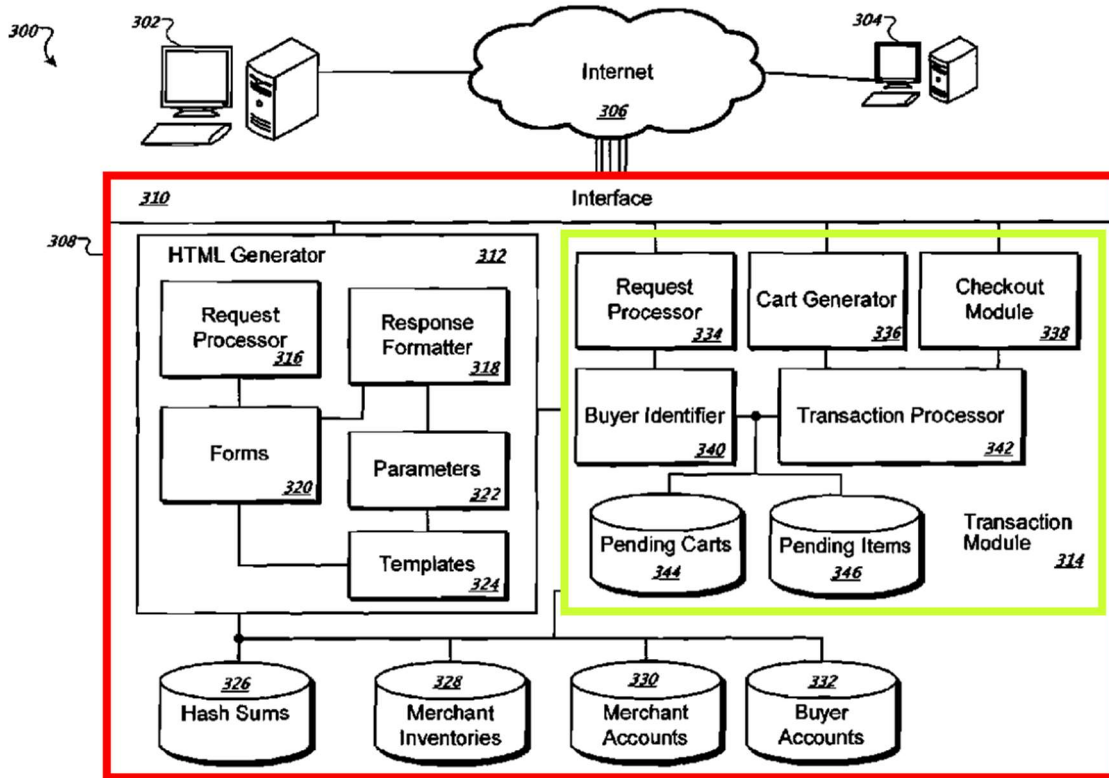


Exhibit 1004, FIG. 3 (annotated)

156. The transaction processor 342 operates the shopping cart by opening and populating the shopping cart. *Id.*, ¶ 67 (“transaction processor 342 may open a new **shopping cart** for the pair, and may **populate the cart** with information associated with the selected control.”). Also, the “pending carts database 344” stores a “pending **shopping cart** table” that “tracks the transaction” and “may include fields for a ConsumerID, a MerchantID, an AnonymousID, and a CartID.” *Id.* Further, a POSITA would have understood that the “Cart Generator 336” is related to operation of the shopping cart at least because element 336 is called a “*cart*” generator. *Id.*, Fig. 3; *see also id.*, ¶¶ 47, 49, 58, 59, 67, 68, and 74. Further, Dierks

discloses that the “cart generator is a structure that receives parameters relating to a proposed purchase and formats them for transmission to a web page of a purchaser. The cart generator 336 may, for example, receive item quantities and dollar values from transaction processor 342, and may format them into an XML message or other form for transmission to a purchaser’s browser.” *Id.*, ¶ 69.

157. Further, Dierks discloses that the “**checkout module** 338 may be accessed when a purchaser decides to finish a transaction and buy or edit items in his or her shopping cart.” *Id.*, ¶ 70. The checkout module 338 manages “**closing out of a transaction**, such as by computing a total cost, any transaction fees, taxes, and shipping/handling, in a conventional manner.” *Id.*; *see also id.*, ¶¶ 50, 51.

158. Accordingly, the payment processing system 308 (“*the buy button provider*”) in Dierks operates the shopping cart and secure checkout (part of “*the electronic commerce store*”) to disclose element 1[h].

2. Dependent Claim 2

a) “The method of claim 1, wherein the embeddable buy button is configured to be embedded into different web sites.”

159. Dierks and Johnston render obvious claim 1, as I discuss above. Dierks discloses the additional limitation of claim 2.

160. Dierks discloses that the embeddable buy button may be in the form of JavaScript. Exhibit 1004, ¶ 37 (“As explained in more detail below, the **button 105**

is generated using JavaScript (or, e.g., Jscript or VBScript) associated with the vendor's underlying web page that generates display 104.”).

161. A POSITA would have understood that JavaScript is used with an HTML document by employing a `<script>` tag at the beginning and end of the JavaScript code. This is consistent with the following disclosure of JavaScript in Dierks that is used for the “add to cart” button 105. As shown, the JavaScript uses the `<script>` tag that allows the code to be embedded into an HTML document. *Id.*, ¶ 37 (“The JavaScript may be *stored by the payment processor* and *referenced by code* on the vendor web site, such as the following example code.”).

```
<script
  type="text/javascript"
  src="https://www.mycheckout.com/shoppingcart.js?sellerId=8675309">
</script>
<span id="mycheckout_status_bar"></span>
<script type="text/javascript"
  src="https://www.cmycheckout.com/shoppingcart.js?sellerId=8675309">
</script>
<span id="mycheckout_status_bar"></span>
```

Exhibit 1004, ¶ 37 (annotated)

162. As noted above, the JavaScript may be “stored by the payment processor and referenced by code on the vendor web site.” *Id.*, ¶ 37. A POSITA would have understood this to mean that JavaScript that is embedded into an HTML document on the vendor's web site provides a link to additional JavaScript code

stored on the “payment processor.” This additional JavaScript code is fetched from the payment processor and provided to the browser to effectuate the “add to cart” button 105.

163. This fetching occurs after the browser parses the HTML document and reaches the “**src**” attribute, shown above, that identifies the location of additional code using the shown uniform resource locator (“URL”), i.e., <https://www.mycheckout.com/shoppingcart.js?sellerId=8675309>. A POSITA would have understood that the “**src**” attribute is used to provide a path to an external file or resource that is linked to the HTML document.

164. The JavaScript of Dierks is not web site dependent. It can be embedded within an HTML document of a first web site to provide the “add to cart” button 105, shown in Figure 1C of Dierks. The JavaScript could also be embedded into an HTML document for a different web site. For each web site, the browser will react the same by accessing the URL and obtaining the additional code that establishes the “add to cart” button 105. Accordingly, the code for the “add to cart” button (“*the embeddable buy button*”) is configured to be embedded into different web sites.

165. Therefore, Dierks and Johnston render obvious claim 2.

3. Dependent Claim 3

a) *“The method of claim 1, wherein the electronic commerce store includes a checkout screen that allows a purchase of the item using a transaction processing service operated by the buy button provider.”*

166. Dierks renders obvious claim 1, as I discuss above. Dierks discloses the additional limitation of claim 3.

167. As I discuss for element 1[g], Dierks discloses a **secure checkout** having a **payment mechanism 124** that facilitates payment processing related to the purchase of an item. As shown below in Figure 1E, one of the items selected for purchase is a “**turquoise necklace**.”

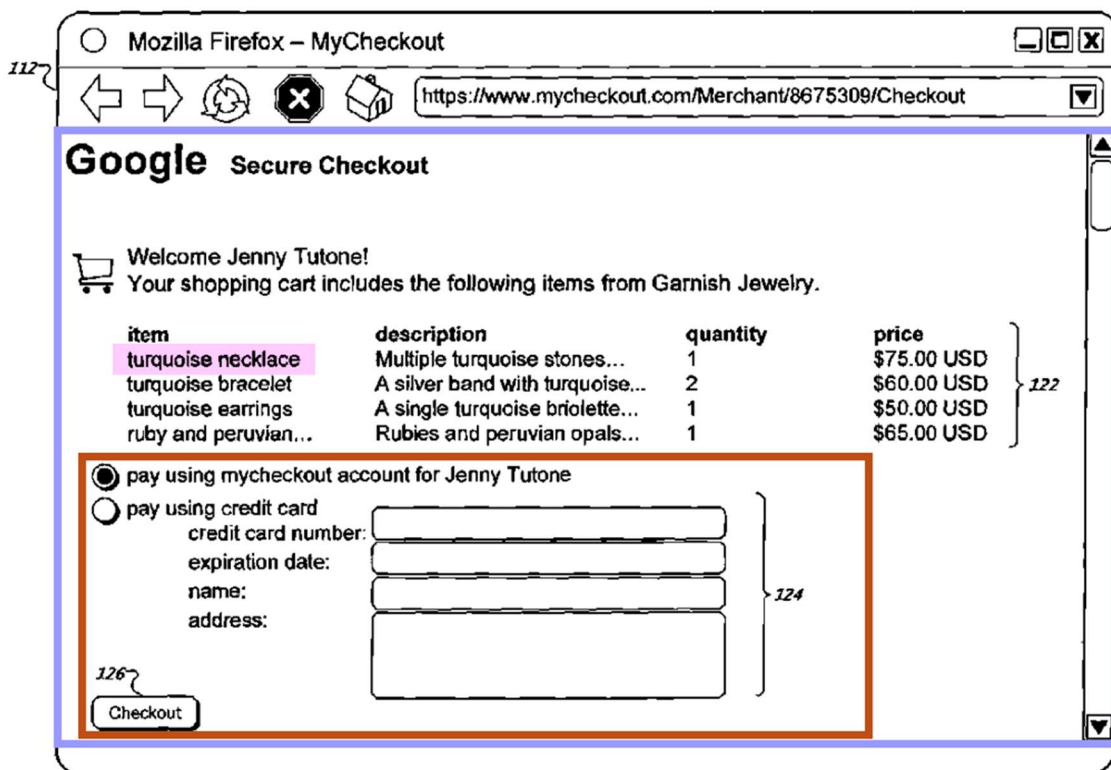


Exhibit 1004, FIG. 1E (annotated)

168. Dierks discloses that the user can “**select a payment method**, such as through a **GOOGLE CHECKOUT** account or a debit or credit card or other payment mechanism 124.” Exhibit 1004, ¶ 48. As I discuss for element 1[g], Dierks discloses that the secure checkout can be displayed within the vendor’s web site, similar to the shopping cart 116. *Id.*, ¶ 34. The secure checkout, including the **payment mechanism 124**, provides a checkout “*screen*” since it is configured as a display. *Id.*, ¶ 48 (“At **display 112**, the user/purchaser has selected a ‘proceed to checkout’ button.¹³...”); *see also id.*, ¶ 98 disclosing that the “checkout request may cause the payment server to generate and serve a checkout page (box 830) which will be displayed on the client.” The terms “page” and “screen” are synonymous, in this context.

169. Thus, Dierks discloses that the **payment mechanism 124** of the **secure checkout** (part of “*the electronic commerce store*”) includes a checkout screen that allows the purchase of an item using GOOGLE CHECKOUT. Dierks discloses that GOOGLE CHECKOUT is a transaction processing service for providing payment to a vendor. *Id.*, ¶ 3 (“For example, person-to-person **transactions** may be paid for

¹³ The “proceed to checkout” button is shown in Figure 1D, which the user presses to access display 112.

through open payment systems such as **GOOGLE CHECKOUT**¹⁴ or PAYPAL, such as when the person providing the product or service (the merchant or vendor) does not accept credit cards.”); *see also id.*, ¶¶ 6, 33, 37, 96; *cf.* Exhibit 1001, 18:8–12 (“In one alternative embodiment, the alternative **transaction processing service** can be any one of Authorize.Net, **Google Checkout**, Storm Pay, PayPal, or other Internet service for transaction processing.”).

170. Further, Dierks discloses that Google Checkout (“*the transaction processing service*”) is operated by the payment processing system 308 (“*the buy button provider*”). *Id.*, ¶ 48 (“At **display 112**, the user/purchaser has selected a ‘proceed to checkout’ button, and is redirected to a web page **operated by the payment server 114**¹⁵.”).

171. Moreover, as I discuss for element 1[h], the payment processing system 308 of Dierks includes a Transaction Module 314 having a “*checkout module 338* [that] may be accessed when a purchaser decides to *finish a transaction and buy* or

¹⁴ A POSITA would have understood that “Google Checkout” refers to “**Google Secure Checkout**” shown in display 112 of Figure 1E because of the overlapping names.

¹⁵ As I discuss in Section VIII.A.1, disclosure regarding the payment server 114 applies to the payment processing system 308.

edit *items* in his or her shopping cart.” *Id.*, ¶ 70. The checkout module 338 manages “*closing out of a transaction*, such as by computing a total cost, any transaction fees, taxes, and shipping/handling, in a conventional manner.” *Id.*

172. Accordingly, Dierks discloses that the payment mechanism 124 of the secure checkout (part of the “*the electronic commerce store*”) includes a checkout screen that allows a purchase of a turquoise necklace (“*the item*”) using Google Checkout (“*a transaction processing service*”) operated by the payment processing system 308 (“*the buy button provider*”).

173. Therefore, Dierks and Johnston render obvious claim 3.

4. Dependent Claim 4

a) “The method of claim 1, wherein the electronic commerce store includes a checkout screen that allows a purchase of the item using a transaction processing service operated by a third-party server.”

174. Dierks and Johnston render obvious claim 1, as I discuss above. Dierks and Johnston also render obvious claim 3, as I discuss above, that requires “wherein the electronic commerce store includes a checkout screen that allows a purchase of the item using a transaction processing service.” Dierks teaches the remaining portion of claim 4 that requires the transaction processing service be “operated by a third-party server.”

175. As shown below in Figure 1E, Dierks discloses that the **payment mechanism 124** of the **secure checkout** allows purchases to be made using a “my

checkout account” and, separately, by using a “credit card.” Exhibit 1004, Figure 1E; *see also id.*, ¶ 48 (“The user may also be permitted to select a payment method, such as through a GOOGLE CHECKOUT account or a **debit or credit card** or other payment mechanism 124.”); *id.*, ¶ 83 (“**submit a charge to a credit card** associated with the user.”); *id.*, ¶ 86 (“For example, the system may make a charge to a **credit card** of the buyer, or to a **bank account** or **other account** associated with the buyer.”).

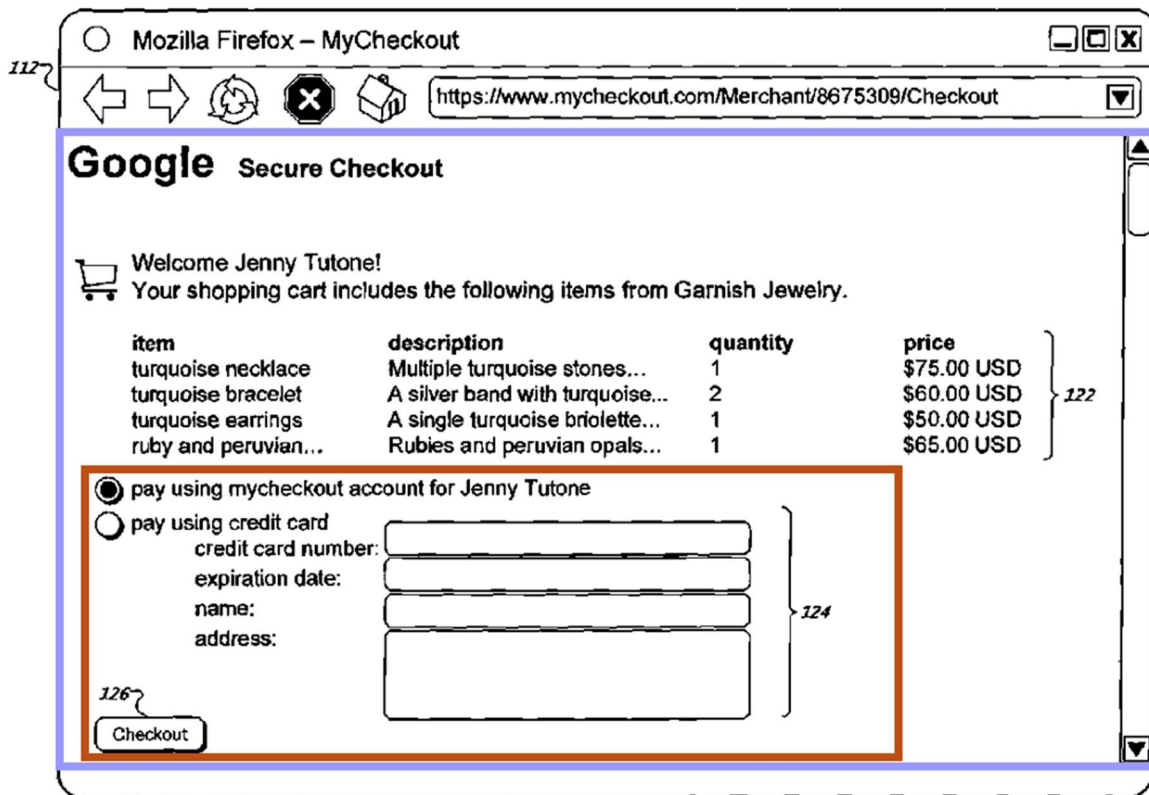


Exhibit 1004, FIG. 1E (annotated)

176. A POSITA would have understood that debit and credit cards are issued by financial institutions and organizations that facilitate transactions, such as credit

unions and banks. It was well-known before the priority date of the '785 patent that conventional credit card authorization systems or debit card payment authorization systems are third-party banking institutions that issue credit or debit cards. *See, e.g.*, Exhibit 1005 (Johnston), ¶ 300 (“credit card vendors”); Exhibit 1006 (Babaria), ¶ 22 (“a third-party credit provider”).

177. Dierks does not limit credit card use to any particular bank or organization. Further, Dierks does not indicate that the payment processing system 308 is a credit provider. Thus, a POSITA would have understood that Dierks’s disclosure of using a credit card as payment, instead of a “my checkout account” would use a transaction processing service operated by a third-party organization that manages credit cards, such as a bank or credit union. It was well-known before the priority date of the '785 patent that transaction processing services for credit cards used servers (“*third-party server*”). *See, e.g.*, Exhibit 1019, ¶ 34 (“The system 100 further includes **credit provider server 112** for approving transactions conducted with a **credit card** and communication network 110 for enabling communications between POS terminal 102 and server 112.”); Exhibit 1020, 2:60–3:2 (“System 100 allows transaction data from ‘front end’ parties, such as merchants, to be combined with payment data from ‘back end’ parties, such as **credit card companies**, issuing banks and other payment systems, so as to **facilitate the processing of payment data**. ...System 100 includes payment processing system

102. Payment processing system 102 can be implemented in hardware, software, or a suitable combination of hardware and software, and can be one or more software systems operating on one or more general purpose **server computing platforms.**”).

178. Accordingly, Dierks discloses that the payment mechanism 124 of the secure checkout (part of the “*the electronic commerce store*”) includes a checkout screen that allows a purchase of a turquoise necklace (“*the item*”) using a transaction processing service operated by a server of a credit card company (“*the third-party server*”).

179. Therefore, Dierks and Johnston render obvious claim 4.

5. Dependent Claim 5

a) “The method of claim 1, wherein the item is a physical good or service.”

180. Dierks and Johnston render obvious claim 1, as I discuss above. Dierks discloses the additional limitation of claim 5.

181. As I discuss for element 1[a], the item is turquoise earrings, i.e., a physical good. Therefore, Dierks and Johnston render obvious claim 5.

6. Dependent Claim 6

a) “The method of claim 1, wherein the representation is selected from a group consisting of a static image, a video file, and a multimedia file.”

182. Dierks and Johnston render obvious claim 1, as I discuss above. Johnston discloses the additional limitation of claim 6 because the “*representation*” is an image (“*a static image*”), as I discuss for element [1a].

183. Therefore, Dierks and Johnston render obvious claim 6.

7. Independent Claim 7

a) 7[preamble]: “A system for embedding buy button for an electronic commerce store, comprising:”

184. In my opinion, regardless of whether the preamble is limiting, Dierks discloses it for at least the same reasons that I discuss above for element 1[preamble].

b) 7[a]: “a first server operated by a buy button provider;”

185. Dierks discloses element 7[a] for at least the same reasons I discuss above for element 1[a].

c) 7[b]: “a database communicatively coupled to the first server, the database configured to store an inventory list of items capable of being offered for sale on the electronic commerce store;”

186. Dierks discloses a “database configured to store an inventory list of items capable of being offered for sale on the electronic commerce store” for at least the same reasons I discuss for element 1[a].

187. As I discuss for element 1[a], Dierks discloses a “Merchant Inventories” database 328 used for storing inventories of items to be offered for sale on the electronic commerce store. The “*item inventory*” I discuss for element 1[a] is an “*inventory list*” because it identifies names of items that are offered for sale. Exhibit 1004, ¶ 53 (“the seller may optionally also open and populate an **inventory database** for goods to be offered for sale by the seller. The seller in particular may identify **names** and descriptions for **items to be sold** (which may include goods or services), along with prices for the items.”).

188. Dierks also discloses that the database is communicatively coupled to the first server. Limitation 7[b] does not require the “database” and the “first server” to have particular structures. As I discuss above in Section X.A.1.b, for limitation 1[a], Dierks discloses that the payment processing system 308 comprises “**one or more** web servers and other such **hardware**, or may **take other appropriate forms**.” Exhibit 1004, ¶ 62.

189. A POSITA would have understood that the Merchant Inventories database 328 in Dierks includes the actual data stored in the database, i.e., merchant inventories data, and when the payment processing system 308 of Dierks is configured with one server, that server would have access to the merchant inventories data and is therefore communicatively coupled to the Merchant Inventories database 328 shown in Figure 3 of Dierks.

190. If the “database” of limitation 7[c] is argued by Patent Owner to require a database that is external to the first server, which is not required by the claim, then such a claim interpretation would have also been taught because Dierks discloses that the payment processing system may comprise multiple configurations, i.e. “one or more web servers and other such hardware, or may take other appropriate forms.” *Id.* ¶ 62. Indeed, it was well-known before the priority date of Dierks to provide databases external to web servers. *See, e.g.*, Exhibit 1024 (Kossmann) Figs. 2-4 showing multiple different known implementations where databases are external to and in communication with web servers. Exhibit 1024, Figs. 2–4; Exhibit 1032. As shown on the cover, Kossmann indicates “*SIGMOD’10*, June 6–11, 2010, Indianapolis, Indiana, USA. Copyright 2010 ACM 978-1-4503-0032-2/10/06.” Exhibit 1024, 579. A POSITA would understand that this means that the paper was publicly available at that conference such that it would have been sufficiently accessible to the public interested in the technology disclosed in Kossmann. This is consistent with ACM digital library for this paper indicating a publication date of “06 June 2010.” *See* <https://dl.acm.org/doi/10.1145/1807167.1807231#>, a printout of which is included as Exhibit 1032.

191. Moreover, Dierks teaches that the database for storing the vendor’s inventory list may be external to the payment processing server. *See, e.g.*, Exhibit 1004, ¶ 41 (“In addition, where the vendor stores information relating to the vendor’s

inventory using a system *associated with* payment server 114, such as **GOOGLE BASE**, the payment server 114 can also *check such a database* for availability of the selected goods.”); *id.*, ¶ 86 (“In addition, *where an inventory of the seller is available to the payment processing system (e.g., through Google Base)*, the inventory may be decreased for each of the purchased items (or an indicator to decrease the inventory may be provided so that the inventory is decreased when the item is shipped).”).

192. As noted above, Dierks refers to Google Base as being a database system that is “associated with” the payment server 114¹⁶ (¶ 41) and “available to the payment process system” (¶ 86). A database system used by the vendor that is “associated with” a server or “available to” a payment processing system suggests a database that is external to, but in communication with, the server or payment processing system. Also, the Merchant Inventories database 328 in Figure 3 of Dierks is shown as being communicatively coupled with the HTML Generator 312 and the Transaction Module 314. Thus, in view of the configuration options disclosed by Dierks, a POSITA would have been taught that one of those

¹⁶ As I discuss in Section VIII.A.1, disclosure regarding the payment server 114 applies to the payment processing system 308.

configurations is to provide the merchant inventory list in a database that is external to and in communication with the first server.

193. Therefore, Dierks discloses element 7[d].

d) 7[c]: “an embeddable buy button builder executed by the first server, wherein the embeddable buy button builder is configured to receive a selection of an item from the inventory list and a representation of the item,”

194. Dierks and Johnston render obvious element 7[c] obvious for at least the same reasons I discuss above for elements 1[a] and 7[b].

195. As I discuss for element 1[a], Dierks renders obvious “receiving, by an embeddable buy button builder executing on a first server operated by a buy button provider, a selection of an item from an item inventory and a representation of the item, wherein the item is to be listed for sale on the electronic commerce store.”

196. Dierks’s disclosure of “an embeddable buy button builder **executing on a first server**” for element 1[a] discloses “an embeddable buy button builder **executed by the first server**,” as recited in element 7[c].

197. Dierks’s teaching of “receiving, a selection of an item from an item inventory and a representation of the item” for element 1[a], likewise teaches “to receive a selection of an item from the inventory list and a representation of the item,” as recited in element 7[c]. While element 1[a] recites “an item inventory” and not an “inventory list,” my analysis for element 7[b] describes how the “item inventory” is an “inventory list.”

198. Further, Dierks discloses that the embeddable buy button builder is “**configured** to receive” a selection, as recited in element 7[c], for the same reasons that Dierks discloses “receiving, by an embeddable buy button builder...a selection,” as I discuss for element 1[a].

199. Therefore, Dierks and Johnston render obvious element 7[c].

e) 7[d]: “the embeddable buy button builder further configured to generate the embeddable buy button,”

200. Dierks discloses element 7[d] for at least the same reasons I discuss above for element 1[b]. As I discuss for element 1[b], the HTML Generator 312 (“*the embeddable buy button builder*”) generates the code for the “add to cart” button and therefore generates the “add to cart” button (“*the embeddable buy button*”). The HTML Generator 312 is “configured” to generate the code for the “add to cart” button by virtue of the HTML Generator 312 generating the code.

f) 7[e]: “wherein the embeddable buy button is configured to be embedded into a web site by the buy button provider,”

201. Dierks discloses element 7[e] for at least the same reasons I discuss above for element 1[d]. As I discuss for element 1[d], Dierks discloses “embedding, by the buy button provider, the embeddable buy button into a web site.” Since Dierks discloses that the buy button provider embeds the embeddable buy button into a web site, the *embeddable buy button is configured to be embedded into the web site by the buy button provider*.

202. Therefore, Dierks discloses element 7[e].

g) 7[f] wherein the embeddable buy button includes a hyperlink; and”

203. Dierks discloses element 7[f] for at least the same reasons I discuss above for element 1[c]. As I discuss for element 1[c], Dierks discloses “wherein the embeddable buy button is configured to display the representation of the item, and a hyperlink.”

204. The “add to cart” button (“*the embeddable buy button*”) in Dierks “*includes*” the hyperlink, as required by element 7[f]. In particular, when the “add to cart” button 105 is pressed, it causes an HTTP request and, therefore, includes a hyperlink. Exhibit 1004, ¶ 38 (“[s]election of the ***button 105 may cause an HTTP request***, in the form of an XMLHttpRequest (XHR), to be made of payment server 114¹⁷, as indicated by Arrow A.”).

205. Therefore, Dierks discloses element 7[f].

h) 7[g]: “a second server configured to execute the web site,

206. Dierks discloses element 7[g] for at least the same reasons I discuss above for element 1[e].

¹⁷ As I discuss in Section VIII.A.1, disclosure regarding the payment server 114 applies to the payment processing system 308.

i) 7[h] wherein the buy button provider controls operation of the embeddable buy button on the web site, and

207. Dierks discloses element 7[h].

208. The payment processing system 308 (*“the buy button provider”*) in Dierks controls operation of the “add to cart” button (*“the embeddable buy button”*) on the web site.

209. **First**, as I discuss for element 1[b], the payment processing system 308 includes the HTML Generator 312 that generates the code for the “add to cart” button 105. The HTML Generator 312 controls operation of the “add to cart” button 105 by virtue of generating the code.

210. **Second**, Dierks discloses that “[s]election of the button 105 may cause an HTTP request, in the form of an XMLHttpRequest (XHR), **to be made of payment server 114**¹⁸.” Exhibit 1004, ¶ 38. Dierks then discloses that the “payment server 114 may then **parse the HTTP request** to obtain an **identifier for the purchaser**, and also to **identify the vendor** and **relevant payment parameters**.” *Id.*, ¶ 40. Thus, the payment processing system 308 is controlling operation of the “add to cart” button by acting on the HTTP request.

¹⁸ As I discuss in Section VIII.A.1, disclosure regarding the payment server 114 applies to the payment processing system 308.

211. *Third*, Dierks discloses that the payment server 114 passes information back to the purchaser based on operation of the “add to cart” button. *Id.*, ¶ 42 (“When the purchaser and vendor have been identified and the balance of the purchaser’s shopping cart incremented, the **changes may be passed back to the client device of the purchaser**, as shown by Arrow B. ... As one example, **the payment server may pass information** reflecting a **description of the selected item(s), their quantity, their price, and a total price of goods** in the purchaser’s shopping cart.”).

212. Figure 1C of Dierks illustrates the **shopping cart** on the web site that is updated based on the user’s selection of the “**add to cart**” button 105. *Id.*, Fig. 1C, ¶ 43 (“Display 106 shows the **result of a user’s selection of button 105** (see FIG. 1C). Specifically, a **shopping cart object shown** as a graphical element such as an iframe 116 has been created on display 106.”).

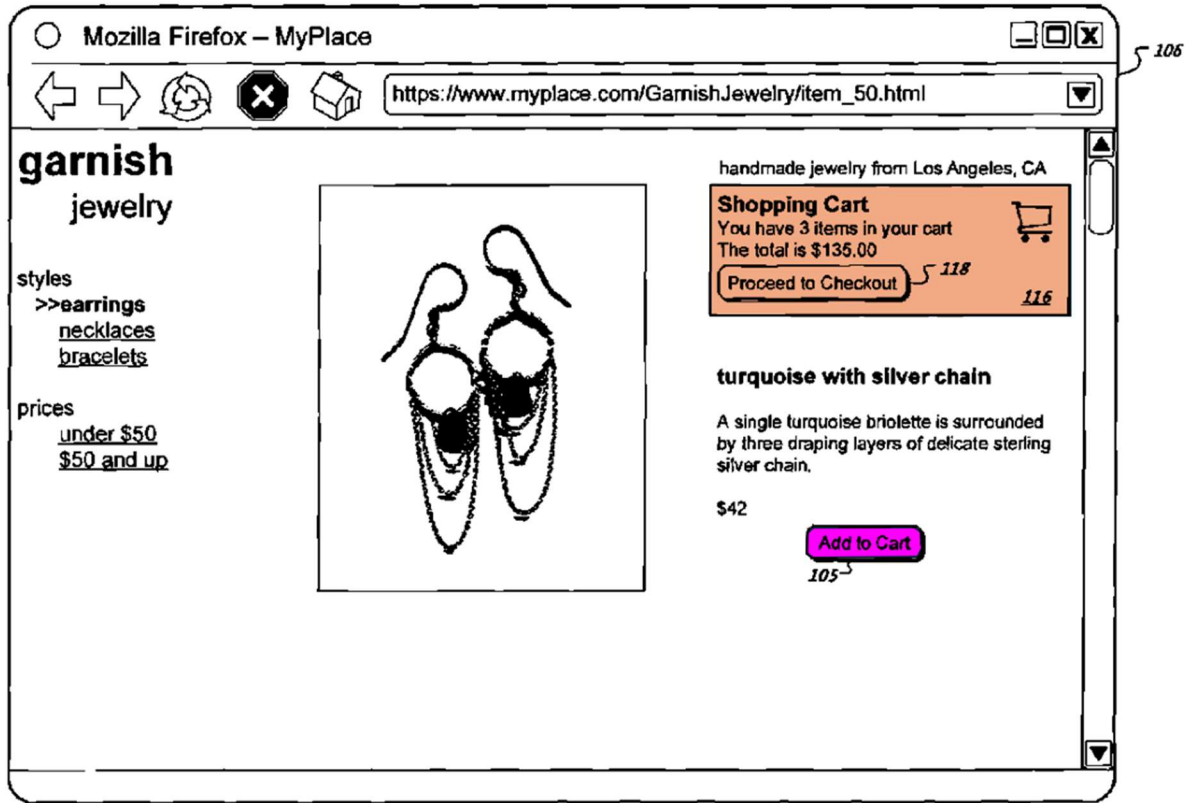


Exhibit 1004, FIG. 1C (annotated)

213. Accordingly, the payment processing system 308 (*“the buy button provider”*) controls operation of the “add to cart” button (*“the embeddable buy button”*) on the web site because (1) the payment processing system 308 creates the “add to cart” button and therefore controls the flow of information that occurs when the “add to cart” button is pressed, (2) the payment processing system 308 parses the information received after “add to cart” button is pressed, and (3) the payment processing system 308 controls the flow of information back to the web site to update the shopping cart based on the pressing of the “add to cart” button.

214. Therefore, Dierks discloses element 7[h].

j) 7[i]: “wherein the buy button provider controls operation of the electronic commerce store on the web site upon selection of the hyperlink.”

215. Dierks discloses element 7[i].

216. As I discuss for element 1[h], the electronic commerce store, including the shopping cart and secure checkout, is operated by the payment processing system 308 (“*the buy button provider*”).

217. As I discuss for element 7[f], the “add to cart” button 105 includes the hyperlink and when the “add to cart” button 105 is pressed, an HTTP request is sent to the payment server 114¹⁹. Exhibit 1004, ¶ 38 (“[s]election of the **button 105 may cause an HTTP request**, in the form of an XMLHttpRequest (XHR), to be made of payment server 114, as indicated by Arrow A.”).

218. As I discuss for element 7[h], selection of the “add to cart” button, and therefore selection of the hyperlink, causes the payment processing system 308 to display the shopping cart on the web site. *Id.*, ¶ 43 (“Display 106 shows the **result of a user's selection of button 105** (see FIG. 1C). Specifically, a **shopping cart object shown** as a graphical element such as an iframe 116 has been created on display 106.”).

¹⁹ As I discuss in Section VIII.A.1, disclosure regarding the payment server 114 applies to the payment processing system 308.

219. Accordingly, the payment processing system 308 controls operation of the shopping cart (part of “*the electronic commerce store*”) by sending code to display the shopping cart upon selection of the hyperlink. *Id.*, ¶ 43 (“The iframe 116 [shopping cart object] is generated using the information received from the payment server 114²⁰.”); ¶ 43 (“Example code for generating the window for the iframe [shopping cart object] may take the form of...”); *id.*, Fig. 3 (“Cart Generator”).

220. The payment processing system 308 is also responsible for controlling operation by updating the shopping cart with information, including a description of the selected items, a quantity of the items, and a price of the items. *Id.*, ¶ 42 (“As one example, **the payment server may pass information reflecting a description of the selected item(s), their quantity, their price, and a total price of goods in the purchaser’s shopping cart.**”).

221. Further, as I discuss for element 1[h], the payment processing system 308 (“*the buy button provider*”) operates and therefore controls the secure checkout, including the “checkout module 338 [that] may be accessed when a purchaser decides to finish a transaction and buy or edit items in his or her shopping cart.” *Id.*, ¶ 70.

²⁰ As I discuss in Section VIII.A.1, disclosure regarding the payment server 114 applies to the payment processing system 308.

222. Accordingly, the payment processing system 308 of Dierks controls operation of the electronic commerce store on the web site upon selection of the hyperlink.

223. Therefore, Dierks discloses element 7[i].

8. Dependent Claim 8

a) “The system of claim 7, further comprising a transaction processing service operated by the buy button provider, the transaction processing service configured to process a payment initiated via the electronic commerce store.”

224. Dierks and Johnston render obvious claim 7, as I discuss above. Dierks discloses the additional limitation of claim 8.

225. As I discuss for claim 3, Dierks discloses Google Checkout (“a transaction processing service”) operated by the payment processing system 308 (“the buy button provider”).

226. Google Checkout is configured to process a payment initiated via the electronic store, such as being initiated through the display 112 of Figure 1E, by debiting the purchaser’s account. Exhibit 1004, ¶ 33 (“The process shows an example for user interaction with an on-line payment system, such as GOOGLE CHECKOUT”); *id.*, ¶ 48 (“The user may also be permitted to select a payment method, such as through a GOOGLE CHECKOUT”); *id.*, ¶ 51 (“When the purchaser

chooses to complete the purchase, their account is debited by the payment server 114. ...”).

227. Therefore, Dierks and Johnston render obvious claim 8.

9. Dependent Claim 9

a) “The system of claim 7, wherein the buy button provider is configured to execute the electronic commerce store.”

228. Dierks and Johnston render obvious claim 7, as I discuss above. Dierks discloses the additional limitation of claim 9.

229. Dierks discloses that the payment processing system 308 (*“the buy button provider”*) is configured to execute the shopping cart (part of the *“electronic commerce store”*) and the secure checkout with payment mechanism 124 (also part of *“the electronic commerce store”*). The payment processing system 308 executes the electronic commerce store by establishing and controlling the shopping cart and secure checkout, as I discuss for element 7[i], and by updating the shopping cart after the user selects the “add to cart” button 105, as I also discuss for element 7[i].

230. Further, the shopping cart and secure checkout in Dierks are executed by the payment processing system 308 when the user selects the “proceed to checkout” button 118 shown in the shopping cart 116, as I discuss for elements 1[g] and 7[i]. Exhibit 1004, ¶ 44 (“The purchaser may also choose to **go to a checkout to pay** or otherwise edit entries in their shopping cart, by **selecting** the ‘**proceed to**

checkout’ control button 116²¹.”). Selecting the “proceed to checkout” button 118 causes the payment processing system 308 to execute functions, including (1) generating the secure checkout, (2) checking that database fields have been updated, and (3) closing out the transaction. *Id.*, ¶ 48 (“At display 112, the user/purchaser has selected a ‘proceed to checkout’ button, and is redirected to a web page operated by the payment server 114²². To generate this page, the **payment server 114 may check its database fields that have been updated through the purchaser’s selection of ‘add to cart’ buttons.** At this point, the **interaction with the user** may occur according to conventional mechanisms to permit closing out of the transaction. For example, the user may be shown a list of items and their prices 122 that have been selected by the user. The user may then select items to delete or may increase or otherwise change the quantities of items. Such changes will be **reflected in associated databases managed by the payment processor.**”).

²¹ The “control button” of the shopping cart is shown in Figure 1C as “118” and not “116.”

²² As I discuss in Section VIII.A.1, disclosure regarding the payment server 114 applies to the payment processing system 308.

231. The payment processing system 308 also executes the electronic commerce store by facilitating and processing payment of the item, as I discuss for claim 8.

232. Accordingly, Dierks discloses that the payment processing system 308 (*“the buy button provider”*) executes the shopping cart and secure checkout (*“the electronic commerce store”*) by providing the aforementioned functionality.

233. Therefore, Dierks and Johnston render obvious claim 9.

10. Dependent Claim 10

a) “The system of claim 7, wherein the embeddable buy button builder generates hypertext-markup-language code that is configured to be inserted into the web site.”

234. Dierks and Johnston render obvious claim 7, as I discuss above. Dierks discloses the additional limitation of claim 10.

235. As I discuss for element 7[d], the HTML generator 312 (*“the embeddable buy button builder”*) is configured to generate the “add to cart” button (*“the embeddable buy button.”*). A POSITA would have understood that HTML stands for hypertext-markup-language. Dierks also discloses that the HTML is “code” and is configured to be inserted into the web site. Exhibit 1004, ¶ 62 (**“The HTML generator, in this example, is configured to receive parameters regarding items to be sold by vendors and to generate code, such as HTML or XHTML, for the vendors to include in their web pages, to generate controls for making ordering**

and selection of the items easier for visitors to the sites.”); *id.*, ¶ 64 (“**HTML generator 312** may then store the parameters in parameters module 322, and may access generic **HTML code** from templates module 324. By combining the template information with the supplied parameters in a conventional manner, the **HTML generator 312** may produce **mark-up code** for the user that the user can employ to provide payment processing services **on their web site.**”).

236. Figure 4 of Dierks, shown below, illustrates an example of hypertext-markup-language that is configured to be inserted into a web site, including **code 418** for the “**add to cart**” button.

Mozilla Firefox - MyCheckout

https://www.mycheckout.com/Merchant/8675309/CreateItemCode/

1. Choose a button type.

Basic Button (402) Add to Cart (404)

2. Choose a button style.

My web page has a White (406) background.

3. Describe your item.

Item Name: turquoise earrings (408) Price: 50.00 (410) USD

Item Description: A single turquoise briolette is surrounded by three draping layers of delicate sterling silver chain. (412)

Create Button Code (414)

Insert this code for shopping cart:

```
<script type="text/javascript" src="https://www.mycheckout.com/shoppingcart.js?sellerId=8675309"></script><span id="mycheckout_status_bar"></span>
```

Insert this code for add to cart button:

```
<form action="https://www.mycheckout.com/Merchant/8675309/add" onSubmit="return false;" method="post"><input type="hidden" name="item" value="PD84bVWgdmVyc2lvcj0IMS4wliBibmNvZGluZz0IVVRGLTgIPz4NCjxJdGVtIHhtbG5zPSJodHRwOi8vY2h1Y2tvdXQuZ29vZ2xLmNvbS9zY2h1bWEvMlI+DQogIDx0dWFudG9eT4xPC9xdWFudG9eT4NCiAgPHVuaXQ
```

FIG. 4

Exhibit 1004, FIG. 4 (annotated)

Id., ¶ 74 (“The remote server may then use the submitted information to generate code for managing a shopping cart, along with code for generating an add to cart button. The code for managing a shopping cart, displayed in area 416, may include JavaScript code or a pointer to JavaScript code stored at a remote server. ... The **code for generating an add to cart button**, displayed in area 418, may **take the form of HTML code** as shown in more detail above.”).

237. The hypertext-markup-language code for the “add to cart” button is configured to be inserted in the web site. *Id.*, ¶ 74 (“The code in both areas 416 and 418 may be highlighted by a user, so that **it can be cut-and-pasted into the user’s web page.**”); *see also* analysis of claim 2, and elements 1[d], 7[e], that discuss how Dierks discloses to embed code into the web site.

238. Thus, Dierks discloses that the HTML Generator 312 (“*the embeddable buy button builder*”) generates hypertext-markup-language code that is configured to be inserted into the web site.

239. Therefore, Dierks and Johnston render obvious claim 10.

11. Dependent Claim 11

a) “The system of claim 7, wherein the representation is selected from a group consisting of a static image, a video file, and a multimedia file.”

240. Dierks and Johnston render obvious claim 7, as I discuss above. Dierks and Johnston render claim 11 obvious for at least the same reasons I discuss above for claim 6.

12. Dependent Claim 12

a) “The system of claim 7, wherein the embeddable buy button is configured to be embedded into different web sites.”

241. Dierks and Johnston render obvious claim 7, as I discuss above. Dierks discloses claim 12 for at least the same reasons I discuss above for claim 2. Therefore, Dierks and Johnston render obvious claim 12.

13. Dependent Claim 13

a) “The system of claim 7, wherein the electronic commerce store includes a checkout screen is configured to facilitate the purchase of the item.”

242. Dierks and Johnston render obvious claim 7, as I discuss above. Dierks discloses the additional limitation of claim 13 for at least the same reasons I discuss for claims 3 and 17.

243. As I discuss for claim 3, Dierks discloses that “*the electronic commerce store includes a checkout screen.*” As I discuss for claim 17, Dierks discloses that “*the checkout screen is configured to facilitate the purchase of the item.*”

244. Therefore, Dierks and Johnston render obvious claim 13.

14. Independent Claim 14

a) 14[preamble]: “A method for embedding a buy button for an electronic commerce store, comprising:”

245. In my opinion, regardless of whether the preamble is limiting, Dierks discloses it for at least the same reasons I discuss above for element 1[preamble].

b) 14[a]: “receiving, by embeddable buy button builder executing on a first server operated by a buy button provider, a selection of an item to sell on a web site and a representation of the item,

246. Dierks and Johnston render obvious element 14[a] for at least the same reasons I discuss above for element 1[a]. Element 14[a] differs from element 1[a] by reciting “*a selection of an item to sell on a web site*” (element 14[a]) instead of “*a selection of an item from inventory*” (element 1[a]). As I discuss for element 1[a], the turquoise earrings (“*an item*”) are selected to sell on a web site. Exhibit 1004, ¶ 73 (“Text entry box 408 receives entry of a **product name**, while text entry box 410 receives a product price, and text entry box 412 receives a product description. Such entries may be coded along with the button, so as to be **displayed on the vendor’s web page.** ...”).

247. Therefore, Dierks and Johnston render obvious element 14[a].

c) 14[b] wherein the web site is executed by a second server not operated by the buy button provider;

248. Dierks discloses element 14[b] for at least the same reasons I discuss above for element 1[e].

d) 14[c]: “generating, by a first server, the embeddable buy button containing software configured to be executed by the first server on the web site;”

249. Dierks discloses element 14[c]. As I discuss for element 1[b], Dierks discloses “*generating, by the first server, the embeddable buy button.*” As I discuss for claims 2 and 10, Dierks discloses that the embeddable buy button contains JavaScript code. Exhibit 1004, ¶ 37 (“As explained in more detail below, the **button 105** is generated using **JavaScript** (or, e.g., Jscript or VBScript) associated with the vendor’s underlying web page that generates display 104.”). Code written using JavaScript is “software” since the code includes instructions used by a computer to execute a task.

250. As I also discuss for claim 2, the JavaScript disclosed by Dierks includes a link to additional JavaScript stored at the payment processor, e.g., the processing system 308. *Id.*, ¶ 37 (“[T]he button 105 is generated using JavaScript. ... That Javascript code ... may be obtained from a payment processor such as the GOOGLE CHECKOUT service. The JavaScript may be **stored by the payment processor and referenced by code on the vendor web site.**”).

251. Dierks discloses the following JavaScript code for the “add to cart” button (“*the embeddable buy button*”).

```
<script
  type="text/javascript"
  src="https://www.mycheckout.com/shoppingcart.js?sellerId=8675309">
</script>
<span id="mycheckout__status__bar"></span>
<script type="text/javascript"
  src="https://www.cmyheckout.com/shoppingcart.js?sellerId=8675309">
</script>
<span id="mycheckout__status__bar"></span>
```

Exhibit 1004, ¶ 37 (annotated)

252. As I discuss for claim 2, A POSITA would have understood that JavaScript is used with an HTML document by employing the `<script>` tag at the beginning and end of the JavaScript code. The source or “src” attribute (shown above) is used to access JavaScript stored at the payment processor, i.e., stored at the payment processing system 308. *Id.*, ¶ 37 (“The JavaScript may be **stored by the payment processor and referenced by code on the vendor web site. ...**”); *id.*, ¶ 78 (“code may also be stored remotely of the client device, and **pointers to the code may be placed in the seller's web page**. In addition, the code incorporated into the web page may itself **point to other code, such as Javascript code.**”).

253. The JavaScript code shown above as being bound by the `<script>` tag is executed by the first server because the first server (1) receives an HTTP request

for additional code, that is triggered by the above-shown JavaScript code, and (2) provides the web site with the additional code that is used to display the shopping cart. *Id.*, ¶ 38 (“**Selection of the button 105** may cause an **HTTP request**, in the form of an XMLHttpRequest (XHR), to be made of **payment server 114**²³, as indicated by Arrow A”); *id.*, ¶ 43 (“Display 106 shows the result of a user’s selection of button 105 (see FIG. 1C). Specifically, a **shopping cart object** shown as a graphical element such as **an iframe 116 has been created on display 106**. The iframe 116 is **generated** using the information received **from the payment server 114**.”). Thus, the “add to cart” button contains software configured to be executed by the first server because the software causes the HTTP request to the first server so that the first server responds by providing information to generate the shopping cart on the web site.

254. Accordingly, Dierks discloses generating, by a first server, the “add to cart” button (“*the embeddable buy button*”) containing JavaScript code (“software”) configured to be executed by the first server on the web site.

255. Dierks discloses element 14[c].

²³ As I discuss in Section VIII.A.1, disclosure regarding the payment server 114 applies to the payment processing system 308.

e) 14[d]: “embedding, by the buy button provider, the embeddable buy button into the web site, wherein the embeddable buy button includes a hyperlink; and”

256. Dierks discloses element 14[d] for at least the same reasons I discuss above for elements 1[d] and 7[f]. In particular, as I discuss for element 1[d], Dierks discloses “embedding, by the buy button provider, the embeddable buy button into a web site.” As I discuss for element 7[f], Dierks discloses that “the embeddable buy button includes a hyperlink.”

257. Therefore, Dierks discloses element 14[d].

f) 14[e]: “executing, by the buy button provider, the software upon selection of the hyperlink, wherein the software is configured to display the electronic commerce store within the web site.”

258. Dierks discloses element 14[e].

259. As I discuss for element 14[c], the “add to cart” button (“*the embeddable buy button*”) encompasses JavaScript code (“*the software*”) that is embedded in the web site and is configured to display the shopping cart (part of “*the electronic commerce store*”) within the web site.

260. As I discuss for element 14[d], Dierks discloses that the “add to cart” button includes a hyperlink. In particular, when the “add to cart” button 105 is pressed, it sends an HTTP request to the payment processing system 308. Exhibit 1004, ¶ 38 (“[s]election of the *button 105 may cause an HTTP request*, in the form

of an XMLHttpRequest (XHR), to be made of payment server 114²⁴, as indicated by Arrow A.”).

261. Further, as I discuss for element 1[f], “the electronic commerce store is displayed within the web site upon selection of the hyperlink.” The shopping cart object 116 is displayed within the web site upon selection of the “add to cart” 105 button that includes the hyperlink. *Id.*, ¶ 43 (“Display 106 shows the result of a **user’s selection of button 105** (see FIG. 1C). Specifically, a **shopping cart object** shown as a graphical element such as an iframe 116 has been created on display 106.”).

262. The payment processing system 308 executes the software by receiving the HTTP request and causing the shopping cart to be displayed. *Id.*, ¶ 43 (“Specifically, a **shopping cart** object shown as a graphical element such as an **iframe 116** has been created on display 106. The iframe 116 is **generated using the information received from the payment server 114.**”); *id.*, ¶ 67 (“transaction processor 342 [part of the payment processing system 308] may open a new shopping cart for the pair, and may populate the cart with information associated with the selected control.”).

²⁴ As I discuss in Section VIII.A.1, disclosure regarding the payment server 114 applies to the payment processing system 308.

263. Thus, Dierks discloses executing, by the payment processing system 308 (“*the buy button provider*”), the JavaScript code (“*the software*”) upon selection of the hyperlink, wherein the software is configured to display the shopping cart object (part of the “*electronic commerce store*”) within the web site, as required by element 14[e].

264. Therefore, Dierks and Johnston render obvious element 14[e].

15. Dependent Claim 15

a) “The method of claim 14, wherein the embeddable buy button is configured to be embedded into different web sites.”

265. Dierks and Johnston render obvious claim 14, as I discuss above. Dierks discloses the additional limitation of claim 15 for at least the same reasons I discuss above for claim 2.

266. Therefore, Dierks and Johnston render obvious claim 15.

16. Dependent Claim 16

a) “The method of claim 14, wherein the electronic commerce store includes a checkout screen is (sic) operated by the buy button provider.”

267. Dierks and Johnston render obvious claim 14, as I discuss above. Dierks discloses the additional limitation of claim 16. As I discuss for claim 3, Dierks discloses that **the electronic commerce store includes a checkout screen** that allows a purchase of the item using a transaction processing service **operated by the payment processing system 308** (“*the buy button provider*”). Dierks also

discloses that the payment processing system 308 operates the display 112 that includes the checkout screen shown below in Figure 1E. Exhibit 1004, ¶ 48 (“At **display 112**, the user/purchaser has selected a ‘proceed to checkout’ button, and is redirected to a web page **operated by the payment server 114**²⁵.”); *see also id.* ¶ 20 (“FIGS. 1A-1E show example **screen shots** associated with the conceptual flow diagram of FIG. 1.”).

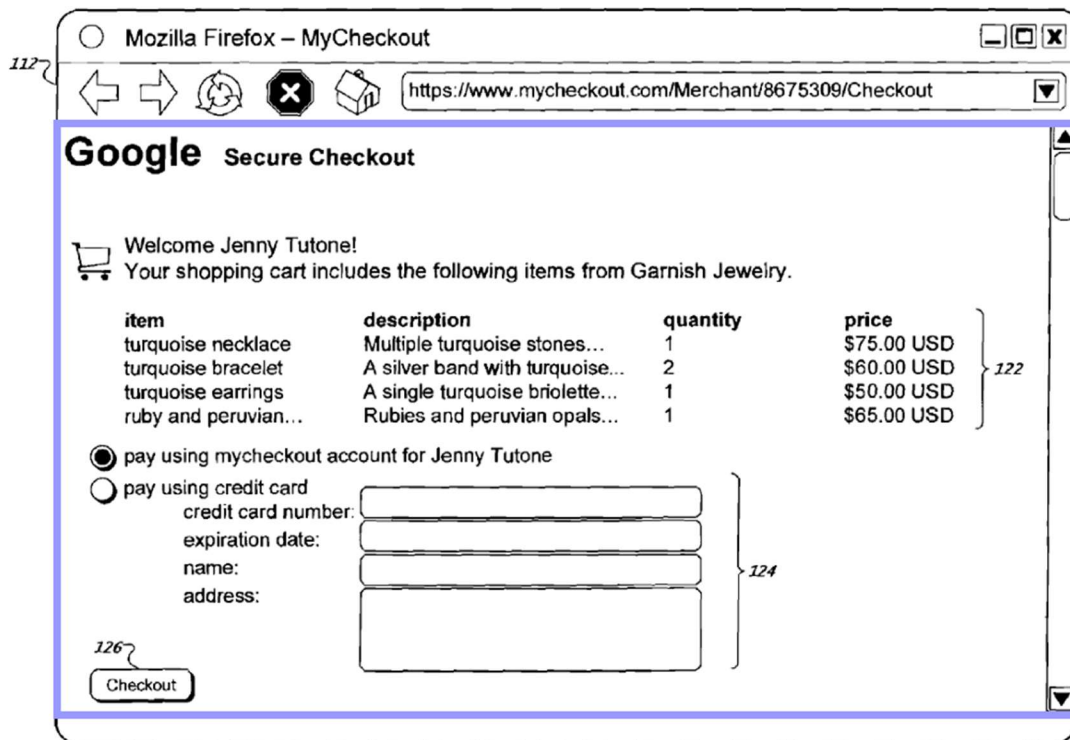


Exhibit 1004, FIG. 1E (annotated)

268. Therefore, Dierks and Johnston render obvious claim 16.

²⁵ As I discuss in Section VIII.A.1, disclosure regarding the payment server 114 applies to the payment processing system 308.

17. Dependent Claim 17

a) “The method of claim 16, wherein the checkout screen is configured to facilitate the purchase of the item by a transaction processing service operated by the buy button provider.”

269. Dierks and Johnston render obvious claim 16, as I discuss above. Dierks discloses the additional limitation of claim 17. As I discuss for claim 3, Dierks discloses that “the electronic commerce store includes a **checkout screen that allows a purchase of the item using a transaction processing service operated by the buy button provider.**” Dierks’s disclosure of a “checkout screen *that allows a purchase of the item*” also discloses that “the checkout screen is *configured to facilitate the purchase of the item.*”

270. The checkout screen of Dierks facilitates the purchase of the item using the payment mechanism 124, as I describe for claim 3, that provides a way for the user to purchase the item. Exhibit 1004, ¶ 48 (“The user may also be permitted to **select a payment method**, such as through a GOOGLE CHECKOUT account or a debit or credit card or other **payment mechanism 124.**”).

271. Therefore, Dierks and Johnston render obvious claim 17.

18. Dependent Claim 18

a) “The method of claim 14, wherein the representation is selected from a group consisting of a static image, a video file, and a multimedia file.”

272. Dierks and Johnston render obvious claim 14, as I discuss above. Dierks and Johnston render claim 18 obvious for at least the same reasons I discuss above for claim 6.

273. Therefore, Dierks in view of Johnston renders claim 18 obvious.

19. Dependent Claim 19

a) “The method of claim 14, wherein the item is a physical good or a service.”

274. Dierks and Johnston render obvious claim 14, as I discuss above. Dierks discloses the additional limitation of claim 19 for at least the same reasons I discuss above for claim 5.

275. Therefore, Dierks and Johnston render obvious claim 19.

20. Dependent Claim 20

a) “The method of claim 14, wherein (sic) checkout screen allows a purchase of the item using at least one of a previously stored credit card, a debit card or bank account information.”

276. Dierks and Johnston render obvious claim 14, as I discuss above. Dierks discloses the additional limitation of claim 20. As I discuss for claim 17, Dierks discloses a secure checkout screen that is configured to facilitate the purchase of the item. Dierks further discloses that the secure checkout screen allows

purchasing the item using at least one of a previously stored credit card, a debit card or bank account information, as recited in claim 20.

277. Dierks's **checkout screen** is shown below in Figure 1E with the **payment mechanism 124** that provides multiple payment options including using a “mycheckout” account and a credit card.

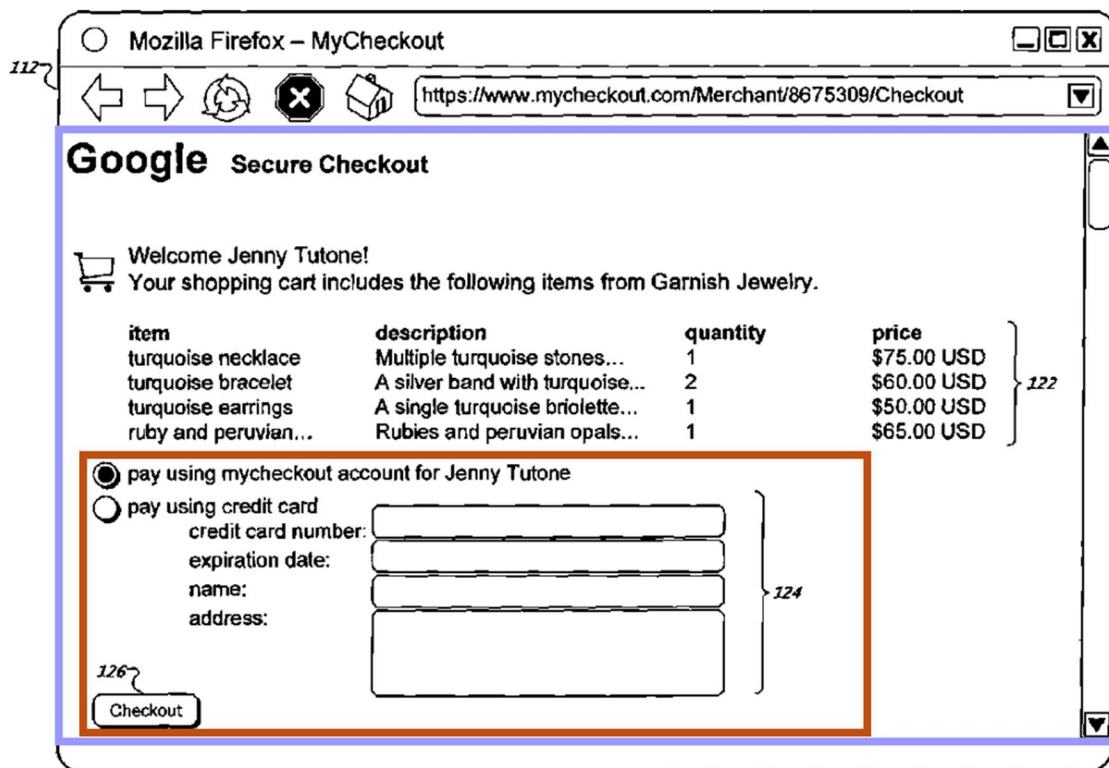


Exhibit 1004, FIG. 1E (annotated)

278. The “pay using mycheckout account” option is shown as being selected in Figure 1E. A POSITA would have understood that the option to pay using “mycheckout” does not require entering credit card information because payment information was previously stored. This is further supported by Dierks disclosing

that the payment processing system 308 stores payment information for future use. Exhibit 1004, ¶ 48 (“payment server 114²⁶ may also **store a preferred payment method** for the user and use that method (or present it as a default) if the user does not select another.”).

279. Dierks also discloses that the stored information may be a credit card that is accessed after the user authenticates themselves. Exhibit 1004, ¶ 83 (“The shopper, or buyer, first **authenticates themselves** to a payment processing system (box 602). Such authentication allows the system to **associate subsequent purchase requests with the user’s account**, such as to remove money from the user’s account or **submit a charge to a credit card associated with the user.**”); *see also id.*, ¶ 86 (“After obtaining final confirmation, **the system debits** and credits the **appropriate accounts**. For example, the system may **make a charge to a credit card of the buyer**, or to a **bank account** or other account **associated with the buyer.**”).

280. Moreover, a POSITA would have understood that the previously stored information is either a credit card, a debit card, or bank account information because Dierks discloses that GOOGLE CHECKOUT may be used for payment. *Id.*, ¶ 48

²⁶ As I discuss in Section VIII.A.1, disclosure regarding the payment server 114 applies to the payment processing system 308.

(“The user may also be permitted to **select a payment method**, such as through a **GOOGLE CHECKOUT account** or a **debit** or **credit card** or **other payment mechanism 124.**”). It was well-known prior to the priority date of the ’785 patent that GOOGLE CHECKOUT stored credit card, debit card or bank account information of users. *See, e.g.*, Exhibit 1023, 5:50-65 (“Consumers 101 having an account with a third party payment processor can make purchases from Internet merchants 120 without disclosing sensitive payment information (e.g., credit card number) to the Internet merchant 120. Instead, the consumer 101 provides payment information for one or more financial accounts (e.g., checking account, credit card account, etc.) to the third party payment processor for use in Internet transactions. When purchasing a product from an Internet merchant 120 that accepts payments via the third party payment processor, the consumer 101 can select from one of the financial accounts to pay for the product. The third party payment processor can then debit the selected account for the amount of the purchase and credit this amount to an account of the Internet merchant 120. Examples of third party payment processors include **GOOGLE CHECKOUT** and **PAYPAL**”).

281. Accordingly, Dierks discloses that the checkout screen allows a purchase of the item using at least one of a previously stored credit card, a debit card or bank account information.

282. Therefore, Dierks and Johnston render obvious claim 20.

X. CONCLUSION

283. Based on the foregoing, it is my opinion that claims 1–20 of the '785 patent are invalid.

XI. AVAILABILITY FOR CROSS EXAMINATION

284. In signing this declaration, I recognize that the declaration will be filed as evidence in a contested case before the Patent Trial and Appeal Board of the United States Patent and Trademark Office. I also recognize that I may be subject to cross examination in the case and that cross examination will take place within the United States. If cross examination is required of me, I will appear for cross examination within the United States during the time allotted for cross examination.

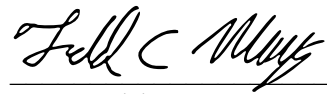
XII. RIGHT TO SUPPLEMENT

285. I reserve the right to supplement my opinions in the future to respond to any arguments that the Patent Owner raises and to take into account new information as it becomes available to me.

XIII. JURAT

286. I declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code.

Dated: November 21, 2024



Dr. Todd Mowry

EXHIBIT A

Todd C. Mowry

CURRICULUM VITAE

November, 2024

Contact Information

Computer Science Department
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA 15213

Phone: (412) 268-3725
Fax: (412) 268-5576
Email: tcm@cs.cmu.edu
URL: <http://www.cs.cmu.edu/~tcm>

Education

Ph.D. in Electrical Engineering, Stanford University, March 1994.

- Thesis: “Tolerating Latency Through Software-Controlled Data Prefetching.”
- Supervisors: Anoop Gupta and Monica Lam.

M.S. in Electrical Engineering, Stanford University, June 1989.

B.S. in Electrical Engineering with Highest Distinction, University of Virginia, May 1988.

Academic Appointments

7/08–Present: Professor, Computer Science Department (with a courtesy appointment in the Department of Electrical and Computer Engineering), Carnegie Mellon University.

8/09–6/10: Associate Department Head for Faculty, Computer Science Department

6/15–Present: Director of the Ph.D. program, Computer Science Department

7/97–6/08: Associate Professor, Computer Science Department (with a courtesy appointment in the Department of Electrical and Computer Engineering), Carnegie Mellon University.

12/93–6/97: Assistant Professor, Department of Electrical and Computer Engineering (with a courtesy appointment in the Department of Computer Science), University of Toronto.

9/89–11/93: Graduate Research Assistant, Stanford University.

Honors and Awards

- ASPLOS Influential Paper Award, 2022 (for “Design and Evaluation of a Compiler Algorithm for Prefetching,” from ASPLOS 1992).
- Best Paper Runner-Up Award, VLDB conference, 2018.
- ACM Fellow, for contributions to software prefetching and thread-level speculation, 2016.
- SCS Doctoral Dissertation Award co-won by Angela Demke Brown (Ph.D. advisee), 2005.
- Most Thought-Provoking Idea Award (for Claytronics), the Wild and Crazy Idea Session IV held at ASPLOS-XI, 2004.
- Best Paper Award, the 20th International Conference on Data Engineering (ICDE), 2004.
- SCS Doctoral Dissertation Award co-won by J. Gregory Steffan (Ph.D. advisee), 2003.
- Runner-Up for Best Paper Award, the ACM SIGMOD Conference, 2001.
- Alfred P. Sloan Research Fellow, 1999–2001.
- TR100 Award (awarded by MIT’s Technology Review magazine to the top 100 most promising young innovators in science and technology), 1999.
- Best Paper Award, the Second Symposium on Operating Systems Design and Implementation, 1996.

Doctoral Thesis Supervision

Student	Thesis Title (or Topic)	Graduation Date
Chi-Keung Luk	Optimizing the Cache Performance of Non-Numeric Applications. <i>(Nominated for the ACM Thesis Award by the University of Toronto Department of Computer Science.)</i>	January, 2000
J. Gregory Steffan	Hardware Support for Thread-Level Speculation. <i>(Co-winner of the SCS Doctoral Dissertation Award.)</i>	April, 2003
Antonia Zhai	Compiler Optimization of Value Communication for Thread-Level Speculation.	January, 2005
Angela Demke Brown	Explicit Compiler-based Memory Management for Out-of-core Applications. <i>(Co-winner of the SCS Doctoral Dissertation Award.)</i>	May, 2005
Christopher Colohan	Applying Thread-Level Speculation to Database Transactions.	November, 2005
Shimin Chen	Redesigning Database Systems in Light of CPU Cache Prefetching.	December, 2005
Amit Manjhi	Increasing the Scalability of Dynamic Web Applications	March, 2008
Olatunji Ruwase	Improving Device Driver Reliability Through Decoupled Dynamic Binary Analysis	May, 2013
Michelle Goodstein	Dataflow Analysis-Based Dynamic Parallel Monitoring	August, 2014
Vivek Seshadri	Simple DRAM and Virtual Memory Abstractions to Enable Highly Efficient Memory Subsystems	May, 2016
Gennady Pekhimenko	Practical Data Compression for Modern Memory Hierarchies	July, 2016
Chris Fallin	Finding and Exploiting Parallelism with Data-Structure-Aware Static and Dynamic Analysis	February, 2019
Prashanth Menon	On Building Robustness into Compilation-Based Main-Memory Database Query Engines	May, 2021
Pratik Fegade	Auto-batching Techniques for Dynamic Deep Learning Computations	January, 2023
Ziqi Wang	Building a More Efficient Cache Hierarchy by Supporting Multiversioning and Leveraging Application-Level Information	January, 2023
Sam Arch	Unifying Declarative and Non-Declarative Code in Relational Database Management Systems	June, 2027 <i>(expected)</i>
Hongyi Jin	Compiler Optimizations for Machine Learning Applications	June, 2027 <i>(expected)</i>
Ruihang Lai	Compiler Optimizations for Machine Learning Applications	June, 2027 <i>(expected)</i>
Patrick Coppock	OS Support for Machine Learning Applications	June, 2027 <i>(expected)</i>
Eliot Solomon	OS Support for Machine Learning Applications	June, 2029 <i>(expected)</i>

Publications

Book Chapters

1. Amir Yazdanbakhsh, Gennady Pekhimenko, Hadi Esmaeilzadeh, Onur Mutlu, and Todd C. Mowry. Towards Breaking the Memory Bandwidth Wall Using Approximate Value Prediction. In *Approximate Circuits: Methodologies and CAD*, Springer International Publishing, 2019, pages 417–441.

Refereed Journal Articles

1. Mishra, Deepanjali, Konstantinos Kanellopoulos, Ashish Panwar, Akshitha Sriraman, Vivek Seshadri, Onur Mutlu, and Todd C. Mowry. Address Scaling: Architectural Support for Fine-Grained Thread-Safe Metadata Management. In *IEEE Computer Architecture Letters*, 23(01), January 2024.
2. Prashanth Menon, Amadou Ngom, Lin Ma, Todd C. Mowry, and Andrew Pavlo. Permutable Compiled Queries: Dynamically Adapting Compiled Queries without Recompiling. In *Proceedings of the VLDB Endowment*, 14(2):101-113, October 2020.
3. Amir Yazdanbakhsh, Gennady Pekhimenko, Bradley Thwaites, Hadi Esmaeilzadeh, Onur Mutlu, and Todd C. Mowry. RFVP: Rollback-Free Value Prediction with Safe-to-Approximate Loads. In *ACM Transactions on Architecture and Code Optimization (TACO)*, 12(4):62, February 2016.
4. Amir Yazdanbakhsh, Gennady Pekhimenko, Bradley Thwaites, Hadi Esmaeilzadeh, Onur Mutlu, and Todd C. Mowry. Mitigating the Memory Bottleneck with Approximate Load Value Prediction. In *IEEE Design and Test*, 33(1), January 2016.
5. Vivek Seshadri, Samihan Yedkar, Hongyi Xin, Onur Mutlu, Phillip B. Gibbons, Michael A. Kozuch, and Todd C. Mowry. Mitigating Prefetcher-Caused Pollution Using Informed Caching Policies for Prefetched Blocks. In *ACM Transactions on Architecture and Code Optimization (TACO)*, 11(4), January 2015.
6. Shimin Chen, Phillip B. Gibbons, Michael Kozuch, and Todd C. Mowry. Log-Based Architectures: Using Multicore to Help Software Behave Correctly. In *Operating Systems Review*, 45(1), January 2011.
7. Shimin Chen, Michael Kozuch, Phillip B. Gibbons, Michael Ryan, Theodoros Strigkos, Todd C. Mowry, Olatunji Ruwase, Evangelos Vlachos, Babak Falsafi, and Vijaya Ramachandran. Flexible Hardware Acceleration for Instruction-Grain Lifeguards. In *IEEE Micro (Top Picks from 2008 Computer Architecture Conferences)*, 29(1), January 2009.
8. Seth C. Goldstein, Todd C. Mowry, Jason D. Campbell, Michael P. Ashley-Rollman, Michael De Rosa, Stanislav Funiak, James F. Hoburg, Mustafa E. Karagozler, Brian Kirby, Peter Lee, Padmanabhan Pillai, J. Robert Reid, Daniel D. Stancil, and Michael P. Weller. Beyond Audio and Video: Using Claytronics to Enable Pario. In *AI Magazine*, 30(2), March 2009.
9. Antonia Zhai, J. Gregory Steffan, Christopher B. Colohan, and Todd C. Mowry. Compiler and Hardware Support for Reducing the Synchronization of Speculative Threads. In *ACM Transactions on Architecture and Code Optimization (TACO)*, 5(1), May 2008.
10. Christopher B. Colohan, Anastassia Ailamaki, J. Gregory Steffan, and Todd C. Mowry. Incrementally parallelizing database transactions with thread-level speculation. In *ACM Transactions on Computer Systems (TOCS)*, 26(1), February 2008.
11. Shimin Chen, Anastassia Ailamaki, Phillip B. Gibbons, and Todd C. Mowry. Improving Hash Join Performance through Prefetching. In *ACM Transactions on Database Systems*, 32(3):1-32, September 2007.

12. Christopher B. Colohan, Anastasia Ailamaki, J. Gregory Steffan, and Todd C. Mowry. CMP Support for Large and Dependent Speculative Threads. In *IEEE Transactions on Parallel and Distributed Systems*, 18(8):1041-1054, August 2007.
13. J. Gregory Steffan, Christopher B. Colohan, Antonia Zhai and Todd C. Mowry. The STAMPede Approach to Thread-Level Speculation. In *ACM Transactions on Computer Systems*, 23(3):253-300, August 2005.
14. Seth Copen Goldstein, Jason Campbell and Todd C. Mowry. Programmable Matter. In *IEEE Computer*, 38(6):99-101, June 2005.
15. Angela Demke Brown, Todd C. Mowry and Orran Krieger. Compiler-Based I/O Prefetching for Out-of-Core Applications. In *ACM Transactions on Computer Systems*, 19(2):111-170, May 2001.
16. Chi-Keung Luk and Todd C. Mowry. Architectural and Compiler Support for Effective Instruction Prefetching: A Cooperative Approach. In *ACM Transactions on Computer Systems*, 19(1):71-109, February 2001.
17. Todd C. Mowry and Chi-Keung Luk. Understanding Why Correlation Profiling Improves the Predictability of Data Cache Misses in Nonnumeric Applications. In *IEEE Transactions on Computers*, 49(4), April 2000.
18. Chi-Keung Luk and Todd C. Mowry. Automatic Compiler-Inserted Prefetching for Pointer-Based Applications. In *IEEE Transactions on Computers*, 48(2), February 1999.
19. Mark Horowitz, Margaret Martonosi, Todd C. Mowry, and Michael D. Smith. Informing Memory Operations: Memory Performance Feedback Mechanisms and their Applications. In *ACM Transactions on Computer Systems*, 16(2):170-205, May 1998.
20. Todd C. Mowry. Tolerating Latency in Multiprocessors through Compiler-Inserted Prefetching. In *ACM Transactions on Computer Systems*, 16(1):55-92, February 1998.
21. Todd Mowry and Anoop Gupta. Tolerating Latency Through Software-Controlled Prefetching in Shared-Memory Multiprocessors. In *Journal of Parallel and Distributed Computing*, 12(2):87-106, 1991.

Refereed Conference Papers

1. Sam Arch, Yuchen Liu, Todd C. Mowry, Jignesh Patel, and Andy Pavlo. The Key to Effective UDF Optimization: Before Inlining, First Perform Outlining. To appear in *Proceedings of VLDB 18*, 2025.
2. Fegade, Pratik, Tianqi Chen, Phillip Gibbons, and Todd Mowry. ACROBAT: Optimizing Auto-batching of Dynamic Deep Learning at Compile Time. In *Proceedings of the Sixth Conference on Machine Learning and Systems (MLSys)*, May 2024.
3. Kai Franz, Samuel Arch, Denis Hirn, Torsten Grust, Todd C. Mowry, and Andrew Pavlo. Dear User-Defined Functions, Inlining Isn't Working Out So Great For Us. Let's Try Batching To Make Our Relationship Work. Sincerely, SQL. To appear in *Proceedings of the 2024 Conference on Innovative Data Systems Research (CIDR)*, January 2024.
4. Wang, Ziqi, Kaiyang Zhao, Pei Li, Andrew Jacob, Michael Kozuch, Todd Mowry, and Dimitrios Skarlatos. Memento: Architectural Support for Ephemeral Memory Management in Serverless Environments. In *Proceedings of the 56th Annual IEEE/ACM International Symposium on Microarchitecture (Micro)*, October 2023.
5. Siyuan Chen, Pratik Pramod Fegade, Phillip Gibbons, Todd C. Mowry, and Tianqi Chen. ED-Batch: Efficient Automatic Batching of Dynamic Neural Networks via Learned Finite State Machines. In *Proceedings of the Fortieth International Conference on Machine Learning (ICML 2023)*, July 2023.

6. Graham Gobieski, Souradip Ghosh, Marijn Heule, Todd C. Mowry, Tony Nowatzki, Nathan Beckmann, and Brandon Lucia. Riptide: A programmable, energy-minimal dataflow compiler and architecture. In *Proceedings of the 2022 55th IEEE/ACM International Symposium on Microarchitecture (MICRO)*, October 2022.
7. Pratik Fegade, Tianqi Chen, Phillip Gibbons, and Todd C. Mowry. The CoRa Tensor Compiler: Compilation for Ragged Tensors with Minimal Padding. In *Proceedings of the Fifth Conference on Machine Learning and Systems (MLSys)*, August 2022.
8. Amadou Ngom, Prashanth Menon, Matthew Butrovich, Lin Ma, Wan Shen Lim, Todd C. Mowry, and Andrew Pavlo. Filter Representation in Vectorized Query Execution. In *Proceedings of the 17th International Workshop on Data Management on New Hardware (DaMoN)*, June 2021.
9. Ziqi Wang, Chul-Hwan Choo, Michael A. Kozuch, Todd C. Mowry, Gennady Pekhimenko, Vivek Seshadri, and Dimitrios Skarlatos. NVOOverlay: Enabling Efficient and Scalable High-Frequency Snapshotting to NVM. In *Proceedings of the 48th Annual International Symposium on Computer Architecture (ISCA)*, June 2021.
10. Pratik Fegade, Tianqi Chen, Phillip Gibbons, and Todd C. Mowry. Cortex: A Compiler for Recursive Deep Learning Models. In *Proceedings of the Fourth Conference on Machine Learning and Systems (MLSys)*, April 2021.
11. Daming D. Chen, Phillip B. Gibbons, and Todd C. Mowry. TardisTM: Incremental Repair for Transactional Memory. In *Proceedings of the Eleventh International Workshop on Programming Models and Applications for Multicores and Manycores (PMAM '20)*, February 2020.
12. Ziqi Wang, Michael A. Kozuch, Todd C. Mowry, and Vivek Seshadri. Multiversioned Page Overlays: Enabling Faster Serializable Hardware Transactional Memory. In *Proceedings of the 28th International Conference on Parallel Architectures and Compilation Techniques (PACT)*, September 2019.
13. Vivek Seshadri, Donghyuk Lee, Thomas Mullins, Hasan Hassan, Amirali Boroumand, Jeremie Kim, Michael A. Kozuch, Onur Mutlu, Phillip B. Gibbons, and Todd C. Mowry. Ambit: in-memory accelerator for bulk bitwise operations using commodity DRAM technology. In *Proceedings of the 50th Annual International Symposium on Microarchitecture (MICRO-50)*, October 2017.
14. Prashanth Menon, Andrew Pavlo, and Todd C. Mowry. Relaxed Operator Fusion for In-Memory Databases: Making Compilation, Vectorization, and Prefetching Work Together At Last. In *Proceedings of VLDB 11(1): 1-13*, September 2017.
15. Andrew Pavlo, Gustavo Angulo, Joy Arulraj, Haibin Lin, Jiexi Lin, Lin Ma, Prashanth Menon, Todd Mowry, Matthew Perron, Ian Quah, Siddharth Santurkar, Anthony Tomasic, Skye Toor, Dana Van Aken, Ziqi Wang, Yingjun Wu, Ran Xian, and Tieying Zhang. Self-Driving Database Management Systems” In *Proceedings of the 2017 Conference on Innovative Data Systems Research (CIDR)*, January 2017.
16. Gennady Pekhimenko, Evgeny Bolotin, Nandita Vijaykumar, Onur Mutlu, Todd C. Mowry, and Stephen W. Keckler. A Case for Toggle-Aware Compression for GPU Systems. In *Proceedings of the 22nd International Symposium on High-Performance Computer Architecture (HPCA)*, March 2016.
17. Michelle L. Goodstein, Phillip B. Gibbons, Michael A. Kozuch, and Todd C. Mowry. Tracking and Reducing Uncertainty in Dataflow Analysis-Based Dynamic Parallel Monitoring. In *Proceedings of the 2015 International Conference on Parallel Architectures and Compilation Techniques (PACT '15)*, October 2015.
18. Nandita Vijaykumar, Gennady Pekhimenko, Adwait Jog, Abhishek Bhowmick, Rachata Ausavarungnirun, Chita Das, Mahmut Kandemir, Todd C. Mowry, and Onur Mutlu. A Case for Core-Assisted Bottleneck Acceleration in GPUs: Enabling Efficient Data Compression. In *Proceed-*

- ings of the 42nd Annual International Symposium on Computer Architecture (ISCA), June 2015.
19. Vivek Seshadri, Gennady Pekhimenko, Olatunji Ruwase, Onur Mutlu, Phillip B. Gibbons, Michael A. Kozuch, Todd C. Mowry, and Trishul Chilimbi. Page Overlays: An Enhanced Virtual Memory Framework to Enable Fine-grained Memory Management. In *Proceedings of the 42nd Annual International Symposium on Computer Architecture (ISCA)*, June 2015.
 20. Gennady Pekhimenko, Tyler Huberty, Rui Cai, Onur Mutlu, Phillip P. Gibbons, Michael A. Kozuch, and Todd C. Mowry. Exploiting Compressed Block Size as an Indicator of Future Reuse. In *Proceedings of the 21st International Symposium on High-Performance Computer Architecture (HPCA)*, February 2015.
 21. Vivek Seshadri, Abhishek Bhowmick, Onur Mutlu, Phillip B. Gibbons, Michael A. Kozuch, and Todd C. Mowry. The Dirty-Block Index. In *Proceedings of the 41st Annual International Symposium on Computer Architecture (ISCA)*, June 2014.
 22. Olatunji Ruwase, Michael A. Kozuch, Phillip B. Gibbons, and Todd C. Mowry. Guardrail: High Fidelity Correctness Checking of Device Drivers for Safeguarding I/O Operations. In *Proceedings of the Nineteenth International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS 2014)*, March 2014.
 23. Gennady Pekhimenko, Vivek Seshadri, Yoongu Kim, Hongyi Xin, Onur Mutlu, Phillip B. Gibbons, Michael A. Kozuch, and Todd C. Mowry. Linearly Compressed Pages: A Low-Complexity, Low-Latency Main Memory Compression Framework. In *Proceedings of the 46th Annual International Symposium on Microarchitecture (MICRO-46)*, December 2013.
 24. Vivek Seshadri, Yoongu Kim, Chris Fallin, Donghyuk Lee, Rachata Ausavarungnirun, Gennady Pekhimenko, Yixin Luo, Onur Mutlu, Phillip B. Gibbons, Michael A. Kozuch, and Todd C. Mowry. RowClone: Fast and Energy-Efficient In-DRAM Bulk Data Copy and Initialization. In *Proceedings of the 46th Annual International Symposium on Microarchitecture (MICRO-46)*, December 2013.
 25. Michelle L. Goodstein, Shimin Chen, Phillip B. Gibbons, Michael A. Kozuch, and Todd C. Mowry. Chrysalis Analysis: Incorporating Synchronization Arcs in Dataflow-Analysis-based Parallel Monitoring. In *Proceedings of the 21st International Conference on Parallel Architectures and Compilation Techniques (PACT-2012)*, September 2012.
 26. Vivek Seshadri, Onur Mutlu, Todd C. Mowry, and Michael A. Kozuch. The Evicted-Address Filter: A Unified Mechanism to Address Both Cache Pollution and Thrashing. In *Proceedings of the 21st International Conference on Parallel Architectures and Compilation Techniques (PACT-2012)*, September 2012.
 27. Gennady Pekhimenko, Vivek Seshadri, Onur Mutlu, Todd C. Mowry, Phillip B. Gibbons, and Michael A. Kozuch. Base-Delta-Immediate Compression: A Practical Data Compression Mechanism for On-Chip Caches. In *Proceedings of the 21st International Conference on Parallel Architectures and Compilation Techniques (PACT-2012)*, September 2012.
 28. Olatunji Ruwase, Shimin Chen, Phillip Gibbons, and Todd C. Mowry. Decoupled Lifeguards: Enabling Path Optimizations for Dynamic Correctness Checking Tools. In *Proceedings of the ACM SIGPLAN 2010 Conference on Programming Language Design and Implementation (PLDI)*, June 2010.
 29. F. Ryan Johnson, Radu Stoica, Anastasia Ailamaki, and Todd C. Mowry. Decoupling Contention Management from Scheduling. In *Proceedings of the Fifteenth International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS 2010)*, March 2010.
 30. Michelle Goodstein, Evangelos Vlachos, Shimin Chen, Phillip B. Gibbons, Michael Kozuch, and Todd C. Mowry. Butterfly Analysis: Adapting Dataflow Analysis to Dynamic Parallel Monitoring. In *Proceedings of the Fifteenth International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS 2010)*, March 2010.

31. Evangelos Vlachos, Michelle Goodstein, Michael Kozuch, Shimin Chen, Babak Falsafi, Phillip B. Gibbons, and Todd C. Mowry. ParaLog: Enabling and Accelerating Online Parallel Monitoring of Multithreaded Applications. In *Proceedings of the Fifteenth International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS 2010)*, March 2010.
32. Amit Manjhi, Charles Garrod, Bruce M. Maggs, Todd C. Mowry, Anthony Tomasic. Holistic Query Transformations for Dynamic Web Applications. In *Proceedings of the 2009 IEEE 25th International Conference on Data Engineering (ICDE)*, March-April 2009.
33. Daniel J. Dewey, Michael P. Ashley-Rollman, Michael DeRosa, Seth Copen Goldstein, Todd C. Mowry, Siddhartha S. Srinivasa, Padmanabhan Pillai, and Jason Campbell. Generalizing metamodules to simplify planning in modular robotic systems. In *Proceedings of the IEEE/RSJ 2008 International Conference on Intelligent Robots and Systems (IROS)*, September 2008.
34. Charles Garrod, Amit Manjhi, Anastasia Ailamaki, Bruce Maggs, Todd Mowry, Christopher Olston, and Anthony Tomasic. Scalable Query Result Caching for Web Applications. In *Proceedings of the 34th International Conference on Very Large Databases (VLDB)*, August 2008.
35. Lei Li, Wenjie Fu, Fan Guo, Todd C. Mowry, and Christos Faloutsos. Cut-and-stitch: efficient parallel learning of linear dynamical systems on SMPs. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, August 2008.
36. Shimin Chen, Michael Kozuch, Theodoros Strigkos, Babak Falsafi, Phillip B. Gibbons, Todd C. Mowry, Michael Ryan, Olatunji Ruwase, and Evangelos Vlachos. Flexible Hardware Acceleration for Instruction-Grain Program Monitoring. In *Proceedings of the 35th Annual International Symposium on Computer Architecture (ISCA)*, June 2008.
37. Olatunji Ruwase, Phillip B. Gibbons, Todd C. Mowry, Vijaya Ramachandran, Shimin Chen, Michael Kozuch, and Michael Ryan. Parallelizing dynamic information flow tracking. In *Proceedings of the 20th Annual ACM Symposium on Parallel Algorithms and Architectures (SPAA)*, June 2008.
38. Brian Kirby, Burak Aksak, James Hoburg, Todd C. Mowry, and Padmanabhan Pillai. A Modular Robotic System Using Magnetic Force Effectors. In *Proceedings of the IEEE/RSJ 2007 International Conference on Intelligent Robots and Systems (IROS)*, October 2007.
39. Michael Ashley-Rollman, Seth Goldstein, Peter Lee, Todd C. Mowry, and Padmanabhan Pillai. Meld: A Declarative Approach to Programming Ensembles. In *Proceedings of the IEEE/RSJ 2007 International Conference on Intelligent Robots and Systems (IROS)*, October 2007.
40. Amit Manjhi, Phillip B. Gibbons, Anastassia Ailamaki, Charles Garrod, Bruce M. Maggs, Todd C. Mowry, Christopher Olston, Anthony Tomasic, and Haifeng Yu. Invalidation Clues for Database Scalability Services. In *Proceedings of the 2007 IEEE 23rd International Conference on Data Engineering (ICDE)*, pages 316-325, April 2007.
41. Michael De Rosa, Peter Lee, Seth Goldstein, Jason Campbell, Padmanabhan Pillai, and Todd C. Mowry. Distributed Watchpoints: Debugging Large Multi-Robot Systems. In *Proceedings of the 2007 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3723-3729, April 2007.
42. Benjamin D. Rister, Jason Campbell, Padmanabhan Pillai, and Todd C. Mowry. Integrated Debugging of Large Modular Robot Ensembles. In *Proceedings of the 2007 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2227-2234, April 2007.
43. Shimin Chen, Phillip Gibbons, Michael Kozuch, Vasileios Liaskovitis, Anastassia Ailamaki, Guy Blelloch, Babak Falsafi, Limor Fix, Nikos Hardavellas, Todd C. Mowry and Chris Wilkerson. Scheduling Threads for Constructive Cache Sharing on CMPs. In *Proceedings of 18th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, pages 105-115, June 2007.

44. Shimin Chen, Babak Falsafi, Phillip B. Gibbons, Michael Kozuch, Todd C. Mowry, Radu Teodorescu, Anastassia Ailamaki, Limor Fix, Gregory R. Ganger, Bin Lin, and Steven W. Schlosser. Log-Based Architectures for General-Purpose Monitoring of Deployed Code. In *Proceedings of the Workshop on Architectural and System Support for Improving Software Dependability (ASID), held with ASPLOS XII*, October 2006.
45. Christopher B. Colohan, Anastassia Ailamaki, J. Gregory Steffan, and Todd C. Mowry. Tolerating Dependences Between Large Speculative Threads Via Sub-Threads. In *Proceedings of the 33rd Annual International Symposium on Computer Architecture (ISCA)*, pages 216-226, June 2006.
46. Amit Manjhi, Anastassia Ailamaki, Bruce M. Maggs, Todd C. Mowry, Christopher Olston, and Anthony Tomasic. Simultaneous Scalability and Security for Data-Intensive Web Applications. In *Proceedings of the 2006 ACM SIGMOD International Conference on Management of Data*, pages 241-252, June 2006.
47. Vasileios Liaskovitis, Shimin Chen, Phillip B. Gibbons, Anastassia Ailamaki, Guy E. Blelloch, Babak Falsafi, Limor Fix, Nikos Hardavellas, Michael Kozuch, Todd C. Mowry, and Chris Wilkerson. Parallel Depth First vs. Work Stealing Schedulers on CMP Architectures. In *Proceedings of 18th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, August 2006.
48. Brian Kirby, Jason Campbell, Burak Aksak, Padmanabhan Pillai, James F. Hoburg, Todd C. Mowry, and Seth Copen Goldstein. Catoms: Moving Robots without Moving Parts. In *Proceedings of the Twentieth National Conference on Artificial Intelligence (AAAI)*, pages 1730-1731, July 2005.
49. Christopher Olston, Amit Manjhi, Charles Garrod, Anastassia Ailamaki, Bruce M. Maggs, and Todd C. Mowry. A Scalability Service for Dynamic Web Applications. In *Proceedings of the Second Biennial Conference on Innovative Data Systems Research (CIDR)*, pages 56-69, January 2005.
50. Christopher B. Colohan, Anastassia Ailamaki, J. Gregory Steffan, and Todd C. Mowry. Optimistic Intra-Transaction Parallelism on Chip Multiprocessors. In *Proceedings of the 31st International Conference on Very Large Data Bases (VLDB)*, pages 73-84, September 2005.
51. Shimin Chen, Anastassia Ailamaki, Phillip B. Gibbons, and Todd C. Mowry. Inspector Joins. In *Proceedings of the 31st International Conference on Very Large Data Bases (VLDB)*, pages 817-828, September 2005.
52. Antonia Zhai, Christopher B. Colohan, J. Gregory Steffan, and Todd C. Mowry. Compiler Optimization of Memory-Resident Value Communication Between Speculative Threads. In *Proceedings of the 2007 International Symposium on Code Generation and Optimization (CGO)*, pages 39-52, March 2004.
53. Shimin Chen, Anastassia Ailamaki, Phillip B. Gibbons and Todd C. Mowry. Improving Hash Join Performance through Prefetching. In *Proceedings of the 20th International Conference on Data Engineering (ICDE)*, pages 116-127, April 2004. (*This paper received the Best Paper Award.*)
54. Antonia Zhai, Christopher B. Colohan, J. Gregory Steffan and Todd C. Mowry. Compiler Optimization of Scalar Value Communication Between Speculative Threads. In *Proceedings of the Tenth International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS-X)*, October 2002.
55. Shimin Chen, Philip B. Gibbons, Todd C. Mowry and Gary Valentin. Fractal Prefetching B+-Trees: Optimizing Both Cache and Disk Performance. In *Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data*, June 2002.
56. J. Gregory Steffan, Christopher B. Colohan, Antonia Zhai and Todd C. Mowry. Improving Value Communication for Thread-Level Speculation. In *Proceedings of the Eighth International Symposium on High-Performance Computer Architecture (HPCA)*, February 2002.

57. Shimin Chen, Philip B. Gibbons and Todd C. Mowry. Improving Index Performance through Prefetching. In *Proceedings of the 2001 ACM SIGMOD International Conference on Management of Data*, pages 235-246, May 2001. (*This paper received the Runner-Up for Best Paper Award.*)
58. Angela Demke Brown and Todd C. Mowry. Taming the Memory Hogs: Using Compiler-Inserted Releases to Manage Physical Memory Intelligently. In *Proceedings of the Fourth Symposium on Operating Systems Design and Implementation (OSDI 2000)*, pages 31-44, October 2000.
59. J. Gregory Steffan, Christopher B. Colohan, Antonia Zhai and Todd C. Mowry. A Scalable Approach to Thread- Level Speculation. In *Proceedings of the 27th Annual International Symposium on Computer Architecture*, pages 1-12, June 2000.
60. Todd C. Mowry and Sherwyn R. Ramkisson. Software-Controlled Multithreading Using Informing Memory Operations. In *Proceedings of the Sixth International Symposium on High-Performance Computer Architecture*, January, 2000.
61. Chi-Keung Luk and Todd C. Mowry. Memory Forwarding: Enabling Aggressive Layout Optimizations by Guaranteeing the Safety of Data Relocation. In *Proceedings of the 26th Annual International Symposium on Computer Architecture (ISCA)*, pages 88-99, May 1999.
62. Chi-Keung Luk and Todd C. Mowry. Cooperative Prefetching: Compiler and Hardware Support for Effective Instruction Prefetching in Modern Microprocessors. In *Proceedings of the 31st Annual International Symposium on Microarchitecture*, pages 182-193, December 1998.
63. J. Gregory Steffan and Todd C. Mowry. The Potential for Using Thread-Level Data Speculation to Facilitate Automatic Parallelization. In *Proceedings of the Fourth International Symposium on High-Performance Computer Architecture*, pages 2-13, February, 1998.
64. Charles Chan, Adley Lo, and Todd C. Mowry. Comparative Evaluation of Latency Tolerance Techniques for Software Distributed Shared Memory. In *Proceedings of the Fourth International Symposium on High-Performance Computer Architecture*, pages 300-311, February, 1998.
65. Todd C. Mowry and Chi-Keung Luk. Predicting Data Cache Misses in Non-Numeric Applications Through Correlation Profiling. In *Proceedings of the 30th Annual International Symposium on Microarchitecture*, pages 314-320, December 1997.
66. Todd C. Mowry, Angela K. Demke and Orran Krieger. Automatic Compiler-Inserted I/O Prefetching for Out-of-Core Applications. In *Proceedings of the Second Symposium on Operating Systems Design and Implementation (OSDI '96)*, pages 3-17, October 1996. (*This paper received the Best Paper Award.*)
67. Chi-Keung Luk and Todd C. Mowry. Compiler-Based Prefetching for Recursive Data Structures. In *Proceedings of the Seventh International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 222-233, October 1996.
68. Edouard Bugnion, Jennifer M. Anderson, Todd C. Mowry, Mendel Rosenblum and Monica S. Lam. Compiler- Directed Page Coloring for Multiprocessors. In *Proceedings of the Seventh International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 244-255, October 1996.
69. Mark Horowitz, Margaret Martonosi, Todd C. Mowry, and Michael D. Smith. Informing Memory Operations: Providing Memory Performance Feedback in Modern Processors. In *Proceedings of the 23rd Annual International Symposium on Computer Architecture*, pages 260-270, May 1996.
70. Todd C. Mowry, Monica S. Lam and Anoop Gupta. Design and Evaluation of a Compiler Algorithm for Prefetching. In *Proceedings of the Fifth International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 62-73, October 1992.

71. Anoop Gupta, John Hennessy, Kourosh Gharachorloo, Todd Mowry, and Wolf-Dietrich Weber. Comparative Evaluation of Latency Reducing and Tolerating Techniques. In *Proceedings of the 18th Annual International Symposium on Computer Architecture*, pages 254-263, May 1991.
72. Anoop Gupta, Wolf-Dietrich Weber, and Todd Mowry. Reducing Memory and Traffic Requirements for Scalable Directory-Based Cache Coherence Schemes. In *Proceedings of International Conference on Parallel Processing*, pages 312-321, August 1990.

Technical Reports

- Evangelos Vlachos, Michelle Goodstein, Michael Kozuch, Shimin Chen, Babak Falsafi, Phillip B. Gibbons, Todd C. Mowry, and Olatunji Ruwase. Parallel LBA: Conference-based Parallel Monitoring of Multithreaded Applications. Carnegie Mellon University Technical Report CMU-CS-09-108, March 2009.
- Michelle Goodstein, Evangelos Vlachos, Shimin Chen, Phillip Gibbons, Michael Kozuch, and Todd C. Mowry. The Butterfly Model: Theoretical Foundations. Carnegie Mellon University Technical Report CMU-CS-08-170, February 2009.
- Amit Manjhi, Anastassia Ailamaki, Bruce M. Maggs, Todd C. Mowry, Christopher Olston, and Anthony Tomasic. Simultaneous Scalability and Security for Data-Intensive Web Applications. Carnegie Mellon University Technical Report CMU-CS-06-116, March 2006.
- Charles Garrod, Amit Manjhi, Anastassia Ailamaki, Phillip Gibbons, Bruce Maggs, Todd Mowry, Christopher Olston, and Anthony Tomasic. Scalable Consistency Management for Web Database Caches. Carnegie Mellon University Technical Report CMU-CS-06-128, July 2006.
- Amit Manjhi, Phillip B. Gibbons, Anastassia Ailamaki, Charles Garrod, Bruce M. Maggs, Todd C. Mowry, Christopher Olston, Anthony Tomasic, and Haifeng Yu. Invalidation Clues for Database Scalability Services. Carnegie Mellon University Technical Report CMU-CS-06-139, July 2006.
- Christopher B. Colohan, Anastassia Ailamaki, J. Gregory Steffan, and Todd C. Mowry. Supporting Large Speculative Threads for Databases and Beyond. Carnegie Mellon University Technical Report CMU-CS-05-109, July 2005.
- Christopher B. Colohan, Anastassia Ailamaki, J. Gregory Steffan, and Todd C. Mowry. Optimistic Intra-Transaction Parallelism on Chip Multiprocessors. Carnegie Mellon University Technical Report CMU-CS-05-118, March 2005.
- Shimin Chen, Anastassia Ailamaki, Phillip B. Gibbons, and Todd C. Mowry. Improving Hash Join Performance through Prefetching. Carnegie Mellon University Technical Report CMU-CS-03-157, October 2003.
- Shimin Chen, Phillip B. Gibbons, Todd C. Mowry, and Gary Valentin. Fractal Prefetching B+-Trees: Optimizing Both Cache and Disk Performance. Carnegie Mellon University Technical Report CMU-CS-02-115, March 2002.
- Spiros Papadimitriou and Todd C. Mowry. Exploring Thread-Level Speculation in Software: The Effects of Memory Access Tracking Granularity. Carnegie Mellon University Technical Report CMU-CS-01-145, July 2001.
- Shimin Chen, Phillip B. Gibbons and Todd C. Mowry. Improving Index Performance through Prefetching. Carnegie Mellon University Technical Report CMU-CS-00-177, December 2000.
- J. Gregory Steffan, Christopher B. Colohan and Todd C. Mowry. Extending Cache Coherence to Support Thread-Level Data Speculation on a Single Chip and Beyond. Carnegie Mellon University Technical Report CMU-CS-98-171, December 1998.
- Todd C. Mowry and Sherwyn R. Ramkisson. Software-Controlled Multithreading Using Informing Memory Operations. Carnegie Mellon University Technical Report CMU-CS-98-169, October 1998.

- Chi-Keung Luk and Todd C. Mowry. Compiler and Hardware Support for Automatic Instruction Prefetching: A Cooperative Approach. Carnegie Mellon University Technical Report CMU-CS-98-140, June 1998.
- J. Gregory Steffan, Christopher B. Colohan and Todd C. Mowry. Architectural Support for Thread-Level Data Speculation. Carnegie Mellon University Technical Report CMU-CS-97-188, November 1997.
- Todd C. Mowry and Chi-Keung Luk. Predicting Data Cache Misses in Non-Numeric Applications Through Correlation Profiling. Carnegie Mellon University Technical Report CMU-CS-97-175, September 1997.
- J. Gregory Steffan and Todd C. Mowry. The Potential for Thread-Level Data Speculation in Tightly-Coupled Multiprocessors. University of Toronto Technical Report CSRI-TR-350, February 1997.
- Chi-Keung Luk and Todd C. Mowry. Predicting Data Cache Misses in Non-Numeric Applications Through Correlation Profiling. University of Toronto Technical Report CSRI-TR-359, February, 1997.
- Mark Horowitz, Margaret Martonosi, Todd C. Mowry, and Michael D. Smith. Informing Loads: Enabling Software to Observe and React to Memory Behavior. Stanford University Technical Report CSL-TR-95-673, July 1995.
- Todd C. Mowry. Tolerating Latency Through Software-Controlled Data Prefetching. Technical Report CSL-TR-94-628, Stanford University, June 1994.

Patents Held

- Todd C. Mowry. U.S. Patent 7,127,586: Prefetching hints. Issued October, 2006.
- Shimin Chen, Phillip B. Gibbons, and Todd C. Mowry. U.S. Patent 6,772,179: System and method for improving index performance through prefetching. Issued August, 2004.
- Todd C. Mowry. U.S. Patent 6,240,488: Prefetching hints. Issued May, 2001.
- Todd C. Mowry. U.S. Patent 5,732,242: Consistently specifying way destinations through prefetching hints. Issued March, 1998.
- Todd C. Mowry and Earl A. Killian. U.S. Patent 5,696,958: Method and apparatus for reducing delays following the execution of a branch instruction in an instruction pipeline. Issued December, 1997.

Distinguished Lectures

- 9/25/13: *“The Log-Based Architectures Project: Exploiting Multicore to Help Parallel Software Behave Correctly”*, University of Wisconsin-Madison, Department of Computer Science, Distinguished Lecture Series.
- 2/7/08: *“Pario: the Next Step Beyond Audio and Video”*, University of Toronto, Department of Electrical and Computer Engineering Distinguished Lecture Series.

Professional Activities

Professional Societies

- Member of the Institute of Electrical and Electronics Engineers (IEEE)
- Member of the Association of Computing Machinery (ACM)

Industrial Employment

5/04–6/07: Intel Corporation, Pittsburgh, Pennsylvania: Director of the Intel Research Pittsburgh Lab.

6/89–11/93: Silicon Graphics, Inc., Mountain View, California (formerly MIPS Computer Systems, Sunnyvale California): Computer Architect (part-time).

Consulting

7/07–2/11: Intel Corporation, Pittsburgh, Pennsylvania: Research Advisor.

8/96–4/04: SandCraft, Inc., Santa Clara, California: Member of the Technical Advisory Board.

1/96–4/04: IBM, Toronto: Visiting Scientist.

12/93–12/96: Silicon Graphics, Inc., Mountain View, California: Computer Architecture Consultant.

University Committee Work and Other Service Activities

Committee Work

- Faculty Hiring Committees:
 - Computer Science Department: 2008, 2010.
 - Computer Systems Area: 2000, 2001, 2002, 2003, 2004.
- Chair of the Computer Science Department Head Selection Committee, 2007.
- CSD Ph.D. Admissions Committee, 2001, 2002, 2003.
- School of Computer Science Council, 2001-2004.
- Carnegie Mellon University Faculty Senate (Presidential Appointee), 2000-2001.
- Computer Systems Area Advocate, 2000-2004.
- Doctoral Review Committee, 1998-2004.
- ACM Thesis Award Nomination Committee, 1999.

Journal, Conference and Workshop Organization

- Editor-in-Chief, ACM Transactions on Computer Systems, 2013-2018.
- Associate Editor, ACM Transactions on Computer Systems, 2001-2013.
- Program Chair of the International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS), 2011.
- Sponsorship Chair, International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS), 2010.
- Member of the ISCA program committee, 1998, 2000, and 2011.
- Member of the ASPLOS program committee, 2000, 2004.
- Member of the ASPLOS external review committee, 2012.
- Co-Program Chair of the International Conference on Parallel Architectures and Compilation Techniques (PACT), 2001.
- Member of the International Symposium on Microarchitecture program committee, 1998.
- Member of the Workshop on Architectural and System Support for Improving Software Dependability (ASID) program committee, 2005.
- Member of the Workshop on Memory System Performance (MSP) program committee (in conjunction with PLDI), 2002 and 2004.
- Member of the IBM CASCON program committee, 1995, 1996, 1997.
- Member of the First SUIF Workshop Program Committee, 1996.
- Organized the CASCON 97 workshop on Compiler Optimization, 1997.
- Initiated and organized the 1st Toronto / Rochester Workshop on Shared-Memory Multiprocessors, 1994.