IN THE UNITED STATES DISTRICT COURT
FOR THE WESTERN DISTRICT OF TEXAS

EXPRESS MOBILE, INC., Plaintiff,	Civil Action No. 6:20-cv-00805-ADA
v.	JURY TRIAL DEMANDED
ATLASSIAN CORP. PLC and ATLASSIAN, INC., Defendants.	

FIRST AMENDED COMPLAINT FOR PATENT INFRINGEMENT

Plaintiff Express Mobile, Inc. ("Express Mobile" or "Plaintiff"), for its complaint against Defendants Atlassian Corp. plc and Atlassian, Inc. (collectively, "Atlassian" or "Defendants"), alleges the following:

NATURE OF THE ACTION

1. This is an action for patent infringement arising under the patent laws of the United States, 35 U.S.C. §§ 1 *et seq*.

THE PARTIES

2. Express Mobile is a corporation organized under the laws of the State of Delaware with a place of business at 38 Washington Street, Novato, CA 94947.

3. Atlassian Corp. plc is an Australian corporation, and Atlassian, Inc., is a corporation organized under the laws of the State of Delaware, with a place of business at 303 Colorado Street, Suite 1600, Austin, TX 78701. Atlassian, Inc., can be served through its registered agent in Texas, Corporation Service Company, d/b/a CSC-Lawyers Inc., 211 E. 7th Street, Suite 620, Austin, TX 78701.

4. Atlassian offers services throughout the United States, including in this judicial District, and introduces services into the stream of commerce that incorporate infringing

technology knowing that they would be used in this judicial District and elsewhere in the United States.

JURISDICTION AND VENUE

5. This Court has subject matter jurisdiction under 28 U.S.C. §§ 1331 and 1338(a).

6. This Court has personal jurisdiction over Atlassian because it has purposefully availed itself of the rights and benefits of the laws of this State and this District. Atlassian resides in the Western District of Texas by maintaining a regular and established place of business at 303 Colorado Street, Suite 1600, Austin, TX 78701. This Court also has personal jurisdiction over Atlassian because it has done and is doing substantial business in this District, both generally and with respect to the allegations in this complaint, including Atlassian's one or more acts of infringement in this District.

7. Venue is proper in this District under 28 U.S.C. §§ 1391(b)-(c) and 1400(b). Atlassian has committed acts of infringement through provision of its website builder in the Western District of Texas and has at least one regular and established place of business in this District, specifically 303 Colorado Street, Suite 1600, Austin, TX 78701. Atlassian's office in Austin is a physical place in the District, it is an established location where Atlassian's business has been carried out for years, and Atlassian publicly advertises its presence in the District. *See In re Cray, Inc.*, 871 F.3d 1355, 1360-61 (Fed. Cir. 2017).

BACKGROUND

8. Plaintiff Express Mobile is an innovator and leader in the business of developing mobile application and website design and creation platforms. Express Mobile is managed by individuals with many years of technology and business experience. The CEO of Express Mobile, Steve Rempell, is the inventor of the breakthrough technology held in Express Mobile's

patent portfolio. Mr. Rempell has over 50 years' working experience in technology companies, with much of that experience focused on web-based technologies and applications.

9. Before the Express Mobile invention at issue, webpages were created, stored, and rendered using code files that defined all the fixed parameters of the webpage, including, for example, the formatting and location of text, or the location, size, and aspect ratio of images. Typically, webpages could not be viewed during the creation process as they would later appear in the various available browsers or on different devices, and each individual webpage of a website needed to be stored as a separate file. The size and formatting of the stored files led to slow download times to the user's computer, increasing the wait time for a page to load.

10. Express Mobile developed groundbreaking improvements in the process for creating, storing, and building webpages and websites. Express Mobile's invention enables defining the webpage as a collection of user settings, storing information related to those settings in a database, and then later using that information to render a webpage. The page can be viewed, as it is created or edited, in the same manner that it would appear on different types of screens when later accessed. The result is not a collection of computer code, but instead a group of user-selected objects and settings describing the final webpage. These can be saved in a database for ease of access and efficient storage. The invention allows faster loading speeds and permits more efficient storage of the data used to later build the webpages. It also makes changing the webpage more efficient through editing user settings rather than editing multiple lines or versions of code.

11. Defendant Atlassian is a leader in enterprise product and content management. Atlassian offers productivity tools for both small and large businesses. One product, Confluence, is a collaborative information system for large organizations. It allows users without technical training to create webpages and other content easily and to share them across the organization.

See https://www.atlassian.com/software/confluence. Another tool, Trello, is a project management tool that allows enterprise teams to track work items and deadlines by visualizing them as "cards" on a "board." *See* https://trello.com. Atlassian also offers a project management tool called JIRA, which is widely used by large companies to track issues (such as bugs or feature requests) in software projects. *See* https://www.atlassian.com/software/jira.

12. Confluence allows users to create webpages for people within their organization using menus to control the appearance and content of the final page. Selection of menu items updates the preview of the final page. Trello and JIRA offer the same features – users can share webpages relating to projects with others and use menus of settings to control the appearance of content, such as cards in Trello and items in JIRA. When users select a setting, those products display the results in real time.

13. Atlassian's products store information about the selected settings in a database. When a user titles a "To Do" item as "Task 1" in Trello, for example, data regarding that setting, including the chosen title, is sent to Atlassian's servers, where it is stored in a database for later retrieval. Confluence and JIRA operate similarly.

14. When users access a Trello, Confluence, or JIRA page, the information regarding the settings for the page is retrieved from the database. And JavaScript run time files inspect that retrieved information to generate and display the final webpage according to the settings.

COUNT I – INFRINGEMENT OF U.S. PATENT NO. 6,546,397 (Confluence)

15. The allegations set forth in the foregoing paragraphs 1 through 14 are incorporated into this First Claim for Relief.

16. On April 8, 2003, U.S. Patent No. 6,546,397 ("the '397 patent"), entitled *Browser Based Web Site Generation Tool and Run Time Engine*, was duly and legally issued by the

United States Patent and Trademark Office. A true and correct copy of the ’397 patent is attached as Exhibit A.

17. The claimed invention of the ’397 patent resolves technical problems related to website creation and generation. Prior to the invention taught and disclosed in the ’397 patent, webpages were generally created, stored, and rendered either by programming directly in HTML, CSS,[1] or JavaScript code, or by using a visual editor that produced HTML files. The result was a collection of pages of computer code – typically HTML, CSS, JavaScript, or Java applets – which defined the visual layout, style, and business logic of websites.

18. Conventional website creation and generation methods suffered from many flaws. Creating a webpage could be cumbersome. Webpages could not be viewed throughout the creation process as they would later appear in various browsers or on different devices. Each individual webpage of a website was stored as a separate HTML, CSS, or JavaScript file, which wasted computer resources and required longer access times in the form of hard drive access while editing websites, and in the form of network traffic while downloading them. Prior-art methods also led to slow downloading of the webpage file to a user’s computer and slower rendering by the browser, which increased the wait time for a page to load.

19. Unlike prior-art methods, the ’397 patent brings together disparate ideas and concepts for creating, storing, and building webpages. The Express Mobile invention at issue defines webpages as combinations of user-selected objects and settings stored in a database, rather than as combinations of computer code. Because code files do not need to be stored, the page structure – the vast majority of the HTML code itself – is created on the fly each time the

[1] CSS, or “Cascading Style Sheets,” is a programming language designed to interoperate with HTML to specify the appearance and placement of web elements.

page is loaded in a user browser. This unconventional step of building the webpage HTML code on the fly is performed by the run time engine of the invention, using data representative of the user settings. This allows the system to optimize the page based on device-specific information, including the operating system, browser, and screen size. Moreover, the process of defining the webpages is done through a "what you see is what you get" environment, so that, as the page is created or edited, it can be viewed in the same manner it will appear on different types of screens when later accessed.

20. Express Mobile's patents are directed at a revolutionary technological solution to a technological problem – how to create webpages for the Internet in a manner that permits "what you see is what you get" editing, and a number of other improvements over the then-existing methodologies. The claims are not drawn so broadly as to be divorced from the patent-eligible technological improvements described in the specification.

21. The invention claimed in the '397 patent is not merely the routine or conventional use of website creation systems and methods. Rather, the invention enables the creation of websites through browser-based visual editing tools such as selectable settings panels that describe website elements, with one or more settings corresponding to commands. The invention also enables retrieving that information to generate a website. Those features are implemented exclusively using computer technology, including using virtual machines.

22. The invention claimed in the '397 patent offers substantial improvements in computer performance and web design. For example, the invention allows for faster loading speeds, more efficient storage of webpage data, and the ability to change the webpage more efficiently by editing user settings rather than multiple versions of code. The invention also permits scaling of webpages and elements within the webpage to most efficiently use the screen

space. Taken separately or together, the claim elements of the invention significantly improve the operation of a computer and the process of web design.

23. The claims of the ’397 patent do not merely recite the performance of some business practice known from the pre-Internet world along with the requirement to perform it on the Internet. Instead, the claims of the ’397 patent recite one or more inventive concepts that are rooted in computerized website creation technology, and overcome problems specifically arising in the realm of computerized website creation technologies.

24. The invention claimed in the ’397 patent neither preempts all ways of using website or webpage authoring tools nor preempts the use of all website or webpage authoring tools or any other well-known or prior-art technology. Accordingly, each claim of the ’397 patent recites a combination of elements sufficient to ensure that the claim in practice amounts to significantly more than a patent on an ineligible concept.

25. Express Mobile is the assignee and owner of the right, title, and interest in and to the ’397 patent, including the right to assert all causes of action arising under that patent and the right to any remedies for infringement of it.

26. Atlassian has manufactured, used, offered for sale, or sold browser-based website building tools that infringed, either literally or under the doctrine of equivalents, the ’397 patent in violation of 35 U.S.C. § 271(a).

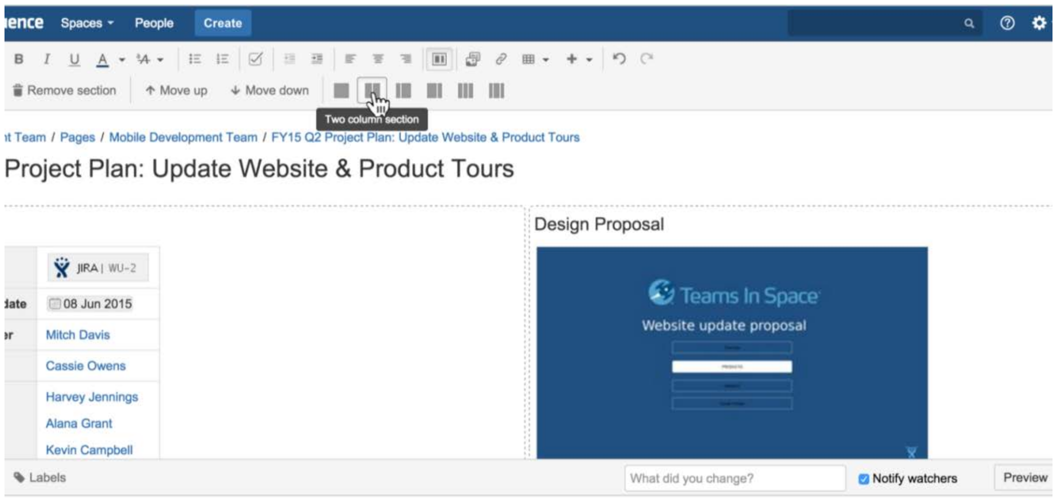
27. Upon information and belief, Atlassian has infringed at least claim 1 of the ’397 patent.

28. Claim 1 of the ’397 patent recites a method to allow users to produce Internet websites on and for computers having a browser and a virtual machine capable of generating displays, said method comprising: (a) presenting a viewable menu having a user-selectable panel of settings describing elements on a website, said panel of settings being presented through a

browser on a computer adapted to accept one or more of said selectable settings in said panel as inputs therefrom, and where at least one of said user-selectable settings in said panel corresponds to commands to said virtual machine; (b) generating a display in accordance with one or more user-selected settings substantially contemporaneously with the selection thereof; (c) storing information representative of said one or more user-selected settings in a database; (d) generating a website at least in part by retrieving said information representative of said one or more user-selected settings stored in said database; (e) building one or more webpages to generate said website from at least a portion of said database and at least one run time file, where said at least one run time file uses information stored in said database to generate virtual machine commands for the display of at least a portion of said one or more webpages.

29. Atlassian infringed at least claim 1 of the ’397 patent. During the relevant time periods, Confluence practiced a method to allow users to produce Internet webpages on and for computers having a browser and a virtual machine capable of generating displays.

30. As shown in the image below, Confluence presented through the user’s web browser a viewable menu with a user-selectable panel of settings. By way of example, the menu included settings to change font, text size, text color, and paragraph justification.



Project Plan: Update Website & Product Tours

Design Proposal

	JIRA | WU-2
date	08 Jun 2015
er	Mitch Davis
	Cassie Owens
	Harvey Jennings
	Alana Grant
	Kevin Campbell

Teams In Space
Website update proposal

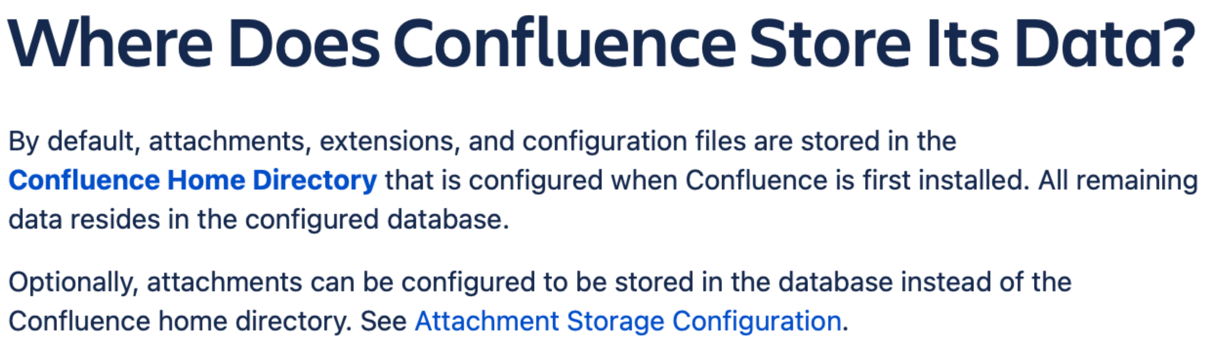
31. By way of further example, Confluence presented a viewable menu that allowed users to specify which elements they wanted to appear on their pages. For example, Confluence's visual page editor allowed users to insert hyperlinks, images, and tables on web pages. Additional viewable menus controlled user-selectable settings controlling the appearance of individual web elements on the page. For example, a user could select how big they wanted an image to be.

32. By way of further example, Confluence presented a viewable menu displaying user-selectable settings that allowed users to specify the design of the pages, including settings that corresponded to subsections of their webpage, as well as the font size, highlights, and effects of the text on the pages. By way of further example, Confluence allowed users to click and move elements appearing on the pages, placing them in their desired locations.

33. The Confluence page editor operated as a "what you see is what you get" environment. In other words, once a user selected settings from any of those menus, Confluence updated the pages in accordance with the selected settings substantially contemporaneously with the selection thereof.

34. As noted on Atlassian's website, Confluence stored user-selected settings in a database hosted on servers operated by Atlassian.[2]

[2] *See* Confluence Support, *Where Does Confluence Store Its Data?* (Jan. 7, 2021), https://confluence.atlassian.com/confkb/where-does-confluence-store-its-data-154029.html.



Where Does Confluence Store Its Data?

By default, attachments, extensions, and configuration files are stored in the **Confluence Home Directory** that is configured when Confluence is first installed. All remaining data resides in the configured database.

Optionally, attachments can be configured to be stored in the database instead of the Confluence home directory. See Attachment Storage Configuration.

35. Those user-selectable settings corresponded to commands to the JavaScript run time engine operating within the user's web browser – a virtual machine. When a setting was selected, Confluence used it to generate settings encoded in JavaScript Object Notation (JSON). One or more run time files containing HTML and JavaScript code communicated with the Confluence web server to send and retrieve the encoded user-selected settings and thus generate the page.

36. By way of example, when a user first loaded Confluence, the Confluence web server sent the run time files to the user's web browser. The run time files then communicated with the Confluence web server to retrieve the user-selectable settings stored in the database – for example, fonts, text effects, and other settings of the elements – and used them to generate virtual machine commands in the form of JSON code, which was sent to the web browser. The web browser's virtual machine executed the JSON code. That execution, in combination with the commands in the run time files, generated the page in accordance with the saved settings.

37. Atlassian's infringement damaged and injured Express Mobile.

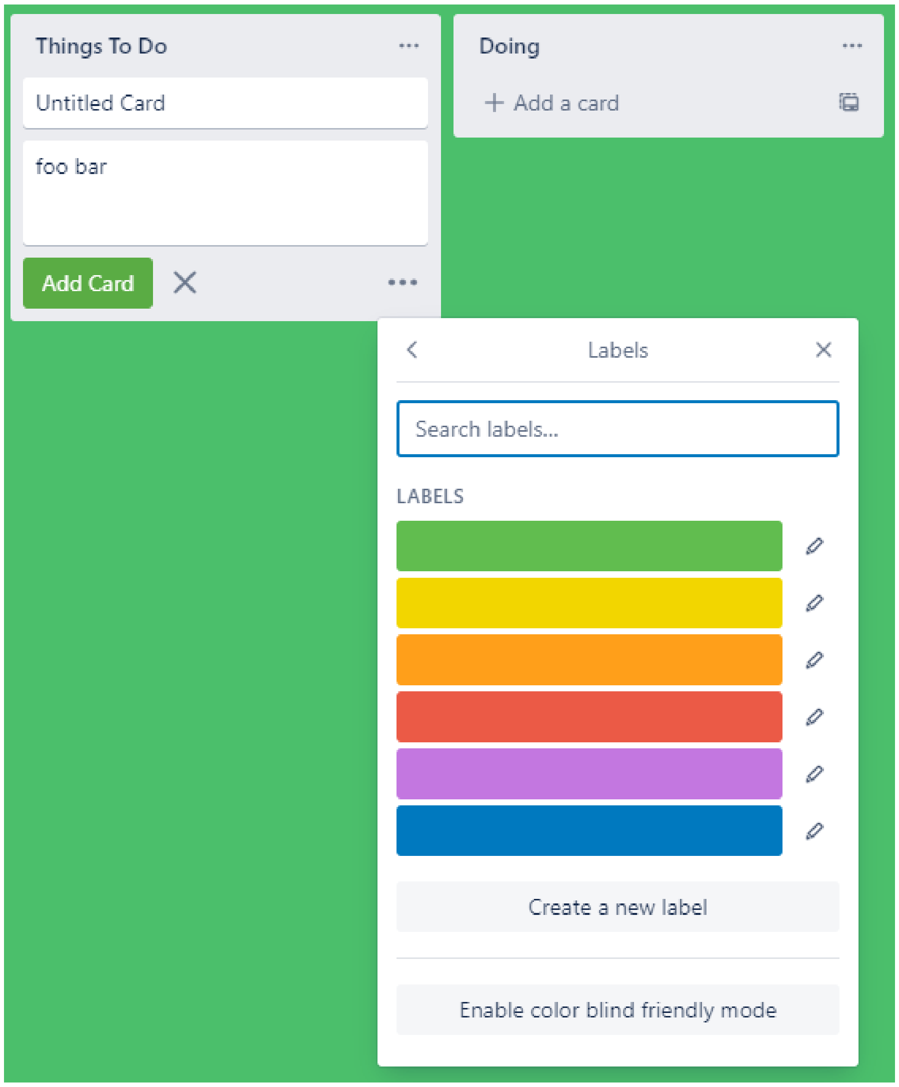
<u>COUNT II – INFRINGEMENT OF U.S. PATENT NO. 6,546,397</u>
<u>(Trello)</u>

38. The allegations set forth in the foregoing paragraphs 1 through 37 are incorporated into this Second Claim for Relief.

39. Atlassian has manufactured, used, offered for sale, or sold browser-based website building tools that infringed, either literally or under the doctrine of equivalents, one or more claims of the ’397 patent in violation of 35 U.S.C. § 271(a).

40. Upon information and belief, Atlassian infringed at least claim 1 of the ’397 patent. During the relevant time periods, Trello practiced a method to allow users to produce Internet webpages on and for computers having a browser and a virtual machine capable of generating displays.

41. As shown below, Trello presented a viewable menu displaying user-selectable settings that allowed users to specify which elements they wanted to appear on their Trello pages. Those elements were organized into columns of “cards” containing text, images, and icons. Additional viewable menus controlled user-selectable settings controlling the appearance of individual elements on the cards. For example, as shown below, a user could add a colored label to selected cards. By way of further example, a user could add images and icons to cards.



42. By way of further example, Trello presented a viewable menu displaying user-selectable settings that allowed users to specify the design of the pages, including settings that corresponded to subsections of the cards, as well as the font size, highlights, and effects of the text on the cards. By way of further example, Trello allowed users to click and move cards on the pages, placing them in different locations.

43. Once a user selected any of these settings, Trello updated and displayed the pages in accordance with the selected settings substantially contemporaneously with the selection

thereof. For example, when a user selected the color of a label on a card, the display of the card was updated substantially contemporaneously.

44. Upon information and belief, Trello stored user-selected settings in a database hosted on servers operated by Atlassian. According to a blog developed by the engineering team behind Trello, Trello stored user-selected settings in a "MongoDB" database. Barry Clark, *The Trello Tech Stack 2016*, https://tech.trello.com/the-trello-tech-stack ("MongoDB is our 'real' database.").

45. Those user-selectable settings corresponded to commands to the JavaScript run time engine operating within the user's web browser – a virtual machine. When a setting was selected, Trello used it to generate settings encoded in JSON. One or more run time files containing HTML and JavaScript code communicated with the Trello web server to send and retrieve the encoded user-selected settings and thus generate the page.

46. By way of example, when a user first loaded Trello, the Trello web server sent the run time files to the user's web browser. The run time files then communicated with the Trello web server to retrieve the user-selectable settings stored in the database – for example, the placement of cards and the text and labels associated with cards – and used them to generate virtual machine commands in the form of JSON code, which was sent to the web browser. The web browser's virtual machine executed the JSON code. That execution, in combination with the commands in the run time files, generated the page in accordance with the saved settings.

47. Atlassian's infringement damaged and injured Express Mobile.

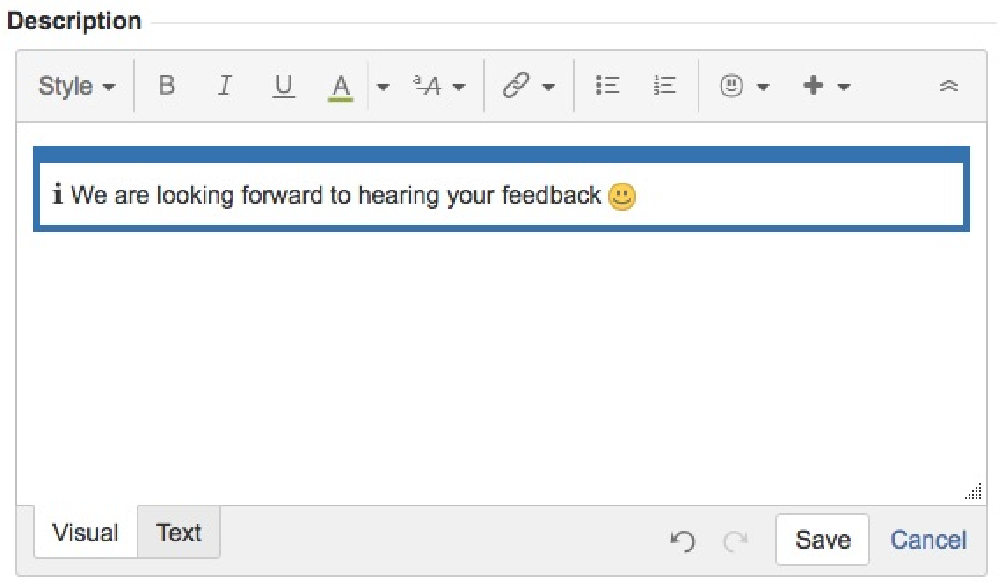
<u>COUNT III – INFRINGEMENT OF U.S. PATENT NO. 6,546,397</u> <u>(JIRA)</u>

48. The allegations set forth in the foregoing paragraphs 1 through 47 are incorporated into this Third Claim for Relief.

49. Atlassian has manufactured, used, offered for sale, or sold browser-based website building tools that infringed, either literally or under the doctrine of equivalents, one or more claims of the ’397 patent in violation of 35 U.S.C. § 271(a).

50. Upon information and belief, Atlassian infringed at least claim 1 of the ’397 patent. During the relevant time periods, JIRA practiced a method to allow users to produce Internet webpages, those pages representing tracked bug reports, tasks, or issues, on and for computers having a browser and a virtual machine capable of generating displays.

51. As shown in the image below, JIRA presented a viewable menu displaying user-selectable settings for editing JIRA issues. The viewable menu included user-selectable settings for style, text-bolding, italics, underlining, bullet formatting, and list formatting.



We are looking forward to hearing your feedback

52. By way of further example, JIRA presented a viewable menu displaying user-selectable settings that allowed users to specify the design of the pages, including settings that corresponded to various fields of a JIRA issue, as well as the font size and other formatting of the text content in an issue.

53. Once a user selected any setting, JIRA updated and displayed the pages in accordance with the selected settings substantially contemporaneously with the selection thereof.

54. Upon information and belief, JIRA stored user-selected settings in a database hosted on servers operated by Atlassian. According to Atlassian's webiste, JIRA stored user-selected settings in an "Azure SQL," "PostgreSQL," "MySQL," "Oracle," or "SQL Server" database. *See* Atlassian, *Connecting JIRA Applications to a Database* (Feb. 26, 2020), https://confluence.atlassian.com/adminjiraserver/connecting-jira-applications-to-a-database-938846850.html.

55. Those user-selectable settings corresponded to commands to the JavaScript run time engine operating within the user's web browser – a virtual machine. When a setting was selected, JIRA used it to generate or modify the attributes for User Interface ("UI") elements from the settings that were encoded in JSON. One or more run time files containing HTML and JavaScript code communicated with the JIRA web server to send and retrieve the encoded user-selected settings and thus generate the page.

56. By way of example, when a user first loaded JIRA, the JIRA web server sent the run time files to the user's web browser. The run time files then communicated with the JIRA web server to retrieve the user-selectable settings stored in the database – for example, the fonts, text effects, and other settings of the elements – and used them to generate virtual machine commands from the settings that were in the form of JSON data, which was sent to the web browser. The web browser's virtual machine executed the HTML and JavaScript code that incorporated the JSON data. That execution, in combination with the commands in the run time files, generated the page in accordance with the saved settings.

57. Atlassian's infringement damaged and injured Express Mobile.

COUNT IV – INFRINGEMENT OF U.S. PATENT NO. 9,063,755 (Confluence)

58. The allegations set forth in the foregoing paragraphs 1 through 57 are incorporated into this Fourth Claim for Relief.

59. On June 23, 2015, U.S. Patent No. 9,063,755 ("the '755 patent"), entitled *Systems and Methods for Presenting Information on Mobile Devices*, was duly and legally issued by the United States Patent and Trademark Office. A true and correct copy of the '755 patent is attached as Exhibit B.

60. The invention claimed in the '755 patent resolves technical problems related to generating and distributing dynamic content on a device display, such as the display on a mobile device. Before the patents-in-suit, content and applications for device displays were generally created using code written for each individual type of device. As device types proliferated, programming content and applications for each device became increasingly expensive and time-consuming. Doing so also limited the ability of providers to update the capabilities of, and increase the available content for, many devices.

61. The invention of the '755 patent resolves technical problems related to generating and distributing dynamic content on a device display. The invention features a computer memory and an authoring tool or Player configured to define a User Interface ("UI") object for display on the device, where the defined UI object corresponds to a web component and where each UI object is either (1) selected by a user or (2) automatically selected by the system as a preferred UI object corresponding to a symbolic name of the web component. Additionally, the computer memory and the authoring tool or Player are configured to build an Application consisting of one or more webpage views to provide for the display of at least a portion of one or more of the webpages. Those features are exclusively implemented using computer technology.

62. Unlike methods in the prior art, the ’755 patent brings together disparate ideas and concepts for generating and distributing content suitable for display on different devices with varying characteristics using a combination of device-independent and device- and platform-dependent code. This can include building a webpage or application using a “what you see is what you get” environment, so that, as the page or app is created or edited, it can be viewed in the same manner it will appear on different types of screens when later accessed. The invention can also include an authoring tool that can create an Application that includes dynamic content, where the Application is device-independent code, and a Player, where the Player is device-dependent code. The Player enables the Application to function on a variety of devices or platforms, with differing functionality. That enables users of the authoring tool to create and distribute device-independent Applications for different device types, without individually tailoring the device-independent dynamic-content Applications for each device type.

63. The claims of the ’755 patent do not merely recite the performance of some business practice known from the pre-Internet world along with the requirement to perform it on the Internet. Instead, the claims of the ’755 patent recite one or more inventive concepts that are rooted in the computerized generation of content on a device display, such as a mobile device, and overcome problems specifically arising in the realm of computerized display content generation technologies.

64. The claims of the ’755 patent recite an invention that is not merely the routine or conventional use of systems and methods for the computerized generation of content on a device display. Instead, the invention describes systems for use with devices with authoring tools or Players specific to each device and Applications that are independent of the device.

65. The invention claimed in the ’755 patent offers substantial improvements in device performance and web or application design. For example, the invention allows for faster

loading speeds, more efficient and flexible processing of webpage or application dynamic content, and the ability to change a webpage or application more efficiently by editing user settings rather than multiple versions of code. The invention also permits scaling of webpages and dynamic content within the webpage, or applications and elements within the application that contain dynamic content, to most efficiently use the screen space. Taken separately or together, the claim elements of the invention significantly improve the operation of a computer and the process of web design.

66. The invention claimed in the ’755 patent neither preempts all ways for the computerized generation of content on a device display, such as a mobile device, nor preempts the use of all authoring tools or Players for the computerized generation of content on a device display, such as a mobile device, or any other well-known or prior-art technology.

67. Accordingly, each claim of the ’755 patent recites a combination of elements sufficient to ensure that the claim in practice amounts to significantly more than a patent on an ineligible concept.

68. Express Mobile is the assignee and owner of the right, title, and interest in and to the ’755 patent, including the right to assert all causes of action arising under the patent and the right to any remedies for infringement of it.

69. Atlassian has manufactured, used, offered for sale, or sold browser-based website building tools that infringe, either literally or under the doctrine of equivalents, one or more claims of the ’755 patent in violation of 35 U.S.C. § 271(a).

70. Upon information and belief, Atlassian has infringed and continues to infringe at least claim 23 of the ’755 patent.

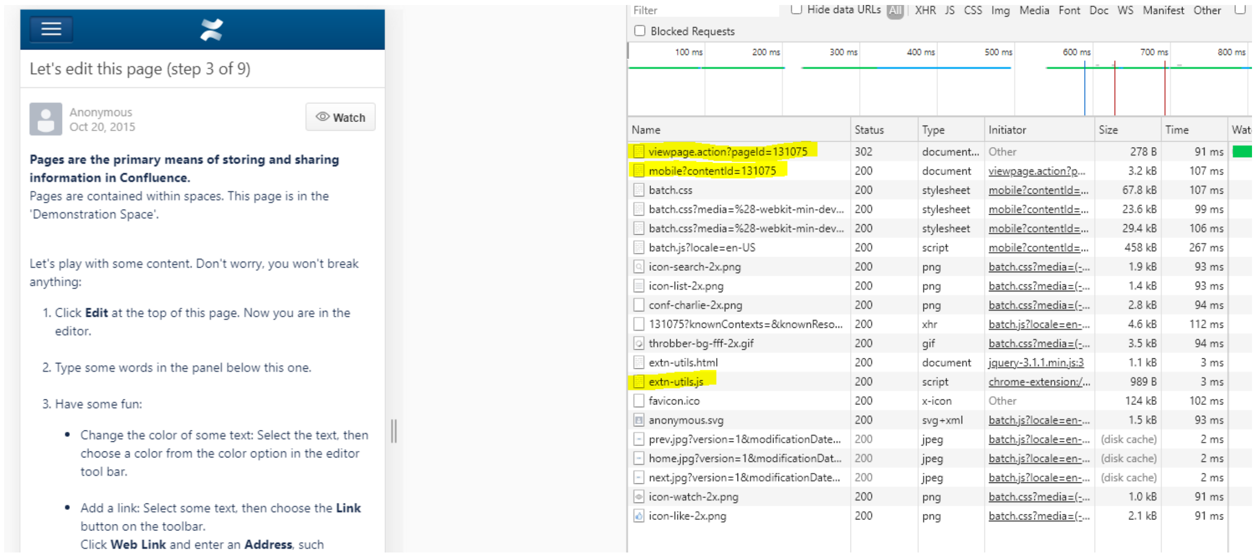
71. Claim 23 of the ’755 patent recites a method of providing information to a device having a display from a web component of a web service to a device on a network, said method

comprising: accepting, on the device, a first code over the network, where said first code is device-dependent; accepting, on the device, a second code over the network, where said second code is device-independent and includes a plurality of symbolic names of inputs and outputs associated with the web service; and executing said first code on the device, where the symbolic names are provided from a registry of one or more web components related to inputs and outputs of a web service obtainable over a network, where the web service requires both an input symbolic name and one or more associated input values and returns one or more output values having an associated output symbolic name, and where the registry includes (a) symbolic names required for evoking one or more web components each related to a set of inputs and outputs of a web service obtainable over a network, where the symbolic names are character strings that do not contain either a persistent address or pointer to an output value accessible to the web service, and (b) the address of the web service; where said executing includes: processing said symbolic names of the second code on the device, transmitting processed instructions from the device to the web service, and accepting a third code on the device over the network, where said third code is a device-independent third code including the output of the web component provided by the web service over the network and in response to the second code.

72. Atlassian infringes claim 23 of the ’755 patent by practicing each limitation of claim 23. Confluence practices a method for providing information to a device platform having a display, including a web browser. The information comes from a web service, the Confluence web server.

73. Confluence accepts and executes code from the Confluence web server, including HTML, CSS, and JavaScript. Some of the code is written for specific device platforms and devices, such as browsers, laptops, or smartphones (“device-platform-dependent code”), while some operates identically on all device platforms and devices (“device-independent code”). As

shown in the images below, and solely by way of example, Confluence accepted on the device a device-platform-dependent first code and device-independent second code, highlighted. The device-platform-dependent code is different for mobile devices (first image) versus desktop devices (second image).



Let's edit this page (step 3 of 9)

Anonymous
Oct 20, 2015

Watch

Pages are the primary means of storing and sharing information in Confluence.
Pages are contained within spaces. This page is in the 'Demonstration Space'.

Let's play with some content. Don't worry, you won't break anything:

1. Click **Edit** at the top of this page. Now you are in the editor.

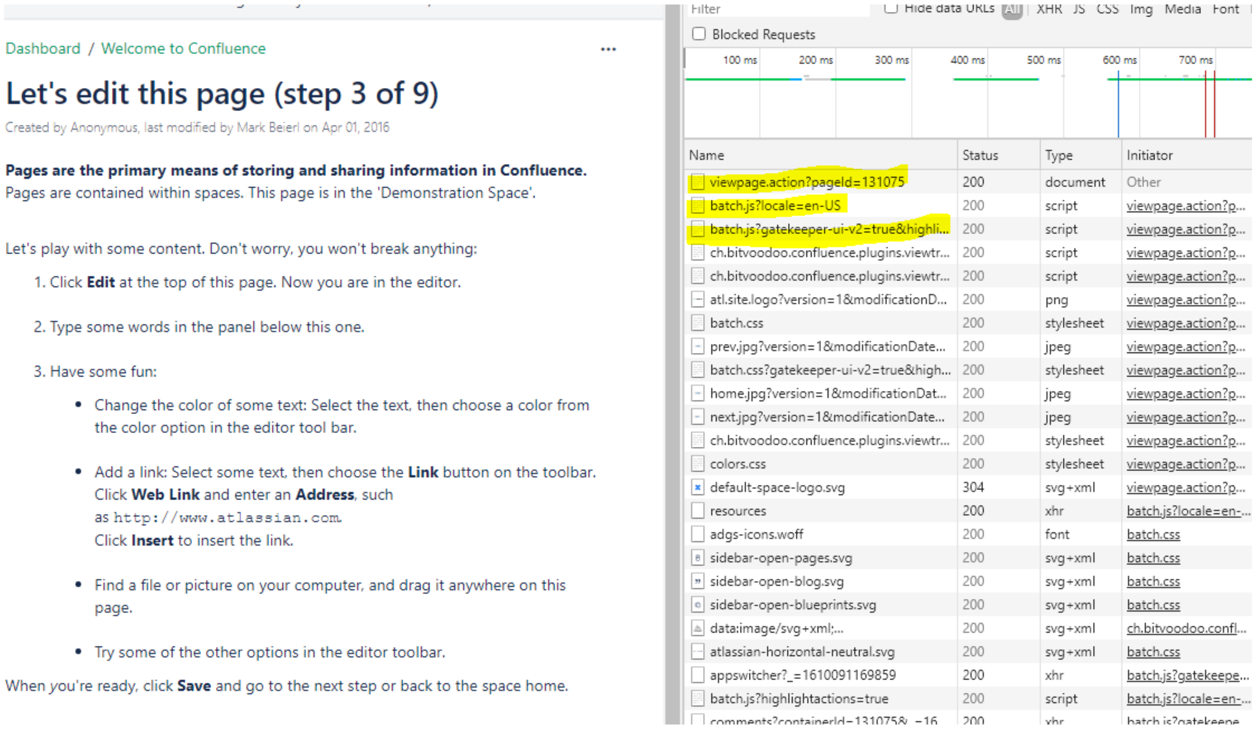
2. Type some words in the panel below this one.

3. Have some fun:

   - Change the color of some text: Select the text, then choose a color from the color option in the editor tool bar.

   - Add a link: Select some text, then choose the **Link** button on the toolbar.
     Click **Web Link** and enter an **Address**, such

Name	Status	Type	Initiator	Size	Time	Wat
viewpage.action?pageId=131075	302	document...	Other	278 B	91 ms	🟩
mobile?contentId=131075	200	document	viewpage.action?p...	3.2 kB	107 ms	
batch.css	200	stylesheet	mobile?contentId=...	67.8 kB	107 ms	
batch.css?media=%28-webkit-min-dev...	200	stylesheet	mobile?contentId=...	23.6 kB	99 ms	
batch.css?media=%28-webkit-min-dev...	200	stylesheet	mobile?contentId=...	29.4 kB	106 ms	
batch.js?locale=en-US	200	script	mobile?contentId=...	458 kB	267 ms	
icon-search-2x.png	200	png	batch.css?media=(-...	1.9 kB	93 ms	
icon-list-2x.png	200	png	batch.css?media=(-...	1.4 kB	93 ms	
conf-charlie-2x.png	200	png	batch.css?media=(-...	2.8 kB	94 ms	
131075?knownContexts=&knownReso...	200	xhr	batch.js?locale=en-...	4.6 kB	112 ms	
throbber-bg-fff-2x.gif	200	gif	batch.css?media=(-...	3.5 kB	94 ms	
extn-utils.html	200	document	jquery-3.1.1.min.js:3	1.1 kB	3 ms	
extn-utils.js	200	script	chrome-extension:/...	989 B	3 ms	
favicon.ico	200	x-icon	Other	124 kB	102 ms	
anonymous.svg	200	svg+xml	batch.js?locale=en-...	1.5 kB	93 ms	
prev.jpg?version=1&modificationDate...	200	jpeg	batch.js?locale=en-...	(disk cache)	2 ms	
home.jpg?version=1&modificationDat...	200	jpeg	batch.js?locale=en-...	(disk cache)	2 ms	
next.jpg?version=1&modificationDate...	200	jpeg	batch.js?locale=en-...	(disk cache)	2 ms	
icon-watch-2x.png	200	png	batch.css?media=(-...	1.0 kB	91 ms	
icon-like-2x.png	200	png	batch.css?media=(-...	2.1 kB	91 ms	



Dashboard / Welcome to Confluence

Let's edit this page (step 3 of 9)

Created by Anonymous, last modified by Mark Beierl on Apr 01, 2016

Pages are the primary means of storing and sharing information in Confluence.
Pages are contained within spaces. This page is in the 'Demonstration Space'.

Let's play with some content. Don't worry, you won't break anything:

1. Click **Edit** at the top of this page. Now you are in the editor.

2. Type some words in the panel below this one.

3. Have some fun:

   - Change the color of some text: Select the text, then choose a color from the color option in the editor tool bar.

   - Add a link: Select some text, then choose the **Link** button on the toolbar.
     Click **Web Link** and enter an **Address**, such as `http://www.atlassian.com`.
     Click **Insert** to insert the link.

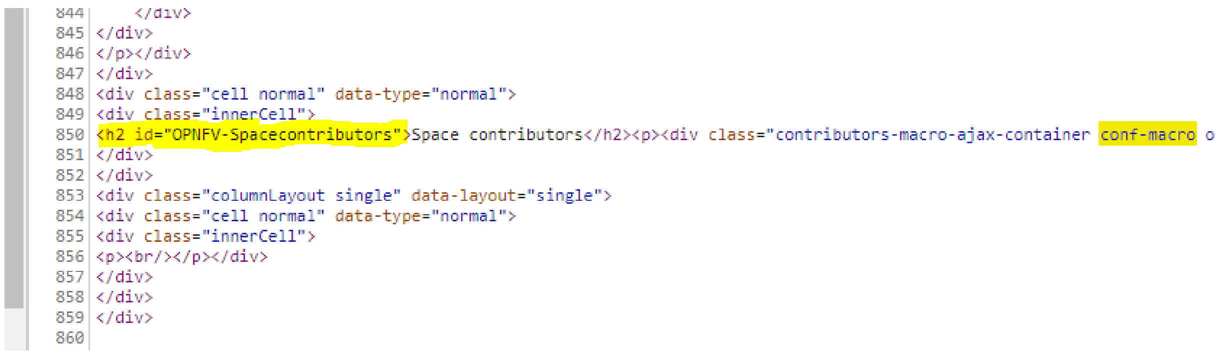
   - Find a file or picture on your computer, and drag it anywhere on this page.

   - Try some of the other options in the editor toolbar.

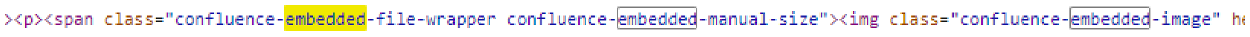
When you're ready, click **Save** and go to the next step or back to the space home.

Name	Status	Type	Initiator
viewpage.action?pageId=131075	200	document	Other
batch.js?locale=en-US	200	script	viewpage.action?p...
batch.js?gatekeeper-ui-v2=true&highli...	200	script	viewpage.action?p...
ch.bitvoodoo.confluence.plugins.viewtr...	200	script	viewpage.action?p...
ch.bitvoodoo.confluence.plugins.viewtr...	200	script	viewpage.action?p...
atl.site.logo?version=1&modificationD...	200	png	viewpage.action?p...
batch.css	200	stylesheet	viewpage.action?p...
prev.jpg?version=1&modificationDate...	200	jpeg	viewpage.action?p...
batch.css?gatekeeper-ui-v2=true&high...	200	stylesheet	viewpage.action?p...
home.jpg?version=1&modificationDat...	200	jpeg	viewpage.action?p...
next.jpg?version=1&modificationDate...	200	jpeg	viewpage.action?p...
ch.bitvoodoo.confluence.plugins.viewtr...	200	stylesheet	viewpage.action?p...
colors.css	200	stylesheet	viewpage.action?p...
default-space-logo.svg	304	svg+xml	viewpage.action?p...
resources	200	xhr	batch.js?locale=en-...
adgs-icons.woff	200	font	batch.css
sidebar-open-pages.svg	200	svg+xml	batch.css
sidebar-open-blog.svg	200	svg+xml	batch.css
sidebar-open-blueprints.svg	200	svg+xml	batch.css
data:image/svg+xml;...	200	svg+xml	ch.bitvoodoo.confl...
atlassian-horizontal-neutral.svg	200	svg+xml	batch.css
appswitcher?_=1610091169859	200	xhr	batch.js?gatekeepe...
batch.js?highlightactions=true	200	script	batch.js?locale=en-...
comments?containerId=131075&_=16	200	xhr	batch.js?gatekeepe

74. The device-independent code includes symbolic names of inputs and outputs. Those symbolic names are assigned to web elements. As shown in the image below, solely by way of example, the symbolic names are associated with web elements through the HTML "class" and "id" attributes (highlighted).



```
    </div>
</div>
</p></div>
</div>
<div class="cell normal" data-type="normal">
<div class="innerCell">
<h2 id="OPNFV-Spacecontributors">Space contributors</h2><p><div class="contributors-macro-ajax-container conf-macro o
</div>
</div>
<div class="columnLayout single" data-layout="single">
<div class="cell normal" data-type="normal">
<div class="innerCell">
<p><br/></p></div>
</div>
</div>
</div>
```



```
><p><span class="confluence-embedded-file-wrapper confluence-embedded-manual-size"><img class="confluence-embedded-image" he
```

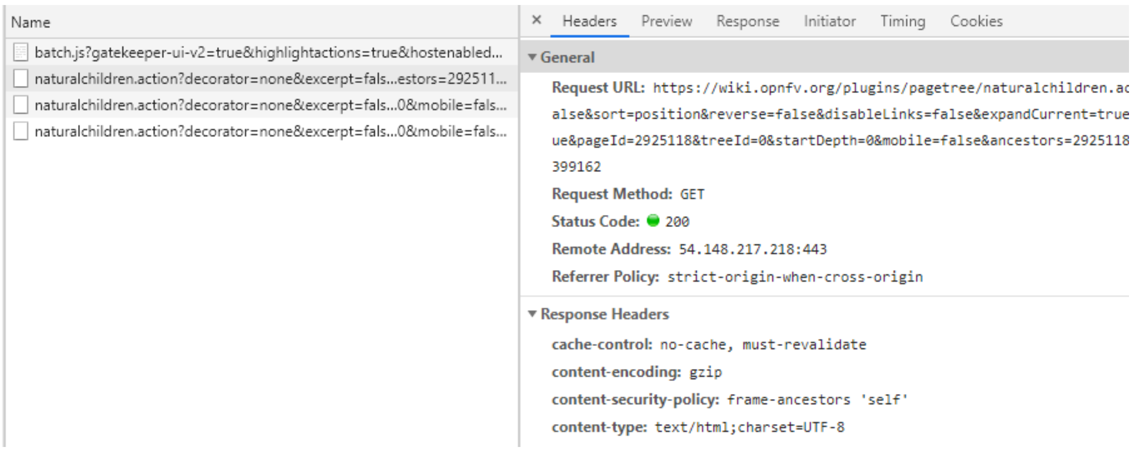
75. The symbolic names are provided from a registry of web components related to inputs and outputs obtainable over a network (the Internet or a corporate intranet). The web components include images, text blocks, tables, and other common web components.

76. The registry that Confluence uses contains the address of a web service available over a network (the Confluence web server, available over the Internet or a corporate intranet) and symbolic names related to inputs and outputs of that web service. As shown in the image in ¶74 above, symbolic names are character strings that do not contain either a persistent address or pointer to an output value. The Confluence web server accepts both an input symbolic name and one or more associated input values from a user – for example, when the user interacts with a web element – and returns one or more outputs having an associated symbolic name.

77. The web services include, but are not limited to, "Soy templates": web services that receive symbolic input names and input values and return one or more outputs having an

associated symbolic name. *See* Atlassian, *Writing Soy Templates in Your Plugin* (Feb. 5, 2020), https://developer.atlassian.com/server/confluence/writing-soy-templates-in-your-plugin.

78. As shown in the image below, and by way of example, when the browser executes the code provided to it, it processes the symbolic names and transmits instructions back to the web service. In response, it accepts new, device-independent code from the web service.



```
batch.js?gatekeeper-ui-v2=true&highlightactions=true&hostenabled...
naturalchildren.action?decorator=none&excerpt=fals...estors=292511...
naturalchildren.action?decorator=none&excerpt=fals...0&mobile=fals...
naturalchildren.action?decorator=none&excerpt=fals...0&mobile=fals...
```

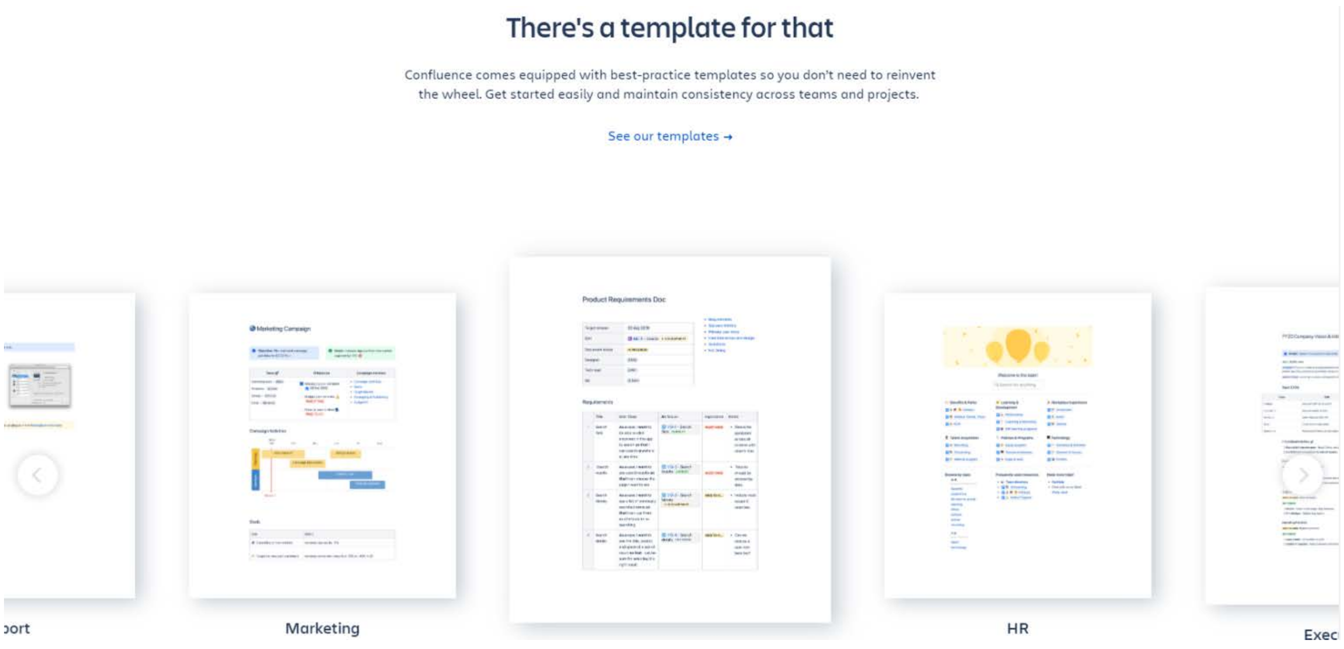
```
▾ General
  Request URL: https://wiki.opnfv.org/plugins/pagetree/naturalchildren.ac
  alse&sort=position&reverse=false&disableLinks=false&expandCurrent=true
  ue&pageId=2925118&treeId=0&startDepth=0&mobile=false&ancestors=2925118
  399162
  Request Method: GET
  Status Code: ● 200
  Remote Address: 54.148.217.218:443
  Referrer Policy: strict-origin-when-cross-origin
▾ Response Headers
  cache-control: no-cache, must-revalidate
  content-encoding: gzip
  content-security-policy: frame-ancestors 'self'
  content-type: text/html;charset=UTF-8
```

79. Atlassian was made aware of the ’755 patent and its infringement thereof at least as early as February 6, 2020, when Express Mobile provided notice to Atlassian of the ’755 patent.

80. Upon information and belief, since at least the time Atlassian received notice, Atlassian has induced and continues to induce others to infringe at least claim 23 of the ’755 patent under 35 U.S.C. § 271(b) by, among other things, and with specific intent or willful blindness, actively aiding and abetting others to infringe, including but not limited to Atlassian’s partners, clients, customers, and end users, whose use of Confluence constitutes direct infringement of at least one claim of the ’755 patent. In particular, Atlassian’s actions that aid and abet others such as customers, clients, partners, developers, and end users to infringe include advertising Confluence as a way for employees to collaborate on pages available on the Web. Upon information and belief, Atlassian has engaged in such actions with specific intent to cause

infringement or with willful blindness to the resulting infringement because Atlassian has had actual knowledge of the ’755 patent and knowledge that its acts were inducing infringement of the ’755 patent since at least the date Atlassian received notice that such activities infringed the ’755 patent.

81. By way of example, Atlassian ran graphic advertisements on its website featuring infringing features of Confluence. For example, Atlassian ran an advertisement explaining that Confluence “comes equipped with best-practice templates,” allowing users to easily create webpages.



There's a template for that

Confluence comes equipped with best-practice templates so you don't need to reinvent the wheel. Get started easily and maintain consistency across teams and projects.

See our templates →

82. Since February 6, 2020, Atlassian’s infringement of the ’755 patent has been willful.

83. Atlassian’s infringement has damaged and continues to damage and injure Express Mobile.

COUNT V – INFRINGEMENT OF U.S. PATENT NO. 9,063,755 (Trello)

84. The allegations set forth in the foregoing paragraphs 1 through 83 are incorporated into this Fifth Claim for Relief.

85. Atlassian has manufactured, used, offered for sale, or sold browser-based website building tools that infringe, either literally or under the doctrine of equivalents, one or more claims of the ’755 patent in violation of 35 U.S.C. § 271(a).

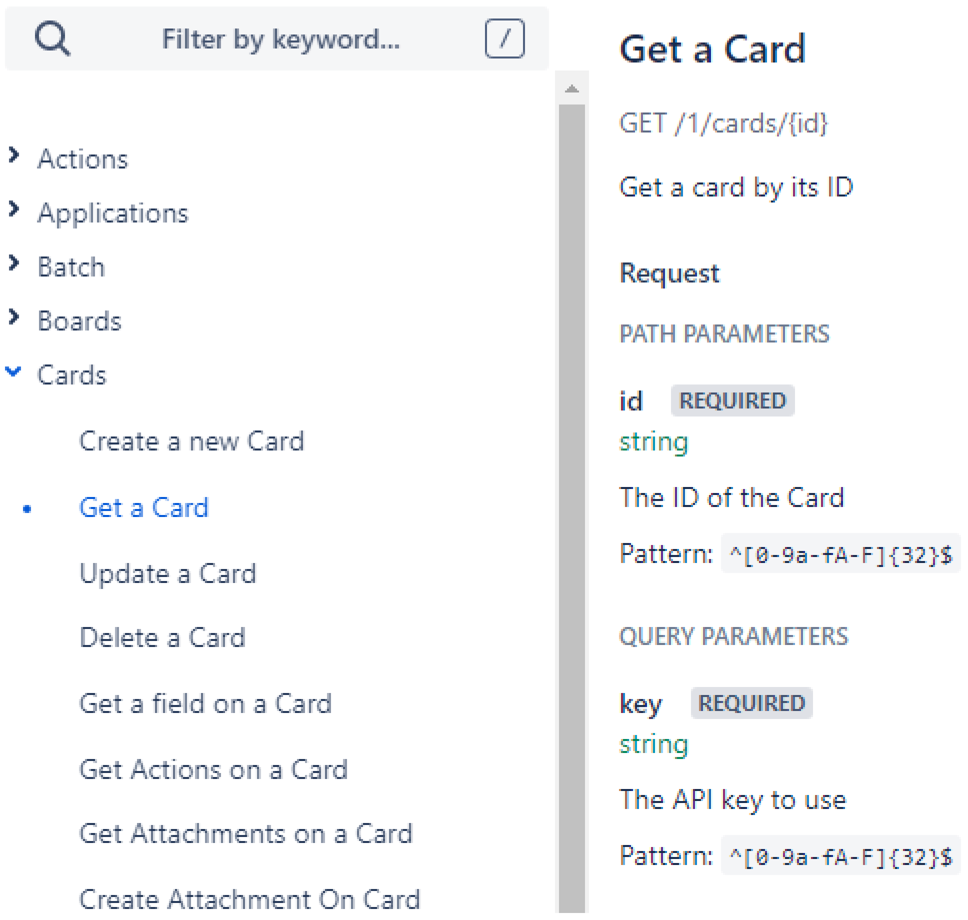
86. Upon information and belief, Atlassian has infringed and continues to infringe at least claim 23 of the ’755 patent. Trello practices a method for providing information to a device having a display, including a web browser. The information comes from a web service, the Trello web server. The webpages comprise columns of cards containing text, images, and icons.

87. Trello accepts and executes device- and platform-dependent code from the Trello web server, including HTML, CSS, and JavaScript. Some of the code is written for specific device platforms and devices, such as browsers, laptops, or smartphones (“device-platform-dependent code”), while some operates identically on all device platforms and devices (“device-independent code”). The device-independent code includes symbolic names of inputs and outputs. Those symbolic names are assigned to web elements, including Trello cards. Solely by way of example, the symbolic names are associated with web elements through the HTML “class” and “id” attributes. Solely by way of further example, individual Trello cards have symbolic names that are assigned upon their creation. *See* Atlassian, *Trello Developer – Cards* (Jan. 8, 2021) (Rest API to “Get a card by its ID”), https://developer.atlassian.com/cloud/trello/rest/api-group-cards/#api-cards-id-checklists-idchecklist-delete.

88. The symbolic names are provided from a registry of web components related to inputs and outputs obtainable over a network (the Internet or a corporate intranet). The web

components include images, text blocks, cards, labels, and other common web components. *See* Atlassian, *Trello Developer – Cards* (Jan. 8, 2021) (Rest API to "Get the stickers on a card"), https://developer.atlassian.com/cloud/trello/rest/api-group-cards/#api-cards-id-stickers-get.

89. The registry that Trello uses contains the address of a web service available over a network (the Trello web server, available over the Internet or a corporate intranet) and symbolic names related to inputs and outputs of that web service. As shown below, symbolic names are character "strings" that do not contain either a persistent address or pointer to an output value. The Trello web server accepts both an input symbolic name and one or more associated input values from a user – for example, when the user interacts with a Trello card, either by moving it or editing it – and returns one or more outputs having an associated symbolic name.



Get a Card

GET /1/cards/{id}

Get a card by its ID

Request

PATH PARAMETERS

id REQUIRED
string
The ID of the Card
Pattern: `^[0-9a-fA-F]{32}$`

QUERY PARAMETERS

key REQUIRED
string
The API key to use
Pattern: `^[0-9a-fA-F]{32}$`

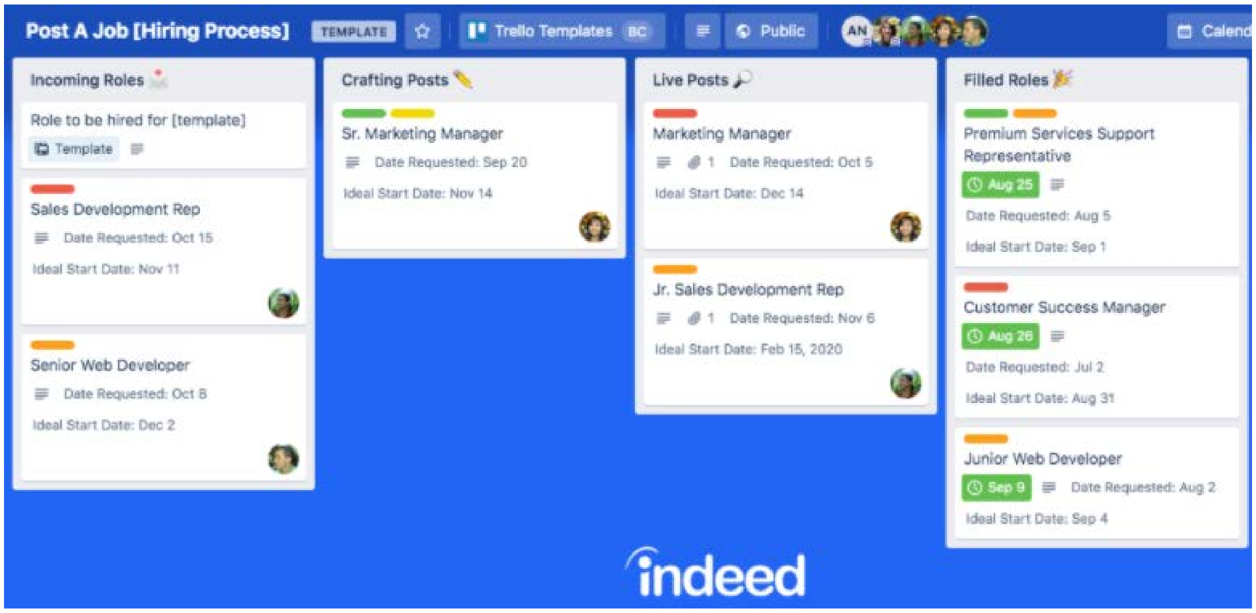
90. When the browser executes the code provided to it, it processes the symbolic names and transmits instructions back to the web service. In response, it accepts new, device-independent code from the web service.

91. Atlassian was made aware of the ’755 patent and its infringement thereof at least as early as February 6, 2020, when Express Mobile provided notice to Atlassian of the ’755 patent.

92. Upon information and belief, since at least the time Atlassian received notice, Atlassian has induced and continues to induce others to infringe at least claim 23 of the ’755 patent under 35 U.S.C. § 271(b) by, among other things, and with specific intent or willful blindness, actively aiding and abetting others to infringe, including but not limited to Atlassian’s partners, clients, customers, and end users, whose use of Trello constitutes direct infringement of at least one claim of the ’755 patent. In particular, Atlassian’s actions that aid and abet others such as customers, clients, partners, developers, and end users to infringe include advertising Trello as a way for employees to collaborate on pages available on the Web. Upon information and belief, Atlassian has engaged in such actions with specific intent to cause infringement or with willful blindness to the resulting infringement because Atlassian has had actual knowledge of the ’755 patent and knowledge that its acts were inducing infringement of the ’755 patent since at least the date Atlassian received notice that such activities infringed the ’755 patent.

93. By way of example, Atlassian ran graphic advertisements on its website featuring infringing features of Trello. For example, Atlassian ran an advertisement showing that users can collaborate with others to organize cards “in a fun, flexible, and rewarding way.”

94. Another advertisement showed Trello’s visual interface for creating, editing, and moving cards in a visual manner. *See* Emily Whitten, *How To Use Trello and Indeed To Hire a Dream Team* (Nov. 18, 2019), https://blog.trello.com/indeed-template-story.



Incoming Roles

Role to be hired for [template]
Template

Sales Development Rep
Date Requested: Oct 15
Ideal Start Date: Nov 11

Senior Web Developer
Date Requested: Oct 8
Ideal Start Date: Dec 2

Crafting Posts

Sr. Marketing Manager
Date Requested: Sep 20
Ideal Start Date: Nov 14

Live Posts

Marketing Manager
1 Date Requested: Oct 5
Ideal Start Date: Dec 14

Jr. Sales Development Rep
1 Date Requested: Nov 6
Ideal Start Date: Feb 15, 2020

Filled Roles

Premium Services Support Representative
Aug 25
Date Requested: Aug 5
Ideal Start Date: Sep 1

Customer Success Manager
Aug 26
Date Requested: Jul 2
Ideal Start Date: Aug 31

Junior Web Developer
Sep 9 Date Requested: Aug 2
Ideal Start Date: Sep 4

95. Since February 6, 2020, Atlassian's infringement of the '755 patent has been willful.

96. Atlassian's infringement has damaged and continues to damage and injure Express Mobile.

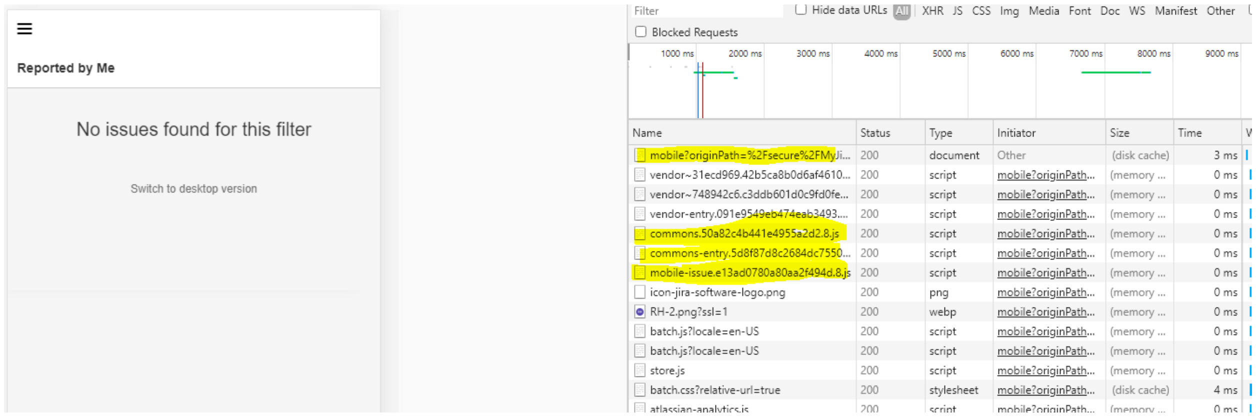
COUNT VI – INFRINGEMENT OF U.S. PATENT NO. 9,063,755 (JIRA)

97. The allegations set forth in the foregoing paragraphs 1 through 96 are incorporated into this Sixth Claim for Relief.

98. Atlassian has manufactured, used, offered for sale, or sold browser-based website building tools that infringe, either literally or under the doctrine of equivalents, one or more claims of the '755 patent in violation of 35 U.S.C. § 271(a).

99. Upon information and belief, Atlassian has infringed and continues to infringe at least claim 23 of the '755 patent. JIRA practices a method for providing information to a device having a display, including a web browser. The information comes from a web service, the JIRA web server.

100. JIRA accepts and executes code from the JIRA web server, including HTML, CSS, and JavaScript. Some of the code is written for specific device platforms and devices, such as browsers, laptops, or smartphones ("device-platform-dependent code," first image below, highlighted), while some operates identically on all device platforms and devices ("device-independent code," second image below, highlighted). The device-independent code includes symbolic names of inputs and outputs. These symbolic names are assigned to web elements. Solely by way of example, the symbolic names are associated with web elements through the HTML "class" and "id" attributes. Solely by way of further example, individual issues are associated with symbolic names. *See* Atlassian, *JIRA Rest API Examples* (Jan. 8, 2021), https://developer.atlassian.com/server/jira/platform/jira-rest-api-examples (Rest API for creating issue with string "id" field).

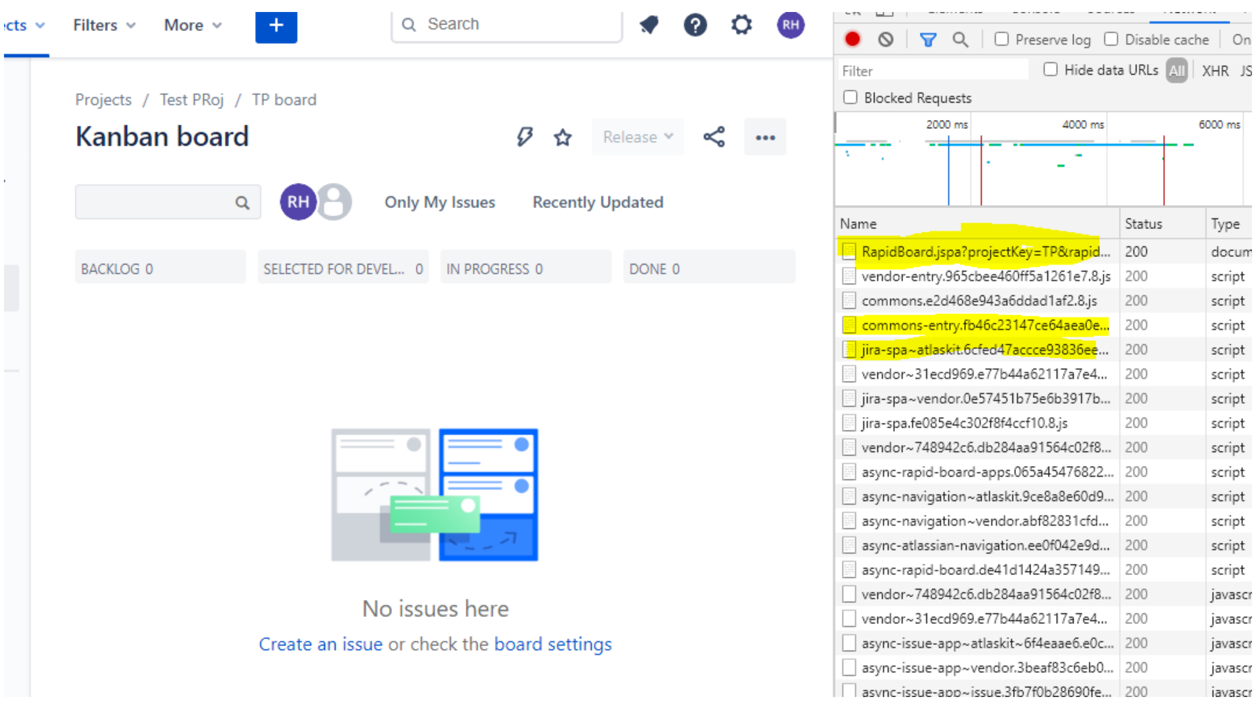


Reported by Me

No issues found for this filter

Switch to desktop version

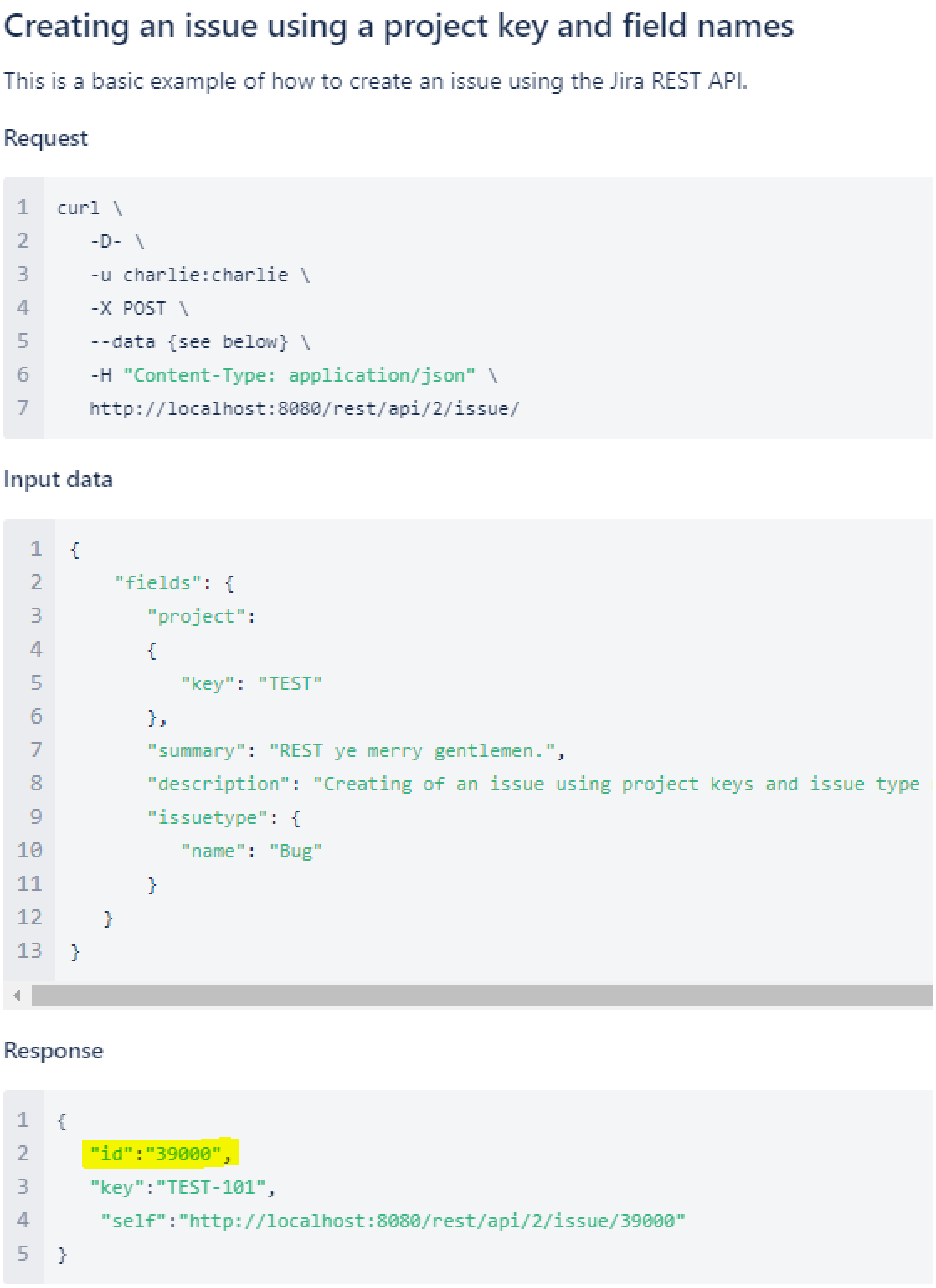
Name	Status	Type	Initiator	Size	Time
mobile?originPath=%2Fsecure%2FMyJi...	200	document	Other	(disk cache)	3 ms
vendor~31ecd969.42b5ca8b0d6af4610...	200	script	mobile?originPath...	(memory ...	0 ms
vendor~748942c6.c3ddb601d0c9fd0fe...	200	script	mobile?originPath...	(memory ...	0 ms
vendor-entry.091e9549eb474eab3493....	200	script	mobile?originPath...	(memory ...	0 ms
commons.50a82c4b441e4955a2d2.8.js	200	script	mobile?originPath...	(memory ...	0 ms
commons-entry.5d8f87d8c2684dc7550...	200	script	mobile?originPath...	(memory ...	0 ms
mobile-issue.e13ad0780a80aa2f494d.8.js	200	script	mobile?originPath...	(memory ...	0 ms
icon-jira-software-logo.png	200	png	mobile?originPath...	(memory ...	0 ms
RH-2.png?ssl=1	200	webp	mobile?originPath...	(memory ...	0 ms
batch.js?locale=en-US	200	script	mobile?originPath...	(memory ...	0 ms
batch.js?locale=en-US	200	script	mobile?originPath...	(memory ...	0 ms
store.js	200	script	mobile?originPath...	(memory ...	0 ms
batch.css?relative-url=true	200	stylesheet	mobile?originPath...	(disk cache)	4 ms
atlassian-analytics.js	200	script	mobile?originPath...	(memory ...	0 ms



Name	Status	Type
RapidBoard.jspa?projectKey=TP&rapid...	200	docum
vendor-entry.965cbee460ff5a1261e7.8.js	200	script
commons.e2d468e943a6ddad1af2.8.js	200	script
commons-entry.fb46c23147ce64aea0e...	200	script
jira-spa~atlaskit.6cfed47accce93836ee...	200	script
vendor~31ecd969.e77b44a62117a7e4...	200	script
jira-spa~vendor.0e57451b75e6b3917b...	200	script
jira-spa.fe085e4c302f8f4ccf10.8.js	200	script
vendor~748942c6.db284aa91564c02f8...	200	script
async-rapid-board-apps.065a45476822...	200	script
async-navigation~atlaskit.9ce8a8e60d9...	200	script
async-navigation~vendor.abf82831cfd...	200	script
async-atlassian-navigation.ee0f042e9d...	200	script
async-rapid-board.de41d1424a357149...	200	script
vendor~748942c6.db284aa91564c02f8...	200	javascr
vendor~31ecd969.e77b44a62117a7e4...	200	javascr
async-issue-app~atlaskit~6f4eaae6.e0c...	200	javascr
async-issue-app~vendor.3beaf83c6eb0...	200	javascr
async-issue-app~issue.3fb7f0b28690fe...	200	javascr

101. The symbolic names are provided from a registry of web components related to inputs and outputs obtainable over a network (the Internet or a corporate intranet). The web components include images, text blocks, tables, and other common web components.

102. The registry that JIRA uses contains the address of a web service available over a network (the JIRA web server, available over the Internet or a corporate intranet) and symbolic names related to inputs and outputs of that web service. As shown in the image below, symbolic names are character strings that do not contain either a persistent address or pointer to an output value. The JIRA web server accepts both an input symbolic name and one or more associated input values from a user – for example, when the user interacts with a web element – and returns one or more outputs having an associated symbolic name.



Creating an issue using a project key and field names

This is a basic example of how to create an issue using the Jira REST API.

Request

```
curl \
   -D- \
   -u charlie:charlie \
   -X POST \
   --data {see below} \
   -H "Content-Type: application/json" \
   http://localhost:8080/rest/api/2/issue/
```

Input data

```
{
    "fields": {
       "project":
       {
          "key": "TEST"
       },
       "summary": "REST ye merry gentlemen.",
       "description": "Creating of an issue using project keys and issue type
       "issuetype": {
          "name": "Bug"
       }
   }
}
```

Response

```
{
   "id":"39000",
   "key":"TEST-101",
    "self":"http://localhost:8080/rest/api/2/issue/39000"
}
```

103. When the browser executes the code provided to it, it processes the symbolic names and transmits instructions back to the web service. In response, it accepts new, device-independent code from the web service.

104. Atlassian was made aware of the ’755 patent and its infringement thereof at least as early as August 31, 2020, when Express Mobile provided notice to Atlassian of the ’755 patent.

105. Upon information and belief, since at least the time Atlassian received notice, Atlassian has induced and continues to induce others to infringe at least claim 23 of the ’755 patent under 35 U.S.C. § 271(b) by, among other things, and with specific intent or willful blindness, actively aiding and abetting others to infringe, including but not limited to Atlassian’s partners, clients, customers, and end users, whose use of JIRA constitutes direct infringement of at least one claim of the ’755 patent. In particular, Atlassian’s actions that aid and abet others such as customers, clients, partners, developers, and end users to infringe include advertising JIRA as a way for employees to collaborate on pages available on the Web. Upon information and belief, Atlassian has engaged in such actions with specific intent to cause infringement or with willful blindness to the resulting infringement because Atlassian has had actual knowledge of the ’755 patent and knowledge that its acts were inducing infringement of the ’755 patent since at least the date Atlassian received notice that such activities infringed the ’755 patent.

106. By way of example, Atlassian ran graphic advertisements on its website featuring infringing features of JIRA. For example, Atlassian ran an advertisement showing that users can create a webpage summarizing data entered into the platform “all in a few clicks.”

107. Other JIRA advertisements touted JIRA’s interface allowing for the easy creation of “flexible” and “customizable” pages.

108. Since August 31, 2020, Atlassian's infringement of the '755 patent has been willful.

109. Atlassian's infringement has damaged and continues to damage and injure Express Mobile.

COUNT VII – INFRINGEMENT OF U.S. PATENT NO. 9,471,287
(Confluence)

110. The allegations set forth in the foregoing paragraphs 1 through 109 are incorporated into this Seventh Claim for Relief.

111. On October 18, 2016, U.S. Patent No. 9,471,287 ("the '287 patent"), entitled *Systems and Methods for Integrating Widgets on Mobile Devices*, was duly and legally issued by the United States Patent and Trademark Office. A true and correct copy of the '287 patent is attached as Exhibit C.

112. The invention claimed in the '287 patent resolves technical problems related to generating and distributing dynamic content on a device display, such as the display on a mobile device. Before the patents-in-suit, content and applications for device displays were generally created using code written for each individual type of device. As device types proliferated, programming content and applications for each device became increasingly expensive and time-consuming. Doing so also limited the ability of providers to update the capabilities of, and increase the available content for, many devices.

113. The invention of the '287 patent resolves technical problems related to generating and distributing dynamic content on a device display, such as the display of a mobile device. The invention of the '287 patent features a registry and an authoring tool or Player configured to define a User Interface ("UI") object for display on the device, where the UI object corresponds to a web component. Each UI object is either: (1) selected by a user or (2) automatically selected

by the system as a preferred UI object corresponding to a symbolic name of the web component and used to produce an Application, where the Application is a device-independent code, and a Player, where the Player is a device- and platform-dependent code. The Application and Player (1) enable the device to provide one or more input values and corresponding input symbolic name to the web service and (2) enable the web service to use the input symbolic name and one or more user-provided input values to generate one or more output values having an associated output symbolic name, while (3) the Player receives the output symbolic name and one or more corresponding output values and provides instructions for the display of the device to present an output value in the defined UI object. Those features are exclusively implemented using computer technology.

114. Unlike methods in the prior art, the ’287 patent brings together disparate ideas and concepts for generating and distributing content on different device displays. This can include building a webpage or application using a “what you see is what you get” environment, so that, as the page or app is created or edited, it can be viewed in the same manner it will appear on different types of screens when later accessed. The invention can also include an authoring tool that can create an Application that includes dynamic content, where the Application is device-independent code, and a Player, where the Player is device- and platform-dependent code. The Player enables the Application to function on a variety of devices or platforms, with differing functionality. That enables users of the authoring tool to create and distribute device-independent Applications for different device types, without individually tailoring the device-independent dynamic-content Applications for each device type.

115. The claims of the ’287 patent do not merely recite the performance of some business practice known from the pre-Internet world along with the requirement to perform it on the Internet. Instead, the claims of the ’287 patent recite one or more inventive concepts that are

rooted in the computerized generation of content on a device display, such as a mobile device display, and overcome problems specifically arising in the realm of computerized display content generation technologies.

116. The claims of the ’287 patent recite an invention that is not merely the routine or conventional use of systems and methods for the computerized generation of content on a device display. Instead, the invention features systems that can be used with devices and methods of using the systems with authoring tools or Players specific to each device platform and Applications that are independent of the device.

117. The invention claimed in the ’287 patent offers substantial improvements in device performance and web or application design. For example, the invention allows for faster loading speeds, more efficient and flexible processing of webpage or application dynamic content, and the ability to change a webpage or application more efficiently by editing user settings rather than multiple versions of code. The invention also permits scaling of webpages and dynamic content within the webpage, or applications and elements within the application that contain dynamic content, to most efficiently use the screen space. Taken separately or together, the claim elements of the invention significantly improve the operation of a computer and the process of web design.

118. The invention claimed in the ’287 patent neither preempts all ways for the computerized generation of content on a device display, such as a mobile device, nor preempts the use of all authoring tools or Players for the computerized generation of content on a device display, such as a mobile device, or any other well-known or prior-art technology.

119. Accordingly, each claim of the ’287 patent recites a combination of elements sufficient to ensure that the claim in practice amounts to significantly more than a patent on an ineligible concept.

120. Express Mobile is the assignee and owner of the right, title, and interest in and to the ’287 patent, including the right to assert all causes of action arising under the patent and the right to any remedies for infringement of it.

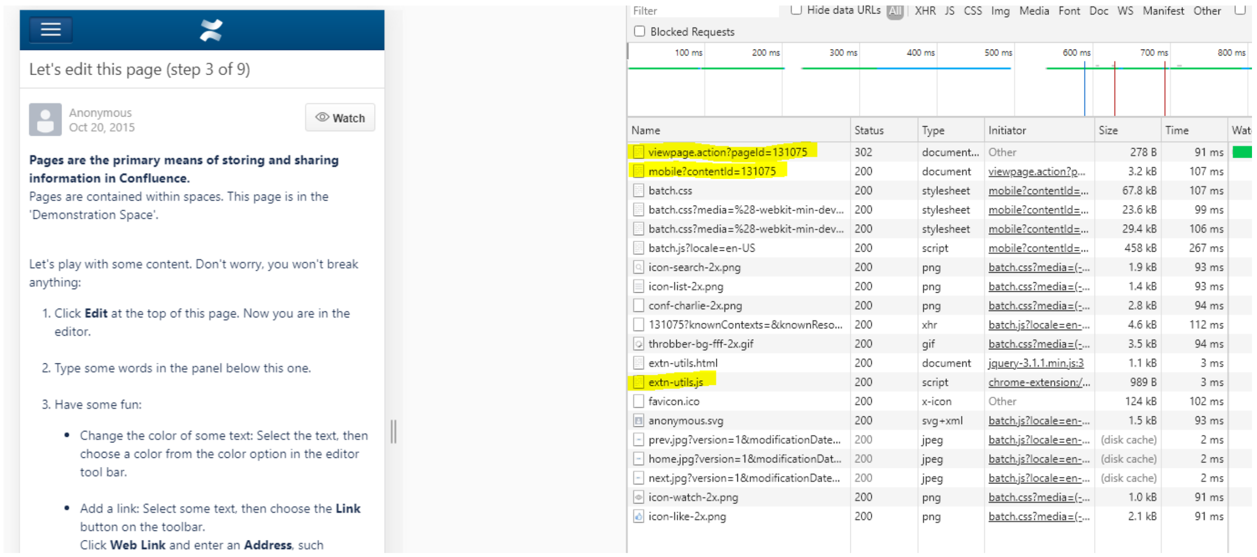
121. Atlassian has manufactured, used, offered for sale, or sold browser-based website building tools that infringe, either literally or under the doctrine of equivalents, one or more claims of the ’287 patent in violation of 35 U.S.C. § 271(a).

122. Upon information and belief, Atlassian has infringed and continues to infringe at least claim 15 of the ’287 patent.

123. Claim 15 of the ’287 patent recites a method of displaying content on a display of a device having a Player, where said Player is a device- and platform-dependent code, said method comprising: defining a user interface (“UI”) object for presentation on the display, where said UI object corresponds to a web component included in a registry of one or more web components selected from a group consisting of an input of a web service and an output of the web service where each web component includes a plurality of symbolic names of inputs and outputs associated with each web service, and where the registry includes: (a) symbolic names required for evoking one or more web components each related to a set of inputs and outputs of the web service obtainable over a network, where the symbolic names are character strings that do not contain either a persistent address or pointer to an output value accessible to the web service, and (b) an address of the web service, and where each defined UI object is either: (1) selected by a user of an authoring tool; or (2) automatically selected by a system as a preferred UI object corresponding to a symbolic name of the web component selected by the user of the authoring tool; selecting the symbolic name from said web component corresponding to the defined UI object, where the selected symbolic name has an associated data format class type corresponding to a subclass of UI objects that support the data format type of the symbolic name,

and has the preferred UI object; associating the selected symbolic name with the defined UI object; and producing an Application including the selected symbolic name of the defined UI object, where said Application is a device-independent code, wherein, when the Application and Player are provided to the device and executed on the device, and when a user of the device provides one or more input values associated with an input symbolic name to an input of defined UI object, (1) the device provides the user provided one or more input values and corresponding input symbolic name to the web service, (2) the web service uses the input symbolic name and the user provided one or more input values for generating one or more output values having an associated output symbolic name, (3) said Player receives the output symbolic name and corresponding one or more output values and provides instructions for a display of the device to present an output value in the defined UI object.

124. Atlassian infringes claim 15 of the ’287 patent through a combination of features that collectively practice each limitation of claim 15. As shown in the images below, Confluence practices a method for displaying content on a display of a computer device platform having a Player – HTML, CSS, and JavaScript code written for a particular device platform, such as browsers, laptops, tablets, and smartphones. The content is provided to the user through a web browser on webpages.



Let's edit this page (step 3 of 9)

Anonymous
Oct 20, 2015

Watch

Pages are the primary means of storing and sharing information in Confluence.
Pages are contained within spaces. This page is in the 'Demonstration Space'.

Let's play with some content. Don't worry, you won't break anything:

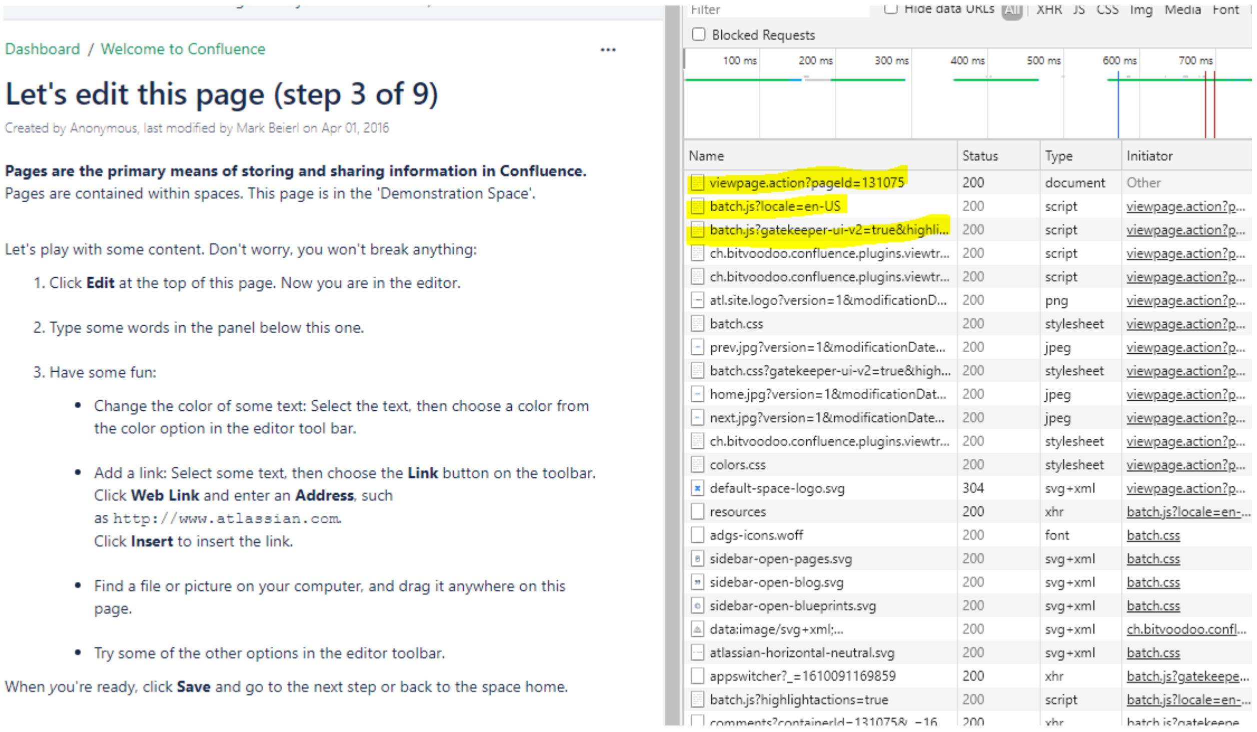
1. Click **Edit** at the top of this page. Now you are in the editor.
2. Type some words in the panel below this one.
3. Have some fun:
   - Change the color of some text: Select the text, then choose a color from the color option in the editor tool bar.
   - Add a link: Select some text, then choose the **Link** button on the toolbar.
     Click **Web Link** and enter an **Address**, such

Filter ☐ Hide data URLs All | XHR JS CSS Img Media Font Doc WS Manifest Other ☐

☐ Blocked Requests

100 ms 200 ms 300 ms 400 ms 500 ms 600 ms 700 ms 800 ms

Name	Status	Type	Initiator	Size	Time	Wat
viewpage.action?pageId=131075	302	document...	Other	278 B	91 ms	🟩
mobile?contentId=131075	200	document	viewpage.action?p...	3.2 kB	107 ms	
batch.css	200	stylesheet	mobile?contentId=...	67.8 kB	107 ms	
batch.css?media=%28-webkit-min-dev...	200	stylesheet	mobile?contentId=...	23.6 kB	99 ms	
batch.css?media=%28-webkit-min-dev...	200	stylesheet	mobile?contentId=...	29.4 kB	106 ms	
batch.js?locale=en-US	200	script	mobile?contentId=...	458 kB	267 ms	
icon-search-2x.png	200	png	batch.css?media=(-...	1.9 kB	93 ms	
icon-list-2x.png	200	png	batch.css?media=(-...	1.4 kB	93 ms	
conf-charlie-2x.png	200	png	batch.css?media=(-...	2.8 kB	94 ms	
131075?knownContexts=&knownReso...	200	xhr	batch.js?locale=en-...	4.6 kB	112 ms	
throbber-bg-fff-2x.gif	200	gif	batch.css?media=(-...	3.5 kB	94 ms	
extn-utils.html	200	document	jquery-3.1.1.min.js:3	1.1 kB	3 ms	
extn-utils.js	200	script	chrome-extension:/...	989 B	3 ms	
favicon.ico	200	x-icon	Other	124 kB	102 ms	
anonymous.svg	200	svg+xml	batch.js?locale=en-...	1.5 kB	93 ms	
prev.jpg?version=1&modificationDate...	200	jpeg	batch.js?locale=en-...	(disk cache)	2 ms	
home.jpg?version=1&modificationDat...	200	jpeg	batch.js?locale=en-...	(disk cache)	2 ms	
next.jpg?version=1&modificationDate...	200	jpeg	batch.js?locale=en-...	(disk cache)	2 ms	
icon-watch-2x.png	200	png	batch.css?media=(-...	1.0 kB	91 ms	
icon-like-2x.png	200	png	batch.css?media=(-...	2.1 kB	91 ms	



Dashboard / Welcome to Confluence ...

Let's edit this page (step 3 of 9)

Created by Anonymous, last modified by Mark Beierl on Apr 01, 2016

Pages are the primary means of storing and sharing information in Confluence.
Pages are contained within spaces. This page is in the 'Demonstration Space'.

Let's play with some content. Don't worry, you won't break anything:

1. Click **Edit** at the top of this page. Now you are in the editor.
2. Type some words in the panel below this one.
3. Have some fun:
   - Change the color of some text: Select the text, then choose a color from the color option in the editor tool bar.
   - Add a link: Select some text, then choose the **Link** button on the toolbar. Click **Web Link** and enter an **Address**, such as `http://www.atlassian.com`. Click **Insert** to insert the link.
   - Find a file or picture on your computer, and drag it anywhere on this page.
   - Try some of the other options in the editor toolbar.

When you're ready, click **Save** and go to the next step or back to the space home.

Filter ☐ Hide data URLs All | XHR JS CSS Img Media Font

☐ Blocked Requests

100 ms 200 ms 300 ms 400 ms 500 ms 600 ms 700 ms

Name	Status	Type	Initiator
viewpage.action?pageId=131075	200	document	Other
batch.js?locale=en-US	200	script	viewpage.action?p...
batch.js?gatekeeper-ui-v2=true&highli...	200	script	viewpage.action?p...
ch.bitvoodoo.confluence.plugins.viewtr...	200	script	viewpage.action?p...
ch.bitvoodoo.confluence.plugins.viewtr...	200	script	viewpage.action?p...
atl.site.logo?version=1&modificationD...	200	png	viewpage.action?p...
batch.css	200	stylesheet	viewpage.action?p...
prev.jpg?version=1&modificationDate...	200	jpeg	viewpage.action?p...
batch.css?gatekeeper-ui-v2=true&high...	200	stylesheet	viewpage.action?p...
home.jpg?version=1&modificationDat...	200	jpeg	viewpage.action?p...
next.jpg?version=1&modificationDate...	200	jpeg	viewpage.action?p...
ch.bitvoodoo.confluence.plugins.viewtr...	200	stylesheet	viewpage.action?p...
colors.css	200	stylesheet	viewpage.action?p...
default-space-logo.svg	304	svg+xml	viewpage.action?p...
resources	200	xhr	batch.js?locale=en-...
adgs-icons.woff	200	font	batch.css
sidebar-open-pages.svg	200	svg+xml	batch.css
sidebar-open-blog.svg	200	svg+xml	batch.css
sidebar-open-blueprints.svg	200	svg+xml	batch.css
data:image/svg+xml;...	200	svg+xml	ch.bitvoodoo.confl...
atlassian-horizontal-neutral.svg	200	svg+xml	batch.css
appswitcher?_=1610091169859	200	xhr	batch.js?gatekeepe...
batch.js?highlightactions=true	200	script	batch.js?locale=en-...
comments?containerId=131075&_=16	200	xhr	batch.js?gatekeepe

125. The method includes computer memory that stores a registry of symbolic names associated with web components – including images, text blocks, tables, and other web components. These components are related to inputs and outputs of a web service – the Confluence web server – obtainable over a network, such as the Internet or corporate intranet.

See Atlassian, *Rest APIs* (Jan. 8, 2021) (describing various components such as "RadioButtons" and "SelectList," the contents of which, based upon information and belief, can be stored in a "memory" and retrieved), https://developer.atlassian.com/server/jira/platform/rest-apis.

126. The names stored in the registry are character strings that do not contain either a persistent address or pointer to an output value accessible to the web service. Each symbolic name has an associated data format class type corresponding to specific user interface objects that support that data format type, and are associated with preferred user interface objects. Confluence's code, including its HTML, JavaScript, and CSS code, associates these symbolic names – represented as element types, classes, and IDs in the browser's Document Object Model – with specific user interface objects.

127. The Confluence registry also includes the address of the Confluence web server, a web service.

128. Confluence also includes an authoring tool that lets users create web pages including user interface objects for presentation on the web browser. These user interface objects correspond to standard web components, including tables and images. The web components are stored in the registry. The authoring tool accesses the computer memory to select a symbolic name corresponding to the web component and associate it with the defined user interface object. A particular symbolic name is only available for particular types of user interface objects, and the defined user interface object is automatically selected by the system as the preferred object corresponding to the symbolic name of the web component selected by the user of the authoring tool. For example, a CSS stylesheet may associate all symbolic names with a given type of object like an image or a table. By way of further example, an "AUI" component may associate symbolic names with particular types of components. *See* Atlassian, *AUI – Tables*, https://aui.atlassian.com/aui/7.9/docs/tables.html.

129. The authoring tool then produces an "Application" in the form of device-independent code, including HTML, CSS, and JavaScript code that includes the selected symbolic name. It also produces a Player in the form of device- and platform-dependent code, also including HTML, CSS, and JavaScript code. Unlike the Application code, the Player code is written for specific device platforms, such as laptops, tablets, or smartphones, or device platforms, such as browsers. The Player and Application code collaborate to provide the page to the user.

130. When the Application and Player code are executed by the web browser, input provided by the user is sent to the web service, which generates output and sends it to be displayed in a user interface object on the device. The Player interprets the response and updates the user interface objects based on the output it received from the web service.

131. Atlassian was made aware of the '287 patent and its infringement thereof at least as early as February 6, 2020, when Express Mobile provided notice to Atlassian of the '287 patent.

132. Upon information and belief, since at least the time Atlassian received notice, Atlassian has induced and continues to induce others to infringe at least claim 15 of the '287 patent under 35 U.S.C. § 271(b) by, among other things, and with specific intent or willful blindness, actively aiding and abetting others to infringe, including but not limited to Atlassian's partners, clients, customers, and end users, whose use of Confluence constitutes direct infringement of at least one claim of the '287 patent. In particular, Atlassian's actions that aid and abet others such as customers, clients, partners, developers, and end users to infringe include advertising Confluence as a way for employees to collaborate on pages available on the Web. Upon information and belief, Atlassian has engaged in such actions with specific intent to cause infringement or with willful blindness to the resulting infringement because Atlassian has had

actual knowledge of the ’287 patent and knowledge that its acts were inducing infringement of the ’287 patent since at least the date Atlassian received notice that such activities infringed the ’287 patent.

133. By way of example, Atlassian ran graphic advertisements on its website featuring infringing features of Confluence. For example, Atlassian ran an advertisement explaining that Confluence “comes equipped with best-practice templates,” allowing users to easily create webpages. *See* ¶ 81, *supra*.

134. Since February 6, 2020, Atlassian’s infringement of the ’287 patent has been willful.

135. Atlassian’s infringement has damaged and continues to damage and injure Express Mobile.

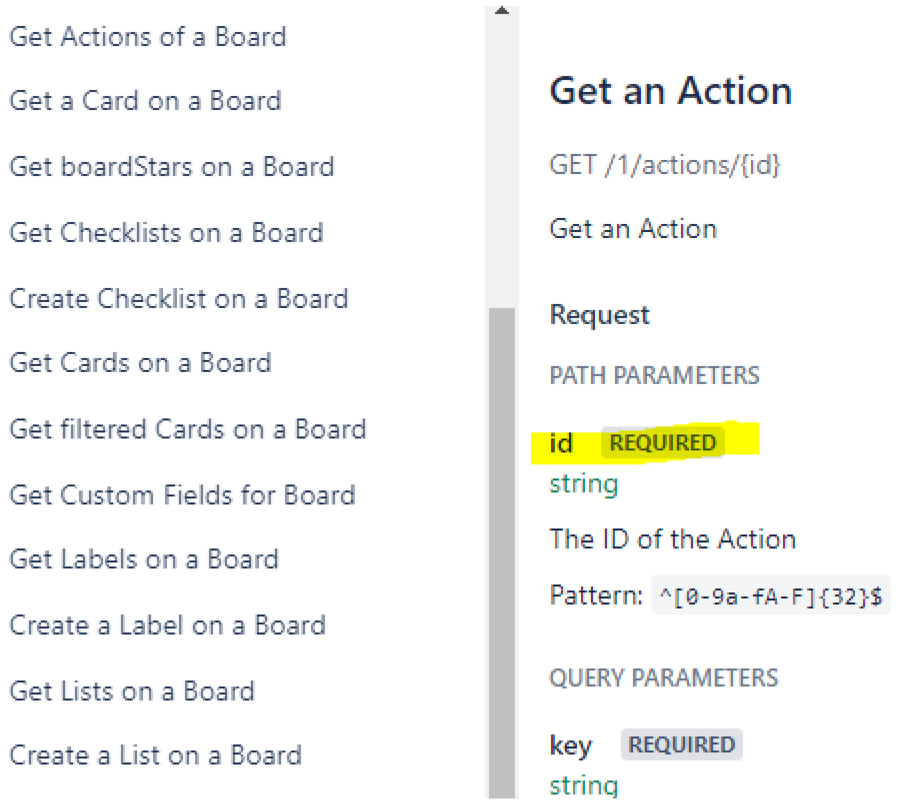
<u>COUNT VIII – INFRINGEMENT OF U.S. PATENT NO. 9,471,287</u> <u>(Trello)</u>

136. The allegations set forth in the foregoing paragraphs 1 through 135 are incorporated into this Eighth Claim for Relief.

137. Atlassian has manufactured, used, offered for sale, or sold browser-based website building tools that infringe, either literally or under the doctrine of equivalents, one or more claims of the ’287 patent in violation of 35 U.S.C. § 271(a).

138. Upon information and belief, Atlassian has infringed and continues to infringe at least claim 15 of the ’287 patent. Trello practices a method for displaying dynamic content on a display of a computer device having a Player – HTML, CSS, and JavaScript code written for a particular device platform, such as browsers, laptops, tablets, and smartphones. The content is provided to the user through a web browser on webpages. *See* ¶ 41, *supra*.

139. As shown below, and by way of example, the method includes computer memory that stores a registry of symbolic names associated with web components – including images, text blocks, tables, and other web components. These components are related to inputs and outputs of a web service – the Trello web server – obtainable over a network, such as the Internet.


Get Actions of a Board
Get a Card on a Board
Get boardStars on a Board
Get Checklists on a Board
Create Checklist on a Board
Get Cards on a Board
Get filtered Cards on a Board
Get Custom Fields for Board
Get Labels on a Board
Create a Label on a Board
Get Lists on a Board
Create a List on a Board

Get an Action

GET /1/actions/{id}

Get an Action

Request

PATH PARAMETERS

id REQUIRED
string
The ID of the Action
Pattern: `^[0-9a-fA-F]{32}$`

QUERY PARAMETERS

key REQUIRED
string

140. The names stored in the registry are character strings that do not contain either a persistent address or pointer to an output value accessible to the web service. Each symbolic name has an associated data format class type corresponding to specific user interface objects that support that data format type, and are associated with preferred user interface objects. Trello's code, including its HTML, JavaScript, and CSS code, associates those symbolic names – represented as element types, classes, and IDs in the browser's Document Object Model – with specific user interface objects.

141. The Trello registry also includes the address of the Trello web server, a web service.

142. Trello also includes an authoring tool that lets users create cards displayed as user interface objects for presentation on the web browser. These user interface objects correspond to combinations of web components stored in the registry. The authoring tool accesses the computer memory to select a symbolic name corresponding to the web component and associates it with the defined user interface object. A particular symbolic name is only available for particular types of user interface objects, and the defined user interface object is automatically selected by the system as the preferred object corresponding to the symbolic name of the web component selected by the user of the authoring tool. For example, a CSS stylesheet may associate all symbolic names with a given type of object like the color of a label on a card.

143. The authoring tool then produces an "Application" in the form of device-independent code, including HTML, CSS, and JavaScript code that includes the selected symbolic name. It also produces a Player in the form of device- and platform-dependent code, also including HTML, CSS, and JavaScript code. Unlike the Application code, the Player code is written for specific devices, such as laptops, tablets, or smartphones, or device platforms, such as browsers. The Player and Application code collaborate to provide the page to the user.

144. When the Application and Player code are executed by the web browser, input provided by the user is sent to the web service, which generates output and sends it to be displayed in a user interface object on the device. The Player interprets the response and updates the user interface objects based on the output it received from the web service.

145. Atlassian was made aware of the '287 patent and its infringement thereof at least as early as February 6, 2020, when Express Mobile provided notice to Atlassian of the '287 patent.

146. Upon information and belief, since at least the time Atlassian received notice, Atlassian has induced and continues to induce others to infringe at least claim 15 of the ’287 patent under 35 U.S.C. § 271(b) by, among other things, and with specific intent or willful blindness, actively aiding and abetting others to infringe, including but not limited to Atlassian’s partners, clients, customers, and end users, whose use of Trello constitutes direct infringement of at least one claim of the ’287 patent. In particular, Atlassian’s actions that aid and abet others such as customers, clients, partners, developers, and end users to infringe include advertising Trello as a way for employees to collaborate on pages available on the Web. Upon information and belief, Atlassian has engaged in such actions with specific intent to cause infringement or with willful blindness to the resulting infringement because Atlassian has had actual knowledge of the ’287 patent and knowledge that its acts were inducing infringement of the ’287 patent since at least the date Atlassian received notice that such activities infringed the ’287 patent.

147. By way of example, Atlassian ran graphic advertisements on its website featuring infringing features of Trello. For example, Atlassian ran an advertisement showing that users can collaborate with others to organize cards “in a fun, flexible, and rewarding way.”

148. Another advertisement showed Trello’s visual interface for creating, editing, and moving cards in a visual manner. *See* ¶94, *supra*; Emily Whitten, *How To Use Trello and Indeed To Hire a Dream Team* (Nov. 18, 2019), https://blog.trello.com/indeed-template-story.

149. Since February 6, 2020, Atlassian’s infringement of the ’287 patent has been willful.

150. Atlassian’s infringement has damaged and continues to damage and injure Express Mobile.

COUNT IX – INFRINGEMENT OF U.S. PATENT NO. 9,471,287 (JIRA)

151. The allegations set forth in the foregoing paragraphs 1 through 150 are incorporated into this Ninth Claim for Relief.

152. Atlassian has manufactured, used, offered for sale, or sold browser-based website building tools that infringe, either literally or under the doctrine of equivalents, one or more claims of the ’287 patent in violation of 35 U.S.C. § 271(a).

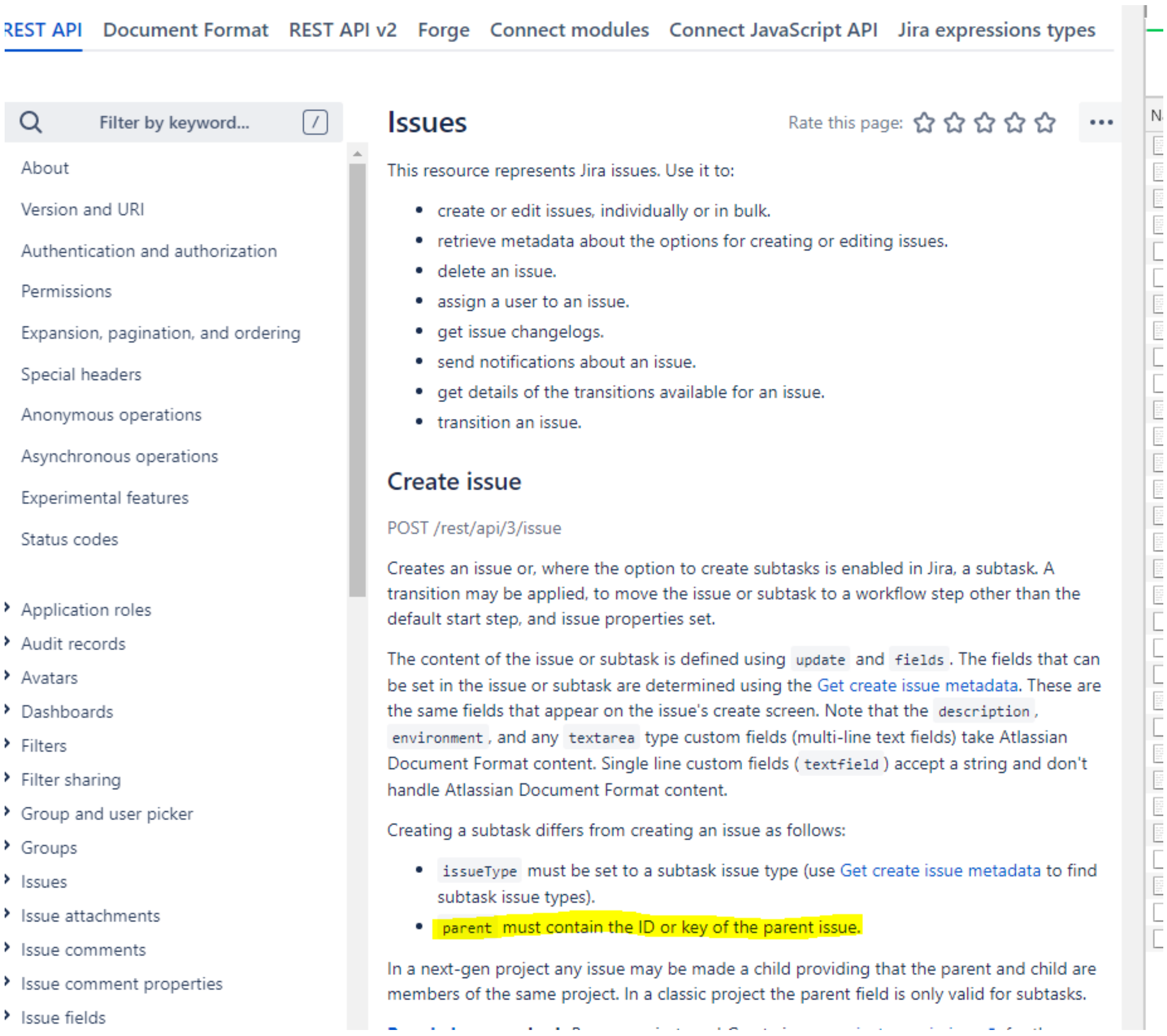
153. Upon information and belief, Atlassian has infringed and continues to infringe at least claim 15 of the ’287 patent. JIRA practices a method for displaying content on a display on a computer device having a Player – HTML, CSS, and JavaScript code written for a particular device, such as laptops, tablets, and smartphones. The content is provided to the user through a web browser on webpages.

154. The method includes computer memory that stores a registry of symbolic names associated with web components – including images, issue fields, and other web components. These components are related to inputs and outputs of a web service – the JIRA web server – obtainable over a network, such as the Internet or corporate intranet.

155. The names stored in the registry are character strings that do not contain either a persistent address or pointer to an output value accessible to the web service. Each symbolic name has an associated data format class type corresponding to specific user interface objects that support that data format type, and are associated with preferred user interface objects. JIRA’s code, including its HTML, JavaScript, and CSS code, associates these symbolic names – represented as element types, classes, and IDs in the browser’s Document Object Model – with specific user interface objects.

156. The JIRA registry also includes the address of the JIRA web server, a web service.

157. JIRA also includes an authoring tool that lets users create issue pages including user interface objects for presentation on the web browser. As shown in the image below, and by way of example, user interface objects correspond to standard web components, including tables and images. The web components are stored in the registry. The authoring tool accesses the computer memory to select a symbolic name (highlighted in the image below) corresponding to the web component and associates it with the defined user interface object. A particular symbolic name is only available for particular types of user interface objects, and the defined user interface object is automatically selected by the system as the preferred object corresponding to the symbolic name of the web component selected by the user of the authoring tool. For example, a CSS stylesheet may associate all symbolic names with a given type of object, like an image or a table.


REST API Document Format REST API v2 Forge Connect modules Connect JavaScript API Jira expressions types

Filter by keyword...

About
Version and URI
Authentication and authorization
Permissions
Expansion, pagination, and ordering
Special headers
Anonymous operations
Asynchronous operations
Experimental features
Status codes

› Application roles
› Audit records
› Avatars
› Dashboards
› Filters
› Filter sharing
› Group and user picker
› Groups
› Issues
› Issue attachments
› Issue comments
› Issue comment properties
› Issue fields

Issues

Rate this page:

This resource represents Jira issues. Use it to:

- create or edit issues, individually or in bulk.
- retrieve metadata about the options for creating or editing issues.
- delete an issue.
- assign a user to an issue.
- get issue changelogs.
- send notifications about an issue.
- get details of the transitions available for an issue.
- transition an issue.

Create issue

POST /rest/api/3/issue

Creates an issue or, where the option to create subtasks is enabled in Jira, a subtask. A transition may be applied, to move the issue or subtask to a workflow step other than the default start step, and issue properties set.

The content of the issue or subtask is defined using `update` and `fields`. The fields that can be set in the issue or subtask are determined using the Get create issue metadata. These are the same fields that appear on the issue's create screen. Note that the `description`, `environment`, and any `textarea` type custom fields (multi-line text fields) take Atlassian Document Format content. Single line custom fields (`textfield`) accept a string and don't handle Atlassian Document Format content.

Creating a subtask differs from creating an issue as follows:

- `issueType` must be set to a subtask issue type (use Get create issue metadata to find subtask issue types).
- `parent` must contain the ID or key of the parent issue.

In a next-gen project any issue may be made a child providing that the parent and child are members of the same project. In a classic project the parent field is only valid for subtasks.

158. The authoring tool then produces an "Application" in the form of device-independent code, including HTML, CSS, and JavaScript code that includes the selected symbolic name. It also produces a Player in the form of device- and platform-dependent code, also including HTML, CSS, and JavaScript code. Unlike the Application code, the Player code is written for specific device platforms, such as laptops, tablets, or smartphones, or device platforms, such as browsers. The Player and Application code collaborate to provide the page to the user.

159. When the Application and Player code are executed by the web browser, input provided by the user is sent to the web service, which generates output and sends it to be displayed in a user interface object on the device. The Player interprets the response and updates the user interface objects based on the output it received from the web service.

160. Atlassian was made aware of the ’287 patent and its infringement thereof at least as early as August 31, 2020, when Express Mobile provided notice to Atlassian of the ’287 patent.

161. Upon information and belief, since at least the time Atlassian received notice, Atlassian has induced and continues to induce others to infringe at least claim 15 of the ’287 patent under 35 U.S.C. § 271(b) by, among other things, and with specific intent or willful blindness, actively aiding and abetting others to infringe, including but not limited to Atlassian’s partners, clients, customers, and end users, whose use of JIRA constitutes direct infringement of at least one claim of the ’287 patent. In particular, Atlassian’s actions that aid and abet others such as customers, clients, partners, developers, and end users to infringe include advertising JIRA as a way for employees to collaborate on pages available on the Web. Upon information and belief, Atlassian has engaged in such actions with specific intent to cause infringement or with willful blindness to the resulting infringement because Atlassian has had actual knowledge of the ’287 patent and knowledge that its acts were inducing infringement of the ’287 patent since at least the date Atlassian received notice that such activities infringed the ’287 patent.

162. By way of example, Atlassian ran graphic advertisements on its website featuring infringing features of JIRA. For example, Atlassian ran an advertisement showing that users can create a webpage summarizing data entered into the platform “all in a few clicks.”

163. Other JIRA advertisements touted JIRA’s interface allowing for the easy creation of “flexible” and “customizable” pages.

164. Since August 31, 2020, Atlassian’s infringement of the ’287 patent has been willful.

165. Atlassian’s infringement has damaged and continues to damage and injure Express Mobile.

COUNT X – INFRINGEMENT OF U.S. PATENT NO. 9,928,044
(Confluence)

166. The allegations set forth in the foregoing paragraphs 1 through 165 are incorporated into this Tenth Claim for Relief.

167. On March 27, 2018, U.S. Patent No. 9,928,044 (“the ’044 patent”), entitled *Systems and Methods for Programming Mobile Devices*, was duly and legally issued by the United States Patent and Trademark Office. A true and correct copy of the ’044 patent is attached as Exhibit D.

168. The invention claimed in the ’044 patent resolves technical problems related to generating and distributing dynamic content on a device display, such as the display of a mobile device. Before the patents-in-suit, content and applications for device displays were generally created using code written for each individual type of device. As device types proliferated, programming content and applications for each device became increasingly expensive and time-consuming. Doing so also limited the ability of providers to update the capabilities of, and increase the available content for, many devices.

169. The invention of the ’044 patent resolves technical problems related to generating and distributing dynamic content on a device display, such as the display on a mobile device. The invention features a computer memory and an authoring tool or Player configured to define a User Interface (“UI”) object for display on the device, where the defined UI object corresponds to a web component and where each UI object is either: (1) selected by a user or (2)

automatically selected by the system as a preferred UI object corresponding to a symbolic name of the web component. Additionally, the computer memory and the authoring tool or Player are configured to build an Application consisting of one or more webpage views to provide for the display of at least a portion of one or more of the webpages. These features are exclusively implemented using computer technology.

170. Unlike methods in the prior art, the ’044 patent brings together disparate ideas and concepts for generating and distributing content on different device displays. This can include building a webpage or application using a “what you see is what you get” environment, so that, as the page or app is created or edited, it can be viewed in the same manner it will appear on different types of screens when later accessed. The invention can also include an authoring tool that can create an Application that includes dynamic content, where the Application is device-independent code, and a Player, where the Player is device- and platform-dependent code. The Player enables the Application to function on a variety of devices or platforms, with differing functionality. This enables users of the authoring tool to create and distribute device-independent Applications for different device types, without individually tailoring the device-independent dynamic-content Applications for each device type.

171. The claims of the ’044 patent do not merely recite the performance of some business practice known from the pre-Internet world along with the requirement to perform it on the Internet. Instead, the claims of the ’044 patent recite one or more inventive concepts that are rooted in the computerized generation of content on a device display, such as a mobile device, and overcome problems specifically arising in the realm of computerized display content generation technologies.

172. The claims of the ’044 patent recite an invention that is not merely the routine or conventional use of systems and methods for the computerized generation of content on a device

display. Instead, the invention describes systems for use with devices with authoring tools or Players specific to each device and Applications that are independent of the device.

173. The invention claimed in the ’044 patent offers substantial improvements in device performance and web or application design. For example, the invention allows for faster loading speeds, more efficient and flexible processing of webpage or application dynamic content, and the ability to change a webpage or application more efficiently by editing user settings rather than multiple versions of code. The invention also permits scaling of webpages and dynamic content within the webpage, or applications and elements within the application that contain dynamic content, to most efficiently use the screen space. Taken separately or together, the claim elements of the invention significantly improve the operation of a computer and the process of web design.

174. The invention claimed in the ’044 patent neither preempts all ways for the computerized generation of content on a device display, such as a mobile device, nor preempts the use of all authoring tools or Players for the computerized generation of content on a device display, such as a mobile device, or any other well-known or prior-art technology.

175. Accordingly, each claim of the ’044 patent recites a combination of elements sufficient to ensure that the claim in practice amounts to significantly more than a patent on an ineligible concept.

176. Express Mobile is the assignee and owner of the right, title, and interest in and to the ’044 patent, including the right to assert all causes of action arising under the patent and the right to any remedies for infringement of it.

177. Atlassian has manufactured, used, offered for sale, or sold browser-based website building tools that infringe, either literally or under the doctrine of equivalents, one or more claims of the ’044 patent in violation of 35 U.S.C. § 271(a).

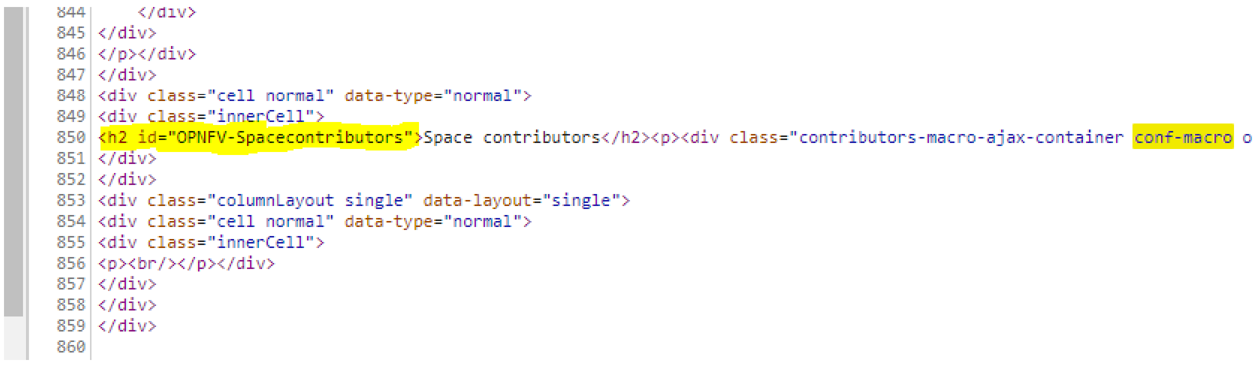
178. Upon information and belief, Atlassian has infringed and continues to infringe at least claim 15 of the ’044 patent.

179. Claim 15 of the ’044 patent recites a method of displaying content on a display of a device platform having a Player and non-volatile computer memory storing symbolic names required for evoking one or more web components each related to a set of inputs and outputs of a web service obtainable over a network, where the symbolic names are character strings that do not contain either a persistent address or pointer to an output value accessible to the web service, where each symbolic name has an associated data format class type corresponding to a subclass of User Interface (“UI”) objects that support the data format type of the symbolic name, and where each symbolic name has a preferred UI object, and an address of the web service, said method comprising: defining a UI object for presentation on the display, where said UI object corresponds to a web component included in the computer memory, where said web component is selected from a group consisting of an input of a web service and an output of the web service where each defined UI object is either: (1) selected by a user of the authoring tool; or (2) automatically selected by the system as the preferred UI object corresponding to a symbolic name of the web component selected by the user of the authoring tool; and selecting the symbolic name from said web component corresponding to the defined UI object, associating the selected symbolic name with the defined UI object, where the selected symbolic name is only available to UI objects that support the defined data format associated with that symbolic name; storing information representative of said defined UI object and related settings in a database; retrieving said information representative of said one or more UI object settings stored in said database; and building an Application consisting of one or more webpage views from at least a portion of said database using the Player, where said Player uses information stored in said database to generate for the display of at least a portion of said one or more webpages, wherein,

when the Application and Player are provided to the device and executed on the device, and when the user of the device provides one or more input values associated with an input symbolic name to an input of defined UI object, (1) the device provides the user provided one or more input values and corresponding input symbolic name to the web service, (2) the web service uses the input symbolic name and the user provided one or more input values for generating one or more output values having an associated output symbolic name, (3) the Player receives the output symbolic name and corresponding one or more output values and provides instructions for a display of the device to present an output value in the defined UI object.

180. Atlassian infringes claim 15 of the ’044 patent through a combination of features that collectively practice each limitation of claim 15. Confluence practices a method for displaying content on a display of a computer device. The content is provided to the user through a web browser on webpages.

181. As shown in the image below, the method includes non-volatile computer memory that stores symbolic names associated with web components – including images, text blocks, tables, and other web components. Those components are related to inputs and outputs of a web service – the Confluence web server – obtainable over a network, such as the Internet or a corporate intranet.


```
    </div>
</div>
</p></div>
</div>
<div class="cell normal" data-type="normal">
<div class="innerCell">
<h2 id="OPNFV-Spacecontributors">Space contributors</h2><p><div class="contributors-macro-ajax-container conf-macro o
</div>
</div>
<div class="columnLayout single" data-layout="single">
<div class="cell normal" data-type="normal">
<div class="innerCell">
<p><br/></p></div>
</div>
</div>
</div>

```

182. The names stored in the computer memory are character strings that do not contain either a persistent address or pointer to an output value accessible to the web service. Each symbolic name has an associated data format class type corresponding to specific user interface objects that support that data format type, and are associated with preferred user interface objects. Confluence's code, including its HTML, JavaScript, and CSS code, associates these symbolic names – represented as element types, classes, and IDs in the browser's Document Object Model – with specific user interface objects.

183. The Confluence computer memory also includes the address of the Confluence web server, a web service.

184. Confluence also includes an authoring tool that lets users define user interface objects for presentation on the web browser. Those user interface objects correspond to standard web components, including tables and images. The web components may be stored in the registry. The authoring tool accesses the computer memory to select a symbolic name corresponding to the web component and associates it with the defined user interface object. *See* Atlassian, *Confluence Support – Macros* (Jan. 8, 2021) (describing various components such as "Gallery" and "Table of Contents" the contents of which, based upon information and belief, can be are stored in a "memory" and retrieved), https://confluence.atlassian.com/doc/macros-139387.html. A particular symbolic name is only available for particular types of user interface objects, and the defined user interface object is automatically selected by the system as the preferred object corresponding to the symbolic name of the web component selected by the user of the authoring tool. For example, a CSS stylesheet may associate all symbolic names with a given type of object, like an image or a table.

185. The authoring tool then stores the symbolic code with the associated settings for the user-defined object in a database. It retrieves that information to produce an "Application" in

the form of device-independent code, including HTML, CSS, and JavaScript code that includes the selected symbolic name. It also produces a Player in the form of device- and platform-dependent code, also including HTML, CSS, and JavaScript code. Unlike the Application code, the Player code is written for specific devices, such as laptops, tablets, or smartphones, or device platforms, such as browsers. The Player code and Application code collaborate to provide the page to the user. *See* ¶ 73, *supra*.

186. When the Application and Player code are executed by the web browser, input provided by the user is sent to the web service, which generates output and sends it to be displayed in a user interface object on the device. The Player interprets the response and updates the user interface objects based on the output it received from the web service.

187. Atlassian was made aware of the ’044 patent and its infringement thereof at least as early as February 6, 2020, when Express Mobile provided notice to Atlassian of the ’044 patent.

188. Upon information and belief, since at least the time Atlassian received notice, Atlassian has induced and continues to induce others to infringe at least claim 15 of the ’044 patent under 35 U.S.C. § 271(b) by, among other things, and with specific intent or willful blindness, actively aiding and abetting others to infringe, including but not limited to Atlassian’s partners, clients, customers, and end users, whose use of Confluence constitutes direct infringement of at least one claim of the ’044 patent. In particular, Atlassian’s actions that aid and abet others such as customers, clients, partners, developers, and end users to infringe include advertising Confluence as a way for employees to collaborate on pages available on the Web. Upon information and belief, Atlassian has engaged in such actions with specific intent to cause infringement or with willful blindness to the resulting infringement because Atlassian has had actual knowledge of the ’044 patent and knowledge that its acts were inducing infringement of

the ’044 patent since at least the date Atlassian received notice that such activities infringed the ’044 patent.

189. By way of example, Atlassian ran graphic advertisements on its website featuring infringing features of Confluence. For example, Atlassian ran an advertisement explaining that Confluence “comes equipped with best-practice templates,” allowing users to easily create webpages. *See* ¶ 81, *supra*.

190. Since February 6, 2020, Atlassian’s infringement of the ’044 patent has been willful.

191. Atlassian’s infringement has damaged and continues to damage and injure Express Mobile.

<u>COUNT XI – INFRINGEMENT OF U.S. PATENT NO. 9,928,044</u> **<u>(Trello)</u>**

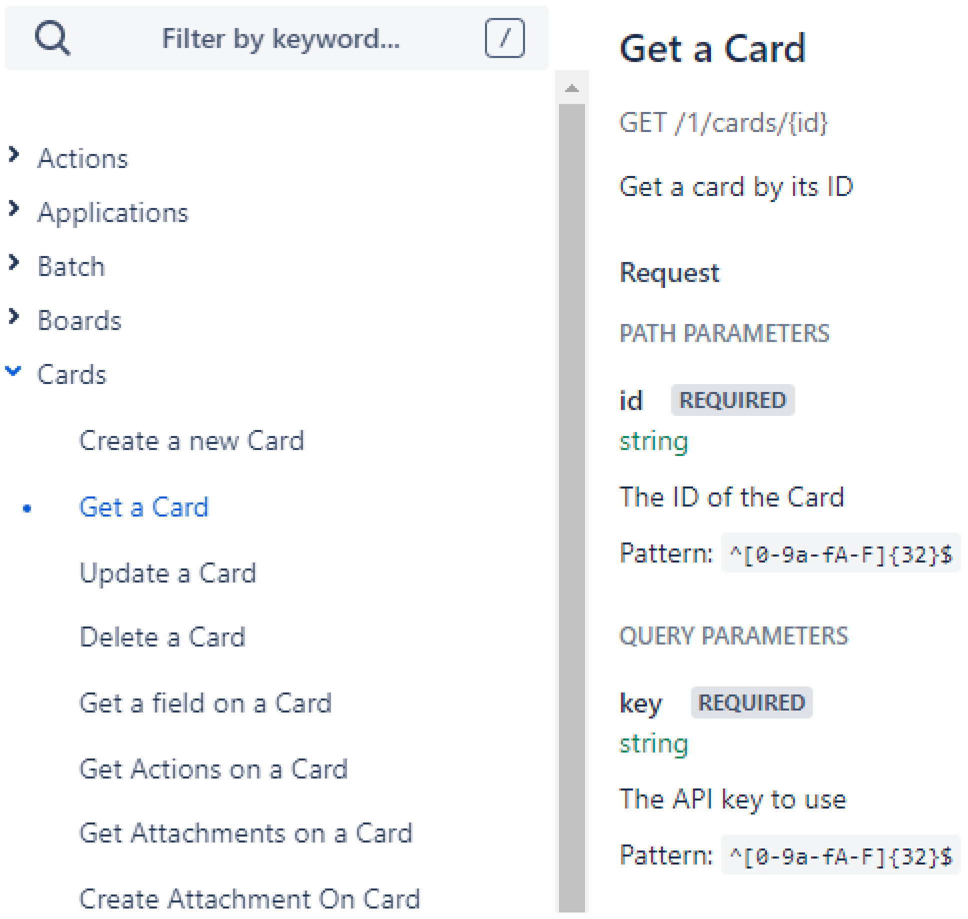
192. The allegations set forth in the foregoing paragraphs 1 through 191 are incorporated into this Eleventh Claim for Relief.

193. Atlassian has manufactured, used, offered for sale, or sold browser-based website building tools that infringe, either literally or under the doctrine of equivalents, one or more claims of the ’044 patent in violation of 35 U.S.C. § 271(a).

194. Upon information and belief, Atlassian has infringed and continues to infringe at least claim 15 of the ’044 patent. Trello practices a method for displaying content on a display of a computer device. The content is provided to the user through a web browser on webpages.

195. The method includes non-volatile computer memory that stores symbolic names associated with web components – including images, text blocks, cards, and other web components. These components are related to inputs and outputs of a web service – the Trello web server – obtainable over a network, such as the Internet or a corporate intranet.

196. The names stored in the computer memory are character strings that do not contain either a persistent address or pointer to an output value accessible to the web service. As shown in the image below, and by way of example, each symbolic name has an associated data format class type corresponding to specific user interface objects that support that data format type, and are associated with preferred user interface objects. Trello's code, including its HTML, JavaScript, and CSS code, associates these symbolic names – represented as element types, classes, and IDs in the browser's Document Object Model – with specific user interface objects.



Get a Card

GET /1/cards/{id}

Get a card by its ID

Request

PATH PARAMETERS

id REQUIRED
string

The ID of the Card

Pattern: `^[0-9a-fA-F]{32}$`

QUERY PARAMETERS

key REQUIRED
string

The API key to use

Pattern: `^[0-9a-fA-F]{32}$`

197. The Trello computer memory also includes the address of the Trello web server, a web service.

198. Trello also includes an authoring tool that lets users create cards displayed as user interface objects for presentation on the web browser. These user interface objects correspond to combinations of web components may be stored in the registry. The authoring tool accesses the computer memory to select a symbolic name corresponding to the web component and associates it with the defined user interface object. A particular symbolic name is only available for particular types of user interface objects, and the defined user interface object is automatically selected by the system as the preferred object corresponding to the symbolic name of the web component selected by the user of the authoring tool. For example, a CSS stylesheet may associate all symbolic names with a given type of object like the color of a label on a card.

199. The authoring tool then stores the symbolic code with the associated settings for the user-defined object in a database. It retrieves that information to produce an "Application" in the form of device-independent code, including HTML, CSS, and JavaScript code that includes the selected symbolic name. It also produces a Player in the form of device- and platform-dependent code, also including HTML, CSS, and JavaScript code. Unlike the Application code, the Player code is written for specific devices, such as laptops, tablets, or smartphones, or device platforms, such as browsers. The Player code and Application code collaborate to provide the page to the user.

200. When the Application and Player code are executed by the web browser, input provided by the user is sent to the web service, which generates output and sends it to be displayed in a user interface object on the device. The Player interprets the response and updates the user interface objects based on the output it received from the web service.

201. Atlassian was made aware of the '044 patent and its infringement thereof at least as early as February 6, 2020, when Express Mobile provided notice to Atlassian of the '044 patent.

202. Upon information and belief, since at least the time Atlassian received notice, Atlassian has induced and continues to induce others to infringe at least claim 15 of the ’044 patent under 35 U.S.C. § 271(b) by, among other things, and with specific intent or willful blindness, actively aiding and abetting others to infringe, including but not limited to Atlassian’s partners, clients, customers, and end users, whose use of Trello constitutes direct infringement of at least one claim of the ’044 patent. In particular, Atlassian’s actions that aid and abet others such as customers, clients, partners, developers, and end users to infringe include advertising Trello as a way for employees to collaborate on pages available on the Web. Upon information and belief, Atlassian has engaged in such actions with specific intent to cause infringement or with willful blindness to the resulting infringement because Atlassian has had actual knowledge of the ’044 patent and knowledge that its acts were inducing infringement of the ’044 patent since at least the date Atlassian received notice that such activities infringed the ’044 patent.

203. By way of example, Atlassian ran graphic advertisements on its website featuring infringing features of Trello. For example, Atlassian ran an advertisement showing that users can collaborate with others to organize cards “in a fun, flexible, and rewarding way.”

204. Another advertisement showed Trello’s visual interface for creating, editing, and moving cards in a visual manner. *See* ¶94, *supra*; Emily Whitten, *How To Use Trello and Indeed To Hire a Dream Team* (Nov. 18, 2019), https://blog.trello.com/indeed-template-story.

205. Since February 6, 2020, Atlassian’s infringement of the ’044 patent has been willful.

206. Atlassian’s infringement has damaged and continues to damage and injure Express Mobile.

COUNT XII – INFRINGEMENT OF U.S. PATENT NO. 9,928,044 (JIRA)

207. The allegations set forth in the foregoing paragraphs 1 through 206 are incorporated into this Twelfth Claim for Relief.

208. Atlassian has manufactured, used, offered for sale, or sold browser-based website building tools that infringe, either literally or under the doctrine of equivalents, one or more claims of the ’044 patent in violation of 35 U.S.C. § 271(a).

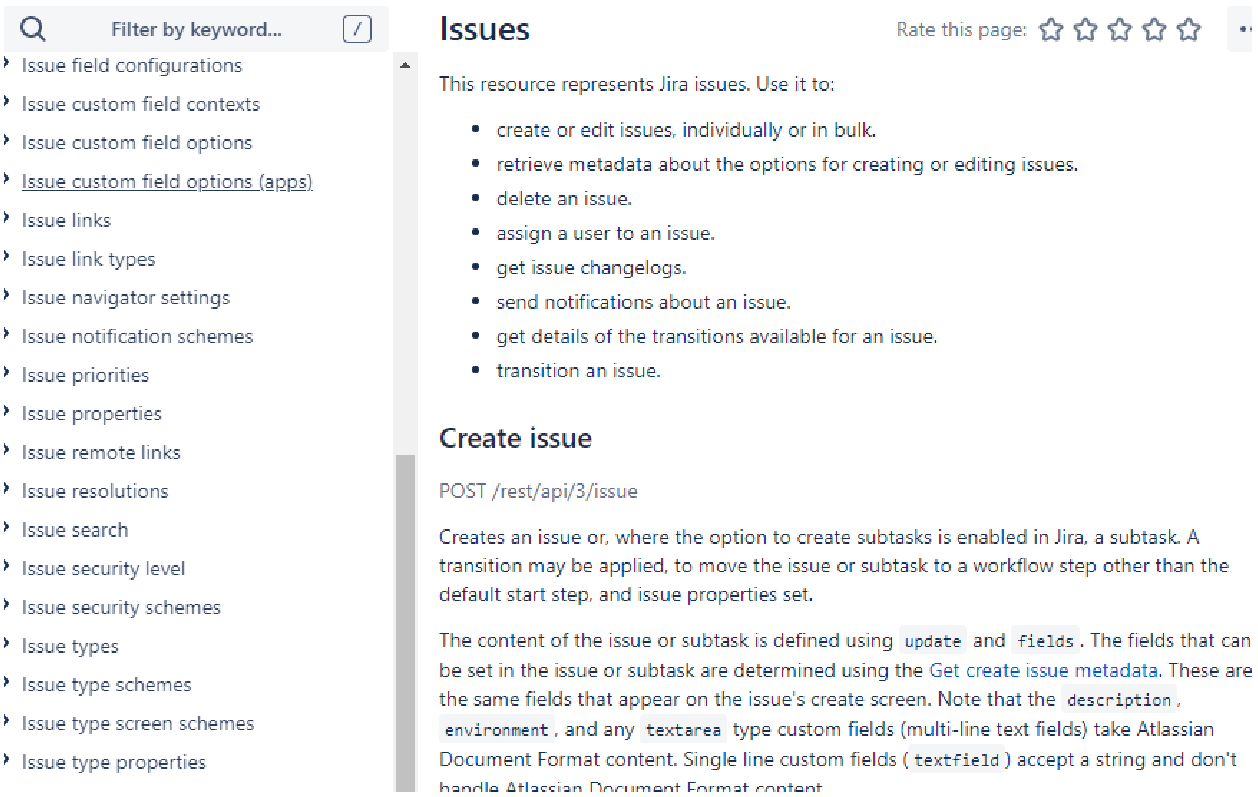
209. Upon information and belief, Atlassian has infringed and continues to infringe at least claim 15 of the ’044 patent. JIRA practices a method for displaying content on a display of a computer device. The content is provided to the user through a web browser on webpages.

210. The method includes non-volatile computer memory that stores symbolic names associated with web components – including images, issue fields, and other web components. These components are related to inputs and outputs of a web service – the JIRA web server – obtainable over a network, such as the Internet or a corporate intranet.

211. The names stored in the computer memory are character strings that do not contain either a persistent address or pointer to an output value accessible to the web service. Each symbolic name has an associated data format class type corresponding to specific user interface objects that support that data format type, and are associated with preferred user interface objects. JIRA’s code, including its HTML, JavaScript, and CSS code, associates these symbolic names – represented as element types, classes, and IDs in the browser’s Document Object Model – with specific user interface objects.

212. The JIRA computer memory also includes the address of the JIRA web server, a web service.

213. JIRA also includes an authoring tool that lets users create issue pages including user interface objects for presentation on the web browser. As shown below, user interface objects correspond to, by way of example, issue properties, such as "resolution," and "link types." The web components may be stored in the registry. The authoring tool accesses the computer memory to select a symbolic name corresponding to the web component and associates it with the defined user interface object. A particular symbolic name is only available for particular types of user interface objects, and the defined user interface object is automatically selected by the system as the preferred object corresponding to the symbolic name of the web component selected by the user of the authoring tool. For example, a CSS stylesheet may associate all symbolic names with a given type of object like an image or a table.



Issues

This resource represents Jira issues. Use it to:

- create or edit issues, individually or in bulk.
- retrieve metadata about the options for creating or editing issues.
- delete an issue.
- assign a user to an issue.
- get issue changelogs.
- send notifications about an issue.
- get details of the transitions available for an issue.
- transition an issue.

Create issue

POST /rest/api/3/issue

Creates an issue or, where the option to create subtasks is enabled in Jira, a subtask. A transition may be applied, to move the issue or subtask to a workflow step other than the default start step, and issue properties set.

The content of the issue or subtask is defined using `update` and `fields`. The fields that can be set in the issue or subtask are determined using the Get create issue metadata. These are the same fields that appear on the issue's create screen. Note that the `description`, `environment`, and any `textarea` type custom fields (multi-line text fields) take Atlassian Document Format content. Single line custom fields (`textfield`) accept a string and don't handle Atlassian Document Format content

214. The authoring tool then stores the symbolic code with the associated settings for the user-defined object in a database. It retrieves that information to produce an "Application" in

the form of device-independent code, including HTML, CSS, and JavaScript code that includes the selected symbolic name. It also produces a Player in the form of device- and platform-dependent code, also including HTML, CSS, and JavaScript code. Unlike the Application code, the Player code is written for specific devices, such as laptops, tablets, or smartphones, or device platforms, such as browsers. The Player code and Application code collaborate to provide the page to the user.

215. When the Application and Player code are executed by the web browser, input provided by the user is sent to the web service, which generates output and sends it to be displayed in a user interface object on the device. The Player interprets the response and updates the user interface objects based on the output it received from the web service.

216. Atlassian was made aware of the ’044 patent and its infringement thereof at least as early as August 31, 2020, when Express Mobile provided notice to Atlassian of the ’044 patent.

217. Upon information and belief, since at least the time Atlassian received notice, Atlassian has induced and continues to induce others to infringe at least claim 15 of the ’044 patent under 35 U.S.C. § 271(b) by, among other things, and with specific intent or willful blindness, actively aiding and abetting others to infringe, including but not limited to Atlassian’s partners, clients, customers, and end users, whose use of JIRA constitutes direct infringement of at least one claim of the ’044 patent. In particular, Atlassian’s actions that aid and abet others such as customers, clients, partners, developers, and end users to infringe include advertising JIRA as a way for employees to collaborate on pages available on the Web. Upon information and belief, Atlassian has engaged in such actions with specific intent to cause infringement or with willful blindness to the resulting infringement because Atlassian has had actual knowledge

of the ’044 patent and knowledge that its acts were inducing infringement of the ’044 patent since at least the date Atlassian received notice that such activities infringed the ’044 patent.

218. By way of example, Atlassian ran graphic advertisements on its website featuring infringing features of JIRA. For example, Atlassian ran an advertisement showing that users can create a webpage summarizing data entered into the platform “all in a few clicks.”

219. Other JIRA advertisements touted JIRA’s interface allowing for the easy creation of “flexible” and “customizable” pages.

220. Since August 31, 2020, Atlassian’s infringement of the ’044 patent has been willful.

221. Atlassian’s infringement has damaged and continues to damage and injure Express Mobile.

JURY DEMAND

Pursuant to Rule 38 of the Federal Rules of Civil Procedure, Express Mobile demands a trial by jury on all issues triable as such.

PRAYER FOR RELIEF

WHEREFORE, Express Mobile demands judgment for itself and against Atlassian as follows:

A. An adjudication that Atlassian has infringed the ’397, ’755, ’287, and ’044 patents;

B. An award of damages to be paid by Atlassian adequate to compensate Express Mobile for Atlassian’s past infringement of the ’397, ’755, ’287, and ’044 patents, and any continuing or future infringement through the date such judgment is entered, including interest, costs, expenses, and an accounting of all infringing acts including, but not limited to, those acts not presented at trial;

C. An award of a reasonable ongoing royalty for future infringement of the ’755, ’287, and ’044 patents;

D. A declaration that this case is exceptional under 35 U.S.C. § 285, and an award of Express Mobile’s reasonable attorneys’ fees; and

E. An award to Express Mobile of such further relief at law or in equity as the Court deems just and proper.

Dated: January 12, 2021

Respectfully submitted,

/s/ Robert Christopher Bunt

Robert Christopher Bunt
State Bar No. 00787165
Charles Ainsworth
State Bar No. 00783521
PARKER, BUNT & AINSWORTH, P.C.
100 E. Ferguson, Suite 418
Tyler, TX 75702
(903) 531-3535 (telephone)
rcbunt@pbatyler.com
charley@pbatyler.com

Steven F. Molo (*pro hac vice*)
NY Bar No. 4221743
Ben Quarmby (*pro hac vice*)
NY Bar No. 4298725
Leonid Grinberg (*pro hac vice*)
NY Bar No. 5652458
MOLOLAMKEN LLP
430 Park Avenue
New York, NY 10022
(212) 607-8160 (telephone)
(212) 607-8161 (fax)
smolo@mololamken.com
bquarmby@mololamken.com
lgrinberg@mololamken.com

Rayiner I. Hashem (*pro hac vice*)
D.C. Bar No. 1046830
Sarah J. Newman (*pro hac vice*)
D.C. Bar No. 1047210
Benjamin T. Sirolly (*pro hac vice*)
D.C. Bar No. 1022426
MOLOLAMKEN LLP
The Watergate, Suite 500
600 New Hampshire Avenue, N.W.
Washington, D.C. 20037
(202) 556-2000 (telephone)
(202) 556-2001 (fax)
rhashem@mololamken.com
snewman@mololamken.com
btsirolly@mololamken.com

Timothy Devlin (*pro hac vice*)
Delaware Bar No. 4241
DEVLIN LAW FIRM LLC
1526 Gilpin Ave.
Wilmington, DE 19806
(302) 449-9010 (telephone)
(302) 353-4251 (fax)
tdevlin@devlinlawfirm.com

Attorneys for Plaintiff Express Mobile, Inc.

CERTIFICATE OF SERVICE

I hereby certify that, on January 12, 2021, I electronically filed the foregoing with the Clerk of the Court using CM/ECF, which will send notification of such filing to all counsel of record who are registered with the CM/ECF system.

/s/ Robert Christopher Bunt
ROBERT CHRISTOPHER BUNT