Squire: A Web Based Collaborative Editor

benz5834, bolt1003, carl7595, guan2264, jank6275, knic1468, mora5651, ratc8795

January 26, 2016

Contents

Chapter 1

Application Domain Specification

1.1 Program Premise (benz5834, carl7595, guan2264, jank6275)

Think of squire as a collaborative version of "pastebin.com" that compiles Java code. Squire will be a web-based collobarative text editor that compiles java code in throwaway "rooms".

1.1.1 Core Features (carl7595)

Web based All editing and collaboration is done in the browser Limited IDE like features Code word color coding Parenthesis counting Missing end of line detection Collaborative tools Way to tell who coded what (color, footnote, etc) Built in chat functionality Seperate coding spaces for projects Access is controlled by user account Compile code and open it on user's computer Rooms are self destructing After some period of inactivity, coding space is recycled without user intervention

1.1.2 ?? (guan2264)

1.1.3 Stuff (jank6275)

security - Security Nightmare? (jank6275) docker and dokku Code will be compiled and run in each container. The project will run on a virtual machine that creates virtual machines for each project. If the user wants to hack their own container... so be it. storage - Where will data be saved? (benz5834) Data will be stored in MySQL. organization - Rooms lol? (benz5834) Domain -¿ User -¿ Project -¿ Files Unused projects will be deleted after 30 days. Multiple files ("pages") allowed per project. Ability to import and export files. execution - Where code will run? Server based in container, or clientside? Execute editor client side thru Ace. Execute compiled code client side. compiler Compile code server side. Built-in compiler/interpreter to show input/output. Built-in, in-line compile-time/interpret-time debugging. language Written in PHP5 - tried and true, required to know for web development. Java - language we support for online compile? browser based?

yes The back-end will be written in PHP The front end will be HTML5, JavaScript, Ace Editor.

1.1.4 Security Nightmare (jank6275)

The issue of security comes to mind every time compiling and executing an outside application in a secure area. Rooting or destroying data and infrastruture is trivial when you are freely alowed to execute arbitrary Java. In order to allow for this functionality we must impliment some type of security strategy. We could limit or block functionality in Java which is a cat and mouse game that breaks useability. We could compile and execute code on the client side. This would require a compiler written in JavaScript and cause compatibility issues on certain clients with reduced permissions or an exotic build environment. Or we could just Docker.

Docker allows us to create containers that contain not only the user's build enviornment, but the server infrastructure to host their project. Containers include the application and all of its dependencies, but share the kernel with other containers. Simply put, users can destroy their own containers but nobody elses. Obviously we will want to harden against known and obvious vectors, but at least any damage a user can do is limited to their own project. Containerization also allows us to take snapshots so that even if something goes wrong, we can always revert back to a working state.

1.1.5 Storage and Organization (benz5834)

The squire program will have a top down directory structure. Under the top domain, each individual registered user will have the ability to create projects (rooms) where they can build their program. Inside the room the user will be able to create any number of files or pages to keep track of their project. They will also have the ability to import and export files to and from their own machine and the project. After creation the user will be able to switch between projects and create and delete pages from within the project itself. They will have the ability to share their projects and invite other users to read or edit their work. All of the user data, as well as each users relevant project and page information, as well as who has access to each project will be stored and maintained by a MySQL database.

1.2 Application Domain Study (knic1468, ratc8795, bolt1003, mora5651)

An examination and test of a similar (previous and/or manual) system.

1.3 Use Case Descriptions

1.3.1 Open project chat (jank6275)

Actors:	User
Goals:	To open the project chat window.
Pre-conditions:	User must be registered, signed in, and in editor Mode.
Summary:	User clicks on open project chat and the chat opens, displaying chat history and updating when needed.
Related use cases:	Join global chat.
Steps:	1. User clicks open project chat. 2. Chat is notified that user has joined. 3. System displays project chat window.
Alternatives:	None.
Post-conditions:	None.

1.3.2 Open global chat (jank6275)

Actors:	User
Goals:	To open the global chat window.
Pre-conditions:	User must be registered, signed in, and anywhere on website.
Summary:	User clicks on open global chat and the chat opens, displaying chat history and updating when needed.
Related use cases:	Join project chat.
Steps:	1. User clicks open global chat. 2. Chat is notified that user has joined. 3. System displays global chat window.
Alternatives:	None.
Post-conditions:	None.

1.3.3 Close project chat (jank6275)

Actors:	User
Goals:	To close the project chat window.
Pre-conditions:	User must be registered, signed in, and in editor Mode.
Summary:	User clicks on close project chat and the chat window closes.
Related use cases:	Close global chat.
Steps:	1. User clicks close project chat. 2. Chat is notified that user has left. 3. Client closes project chat window.
Alternatives:	None.
Post-conditions:	None.

1.3.4 Close global chat (jank6275)

Actors:	User
Goals:	To close the global chat window.
Pre-conditions:	User must be registered, signed in, and anywhere on website.
Summary:	User clicks on open global chat and the chat opens, displaying chat history and updating when needed.
Related use cases:	Close project chat.
Steps:	1. User clicks close global chat. 2. Chat is notified that user has left. 3. Client closes global chat window.
Alternatives:	None.
Post-conditions:	None.

1.3.5 Write to project chat (jank6275)

Actors:	User
Goals:	To send text to project chat.
Pre-conditions:	User must be registered, signed in, a project opened, with the project chat window open, and the text box selected.
Summary:	User clicks in the project chat text box and then types a message then either presses enter or clicks the submit button. The text is displayed to all users in the chat, including the user.
Related use cases:	Write to global chat.
Steps:	1. User clicks in the project chat box. 2. User types a message and then presses enter or clicks submit button. 3. Message is relayed to all clients with project chat open. 4. Message is displayed.
Alternatives:	None.
Post-conditions:	None.

1.3.6 Write to global chat (jank6275)

Actors:	User
Goals:	To send text to global chat.
Pre-conditions:	User must be registered, signed in, anywhere on website, with the global chat window open, and the text box selected.
Summary:	User clicks in the global chat text box and then types a message then either presses enter or clicks the submit button. The text is displayed to all users in the chat, including the user.
Related use cases:	Write to project chat.
Steps:	1. User clicks in the global chat box. 2. User types a message and then presses enter or clicks submit button. 3. Message is relayed to all clients with global chat open. 4. Message is displayed.
Alternatives:	None.
Post-conditions:	None.

1.3.7 Modify chat font (jank6275)

Actors:	User
Goals:	To change a users font style inside the global and project chat.
Pre-conditions:	User must be registered, signed in, the user settings window opened, and the chat settings tab open.
Summary:	The user clicks the settings menu and changes their font style for both the project and global chat through a drop down box of available fonts.
Related use cases:	Modify chat color.

Steps:

1. User clicks the settings menu.
2. User clicks chat settings tab.
3. User clicks chat font drop down box.
4. User clicks desired font.
5. User clicks save.
6. The user's selection is saved in the database.
7. All further chat messages will use the selected font.

Alternatives:	None.
Post-conditions:	None.

1.3.8 Modify chat color (jank6275)

Actors:	User
Goals:	To change a users font color inside the global and project chat.
Pre-conditions:	User must be registered, signed in, the user settings window opened, and the chat settings tab open.
Summary:	The user clicks the settings menu and changes their font color for both the project and global chat through a drop down box of available colors.
Related use cases:	Modify chat font.

Steps:

1. User clicks the settings menu.
2. User clicks chat settings tab.
3. User clicks chat color drop down box.
4. User clicks desired color.
5. User clicks save.
6. The user's selection is saved in the database.
7. All further chat messages from the user will use the selected color.

Alternatives:	None.
Post-conditions:	None.

1.3.9 Create Project (bolt1003)

Actors:	Users of sQuire.
Goals:	Create a Project.
Pre-conditions:	The user is logged in.
Summary:	User opens the project manager and creates a project.
Related use cases:	None.
Steps:	1. User selects the project manager from the main menu. 2. User selects open. 3. Create new project is selected from the task bar. 4. A name is choosen for the project. 5. A location is selected for the project. 6. Language is selected from a drop down menu. 7. User clicks finish.
Alternatives:	None.
Post-conditions:	None.

1.3.10 Choose Project (bolt1003)

Actors:	Users of sQuire.
Goals:	Open a Project.
Pre-conditions:	A project has been created and the user is logged in.
Summary:	User looks through the list of project and selects the desired project.
Related use cases:	None.
Steps:	1. User selects the project manager from the main menu. 2. User selects open. 3. User selects the desired project. 4. User clicks open in the list of options.
Alternatives:	None.
Post-conditions:	None.