sQuire: A Collaborative Software Development Tool

jank6275, mora5651, boss2849, bolt1003, gall7417, brec9824, snev7821, mars2681

February 10, 2016

Contents

1 Introduction **2**
1.1 Program Premise 2
1.2 Use Case Overview (jank6275) 2

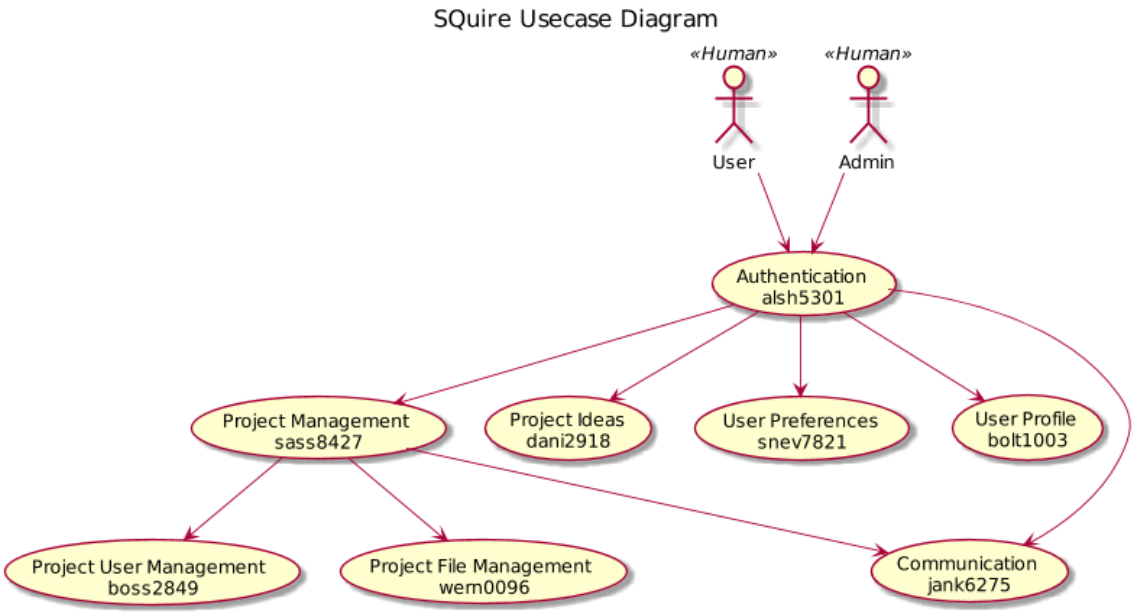
2 Requirements Documentation **4**
2.1 Functional Requirements 4
2.1.1 A 4
2.1.2 P 4
2.1.3 P 5
2.1.4 U 5
2.1.5 P 5
2.1.6 C 6
2.1.7 S 6
2.1.8 C 7
2.2 Non-Functional Requirements 7
2.3 Unused Requirements 9

Chapter 1

Introduction

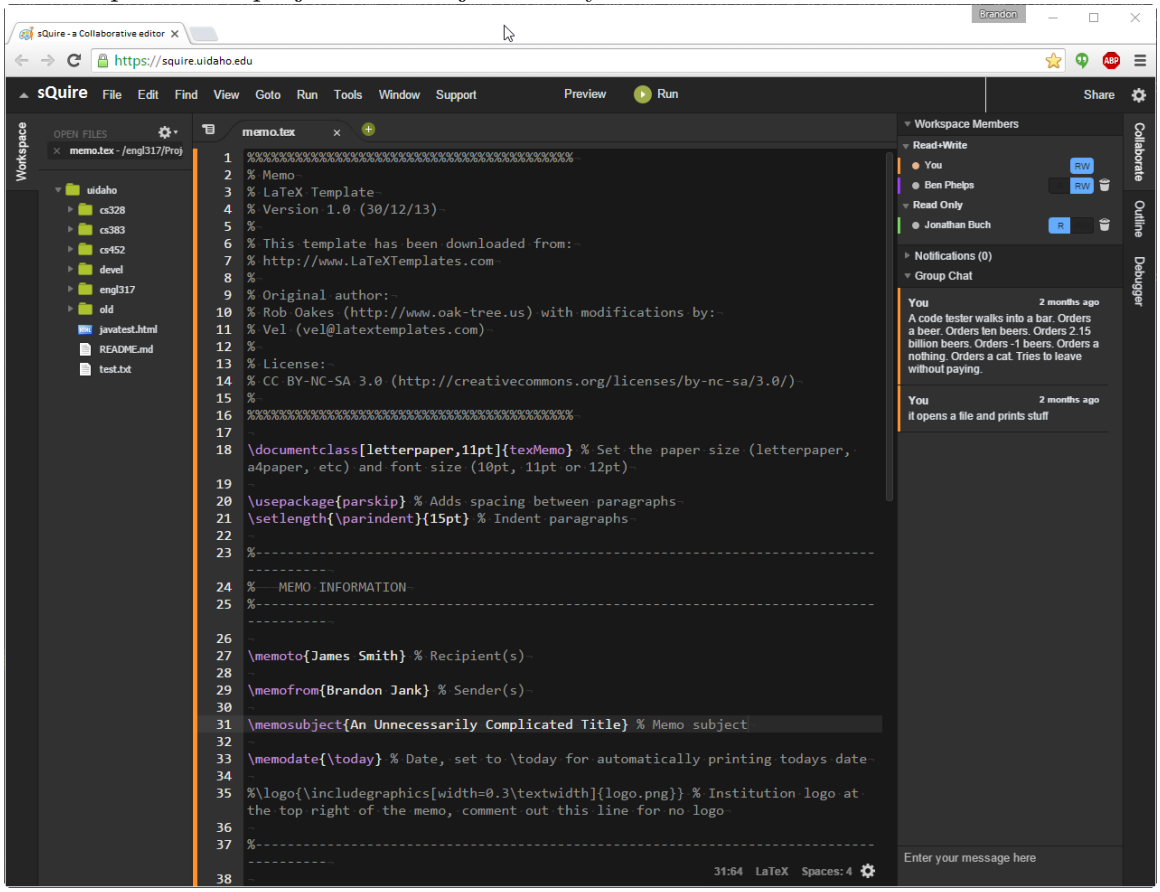
1.1 Program Premise

1.2 Use Case Overview (jank6275)



A usecase diagram that shows all of sQuireś features.

Figure 1.1: Squire will be a web-based collaborative software development environment with a project development center. Squire will allow multiple users to edit files and communicate in real time. First, projects are stubbed out by a user and then other users can join and/or vote to support for their favorite projects. After a certain amount of support, planning, and documentation is reached for a project, the project becomes a fully fleged project and then community development can start. Think "kickstarter for code" where people pledge their help with the project and not just money.



Chapter 2

Requirements Documentation

2.1 Functional Requirements

Functional requirements will specify a behaviour or function, for example “Display the name, total size, available space and format of a flash drive connected to the USB port” Other examples are “add customer” and “print invoice”. Some of the more typical functional requirements include:

2.1.1 A

uthentication (mars2681)

- First, the user can input their email address and password in order to sign into their account. If they do not have an account, they can sign up by providing an email address and a password. A confirmation email will then be sent to their email address, which they can use to confirm and access their account. If the user forgets their password, they can click forgot password and input their email address. The system with confirm an email addresses associated with an account, and email the user a temporary password.

2.1.2 P

roject Management (mora5651)

- Admin controls features.
- Public and private sharing of project.
- Controlled reading and writing permissions to users.
- Password controls.
- File management.

2.1.3 P

roject Ideas (snev7821)

- Forum to browse potential projects
- Up- and down-votes for project selection
- Different ways to sort projects (date, projected team size, votes)
- Ability to post a project
- Ability to delete a project (but only by project author)
- Convert potential project stub into active working project

2.1.4 U

ser Profile (brec9824)

- Viewable profile by other users and by oneself
- Includes editable email, profile image, password, and personal bio.
- Setting for public, friends only, or private viewing of online status, email address, personal bio, project ownerships, project memberships, and friends list.
- Option for settings users prefered color and shape to be displayed in projects if available.
- Option to have account deleted.
- Direct access to projects that are listed under project ownerships and project memberships.

2.1.5 P

roject File Editor (jank6275)

- The editor will be multi-user, up to 32 users can share an IDE session.
- Text will be underlined in the userś color, if they edited/created that text.
- The line number will be highlighted in the users color, if they created the line.
- Each line edited by a user should be saved to create a edit history for each user in a document.
- Users will be able to edit the same document concurrently.
- Any user can save the document at any time.
- The current location of any userś cursor can be quickly jumped to by any user with the project open.

2.1.6 C

ommunication (bolt1003)

- Built-in, global text chat per project.
- Private communications between other users.
- A friends list with friend status icons and avatar.
- A global list of current mememembers in the project.
- A list of current users working on the project.
- A dialog with the user's name and file history when hovering over their icon.
- Text chat will be built using standard protocols such as XMPP or IRC.
- Allows the use of third-party chat clients.

2.1.7 S

ecurity (gall7417)

- Resource defense strategy that includes not subjecting any sQuire server to becoming unresponsive due to a runaway program, and not allowing any sQuire server to give up shell access via an executing program in sQuire.
- Require authentication to access all user files and user information.
- Ensure confidentiality of all user information.
- Use password hashing on a trusted system to ensure password privacy.
- Hide password entry on user interface.
- Allow password change and reset in case of compromise.
- Mitigate security threats by testing against common abuse cases and vulnerabilities.
- Validate user integrity before any processing is performed.
- Ensure proper character sets for all input given.
- All validation failures must result in rejection.
- Implement a force halt procedure for runaway programs.
- Establish system inactivity timeout after arbitrary amount of time.
- Enforce authorization controls on all system requests.
- Restrict access to resources and files outside of the users given resources.
- Deny access to security protocols and configurations.

2.1.8 C

ompiler (boss2849)

- Capable of supporting editing, compilation, and execution of Java programs. Programs to be composed as projects and support multiple source code files across multiple directories within a shared top-level directory.
- Design as a 'plugin' to the system - initial will be a Java compiler, but allow new compiler plugins to be written for other languages.
- Compile code for execution within IDE.
- Compile code and package to a JAR.
- Compile file, file and dependents, project sub-modules, or entire project.
- Smart compilation - recompile only what has been changed.
- Allow temporary code freeze before compilation.
- Cache snapshots of code on compilation.

2.2 Non-Functional Requirements

Non-functional requirements cover all the remaining requirements which are not covered by the functional requirements. They specify criteria that judge the operation of a system, rather than specific behaviours, for example: "Modified data in a database should be updated for all users accessing it within 2 seconds." Some typical non-functional requirements are:

- Squire should perform good, and stuff. (jank6275)
- Scalability is important to keep in mind during development. The system should be designed in such a way that will easily and reliably scale to accommodate a growing user base. One method of dealing with scalability would be allowing the core system to reactively spawn new slaves to aid in computational needs, such as compilation (as that will be more resource intensive the user base grows). (boss2849)
- Limited space for projects that haven't been initiated (enough for documentation, images, etc.) (boss2849)
- Reactively increasing capacity for projects proportional to the absolute needs of the projects. (no fluff) (boss2849)
- Encourage developers to store large files elsewhere, e.g. GitHub LFS, AWS, etc. (boss2849)
- Since sQuire is web based, downtime for the servers most be kept to a minimum and be no more then once or twice a week for a few hours. This downtime will allow for database uprgrades and backups. (brec9824)

- sQuire will be a web application and will leverage the strengths of web technologies to make it reliable. sQuire will use a webhost such as, Amazon Web services. The webhost's infrastructure provides, redunancy for hardware, power and internet service. (bolt1003)
- SQuire will have the ability for the user to save on sQuire's database for recoverability isurance. It will also incorporate autosave feature. (mora5651)
- Since sQuire is a web based application running on a webhost. There are many maintenance tools that can be used to track performance. Maintainability of sQuire will also include regular backup schedules, speed test, and security monotoring. (mora5651)
- Running sQuire as a web application allows it to quickly and easily rollback to a previous version or rollforward to a new version. This allows for rapid bug fixing. Infrastructure is redunane so equipment can be taken offline for repair without interupting the users. (bolt1003)
- Docker, dokku-alt, docker, dokku, docker. (jank6275)
- Before the user can create an account, they have to agree with several laws. They must comply with the DMCA, and agree that any projects they create that are subjected to copyright violation isnt associated with sQuire. They must also confirm they are over 18 years of age, complying with COPPA law. (mars2681)
- Will include monitoring of subsystems to collect and report performance data. (brec9824)
- Subsystem monititory must be highly efficient and take up less than 10% of resources to keep noticeability to a minimum. (brec9824)
- Subsystems must be modular to allow for efficient monitoring, implementation of subsystems, and the addition of new subsystems. (brec9824)
- sQuire will run as a browser application in google chrome. (gall7417)
- sQire will run on virtually all modern processors. (gall7417)
- sQuire will use tcp to for reliable data communications. (gall7417)
- sQuire will use error checking upon large changes to ensure no drastic data corruption occurs. (gall7417)
- sQuire will use a history/autosave feature in case of data loss. (gall7417)
- Provides code compilation and file system for users with limited resources. (snev7821)
- Easy to learn system, helpful syntax highlighting. (snev7821)
- More satisfying than competitors products, because of social/kickstarter aspect. (snev7821)
- Provides github integration. (snev7821)
- Runs on any of the main browsers (chrome, firefox, IE, safari). (snev7821)
- Due to web based design, works on lower end machines. (snev7821)

2.3 Unused Requirements

- Since multiple people might edit a given line over its history, support for past history or anyhow multiple persons in this indication is strongly recommended.
- Teams should decide if voice or video is essential. Voice, if supported, might be restricted as to number of listeners and/or number of simultaneous transmitters. Video, if supported, might be restricted as to number of viewers and/or number of simultaneous. - NO, we have skype for a reason.