May15-31

SENIOR DESIGN I - CODERLAB

Project Plan

Jake Bertram
Erich Kuerschner
Bryan Passini
Daniel Smith
Kyle Tietz
Jacob Wallraff

TABLE OF CONTENTS

1 PROJECT STATEMENT

The internet has revolutionized the way we communicate, do business, and most importantly how we learn. Our mission is to reinvent the way students learn to code. By utilizing existing software tools and the power of the web, we propose a solution which will improve the experience of learning and teaching code; its name is CoderLab. The goal of our project is to create a browser-based real-time collaborative code editing solution geared primarily for a classroom environment which creates an enhanced teacher-student experience when learning new coding concepts and languages.

2 DESIGN GOALS

This project will be composed of a user web interface for code development as well as a server backend responsible for compiling code, maintaining files, and managing user sessions.

A completed project will include the following:

- Web-based code editor, including
    - Multiple editor tabs
    - Basic code completion (suggesting variable/function names)
    - Syntax highlighting
    - Code compilation support for C, Java, and MATLAB
    - Support for multiple file compilation
- Collaborative aspect to the editor, including
    - Multiple users simultaneously editing the same code
    - Support for multiple displayed cursors
    - System for inviting others to collaborate on a project
- An integrated shell, capable of
    - Accepting input from a user
    - Relaying the input to a running program
    - Compiling and executing the code
    - Displaying output from the program
    - Allowing basic terminal commands useful for development
- A file system, allowing
    - Persistent code project storage
    - Ownership of files
    - Directory structure
- Security, in the form of
    - User authentication through ISU single sign-on
    - Permissions system to restrict editing and collaboration
    - Preventing malicious code being run on the virtual shell

3 MARKET/LITERATURE SURVEY

After searching the market for similar products to CoderLab, we found that there are several tools that already exist: CoderPad, FirePad, EtherPad, Cloud9, CodeBunk and IDEOne. Each of these tools differs in slightly different way that makes each desirable for different reasons. We have taken a look at each of these tools to get an idea of what we want and do not want to include in CoderLab. CoderPad provides a real time code execution window and a user list window that we would like to replicate in our product. Cloud9 provides a full blown browser based IDE to its users. This is too much for our product; we do not want to overwhelm new students to programming with a complex IDE. We want to keep it simple so students can focus on coding. While some of these tools are very similar to what we envision CoderLab to be, each of them lacks one important item: integration with ISU. We want our product to be tailored specifically for ISU students and faculty.

4 REQUIREMENTS

4.1 FUNCTIONAL REQUIREMENTS

Project Element	Requirement
Website UI	The website shall be a single page application.
	The website shall be easy to navigate – the user shall be able to navigate between views with no more than one click.
	The user shall be presented with all required windows upon logging in: code collaboration window, user panel, file panel, and shell window.
	A room administrator shall have access to the user privileges panel.
	Each window shall be resizable to suit the user's tastes and monitor resolution.
	Each window shall be collapsible.
Shell	The shell shall be based on the Unix shell and shall support a subset of Unix commands.
	The shell shall be an interactive terminal, allowing the user to provide input to running code and view the output.
Coding Environment	The editor shall support syntax highlighting for supported languages.
	The editor shall display text cursors for all users with editing privileges currently editing the document.
	There shall be an option to select the language to use for a created project.
	The editor shall support basic code completion.
User Panel	The panel shall list all the users with whom the current project has been shared.
	The panel shall visually distinguish users who are online from those who are offline.
	The panel shall visually distinguish read, write, and execute permissions for users.
Files	The user shall be able to create and rename files.
	The user's files shall persist beyond a classroom session.
	The owner of a file shall be able to delete a file.

	The user shall be able to import and export files.
File Panel	The panel shall list all files in the current project.
Authentication	The website shall require users to log on using ISU's single sign-on system.

4.2 NON-FUNCTIONAL REQUIREMENTS

Quality	Description
Ease of Use	User is able to navigate the site without the need for assistance from teachers or peers.
Fault Tolerance	The web application shall maintain its functionality in the event of a classroom session crashing.
Documentation	The website will contain a help section.
	The website will contain contextual help for common operations.
Configurability	User can customize the layout of the home page modules.
	Modules may be hidden or shown depending on the users preference
Security	No submitted code or shell command should cause the server to crash.
Responsiveness	Code compilation and execution should be fast and responsive (no more than a couple of seconds)
Scalability	A single classroom session should support a class size of users.
	The entire application should support many classroom sessions.

5 CONCEPT IMAGES & UI DESCRIPTION

5.1 SYSTEM OVERVIEW

The project will consist of several different modules that interact with each other:

Authentication and Authorization (Auth)
Provides to editor rooms an interface to the ISU Single Sign-on authentication and access control. Handles the server-side ShareJS message filtering needed to restrict editing permissions of files within a room.

Room Manager
Spawns new Docker Rooms and manages their lifetimes and associations with users. When the Web Application wants to initiate a new classroom, it requests the Room Manager to start up a new Docker Room.

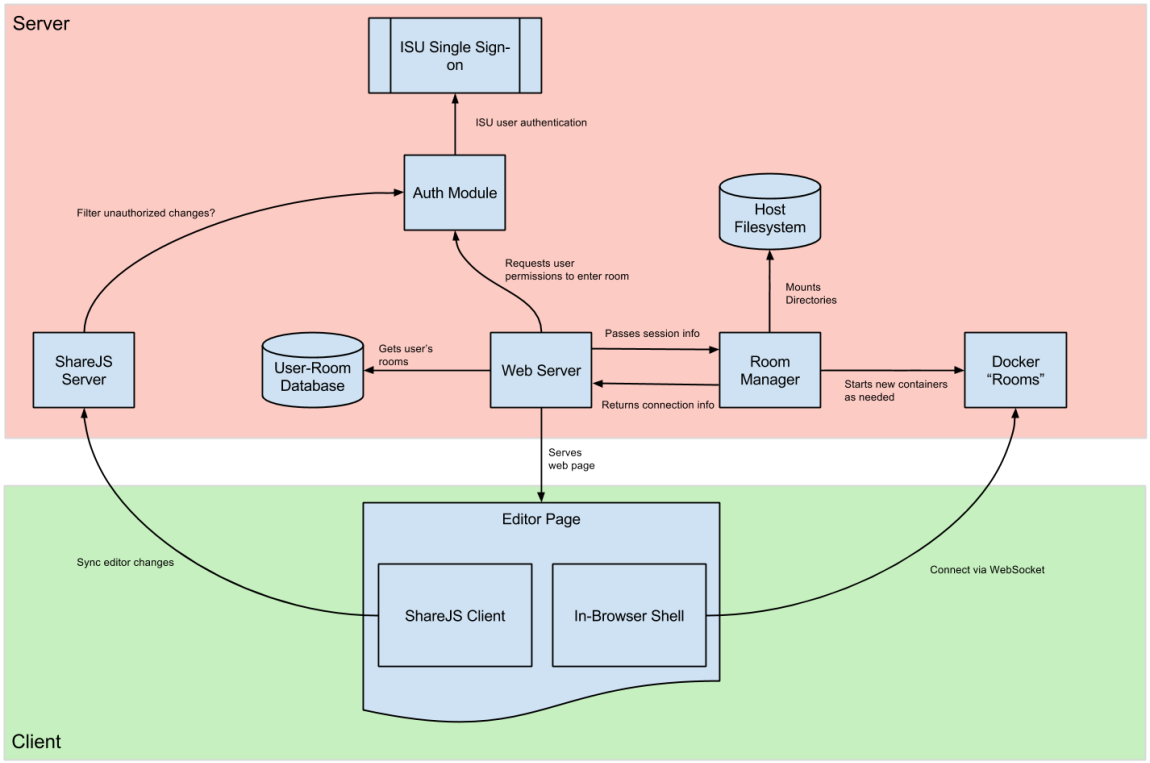
Docker Room
Docker image setup with a web service that accepts code from a classroom and streams a shell to the clients.

Web Server
Serves web page requests and acts as a connector for the other modules. Interacts with and serves the client-side files. Based on the client-server interactions, it will also interact with the room manager, ShareJS, and Docker rooms.

ShareJS Server
Receives and sends text operational data with connected clients. Provides a hook to process message data and keeps the editing components synchronized with one another.

ShareJS Client
Resolves text operations from concurrent users using operational transforms.

5.2 System Diagram



5.3 Docker Server

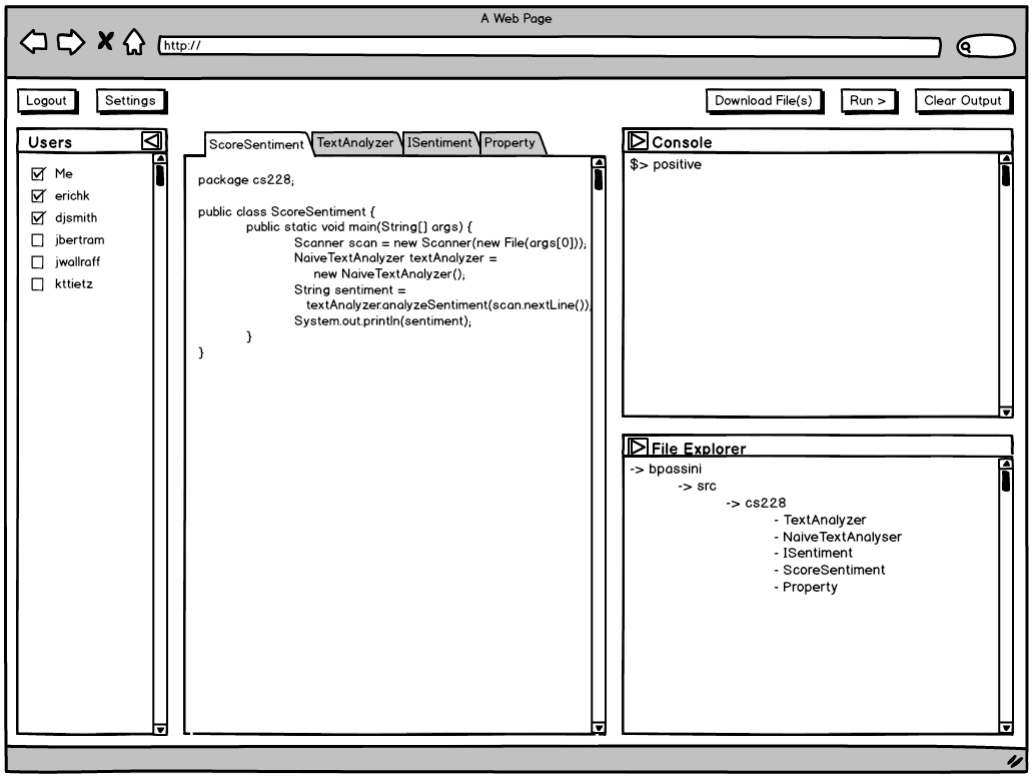
To execute arbitrary code safely and allow interactive shells for users, Docker will be used. With Docker, we can make use of Linux containerization, a concept that allows separation of multiple user spaces while using the same kernel and hardware resources. Basically, the systems appear to run like virtual machines, but are more light-weight and require significantly fewer resources, while still providing an illusion of multiple systems and the security that comes with this.

Our containers, called classroom sessions, will be managed by a web service that allows for starting sessions, connecting to sessions, and managing the authentication of users within the classrooms. This service will be called the Room Manager.

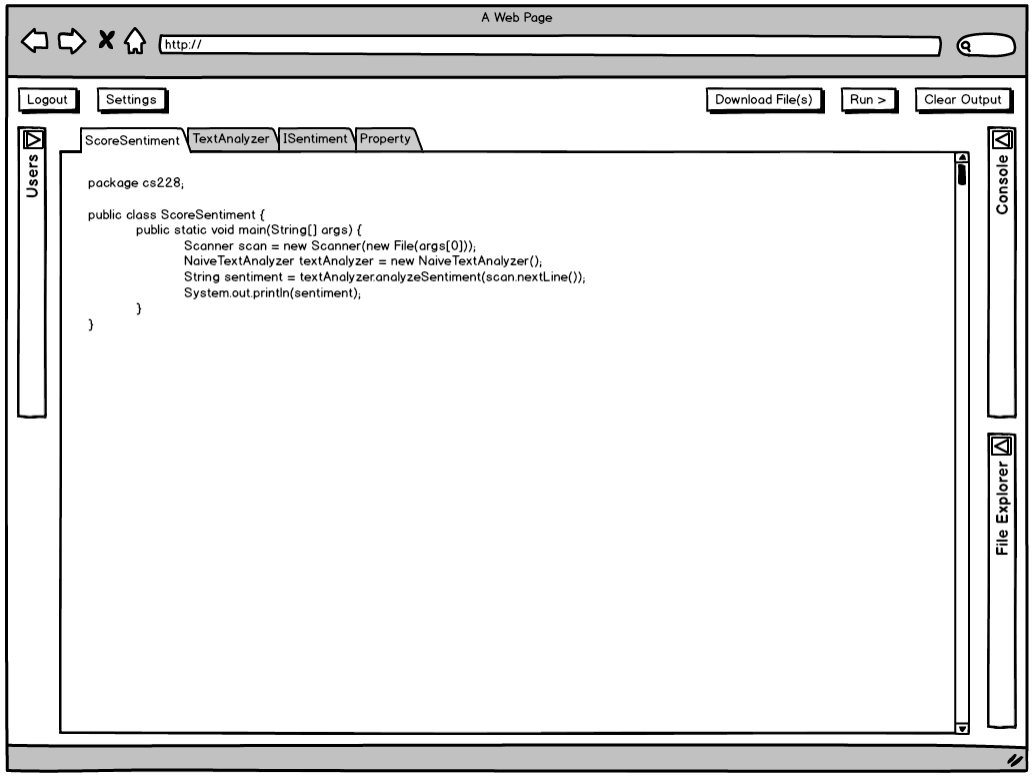
Each container will contain a web service to allow streaming interactive shell sessions (e.g. through bash or another common shell) over a websocket connection, compiling and running of projects, and managing the file system of a project.

5.4 MOCKUPS

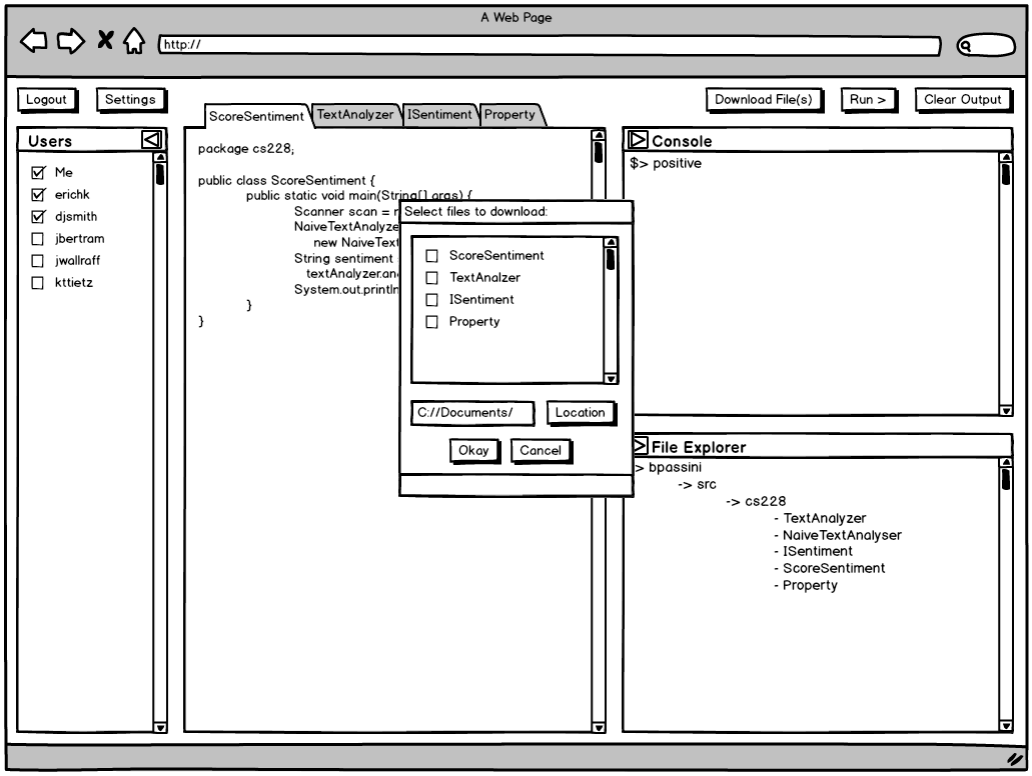
5.4.1 Instructor View



5.4.2 Instructor View with Minimized Windows



5.4.3 Instructor View with File Download Modal



6 VERIFICATION & VALIDATION

6.1 VERIFICATION

6.1.1 Static Testing

Before major updates to the project are pushed to the repository, the code is sent through a brief code review process. Another group member or members will inspect the code and give the go-ahead if they agree with the proposed changes. This extra step is useful for finding bugs and getting different perspectives of the design choices of all the group members' work.

Twice each semester we will hold a detailed code walkthrough. In these walkthroughs we will take a look into each module of the project and the author of each part will talk the rest of the group through their design and how the code works. This way we can be sure that everyone in the group has a broad understanding of the how the whole system works even though they have only worked on parts of it individually. This is also another way to check that we are following our schedule, meeting milestones and following the original project requirements.

6.1.2 Integration Testing

Many features may work in isolation and yet fail to integrate with the larger project framework. As such we will perform component tests in isolated environments as well as larger integration tests with the whole project. Both the file server and the shareJS server will need to be unit tested to determine whether or not the services they offer are working properly under all possible input conditions coming from other external modules.

6.1.3 Stress Testing

We will need to consider a couple of stress related scenarios. In order to meet the goal of supporting multiple classrooms running the CoderLab application at the same time, we will need to stress test the Docker room manager server. An ideal performance would be if the room manager can sustain at least 30 Docker rooms at the same time. We may have a better idea of the upper limit of rooms we will need to support closer to release but for now we will aim at reaching the goal of 30 rooms. We will also stress test the file server and check whether or not we can support a reasonable number of simultaneous, open files across all the Docker rooms.

6.1.4 Security Testing

Some of the last testing we will need to do is security testing. There are several potential security threats which we will need to consider. Some of them involve our use of Iowa State's single sign on authentication. Given a URL shared by a professor to a specific Docker room (intended for student use), we will want to ensure that only users who have passed through ISU's authentication will be able to enter the room and view/modify the files inside. This way no outsiders (non ISU students) will be able to break in and compromise students' work. We will also want to make sure that our in-app authentication and user recognition works as it is intended to. For example, a basic user (participating in a room but not the administrator) should not be able to change the privileges of other users or assign access to the current room to new users.

6.2 VALIDATION

6.2.1 Prototyping

Up until now we have maintained a prototype of the CoderLab tool. This prototype currently includes basic forms of the ShareJS collaboration, filesystem, and UI features which will be seen in the final CoderLab release. This prototype has been very helpful in validating our work thus far. We have been able to hold several demos with our client/advisor where we obtained feedback on our work with the prototype. This feedback has been important in ensuring we have been developing features to the meet the client's needs.

6.2.2 Customer Testing

Our ultimate goal is to get CoderLab into the classroom and used by professors and TAs across campus. An important step in our validation process, before we put CoderLab into the hands of the public, will be to give access to select professors to test it in a real classroom setting. We aim to do this with 1 -2 months left in the second semester. By then we hope to have, if not the final version, a version close to the final product. This validation will help us see if CoderLab meets the use cases set by our client/advisor. We hope to obtain useful feedback and criticism which we can take and apply directly to the product to improve its usability and attractiveness to the final target users.

7 DELIVERABLES

7.1 SEMESTER 1 – FALL 2014

- Design Document – a document that describes the overall design of this project. This includes a detailed design of the multiple modules of our system.
- Determine the best technologies to be used for collaboration, website UI, shell streaming, file management, file storage, authorization, and containerization.
- Collaboration prototype – a proof of concept prototype that demonstrates that our chosen collaboration tool will be feasible to implement within our timeframe and will meet the requirements of this project.
- Docker prototype – a proof of concept prototype that demonstrates code compilation in a container environment.
- Shell prototype – a proof of concept prototype that demonstrates an interactive terminal session through a browser stream.

7.2 SEMESTER 2 – SPRING 2015

- Single Sign on demonstration – demonstrate to our client that we have successfully integrated with ISU's single sign on service.
- User Interface demonstration I – show our client the user interface and get feedback as to how it could be improved.
- User Interface demonstration II – show out client the improved user interface.
- Collaboration demonstration – demonstrate to our client that we have successfully incorporation collaboration technologies with our system.

- Permissions demonstration – demonstrate to our client that we have incorporated read-only and write permissions into our system.
- Compilation demonstration – demonstrate to our client that our system can successfully send code to the server, compile it, and display the results back to the user.
- Final demonstration and release of our software.
- Updated design document so our system is easy to maintain and expand upon in the future.

8 GROUP MEMBER ROLES & TITLES

Kyle Tietz – Team Lead
Lead discussions at meeting when necessary and ask probing questions. Observe the project from a high-level perspective and make sure pieces are falling into place.

Jake Bertram – Communications
Contact with people outside of the project as needed, for things like server resources or Single Sign-On integration. Work on Weekly Reports and ensuring they are completed in time and sent out to team/adviser.

Bryan Passini – Communications
Assists Jake in the compilation of weekly reports and project reports, as needed.

Daniel Smith – Key Concept Holder
Manage the central ideas that make up the project structure. Ensure that project tasks are limited to an appropriate scope.

Erich Kuerschner – Webmaster
Manage content on team website (uploading reports and design documents) and lead development of client facing web page for the CoderLab solution.

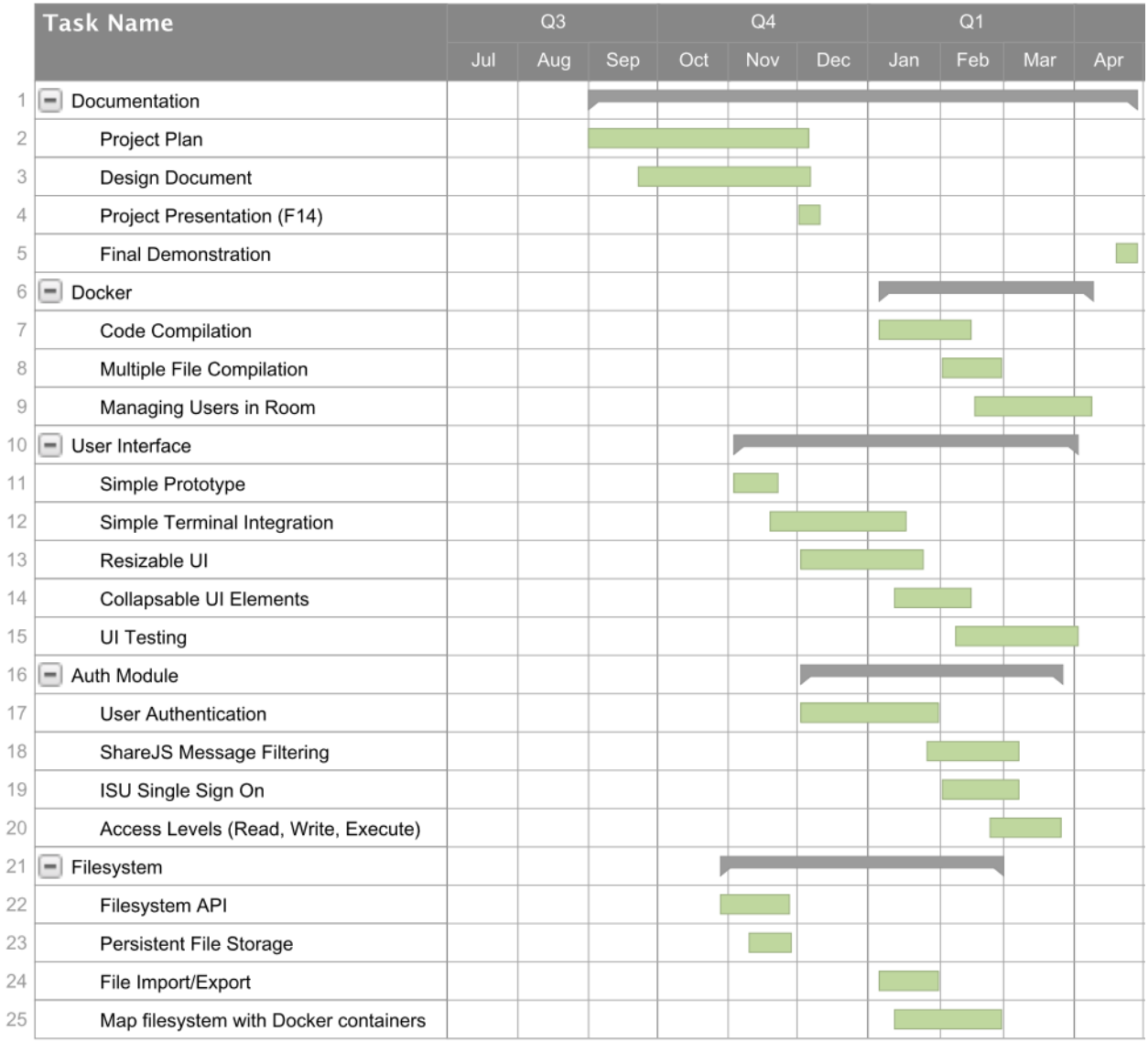
Jacob Wallraff – Webmaster
Manage content on team website (uploading reports and design documents) and lead development of client facing web page for the CoderLab solution.

9 PROJECT SCHEDULE & WORK ITEMS

Listed below are the major work items for the first (fall) semester:

- Assemble the client-facing web application
- Create a working demo of collaborative editing using ShareJS
- Integrate JavaScript libraries to enhance text editor (Syntax highlighting, cursor tracking, etc.)
- Prototype a service that interprets and compiles C code using Docker containers
- Add support for multiple files
- Obtain server space and install components we may need for compiling and executing code
- Solidify solution for integrating ISU single sign-on with the web app

The figure below shows an estimated timeline for the major code development tasks, most of which will occur in the second (spring) semester:



We have selected dates for several important deadlines for the spring 2015 semester:

- Placement of Parts into UI — January 23
- Customizable UI Appearance — February 13
- Major Client Demo #1 — February 20
- User File System Implemented — February 27
- Integration with ISU Single Sign-On — March 6
- User Permissions Implemented — March 27
- Handling for Multiple Rooms — April 3
- Major Client Demo #2 — April 3
- Customer Testing #1 — April 17
- Customer Testing #2 — April 24
- Final Demonstration — May 1

10 Risks

10.1 Cost Considerations

There are no serious financial risks associated with our project. However, there is an investment which the university will have to make. This investment comes in the form of server space and the maintenance required to keep the servers up and running.

10.2 Low Responsiveness

Perhaps the most important aspect pertaining to the usability of the project is the responsiveness of the interface. If multiple people are editing the same code source and the interface is not updated in a timely manner, then the final result of the code could look very poor. This problem may only be compounded when the same users try to fix the code. This risk applies also to the shell and output of the program. If this issue is based in the software, it would happen with very high frequency and significantly affect the service's usability.

10.3 Lack of Users/Clients

This project is being created to solve a deficiency which has been identified but which has not necessarily been clearly outlined by the potential users. We are creating a piece of software based on issues identified in personal experiences and the experiences of students in the targeted introduction to programming classes, but we do not know how well this solution will actually play out in the classroom. The consequences of this risk are the final project never being used in a classroom environment in any serious capacity other than novelty.

10.4 Time Frame

We are confident that our group can finish the project on time. However, if we do not allow time for user testing and feedback, we will only have our own biased feedback off which to base further changes to the product. If there are a few must-have features which we have overlooked, we would cut off a large portion of the potential user-base who would otherwise be interested in the product. If this problem were to arise, there are no project-endingly adverse effects, but the impact of the project could be greatly reduced.

10.5 Small Language Set

While we are initially planning on supporting only the 3 languages from the intro-level programming courses at ISU (C, Java, and MATLAB), there is a lot of room to expand to support other languages. These are not exceedingly modern languages even though they are widely used, and as such new courses using different languages would again run into the same problems this project tries to solve. On the other hand, supporting too many diverse languages could dilute the service and make it unwieldy.

10.6 Server and Hosting Resource Availability

Our team has acquired a pair of virtual machines on which we will deploy on web application. These virtual machines are managed by Iowa State and allotted to us for a specific timeframe. We have not yet examined the long-term availability of our host for the application and whether it will continue to be hosted by Iowa State.

10.7 Lack of Features in ShareJS

The ShareJS library provides the basic functionality needed for simple code collaboration. However, several features such as editor cursors are not directly supported and may need to be implemented by our team. This runs the risk of poor implementations of features as we need to directly change the ShareJS library.

11 Conclusion

CoderLab is a browser-based real-time collaborative coding environment that will allow multiple users to write code simultaneously. Users will be able to collaboratively edit, compile, and save code. We will use Docker to create the room environment, ShareJS to provide collaboration, and ISU Single Sign-On as our authorization interface. CoderLab is being built with professors and students in mind. We are developing a tool that makes it easier for professors to teach coding principles and improves the student coding experience. By allowing students to collaboratively code with professors, TA's and peers, we hope that students will find it easier and more engaging to learn new languages and coding principles.