May15-31

SENIOR DESIGN 1 - CODERLAB

Project Plan

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

TABLE OF CONTENTS

1 PROJECT STATEMENT

Try to imagine a time before the internet. It is apparent just how much the world has changed since the emergence of the web. 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 CoderLab 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 using Makefiles to perform multi-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
- A file system, allowing
    - Persistent code project storage
    - 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 REQUIREMENTS

3.1 FUNCTIONAL REQUIREMENTS

Project Element	Requirement
Website UI	The website shall be easy to navigate – the user shall be able to navigate between pages with no more than one click.
	The user shall be presented with all required windows upon logging in: code collaboration window, output window, and shell window.
	An administrative user shall additionally have access to the user privileges panel.
	Each window shall be resizable to suit the user's tastes and monitor resolution.
Shell	The shell shall be based on the Unix shell and shall support a subset of Unix commands.
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 the active project.
	The editor shall support basic code completion.
User Info Panel	The panel shall list all the users that the opened project has been shared with.
	The panel shall visually distinguish users who are online vs offline and users who have editing privileges.
Output Panel	The panel shall display the results of executed code along with any thrown errors.
Authentication	The website shall have an option to log on using ISU's single sign-on system.
	*Guest usage (non –ISU user) may be considered in the future

3.2 NON-FUNCTIONAL REQUIREMENTS

Quality	Description
Ease of Use	User is able to navigate the site with ease without the need for assistance from teachers or peers.
Configurability	User can customize the layout of the home page modules (output, textbox(s) and shell) in any arrangement they want.
	Modules may be hidden or shown depending on the users preference
Security	No submitted code or shell command should cause the shell to hang, page to become unresponsive, or server to crash.
Responsiveness	Code compilation and execution should be fast and responsive (no more than a couple of seconds)

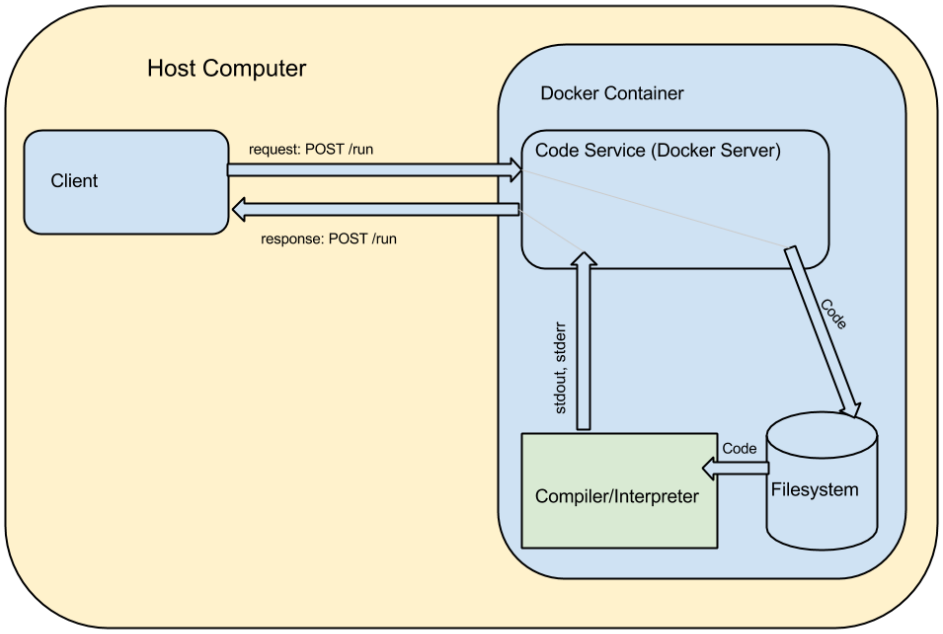
4 CONCEPT IMAGES & UI DESCRIPTION

4.1 SYSTEM OVERVIEW

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

- Front-end - The client-side, customer-facing single page web application, including features such as:
    - The code editing component with support for multiple users
    - Ability to see the other users in the same room, including their cursors on the code editor
    - A way to manage multiple files, e.g. through tabbed documents
    - Client-side authentication/authorization
    - A shell or input/output section that either streams or outputs results of code execution/compiling to the user.
- Back-end - Communicates with front-end module to provide it with functionality:
    - Handle classroom management - assigning users to rooms and allocating the correct resources to them, such as a remote shell instance, the shared code, and permissions.
    - Integrate with Single Sign-On to provide authentication
- Collaboration - May exist on the same web application as the Back-end, or in its own application
    - Keeps track of revision histories and syncing users' code editor components.
    - Operational transforms and other complex problems involved in collaborative editing can be handled using technologies such as firepad or ShareJS
- Container Manager - manages Docker containers
    - Cleans up / destroys unused sessions
    - Starts up a container (running a Docker Server) and keeps track of its port mappings and classroom assignments
- Docker Server - a docker container with a web service for accepting code, building, and running it within the container
    - Contain rules that decide how to compile and run given code (e.g. language detection, compiler flags, …)
    - Stream inputs/outputs/errors back to the container manager

4.2 DOCKER SERVER



1-Diagram of Docker Server

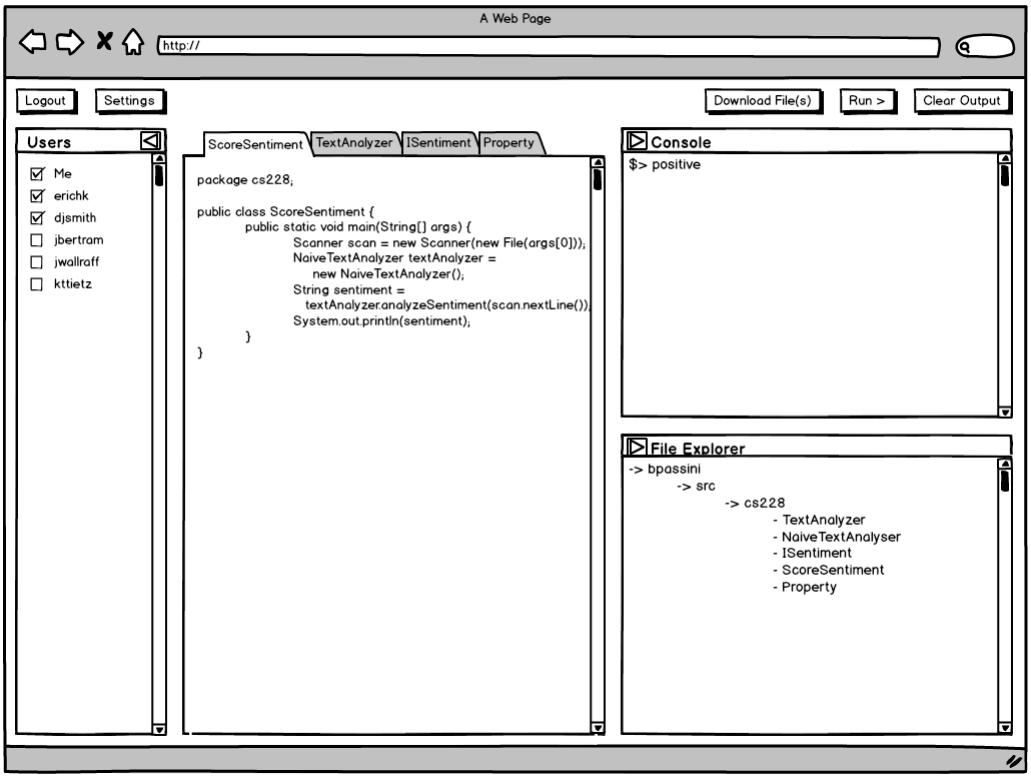
To execute arbitrary code and multi-file projects safely, Docker (www.docker.com) will be used. With Docker, we can have lightweight containers that are treated a lot like Virtual Machines (VMs), but take up much less computer resources. Our containers will run a web service that the *host* (the computer running Docker containers) may post code to by initiating an HTTP POST request whose body is a supported file type or a .tar.gz archive.

The service running in Docker will take this request and see if its Content-Type header is that of a supported type (e.g. application/x-gtar, application/x-c, etc.) and from that, determine how to compile/run the given code.

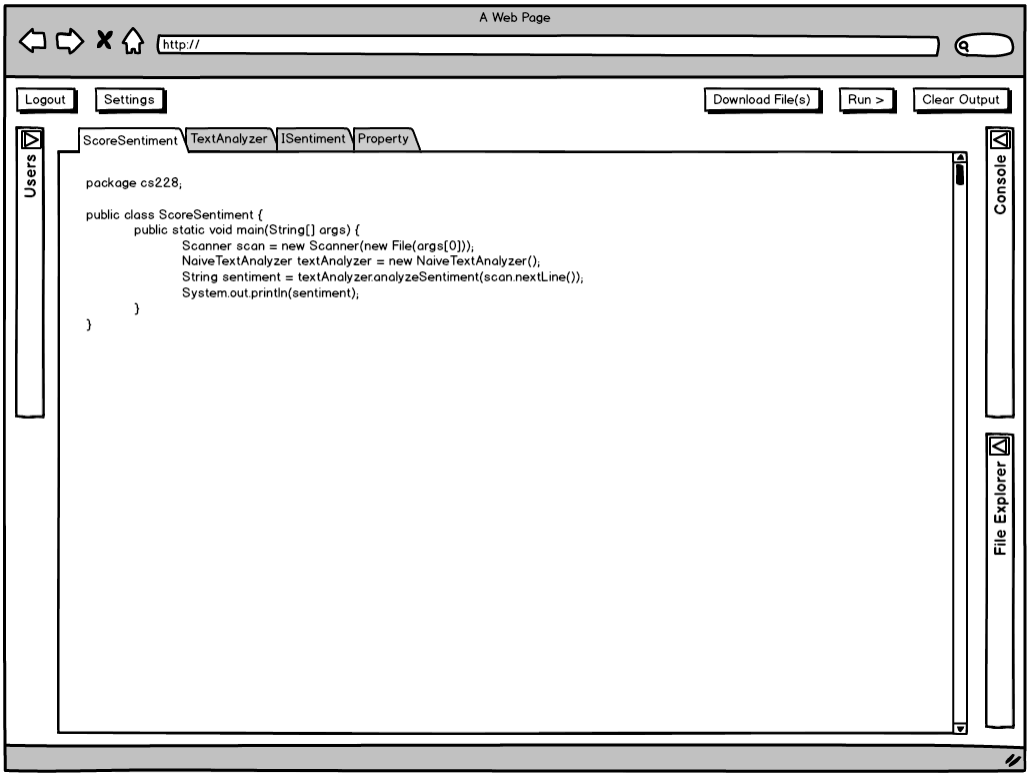
Upon success, the response body will contain the stderr and stdout results in a JSON object. Invalid code or requests will return a status code of 400 (bad request), and other errors with the build system will result in a 500 internal server error.

4.3 MOCKUPS

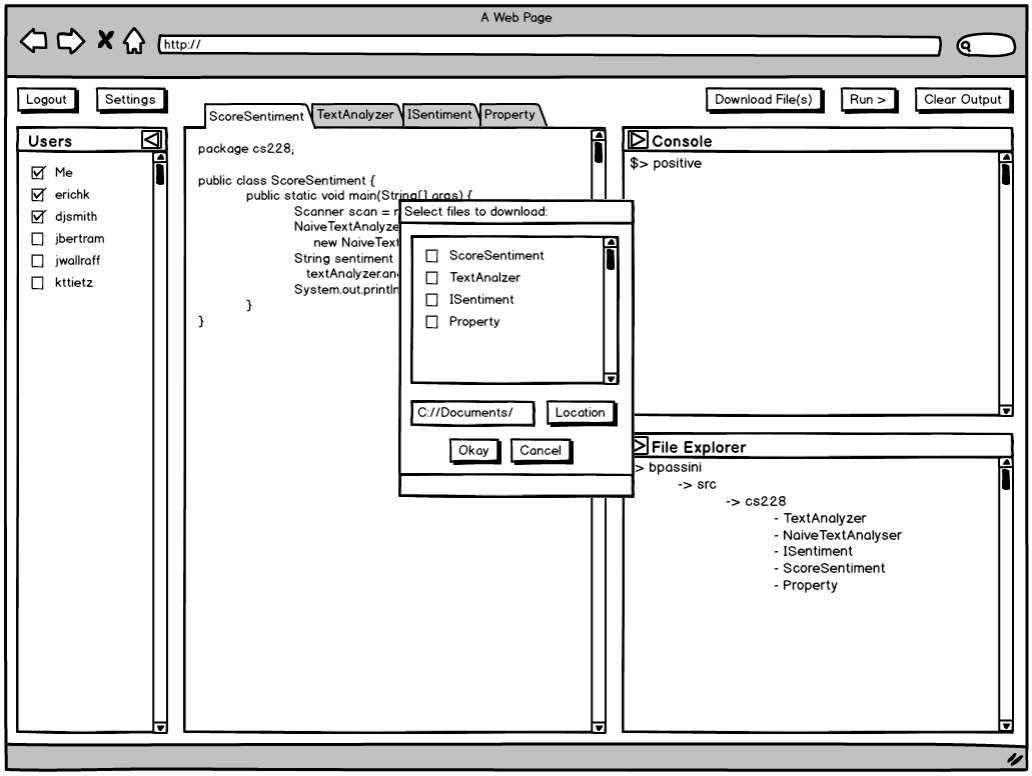
4.3.1 Instructor View



4.3.2 Instructor View with Minimized Windows



4.3.3 Instructor View with File Download Modal



5 VERIFICATION

There are many components to this project and many features to verify.

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.

6 DELIVERABLES

6.1 SEMESTER 1 – FALL 2014

- Design Document – a document that describes the overall design of this project. This includes both a detailed design of the front end and back end modules of our system.
- Determine the best technology to be used to implement our backend.
- Server prototype – a proof of concept prototype that demonstrates that our backend design will in fact work and will meet the requirements of this project.
- Determine the best tool to use to implement code collaboration.

- 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.

6.2 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.

7 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.

8 PROJECT SCHEDULE & WORK ITEMS

Listed below are the major work items for the fall semester:

- Assemble the client-facing web application
- Create a working demo of collaborative editing using ShareJS or TogetherJS
- Integrate decorative Javascript libraries to enhance text editor (Syntax highlighting, cursor tracking, etc.)
- Prototype a service that interprets and compiles C code using Docker containers
- More code support / 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

9 RISKS

9.1 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 bad. This problem is only compounded when the same users try to fix the code. This risk applies also to the shell and output of the program. If this problem was found in the software, it would happen with very high frequency and significantly affect the service's usability.

9.2 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 don't 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.

9.3 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.

9.4 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.

9.5 Server and Hosting Resource Availability

There are a number of cloud hosting services available, and each have different pricing models. Depending on the success of the project, the cost of hosting, access, data transfer, varies between each provider and could be moderately expensive. As this is a purely code-based project (i.e. no hardware components), there is no budget allocated for it. If this problem were to arise, the project might cease working because we have run out of our resource allotment on a specific model and we would likely have to turn to the college and ask if they would fund it.