Joey DiMaria
Paul Kwak
Manuel Marroquin
Taylor Murray
Software Engineering
Professor Jane Cleland-Huang

Team Portfolio

1. Project Overview

The purpose of this project was to design and implement a SCRUM Board, a software development tool meant to facilitate collaboration amongst developers, for a team of <10 developers. The project was to be implemented using a client and server structure that used a shared data structure across clients for real time updates across clients.

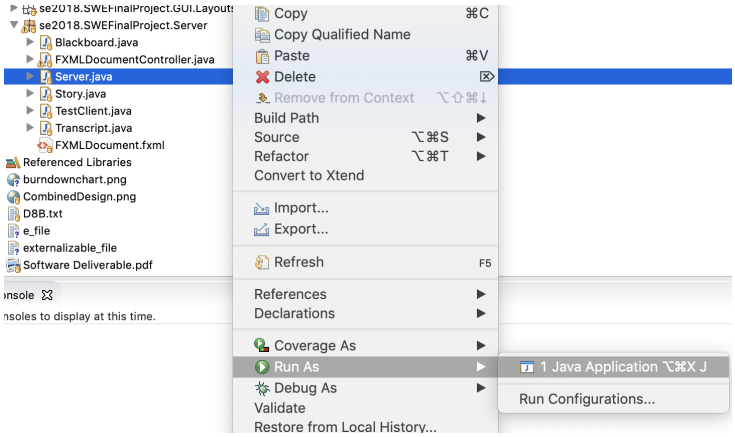
The implementation of the project required a sprint backlog (TODO swimlane in our case), a regular backlog, a burndown chart, addition/modification/deletion of user stories, and a SCRUM board to keep track of stories and their progress to completion.

Our SCRUM implementation included 4 swimlanes TODO, In Progress, Testing, and Done. A user could add stories through a button, see more details about stories by clicking them, and drag and drop stories to a new swimlane. When changes are made in our implementation, the changes propagate up to server and update the shared data structure across the clients. The time for this change is about 3 seconds.

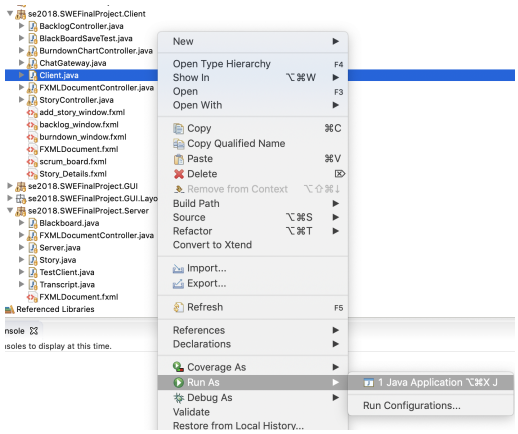
1.1 Documentation

Instructions for Running:

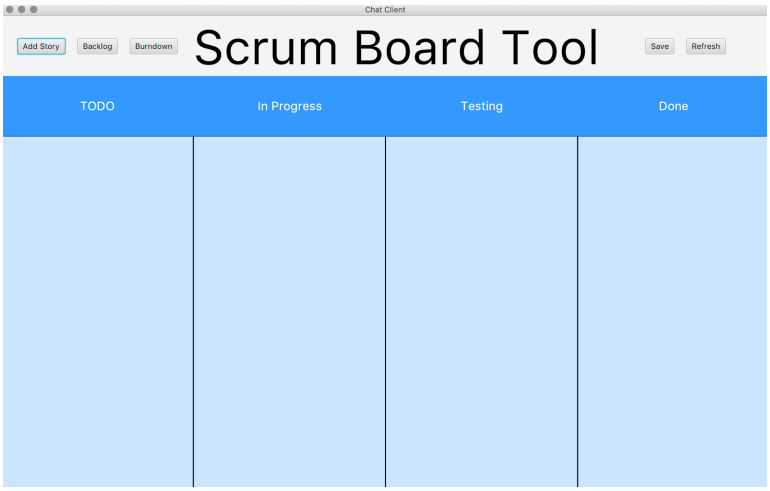
In order to run this program you must start the server before the client by going to the *se2018.SWEFinalProject.Server* package and right click and run *Server.java* as follows:



Second, you will need to start up the client(s). Go to the *se2018.SWEFinalProject.Client* package in the source directory and right click and run *Client.java* as follows:

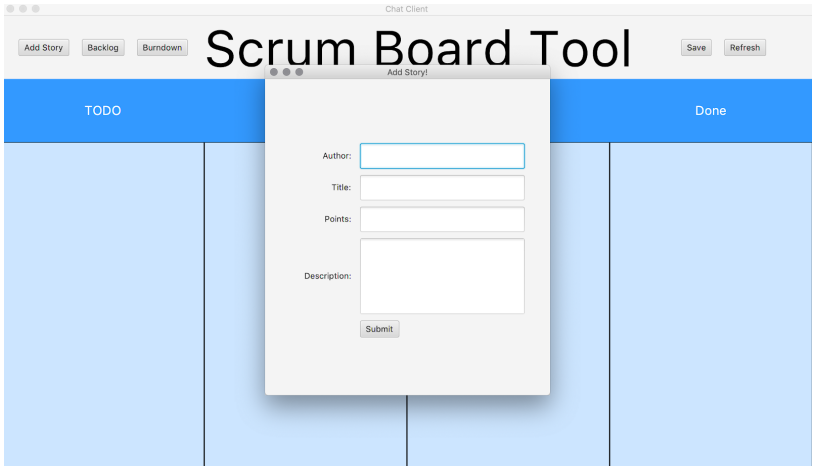


At startup you should see the server start up and the following window appear:



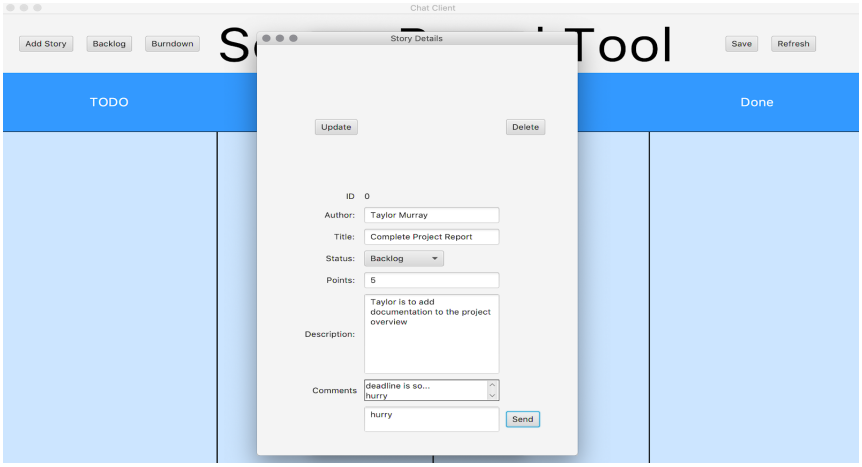
Adding User Stories:

Simply Click the ‘Add Story’ button to Create a new story. When in the add story window you can set the author, title, points, and its description. Hit the ‘Submit’ button for the story to be send to the server product backlog. When you add a new user story it will go to the backlog first.



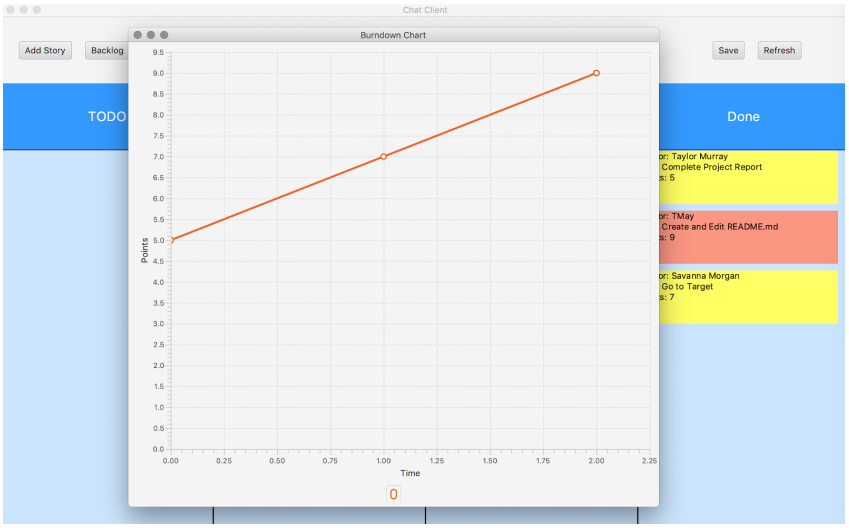
Product Backlog, Edit, Delete, and Comments:

After adding a story it will go straight to the backlog. The following is an image of the backlog. Clicking on an story pane (colored tile with story summary) will cause another window to open with which you can edit the story. This is show in the second image below. To edit simply type in one of the editable boxes and when done click the ‘Update’ button. To delete the story simply click the ‘Delete’ button. The status can be updated by switching the status with the dropdown menu and then clicking ‘Update’. To add a comment to the ‘Comments’ section, type in the editable text box at the bottom of the window then hit ‘Submit’. The ‘Comments’ section will update with your message. The project keeps all of this information up to date across the clients. You should wait for 3 seconds before the refresh is complete.



Backlog:

The backlog updates a line graph on the user story points over time (as stories are set as done). Please note that a story going straight from backlog to done means it was not “developed” through the swimlanes and will not populate the burndown chart. A story must be moved from one of the first three swimlanes and then to done to populate the burndown chart. Below is a image of the burndown chart.



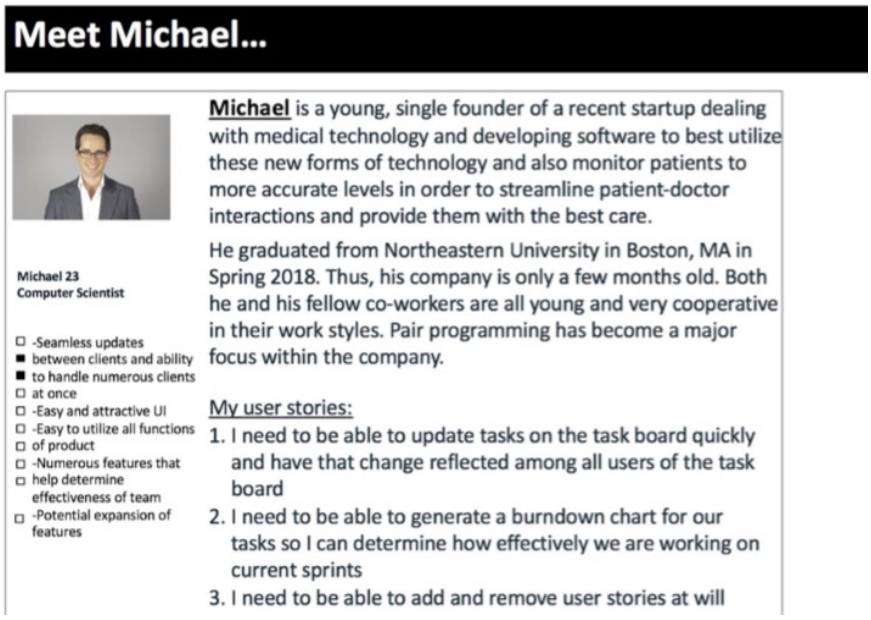
Save:

Clicking the save file will save the sprint to your computer under the name externalizable_file.

2. Three user stories

	**Jake** is a junior software engineer who recently joined the team. He actually doesn't have a college degree, but instead came from a bootcamp which focused more on the basics of computer science. He prefers coding in Python and plans on being a part of the team that's going to use this software. However, since he's unfamiliar with SCRUM practice, he'd hope it'd be easy to pick up.
Jake: Age: 22, Junior Software Programmer ❏ Fast trace retrieval ■ Platform selection ❏ Language selection ■ Reliability ❏ Extensibility ❏ Ease of component upload ■ Highly intuitive interface ❏ Data confidentiality ❏ Broad adoption	My user stories: 1. I need to be able to easily understand what I'm looking at. 2. I need to be able to access different parts of the scrum easily. 3. I want to see how I'm doing as well as the team in terms of the timeline. My anti-stories: 1. I won't use it if the UI doesn't make sense

Persona:



SWE FINAL PROJECT PERSONA

Justin
Age: 24, Developer

- ▪ Collaborative
- ▪ Accessible
- ☐ Consistent
- ▪ Ease of story creation
- ☐ Reliability

Justin is a recent college grad who graduated with a degree in computer science. He has been hired by a software company as a developer. He has no previous work experience as he just graduated, but he is eager to contributions to his new team.

Even though Justin lacks in experience, he is a good team player and a self-motivated individual.

My user stories:

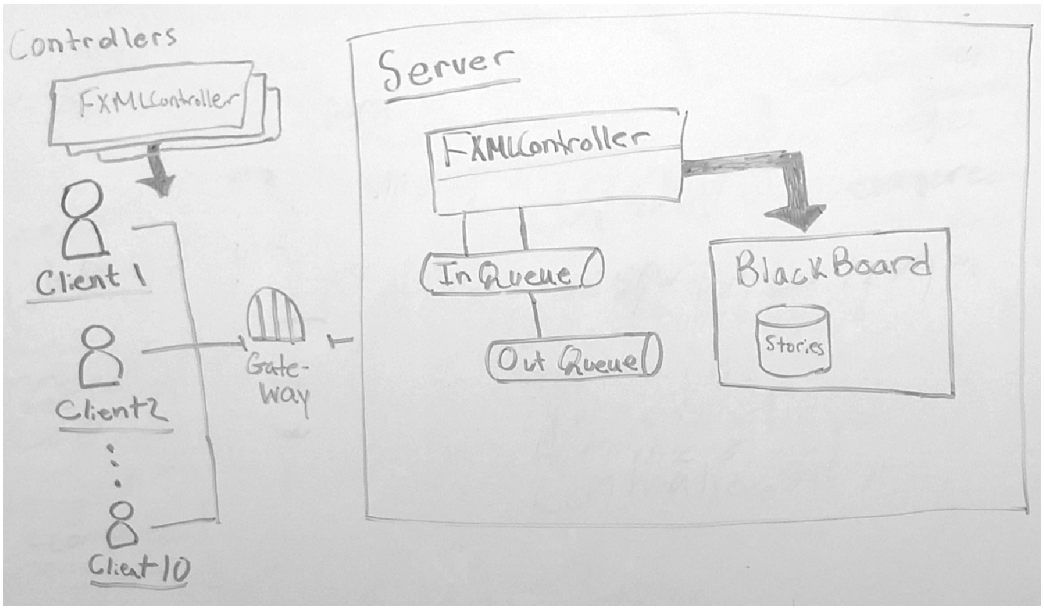
1. I need a straightforward interface for messaging and collaborating with coworkers on user stories.
2. I need to be able to quickly find and contact the owner of a story
3. I want to quickly assess the progress of my team's project with a burndown chart.
4. It should be easy to look at the swim lanes of the SRUM board to know the workflow my coworkers expect from me.

3. Fully functioning, tested, code base to github.

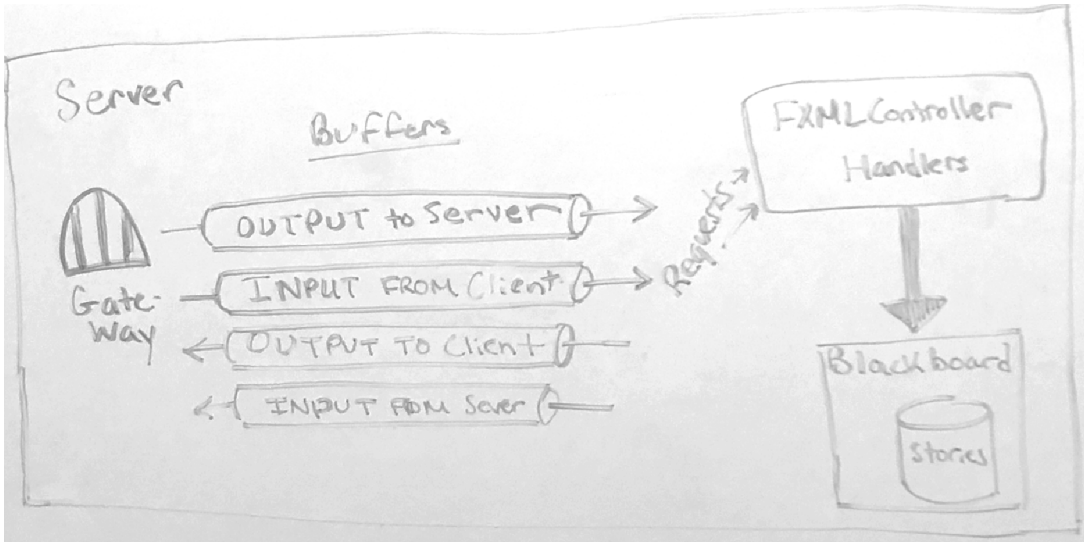
Our project can be found the public repo at https://github.com/TayMurr/SWEFinalProject/ .

4. Architectural design (a diagram and a discussion of design decisions)

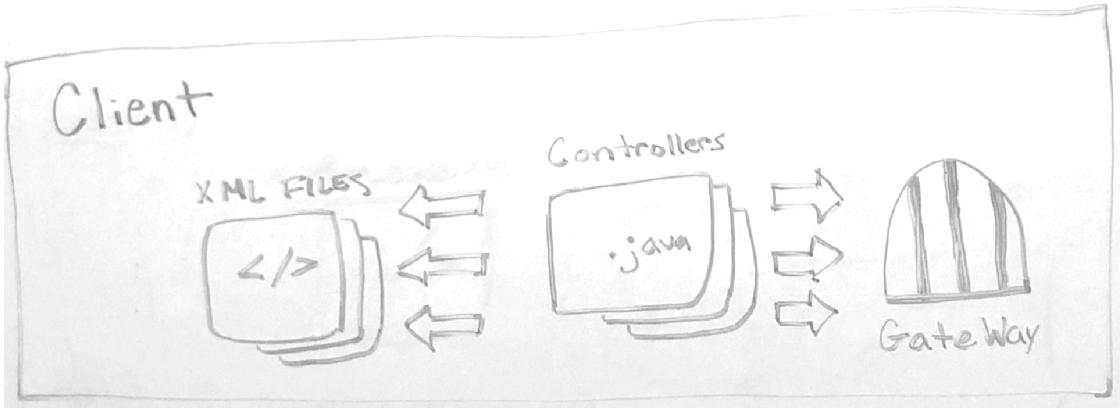
4.1 Overall Architecture



4.2 Server Architecture



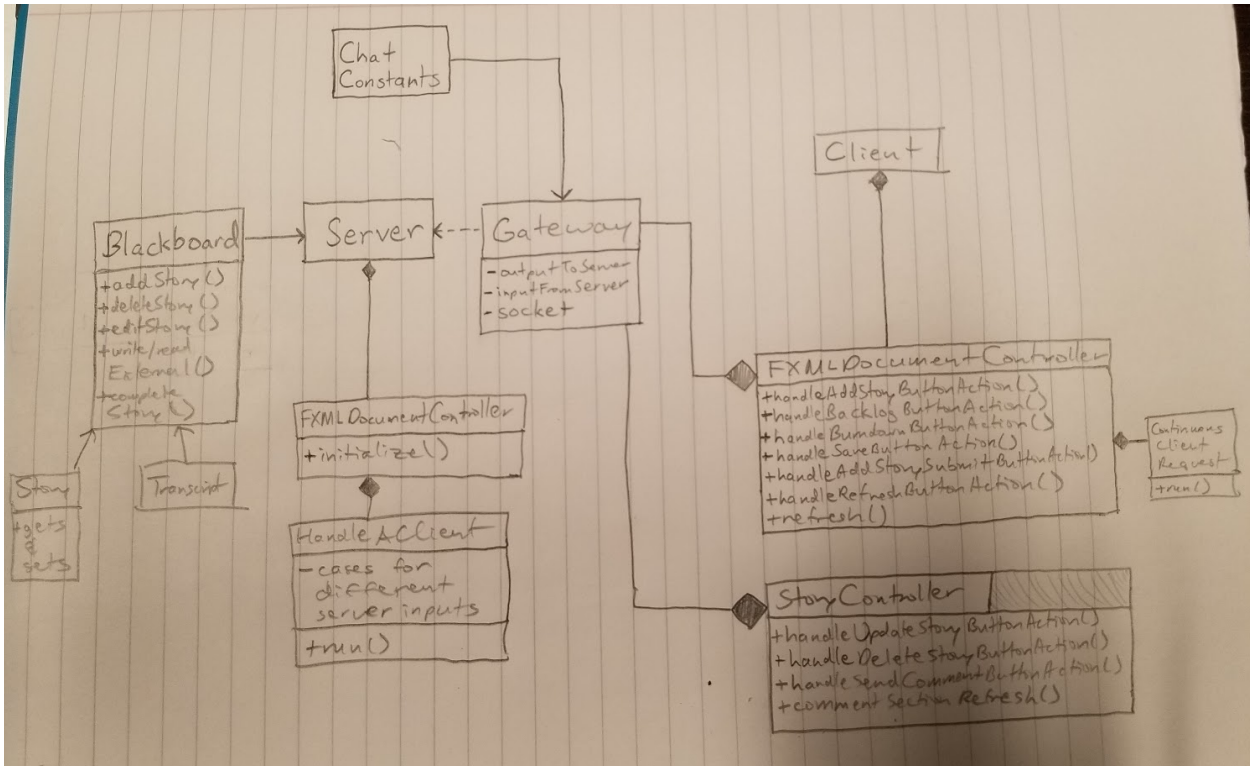
4.3 Client Architecture



4.4 Design Decision

For the overall architecture, as suggested by the professor we decided to go with a loosely structured blackboard architecture. The blackboard design pattern is a behavioral design pattern typically used when diverse modules need access to one central data source. The modules (controllers in our case) interface with the blackboard with a gateway class. This follows the software engineering design principle Single Responsibility as the Gateway is solely responsible for sending the requests to the server, providing a clean interface for the controllers to fetch info from the blackboard. The gateway and the server communicate via message buffers that information in string format gets printed to. This is reflected in the server specific architecture diagram. To maintain the Single Responsibility principle, we tried to abstract as much functionality to each window to its own controller besides the FXMLDocumentController in the Client package. The alternative would have resulted in a very large FXMLDocumentController and code reusability would be reduced greatly

5. Class Diagram for major components.



6. Test plan and results (Unit tests, acceptance tests). Acceptance tests that evaluate your user stories.

The unit tests are in our repository (test in class name), and the acceptance tests are described below:

Acceptance Test: Multiple Clients run on Same Server (Michael)
Preconditions: Server must run before any instantiation of Client. Then once the GUI for the client is brought up, he can either add a story or update a story or comment section.

Happened: For adding a story or updating a story, he'll be able to add the story's information and then see it reflected once the refresh button is hit.
For adding a comment, he'll be able to add text to a textbox and once he hits send, the comment section will be updated in "real time".

Acceptance Test: Comments (Justin)
Preconditions: Server must run before any instantiation of Client. Then once the GUI for the client is brought up, then a story is added, then you can click on the story to see the comment section for each story.

Happened: For adding a comment, he'll be able to see the comment section in "real time" as well as add his own comments. Also, he'll be able to see who wrote a story by just seeing the author of the story clearly displayed.

Acceptance Test: The GUI has a good UI interface (Jake)

Preconditions: Server must run before any instantiation of Client. Then once the GUI for the client is brought up, he'll be able to see it.

Happened: Once the GUI is brought up, he'll see the stories divided up by columns representing the SCRUM board. He'll also see the intensity of each story represented by color. Also, he'll see the buttons at the top cleanly displayed into different sections along the header.