Live Course

(Working Title)
Project Plan Document

Team 4
Darren Cheng
Lee Engelman
Brandon Klen
Hayden McAfee
Jeremy Meyer
Lars Sorenson
Table of Contents

What went well during the last sprint?

General
- Communication between team members was sufficient
- Self reliant tasks proceeded smoothly while tasks reliant on other tasks got stalled
- Work was organized wonderfully

Tasks
- HTML5 Web App & Backend
- Android App

What didn't go well during the last sprint?

General
- Not enough members working on the API
- Almost no functionality was implemented due to tasks that are reliant on other tasks
- Team members started to fall behind in work
- Coding standards caused team members to overwrite each others work
- Underestimated amount of hours expected for each task
- Overestimated hours required for testing purposes

How should the team improve for the next sprint?

General
- More accurately estimate hours required for each task
- Conduct more meetings for “group coding”
- Save testing for Sprint 3 and focus more on development
What went well during the last sprint?

General

- Communication between team members was sufficient
  - Do to the ease of communication (facebook app on phones), we kept each other up to date on new code and new features being finished as well as what needed to be done and what should people do.
- Self reliant tasks proceeded smoothly while tasks reliant on other tasks got stalled
  - Tasks that did not rely much or at all on other tasks (such as most of the UI) went smoothly as they have no code that they relied on in order to code. This, however, causes the problem that there is no functionalities and we ended up having just the UI (which was beautiful for both the web-front and android-front).
- Work was organized wonderfully
  - We split the work down between the web-front, android-front, and the server back-end with some intermixing between web-front and server back-end. this allowed us to work mostly only on our portions of code and not touch each other’s unless we had to, making it easy to code and had little conflicts.

Tasks

Tasks are listed in the following format:

- User Stories
  - Tasks
    - Comments

Notes:
A lot of the comments say the same thing or very similar in nature as a lot of the problems or success are caused by the same issues. Most of these issues are listed in further detail in the three sections: “What went well during the last sprint?”, “What didn’t go well during the last sprint?”, “How should the team improve for the next sprint?”

HTML5 Web App & Backend

- As a User, I need to See Chat Rooms and Their Contents
  - Create an appropriate SQL Schema for managing rooms and messages
    - Chat Rooms were associated with physical rooms. Physical rooms were associated with buildings, and buildings were associated with institutions. Messages were associated with chat rooms and authors.
Create a UI for viewing chat room contents
  - UI was the first thing created, later connected to the API using Javascript to populate with data and make dynamic.

Add API method for grabbing live chat room contents
  - Grabbing live chat room contents was never implemented due to a lack of design being properly discussed or finalized. Since we did not determine how the data was to be stored, we couldn't decide how it should be retrieved, in what format etc.

Add mechanism for updating client with new messages
  - Since we had not determined how to store the chat data, methods to update the client were not implemented either. No API for storing or retrieving the data exists so mechanisms to update the client were not necessarily a priority.

Test reliability of message delivery
  - Cannot complete due to lack of functionality

Test message fidelity (invalid characters, etc.)
  - Cannot complete due to lack of functionality

Add API methods for retrieving room information
  - Retrieving room information was fairly easy to implement so it was done quickly but due to some standards and design issues, there was some back tracking. It needs to be finalized but was implemented in a basic sense.

As a User, I can identify myself

Create SQL Schema for Storing Users
  - Basic SQL with some references to institutions was added.

Create a preliminary log-in / registration UI
  - Resulted in development of a dynamic dialog platform

Create API methods for storing, retrieving user information (authentication)
  - Implementing basic SQL to retrieve and store user information was easy and done relatively quickly. Authentication took a little extra time.

Create session management mechanism for HTML5 App
  - Stored authentication keys in cookies using javascript.

Test user registration (SQL injection, invalid chars, etc)
  - Cannot complete due to lack of functionality

Test log in (invalid credentials, invalid characters, SQL injection)
  - Cannot complete due to lack of functionality

As a User, I need to Join a Room

Create SQL Schema relating users to rooms
  - Basic SQL storing user IDs and relating them to chat room IDs was sufficient

Test inability to join non-existent or invalid rooms
- Cannot complete due to lack of functionality
  - Create API methods for joining, leaving, and retrieving participants of a room
    - Joining and leaving a room were relatively simple SQL queries built using the Active Records feature of CodeIgniter. Retrieving participants was also equally easy, leaving the whole of this done relatively quickly, but its implementation needs to be rounded out and functionality extended.
  - Create a UI for joining a specific room
    - Basic search queries for finding rooms was implemented, however more advanced search methods need to be added

- As a User, I need to send text to room chat
  - Add API methods for sending text to chat room
    - Without a chat client, chat data, or any other related API it was difficult to decide how to transfer text from the client to a chat room.
  - Add UI for sending text to room, handling invalid text, etc.
    - Incomplete
  - Test against invalid input
    - Cannot complete due to lack of functionality
  - Test to ensure reliability of sending mechanism
    - Cannot complete due to lack of functionality

- As a User, I need to view the history of room
  - Create a UI for viewing historical chat messages
    - Incomplete
  - Create API methods for retrieving historical chat messages
    - With no chat messages stored for the present, and no API for those, historical chat messages were not implemented in even a basic sense.
  - Test reliable retrieval of historical chat room messages
    - Cannot complete due to lack of functionality
  - Test fidelity of historical chat room messages
    - Cannot complete due to lack of functionality

- As a User, I need to view a room's participant list
  - Create a UI for viewing room participants
    - UI was completed, but was not connected to the API
  - Test to ensure accurate representation of room participants / listing of weirdo characters
    - Cannot complete due to lack of functionality

Android App

- As a User, I need to access and view chat rooms via Android App
  - Create a base UI for everything
- Completed without much problems except the fact that it took more hours than originally estimated
  - Create a UI for accessing and viewing chat rooms and their content
    - Completed without much problems except the fact that it took more hours than originally estimated
  - Test fidelity of messages
    - Cannot complete due to lack of functionality
  - Add hooks to RESTful API for populating chat list and chat content
    - Lack of API made it difficult to even see how we would implement it as we would not know:
      - how the input from the server would look like
      - how we should structure our SQLite database
      - how to format said database into our list adapters
  - Test reliability of message delivery
    - Cannot complete due to lack of functionality

- As a User, I can identify myself on Android
  - Create UI for identifying one's self on Android
    - Completed without much problems except the fact that it took more hours than originally estimated
  - Test User Registration
    - Cannot complete due to lack of functionality
  - Create hooks to RESTful API for user authentication / information retrieval
    - Lack of API made it difficult to even see how we would implement it as we would not know:
      - how the input from the server would look like
      - how we should structure our SQLite database
      - how to format said database into our list adapters
  - Test log-in
    - Cannot complete due to lack of functionality

- As a user, I need to send text to chat room via Android
  - Investigate Push notification on server front
    - Lack of functionality and API made this pointless and so it was pushed back and in the end it was deemed to be done in the second sprint
  - Add hooks to RESTful API for submitting text to chat rooms
    - Lack of API made it difficult to even see how we would implement it as we would not know:
      - how the input from the server would look like
      - how we should structure our SQLite database
      - how to format said database into our list adapters
  - Create UI for sending text to chat rooms
    - Completed without much problems except the fact that it took more
Investigate Push Notification Mechanism on Android Front
  - Lack of API made it difficult to even see how we would implement it as we would not know:
    - how the input from the server would look like
    - how we should structure our SQLite database
    - how to format said database into our list adapters

Test against invalid input
  - Cannot complete due to lack of functionality

Test to ensure the reliability of sending mechanism
  - Cannot complete due to lack of functionality

As a User, I need to view a room's participant list on Android App
  - Create RESTful API hooks for retrieving chat participant list
    - Lack of API made it difficult to even see how we would implement it as we would not know:
      - how the input from the server would look like
      - how we should structure our SQLite database
      - how to format said database into our list adapters
  - Create UI for viewing room participants
    - Completed without much problems except the fact that it took more hours than originally estimated
  - Test to ensure accurate representation of room participants / listing of "weirdo" characters
    - Cannot complete due to lack of functionality

As a User, I need to view a room’s text history on Android App
  - Test reliable retrieval of historical chat room messages
    - Cannot complete due to lack of functionality
  - Test fidelity of historical chat room messages
    - Cannot complete due to lack of functionality
  - Create a UI for viewing historical chat messages
    - The overestimates in hours from the other UI sections made it so that we were short on hours to finish this UI
  - Create RESTful API hooks for retrieving historical chat content
    - Lack of API made it difficult to even see how we would implement it as we would not know:
      - how the input from the server would look like
      - how we should structure our SQLite database
      - how to format said database into our list adapters
What didn't go well during the last sprint?

General

- Not enough members working on the API
  - The API was our major bottleneck for work done in the first sprint and the cause for most of our delays and lack of functionality. The root cause of this is mostly due to the fact that we did not assign enough people to work on it. As a result, those who were assigned the API ended up doing a lot of work regarding the server as well as the web-front and the amount of time dedicated to the API fell.

- Almost no functionality was implemented due to tasks that are reliant on other tasks
  - A lot of our tasks relied on the API, which is a difficult and long thing to implement, as a result a lot of our tasks can barely be done or even get started due to the API bottleneck.

- Team members started to fall behind in work
  - We assigned two members to test our functionality. As the API was bottlenecking our functionality implementation, they had nothing to really test. And so team members started to fall behind on work.

- Coding standards caused team members to overwrite each others work
  - We have some conflicts regarding how specific methods and modules would work and the coding standards attached to them. This caused some overwrites and rewrites on the same code, pulling us behind schedule.

- Underestimated amount of hours expected for each task
  - Starting off, a lot of us have problems meeting the hours listed in our hour estimation simply because we initially do not know how to do some of the tasks and required a decent amount of time to read and learn about it as well as code it. This put us over the hours listed by many of the tasks given, resulting in the tasks we work on taking longer and not able to proceed as smoothly as we had wanted.

- Overestimated hours required for testing purposes
  - We initially thought we needed a lot of hours for testing and that when we actually needed everyone to be in development instead. The lack of developers made it hard to finish a lot of the tasks in a timely manner, making it difficult for the testers to actually test.

How should the team improve for the next sprint?
General

● More accurately estimate hours required for each task
  ○ After us messing up our estimations for sprint one, we had learned more about how much hours are required for each task. This hopefully will help us allocate more accurate hours for tasks in sprint two and allow us to complete everything in a timely manner.

● Conduct more meetings for “group coding”
  ○ One of the problems is lack of free time where we can all meet. All of our schedules are very busy and as a result we do not have consistent meeting times and often when we meet it is not with the entire group. For sprint two we decided to set aside late Thursday night as a group coding section to allow us all to help each other when everyone is present (or try to get everyone to be present at those times). We will also hold meetings with static times during tuesdays and fridays. Instead of planning meetings around everyone’s schedules, the idea is to set the meetings at set times and have everyone plan schedules around them, allowing us to actually have time to meet, discuss, and code without worrying about who needs to go where in 15 minutes or so.

● Save testing for Sprint 3 and focus more on development
  ○ One of the major problems we had is lack of developers in sprint 1 and overabundance of testers. Since we barely did anything testing in sprint one due to lack of any functionality we decided that we will save testing for sprint 3 and focus purely on getting together a working product with as much functionality as possible in sprint 2. As such, everyone has been assigned to the role of developer for sprint 2.