SwoleMate

Team 7

Sprint 1 Retrospective

**Contributors**: Jaden Zerbe, Kevin Murcia, Ryan Sand, Samuel Ingram, Steven Dellamore

**What went well?**

We’ve created the backbone of our project with a robust client <--> server <--> database model. We’ve also abstracted out functions to connect to the database and insert, update, remove, or request information from the database. This in addition to the user, matches, and conversations structures in our database, and a prototype UI to display that information, has laid the foundation for the rest of our project.

**User Story 1**:

As a user, I would like to be able to create a SwoleMate account.

Task Table:

|  |  |  |  |
| --- | --- | --- | --- |
| Task Number | Description | Time (hr) | Name |
| 1 | Creating User Class | 1.5 | Jaden |
| 2 | Create Login UI | 1 | Jaden |
| 3 | **Create User Database** | 3 | Steven |
| 4 | **Unit Test for Creating a User. Test Input and Failure/Success Cases** | 2 | Steven/Jaden |

**Completed:** We figured out we do not really need a user class but fetching the data from the database via the server was a better option. But we also found another good way to pass information between screens.

**User Story 2**:

As a user, I would like to easily login and logout of my profile.

Task Table:

|  |  |  |  |
| --- | --- | --- | --- |
| Task Number | Description | Time (hr) | Name |
| 1 | Send hashed password to Server to verified | 2 | Kevin |
| 2 | Create login functionality on create account screen | 2 | Kevin |
| 3 | Create a button to logout | 1 | Kevin |
| 4 | Checking password with database | 3 | Steven |
| 5 | Unit tests for logging in and logging out of my profile | 2 | Steven/Kevin |

**Completed:** We figured that we did not need to send the password hashed over https, so we hashed it once it got to the server. We then compared the hashed password to incoming passwords to see if they match. The login UI allows for the user to enter their information and waits for the server to respond before changing pages if successful and displays an error message if not. The logout button logouts out the user and transitions back to login screen

**User Story 3**:

As a user, I would like to create and edit my profile.

Task Table:

|  |  |  |  |
| --- | --- | --- | --- |
| Task Number | Description | Time (hr) | Name |
| 1 | Creating a way to edit User Class | 3 | Ryan |
| 2 | Within the page to edit data, display current data to be edited | 2 | Ryan |
| 3 | Creating an Edit UI | 4 | Sam |
| 4 | Unit Test for Editing the Profile. Cover tests for incorrect/correct formatting. | 2 | Ryan/Sam |

**Completed:** We successfully made an edit screen in which users can edit their profile information and have it saved back to the database successfully.

**User Story 4**:

As a user, I would like to delete my profile if I so desire to.

Task Table:

|  |  |  |  |
| --- | --- | --- | --- |
| Task Number | Description | Time (hr) | Name |
| 1 | Create UI functionality to delete user | 2 | Jaden |
| 2 | Delete User from database | 2 | Sam |
| 3 | Delete match data | 3 | Sam |
| 4 | Unit Tests | 2 | Jaden/Sam |

**Completed:** This one was relatively easy in terms of UI as we just needed a place to put a button for the user. Deleting a user turned out to be rather easy as long as we did not delete all of the users by mistake.

**User Story 5**:

As a user, I would like to create a bio that describes me and includes workout information and statistics.

Task Table:

|  |  |  |  |
| --- | --- | --- | --- |
| Task Number | Description | Time (hr) | Name |
| 1 | Create Profile UI | 1 | Kevin |
| 2 | Creating text fields for the user to create a bio | 2 | Kevin |
| 4 | Link text boxes to Database | 2 | Ryan/Kevin |
| 5 | Unit Test | 2 | Ryan/Kevin |

**Completed:**

We created a text box where the user can write a bio in and then can be saved to the server.

**User Story 6**:

As a user, I would like to share my location to view nearby profiles.

Task Table:

|  |  |  |  |
| --- | --- | --- | --- |
| Task Number | Description | Time (hr) | Name |
| 1 | Connect to GoogleMaps API | 3 | Kevin |
| 2 | Create a radius to match with other people | 4 | Sam |
| 3 | Create table for location in Database, and save User location to database | 2 | Sam |
| 4 | Unit Test | 3 | Sam/Kevin |

**Completed:** We were able to successfully implement the location sharing on our app. We decided to not use GoogleMaps API and choose instead to work with the React Native geolocation API because its built into React Native and it was simple to implement and use. The frontend logs the location whenever a user signs up or logs in, and then the database indexes users based on their location. With this indexing, users can then be found within a certain radius of other users.

**User Story 7**:

As a user, I would like to upload a photo for my profile or open my camera to take a new photo for my profile.

Task Table:

|  |  |  |  |
| --- | --- | --- | --- |
| Task Number | Description | Time (hr) | Name |
| 1 | Adding camera functionality | 3 | Jaden |
| 2 | Able to upload Photo from “photos” | 1 | Jaden |
| 3 | Send chosen photo to database | 1 | Jaden |
| 4 | Saving Photo to database | 2 | Steven |
| 5 | Unit Test | 2 | Steven/Jaden |

**Completed:** Adding Camera functionality was mostly completed. The selection of a photo was done but the actual sending to the server was not. Although once the image is sent, the server should be ready to save the photo to the database.

**User Story 8**:

As a user, I would like to display sports or activities I am interested in on my profile.

Task Table:

|  |  |  |  |
| --- | --- | --- | --- |
| Task Number | Description | Time (hr) | Name |
| 1 | Create UI components to indicate sports | 2 | Jaden |
| 2 | Add fields to the user in the database to hold sports interests | 2 | Ryan |
| 3 | Add edit functionality | 2 | Jaden/Ryan |
| 4 | Unit Test | 1 | Jaden/Ryan |

**Completed:** Creation of the UI was relatively easy and the selection of the sports proved to be easier than expected. An array is used to both log selections and store the data in the database which made things simpler.

**User Story 9**:

As a user, I would like my data to be securely stored.

Task Table:

|  |  |  |  |
| --- | --- | --- | --- |
| Task Number | Description | Time (hr) | Name |
| 1 | Hash password before sending over to server | 2 | Kevin |
| 2 | Authentication of User | 1 | Steven |
| 3 | Making database secure and accessible by front end | 2 | Steven/Kevin |
| 4 | Unit Test | 2 | Kevin/Steven |

**Completed:** We figure that sending a password to the server over https was secure, so we decided that we will hash the password on the server and store it into the database. We then compare a person's login password and hashed password relating to the login email and log in the person if they match. We created a edit class where the frontend can easily view and edit the database depending on what they need.

**User Story 10**:

As a user, I would like to choose and display my fitness experience level.

Task Table:

|  |  |  |  |
| --- | --- | --- | --- |
| Task Number | Description | Time (hr) | Name |
| 1 | Create UI components | 2 | Sam |
| 2 | Add API and Database functionality for fitness level | 3 | Steven |
| 3 | Add edit functionality | 2 | Sam/Steven |
| 4 | Unit Test | 1 | Sam/Steven |

**Completed:** The edit function on the server side currently supports being able to add these experience fields, but we do not have anything on the front end to support this.

**User Story 11**:

As a user, I would like to easily view all my matches and direct messages.

Task Table:

|  |  |  |  |
| --- | --- | --- | --- |
| Task Number | Description | Time (hr) | Name |
| 3 | Accessing past matches and sending to match page UI | 3 | Ryan |
| 4 | Access past messages from the database | 2 | Ryan |
| 5 | Unit Test | 2 | Ryan/Jaden |

**Completed:** Routes now exist to retrieve data correctly from the match and conversation collections in the database. This information can then be called for in the frontend and returned as a json string.

**User Story 12**:

As a developer, I would like to write readable and commented code

Task Table:

|  |  |  |
| --- | --- | --- |
| Task Number | Description | Time (hr) |
| 1 | Weekly progress reports | 10 (2 x 5) |
| 2 | Weekly code review | 10 (2 x 5) |
| 3 | Planning Documents | 15 (3 x 5) |
| 4 | Create documentation throughout the coding progresses | 10 (2 x 5) |

**Completed:** We met every Tuesday and Thursday at 5:30 to go over what we have done, things we are struggling with, and what we are currently working on. These meetings helped us work as a team to assemble our project in a timely fashion.

**What did not go well?**

We will endeavor to not do most of the sprint the night before next sprint, we will achieve this via better estimations on time expected per user story. Attempting to write tests for every user story also proved to be an unrealistic goal in some instances.

**User Story 11**:

As a user, I would like to easily view all my matches and direct messages.

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | Create the Match conversation page UI | 2 | Jaden |
| 2 | Create the direct Messages UI to display past/recent messages | 2 | Jaden |

**Not Completed:** The user interface was however not completed so the server information is not yet displayed. A basic screen with minimal components was created but nothing to actually display messages received from the server.

**User Story 5**:

As a user, I would like to create a bio that describes me and includes workout information and statistics.

|  |  |  |  |
| --- | --- | --- | --- |
| 3 | Create bio, and statistics tables in database | 2 | Ryan |

**Not Completed:**

We didn’t implement the statistics tables like we wanted to but they can still be put into the bio manually by the user if they really wanted to.

**How should we improve?**

Over the course of the first sprint, we learned many valuable lessons. Of all those, the most important was friendship. Communication plays an important role in team projects, especially with regards to software engineering. Through the times we spent together, we learned how to collaborate on ideas and communicate new technical ideas. One area of improvement that we want to work on though is having the front and and back end teammates communicate better on api routes and what is meant to be sent back and forth between the client and the server. Unnecessary confusion arose when data was being passed to the back end in a format that was not expected or vice versa. We intend to start writing some more documentation or references on what our data should look like.

One of the biggest issues we had during the first sprint was a lack of time management. We ended up not fully getting committed until later in the sprint. The first week of the sprint we spent researching the frameworks and setting up the whole project. Moving forward we want to be able to start on our user stories in the first week giving us time to comfortably finish all of them in the time allotted.

Estimations on time expected per user story can be adjusted. Namely, it may be necessary to estimate each user story to take slightly more time than expected. Also, some of the creation of the UI will take less time while the actual logic and connection of user collected date to the server and database will take longer.

Additionally, we want to write tests for every user story to make testing our app more streamline. For the front end, we can look into frameworks for testing react native apps or alternatively list out steps to procedurally go through to test the UI.

Although we did not get every user story done, we learned that spreading the workload evenly across the sprint will help us be more efficient in the long term. Also, we can never have too much communication between team member when it comes to integrating different components of our application.