SwoleMate

Team 7

Sprint Planning Document

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

**Sprint Overview**

This sprint we’re going to be focusing on setting up the foundations of our app. This will include setting up our development environment and database. Following this, we will implement our models and classes in order to allow for the creation and authentication of users. Next, we will allow for further editing of user profiles. Once that is complete, we will begin with the preliminary steps of our matching system by creating the proper views and actions to allow a user to scroll through profiles.

**SCRUM Master:** Ryan Sand

**Meeting Schedule**

Tuesday Thursday 5:30pm - 7:30pm

**Risks and Challenges**

Due to this being the first sprint, we anticipate many challenges relating to the initial setup for the implementation of our product. The added challenge for us all to become fluent with our development environments will take up a significant amount of time in addition to our normal predictions for tasks. This will also be the stage where we set our group coding and git standards for the rest of the project. As we set up the backbone of our application, we risk creating many structural issues that will have to be identified and fixed before being able to move on with integrating our tasks, creating another time sink as well.

**Current**​ ​**Sprint**​ ​**Detail**

**User Story 1**:

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

User Story Owner: Jaden Zerbe/Steven Dellamore

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 |

Acceptance Criteria:

* A user object is created with the proper information in the backend.
* The user’s profile data is successfully stored in the database.
* A screen exists for the purpose of creating a user account.
* Implement constraints to avoid improper information being entered and ensure working status with unit tests.

**User Story 2**:

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

User Story Owner: Kevin Murcia/Steven Dellamore

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 |

Acceptance Criteria:

* Login function securely sends password for verification.
* Properly matches entered password to correct password in database and succeeds or fails appropriately.
* A screen exists for the purpose of logging into the app.
* Logout button is functional within the app after logging in.
* Tests for password hashing/matching and button functionality work correctly and are passed.

**User Story 3**:

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

User Story Owner: Samuel Ingram/Ryan Sand

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 |

Acceptance Criteria:

* A screen exists where one can enter and modify user data.
* Current data is displayed on the page to be edited.
* Changes to user data are kept in database.
* Tests to confirm the successful change of data and constraints for data fields pass.

**User Story 4**:

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

User Story Owner: Jaden Zerbe/Samuel Ingram

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 |

Acceptance Criteria:

* Screen or button to delete a user exists.
* Transfers user to login/register screen after deletion.
* Login attempts not longer log user in after deletion.
* Tests verify that user can no longer be found and data no longer exists in database.

**User Story 5**:

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

User Story Owner: Kevin Murcia/Ryan Sand

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 |
| 3 | Create bio, and statistics tables in database | 2 | Ryan |
| 4 | Link text boxes to Database | 2 | Ryan/Kevin |
| 5 | Unit Test | 2 | Ryan/Kevin |

Acceptance Criteria:

* A profile screen exists that displays user bio and other statistics.
* Statistics and bio information can be stored under correct user in database.
* Tests ensure that bio information is stored correctly in database and displays properly for the user.

**User Story 6**:

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

User Story Owner: Kevin Murcia/Samuel Ingram

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 |

Acceptance Criteria:

* Location can be gathered using Google Maps API.
* There exists a way to set a preferred viewing distance.
* The location of the user is stored in the database.
* Tests verify correct updating of location through Google API in database and proper display of users with set radius.

**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.

User Story Owner: Jaden Zerbe/Steven Dellamore

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 |

Acceptance Criteria:

* Camera can be used within app
* Photos can be loaded into app and sent to database
* Photos are stored in database and can be sent back to a client upon request
* Tests that a photo can be taken and stored in database and server can send photos back to client to be displayed

**User Story 8**:

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

User Story Owner: Jaden Zerbe/Ryan Sand

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 |

Acceptance Criteria:

* Icons or text exist to represent sports
* Sports interests can be modified
* Database can store information on interested sports and can provide said information
* Tests verify data gathered from client are correctly stored in database

**User Story 9**:

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

User Story Owner: Kevin Murcia/Steven Dellamore

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 |

Acceptance Criteria:

* Be able to have a function hash the password and send to server
* Able to tell if a user is in our database and prevent people from accessing our database
* Preventing frontend commands to directly access our database, backend must be a filter
* Testing front end commands to make sure the backend is secure against front end attacks

**User Story 10**:

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

User Story Owner: Samuel Ingram/Steven Dellamore

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 |

Acceptance Criteria:

* A UI exists that displays information about the users fitness abilities
* The fitness level of the user can be stored
* Fitness abilities can be edited and database records can be updated
* Tests for ensuring the correct data was gathered and stored and can be retrieved

**User Story 11**:

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

User Story Owner: Jaden Zerbe/Ryan Sand

Task Table:

|  |  |  |  |
| --- | --- | --- | --- |
| Task Number | Description | Time (hr) | Name |
| 1 | Create the Match conversation page UI | 2 | Jaden |
| 2 | Create the direct Messages UI to display past/recent messages | 2 | Jaden |
| 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 |

Acceptance Criteria:

* A screen exists where user matches are displayed
* A screen exists after selecting match to send instant messages
* The server can access the database and send back all matches for a particular user
* The server can manage incoming and outgoing messages and properly store them
* Tests that messages are sent and received properly including appearance on screen
* Tests account for race conditions

**User Story 12**:

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

User Story Owner: Jaden, Kevin, Steven, Sam, Ryan

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) |

Acceptance Criteria:

* Creating reports that describe what parts of the sprint we have finished, currently working on, and have not started yet
* Look over each other code to make sure it is readable and commented well
* Create a document that will give us a time frame of what we need to do and how fast we need to do it
* Create documentation of the different steps we have done throughout the process of our project.

**Backlog**

1. **~~As a user, I would like to be able to create a SwoleMate account.~~**
2. **~~As a user, I would like to create and edit my profile.~~**
3. As a user, I would like to login through Facebook or Google.
4. As a user, I would like to be able to reset my password if I have forgotten it.
5. As a new user, I would like a simple tutorial the first time I use the app.
6. **~~As a user, I would like to create a bio that describes me and includes workout information and statistics.~~**
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.~~**
8. **~~As a user, I would like to share my location to view nearby profiles.~~**
9. As a user, I would like to hide some information on my profile from being seen by other users.
10. As a user, I would like to change the distance that I search for other users in.
11. As a user, I would like to easily view and scroll through other user’s profiles.
12. As a user, I would like to intuitively accept or reject other user’s profiles through simple gestures.
13. As a user, I would like to report malicious, spam, or fake accounts.
14. As a user looking for matches, I would like to tap a profile to view more information on that profile.
15. As a user, I would like to receive notifications when I have a new match or message.
16. **~~As a user, I would like to easily view all my matches and direct messages.~~**
17. As someone with a match, I would like to direct message other profiles who I have matched with.
18. As a user, I would like to unmatch with profiles I have matched with.
19. As a user, I would like to be able to keep track of my exercise stats.
20. **~~As a user, I would like to display sports or activities I am interested in on my profile.~~**
21. As a user, I would like to filter what profiles are shown to me.
22. As a user, I would like to add future workouts to a calendar.
23. As a user, I would like to share my workout calendar with users I match with.
24. As a user, I would like my previous workouts to be added to a workout history on my calendar.
25. **~~As a user, I would like to easily login and logout of my profile.~~**
26. **~~As a user, I would like to delete my profile if I so desire to.~~**
27. **~~As a user, I would like my data to be securely stored.~~**
28. As a user, I would like to share another user’s profile with others who might be interested.
29. As a user, I would like to display or share any milestones I reach in my workouts.
30. As a user, I would like to be able to hide my profile from other users and not show up in their results.
31. As a user, I would like to be able to switch between imperial and metric units.
32. As a user, I would like a help/FAQ section on the app.
33. As a user, I would like to set goals that are recorded on my profile.
34. As a user, I would like to set a favorite gym or workout location.
35. **~~As a user, I would like to choose and display my fitness experience level.~~**
36. As a user, I would like to access the different sections of the app through a side menu accessible with the press of a button.
37. **~~As a developer, I would like to write readable and commented code~~**