Golf Player Time Management

Alyssa Beeker, Bailey Carlin, Daniel Teel, Elijah Hunt, Adam Horle

**Table of Contents**

1. Project Vision

1.1. Backgrounds

1.2. Socio-economic Impact, Business Objectives, and Gap Analysis

1.3. Security and ethical concerns

1.4. Glossary of Key Terms

2. Project Execution and Planning

2.1. Team Information

2.2. Tools and Technology

2.3. Project Plan

2.4. Best standards and Practices

3. System Requirement Analysis

3.1. Function Requirements

3.2. Non-functional Requirements

3.3. On-Screen Appearance of landing and other pages requirements.

3.4. Wireframe designs

4. Functional Requirements Specification

4.1. Stakeholders

4.2. Actors and Goals

4.3. User stories, scenarios and Use Cases

4.4. System Sequence / Activity Diagrams

5. User Interface Specifications

5.1. Preliminary Design

5.2. User Effort Estimation

6. Static Design

6.1. Class Model

6.2. System Operation Contracts

6.3. Mathematical Model

6.4. Entity Relation

7. Dynamic Design

7.1. Sequence Diagrams.

7.2. Interface Specification

7.3. State Diagrams

8. System Architecture and System Design

8.1. Subsystems / Component / Design Pattern Identification

8.2. Mapping Subsystems to Hardware (Deployment Diagram)

8.3. Persistent Data Storage

8.4. Network Protocol

8.5. Global Control Flow

8.6. Hardware Requirement

9. Algorithms and Data Structures

9.1. Algorithms

9.2. Data Structures

10. User Interface Design and Implementation

10.1. User Interface Design

10.2. User Interface Implementation

11. Testing

11.1. Unit Test Architecture and Strategy/Framework

11.2. Unit test definition, test data selection

11.3. System Test Specification

11.4. Test Reports per Spring

12. Project Management

12.1. Project Plan

12.2. Risk management

13. References

**1. Project vision**

1.1. Background

The name of our project is Gold Player Time Manager (GPTM). GPTM has two components, a cloud-based player management application and a mobile phone application. The issue we are aiming to solve with golf is the player frustration due to the lack of easy coordination between each player on a course, as well as communication between golf course staff and players. With the nature of golf being played on a large field, players cannot reach each other to determine wait times themselves. This is also an issue that golf course staff cannot currently alleviate effectively due to the lack of online communicative technology, as they still rely on manual solutions to inform players. Not only does this lack of technology affect the issue of wait times, but players also cannot easily request assistance or services. Golf course players and staff both lack effective means of communication involving modern online technology.

1.2. Socio-economic Impact, Business Objectives, and Gap Analysis

Our app will improve the down time between rounds for players, this optimization will make playing golf at a participating location more favorable. This means better business for a golf course that utilizes our app as well as the ability to host more players and games. It also gives players easier access to a golf course’s services, boosting sales for refreshments and additional rounds. This could potentially create new jobs on golf courses who use our app.

1.3. Security and ethical concerns

Since GPTM relies on GPS location tracking, location privacy is the largest security concern. To avoid attacks, we will be using two secure GPS services, Smart Location and the Google Maps API. Another concern is account information privacy, because we require our players and admins to log in to use our services.

The biggest ethical concern of our project is that it introduces more work for golf course employees. This can heighten stress levels for employees and thus it is important for us to make our app highly functional.

1.4. Glossary of Key Terms

* Golf Player Time Manager (GPTM): This is the name of our project, abbreviated as GPTM.
* Golf Player/Player: This refers to the person participating in a game of golf in real life. They do not work for the golf course and are using our app as a standard user. This type of user is referred to as a “Player” in our design.
* Golf Course Employee: This refers to the person who is employed at a golf course and will be involved in our app through golf course management, tending to services and requests.
* Golf Course Manager/Admin: Refers to the person that sets a request to have their golf course be geofenced as well as us (GPTM team)
* Application Programming Interface (API): a set of functions and procedures allowing the creation of applications that access the features or data of an operating system, application, or other service. More specifically, in this project, things like Firebase, Google Maps, etc.
* Global Position System (GPS): Navigational system that tells the player where they are and if they are in close proximity to or at a golf course
* Geofence: Geographical location in which some arbitrary code will execute / player is given special permissions, i.e. in this case access specific golf course pages

**2. Project Execution and Planning**

2.1. Team Information

The members in our team are Alyssa Beeker, Bailey Carlin, Daniel Teel, Elijah Hunt, and Adam Horle. Together our team has a strong background in mobile and web app development, as well as user interface design. As a team we are adept at Java, Typescript, HTML, and CSS which are required for GPTM.

2.2. Tools and Technology

The primary languages we will be using are Java for the mobile app, and Typescript and HTML with CSS for the web app. For our IDEs we will be using Android Studio and the text editor of each team members choice for web development. We chose Android Studio because it is Android's official development software and some team members have experience using it. We will use Microsoft Visio for flowchart design. We chose this because it is accessible to each teammate and is a simple diagramming tool everyone is familiar with. Our AI technology will be from Tensorflow, specifically we will be using Tensorflow lite because it is open source. For GPS services, we will be using Smart Location and the Google Maps API. Smart Location is an Android library project which will simplify the usage of location providers and activity recognition. Google Maps API offers a map for us to use in our application for our location features. Our databases are SQLite and Firebase. SQLite is a common database management system that our team is familiar with. Firebase is another backend service that allows us to work on both our mobile and web application, and offers account authentication. Angular 9 is being utilized as the framework for the web app. Each tool was specifically chosen because it was compatible, easy to understand, and modern.

For collaboration and communication, we will be using Github for version control, Google Drive for all of our written and visual documents, and Trello for planning and delegation. For further communication we have set up an accessible chat group for the team to use.

2.3. Project Plan

|  |  |  |
| --- | --- | --- |
| Sprint | Task | Due Date |
| 1 | User Auth UI/UX | 1/20/20 |
| 1 | User Auth User Stories | 1/20/20 |
| 1 | Develop Mobile User Auth | 1/20/20 |
| 1 | Mobile User Auth QA | 1/22/20 |
| 1 | Golf Course Registration User Stories | 1/22/20 |
| 1 | Continuous Integration Tests | 1/22/20 |
| 1 | Golf Course Registration UI/UX | 1/22/20 |
| 1 | Sprint 1 Presentation | 1/22/20 |
| 2 | Admin Site UI/UX | 1/24/20 |
| 2 | Admin Site User Auth User Stories | 1/24/20 |
| 2 | Develop Admin User Auth | 1/26/20 |
| 2 | Admin Site Nav User Stories | 1/31/20 |
| 2 | Develop Admin Nav/Layout | 1/29/20 |
| 2 | Set Up Geofencing | 2/3/20 |
| 2 | Admin Site Auth, Nav QA | 2/2/20 |
| 2 | Game Update User Stories | 2/2/20 |
| 2 | Admin Course Overview User Stories | 2/3/20 |
| 2 | Game Update UI/UX | 2/3/20 |
| 2 | Sprint 2 Presentation | 2/3/20 |
| 3 | Requests User Stories | 2/16/20 |
| 3 | Develop Course Registration | 2/14/20 |
| 3 | Develop Admin Course Overview | 2/15/20 |
| 3 | Golf Course Registration QA | 2/16/20 |
| 3 | Requests UI/UX | 2/16/20 |
| 3 | Sprint 3 Presentation | 2/17/20 |
| 4 | Rework Course Registration and Overview | 2/29/20 |
| 4 | Play Speed Prompt UI/UX | 3/1/20 |
| 4 | Develop Geofencing in the Admin Site | 3/1/20 |
| 4 | Play Speed Prompt User Stories | 3/1/20 |
| 4 | Develop Requests | 3/1/20 |
| 4 | Requests QA | 3/1/20 |
| 4 | Assistance User Stories | 3/1/20 |
| 4 | Assistance UI/UX | 3/1/20 |
| 4 | Player Overview User Stories | 3/2/20 |
| 4 | Player Overview UI/UX | 3/2/20 |
| 4 | Sprint 4 Presentation | 3/2/20 |
| 5 | Develop Assistance | 3/15/20 |
| 5 | Develop Play Speed Prompt | 3/15/20 |
| 5 | Develop Admin Player Overview | 3/15/20 |
| 5 | Develop Location Service | 3/15/20 |
| 5 | Assistance QA | 3/15/20 |
| 5 | Play Speed Prompt QA | 3/15/20 |
| 5 | Admin Player Overview QA | 3/15/20 |
| 5 | UI Style Mockups for Mobile App | 3/15/20 |
| 5 | Sprint 5 Presentation | 3/16/20 |
| 6 | Admin Support Page | 3/29/20 |
| 6 | Mobile App UI Work | 3/29/20 |
| 6 | Admin Site UI Work | 3/29/20 |
| 6 | Sprint 6 Presentation | 3/30/20 |
| Final | Final Project Poster | 4/13/20 |
| Final | Dry Run Presentation | 4/13/20 |
| Final | Admin Dashboard QA | 4/21/20 |
| Final | Mobile App QA | 4/21/20 |
| Final | Bug Bash | 4/21/20 |
| Final | Final Presentation | 4/22/20 |

2.4. Best standards and Practices

Rather than repeating code in multiple places make a function. Write code as efficiently as possible. Functions for common operations in the app will enable scalability and make things a lot more manageable. When working in Angular linting before a push will promote standardization of code.Only one person should be working on one component at a time to decrease merge conflicts. Also, merging master into the current working branch will help with the prevention of merge conflicts. Leaving comments on code where the purpose or flow is apparent is necessary to enable other team members to start working on the code quicker.

**3. System Requirement Analysis**

3.1. Functional Requirements

Administrators need to be able to register a new course and edit the details of the course. Administrators also need to be able to see current players and their locations as well as game requests.

Players need to be able to see the course details along with wait times. Players also need to be able to make game requests. If they have a registered account then players need to be able to register their scores.

3.2. Non-functional Requirements

The application will have two separate components with a simple and intuitive user interface for each. Information needs to be displayed in an easy to read manner. This means appropriately large buttons with clear and concise language to depict where they take the user. All text in the application must be understandable by the target audience, and also be written to accommodate users with less understanding of the English language.

3.3. On-Screen Appearance of landing and other pages requirements.

The landing page for the Administrator website needs to have all the details of the course divided into the main categories (player details and course details). When an administrator logs in it should be apparent which tab they need to navigate to in order to manage their course efficiently.

3.4. Wireframe designs

**4. Functional Requirements Specification**

4.1. Stakeholders

*Golf players* are the primary users for our application. They operate on the mobile app. As primary users, they will have the biggest part in the design of our app. Firstly, they will want to be able to select a golf course to play at. They will then want the app to assist them throughout playing, this includes directions around the course, tips for playing a certain hole, the time they are taking as well as other players time, player queue information, accurate estimations, and easy access to requesting assistance or refreshments from employees. Other features include general account creation and management.

*Golf Course Managers* are the admin users for our application. They operate on the web app. On the web app, they will want to be able to register their admin account with their golf course, as well as be able to add more golf courses after initial creation if desired. After creation, they will want to be able to edit their golf course’s information, add holes and corresponding information for those holes, receive and manage requests from players, and general account management similar to player account management.

Golf Course Employees (Caddy) are a minor user for our application. Their role is accepting or declining the player requests, as well as fulfilling them. The Golf Course Manager will handle how this is done, but the request should be available to the Caddy no matter how it is done. The Caddy’s do not have accounts on our application of their own.

4.2. Actors and Goals

* Player: When “player” is used as an actor, it’s referring to the person using the mobile application to participate in a game of golf.
* Admin: When “admin” is used as an actor, it’s referring to the person using the web application to manage one or more golf courses.
* Database: Our database is managed through Firebase. When it’s used as an actor, it’s to demonstrate how Firebase interacts with our application.
* Cloud: The cloud we use is also from Firebase. The cloud is an important part of the functionality of our app and therefore it’s important to differentiate it from the Database when diagramming.
* Web App: “Web App” refers to the website that is used by admin accounts. It’s used in diagrams to specify where the activity or sequence is taking place.
* Mobile App: “Mobile App” refers to the mobile application that is used by player accounts. It’s used in diagrams to specify where the activity or sequence is taking place.

4.3. User stories, scenarios and Use Cases

**Mobile App User Authentication**

“As a Golfer, I want to securely log in to the application so that my history, requests, and preferences are saved.”

* When I launch the app, then I should be directed to the authentication screen on start up.
* If I do not have an account, then the following events will happen:
  + If I do not have an account already, then there should be a selectable option for creating one.
  + If I select the “Sign Up” option, then I should be prompted to either sign in with an existing Google account or create a new one.
  + If I do not sign in with an existing Google account, then I will be asked to provide an email and password to sign up.
  + If I input a password, then I should be asked to confirm and type the same password twice.
  + If I do provide a password, then the field will have validators to ensure that it meets minimum security requirements.
  + If I leave any of the fields blank, then the “Sign Up” button will be disabled and I will not be able to proceed.
  + If I provide valid information, then I will be logged in and redirected to the “Check In” screen to register at a golf course.
  + If an email that I want to register is already in use I would like the system to tell me to use a new email
  + If I am registering an account and I forget to put information into one of the fields (email or password) I would like the app to disallow me from registering.
* If I do have an account, then the following events will happen:
  + If I do have an existing account, then I will select “Sign In” on the startup screen.
  + If the email and password match an existing email and password combination, then I will be able to log in.
  + If I provide the correct credentials, then the “Sign In” button will be active and I will be able to select it.
  + If the “Sign In” button is selected, then I will be redirected to the “Check In” screen to register at a golf course.
  + As an admin I would like to have a separate login page.
  + As an admin, to stop spam accounts, I would like that all emails be legitimate, i.e. [test@test.com](mailto:test@test.com) will NOT work.
  + If I am a returning user and I forgot my password I would like to be able to create a new one through email

**Golf Course Registration User Stories**

“As a player, I want to view a searchable list of golf courses and choose which one I want to

Play.”

● After securely logging into the app, I should be directed to the Golf Course Registration

screen.

● As a player, I can see a list of golf courses sorted by distance, with closest at the top of

the list and furthest at the bottom.

● As a player, I can type in a search field at the top of the screen that will allow me to

search for a golf course.

● As a player, after selecting a golf course from the list I will be taken to another screen

with more information about the corresponding golf course, including address, hours,

website URL, and phone number.

● As a player, on the second screen I can select “Yes” or “No”.

● As a player, If I select “Yes” I will be directed to the Main Game screen.

● As a player, If I select “No” I will be returned to the Golf Course Registration list screen.

“As an admin, I want to be able to register my golf course so that players can use it on the app.”

After securely logging into the app, I should be directed to the Golf Course Registration screen.

● As an admin, I can click on a button on the Admin Dashboard page that will redirect me

to the Golf Course Registration page.

● As an admin, I will be prompted to fill out a form asking for my golf course’s name,

address, state, zip code, description, website, and phone number.

● As an admin, I can press submit after filling in every form, which then adds the

information to the database.

● As an admin, if I do not want to submit the form, there will be a return button so that I

can return to the Admin Dashboard.

● As an admin, I will be redirected to the next screen after pressing submit, which will have

a message according to the success or failure of the submission.

● As an admin, on the success/failure screen there will be a return button so that I can

return to the Admin Dashboard.

**Admin Site Login User Story**

“As an administrator, I want to securely login to a dashboard that provides information on current player.”

* When an administrator launches the website, then he will be automatically directed to the sign in page.
* When the administrator reaches the sign in page, two fields (email and password) must be entered.
* When an already created email and password is entered, then the administrator is redirected to the dashboard page.
* When invalid credentials are entered, then the following events will happen:
  + When an administrator enters an invalid email/password, then an error message will be displayed alerting them that they have entered an invalid username and/or password.
  + When an administrator enters no information on one or both of the fields, then an error message will be displayed saying that the field is required.

NOTE: Since this is an administrator dashboard that only they will have access to, there is no registration option on this page.

**Admin Site Navigation User Story**

“As a logged in administrator, I’d like to be able to navigate to multiple tabs corresponding with major functions to keep the layout easier to read.”

* When an administrator logs in, then they will be redirected to the home page.
* When an administrator logs in, there will be a toolbar at the top of the page with the app name and a logout button.
  + If an administrator selects the log out button, then the user’s session will expire and they will be redirected to the login page.
* When an administrator is on the home page, then there will be a navigation bar on the top of the screen under the toolbar.
  + If an administrator is logged in, then the navigation bar will be split into 5 categories.
    - The categories are: home, player overview, course overview, register, and support.
      * If an administrator selects the home tab, then they will be redirected to the home page.
      * If an administrator selects the player overview tab, then they will be redirected to the player overview page with data pertaining to the current users.
        + The data will be split into player name, status, and the course that they’re playing at.
      * If an administrator selects the course overview tab, then they will be redirected to the course overview page with data pertaining to all the courses enrolled.
      * If an administrator selects the register tab, then they will be redirected to the register page and can create a new user account.
        + When invalid credentials are entered, then the following events will happen:

When an administrator enters an invalid email/password, then an error message will be displayed alerting them that they have entered an invalid username and/or password.

When an administrator enters no information on one or both of the fields, then an error message will be displayed saying that the field is required.

* + - * + If valid credentials are entered and the register button is clicked, then the account will be created.
        + If the Register with Google button is clicked, then an account can be created from an already existing Google account.
      * If an administrator selects the support tab, then they will be redirected to the support page with helpful hints.
        + If an administrator is on the support page, then they will have access to a walkthrough on how to register a new player, a link to download the app from the Google Play store, a contact email, and the app’s license.

**Player Overview User Story (Admin Dashboard)**

* As an authenticated administrator, when I select the Player Overview tab then I will be redirected to the Player Overview tab that houses player data for active users at my course.
* As an authenticated administrator who is on the Player Overview tab there are two headers: Player Requests and Active Players.
  + As an administrator, I would like the Player Requests accordion to house a table of requests that shows:
    - The type of request
    - The player making the request
    - The player’s location
    - An action button to acknowledge the request to let the player know that the request is in the process of being fulfilled
  + As an administrator, I would like the Active Players tab to have data pertaining to who is playing and where. The table should include:
    - Player name(s)
    - Player locations
    - Running clock of play time

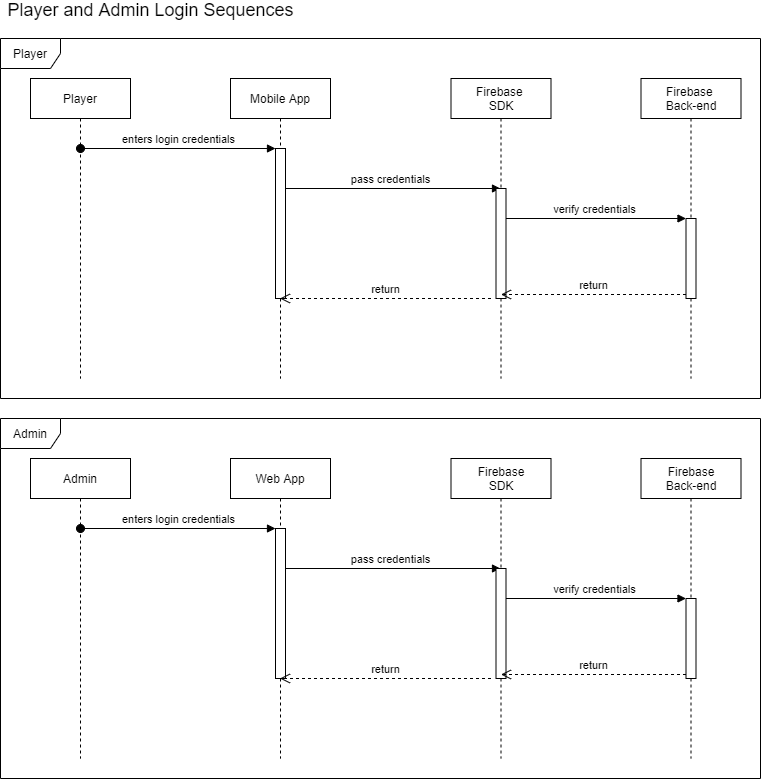
**Requests User Stories:**

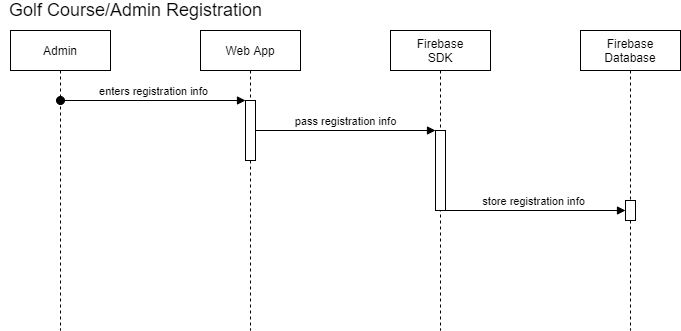
* As a golf course owner, I would like to be able to register my golf course.
* As a golf course owner, I would like to be able to register multiple golf course under my franchise.
* As a player, I would like to be able to use the GPTM app without having to log into some account.
* As a player, I would like to know what the hole map looks like using the GPTM app.
* As a player, I would like to be able to know what the par score is for a given hole.
* As a player, I would like to be able to keep track of what my score currently is using GPTM.
* As a player, I would like to be able to add to my own score using something like a button on GPTM. i.e. pressing = score +1
* As a player, I would like to be able to subtract to my own score using something like a button on GPTM incase I make a mistake. i.e. pressing = score -1
* As a player, I would like to be able to subtract my own score using something like a button on GPTM incase I make a mistake. i.e. pressing = score -1
* As a player, I would like GPTM to tell me if I was above par.
* As a player, I would like GPTM to tell me how much above par I was.
* As a player, I would like GPTM to tell me if I was below par.
* As a player, I would like GPTM to tell me how much below par I was.
* As a player, I would like to know which hole I am currently on using GPTM.
* As a golf course owner, I would like more functionality on the GPTM site.

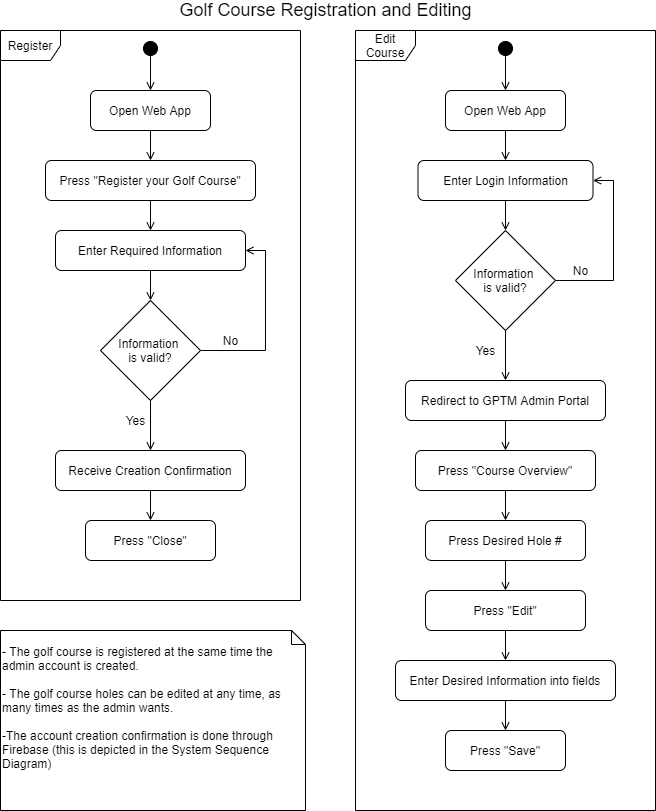
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

* As a player I would like to be able to make a request to the front desk for a food product, a different club, etc.
* As the golf course worker I would like to know which hole a player is at when I go to deliver his request.
* As a golf course owner I would like to be able to decline a request if I don’t want to fulfill the request.
* As a player I would like to receive some sort of notification if the request went through.
* As a player I would like to receive some sort of notification if the request is denied.
* As a player I would like to receive some sort of notification if the request is accepted.
* As a golf course owner I would like some notification for receiving a request.
* As a golf course owner I would like to be able to see all of my requests on some web page.
* As a golf course owner I would like to see what user sent me the request.
* As a golf course owner I think that only registered users will be able to make requests.

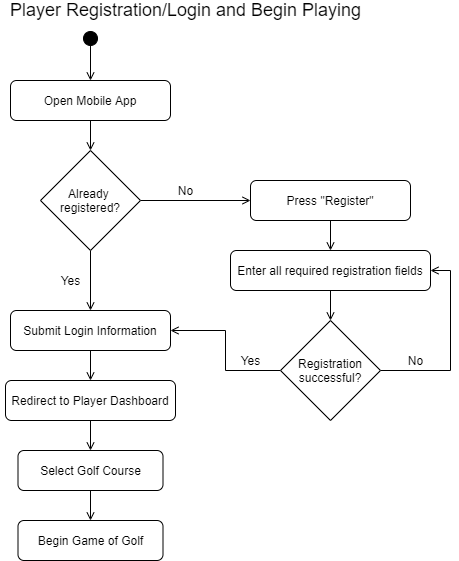
4.4. System Sequence / Activity Diagrams

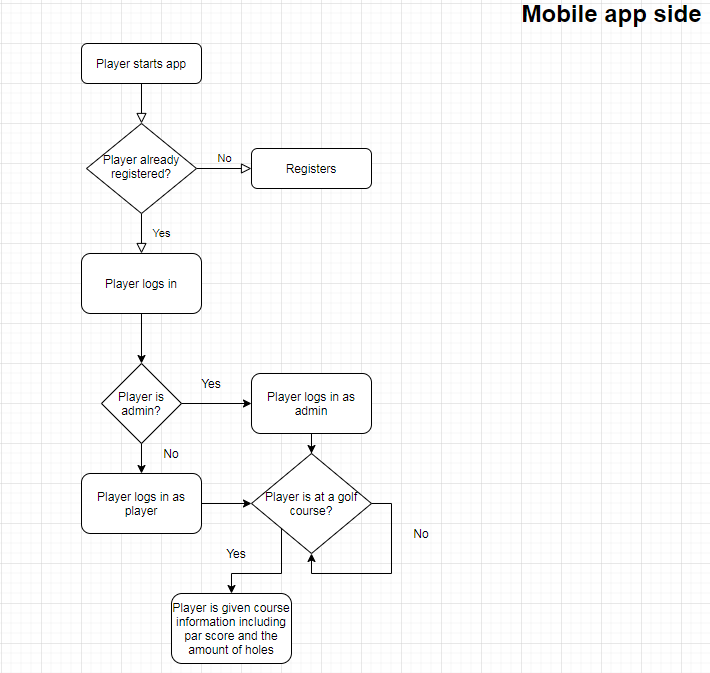






4.4.1 Mobile App Activity Diagram





(Old mobile app activity diagram above)

**5. User Interface Specifications**

5.1. Preliminary Design

Player and Admin Registration Page

For our preliminary registration page design, we will include fields for the user to enter an email and password. There will also be a button for the user to use their google account to register. After the user registers they will be taken to the dashboard page for their respective account type.

Admin Dashboard Page

The Admin Dashboard Page is a webapp only, there is no mobile application for admins as it is unnecessary. On this page the admin can view tabs for their Home, Course Overview, Player Overview, Requests, and Support.

* On the Home page
* The Course Overview page will have a navbar to select the hole and a 5 tile format to show the hole’s data.
* On the Player Overview page the admin can view the players currently playing on their course, including where they are, as well as any player requests.
* On the Support page the admin can view helpful information and tutorials about the webapp. They can also get help with their account here.

Player Dashboard Page

The Player Dashboard Page is a mobile app only. This is where players are directed after logging in. After selecting a golf course, there is where the player will be able to manage their game. They will also be able to request assistance from the golf course from a button on this page. The buttons we expect to include on this page are as following:

* Join Game: This button will allow the player to find a location nearby to select and then join.
* Next Hole: We need the player to tell us when they are done with their hole. After pressing this the app will put them in queue for the next hole.
* Add/Remove Stroke: Simple buttons we will include to help players keep track of their game. There will be a counter next to these buttons.

Request Assistance Page

This page will have a version for admins and players. The admin webapp version is where admins can go to view requests, including the requesting players name, location, and request description. Admins can accept or deny these requests. Players can find the request assistance button on their player dashboard. When that button is pressed, they will be taken to a screen where they fill in their request description. They will then receive a notification when the admin accepts or declines their request.

Error/Exceptions

Since golf courses can have varying wifi or service strength it is important for the sake of user enjoyment that we secure their space in a fair way. In the event a player abruptly disconnects from the app, whether by loss of Internet connection or their phone shutting off from low battery, we plan to save their space in their selected queue or course for 5 minutes. If the player does not log back in before 5 minutes, they will be removed from the queue or course they were registered in. This amount of time allows the player to make adjustments, but also doesn't aggravate other players who may be waiting behind them.

5.2. User Effort Estimation

Ultimately, we do not want our users to expend much effort using our apps. Our web and mobile applications should be quick, intuitive, and easy to make swift actions. The user should not spend more than a few minutes using our app at a time. For example, players should be able to quickly open their phone, add a stroke, and put it back in their pocket. They should not be expected to log back in during their game, nor should they have to navigate through multiple screens or menus to achieve their goal. For admins, their primary function during long term usage of the app is to be able to check requests. All related information to player requests should be upfront and on one page. When they accept or decline a request, a message of confirmation should pop up on the screen but not a dialog box. The request should then disappear automatically as well. This interaction, like the example player one, should only take a few seconds.

**6. Static Design**

6.1. Class Model

6.2. System Operation Contracts

6.3. Mathematical Model

6.4. Entity Relation

**7. Dynamic Design**

7.1. Sequence Diagrams

7.2. Interface Specification

7.3. State Diagrams

**8. System Architecture and System Design**

8.1. Subsystems / Component / Design Pattern Identification

8.2. Mapping Subsystems to Hardware (Deployment Diagram)

8.3. Persistent Data Storage

8.4. Network Protocol

8.5. Global Control Flow

8.6. Hardware Requirement

Our apps require minimal hardware, our apps are accessible on platforms that our target audience is expected to own. Both admins and players need a device that has access to the internet. Admins will need a current web browser, such as Google Chrome or Microsoft Edge. Players will need an Android device that is recent enough to be able to download and run apps from the Google Play Store. Our mobile app will be able to run on most supported Android updates.

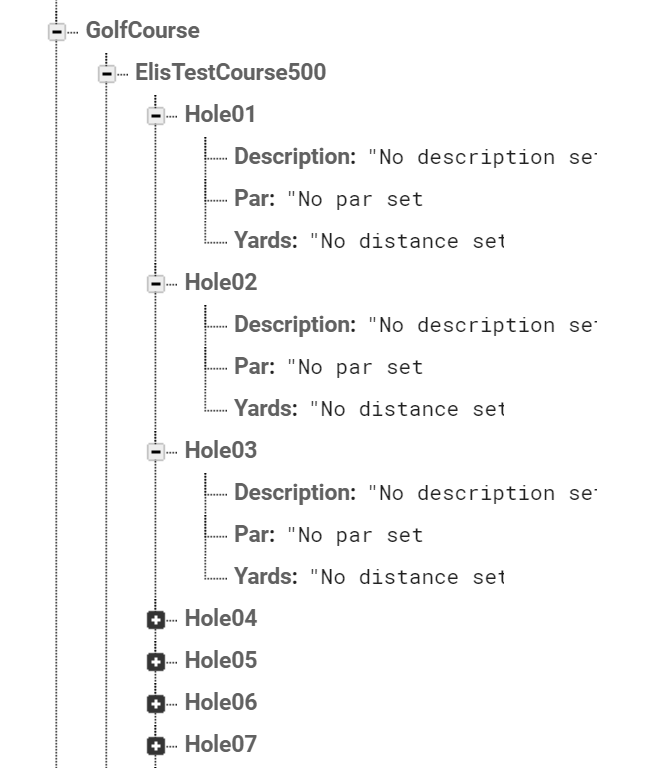
**9. Algorithms and Data Structures**

9.1. Algorithms

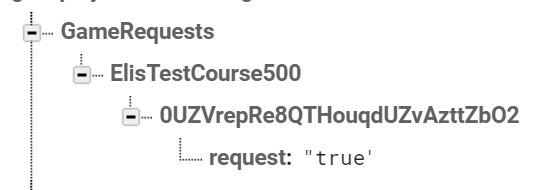
9.2. Data Structures:

Firebase (database)

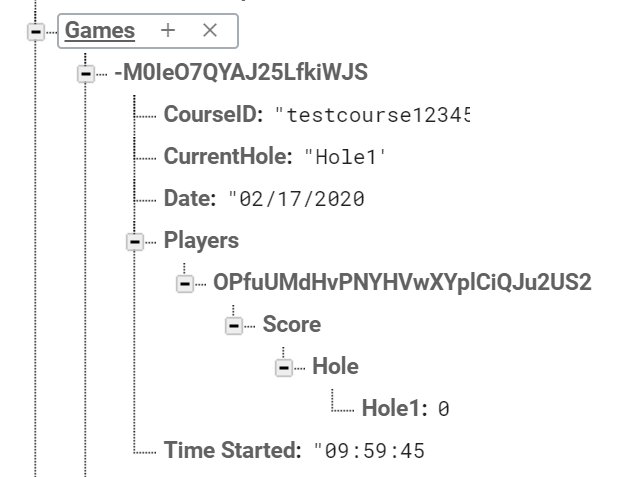
Golf Course Table:



Game Request Table:



Game Request Table:

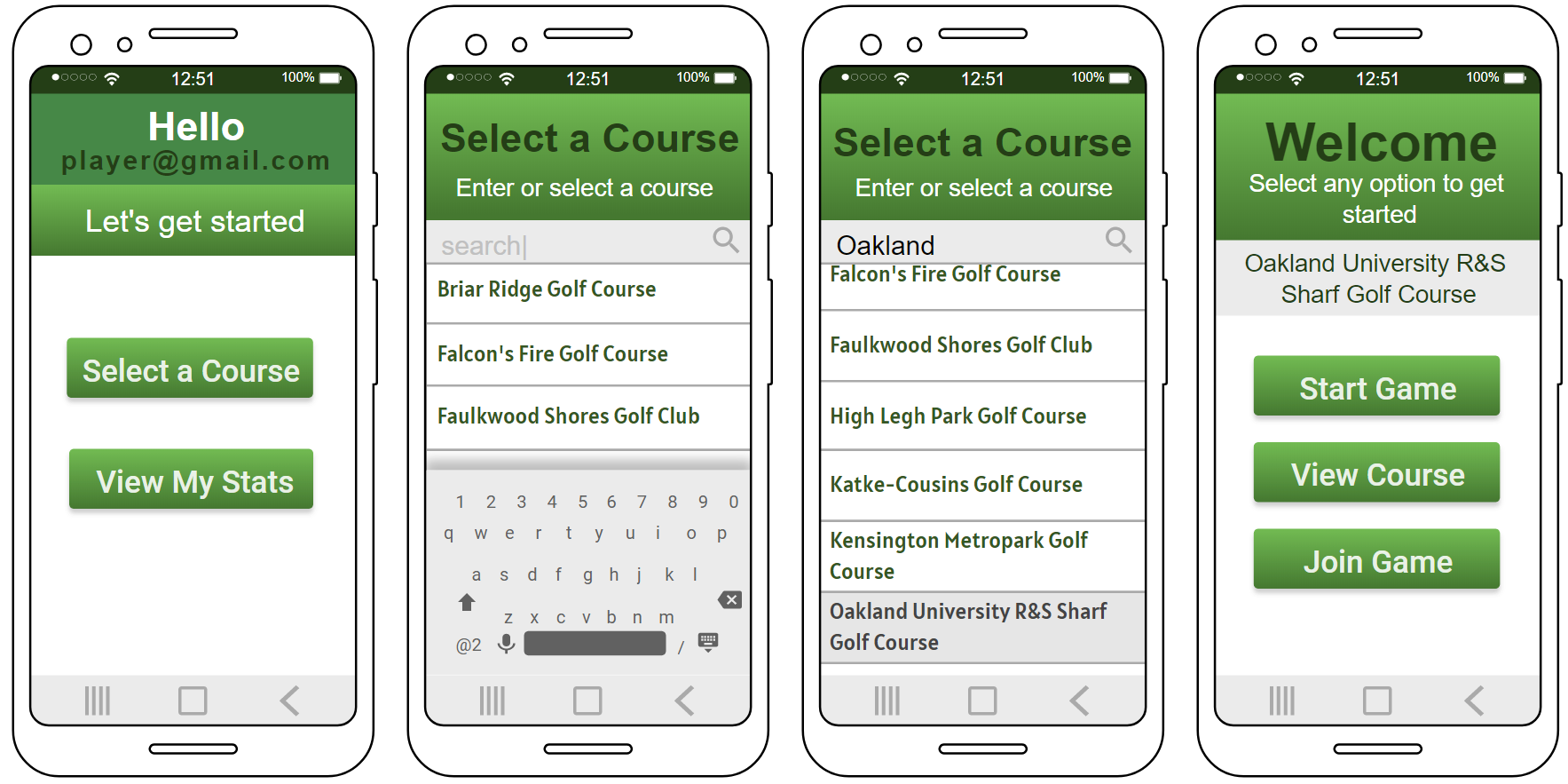
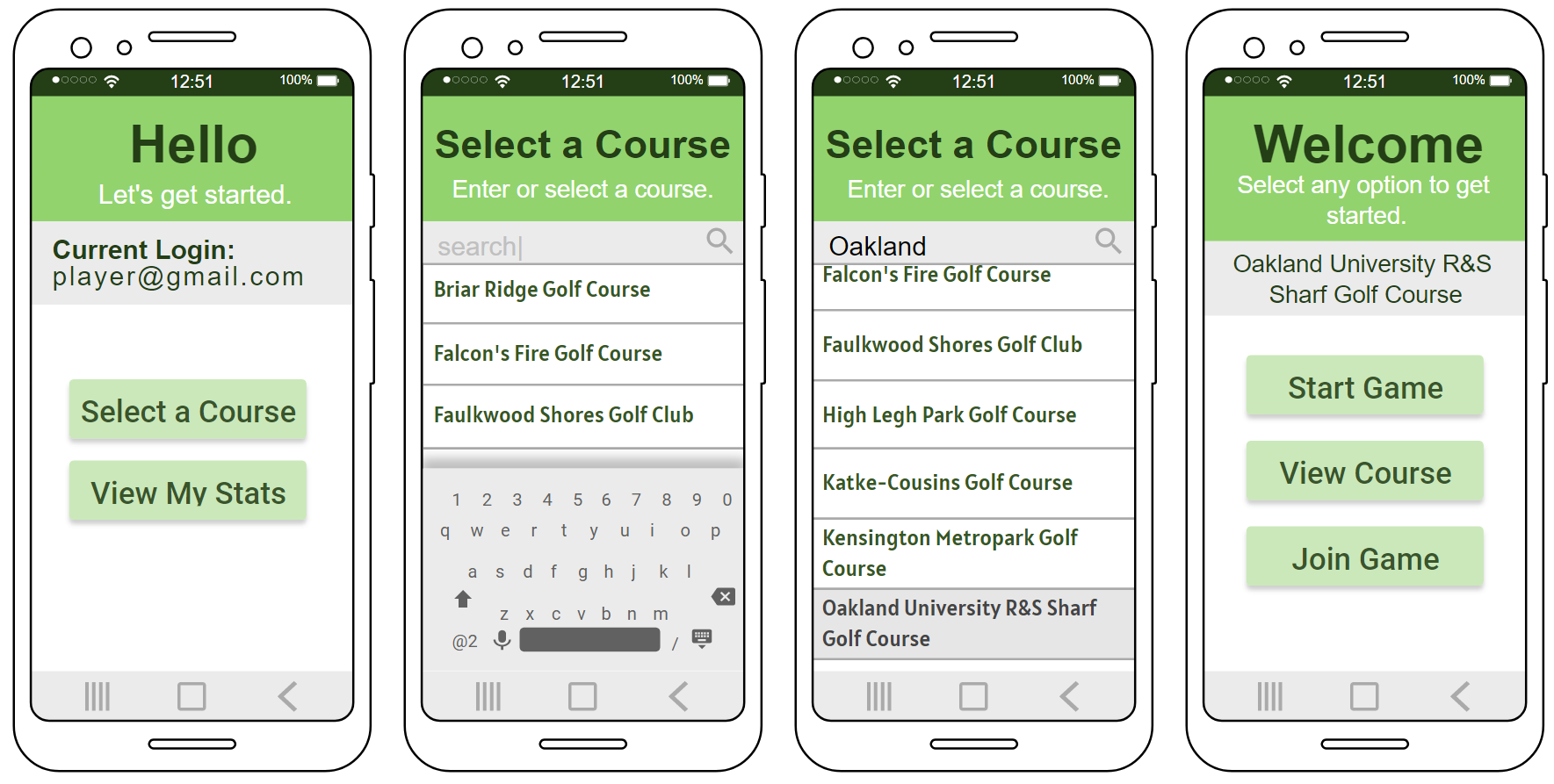
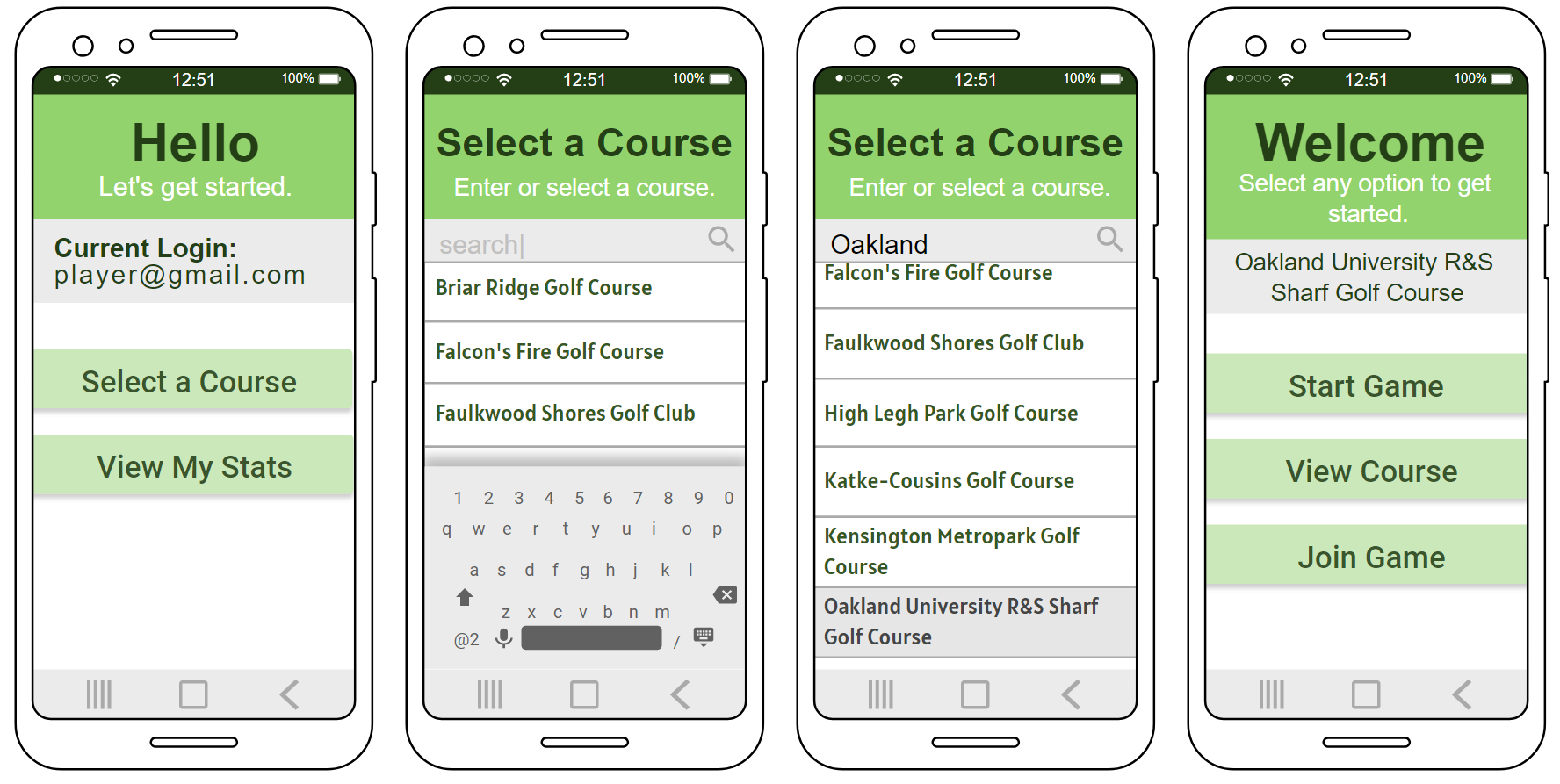
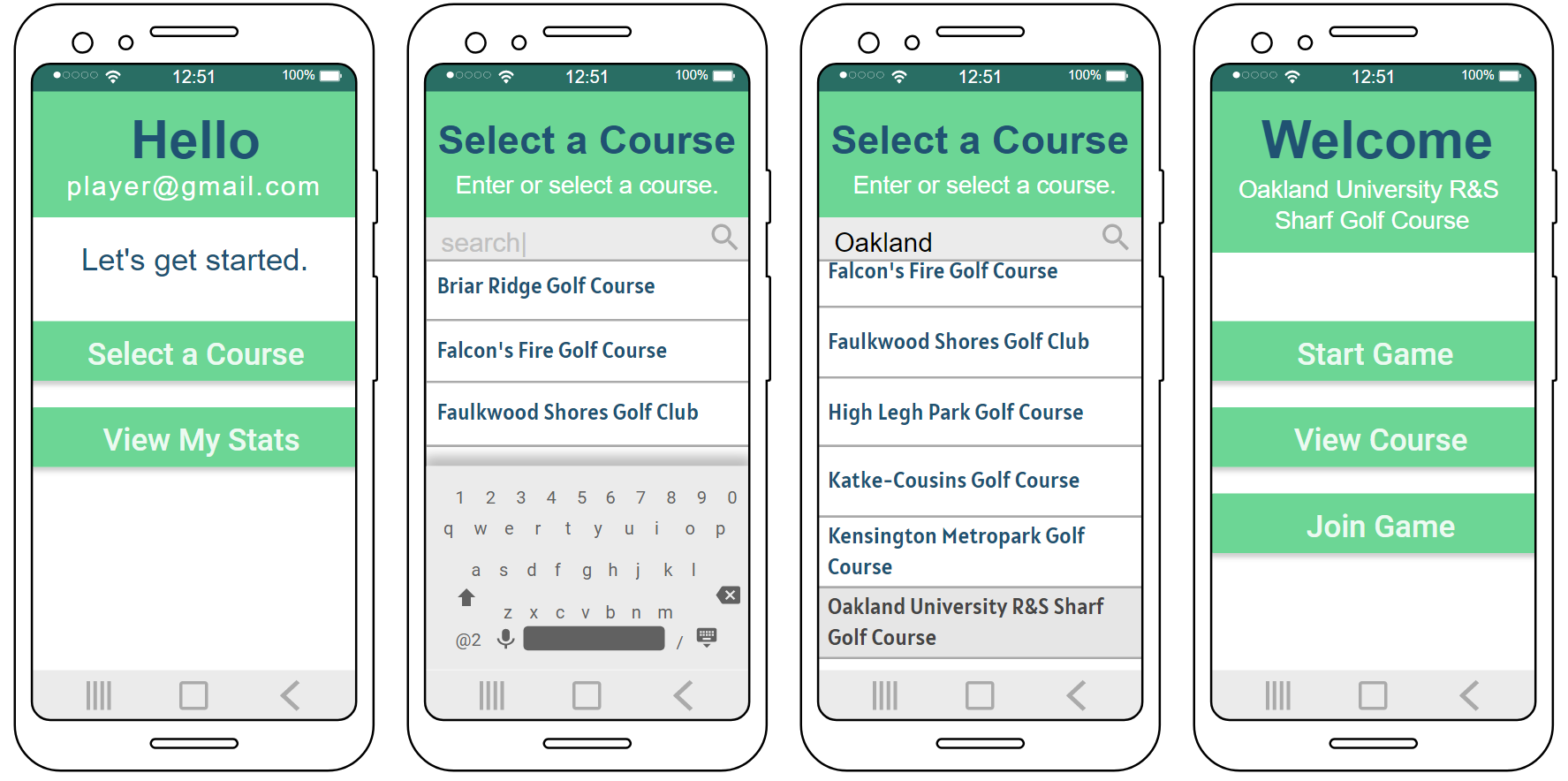


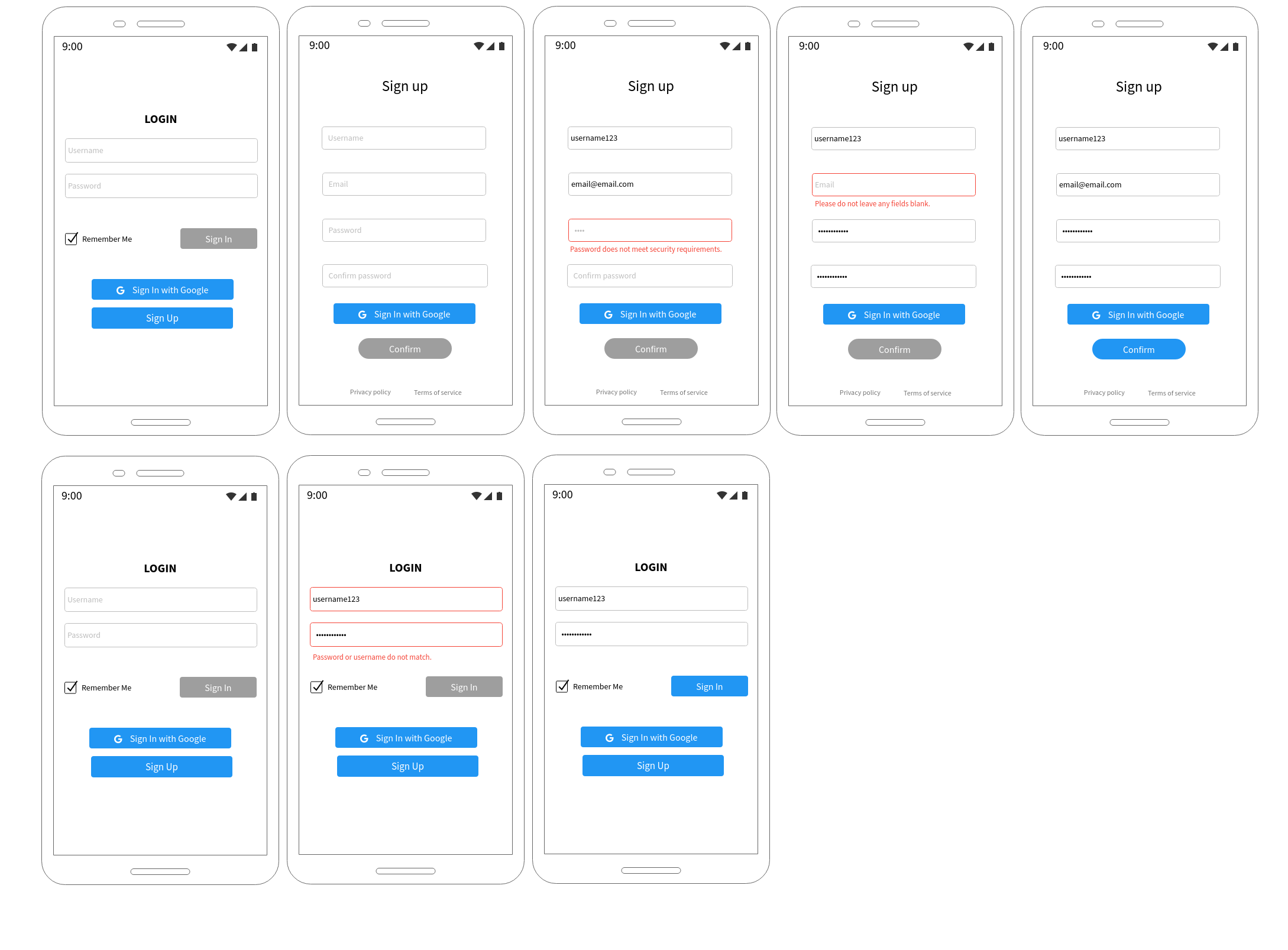
On the Admin Website arrays are utilized when a polygon is drawn. Each point is pushed to the aray and after the “polygoncomplete” listener is triggered the array is finalized. Once the user clicks save this array is pushed to Firebase where it resides in the given hole’s JSON structured tree. All of the entered data is pushed to Firebase as children of nodes. Nodes are created to structure the tree. When data needs to be imported then there is a transaction against the JSON structure of Firebase where the nodes need to be transversed to read the desired data.

**10. User Interface Design and Implementation**

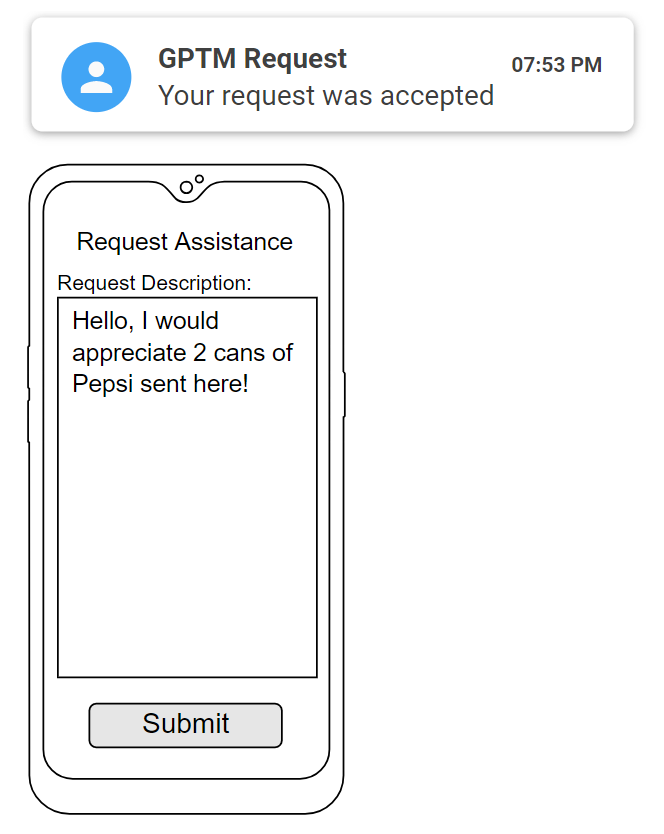
10.1. User Interface Design

Before every development task, we have multiple tasks for design. We first start with a user story to describe what we want a particular feature to do. This is broken up by the user and events involved. Then, we design a mockup based on the user story. This mockup isn’t an exact representation of what the app will look like but it provides the developer a visual aid to work off of.

Iterations of User Interface Design:

Mobile Login (with error messages)

Player Request Assistance (with example notification):



10.2. User Interface Implementation

For implementing the user interface for all of our app and website pages we had to take many scenarios and requirements into consideration. For the mockups, we only needed to make a vague outline of the interface. But when implementing, there are more things to consider, such as visual styles, support for standard user input, as well as testing for the best positioning and functionality. Through research, what we've learned along the way in school, and our instructor's input, we found that a simple and reduced UI worked best. A decluttered and fluid design is modern and works the best for our project. As always during user design implementation optimizing is key. So for implementing, we often had to return to older areas and rework them to this new vision. We want the player to be able to use the app quickly, without thinking much or get distracted, and enjoy a sense of consistency.

In order to get the user through each step quickly and efficiently, we had to test multiple layouts of our player dashboard. We also had to have multiple iterations of language used to shorten the learning curve, and to expand our reach to a larger audience. We were especially critiqued on a distracting layout, so we made sure to start focusing on usability. Instead of going through multiple registration processes for an admin account and golf course, we streamlined it to just one. We also modified the admin dashboard to rely less on accordion menus. Consistency is also important in any app design, as a user on the app you want to see consistent language and locations of entities. We struggled with this during our developmental phases as multiple people were working on implementation, however, time was spent on ensuring this consistency. For example, if we have a submit button, the word used on the button should be "Submit" throughout the entire app or web app. It should not say "OK", "Enter", etc., anywhere. The submit button should also be in the same or similar intuitive location everywhere it appears.

**11. Testing**

11.1. Unit Test Architecture and Strategy/Framework

1. Look at the User Stories to determine what the expected behaviour is.
2. Test the scenarios that will yield the expected results and record these scenarios.
3. Test fringe scenarios that could potentially occur with the validation and functionality in mind and record these scenarios.
4. If there are any scenarios that yield unexpected results file a bug ticket with the scenario and the environment in the description.
5. Reguarily test the scenarios that have filed bugs to see if they were fixed. If so, mark the issue (bug) as closed. If not, keep the issue open.

11.2. Unit test definition, test data selection

11.3. System Test Specification

11.4. Test Reports per Sprint

Each sprint the test report is updated. The resulting reports are attached below.

(Attach QA files at the end of sprint 6)

**12. Project Management**

12.1. Project Plan

12.2. Risk management

In order to manage risks we needed to come up with multiple plans and implementations to be able to handle any problems that arise.

Potential Database Issues

We selected a service that is known to be reliable and we should not encounter any issues with its performance. However, there are issues with the service itself.

Risk mitigation strategy: Firebase is known to have limited capability when scaled up. As well as being costly over time because of the overcharge fees. Due to these known issues, we will limit our potential clients and offered golf courses for our initial roll out and look towards merging our software/app into a larger developer cloud that would be able to handle a larger quantity of traffic for less cost.

Mobile/Web App Communication Issues

Information about golf courses is entered from an Admin on the web application. This has to display on the mobile app for players who are viewing that golf course.

Risk mitigation strategy: If the web and mobile apps do not communicate with one another, a player will be unable to continue his game. In the development of this application, we will utilize services that are known to be reliable and have minimal downtime. Although there will always be a small risk of a player's game ending prematurely due to a connection error, by using well regarded services, we can minimize the frequency of these occurrences.

Location Service Issues

Parts of our app rely on the Google Maps API to not only track location but to display and interact with it in the app.

Risk mitigation strategy: In the event that the Google Maps services API goes down for whatever reason, the app will save the player's current position on the map and give them a grace period to reconnect to the Google Maps services in order to continue the game.

**13. References**

Firebase Documentation:

<https://firebase.google.com/docs/cli/auth#authexport>

<https://firebase.google.com/docs/auth/admin/manage-users>

<https://firebase.google.com/docs/database/web/read-and-write>

<https://firebase.google.com/docs/reference/js/firebase.database.Query>

Google Maps API Documentation:

<https://developers.google.com/maps/documentation/javascript/drawinglayer>

<https://developers.google.com/maps/documentation/javascript/geometry>

<https://developers.google.com/maps/documentation/javascript/places>

<https://developers.google.com/maps/documentation/javascript/controls>

Demo Golf Course Model:

<https://oakland.edu/golf/courses/katke/>