SCAVENGER HUNT APP

by

Ari Zaravelis

A Capstone Project
Submitted to
Ankur Agrawal, PhD

Manhattan College

May 2022

© Copyright 2022 by Ari Zaravelis

All Rights Reserved

Abstract

Scavenger Hunt App is an iOS application which modernizes classical scavenger hunt games. We wanted to make scavenger hunts easier to set up and play while simultaneously adhering to its roots of being interactive and communicative. In other words, our app makes it easier for players to play and creators to create scavenger hunts. This app incorporates many technological features/systems such as QR codes, scanning, messaging, an authentication system, and even a leaderboard system. This app was developed with Xcode using the Swift programming language while also utilizing Firebase for the backend. It is our hope that this app simplifies scavenger hunts and provides fun/enjoyable experiences to our users.

“Keywords:” iOS, Xcode, Swift, Firebase

Acknowledgments

I would like to acknowledge my group mates, Daniel Goldberg, Raziel BenReuben, and Daniel Garcia for continuously working with me throughout the semester to complete and finalize this project. I would also like to acknowledge and thank all my computer science professors for challenging and guiding me throughout the years.

Table of Contents

1. Introduction

Our Scavenger Hunt App is a mobile iOS app intended as a fun, interactive game for people of all ages. Users will be able to either create lobbies or join lobbies, allowing for an easy and intuitive way to play scavenger hunts. The main premise/goal of playing in these scavenger hunts is to locate and scan QR codes which correspond to designated clues created by the host of a lobby. The winner of a scavenger hunt will be the player with the most points which can be viewed in the leaderboard. There is also a messaging system for our app which allows for communication and conversation amongst players.

2. Requirement Specifications

2.1 Application Requirements

This section contains all the functional and non-functional requirements of the software component of the system. It gives a detailed description of the software component and its features.

2.1.1 Functional Requirements

This section includes the requirements that specify all of the fundamental actions of the software system.

- The user shall be able to create an account.
- The user shall be able to sign in.
- The user should have the ability to sign out.
- The user should have the option to either create a lobby or join an already existing lobby. They do not need an account to join a lobby.
- If the user decides to create a lobby, they will be prompted to enter some details for the lobby. These details include creating a name for the scavenger hunt, creating a brief description for it, and creating the clues for the game. The creator will also have to either set a start time or choose the option for starting the game manually. Once the creator has finished setting up the game, they will be emailed the QR codes for the corresponding clues that they created in order to print them out.
- A game host will have the ability to send a notification or hint to all players

- A game host has the ability to end the game at any time, or add time to the timer. Adding time will notify the players, and ending the game will bring all players to the stats screen and end the game.
- A game host has the ability to see the chat and the leaderboard
- If the user decides to join an already existing lobby, they must enter the correct join code.
- Once the user enters the correct join code, they will have the ability to see the name, description, number of clues, and author of the lobby. If the game has not already started, there shall be a “time to start” section for the user to see. If the game has already started, there shall be a “time to end” section for the user to see.
- Once the game officially starts, the players will have the ability to see the list of all the clues needed to win the game. It will also indicate their progress thus far. For instance, if a player already completed a clue, it will be checked marked green. The clues will also list the number of people who have already found the clue.
- The player shall also have the ability to access their camera directly from the app in order to scan the QR codes needed to complete clues. The QR code does not open anything, it just reads the code and compares it to the QR codes stored in the database. When a player scans a correct QR code, it will tell the player so, and update the database and the players storage that they have found the clue. If the QR code is not valid, it will tell the player.

- The players will also have the ability to view the current leaderboard of the game. Players are ranked based on a point system which is determined by the clues completed and the time taken to complete them. This leaderboard constantly changes as the game progresses. Once the game ends, the player at the top of the leaderboard wins.
- Players will be able to chat!

2.1.2 Non-Functional Requirements

This section includes the requirements that specify the behavior and performance of the software system.

- The app shall be relatively easy to use and understand.
- The app shall be able to render to various screen sizes.
- It shall run without too much noticeable lag.
- It should be aesthetically pleasing.

2.2 Database Requirements

The application shall connect to a database hosted online, as this is where a lot of information will be stored concerning accounts, games, and scores.

2.2.1 Functional Requirements

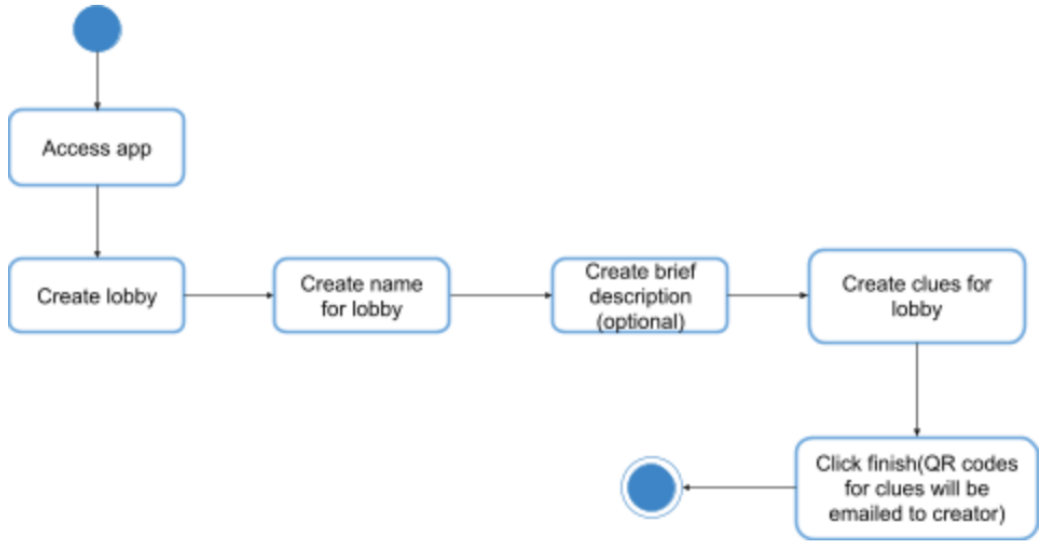
- In the event of a data breach, the attackers should not be able to learn any useful information concerning user's emails, usernames, and passwords (information will be stored in some sort of hashed or encrypted format).
- The app can reliably retrieve all the online information it needs, because all the online information needed will be stored in the online database.
- When a user adds data (creates/joins a scavenger hunt) or creates an account, the changes should be reflected/added into the database.
- Each username and email shall be unique. Passwords should be longer than 7 characters and contain at least one numeric character and one letter.
- Database will store account information (email, username, password, gamesCurrentlyHosting), game information (hosted by, list of players and their specific clue progress, list of clues and their progress, notifications, chat, game code/id, QR code id's, timeToEnd, and list of winners and the time it took to find all clues (a winner is anyone who completed all clues).
- Each player's username in a game must be unique.

2.2.2 Non-Functional Requirements

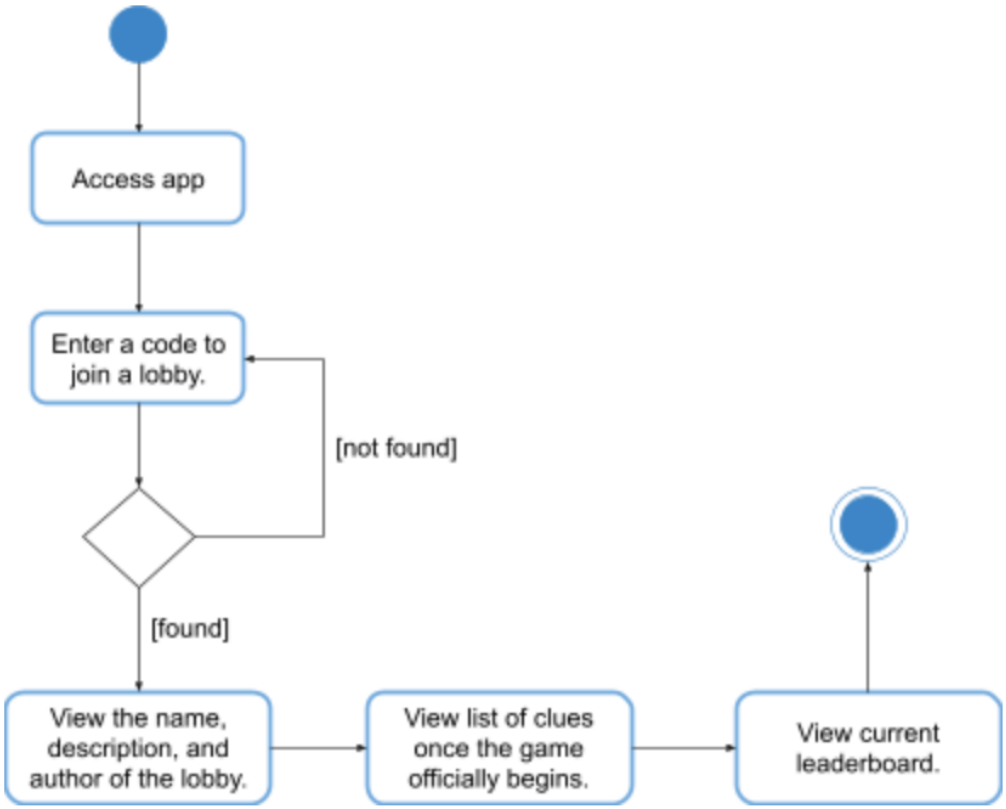
- Access to the database shall be quick (no noticeable lag).
- Hosting the database shall be free.
- The database shall be scalable.
- Should be somewhat secure/have integrity against attacks.

3. Design Techniques

3.1 Activity Diagram – Creating a Lobby



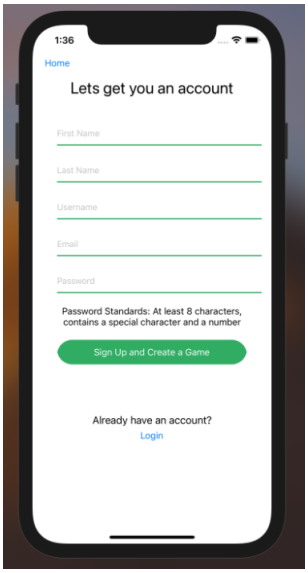
3.2 Activity Diagram – Joining a Lobby



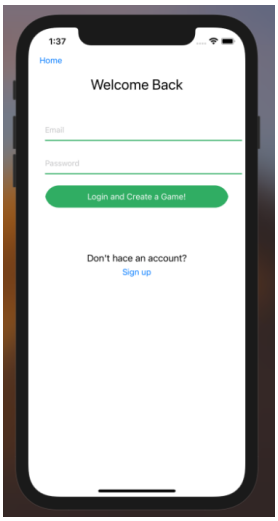
4. Implementation

We created many view controllers each corresponding and managing a set of views which helped in making the app's user interface. Below is a list of some of the most important view controllers and classes we created with a brief description of what they do.

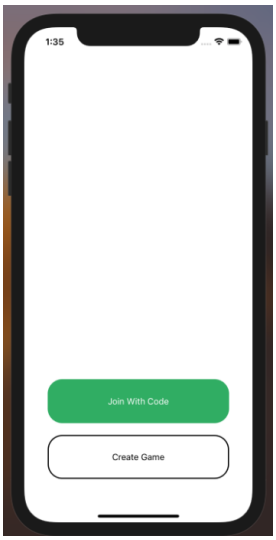
SignUpViewController: Used Firebase's authentication system to provide backend services and easy to use SDKs to help authenticate users of our app. In order to sign up, all fields such as firstName, lastName, username, email, and password should be filled. If all fields are not filled, an error message gets displayed indicating to the user that they should fill all fields. There is also an email validation feature which indicates an error message if an email is not valid. We also included a password validation feature which ensures that a strong password is entered. If a weak password is entered, an error message gets displayed ensuring that the user enters a password that is at least 8 characters, contains a special character, and a number as well.



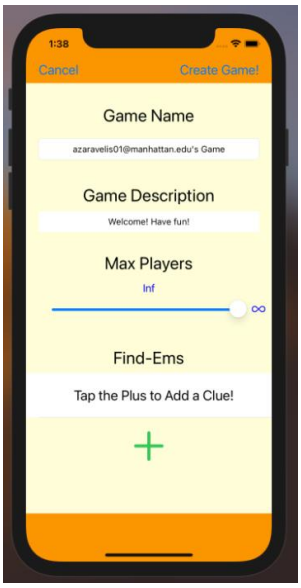
LoginViewController: Both fields for email and password must be entered correctly in order to sign in. If the user enters incorrect credentials, an error message gets displayed indicating to the user that an incorrect email or password was entered.



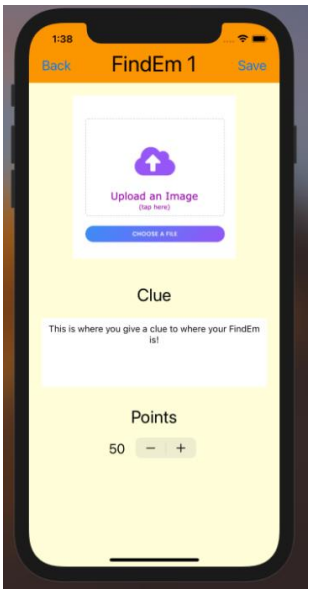
HomeScreenViewController: If user presses on “Join With Code” button, it segues to JoinCodeViewController which will be briefly explained later. If user presses on “Create Game” button, it first segues to SignUpViewController and then will be segued to GameCreationViewController depending on what the user presses.



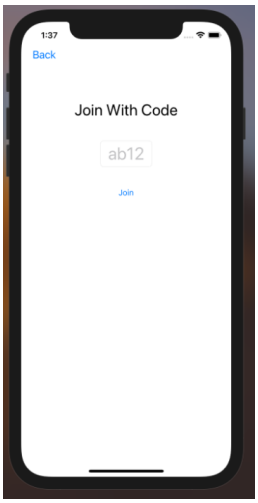
GameCreationViewController: This is where the host of a game will create a game. He/she will be able to enter a game name, a game description, the number of players that will be able to join, and the clues he wants to create for the lobby. When the plus button for adding new clues is pressed, it segues to NewGameDetailViewController.



NewGameDetailViewController: Host has the ability to optionally upload a photo for a clue or enter a short description describing where the clue can be located. Host also has the ability to adjust how many points a particular clue is worth.

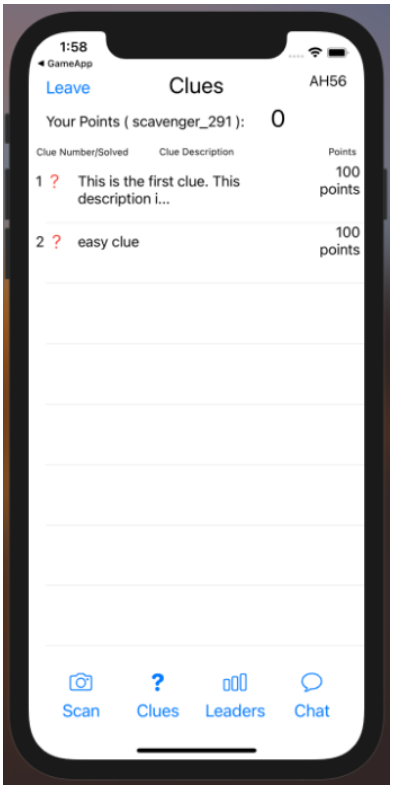


JoinCodeViewController: A player will be able to enter a 4 character unique join code to join a lobby that was previously created. If an incorrect join code is entered, an error message will be displayed letting the player know that he/she entered a join code that does not exist. Once a player enters the correct join code for the game he/she wants to join, a segue to JoinedScreenViewController will occur.

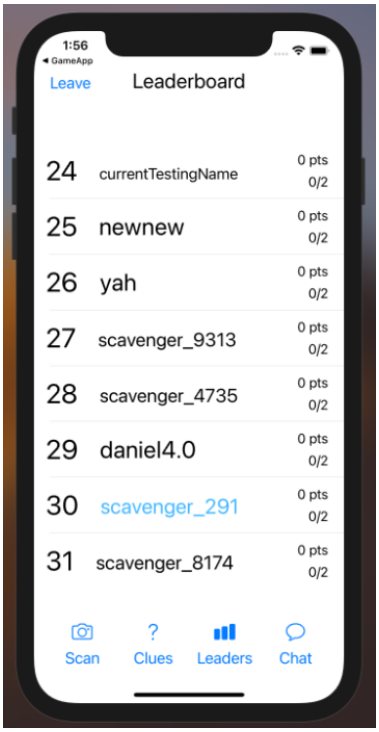


JoinedScreenViewController: A list of all players in the game will be displayed. There is a feature that allows a player to change his/her name. If a player enters a name that is already taken, there will be an error message that gets displayed letting the player know. Once a player presses on “Enter Game” button, it segues to GamePlayViewController.

GamePlayViewController: Displays all clues that players must find along with the points that each clue is worth. These were all created by the host when he/she created the lobby. There is a functionality that when a player clicks on a specific clue, it expands the clue into the full description and displays the count of how many players already solved that clue. It might also display an image if the host posted an image for that specific clue. There is a tab bar in the bottom that lets the player segue into different view controllers.



LeaderBoardViewController: Displays a leaderboard based on the points acquired in a game. This is constantly updating and changing as games progress.



QRCodeScannerViewController: Prompts player to grant access for camera usage. If access is granted, players will be able to use their cameras to scan QR codes which are connected to the database. Once a user scans a QR code corresponding to one of the clues in a game, that player will gain the specified points for that clue.

ChatViewController: Players are able to chat with one another in a lobby.

5. Testing and Debugging

Throughout each sprint cycle, we would continuously test our app, especially after adding new functionalities. Since our app was highly dependent on the backend, we ensured that we tested our database and avoided issues like overriding data and querying conflicts. At first, we would only test our app using the simulators that Xcode provided, however as time progressed, we decided to start testing with our own iPhones since we needed to access hardware such as the camera to test the scanning functionality. We made a lot of silly mistakes throughout the process of building this app, such as improper segueing, mistakes in querying to Firebase, etc. However, we would always debug these issues and continuously solve these problems as they arose.

6. Discussion

a. Project Discussion

Our Scavenger Hunt app fulfilled the promises we made at the beginning of the semester. We were pleased with the app's features that were implemented, as well as its performance and aesthetics in general. In terms of difficulty, we had issues combining individual work from all group members into a single version after every sprint cycle. We used GitHub throughout the sprint cycles, but a lot of mishaps occurred along the way, which was probably due to our inexperience with using it. Regarding my primary and personal contributions to the app, my main focus was in developing the QR code functionality. I worked on QR code generation as well as the email functionality that would send emails to hosts containing the QR codes they created corresponding to the clues that they made. I also worked on the scanning component of our app which allowed users to access their camera and scan QR codes to gain points. I also helped with working on the "create lobby" component. For future work, if we do decide to continue developing this app, our main focus would be to refine it and add a few extra functionalities that we didn't have enough time to implement. One such functionality might be a report system that would allow for players to report other players for inappropriate messaging using our chat feature. Another functionality that would be crucial to add if we were to continue working on this app would be a "forgot your password" feature.

b. **Ethical Issues**

A huge ethical issue that our app faces is in regards to the backend with using Firebase as our database. If, for some reason, Firebase's servers were compromised, this would potentially risk our users' personal data. We did take this into account and have chosen to use Firebase's authentication system which properly hashes sensitive info such as passwords. However, some information such as emails and game information can still be compromised. Since privacy and security are always regarded as a top priority to uphold for customers, this potential security risk poses a huge ethical issue for our app. Another ethical issue that our app faces is in regards with our chatting functionality available in games. We have not implemented any features for moderating chats or reporting players which is unethical since bullying and inappropriate behavior might occur in chats.

c. **Life-long Learning**

During this semester, I have learned a lot of life-long lessons that will surely help me in the future. One of the most important things I learned while working on this app, was learning to effectively work with others in a team to achieve a common goal. This is a crucial skill for any software developer to have since developing large scaled software requires a lot of team work and coordination. Another skill that I learned during this semester was learning the appropriate way to document work and stay on track in terms of project management. This skill was learned through working on the SRS at the beginning of the semester as well as completing backlogs every two weeks to help my group

stay on schedule. In terms of programmatic and technical learning, I learned to use Firebase and successfully incorporate it with Swift.

d. Project Timeline

Week 1-2: Decided to use Firebase as the database we would be utilizing. Connected Firebase to our project and did some simple testing.

Week 3-4: Worked on signing up and signing in functionality. Also worked on email authentication functionality, as well as signing out functionality.

Week 5-6: Worked on “create game” and “join game” components for our app. Also worked on QR code generation and emailing QR codes to host.

Week 7-8: Worked on scanning functionality and fine tuned “create game” and “join game” components.

Week 9-10: Worked on messaging and chat functionality. Also added the ability for hosts to upload photos when creating clues for games.

7. Conclusion

Working on this Scavenger Hunt app proved to be very enjoyable as well as extremely beneficial personally. It has helped me build the necessary skills for software development and taught me how to effectively manage and work in a team. Our app has also fulfilled the promises we made in our SRS and has turned out to be quite fun to play, which was one of our main objectives. There were many hurdles and difficulties along the way, however, as a group, I would say that we did a good job in overcoming them. Overall, I believe that this app is something to be proud of and pleased with.

8. References

SRS_ScavengerHunt.docx

https://github.com/rbenreuben/Scavenger-Hunt