PeayPark Design

Jnaria Wheeler

Mitchell Sollmann

Connor Dial

# Revised Requirements

An updated requirements list based on my comments from the last requirements task.

1. User Login Page
   1. APSU Parking App logo
   2. Textbox prompting user to enter their email
   3. Textbox prompting user to enter their password
   4. Login button
   5. “Forgot Password” link
   6. “Create an Account” button
   7. APSU logo
2. Create an Account
   1. Textbox for Email
   2. Textbox for Password
   3. Textbox for reentering Password
   4. Next button to Verification Page
      1. Textbox to enter verification Code
      2. Next button to Create a Username page
         1. Textbox for Username
         2. Create an Account button
3. Map Page
   1. Must use Google Maps API for physical mapping of parking lots.
   2. Must utilize Google API keys to connect application session to Google Maps API
   3. Request permission to access user’s location.
   4. Must include Menu Bar for filtering which parking lots are shown on the map.
      1. Must include checkbox labeled “Select All”
      2. Must include checkbox labeled “Resident” for parking lots.
      3. Must include a checkbox labeled “Student” for parking lots.
      4. Must include a checkbox labeled “Staff” for parking lots.
      5. Must include a checkbox labeled “Commuter” for parking lot.
   5. Parking Features
      1. Must include Pop-up Label over each parking lot.
         1. Must include total parking spaces used at the current time in whole numbers.
         2. Must include total parking spaces available in the current selected parking lot in whole numbers.
         3. The total cars currently in parking lot and total spots available must be arranged in a label together separated by a single “/”.
         4. Must include the time range in which the parking lot is open and who it is open to. (Guest, Resident, Staff, Commuter.)
4. Administrator Page
   1. Must have label at top of page
   2. Will have a text box for Admin to enter the email address the account they wish to (modify, or delete)
   3. Below the text box will be three buttons named (confirm, modify, and delete)
      1. The confirm button will confirm the changes made to the account.
      2. The modify button will pull up a box that will let the administrator edit the user's information
         1. The user’s information will be in the same format as the sign-up feature.
      3. The delete button will simply delete the user's account.
5. Settings Page
   1. Must include setting to switch overlay to view maps in at least 2 different ways (Default and Aerial Overview)
   2. Must include quit button labeled “Quit” to end program.
   3. Must include button labeled “Refresh” that refreshes data to most current.
   4. Must include button labeled “Log Out” to log out of account if logged in.
   5. Must include button labeled “Login” to log into the account
   6. Must include button labeled “Allow GPS” that allows GPS location services permissions from the device.
   7. Must include button labeled “Report” that pulls up reporting page over current screen.
6. Notifications
   1. Application must be capable of sending notifications while the application is actively running.
      1. Notifications must inform users of the current availability of parking spaces within the parking lot(s). (e.g. 3 parking spaces left.
7. SupaBase
   1. Must utilize SupaBase as database.
      1. Must allow queries to be sent and received.
      2. Must store account data for each user that registers and successfully creates an account.
         1. Must store username and password as strings in Table View.
         2. The database schema table must be organized according to email address and password.
   2. Must integrate SupaBase with mobile Graphical User Interface.
   3. Must integrate SupaBase with desktop Graphical User Interface.
8. JavaScript
   1. Must utilize React Native Frameworks
   2. Must utilize Node.js Frameworks
   3. Must communicate and send queries to SupaBase.
      1. These queries must contain account information and parking lot information.

7.4. Parking Application must be able to run standalone on iOS and Android devices

1. Users
   1. Must differentiate between guests, students, and staff based on email.
      1. Must use regular expressions to read email strings of users to classify them as Guest, Staff, or Student.
      2. If the email address on the account ends with [@my.apsu.edu](https://my.apsu.edu/) then they should be placed into student classification.
      3. If the email address on the account ends with [@apsu.edu](https://my.apsu.edu/) then they should be placed into staff classification.
      4. If the email address on the account ends with a non-university email tag, then they should be placed into guest classification.
2. Reporting Page
   1. Must include button labeled “Take Picture for Report” to access device camera and take photo.
   2. Must include a textbox labeled “Add Details to Report” to allow user to add details underneath their photo.
   3. Must include submit button labeled “Submit Report” that is located on bottom of screen that sends a report to email for parking authority. (We will use dummy email for this for testing purposes).
3. Performance
   1. Application must run without stuttering effects on Android and iPhone platforms.
   2. The application must be responsive, meaning it must be able to handle a combination of touch inputs without freezing or lagging.
   3. Application must not crash.
4. Miscellaneous
   1. Application must have a back button labeled “back” on all pages except for the default page upon opening that takes the user back to the previous page.
   2. Application must handle at least 20 users simultaneously without major performance detriments.
   3. Application must send out less than 50,000 queries every 24 hours and no more than 1 query every 2 seconds.

# Design Description

**Peay Parking App: Architecture Description**

The APSU Parking App is a JavaScript application designed to operate on both Android and iOS mobile devices. It is built using the React Native framework. The purpose of this framework is to ensure a high performance and a native-like user experience across platforms. React Native accelerates development by enabling code reuse between Android and iOS, while maintaining the look and feel of a native application. The app is controlled through an intuitive graphical user interface (GUI) that provides users with various functionalities, from locating parking lots to managing administrative tasks.

**Backend Architecture and Data Management**

The application uses SupaBase as its backend service for data management and server-side processing. SupaBase handles user account information, including usernames and passwords, which are securely stored in hashed format within database tables. These tables are accessible only to designated administrator accounts, such as parking enforcement and/or law enforcement personnel. SupaBase also manages parking lot data, such as the number of available spots, lot classifications, and operational hours. The mobile app sends queries to SupaBase based on the user’s geographic location, updating the database in real-time. For example, when a user enters or exits a parking lot, the app sends a query to increment or decrement the total number of cars in the lot. The SupaBase server service receives these queries that are sent from the application and updates the appropriate table's row and fields for that specific parking lot. Any and all changes in the database are then reflected in the app’s GUI within a few seconds of the change taking place.

**User Interface and Pages**

The app consists of six pages or screens, each serving distinct functions:

1. **User Login Page**

The login page is the first screen that a users encounter when launching the application for the first time on their mobile device. The name of the application is displayed with the login credential boxes that are used to enter in the username and password that belongs to the specific user. This page contains a button labeled "sign-up" that takes the user to another page that allows the user to sign-up and create an account that is inserted into the SupaBase account table.

2. **Create an Account Page**

This page allows new users to register and create an account in order to use the application. This page includes text boxes for entering an email address and entering a user-chosen password two times. A button labeled "Submit" submits that information to SupaBase. This information is then added as a new account in the SupaBase's appropriate table within its database. If the user enters an email during the signup process that does not exist, meaning it does not have an “@” symbol or valid domain, then an alert box will be shown to the user when the “Submit” button is pressed that states “Please enter valid email to continue.” It includes a single button labeled “Okay” that when pressed, takes the user back to the account creation screen and allows them to attempt to make an account until a valid email is entered by the user.

3. **Map Page**

The map page is the app’s central feature, displaying an interactive map powered by the Google Maps API. Upon successful login and GPS access users are taken to this page.The map highlights parking lots on campus, with each lot represented as a clickable area. A Menu Bar at the top of the screen allows users to filter parking lots by classification (e.g. Guest, Student, Staff) using checkboxes. A “Select All” checkbox enables users to display all parking lots simultaneously. Pop-up labels over each parking lot provide essential information, including the total number of spaces used and available, operational hours, and lot classification. The data is presented in a clear format, such as “COMMUTER LOT 15/50 8AM-430PM”. This means that in the parking lot there are 15 spots used out of 50 total parking spots in this specific lot that is designated for students who commute to class daily between the hours of 8 AM and 4:30 PM.

4. **Administrator Page**

This page is accessible only to users with administrative privileges. It features a label at the top denoting it as the Administrative Page. Administrators can search for user accounts by entering an email address in a designated text box. Below the search field, there are two buttons with two different labels "Modify” and “Delete”. These buttons allow administrators to manage accounts. The “Modify” button opens a form where administrators can edit user information, such as email and password. When the “Delete” button is pressed, a box will show up in the center of the screen stating “Confirm?” with a button labeled “yes” and a button labeled “no”. If the administrator presses the “no” button, then the account is not deleted and the box closes. If the administrator presses the “yes” button, then the box is closed and the selected account is deleted and removed from the database. Administrators can manually adjust parking lot data as well. This includes the number of cars, lot size, location, operational hours, and classification. Other approved individuals such as parking enforcement or public safety can also add, remove, and/or modify parking lot details as needed. When an administrative user attempts to enter and submit an email address that does not exist within the SupaBase user table, an error message is displayed to the user stating that it does not exist.

5. **Settings Page**

The settings page provides users with a customization option for the map overlay to switch between Default and Aerial Overview. It includes a button labeled "Quit" that allows users to exit the app and return to their mobile device's home screen. It includes a button labeled "Refresh" that updates the app’s data to match the latest information in SupaBase. It includes a button labeled "Log Out" that logs users out of their accounts. It includes a button labeled "Login" that redirects users to the login page if they are not already logged in. A button labeled “Allow GPS” that grants the app permission to access the device’s location services. If the user does not allow GPS permissions, the application will create a pop-up when attempting to use the map screen stating "Please enable location services in order to use this application." It includes a button labeled "Report" that opens a reporting page, where users can take photos using their device’s camera (after getting camera permissions from user) and add details in a text box. Underneath this text box, there is a button labeled "Submit Report" that sends the report to the designated parking authority’s email address and returns the user to the settings page.

6. **Reporting Page**

This page allows users to report parking violations or other issues. It includes a button labeled “Take Picture for Report” that temporarily accesses the device’s camera to capture a photo. A text box labeled “Add Details to Report” enables users to provide additional information. A Submit Report button at the bottom of the screen sends the report to the parking authority’s email address and closes the page, returning the user to the settings page.

**User Classification and Permissions**

User accounts are classified based on the email address used during registration. The app employs regular expressions to parse the email domain and assign the appropriate classification:

• Student: Accounts with the domain @my.apsu.edu.

• Staff: Accounts with the domain @apsu.edu.

• Guest: Accounts with non-university email domains.

Guest accounts have limited access compared to student and staff accounts, ensuring that only authorized users can access certain parking lots or features.

**Integration with Google Maps API**

The application integrates the Google Maps API to provide an interactive map for users. The API is used to display parking lots on a two-dimensional map, calculate geographic locations, and overlay essential information such as lot availability and operational hours. The app also uses Google API keys to establish a secure connection between the application session and the Google Maps service. To optimize performance, the app is constrained to sending no more than 50,000 queries to the Google Maps server and SupaBase daily, with a limit of one query every two seconds.

**Notifications**

The app is capable of sending real-time notifications to users while the application is actively running. These notifications help to keep students and faculty informed about the current availability of parking spaces in their selected lots ( e.g “3 spaces remaining in the Technology Building Parking Lot”). If a user disables or does not enable notifications, then they will not be able to receive automatically generated updates about parking lot availabilities unless they are actively looking at GUI in application. The user is informed on their application once in center of screen when opening application via a dialog box that states “Enable notifications for better parking!”. This alert box only shows up once in a 24 hour period and can be closed by the user pressing the “Okay” button located on the dialog box.

**Performance**

The application is designed to run smoothly on both Android and iOS devices without stuttering, freezing, or crashing. It is highly responsive. The application supports at least 20 concurrent users without significant performance issues. The application includes a button labeled “Back" is included on all pages (except the login page) to allow users to return to the previous screen. Additionally, the app ensures that all queries to SupaBase and Google Maps are within the specified limits to maintain optimal performance.

Instructions for all appendices:

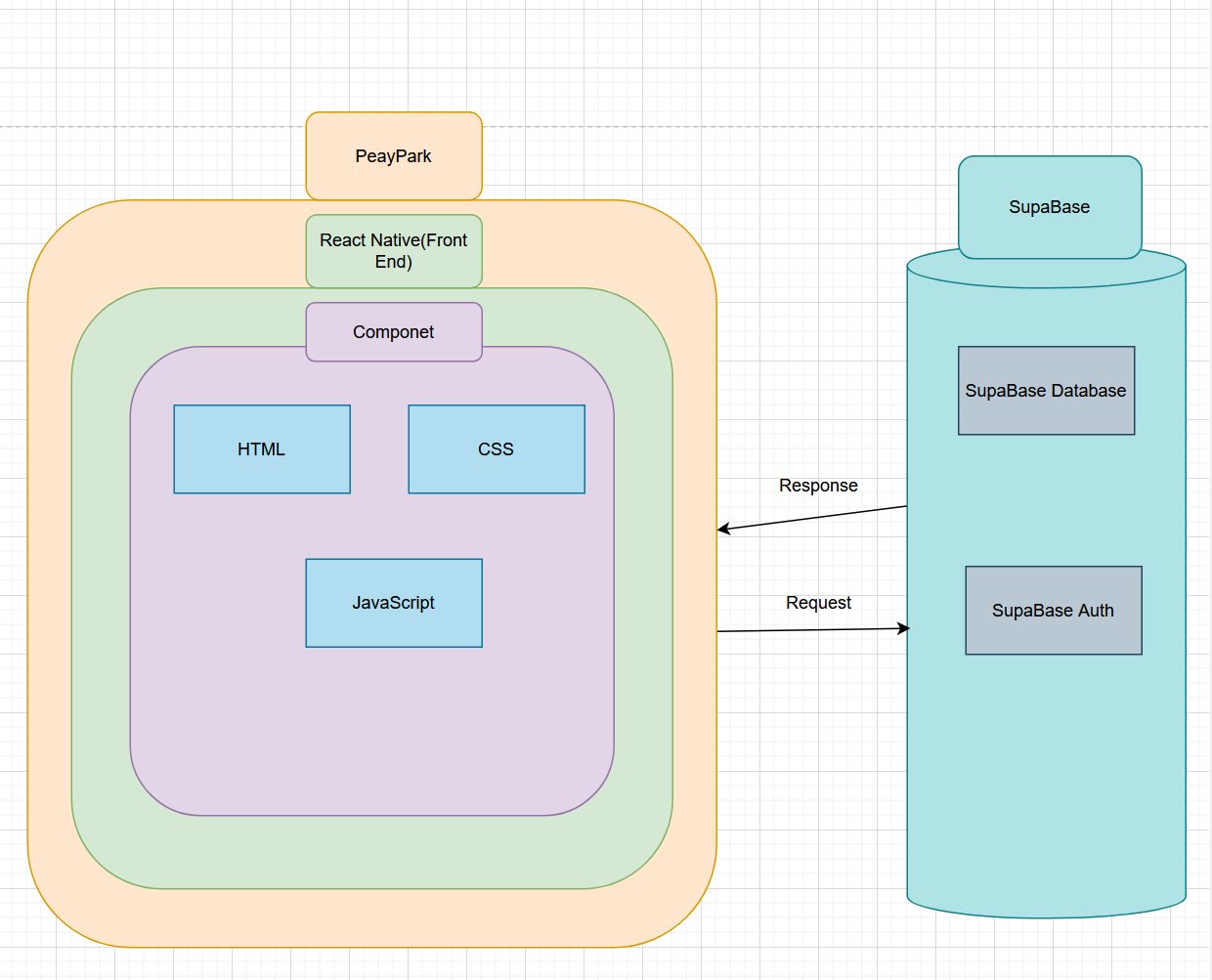
Add these items as appendices. If you make an image too large to be readable in your document, save the image as a separate file, and in your document, add a blurb that says something such as “See data.png for the class diagram.”

* All diagrams should look nice. Arrange components so that:

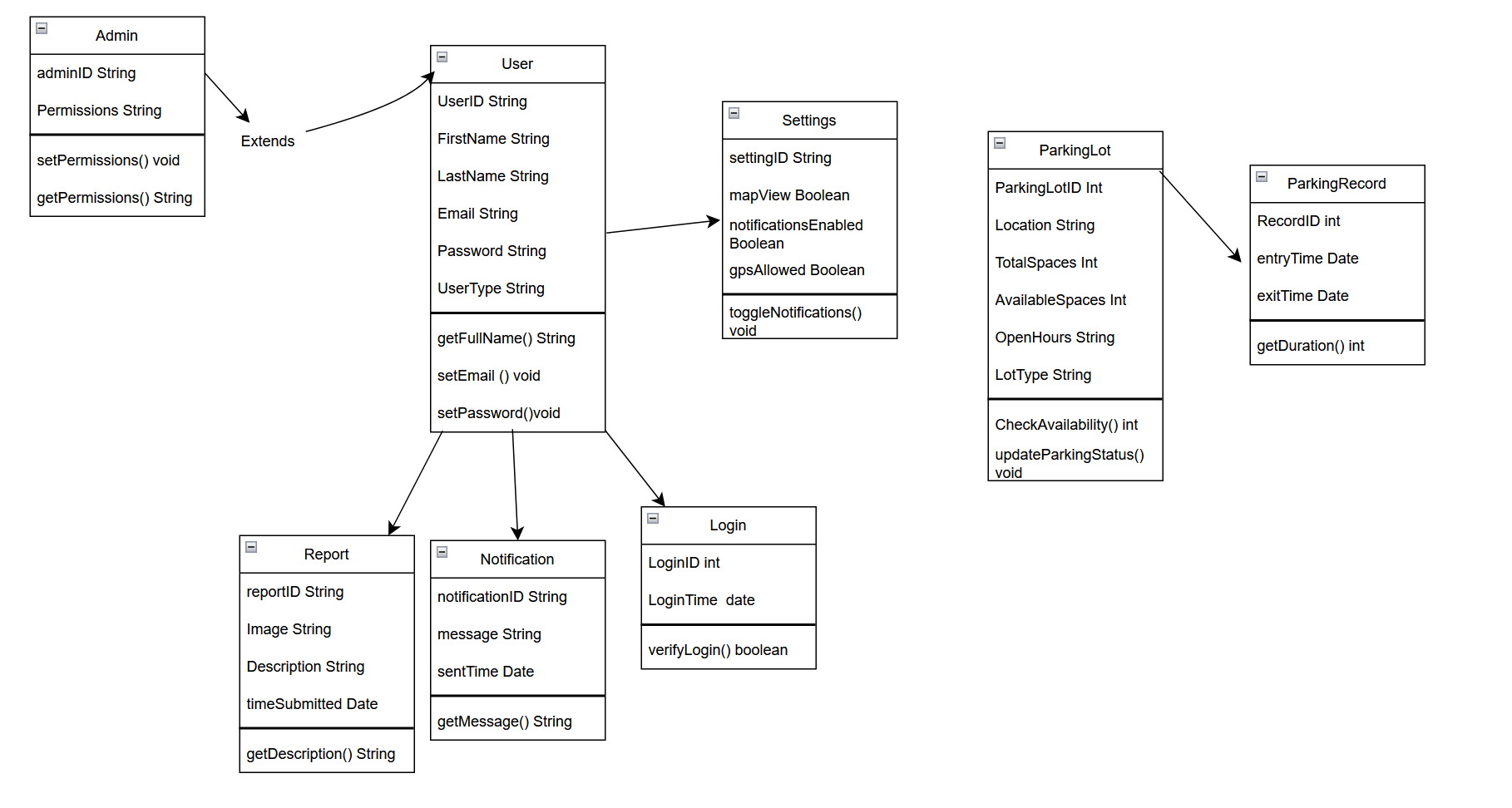
1. Objects do not overlap
2. There are a minimal number of lines that cross one another.
3. There are a minimal number of diagonal lines.

* Add textual notes to the diagrams to clarify details.
* All diagrams should have a title.
* Add an editable version of any diagram or image, not just a final PNG or PDF image, to your GitHub repository.

# APPENDIX – BLOCK DIAGRAM



# APPENDIX – COMPONENT DIAGRAM



# APPENDIX – USER INTERFACE STORYBOARD

Provide a user interface storyboard. This consists of two things.

* Simple sketches of every window and dialog box in the user interface. You do not need to draw every little detail, but each screen sketch should document the main layout of the major components on the page. Generally, these images should look simpler than the screens in the final application. Do not use hand-drawn images; use a computer-based tool to create the images.
* Show connections between pages and windows. For example, you might have a box labeled “Login Screen” with an arrow to a box labeled “Login Error.” The “Login Error” box would have an arrow connected to the “Login Screen” box.

# APPENDIX – MESSAGE DOCUMENTATION

A computer screen with text on it

AI-generated content may be incorrect.

The purpose is to authenticate a user when they attempt to log in.  
  
A screenshot of a computer program

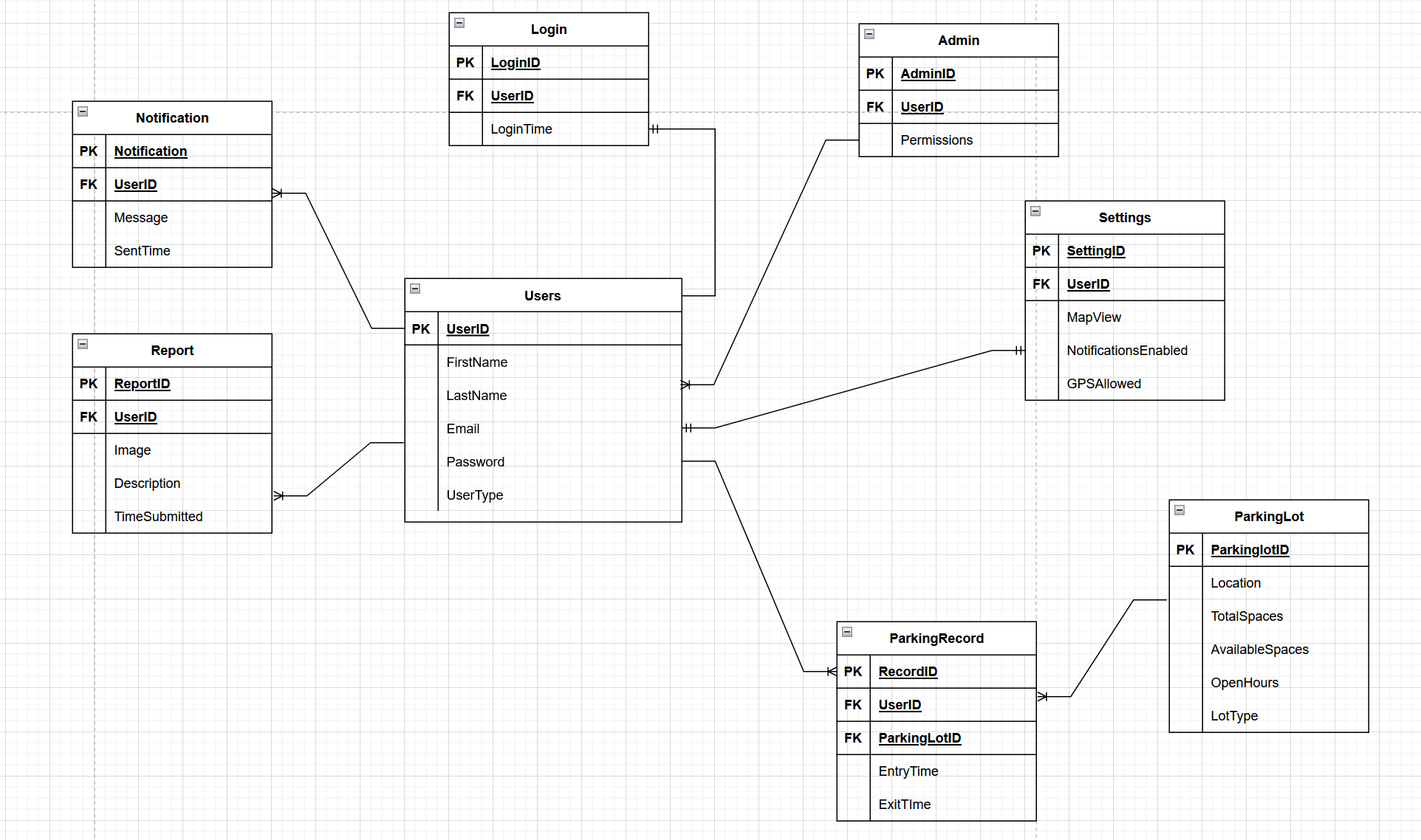
AI-generated content may be incorrect.

The purpose is to register a user when they attempt to register in app.   
  
A computer screen with text on it

AI-generated content may be incorrect.

The purpose is to find the right parking lot associated with the user’s location.

# APPENDIX – STORAGE DOCUMENTATION (required if storing data)



If you are storing data, you must describe how the data is being stored and the formats involved.

1. If using a relational database, e.g., MySQL, or an embedded database, e.g., SQLite, you must provide descriptions of every table – the table names, the column names, the data types, primary keys, and foreign keys. Code, such as triggers, stored procedures, etc., should also be documented. You should do this as an ERD diagram ( a diagram of the tables).
2. If using a NoSQL database, you must provide descriptions of every table and the data structure. You can document the database structure using an ERD diagram or describe it in JSON. Use the approach that best helps you to understand the data.
3. If storing data in files, you must describe each file format used and the data in the file.
   1. If it is a custom format, such as JSON, XML, or binary, Describe the data and how the file is formatted. If the data is text-based, you should include a sample file as an example.
   2. Describe what the image represents if you store binary data in a standard format such as PNG. You do not need to describe the exact format of the file.
4. If you are storing data some other way, describe what data is stored and its datatypes. If it is text-based, you should include an example.
5. In all cases, you must give an overview of how the data will be stored and retrieved.

**Submission**

For the presentation, prepare a set of PowerPoint slides giving an overview of the design. Plan for a 15-minute presentation and each member is expected to participate. You do not need to include the updated requirements in the presentation.

Upload the description as a PDF and your slides as PowerPoint files to D2L by the beginning of class on the due date.