Project name

PEAYPARK

Team member names

Jnaria Wheeler

Connor Dial

Mitchell Sollmann

Abstract

The Austin Peay State University (APSU) Parking lot application is designed to make parking easier for students and staff by providing real-time updates on a user-friendly interface. Users can view parking availability across campus, check detailed information about a single lot, see popular and low capacity lots, and get directions. The application also includes a reporting option for incorrectly parked vehicles, allowing users to submit a photo with the reason behind it. With these features implemented, the application goal is to make the process of parking on campus more manageable and less stressful for everyone while also reducing traffic and promoting responsible parking.

Tools & Technologies

* List of tools and technologies that will be used to help develop the project.

Integrated Development Environment(s)

* JetBrains WebStorm

Map Tools

* Google Maps API

Frontend

* JavaScript
* React Native

Backend

* Node.js

Requirements list

Dashboard – (could show parking maps and see if they are full or empty, possibly have maps refresh every 2 seconds to show real time updates)

1. User Login Page
   1. The login page will be one component, with the following:
      1. APSU Parking App logo
      2. Header text saying “Sign up” or “Sign in to your account”
      3. Text saying select login method.
   2. Login in method option
      1. Login method will separate label for the 3 three selections
         1. Student
         2. Staff
         3. Guest
      2. Will have separate text boxes for each login method.
         1. One option will be a button labeled “Guest”
         2. One option will be a button labeled “Staff”
         3. One option will be a button labeled “Student”
   3. Sign up option
      1. Selecting “Sign up” will lead the user to a registration form
         1. Header text saying “Register” -
         2. Tag under header text saying” Please fill out form”
         3. Label for first name, last name, email, password, reenter password and License Plate
         4. Password requirements
            1. 10- 16 characters with 2 special characters and 2 numbers
            2. Will be case sensitive
         5. Possible Terms and Privacy agreement?
            1. Will have a check box for users to agree to the terms and privacy agreement. (Users will not be able to register until confirmed?)
         6. Possible “I am not a robot captcha”?
         7. Button at the bottom of the screen saying “register” to confirm account.
         8. Link at the bottom of the screen saying, “Already have an account, please sign in here” Link will redirect user to login page.
   4. Username (Email)
      1. Users will enter valid email that was used to create account
      2. Every email may only be associated with one account.
   5. Password
      1. Users will enter their password that was used to create the account
      2. Password must be 16 characters long and have no special characters.
2. Map Page
   1. Must use Google Maps API for physical mapping of parking lots.
      1. Must utilize Google API keys to connect application session to Google Maps API
   2. If the user is already logged in, then Map page will display instead of Login Page.
   3. Must include Menu Bar for filtering which parking lots are shown on the map.
      1. Must include checkbox labeled “Resident” for parking lots.
      2. Must include a checkbox labeled “Student” for parking lots.
      3. Must include a checkbox labeled “Staff” for parking lots.
      4. Must include a checkbox labeled “Commuter” for parking lot.
   4. Parking Features
      1. Must include support for mapping of parallel parking spots.
         1. You must separate parallel parking sections into individual sections of 10 and then map them as custom parking lots.
      2. Must include support for mapping of temporary parking spots.
      3. 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. (Students, Resident, Staff, Commuter.
3. 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.
4. 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.
5. 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).
6. 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.
7. 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.
8. Expo
   1. Parking Application must be able to run in Expo Application on Android 15 or later.
   2. Parking Application must be able to run in Expo Application on iOS 17 or later.
9. 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.
10. 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).
11. 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.
12. 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.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | All | Jnaria | Connor | Mitchell |
| 1 | Ensure GitHub and IDE working appropriately for all members. |  | Develop GUI for mobile | Devrlop GUI with JavaScript |
| 2 | Developing application GUI | Develop GUI for web | Develop GUI with JavaScript for mobile. | Work on back-end set up for GUI with JavaScript in Node.js |
| 3 | Integrate Supa base with parking web application. |  | Test mobile GUI and GPS functionality on mobile. | Work on back-end set up for GUI |
| 4 | Integrate Supa base with parking web application. | Help where needed | Develop interactive parking map (Scroll and click parking lots) and Integrate Google Maps API | Work on back-end code for Google Maps API and functionality. |
| 5 | Testing and Continuous integration of Supa base with .js front-end. | Help where needed | Add parallel parking spots to application and classify as custom lots. | Work on back-end code and functionality. |
| 6 | Testing and Continuous integration of Supa base with .js front-end. |  | Add report feature that sends e-mail to APSU parking department. | Set up login page and connect it with SupaBase |
| 7 | Testing and Continuous integration of Supa base with .js front-end. |  | Add feature to report tow trucks.  Add feature to report gas in electric parking spots. | Test login page and debug any issues |
| 8 | Testing and Continuous integration of Supa base with .js front-end. |  | Add feature to allow addition and removal of parking lots. | Help where needed |
| 9 | Testing and Continuous integration of Supa base with .js front-end. |  | Add low availability notifications.  Add high availability notification. | Help where needed |
| 10 | Testing and Continuous integration of Supa base with .js front-end. | Help where needed | Add Favorite Lots – ability to favorite lots most relevant to a user | Work on any deficiencies in the code. |
| 11 | Bug Testing and Refactoring on all software |  | Polish GUI and map integration anomalies. | Extraneous bug fixing and refactoring |
| 12 | Bug Testing and Refactoring on all software |  | Extraneous Bug Fixing and refactoring. | Running real-world final testing |

* Do not assume anything about the system; write down all requirements.
* In general, how the GUI looks is not a requirement unless the system requires the feature to work.
* Not necessary: The OK button will be on the bottom right.
* Necessary: Every dialog box displaying an informational message will have an OK button that closes the dialog when clicked.

Updated Timeline

* Provide a weekly timeline that outlines the work that will be accomplished each week on the project from now to the final presentation.