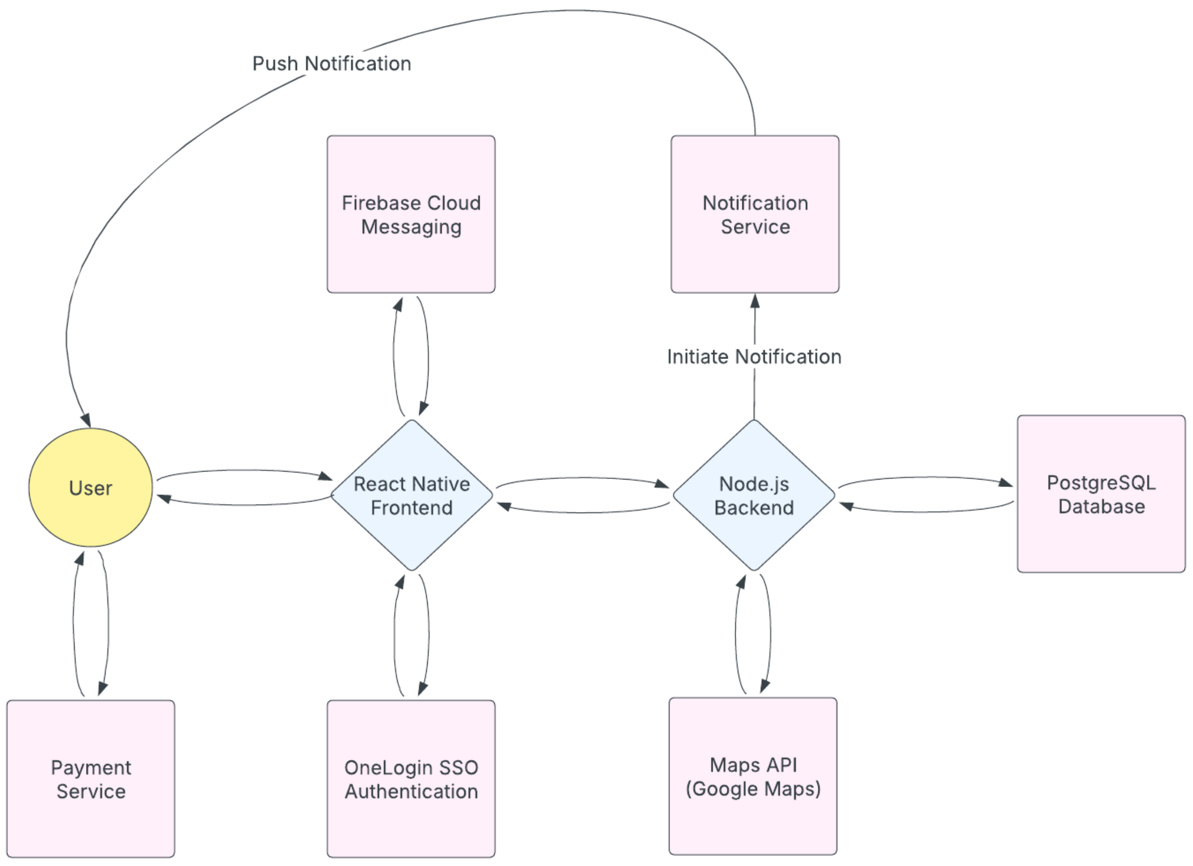
Sprint 0 Plan

1. **System Description**
    a. **Purpose:** Lynx Lifts is a mobile application that is designed to address transportation challenges at Rhodes College. The system's goal is to connect student drivers with student passengers. The platform will allow drivers to state their online/offline status, their coverage area and accept/decline rides. Passengers will be given a space where they can request rides on X day for X distance at X time. Lynx Lifts aims to provide safe, cheaper, convenient, and efficient transportation that addresses the ongoing concerns for students.
    b. **Basic functionality:**
        i. Students create an account using their @rhodes.edu outlook accounts (OneLogin authentication)
        ii. Students have access to two different interfaces: driver and passenger
        iii. Driver interface: a button indicating their status (online/offline), list their coverage area, view posts on the feed (accept or decline requests), and vehicle details
        iv. Passenger interface: post on the feed requesting rides, browse available drivers, select map location (origin to destination), payment calculation
        v. Students will receive a notification reminding them about the scheduled ride
        vi. There will be a report system for students to keep the app a safe environment
        vii. Payment for secure transactions between the driver and passenger(s)
        viii. In-app chat option to facilitate communication between the driver and passenger(s)
        ix. Optional: carpooling feature for passengers heading in the same direction
2. **Architectural Overview**
    a. The system follows a mobile application architecture that uses Node.js for the backend and PostgreSQL for the database, with React Native for front-end development. The backend handles data processing, account creations, and storing data for two different interfaces. The passenger account will store posts the student posted, map selection/payment calculation for a ride, and chat history. While the driver account will store viewed posts that the student accepted, the total payment amount received, coverage area, online/offline status, and chat history. The front end will be responsible for creating the driver and passenger interfaces and facilitating user interactions.
    b. **Major System Components:**
        i. React Native–JS (frontend): provides the UI for drivers and passengers, manages displaying ride data, and handles real-time updates (e.g. driver availability status)

ii. Node.js (backend): processes ride requests, manages user authentication, stores & retrieves user data, rides, payments, and chat logs
iii. Maps API: helps with geolocation for passenger input of location selection
iv. PostgreSQL (database): stores user profiles (e.g. driver/passenger status), ride data, payment transactions, etc.
v. OneLogin (or just using @rhodes.edu format) (authentication): Only Rhodes students can create an account and log in to Lynx Lifts
vi. Payment integration of different options (e.g. Paypal): allows passengers to pay drivers securely
vii. Firebase Cloud Messaging/WebSocket (chat system/notifications): allows real-time chat between drivers and passengers and sends push notifications to users

c. System Diagram



3. Functional Requirements

a. EPIC 1–Communication System

As a user of Lynx Lifts,

I want to be able to communicate with other users and view their bio,
So that I know who I'm coordinating rides with

- US1, US13, US7, US15

b. *EPIC 2–Chauffeur Features*

As a driver of Lynx Lifts,
I want to find students around me who need rides,
So that I can earn money

- US2, US9, US10, US12

c. *EPIC 3–Rider Features*

As a passenger of Lynx Lifts,
I want to be able to have the ease of finding drivers around me that match my needs,
So that I can get to my destination safely

- US3, US8, US11, US14, US16

d. *EPIC 4–In-App Environment*

As a user of Lynx Lifts,
I want to have the convenience of doing everything in-app (e.g. payment, notifications, etc.)
So that I don't have to deal with downloading/utilizing other applications for processing my ride needs

- US4, US5, US6

4. Non-Functional Requirements

a. *Non-Functional Requirement 1:* Security
   i. User authentication must be secure, exclusive to Rhodes students
b. *Non-Functional Requirement 2:* Performance
   i. The app must handle at least 2000 simultaneous users without performance decline
c. *Non-Functional Requirement 3:* Usability
   i. The interface must be intuitive and user-friendly; users can navigate the application without assistance
d. *Non-Functional Requirement 4:* Scalability
   i. The backend should support future expansion by Rhodes College

5. Technologies and Frameworks

a. Frontend development: React Native–JS
   i. Supplement with Figma for UI/UX design
b. Backend: Node.js
c. Maps API
d. Database: PostgreSQL
e. Authentication: OneLogin (or just using @rhodes.edu format)
f. Payment integration of different options (e.g. Paypal, Venmo)

g. Chat system/Notifications: Firebase Cloud Messaging/WebSocket

6. **Define your Minimum Viable Product (MVP)**
    a. Students can create an account using their @rhodes.edu outlook accounts–US1
    b. There is a feed interface for both the driver and passenger–US3 & US2
    c. Passengers can select map location, calculate payment, and post on feed–US14, US8, US3
    d. Drivers can view posts on the feed, receive payment, online/offline status, coverage area, and accept ride requests–US2, US9, US10, US12
    e. There will be a chat system to coordinate communication between the driver and passenger(s) and a report system to maintain the in-app experience–US7, US5
7. **Roadmap**
    a. *Sprint 1:* Focus on developing the accounts for the driver and passenger interfaces and begin working on the passenger interface
        i. US1: Sign Up
        ii. US3: Post on Feed
        iii. US14: Select Map Location
    b. *Sprint 2:* After having a bare-bones for passenger interface, focus on adding additional features for the passenger interface and begin on the driver interface
        i. US8: Payment Calculation
        ii. US9: Online/Offline Status
        iii. US2: View Feed
    c. *Sprint 3:* The foundation of the application is implemented, expand on the driver interface
        i. US12: Accept Rides
        ii. US10: Specify Coverage Area
    d. *Sprint 4:* Begin working on the additional features that apply to both the driver and passenger interfaces
        i. US7: Chat System
        ii. US4: Payment Options
        iii. US6: Notifications
    e. *Sprint 5:* More features expansion to the application
        i. US5: Report System
        ii. US11: Browse Available Drivers
        iii. US13: Bio
    f. *Sprint 6:* Iron out any bugs in the application, and work on the optional feature(s)
        i. US15: Carpool
        ii. US16: Chauffeur Rides