### 1. Team info & policies

* **Team Members and Roles**:
  + List the names of your team members and their roles:
    - Alice - Backend Engineer (Node.js, API)
    - Haytham - Backend Engineer (Node.js, MySQL, )
    - Yuzhou - Frontend Engineer (React Native)
    - Wells - Frontend Engineer (React Native)
* **Project Artifacts**:
  + GitHub repository: https://github.com/aliceduongg/Stuber
  + Communication channels: Google chat, GitHub Issues for tracking, In-person meeting
  + Development tools: VS Code, Postman (for API testing)

### 2. Product description

Mobile application connecting student drivers with community people needing rides.

* **4 major features:**
  + Authentication: login, logout, signup
  + Booking system
  + Driver and Rider Profiles
  + Notification System
* **2 stretch goals:**
  + Real time GPS access:Allow riders to see driver locations on campus.
  + Rating system: Enable both drivers and riders to rate each other for better accountability.

### 3. Use Cases (Functional Requirements)

**Actors:**

* **Primary Actor:** User (Passenger)
* **Supporting Actors:**
* Stuber App (Ride-hailing Platform)
* Driver
* GPS/Map System

**Trigger:**

* The user needs transportation and opens the Stuber web app to find a ride.

**Preconditions:**

* The user has logged in.
* The user has enabled GPS.
* Drivers are available.

**Postconditions (Success Scenario):**

* The booking is processed.
* The user receives a success receipt and has the option to rate the driver.
* The user is successfully picked up and dropped off at the destination.

**List of Steps (Success Scenario):**

* **User opens the Stuber web app.**
* The app displays the home screen with a map showing the user’s current location.
* **User enters the destination.**
* The app posts a request to find a driver.
* **Driver confirmed the ride.**
* The app displays the driver’s details (name, car, license plate) and a booking receipt.
* **Driver arrives and picks up the user.**
* The app tracks the ride’s progress by GPS.
* **User is dropped off at the destination.**
* The driver and user confirmed the completion of the ride.
* **User rate the driver.**
* The user is prompted to rate the ride and provide feedback.

**Extensions/Variations of the Success**

**Scenario:**

* **Variation 1: Multiple Stops**

The user can add multiple stops to the ride.

* **Variation 2: Ride Scheduling**

The user schedules a ride for a particular time. The app confirms the booking and assigns a driver when the time approaches.

* **Variation 3: Help**

The user is facing some problems and need some help by clicking on the

help button.

**Exceptions: Failure Conditions and**

**Scenarios:**

**No Drivers Available:**

* User confirms the ride request.
* The app fails to find any available drivers.
* The system displays a message that no drivers are available and suggests trying again later or choosing a different ride type.

**Driver Cancels the Ride:**

* After the ride is confirmed, the driver cancels.
* The app notifies the user of the cancellation and searches for another driver.
* If no other drivers are available, the system cancels the request and notifies the user.

**Connection failure:**

* User attempts to book a ride, but the connection is poor.
* The system fails to process the request.
* The app shows a network error and prompts the user to retry.

**Driver Cannot Find the User:**

* Driver arrives at the pick-up location but cannot locate the user.
* The system notifies the user of the driver’s arrival and facilitates communication between the driver and user.
* If the issue is unresolved, the driver may cancel the ride.

### 4. Non-functional Requirements

**Scalability:**

* Support small amount of concurrent users without performance degradation
* Allow for easy infrastructure scaling as user base grows

**Security and Privacy:**

* Encrypt all user data.
* Implement strong authentication, including passwords hashing
* Comply with relevant data protection regulations

**Usability and Accessibility:**

* Design an intuitive interface.
* Support multiple languages.
* Ensure consistent design across iOS and Android platforms.

### 5. External Requirements

Robust Error Handling: The app will handle invalid user input by providing clear error messages (e.g., invalid login credentials).

Public Application: The app will be deployed on the App Store for easy access by customers.

### 6. Team process description

**Software toolset used:**

* + Postman: used for testing API calls
  + Visual Studio Code: IDE used for collaborative coding, easy to use with github
  + MySQL: Database used to store relevant information about users.
  + Version Control: GitHub for collaboration and version tracking.
  + React Native: Allows building native mobile apps for both iOS and Android platforms using a single codebase.

**Schedule and Milestones**

Backend Team (Alice & Haytham)

1. \*\*October 3, 2024\*\*: Complete API design and database schema

2. \*\*October 15, 2024\*\*: Implement core backend functionality (auth, booking)

3. \*\*November 1, 2024\*\*: Develop advanced features (profiles, notifications)

4. \*\*November 20, 2024\*\*: Finalize API, optimize performance, complete testing

Frontend Team (Yuzhou & Wells)

1. \*\*October 5, 2024\*\*: Finish UI/UX design and component structure

2. \*\*October 20, 2024\*\*: Implement main screens (auth, booking, profiles)

3. \*\*November 5, 2024\*\*: Develop additional features (notifications, scheduling)

4. \*\*November 25, 2024\*\*: Complete integration with backend, final testing

**Major Risks**

1. \*\*Integration Challenges\*\*: Difficulties in aligning backend and frontend development could cause delays.

- Mitigation: Regular cross-team meetings, clear API documentation.

2. \*\*Performance Issues\*\*: High user load might affect app responsiveness.

- Mitigation: Early performance testing, optimizing database queries.

3. \*\*Security Vulnerabilities\*\*: Potential data breaches could compromise user trust.

- Mitigation: Regular security audits, implementing robust authentication.

**External Feedback Process**

1. \*\*Mid-October\*\*: Share UI prototypes with potential users for usability feedback.

2. \*\*Early November\*\*: Release beta version to select users for functional testing.

3. Maintain ongoing communication with beta testers throughout development.