Ride Share Application

Executive Summary

The project aims to develop a ride-hailing platform that connects passengers with drivers through a user-friendly mobile application. The platform will cater to three main user roles: passengers, who request rides; drivers, who provide transportation services; and administrators, who manage the platform's operations. Key functionalities include ride requests, real-time tracking, automated payments, and driver onboarding. The expected outcome is to create a seamless experience for users while ensuring efficient management and support for the platform's operations.

Core Functionalities

- **Ride Request System:** Allows passengers to request rides by setting pickup and drop-off locations through the app. (Priority: **High**)
- **Driver Matching Algorithm:** Matches passengers with available drivers based on proximity and availability. (Priority: **High**)
- **Real-Time Tracking:** Enables passengers to track their driver's location and estimated arrival time in real-time. (Priority: **High**)
- **Payment Processing:** Facilitates automatic payment processing after ride completion, including receipt generation. (Priority: **High**)
- **Admin Dashboard:** Provides administrators with tools to monitor platform activity, manage drivers, and handle customer support issues. (Priority: **Medium**)

Tech Stack

- **Frontend:** Next.js, ShadCN UI
- **Backend:** Convex, Node.js
- **Database:** Prisma
- **Database Management:** PostgreSQL
- **Real-time communication:** Socket.IO
- **Styling:** Tailwind CSS
- **User Authentication:** Clerk

Project Timeline

Tasks are categorized by complexity to guide time estimations: straightforward (S), moderate (M), challenging (C).

Roles:

- **Frontend Developer** (FD)
- **UI/UX Designer** (UXD)

- **Backend Developer** (BD)
- **DevOps Engineer** (DE)
- **Database Administrator** (DBA)
- **Quality Assurance Tester** (QA)
- **Technical Writer** (TW)
- **QA Tester** (QA)
- **QA Engineer** (QA)
- **Product Manager** (PM)
- **Data Scientist** (DS)
- **Mobile Developer** (MD)

Milestone 1: User Authentication and Landing Page Setup

- **Passenger Account Creation:** *As a passenger, I want to create an account so that I can access the app's features and request rides.* - The account creation form must include fields for email, password, and phone number. - The system must send a verification email or SMS to confirm the account. - The passenger must be able to log in after verifying their account.
    - Design and implement the account creation form, including fields for email, password, and phone number. Ensure proper validation for each field. - (M)[FD][UXD]
    - Implement the functionality to send a verification email or SMS to the user after they submit the account creation form. This should include a unique verification link or code. - (M)[BD][DE]
    - Create the endpoint to verify the user's account using the link or code sent via email/SMS. Update the user's status in the database upon successful verification. - (M)[BD][DBA]
    - Implement the functionality that allows the user to log in after their account has been verified. This should include handling login requests and session management. - (M)[BD][FD]
- **Passenger Login:** *As a passenger, I want to log in to my account so that I can access my ride history and request new rides.* - The login form must include fields for email and password. - The system must validate the credentials and provide appropriate error messages for incorrect logins. - The passenger must be redirected to the home screen after a successful login.
    - Design and implement the login form that includes fields for email and password. Ensure the form is user-friendly and accessible. - (S)[UXD][FD]
    - Implement backend logic to validate user credentials against the database. Provide appropriate error messages for incorrect logins. - (M)[BD][DBA]
    - Implement the logic to redirect the passenger to the home screen after a successful login. Ensure session management is handled correctly. - (M)[BD][FD]
    - Create error handling mechanisms to display appropriate messages for incorrect logins and other potential issues during the login process. - (M)[FD][BD]

    - Conduct thorough testing of the login functionality, including unit tests, integration tests, and user acceptance testing to ensure all acceptance criteria are met. - (C)[QA][BD][FD]

- **Passenger Password Reset:** *As a passenger, I want to reset my password if I forget it so that I can regain access to my account.* - The login screen must have a 'Forgot Password?' link. - The system must send a password reset link to the registered email. - The passenger must be able to set a new password after following the reset link.
    - Implement the 'Forgot Password?' link on the login screen that directs users to the password reset page. - (S)[FD]
    - Create a backend service that sends a password reset link to the registered email address of the passenger. - (M)[BD][DE]
    - Implement the logic to validate the password reset link when the user clicks on it from their email. - (M)[BD]
    - Create a form for the passenger to set a new password after validating the reset link. - (S)[FD]
    - Implement the backend logic to update the passenger's password in the database after they submit the new password. - (M)[BD]
    - Send a confirmation message to the passenger indicating that their password has been successfully reset. - (S)[BD]
    - Conduct testing to ensure the password reset functionality works as intended, including link validation, email sending, and password updating. - (M)[QA]
    - Create documentation for the password reset process, including user instructions and technical details for developers. - (S)[TW]

- **Passenger Logout:** *As a passenger, I want to log out of my account so that I can ensure my account is secure when I am not using the app.* - The app must have a visible 'Log Out' option in the user menu. - The system must confirm the log-out action and redirect the user to the login screen. - The passenger must not be able to access their account without logging in again.
    - Design and implement the 'Log Out' option in the user menu, ensuring it is visible and accessible to passengers. This includes updating the frontend UI to include the logout button. - (S)[UI/UX][FD]
    - Implement a confirmation dialog that appears when the passenger selects the 'Log Out' option. This dialog should ask the user to confirm their action before logging them out. - (S)[FD][UI/UX]
    - Develop the backend functionality to handle the logout process. This includes invalidating the user's session and ensuring that they cannot access their account without logging in again. - (M)[BD][QA]
    - Implement the logic to redirect the user to the login screen after they have successfully logged out. This should be seamless and user-friendly. - (S)[FD][BD]

    - Conduct thorough testing of the logout functionality, including UI testing for the logout button, confirmation dialog, and backend testing to ensure session invalidation works correctly. - (M)[QA][BD]

- **Clear Introduction to Service:** *As a passenger, I want to see a clear and engaging introduction to the ride-hailing service so that I understand the benefits of using the app.* - The landing page includes a brief description of the service. - Visual elements (images or icons) highlight key features (e.g., convenience, safety). - The design is responsive and visually appealing on both mobile and tablet devices.
    - Design the landing page layout that includes a brief description of the ride-hailing service, ensuring it is visually appealing and responsive for both mobile and tablet devices. - (M)[UI/UXD][FD]
    - Create visual elements (images or icons) that highlight key features of the service, such as convenience and safety, and integrate them into the landing page design. - (M)[UI/UXD][FD]
    - Implement the responsive design for the landing page, ensuring that it adapts seamlessly to different screen sizes and devices. - (M)[FD][QA]
    - Conduct usability testing on the landing page to gather feedback on the introduction and visual elements, making necessary adjustments based on user input. - (M)[QA][UI/UXD]
    - Finalize the landing page content and visuals, ensuring all elements are aligned with the branding and messaging of the ride-hailing service. - (S)[UI/UXD][FD]
- **Easy Login and Signup Options:** *As a passenger, I want to have the option to log in or sign up easily so that I can access my account or create a new one without hassle.* - The landing page displays prominent buttons for 'Log In' and 'Sign Up.' - Clicking on these buttons leads to the respective login or sign-up forms. - The forms are user-friendly and include necessary fields (e.g., email, password).
    - Design the landing page to include prominent 'Log In' and 'Sign Up' buttons that are visually appealing and easy to locate. - (S)[UXD][FD]
    - Implement the 'Log In' form with fields for email and password, ensuring proper validation and error handling. - (M)[FD][BD]
    - Implement the 'Sign Up' form with necessary fields (e.g., email, password, confirm password) and validation rules. - (M)[FD][BD]
    - Create backend endpoints for user authentication (login and signup) that handle requests and responses appropriately. - (M)[BD][DBA]
    - Integrate the frontend forms with the backend endpoints to ensure data is sent and received correctly during login and signup processes. - (M)[FD][BD]
    - Test the login and signup functionalities thoroughly, including edge cases and error handling, to ensure a smooth user experience. - (M)[QA][FD]
- **Display Promotions and Discounts:** *As a passenger, I want to see any promotional offers or discounts available so that I can take advantage of them when using the service.* - The landing page

includes a section for current promotions or discounts. - Promotions are clearly stated with expiration dates if applicable. - There is a call-to-action button to learn more about the promotions.

  - Design the promotions section on the landing page to display current promotions and discounts available to passengers. Ensure that the design is user-friendly and visually appealing. - (M)[UXD][FD]
  - Implement the backend logic to fetch current promotions from the database and serve them to the frontend. This includes creating an API endpoint that retrieves promotions and their details. - (M)[BD]
  - Integrate the promotions API with the frontend landing page to display the fetched promotions. Ensure that the promotions are displayed with clear expiration dates and a call-to-action button. - (M)[FD][QA]
  - Test the promotions feature to ensure that promotions are displayed correctly, including expiration dates and functionality of the call-to-action button. Conduct user testing to gather feedback. - (S)[QA][UXD]
  - Deploy the promotions feature to the production environment and monitor for any issues or feedback from users. Ensure that the promotions are updated regularly based on the backend data. - (S)[DE][BD]

- **Access to Customer Support Information:** *As a passenger, I want to have access to customer support information so that I can easily find help if needed.* - The landing page includes a visible link or button for customer support. - Clicking the link leads to a support page with FAQs and contact options. - Support information is easy to read and understand.

  - Design the customer support section on the landing page, ensuring it includes a visible link or button for customer support access. - (M)[UXD][FD]
  - Implement the functionality for the customer support link/button on the landing page to navigate to the support page. - (S)[FD]
  - Create the support page that includes FAQs and contact options, ensuring the information is easy to read and understand. - (M)[UXD][FD][BD]
  - Integrate the backend support functionality to fetch FAQs and contact options for the support page. - (M)[BD][QA]
  - Test the customer support functionality to ensure all links work correctly and the information is displayed as intended. - (S)[QA]

- **User Testimonials and Ratings:** *As a passenger, I want to see user testimonials or ratings so that I can trust the service based on others' experiences.* - The landing page features a section for user testimonials or ratings. - Testimonials are from verified users and include their first names or initials. - The testimonials are updated regularly to reflect current user experiences.

  - Design the layout for the user testimonials section on the landing page, ensuring it is visually appealing and easy to read. Include space for user names or initials and ratings. - (M)[UI/UXD][FD]
  - Implement the backend service to fetch user testimonials from the database. This service should return testimonials along with user initials and ratings. - (M)[BD][DBA]

- Create a database schema for storing user testimonials, including fields for user ID, testimonial text, and user initials. Ensure it supports efficient querying. - (S)[DBA]
- Develop the frontend component to display the testimonials on the landing page. This component should dynamically render testimonials fetched from the backend service. - (M)[FD]
- Set up a cron job or scheduled task to regularly update the testimonials from the database to ensure they reflect current user experiences. - (M)[BD][DevOps]
- Conduct QA testing to ensure that the testimonials section displays correctly on various devices and browsers, and that the backend service returns the correct data. - (M)[QA]

Milestone 2: Passenger Ride Request and Tracking

- **Set Pickup Location:** *As a passenger, I want to set my pickup location so that I can specify where I want to be picked up.* - The app allows me to enter or select my current location. - The app displays a map with my selected pickup location marked. - I can confirm my pickup location before requesting the ride.
  - Design and implement the user interface for setting the pickup location, including a map view and input fields for manual entry. - (M)[UXD][FD]
  - Develop the logic to handle the selection of the pickup location, including geolocation services to fetch the current location and update the map accordingly. - (M)[FD][BD]
  - Implement the confirmation process for the selected pickup location, ensuring that the user can review and confirm their choice before proceeding to request a ride. - (S)[FD][QA]
  - Create a backend service to handle the storage and retrieval of pickup location data, ensuring it integrates with the ride request process. - (M)[BD][DBA]
  - Conduct thorough testing of the pickup location feature, including unit tests, integration tests, and user acceptance testing to ensure functionality meets acceptance criteria. - (M)[QA][FD]
- **Set Drop-off Location:** *As a passenger, I want to set my drop-off location so that I can indicate where I want to go.* - The app allows me to enter or select my desired destination. - The app displays a map with my selected drop-off location marked. - I can confirm my drop-off location before requesting the ride.
  - Design the UI for setting the drop-off location, including input fields and map integration. - (M)[UXD][FD]
  - Implement the functionality to allow passengers to enter or select their drop-off location on the map. - (M)[FD][BD]
  - Integrate the map service to display the selected drop-off location and allow for adjustments. - (C)[FD][BD]
  - Create backend API endpoints to handle the drop-off location data submission and retrieval. - (M)[BD][DBA]
  - Implement confirmation logic to ensure the passenger can review and confirm their drop-off location before requesting the ride. - (M)[FD][BD]

    - Conduct QA testing to ensure the drop-off location feature works as intended and meets acceptance criteria. - (S)[QA]

- **Request Ride:** *As a passenger, I want to request a ride with a single button click so that I can quickly initiate the ride-hailing process.* - There is a clearly labeled 'Request Ride' button on the screen. - The app confirms my ride request with a notification. - The app transitions to a screen showing the status of my ride request.
    - Design the 'Request Ride' button UI component, ensuring it is clearly labeled and accessible on the main screen. - (S)[UXD][FD]
    - Implement the functionality for the 'Request Ride' button to trigger a ride request process when clicked. - (M)[FD][BD]
    - Create a notification system to confirm the ride request to the passenger after the button is clicked. - (M)[BD][FD]
    - Develop the transition logic to navigate the user to a screen that shows the status of their ride request after it has been confirmed. - (S)[FD]

- **Receive Ride Request Confirmation:** *As a passenger, I want to receive a confirmation of my ride request so that I know my request has been successfully submitted.* - The app displays a confirmation message after I request a ride. - I receive a notification with details of the driver assigned to my ride. - The app provides an estimated time of arrival for the driver.
    - Design the confirmation message UI that will be displayed to the passenger after they request a ride. This includes the layout, text, and any icons or images needed. - (S)[UXD][FD]
    - Implement the backend logic to handle ride request confirmations. This includes creating an endpoint that sends a confirmation message to the passenger after a ride request is made. - (M)[BD][DBA]
    - Set up a notification system that sends push notifications to the passenger's device with the details of the assigned driver and estimated time of arrival (ETA). - (M)[BD][DE]
    - Create a database schema to store ride request confirmations, including fields for passenger ID, driver ID, confirmation status, and ETA. - (S)[DBA][BD]
    - Test the entire confirmation process, including UI display, backend logic, and notification delivery to ensure everything works as expected. - (M)[QA][FD][BD]

- **View Estimated Fare:** *As a passenger, I want to view the estimated fare for my ride so that I can decide if I want to proceed with the request.* - The app calculates and displays an estimated fare based on the pickup and drop-off locations. - The fare estimate updates if I change either the pickup or drop-off location. - The fare estimate includes any applicable fees or surcharges.
    - Implement the logic to calculate the estimated fare based on the pickup and drop-off locations provided by the passenger. This will involve creating a function that takes the coordinates of both locations and applies the fare calculation algorithm, including any applicable fees or surcharges. - (M)[BD]
    - Create the frontend component that displays the estimated fare to the passenger. This component should be responsive and update in real-time as the user changes the pickup or

drop-off locations. It should also handle displaying any applicable fees or surcharges clearly. - (M)[FD][UXD]
    - Implement the functionality to listen for changes in the pickup and drop-off locations and trigger the fare calculation logic accordingly. This will involve setting up event listeners in the frontend that call the backend fare calculation function whenever the locations are updated. - (M)[FD][BD]
    - Write unit tests for the fare calculation logic to ensure accuracy and reliability. This will involve creating test cases that cover various scenarios, including different distances, fees, and surcharges. - (S)[QA]
    - Conduct user acceptance testing (UAT) to validate that the estimated fare feature meets the acceptance criteria outlined in the user story. Gather feedback from users and make necessary adjustments based on their input. - (M)[QA][PM]

- **Cancel Ride Request:** *As a passenger, I want to be able to cancel my ride request if I change my mind so that I have control over my ride requests.* - The app provides a 'Cancel Ride' option after I request a ride. - I receive a confirmation prompt before the ride is canceled. - The app updates my status to reflect the cancellation.
    - Design and implement the 'Cancel Ride' button in the passenger interface, ensuring it is visible after a ride request is made. This includes creating a confirmation prompt that appears when the button is clicked, asking the user to confirm the cancellation. - (M)[UXD][FD]
    - Develop the backend logic to handle the cancellation of ride requests. This includes updating the ride status in the database and notifying the driver of the cancellation. Ensure that the cancellation is logged for future reference. - (M)[BD][DBA]
    - Implement a notification system to inform the passenger that their ride request has been successfully canceled. This should include a confirmation message displayed in the app and an optional email notification. - (S)[BD][FD]
    - Conduct testing to ensure that the cancellation feature works as intended. This includes unit tests for the backend logic and integration tests for the UI components. - (M)[QAT][BD]
    - Update the project documentation to include details about the cancellation feature, including how it works, any limitations, and how to use it from the passenger's perspective. - (S)[TW][PM]

- **Real-Time Driver Location:** *As a passenger, I want to see the driver's real-time location on the map so that I can know when to expect their arrival.* - The driver's location updates in real-time on the map. - The estimated time of arrival (ETA) is displayed alongside the driver's location. - The map is zoomed in to show the pickup location and the driver's route.
    - Implement the real-time location tracking feature for the driver on the passenger's app. This includes integrating GPS data to update the driver's position on the map every few seconds. - (M)[FD][BD]
    - Develop the logic to calculate and display the estimated time of arrival (ETA) based on the driver's current location and traffic conditions. This will involve fetching real-time traffic data and calculating the ETA accordingly. - (C)[BD][DS]

    - Create a user interface component that displays the driver's real-time location on a map, including zoom functionality to focus on the pickup location and the driver's route. - (M)[FD][UXD]
    - Implement the backend service that handles the real-time updates of the driver's location and ETA, ensuring that the data is sent to the passenger's app efficiently. - (C)[BD][DE]
    - Conduct thorough testing of the real-time driver tracking feature, including unit tests, integration tests, and user acceptance testing to ensure the feature meets the acceptance criteria. - (M)[QA][FD][BD]

- **Driver Arrival Notifications:** *As a passenger, I want to receive notifications when the driver is approaching so that I can be ready for pickup.* - A notification is sent when the driver is 5 minutes away. - The notification includes the driver's name and vehicle details. - The notification can be dismissed or viewed in the app.
    - Set up the notification system to send alerts to passengers when the driver is approaching. This includes integrating with the push notification service and defining the notification payload structure. - (M)[BD][DE]
    - Implement the logic to calculate the distance between the driver and the passenger's pickup location. This will involve using the mapping API to determine the driver's current location and the estimated time of arrival (ETA). - (M)[BD][FD]
    - Create the logic to trigger notifications when the driver is 5 minutes away from the pickup location. This will involve setting up a timer or a polling mechanism to check the driver's distance at regular intervals. - (M)[BD]
    - Design the content of the notification to include the driver's name, vehicle details, and a dismiss option. This will involve collaboration with the UI/UX team to ensure the notification is user-friendly. - (S)[UX][FD]
    - Implement the functionality to allow passengers to dismiss notifications. This will involve updating the notification service to handle user interactions. - (S)[FD][BD]
    - Conduct testing to ensure that notifications are sent correctly and that the content is displayed as intended. This includes unit tests and user acceptance testing (UAT). - (M)[QA][BD]

- **Driver Profile and Vehicle Info:** *As a passenger, I want to view the driver's profile and vehicle information so that I can identify them easily.* - The driver's name, photo, and vehicle details (make, model, color, license plate) are displayed. - The driver's rating is visible to help me assess their reliability. - The information is accessible from the tracking screen.
    - Design the UI component for displaying the driver's profile, including their name, photo, and vehicle details (make, model, color, license plate). Ensure it is visually appealing and aligns with the overall app design. - (M)[UI/UXD][FD]
    - Implement the backend API endpoint to fetch the driver's profile and vehicle information based on the ride request. This should include the driver's name, photo, vehicle details, and rating. - (M)[BD][DBA]
    - Integrate the frontend component with the backend API to display the driver's profile and vehicle information on the tracking screen. Ensure that the data is fetched and displayed

correctly when the passenger is tracking their ride. - (M)[FD][QA]
    - Conduct testing to ensure that the driver's profile and vehicle information is displayed correctly and that the API integration works seamlessly. This includes unit tests and user acceptance testing. - (M)[QA]

- **Share Ride Details:** *As a passenger, I want to share my ride details with a friend so that they can track my ride for safety.* - There is an option to share ride details via messaging apps or social media. - The shared details include the driver's name, vehicle information, and ETA. - The shared link allows the recipient to view the driver's real-time location.
    - Design the UI component for sharing ride details, including options for messaging apps and social media. Ensure it is user-friendly and accessible. - (M)[UXD][FD]
    - Implement the backend logic to handle sharing ride details. This includes creating an API endpoint that formats the ride details (driver's name, vehicle info, ETA) for sharing. - (M)[BD]
    - Integrate real-time location tracking into the shared ride details. This involves setting up a mechanism to allow the recipient to view the driver's real-time location. - (C)[BD][FD]
    - Test the sharing functionality thoroughly, ensuring that all details are correctly formatted and that the real-time tracking works as intended. Include unit tests and integration tests. - (M)[QA][BD]
    - Deploy the new feature to the production environment and monitor for any issues or feedback from users regarding the sharing functionality. - (S)[DE][BD]

- **Receive Digital Receipt:** *As a passenger, I want to receive a digital receipt for my ride so that I have a record of my payment.* - The receipt is sent to my registered email address. - The receipt includes the fare breakdown and payment method. - The receipt is accessible within the app under my ride history.
    - Design the digital receipt template that will be sent to passengers via email. This template should include the fare breakdown, payment method, and ride details. - (M)[UI/UXD][FD]
    - Implement the backend logic to generate the digital receipt after a ride is completed. This will involve creating a function that compiles the ride details and formats them into the receipt template. - (M)[BD][DBA]
    - Set up the email service to send the digital receipt to the passenger's registered email address. This includes configuring SMTP settings and ensuring the email is sent upon ride completion. - (M)[BD][DevOps]
    - Create a section in the passenger's ride history within the app to allow users to view their digital receipts. This will involve frontend development to display the receipts in a user-friendly manner. - (M)[FD][UI/UXD]
    - Conduct testing to ensure that the digital receipt is generated correctly, sent to the passenger's email, and displayed in the app's ride history. This includes unit tests and integration tests. - (M)[QA][BD]

- **Rate Driver:** *As a passenger, I want to rate my driver after the ride so that I can provide feedback on my experience.* - I can rate the driver on a scale of 1 to 5 stars. - I have the option to leave

additional comments about my ride. - My rating is submitted successfully and reflected in the driver's profile.

  - Design and implement the UI for the rating system, allowing passengers to select a star rating and leave comments. This will include creating a modal or a dedicated page for the rating process. - (M)[UI/UXD][FD]
  - Develop the backend logic to handle the submission of the driver's rating and comments. This includes creating an API endpoint that validates and stores the rating in the database. - (M)[BD][DBA]
  - Implement a notification system to inform the driver of the new rating and comments submitted by the passenger. This will involve creating a service that sends notifications to the driver. - (M)[BD][DE]
  - Conduct testing for the rating feature, including unit tests for the backend logic and UI tests for the frontend. Ensure that the rating system works as expected and that data is correctly stored and retrieved. - (M)[QA][FD][BD]
  - Integrate the rating feature with the existing ride completion process, ensuring that passengers are prompted to rate their driver after the ride ends. - (M)[FD][BD]

Milestone 3: Driver Ride Acceptance and Navigation

- **Receive Ride Requests:** *As a driver, I want to receive ride requests from passengers so that I can accept rides and provide transportation.* - The driver receives a notification for a new ride request. - The ride request displays the passenger's pickup location and destination. - The driver can see the estimated fare for the ride.
  - Implement the notification system for new ride requests, ensuring drivers receive alerts on their devices when a passenger requests a ride. - (M)[BD][MD]
  - Create the ride request interface that displays the passenger's pickup location and destination to the driver. - (M)[FD][UXD]
  - Calculate and display the estimated fare for the ride based on the pickup and drop-off locations provided by the passenger. - (M)[BD][FD]
  - Integrate the ride request acceptance logic, allowing drivers to accept or decline ride requests through the app. - (M)[BD][MD]
  - Test the entire ride request flow to ensure that notifications, fare calculations, and acceptance logic work seamlessly together. - (S)[QA][BD][FD]

- **Accept or Decline Ride Requests:** *As a driver, I want to accept or decline ride requests so that I can manage my availability and workload.* - The driver can accept the ride request with a single tap. - The driver can decline the ride request with a single tap. - The app provides feedback (confirmation or cancellation) after the driver takes action.
  - Design the UI for the ride request acceptance screen, including buttons for accepting and declining requests. Ensure that the design is user-friendly and accessible. - (M)[UXD][FD]
  - Implement the functionality for the accept and decline buttons. Ensure that tapping the accept button sends a confirmation to the backend and updates the driver's status to 'busy'. - (M)[FD][BD]

    - Create the backend endpoint to handle ride request acceptance. This should update the ride status in the database and notify the passenger of the driver's acceptance. - (M)[BD][DBA]
    - Implement feedback mechanisms for the driver after accepting or declining a ride request. This includes displaying a confirmation message or an error message if the action fails. - (S)[FD][QA]
    - Conduct testing for the acceptance and decline functionalities, ensuring that all edge cases are covered and that the UI responds correctly to user actions. - (M)[QA][FD]

- **View Ride Request Time Limit:** *As a driver, I want to see the time limit for accepting a ride request so that I can make quick decisions.* - The app displays a countdown timer for the ride request acceptance. - The timer resets if the request is declined or accepted. - The driver receives a notification if the timer expires without action.
    - Design the UI component for displaying the countdown timer for ride request acceptance. Ensure it is visually prominent and updates in real-time. - (M)[UI/UXD][FD]
    - Implement the backend logic to handle the countdown timer for ride requests. This includes starting the timer when a ride request is sent and resetting it upon acceptance or decline. - (M)[BD]
    - Create a notification system that alerts the driver if the timer expires without action. This should include both in-app notifications and push notifications. - (C)[BD][FD][DE]
    - Test the countdown timer functionality to ensure it resets correctly and that notifications are sent when the timer expires. This includes unit tests and integration tests. - (M)[QA][BD]
    - Document the implementation details of the countdown timer feature, including how it interacts with other components of the ride request system. - (S)[TW][BD]

- **Receive Alerts for New Requests:** *As a driver, I want to receive alerts for ride requests while I am navigating to a previous passenger so that I can maximize my ride opportunities.* - The driver receives a notification for a new ride request even while en route to another passenger. - The driver can choose to accept the new request or continue to the current passenger. - The app provides a clear indication of the current ride status and any new requests.
    - Design the notification system to alert drivers of new ride requests while they are en route to a previous passenger. This includes defining the data structure for notifications and how they will be displayed in the app. - (M)[BD][FD][UXD]
    - Implement the backend logic to handle incoming ride requests and send notifications to drivers. This includes creating endpoints for ride requests and integrating with the existing ride management system. - (C)[BD][DBA]
    - Develop the frontend component to display ride request notifications to drivers. This includes designing the UI for notifications and ensuring it integrates seamlessly with the existing driver interface. - (M)[FD][UXD]
    - Create a mechanism for drivers to accept or decline new ride requests. This includes updating the ride status and notifying the passenger of the driver's decision. - (M)[BD][FD]

    - Test the entire flow of receiving ride requests while en route to another passenger, ensuring that notifications are received correctly and that the acceptance/decline functionality works as intended. - (M)[QA][BD][FD]
    - Document the new features and update the existing documentation to reflect changes in the ride request handling process and notification system. - (S)[TW][BD]

- **Turn-by-Turn Navigation:** *As a driver, I want to access turn-by-turn navigation to the passenger's pickup location so that I can reach them without getting lost.* - The app integrates with a navigation service (e.g., Google Maps) to provide directions. - The navigation updates in real-time as I drive. - The app allows me to switch between map views (e.g., satellite, street view).
    - Integrate a navigation service (e.g., Google Maps) into the app to provide turn-by-turn directions to the passenger's pickup location. - (M)[FD][BD][UXD]
    - Implement real-time updates for navigation as the driver progresses towards the pickup location, ensuring the app reflects any changes in route or traffic conditions. - (M)[FD][BD][DE]
    - Allow drivers to switch between different map views (e.g., satellite, street view) within the navigation interface. - (S)[FD][UXD]
    - Test the navigation feature to ensure it works seamlessly, including edge cases like poor connectivity or incorrect location data. - (M)[QAT][FD]
    - Deploy the navigation feature to production and monitor its performance, making adjustments as necessary based on user feedback. - (M)[DE][BD]

- **Traffic Alerts and Road Closures:** *As a driver, I want to see any traffic alerts or road closures on my route so that I can adjust my navigation accordingly.* - The app notifies me of any traffic incidents that may affect my route. - The app suggests alternative routes if there are significant delays. - The traffic information is updated in real-time.
    - Design the traffic alerts feature to notify drivers of any traffic incidents affecting their route. This includes defining the user interface elements that will display alerts and the logic for fetching real-time traffic data. - (M)[UX][FD]
    - Implement the backend service that fetches traffic alerts from a third-party API and processes this data to determine which alerts are relevant to the driver's current route. - (C)[BD][DE]
    - Integrate the traffic alerts feature with the driver's navigation system, allowing the app to suggest alternative routes if significant delays are detected. - (C)[BD][FD]
    - Create a real-time update mechanism for the traffic alerts, ensuring that the driver receives notifications as soon as new traffic incidents are reported. - (M)[BD][FD]
    - Conduct testing of the traffic alerts feature, including unit tests for the backend service and user acceptance testing for the frontend notifications. - (M)[QA][BD]

- **Call Passenger Option:** *As a driver, I want to have the option to call the passenger if I have trouble finding their location so that I can ensure a smooth pickup.* - The app provides a 'Call Passenger' button that connects me directly to their phone number. - The passenger's phone number is masked for privacy. - The call feature is accessible from the navigation screen.

    - Design the 'Call Passenger' button UI component that will be displayed on the navigation screen for drivers. Ensure that it is visually distinct and easily accessible. - (S)[UI/UXD][FD]
    - Implement the backend logic to handle the 'Call Passenger' feature. This includes creating an API endpoint that retrieves the passenger's phone number while ensuring privacy by masking it. - (M)[BD][DBA]
    - Integrate the 'Call Passenger' button with the backend API. Ensure that when the button is clicked, it triggers the API call to retrieve the passenger's masked phone number and initiates the call. - (M)[FD][BD]
    - Test the 'Call Passenger' feature to ensure it works correctly. This includes unit tests for the backend API and integration tests for the frontend button functionality. - (M)[QA][BD]
    - Deploy the 'Call Passenger' feature to the staging environment for further testing and validation before going live. - (S)[DevOps]

- **Cancellation Notification:** *As a driver, I want to receive notifications if the passenger cancels the ride while I am en route so that I can adjust my plans accordingly.* - The app sends a push notification if the ride is canceled. - The notification includes the reason for cancellation, if provided. - The app updates my status to 'Available' after cancellation.
    - Set up the backend service to handle ride cancellation notifications for drivers. This includes creating an endpoint that listens for ride cancellation events and triggers notifications to the respective driver. - (M)[BD]
    - Integrate a push notification service to send cancellation notifications to drivers. This involves configuring the service and ensuring it can send notifications based on the data received from the backend. - (M)[BD][DE]
    - Implement logic to update the driver's status to 'Available' after a cancellation. This requires modifying the driver's status in the database and ensuring the frontend reflects this change. - (S)[BD][FD]
    - Add functionality to include the reason for cancellation in the notification if provided. This involves modifying the notification payload to include this information. - (M)[BD]
    - Conduct thorough testing of the cancellation notification feature, including unit tests, integration tests, and user acceptance testing to ensure the feature works as intended. - (M)[QA][BD]

- **Complete Ride Drop-off:** *As a driver, I want to drop off the passenger at their destination so that I can complete the ride and move on to the next request.* - The app provides clear navigation to the passenger's drop-off location. - The app confirms the drop-off once the passenger exits the vehicle. - The ride status updates to 'completed' in the app.
    - Implement navigation to the passenger's drop-off location using the mapping service. Ensure that the driver can see the route clearly on their app. - (M)[BD][FD]
    - Create a confirmation dialog that appears once the passenger exits the vehicle, allowing the driver to confirm the drop-off. - (S)[FD][UXD]
    - Update the ride status to 'completed' in the backend once the drop-off is confirmed by the driver. Ensure that this status is reflected in the driver's app. - (M)[BD][QA]

    - Implement a notification system to inform the passenger that the ride has been completed and provide them with a receipt. - (M)[BD][FD][QA]

- **View Ride Summary:** *As a driver, I want to view the ride summary after completing the ride so that I can review the details of the trip.* - The app displays a summary of the ride, including pickup and drop-off locations, distance traveled, and fare. - I can access the ride summary from my ride history. - The summary includes an option to provide additional feedback if necessary.
    - Design the UI for the ride summary page that displays the ride details including pickup and drop-off locations, distance traveled, and fare. Ensure the design is user-friendly and aligns with the overall app theme. - (M)[UI/UXD]
    - Implement the backend logic to fetch ride summary details from the database after a ride is completed. This includes querying the ride details based on the driver's ID and the ride ID. - (M)[BD]
    - Integrate the frontend ride summary page with the backend API to display the fetched ride details. Ensure that the data is correctly rendered on the UI and handle any loading states or errors. - (M)[FD][BD]
    - Add functionality for the driver to provide additional feedback on the ride summary page. This includes creating a feedback form and implementing the backend logic to save the feedback. - (M)[FD][BD]
    - Conduct testing to ensure that the ride summary page displays the correct information and that the feedback functionality works as intended. This includes unit tests and integration tests. - (M)[QA]

Milestone 4: Administrator Platform Management and Support Handling

- **View Real-Time Ride Activity:** *As an administrator, I want to view real-time ride activity on the platform so that I can monitor ongoing rides and ensure service quality.* - The dashboard displays a live map with all active rides. - I can see details of each ride, including driver and passenger information. - I receive alerts for any unusual activity, such as long wait times or canceled rides.
    - Design the dashboard layout to display real-time ride activity, including a live map and ride details. - (M)[UXD][FD]
    - Implement the live map feature to show active rides using a mapping library (e.g., Google Maps API). - (C)[FD][BD]
    - Create a backend service to fetch real-time ride data from the database and push updates to the frontend. - (C)[BD][DBA]
    - Integrate alerts for unusual activity, such as long wait times or canceled rides, into the dashboard. - (M)[BD][FD]
    - Test the dashboard functionality, ensuring all features work as expected and alerts are triggered correctly. - (M)[QA][FD]

- **Access Driver Performance Metrics:** *As an administrator, I want to access driver performance metrics so that I can evaluate and support drivers effectively.* - I can view ratings and feedback for each driver. - I can see the number of rides completed by each driver over a specified period. - I can identify drivers with low performance ratings for follow-up.

    - Design the UI for the driver performance metrics dashboard, including sections for ratings, feedback, and completed rides. - (M)[UXD][FD]
    - Implement the backend API to fetch driver performance metrics, including ratings, feedback, and completed rides. - (M)[BD][DBA]
    - Create database schema for storing driver performance metrics, including fields for ratings, feedback, and completed rides. - (S)[DBA]
    - Develop the logic to identify drivers with low performance ratings for follow-up actions. - (M)[BD]
    - Test the driver performance metrics dashboard to ensure all data is displayed correctly and the UI is user-friendly. - (M)[QA][UXD]

- **Manage User Accounts:** *As an administrator, I want to manage user accounts so that I can onboard new drivers and handle any issues with passengers.* - I can approve or reject driver applications. - I can deactivate or suspend driver accounts if necessary. - I can view and manage passenger accounts, including resolving disputes.
    - Implement the functionality for administrators to approve or reject driver applications. This includes creating a user interface for viewing applications and a backend service to handle the approval process. - (M)[BD][FD][UXD]
    - Develop the feature to deactivate or suspend driver accounts. This will involve creating a toggle in the admin interface and a backend service to update the driver's status in the database. - (M)[BD][FD]
    - Create a dashboard for administrators to view and manage passenger accounts. This includes listing all passengers, viewing their details, and resolving disputes. - (C)[BD][FD][UXD]
    - Implement the functionality for administrators to resolve disputes between drivers and passengers. This will require a dedicated interface and backend logic to track and manage disputes. - (C)[BD][FD][UXD]

- **View All Drivers:** *As an administrator, I want to view a list of all drivers so that I can manage their accounts effectively.* - The system displays a searchable and filterable list of all drivers. - Each driver entry includes their status (active, inactive) and performance metrics (ratings, completed rides). - The administrator can click on a driver to view detailed information and history.
    - Design the UI for the driver management page that displays a list of all drivers. This includes creating a searchable and filterable table layout that shows driver names, statuses, and performance metrics. - (M)[UI/UXD][FD]
    - Implement the backend API endpoint to retrieve the list of drivers from the database. This endpoint should support filtering and searching based on driver attributes. - (M)[BD][DBA]
    - Create the database schema for storing driver information, including fields for driver status and performance metrics. Ensure that the schema supports efficient querying for the management page. - (M)[DBA][BD]
    - Develop the frontend functionality to call the backend API and display the retrieved driver data in the UI. This includes handling loading states and error messages. - (M)[FD][QA]

    - Implement the filtering and searching functionality on the frontend to allow administrators to easily find specific drivers based on their attributes. - (M)[FD][QA]
    - Conduct testing to ensure that the driver management page works as expected, including unit tests for the backend API and integration tests for the frontend functionality. - (M)[QA][BD][FD]

- **Manage Driver Status:** *As an administrator, I want to deactivate or reactivate drivers so that I can manage their availability on the platform.* - The system allows the administrator to change a driver's status to inactive or active. - The system prompts for confirmation before changing the driver's status. - The driver receives a notification regarding their status change.
    - Design the UI for the admin dashboard to manage driver status, including buttons for activating and deactivating drivers. Ensure the UI is intuitive and provides feedback on actions taken. - (M)[UXD][FD]
    - Implement the backend logic to handle driver status changes. This includes creating endpoints to activate and deactivate drivers, and ensuring that the changes are reflected in the database. - (M)[BD][DBA]
    - Integrate the frontend UI with the backend API to ensure that the admin can successfully change a driver's status. This includes handling API responses and updating the UI accordingly. - (M)[FD][BD]
    - Create a confirmation dialog that prompts the admin to confirm the action before changing a driver's status. This should include success and error messages based on the outcome of the action. - (S)[UXD][FD]
    - Implement notification logic to inform drivers of their status change via email or in-app notification. Ensure that the notification includes the reason for the status change. - (M)[BD][DE]
    - Conduct testing (unit and integration) to ensure that the driver status management feature works as intended and meets the acceptance criteria. This includes testing the UI, API, and notification system. - (M)[QA][BD]

- **Handle Driver Disputes:** *As an administrator, I want to handle driver disputes so that I can resolve issues between drivers and passengers effectively.* - The system provides a dashboard for viewing active disputes. - The administrator can review dispute details and communicate with involved parties. - The system allows the administrator to document the resolution process and outcome.
    - Create a dashboard for administrators to view all active driver disputes, including details such as involved parties, dispute status, and timestamps. This will involve designing the UI and implementing the backend logic to fetch and display the data. - (M)[FD][BD][UXD]
    - Implement functionality for administrators to click on a dispute from the dashboard to view detailed information, including comments from both drivers and passengers. This will require backend endpoints to fetch dispute details and frontend components to display them. - (M)[FD][BD]
    - Develop a communication feature that allows administrators to send messages to both drivers and passengers involved in a dispute. This will involve creating a messaging interface and backend logic to handle message sending and retrieval. - (C)[FD][BD][QA]

    - Create a feature for administrators to document the resolution process of disputes, including notes and outcomes. This will require backend storage for resolution data and a frontend interface for input. - (M)[FD][BD]
    - Implement a feature that allows administrators to view the history of resolved disputes for reference. This will involve creating a new section in the dashboard and backend logic to fetch historical data. - (M)[FD][BD]

- **Manage Driver Payments:** *As an administrator, I want to manage driver payments so that I can ensure timely compensation for their services.* - The system displays a summary of driver earnings and payment history. - The administrator can initiate payments or adjustments as needed. - The system sends notifications to drivers regarding payment status.
    - Design the user interface for the driver payments management dashboard, including sections for payment summary, payment history, and action buttons for initiating payments. - (M)[UI/UXD][FD]
    - Implement the backend logic to retrieve driver earnings and payment history from the database, ensuring data integrity and security. - (M)[BD][DBA]
    - Create the API endpoint for initiating payments or adjustments, including validation of payment details and integration with the payment processing system. - (C)[BD][DE]
    - Develop the notification system to inform drivers about their payment status, including successful payments and any adjustments made. - (M)[BD][DE]
    - Conduct testing for the payment management system, including unit tests for the backend logic and user acceptance testing for the UI. - (M)[QA][BD]

- **Respond to Support Requests:** *As an administrator, I want to respond to support requests directly through the platform so that I can provide timely assistance to users.* - The administrator can select a support request and enter a response. - The response is sent to the user and logged in the system. - The user receives a notification of the response.
    - Design the UI for the support request response feature, allowing administrators to select a request and enter a response. - (M)[UXD][FD]
    - Implement the backend logic to handle the submission of responses to support requests, ensuring they are logged in the system. - (M)[BD][DBA]
    - Create a notification system to alert users when their support request has been responded to by an administrator. - (M)[BD][FD]
    - Test the entire support request response feature to ensure that responses are sent correctly and logged in the system. - (S)[QAT][BD]

- **Escalate Support Issues:** *As an administrator, I want to escalate support issues to higher management if they cannot be resolved quickly so that critical issues are addressed promptly.* - The administrator can mark a support request as escalated. - Escalated requests are highlighted in the support request list. - Higher management receives a notification of the escalated issue.
    - Design the user interface for escalating support issues, including a button to mark requests as escalated and a visual indicator for escalated requests in the support request list. - (M)[UI/UXD][FD]

    - Implement the backend logic to handle the escalation of support requests, including updating the request status in the database and notifying higher management. - (M)[BD][DBA]
    - Create a notification system that alerts higher management when a support issue is escalated, ensuring they receive timely updates on critical issues. - (C)[BD][DE]
    - Test the escalation feature to ensure that it works as intended, including verifying that escalated requests are highlighted and notifications are sent to higher management. - (M)[QA][BD]

- **Generate Support Request Reports:** *As an administrator, I want to generate reports on support requests so that I can analyze trends and improve service quality.* - The system allows the administrator to select a date range for the report. - The report includes metrics such as the number of requests, average response time, and resolution rate. - The administrator can export the report in a downloadable format (e.g., CSV, PDF).
    - Design the user interface for the support request report generation feature, allowing administrators to select a date range and view metrics. - (M)[UI/UXD][FD]
    - Implement the backend logic to fetch support request data based on the selected date range, including metrics such as number of requests, average response time, and resolution rate. - (C)[BD][DBA]
    - Create the report generation functionality that compiles the fetched data into a downloadable format (e.g., CSV, PDF). - (M)[BD][QA]
    - Develop the export feature that allows administrators to download the generated report in the selected format. - (S)[FD][BD]
    - Conduct testing to ensure the report generation feature works correctly, including unit tests and user acceptance testing. - (M)[QA][BD]