Landing Page

Visual and Interactive Elements

1. Logo:

• The wolf logo will appear in the center of the screen.

• The logo will have a smooth bounce animation, transitioning in size from 1.0x to 2.0x and back to 1.0x repeatedly while the app performs background tasks.

2. Background:

• A clean black background ensures focus on the logo.

• This minimalist approach reduces distractions and emphasizes the branding.

Functionalities

1. Animated Logo Behavior:

• The logo will:

• Start at 1.0x size.

• Smoothly grow to 2.0x size.

• Shrink back to 1.0x size in a continuous loop until the main action (e.g., location detection) is complete.

2. Location Detection:

• The app will initiate location services in the background.

The app will:

• Request location permissions if not already granted.

• Detect the user's location using GPS, Wi-Fi, or cellular data.

• If location services are disabled:

• Show a prompt or pop-up asking the user to enable location access.

• Provide an option to proceed without location detection, if applicable.

3. Transition Effect:

• Once the location is successfully detected:

• The logo will enlarge from its current size (1.0x) to 2.0x.

• The logo will fade out or disappear smoothly.

• The app will transition to the next screen (e.g., a dashboard or home screen) with no delay.

4. Fallback/Error Handling:

• If location detection fails (e.g., no GPS signal or permissions denied):

• Show an error message or prompt to retry.

• Allow manual input of location or provide an option to proceed without location access.

Technical Details

1. Location Services:

• Use the device's location APIs:

• iOS: Core Location framework.

• Android: Fused Location Provider API.

• Ensure smooth handling of location permissions:

• If permissions are denied, show a dialog with a link to the device's settings to enable them.

2. Animation Framework:

• Implement the bounce animation using:

• CSS or JavaScript for web apps.

• Android's ObjectAnimator or PropertyAnimation.

• iOS UIView.animate with scale transforms.

• The fade-out and transition effects will use opacity changes and smooth scaling.

3. Next Screen Transition:

• After the animation ends and location detection is complete:

• Navigate the user to the next screen using a fade or slide transition.

• Ensure that any resources (like location APIs) are released when the user transitions to the next screen.

Enhancements for Engagement

1. Loading Indicator: Subtly integrate a loading progress bar or spinner behind the logo to indicate that the app is working in the background.

2. Sound Effect (Optional): Add a subtle sound effect when the logo reaches 2.0x size for added interactivity.

3. Onboarding Screen (Optional): If this is the first launch, transition to a brief onboarding/tutorial screen after location detection.

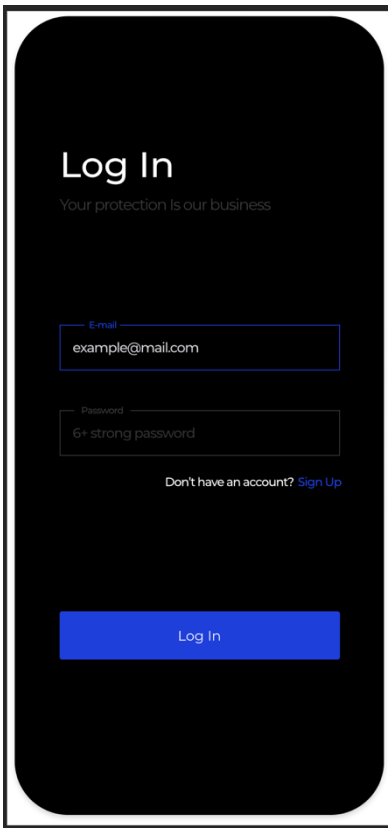
User Flow

1. App opens → The wolf logo animates (bounce effect) → Location detection starts.

2. Location is successfully detected: Logo grows to 2.0x → Fades out/disappears → User transitions to the next screen.

3. If location detection fails: Prompt the user to enable location access or manually input their location.

Sign Up/Sign In Page



Visual and Interactive Elements

1. Title and Subtitle:

• "Log In" displayed prominently as the title.

• A subtitle, "Your protection is our business," serves as a tagline to reassure users of the app's purpose.

2. Input Fields:

• Email Field:

• Placeholder: example@mail.com.

• Input validation for proper email format (e.g., @ and .com required).

Password Field:

• Placeholder: 6+ strong password.

• Includes masking for password input with an option to toggle visibility (eye icon).

• Validation to ensure the password meets the required criteria (e.g., at least 6 characters).

3. Login Button:

• Blue button with the label "Log In."

• Becomes active only when valid credentials are entered (e.g., email format and password strength).

4. Sign-Up Link:

• A clickable "Sign Up" text below the form, redirecting users to the sign-up screen.

5. Social Login Options:

• Buttons for Google, X (formerly Twitter), Microsoft, and Facebook sign-in/sign-up displayed below the form.

Each button includes:

• Icon of the respective platform.

• Label (e.g., “Sign in with Google”).

• Buttons redirect users to the corresponding OAuth login flow.

Functionalities

1. User Login with Email/Password:

• Users input their credentials and tap “Log In.”

The app validates:

• Email format.

• Password strength and match (if stored).

• If validation fails, an error message appears (e.g., “Invalid email or password”).

2. Social Sign-In and Sign-Up:

• Integration with OAuth 2.0 for:

• Google

• X (Twitter)

• Microsoft

• Facebook

Each option:

• Redirects to the respective platform’s authorization page.

On successful authentication, retrieves user profile data (e.g., name, email, profile picture).

• Registers new users if no account exists.

•Post-login, users are redirected to the app’s dashboard.

3. Form Validation:

• Real-time validation for email format and password strength.

• Displays error messages below fields if input is invalid.

4. Forgot Password (Optional):

- Add a “Forgot Password?” link near the password field.

- Redirects users to a reset password page or initiates an email-based password reset process.

5. Sign-Up Flow: Clicking “Sign Up” redirects users to a dedicated sign-up screen or launches a modal form for account creation.

6. Responsive Behavior:

- Ensures all elements resize and align correctly across various screen sizes.

7. Error Handling:

- If login fails (e.g., incorrect credentials, server issues):

- Display error message like “Unable to log in. Please try again later.”

- Retry option for failed attempts.

Technical Details

1. Backend:

- Authentication handled using a secure backend with hashed passwords (e.g., bcrypt).

- OAuth integrations with APIs for Google, X, Microsoft, and Facebook.

2. Frontend:

- Input field validation done client-side with fallback server-side validation.

- Real-time feedback for errors using JavaScript or a front-end framework (e.g., React, Flutter).

3. Security:

- HTTPS enforced for all requests.

- OAuth tokens securely stored (e.g., encrypted storage or Secure Enclave on iOS).

- Brute-force prevention mechanisms like rate-limiting.

4. Transition to Dashboard:

• After successful login, users are redirected to the app's main dashboard or home screen.

• Ensure session management is in place (e.g., JWT tokens).

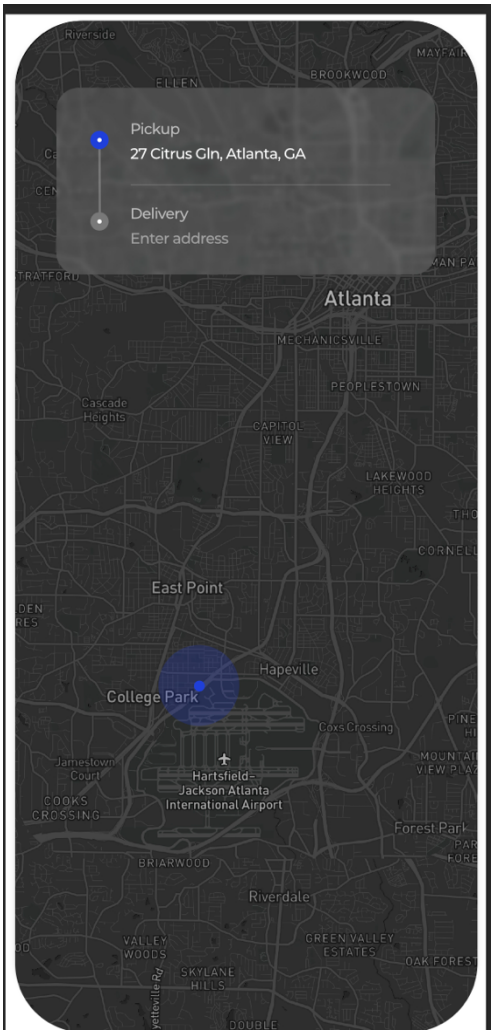
User Flow

1.User enters email and password → taps "Log In" → credentials are validated → user is redirected to the dashboard.

2.User taps "Sign in with Google/X/Microsoft/Facebook" → redirected to OAuth flow → successful login redirects to the dashboard.

3.User clicks "Sign Up" → redirected to the sign-up screen → completes registration → redirected to the dashboard.

Map-based screen



Visual and Interactive Elements:

1. Map Background:

• Dark-themed map with visible location details such as streets and landmarks.

• Real-time GPS location tracking enabled.

• User's current location highlighted with a blue dot.

2. Pickup and Delivery Information Box:

• Floating semi-transparent card at the top:

• Pickup Address: Pre-filled or manually entered by the user (e.g., 27 Citrus Gln, Atlanta, GA).

• Delivery Address: Input field for the destination address (e.g., "Enter address").

• Dynamic vertical line connecting "Pickup" and "Delivery" icons:

• "Pickup" icon: Blue dot indicating the starting location.

• "Delivery" icon: Grey dot that turns blue once an address is entered.

3. Add Stop Button:

• Positioned below the Pickup/Delivery card or in the floating card itself.

• Clicking the button:

• Adds a new "Stop" field between Pickup and Delivery.

• Updates the connecting line to include the new Stop.

• Each Stop has its own icon (e.g., a grey or blue circle) and editable address field.

Functionalities:

1. Real-Time Location Updates:
   - Continuously tracks the user's location and updates the map.
   - Centers the map view around the user's current location by default.
2. Address Input:
   - Users can manually type or search for Pickup, Delivery, and Stop addresses.
   - Auto-complete suggestions provided using a geolocation API (e.g., Google Maps API, Mapbox API).
   - On address selection, the map updates to highlight the location with a pin.
3. Add Stop Feature:
   - Users can add intermediate stops between Pickup and Delivery.
   - Each Stop dynamically adjusts the route on the map.
   - Limit the number of stops if required (e.g., maximum 3 stops).
4. Dynamic Routing:
   - Calculates the optimal route based on Pickup, Stops, and Delivery locations.
   - Displays a highlighted path on the map connecting all points in the order entered.
5. Interactive Map:
   - Users can zoom in/out and pan around the map.
   - Tapping on a pin highlights its address in the Pickup/Delivery/Stop fields.
6. Error Handling:
   - Validates addresses to ensure they are reachable.
   - Shows error messages for incomplete or invalid addresses.

Technical Details:

1. Backend:

• Use a routing API (e.g., Google Directions API, Mapbox Directions API) to calculate routes between multiple points (Pickup, Stops, Delivery).

• Integrate a geocoding API for address validation and auto-complete suggestions.

2. Frontend:

• Map rendered using libraries like Google Maps SDK, Mapbox SDK, or Leaflet.js.

• Smooth animations for adding/removing Stops and updating routes dynamically.

3. Responsive Design:Ensure compatibility with various screen sizes and orientations.

4. Security:

• Secure API requests with tokens to prevent unauthorized access to geolocation services.

• Handle user location data in compliance with privacy regulations (e.g., GDPR, CCPA).

User Flow:

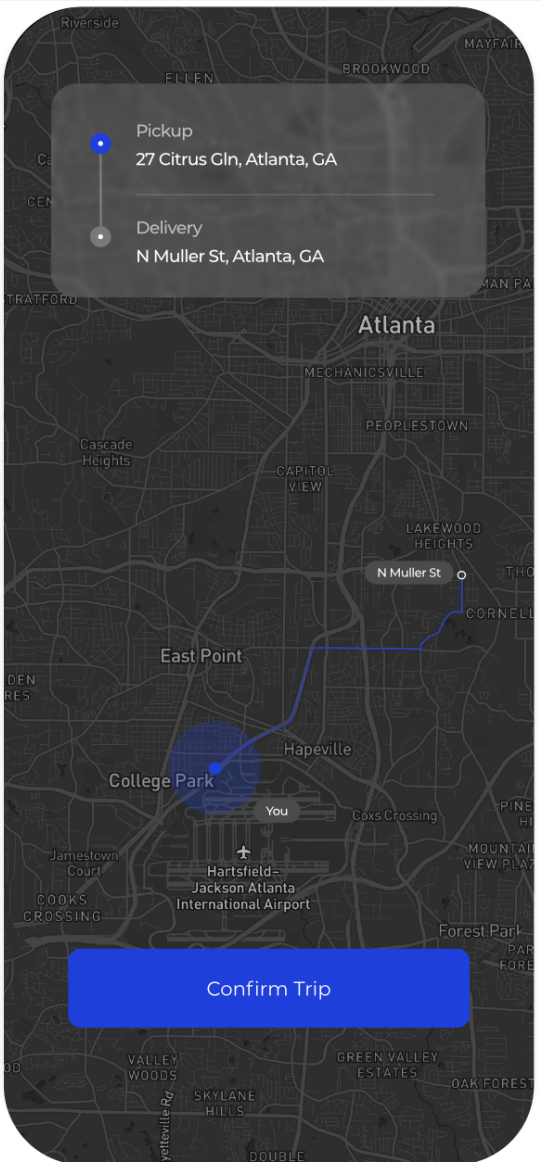
1. User opens the screen → Pickup location is pre-filled or entered manually.

2. User enters the Delivery address → Map updates with a route.

3. User taps “Add Stop” → A new Stop field appears → User enters Stop address → Map recalculates the route.

4. User reviews all locations (Pickup, Stops, Delivery) → Proceeds to the next step.

Confirm Trip



Visual and Design Elements:

1. Map Background:

• The map is dark-themed with visible streets, landmarks, and locations, ensuring high contrast for the interactive elements.

• The user's current location is displayed as a black circular pin labeled "You."

• A white line connects the Pickup, Stop(s) (if any), and Delivery points to represent the route.

2. Pickup and Delivery Pins:

• Pickup Pin: Displayed as a black pin marking the starting location.

• Delivery Pin: Displayed as a red pin marking the destination.

• Any additional Stop points, if added, are displayed as black pins similar to the Pickup pin.

• The pins are draggable, allowing users to adjust their locations on the map.

3. Floating Address Card:

• A semi-transparent floating card at the top displays:

• Pickup Address: Pre-filled or entered by the user, such as 27 Citrus Gln, Atlanta, GA.

• Delivery Address: Manually entered or selected, such as N Muller St, Atlanta, GA.

• A vertical line connects the Pickup and Delivery fields with their respective colors: black for Pickup and red for Delivery.

4. Route Line:

• A smooth white line is drawn on the map, dynamically updating based on the entered addresses (Pickup, Stop, and Delivery locations).

• The route recalculates in real-time when any address is modified or if the pins are dragged.

5. Confirm Button:

• A large blue “Confirm Trip” button is positioned at the bottom of the screen.

• Tapping this button finalizes the trip details and transitions to the next screen (e.g., trip summary or dispatch).

Functionalities:

1. Dynamic Address Input:

• Users can manually enter addresses for Pickup, Delivery, and additional Stops.

• Auto-complete suggestions are provided via a geolocation API (e.g., Google Maps API, Mapbox API).

• On selecting an address, the map updates with a corresponding pin.

2. Real-Time Routing:

• The app calculates and displays the optimal route connecting all points in the order entered (Pickup → Stops → Delivery).

• The route updates dynamically when locations are modified.

3. Draggable Pins:

• Users can drag the black or red pins on the map to adjust their locations.

• The address fields update automatically to reflect the new pin locations.

4. Stops Functionality:

• Users can add intermediate stops between Pickup and Delivery points.

• Stops are represented as black pins and are included in the route calculation.

5. Zoom and Pan:

• Users can zoom in, zoom out, and pan around the map for a better view of the route.

• A button to reset the view centers the map on the user's current location.

6. Error Handling:

• If an address is invalid or unreachable, an error message is displayed (e.g., "Invalid address, please try again").

• The app prevents users from proceeding until all fields are valid.

7. Confirm Trip:

•Tapping the "Confirm Trip" button validates all input fields and finalizes the trip.

•The app transitions to the next screen, such as a trip summary or dispatch notification.

8. Location Services:

• The app requests location permissions and uses GPS to determine the user's current location.

• If permissions are denied, the app prompts the user to enable them or manually enter the Pickup address.

Technical Details:

1. Geolocation and Routing:

• The app integrates with APIs such as Google Maps Directions API or Mapbox Directions API for route calculation.

• Geocoding APIs handle address validation and auto-completion.

2. Frontend Framework:

• The map is rendered using a mapping library like Google Maps SDK, Mapbox SDK, or Leaflet.js.

• Smooth animations for pin dragging, route updating, and screen transitions.

3. Backend Integration:

• A backend server manages address validation, route optimization, and trip confirmation data.

• All sensitive data (e.g., user location) is securely transmitted via HTTPS and handled in compliance with privacy regulations.

4. Responsive Design:The layout adapts to various screen sizes, ensuring usability on smartphones and tablets.

5. Error Prevention:

• Validates all addresses and routes before allowing users to confirm the trip.

• Displays appropriate messages for errors (e.g., “Route not found” or “Enter a valid address”).

User Flow:

1. The user opens the screen → The Pickup location is pre-filled or entered manually.

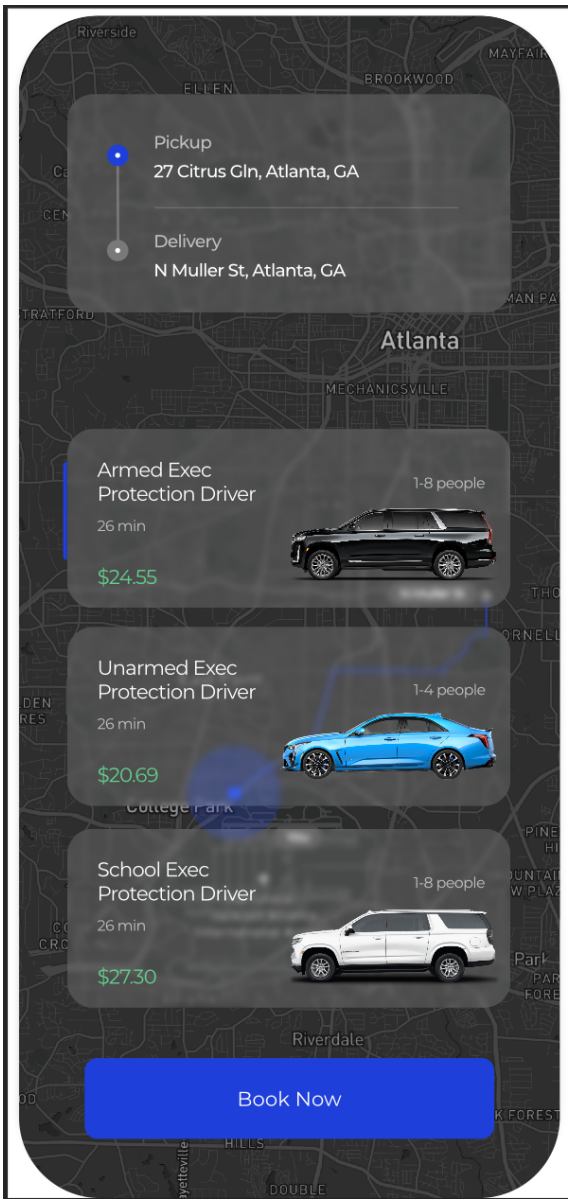
2. The user enters the Delivery address → The map updates with a route connecting the points.

3. Additional Stops are added (if required) → The map recalculates the route dynamically.

4. The user adjusts pin locations by dragging (if needed) → The addresses update automatically.

5. The user taps “Confirm Trip” → The app validates the data and transitions to the next step.

Cab Selection



Visual and Design Elements

1. Map Background:

• The map is dark-themed, displaying streets and landmarks.

• A white route line dynamically connects the Pickup and Delivery locations.

• The Pickup point is marked with a black pin, and the Delivery point is marked with a red pin.

• The map centers around the route to give the user a clear view of the trip.

2. Pickup and Delivery Details:

• A semi-transparent floating card at the top of the screen displays:

• Pickup Address: For example, 27 Citrus Gln, Atlanta, GA, with a corresponding black icon.

• Delivery Address: For example, N Muller St, Atlanta, GA, with a corresponding red icon.

• A vertical line visually connects the Pickup and Delivery points on the card.

3. Service Cards:

• Each service option is presented as a card below the map, featuring:

• Service Name: For example, “Armed Exec Protection Driver.”

• Travel Time: The estimated trip time, such as “26 minutes.”

• Price: The cost of the service, displayed prominently in green (e.g., “$24.55”).

• Seating Capacity: For example, “1–8 people.”

• Vehicle Image: A clear, high-resolution image of the vehicle type associated with the service (e.g., a black SUV for armed service).

• Cards are arranged in a vertical stack, allowing users to scroll if more options are available.

• Selected cards are highlighted to indicate the user's choice.

4. Confirmation Button:

• A large blue "Book Now" button is positioned at the bottom of the screen.

• It provides a clear call to action, enabling users to finalize their booking.

Functionalities

1. Map and Route Display:

• The map shows the route between the Pickup and Delivery locations using a white line.

• The black pin marks the Pickup point, and the red pin marks the Delivery point.

• The route dynamically updates if the user adjusts the Pickup or Delivery location.

2. Service Selection:

• Users can tap on any service card to select it.

• Upon selection, the card is visually highlighted (e.g., with a border or shadow effect).

• Each card displays the travel time, price, seating capacity, and a vehicle image, helping users make an informed decision.

3. Dynamic Pricing and Availability:

• Pricing and availability for each service update dynamically based on factors like time of day, demand, and distance.

4. Scrolling:

• If there are more service options than can fit on the screen, users can scroll vertically to view additional cards.

5. Booking Confirmation:

• Tapping the "Book Now" button validates the user's selection.

• The app checks for errors (e.g., no service selected) and prompts the user to correct them if necessary.

• Once confirmed, the app transitions to the next screen, such as a booking summary or payment page.

6. Error Handling:

• If a user tries to proceed without selecting a service, an error message is displayed (e.g., “Please select a service before proceeding”).

• If a service becomes unavailable after selection, the app prompts the user to choose a different option.

7. Responsive Design:

• The layout adjusts seamlessly to fit different screen sizes, ensuring usability on both smartphones and tablets.

Technical Details

1.Frontend:

• Developed using a mobile-friendly framework like Flutter, React Native, or Swift.

• Smooth animations for card selection, route updates, and transitions between screens.

2. Backend:

• Integration with a backend system to fetch service options, pricing, and availability in real time.

• Route optimization and travel time calculation using a mapping API (e.g., Google Maps Directions API or Mapbox).

3. Security:

• All communication between the app and server is secured via HTTPS.

• Sensitive user data (e.g., location) is handled in compliance with privacy laws like GDPR and CCPA.

4. Localization:

• Support for multiple languages and currencies, ensuring usability across regions.

User Flow

1. The user opens the screen → The Pickup and Delivery locations are pre-filled or entered manually.

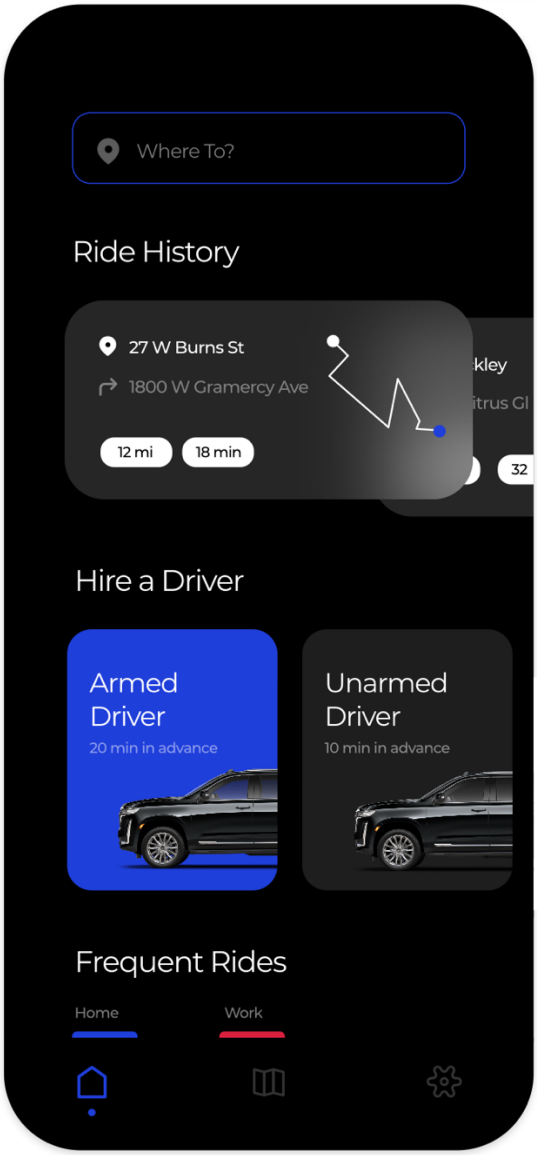
2. The route is displayed on the map with a white line connecting the points.

3. Service options are displayed below the map as scrollable cards.

4. The user taps a card to select a service → The card is highlighted to confirm the selection.

5. The user taps the “Book Now” button → The app validates the selection and transitions to the booking summary screen.

Home Screen



Core Functionalities

1. Ride Booking:
   - Where To? field for entering destinations with autocomplete using Google Maps API or Mapbox.
   - Distance and time estimations displayed below the address.
2. Ride History:

   • A list of past rides, including addresses, dates, and routes.

   • Feature to reorder rides or view detailed trip summaries.
3. Driver Options:

   • Armed Driver: Premium option with a higher price and longer wait time.

   • Unarmed Driver: Standard option with shorter wait times.

   • Real-time driver tracking using GPS integration.
4. Frequent Rides:

   • Add “Home,” “Work,” or custom frequently visited destinations for quick booking.
5. Safety and Features:

   • Panic button for emergencies, immediately alerting authorities or the company.

   • Driver details with verification ratings and real-time updates.
6. Settings:

   • Profile management, payment methods, and preferences.

   • Option to toggle between light and dark modes.

Backend Development

1. Technologies:

    • Server: Node.js or Python (Flask/Django).

    • Database: PostgreSQL or MongoDB to store user data, ride history, and driver details.

    • Cloud Services: AWS or Firebase for scalability and real-time notifications.

2. APIs:

    • Google Maps or Mapbox for location-based services.

    • Stripe or PayPal for secure payment integration.

    • Twilio for SMS notifications or driver communication.

3. Features:

    • Authentication with OAuth (Google, Apple).

    • Secure storage of sensitive data (payment methods, ride history) with encryption.

Frontend Development

1. Technologies:

    • Framework: React Native or Flutter for cross-platform compatibility.

    • UI Libraries: Tailwind CSS or Material UI for responsive design.

2. Design Elements:

    • Navigation Bar: Includes “Home,” “Ride History,” “Frequent Rides,” and “Settings.”

    • Card Designs: For ride options and driver details with sleek, modern UI.

3. Animations:

    • Smooth transitions between screens (e.g., booking to ride summary).

    • Real-time map updates showing driver movement.

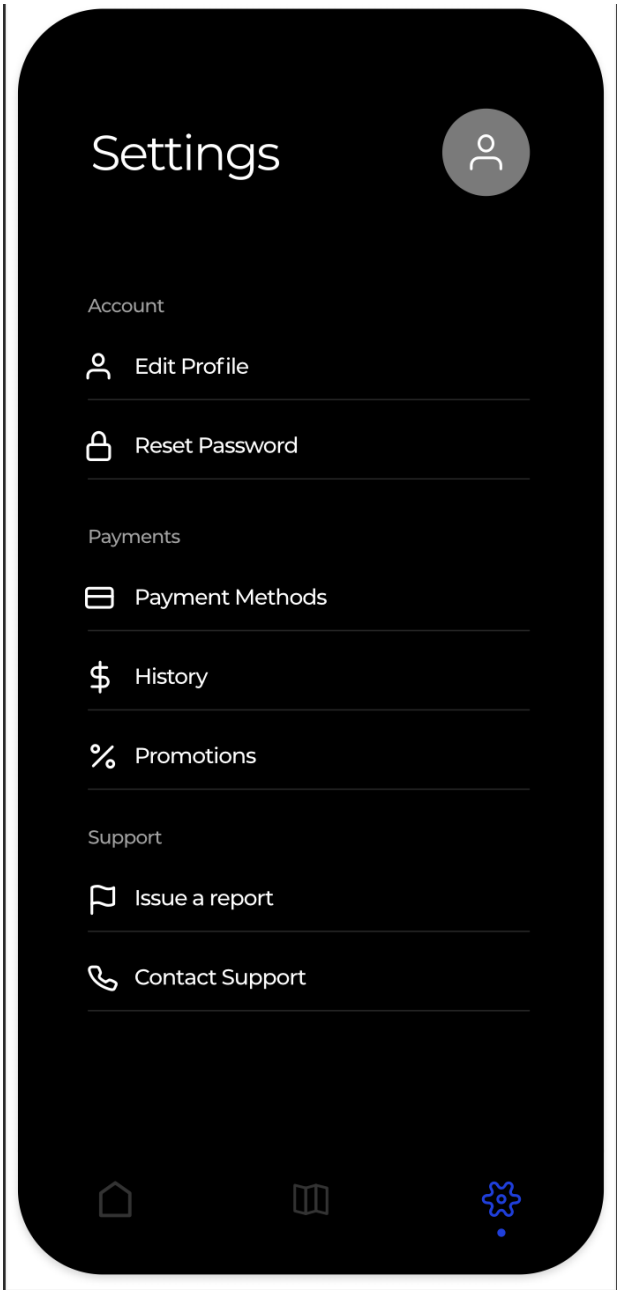
Deployment

- Hosting: Use AWS or Firebase for the backend and database hosting.
- Mobile Platforms: Deploy on both Google Play Store (Android) and App Store (iOS).
- CI/CD Pipeline: Automate deployment using tools like Jenkins or GitHub Actions.

Next Steps

1. Finalize the detailed Figma design.
2. Build a prototype with basic functionality to test the user flow.
3. Expand the backend with APIs and connect the database.
4. Integrate the frontend with backend services and test extensively.

Settings Screen



Settings Screen Functionalities

1. Account Management:

• Edit Profile:

• Allows users to update their name, email, phone number, and profile picture.

• Reset Password:

• Provides a secure interface to change passwords with email or phone verification.

2. Payment Settings:

• Payment Methods:

• Manage saved credit/debit cards, add new payment methods, or integrate PayPal/Apple Pay.

• History:

• View detailed transaction history, including ride costs, dates, and payment methods used.

• Promotions:

• Apply promo codes or view active discounts/offers.

3. Support Options:

• Issue a Report:

•File complaints or report issues related to rides, payments, or the app itself.

• Includes dropdown options like “Driver Issue,” “Payment Issue,” “App Bug.”

• Contact Support:

• Quick access to support via phone, chat, or email.

Design Details

1. Visual Style:

• Dark Mode Theme with clean white typography and minimalistic icons for better visibility and focus.

• Rounded edges for sections to create a modern feel.

2. Icons and Labels:

• Use universally recognized icons for sections (e.g., user for profile, lock for password, flag for reports).

• Descriptive labels that are straightforward and user-friendly.

3. Navigation:

• Bottom Navigation Bar:

• Includes Home, Ride History, and Settings for quick navigation.

• Ensure active tab (Settings) is highlighted.

4. Interactive Elements:

• All sections are tappable cards with a slight hover or shadow effect to indicate interactivity.

• Include feedback animations (e.g., slight color change) when a section is tapped.

Backend Functionality

1. User Profile API: Fetch and update user data securely.

2. Payment Gateway API: Integrate Stripe or PayPal for managing payment methods and transaction history.

3. Support Backend: Connect issue reporting to a ticketing system (e.g., Zendesk or Freshdesk).

4. Database: Store user preferences, payment history, and support queries in a secure database like MongoDB or PostgreSQL.

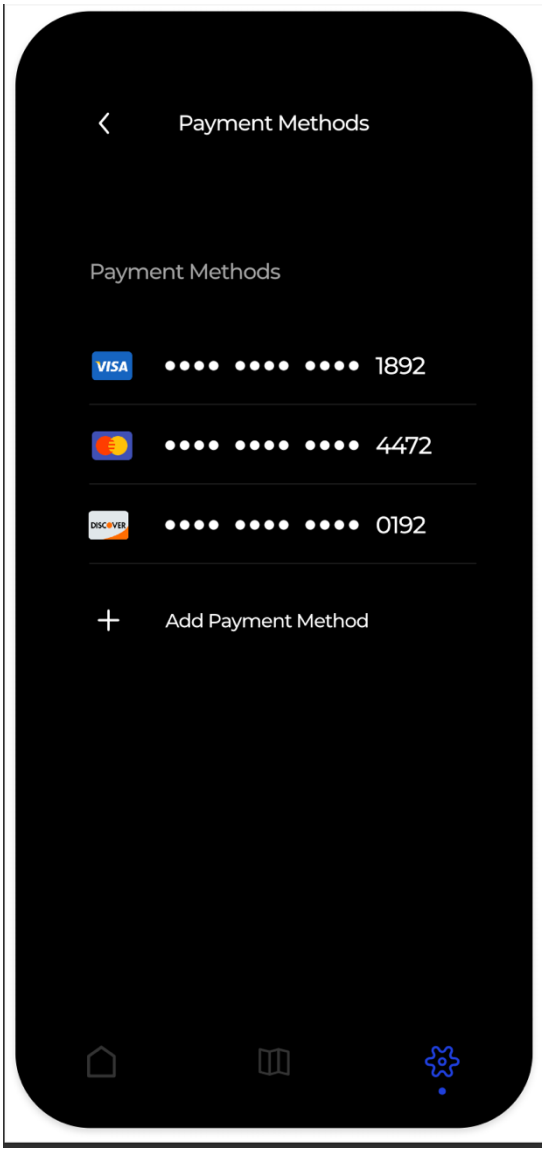
Frontend Development

1. Technologies:
    - Framework: React Native or Flutter for cross-platform compatibility.
    - State Management: Redux (React) or Provider (Flutter) to handle user data.
2. Components:
    - Custom cards for each section (e.g., Account, Payments, Support).
    - Modal popups for detailed actions (e.g., Reset Password, Add Payment).

Additional Features

- Push Notifications: Notify users of promotions or payment issues.
- Localization: Support multiple languages for accessibility.
- Accessibility: Ensure screen reader compatibility and easy navigation.

Payment Page



Functionalities

1. Payment Methods List:

• Credit/Debit Cards: Display saved cards with their logos (Visa, Mastercard, etc.) and masked card numbers for security.

• Apple Pay: Integration for Apple Pay with the Apple logo and an “Activate/Use Apple Pay” option.

• BlackWolf Wallet: Display the wallet balance and a “Top Up” button to add funds.

2. Add Payment Method:

• New Credit/Debit Card: Add new card details (card number, expiration date, CVV) via a secure form.

• Connect Apple Pay: Guide users to activate Apple Pay if not set up.

• Top-Up Wallet: Option to add funds to the BlackWolf Wallet via linked cards or Apple Pay.

3. Actions:

• Edit/Remove Payment Methods: Allow users to delete or update saved cards.

• Set Default Method: Option to select a default payment method (e.g., Apple Pay or a specific card).

• Secure Transactions: Enable two-factor authentication for certain actions (e.g., adding a new card).

Design Features

1. Visual Style:

    • Dark theme consistent with the overall app design.

    • Payment logos (Visa, Mastercard, Apple Pay) for easy recognition.

    • Rounded cards for each payment method with hover effects for interactivity.

2. Sections:

    • Payment Methods: Display existing payment options.

    • Add Payment Method: Prominent "+" button with clear labeling.

    • BlackWolf Wallet: A separate card displaying wallet balance with a "Top Up" option.

3. Interactive Elements:

    • Tap on a payment method to view/edit details.

    • Use tooltips or info icons to explain features (e.g., "What is BlackWolf Wallet?").

Backend Integration

1. Payment Gateway:

    • Use Stripe or PayPal for processing credit/debit card payments.

    • Apple Pay integration through Apple Developer APIs.

    • Custom wallet transactions through a secure backend system.

2. APIs:

    • Retrieve payment methods (GET request).

    • Add new payment methods (POST request).

    • Delete or update payment methods (DELETE/PUT request).

3. Database:

    • Store encrypted payment details for security.

- Maintain wallet balances and transaction history.

Frontend Development

1. Technologies:
   - React Native or Flutter for a responsive UI.
   - Integrate Apple Pay and custom wallet UI components.
2. Components:
   - Dynamic card display for each payment method.
   - Modal for adding new payment methods or topping up the wallet.

Additional Features

1. **BlackWolf Wallet Rewards**:
   Display rewards or cashback earned through the wallet.
2. **Payment History**:
   Option to view transactions made using each payment method.
3. **Localization**:
   Support for multiple currencies in the wallet and payment systems.