**USER ACCEPTANCE TESTS**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Case** | **Pre-conditions** | **Test Steps** | **Test Criteria** | **Success** |
| **Test Case 1: User Account Creation (Sign Up)** | * User does not have an existing account * User is on the application home page * User is not currently logged in | 1. Expand the navigation bar 2. Click on the "Sign Up" button/link 3. Verify that a signup modal form appears 4. Fill out the form with valid data 5. Click the "Sign Up" button | * Signup modal appears correctly with all required fields * Form validates input appropriately * User receives confirmation that signup was successful (notification) * User is automatically logged in after successful signup * Navbar changes to authenticated state displaying username/profile * Logout button appears in place of signup/login buttons * User account is created in the database * User can navigate to other pages while remaining logged in | **✓** Pass if user account is created successfully, user is automatically logged in, and UI reflects authenticated state |
| **Test Case 2: User Login with Existing Account** | * User has an existing account in the system * User is on the application home page * User is not currently logged in | 1. Expand the navigation bar 2. Click on the "Login" button 3. Verify that a login modal form appears 4. Fill out the form with valid credentials: 5. Click the "Login" button | * Login modal appears correctly with username and password fields * Form accepts valid credentials * User receives confirmation that login was successful (alert/notification) * Navbar changes to authenticated state displaying username/profile * Logout button appears in place of signup/login buttons * User session is created and maintained * User can access protected/authenticated features | **✓** Pass if user successfully logs in with valid credentials and UI reflects authenticated state |
| **Test Case 3: User Logout** | * User is logged in to the application * Navbar shows authenticated state | 1. Locate the "Logout" button in the navbar 2. Click the "Logout" button | * User is logged out successfully * User receives confirmation of logout * Navbar returns to unauthenticated state * Username/profile is removed from navbar * Login and Sign Up buttons reappear | **✓** Pass if user is logged out and UI reflects unauthenticated state |
| **Test Case 4: Driver Schedules a Drive to a Street** | * Driver account exists in the system * Driver is logged in to the app * At least one street exists in the system | 1. Navigate to the "Schedule Drive" section 2. Select a target street from the list 3. Enter the scheduled time for the drive 4. Enter the starting location 5. Submit the drive schedule | * Drive is successfully created and saved to the database * Drive appears in the driver's schedule list * Drive details (street, time, location) are accurate * Residents on the scheduled street receive the notification in their inbox | **✓** Pass if drive is created, driver can view it in their schedule, and relevant residents receive inbox notifications |
| **Test Case 5:**  **Resident Views Inbox for Scheduled Drives** | * Resident account exists with an assigned street * Resident is logged in to the app * At least one drive has been scheduled for the resident's street | 1. Navigate to the inbox section 2. View list of scheduled drives for their street 3. Verify drive details (driver info, scheduled time, current location) 4. Filter inbox by specific street | * Inbox displays all scheduled drives for the resident's street * Each drive entry shows complete information (driver, time, location) * Only drives relevant to the resident's street are displayed * Inbox updates in real-time when new drives are scheduled | **✓** Pass if inbox correctly displays all relevant scheduled drives with accurate information |
| **Test Case 6: Resident Requests a Stop** | * Resident account exists and is logged in * At least one active drive is scheduled for the resident's street * Drive is available for stop requests | 1. View the inbox and select an active scheduled drive 2. Click "Request Stop" button for the selected drive 3. Confirm the stop request with street information 4. Submit the stop request | * Stop request is successfully created and linked to the specific drive * Stop request is saved to the database with correct resident\_id and drive\_id * Confirmation message is displayed to the resident * Driver receives notification of the stop request * Stop request appears in the driver's stop request list | **✓** Pass if stop request is created, saved correctly, and driver is notified |
| **Test Case 7: Resident Views Driver Status and Location** | * Resident account exists and is logged in * Driver is actively on a scheduled drive * Driver has a current status and location | 1. Navigate to the "Track Driver" section 2. Select a driver/drive from the list 3. View the driver's current status 4. View the driver's current location on a map 5. Refresh or monitor for real-time updates | * Driver's status is displayed accurately * Driver's current location is displayed correctly * Location updates in real-time * Map interface shows driver's position accurately | **✓** Pass if resident can view accurate, current driver status and location information |