

**UniRoutes**

Functional Requirements

1. Bus Company Management:

1.1 Admins can add, update, and delete bus companies.

1. Driver and Staff Management:

2.1 Admins can add, remove, and update driver and supervisor profiles.

2.2 Admins assigns drivers and supervisors to bus routes and groups them in chat rooms.

1. Bus Route and Stops Management:

3.1 Admins can add, remove, and update routes and bus stops, route.

3.2 Admins can set up temporary stations or change routes.

1. User Roles and Permissions:

4.1 The system provides role-based access for students, drivers, supervisors, and admins, with each role having specific privileges and access.

4.2 Roles can be assigned or updated based on user actions, like becoming a route supervisor or admin.

1. Advanced Subscription Management:

5.1 Users can view available seats, reserve seats, and rebook frequent trips.

5.2 The system allows booking via daily or semester subscription and provides reminders for upcoming reservations.

1. Notifications and Alerts:

6.1 Admins can send global, route-specific, bus-specific or personalized messages.

6.2 The system sends automatic alerts for bus delays, cancellations, or emergencies and weather-related notifications.

1. Real-Time Bus Tracking:

7.1 Students, Supervisors, and admin can view buses on a live map, showing current location, estimated arrival times, and any delays.

7.2 Admins can track all buses, while individual companies and students can track assigned buses only.

1. Emergency and Safety Features:

8.1 Students and Supervisors can send emergency alerts with location information to dispatchers and relevant staff.

8.2 Admins can deploy nearby buses for assistance and notify users of emergency plans via push notifications.

1. Feedback:

9.1 After each trip, users can submit feedback on the bus, driver, or route, with options for ratings and detailed comments.

9.2 Admins can view feedback to make service improvements, identify trending issues, and monitor driver performance.

10. Weather Monitoring and Alerts:

10.1 The system integrates weather APIs to monitor conditions affecting bus schedules.

10.2 Admins receive alerts about weather impacting routes

10.3 Users are notified of delays or schedule adjustments due to severe weather.

11. Automated Maintenance Scheduling:

11.1 Based on usage and feedback, the system automatically notifies admins of buses in need of maintenance.

11.2 Admins and bus companies can view bus maintenance history.

11.3 Admins and bus companies can book bus for maintenance and request additional inspections when needed.

12. Lost and Found Management:

12.1 Users can report lost items, entering details such as item description and bus number.

12.2 Admins or Supervisors can update the system if items are found, and users receive notifications to retrieve them.

13. Google Maps and Navigation Integration:

13.1 Integrated Google Maps provides real-time traffic updates and route navigation for users and drivers.

13.2 Users can view estimated arrival times, find optimal pickup points, and get directions to designated stops.

14. Payment Management:

14.1 Users can pay for daily or semester subscription, view transaction history, and handle refunds.

14.2 Admins can track payments, generate financial reports, and monitor spending patterns.

14.3 Admins can issue fines to Students for breaking rules or destruction of property.

15. User Suggestions and Voting System:

15.1 Users can submit suggestions for improvements or changes to routes, schedules, or features.

15.2 Other users can upvote suggestions

15.3 Admins can review and decide on implementing popular requests.

16. Offline Mode for Essential Information:

16.1 Users can access route maps, schedules, and ticket details offline, allowing them to plan trips or check reservations without internet access.

17. Help Center and Chatbot Support:

17.1 An AI-powered chatbot assists users with common questions, like finding routes, viewing schedules, and reporting issues.

17.2 The help center includes FAQs, live chat support, and email contact options for further assistance.

18. Trip History and Analytics:

18.1 Users can view a history of past trips, while admins and companies can analyze trip data for insights.

18.2 The system provides reports on usage patterns, peak times, and route performance.

19. QR Code Ticketing System:

19.1 Each bus ticket has a unique QR code that serves as a digital boarding pass, scannable by the bus supervisor.

20. Group Chat for Route Communication:

20.1 Chat rooms are created for each route, allowing supervisors and Students to.

21. Accessibility Features for Users with Disabilities:

21.1Users with disabilities can navigate the system using voice commands, and the interface supports text-to-speech for visually impaired users.

22.Balance Management :

22.1 User can access and top up his wallet to use it in his payments

23. Bus Management :

23.1 Admins can manage its fleet, including adding, updating, and removing buses, assigning buses to specific routes, and viewing bus availability.

24. Rating System:

24.1 Users can rate buses, driver, supervisors and their overall travel experience

Non-Functional Requirements:

1. Security:

1.1 Protection of students' personal data.

The system must encrypt all sensitive personal data (e.g., student names, phone numbers, and bus subscription details) both in transit and at rest.

1.2 Ensuring restricted access to data.

The system must ensure that only authorized personnel (e.g., admins and supervisors) can access bus-related information, such as schedules, routes, and student subscriptions.

2. Usability:

2.1 Easy system layout understanding for students.

The system must have a simple, user-friendly interface that is easy for students to navigate, requiring minimal instructions for new users.

2.2 Simplified bus subscription management.

The system must allow students to easily subscribe to the bus service (daily or semester basis) and review bus schedules with clear and straightforward steps.

3. Accessibility:

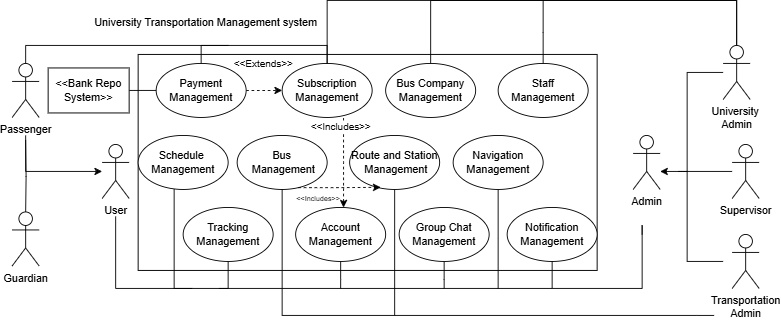
3.1 multi-device accessibility.

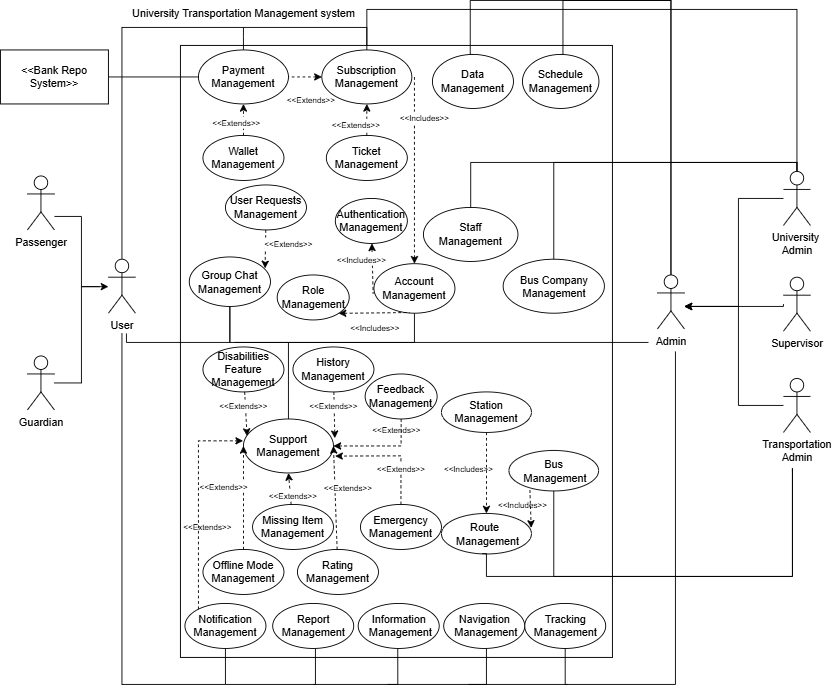
The system must be accessible from all devices, including computers (Windows, Mac OS, Linux), smartphones, and tablets, ensuring that students and staff can access the system from anywhere.

4. Availability:

4.1 High availability for student use.

The system must be available at least 99.9% of the academic year, ensuring students and staff can rely on it for managing bus schedules and subscriptions with minimal downtime.

**High Level Use Case**



**Subsystems**

Subsystems

|  |  |  |  |
| --- | --- | --- | --- |
|  | Subsystem | Subsystem Functions | Subsystem Interface |
| A | GPS system | -Tracks Real Time location of university buses  -Provides live maps showing buses, stations and estimate travel time | Interface GPS {  -public void track\_location(int bus\_id)  -public Map Live\_Map()  -public void Estimate\_Travel(int bus\_id, int Station\_id)} |
| B | Weather system | -Retrieves and displays current forecast conditions  -Provides Weather impact analysis on operations  -Sends alerts for severe weather | Interface Weather{  -public String Forecast()  -public Boolean Weather\_Warning()  -public void Display\_Weather\_Impact()} |
| C | Rating system | -Collects Feedback from passengers  -Evaluates integrity of ratings | Interface Raiting{  -public boolean collect\_feedback(int route\_id, int passenger\_id, double rating)  -public boolean evaluate\_raiting(int rating\_id)} |
| D | Maintenance system | -Tracks maintenance management with engineers  -Logs Maintenance to ensure bus safety | Interface Maintenance{  -public Boolean track\_maintenance(int bus\_id)  -public condition Bus\_condition(int bus\_id)} |
| E | Financial system | -Manages Budget for Transportation department  -Tracks Revenue  -Handles customer purchases | Interface Finances{  -public double budget()  -public double Revenue()  -public boolean purchase(int passenger\_id)} |
| F | Authentication system | -Provides secure login for students and staff  -Allows authenticated users to access certain features of the app according to their level | Interface Authentication{  -public boolean secure\_login(int user\_id)  -public boolean Authorized(int user\_id)  -public boolean Authorization\_level(int user\_id)} |
| G | Scheduling system | -Organizes transportation scheduling for buses  -Allocates Buses according to multiple factors  -Sends alerts to passengers for schedule changes | Interface Schedule {  -public boolean Organize\_shedule(int route\_id,DateTime time)} |
| H | Notification system | -Sends notifications to students and staff  -Provides real-time updates via emails or in app notifications | Interface Notifications{  -public boolean Send\_Notification(String Massage)  -public boolean RealTime\_Updates(String Massage)} |
| I | History-Data system | -Stores and retrieves data and trip history  -provides better analysis and visualization | Interface Data{  -public boolean Store\_data(int trip\_id)  -public Data Retrieve\_DATA(int trip\_id)  -public Analysis Perform\_Analysis(Data data)} |
| J | Booking system | -Allows the user to book his transportation plans  -Allows the user to book a specific seat | Interface Booking{  -public Booking book\_transportation(int user\_id, int route\_id, DateTime time)  -public Booking book\_seat(int user\_id, int seat\_number)} |
| K | Artificial intelligence system | -Uses ai to optimize route planning  -Predictive maintenance scheduling  -Improves user experience with chatbot  -Driver Behavior Analysis  -Classification for reporting system | Interface AI integration{  -public RoutePlan optimize\_route(int bus\_id)  -public boolean schedule\_predictive\_maintenance(int bus\_id)  -public Chatbot Chatbot(int user\_id)  -public Report analyze\_driver\_behavior(int driver\_id)} |
| L | Report system | -Allows users to report issues and requests  -Allows For report generation | Interface Reports{  -public boolean submit\_issue (int user\_id, String report)  -public Report generate\_report(int report\_type)} |
| M | Group Chat | -Create and manage chatrooms | Interface Chatroom{  -public ChatRoom create\_chat\_room(String room\_name)  -public boolean manage\_chat\_room(int room\_id)} |

**Traceability Matrix**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **GPS** | **Weather** | **Rating** | **Maintain** | **Finance** | **Auth** | **Schedule** | **Notify** | **Data** | **Booking** | **Artificial** | **Report** | **Chats** |
| **1-Comp Man.** |  |  |  | 🗸 | 🗸 |  |  | 🗸 | 🗸 |  |  | 🗸 |  |
| **2-Bus Man.** | 🗸 | 🗸 | 🗸 | 🗸 | 🗸 |  | 🗸 | 🗸 | 🗸 |  |  | 🗸 |  |
| **3-R/S Man.** | 🗸 | 🗸 | 🗸 |  |  |  | 🗸 | 🗸 | 🗸 |  | 🗸 | 🗸 |  |
| **4-Notification** | 🗸 | 🗸 | 🗸 | 🗸 | 🗸 | 🗸 | 🗸 | 🗸 | 🗸 | 🗸 | 🗸 | 🗸 | 🗸 |
| **5-Tracking** | 🗸 |  |  |  |  |  |  | 🗸 | 🗸 |  | 🗸 | 🗸 |  |
| **6-Alerts** | 🗸 |  |  |  |  | 🗸 |  | 🗸 | 🗸 |  | 🗸 | 🗸 | 🗸 |
| **7-About** |  |  |  |  |  | 🗸 |  |  | 🗸 |  |  |  |  |
| **8-Feedback** |  |  | 🗸 |  |  | 🗸 |  |  | 🗸 |  | 🗸 | 🗸 |  |
| **9-Staff Man.** |  |  |  |  |  | 🗸 |  |  | 🗸 |  |  |  | 🗸 |
| **10-Reports** |  |  | 🗸 | 🗸 |  |  |  |  | 🗸 |  | 🗸 | 🗸 |  |
| **11-Maintain** |  |  |  | 🗸 |  |  |  | 🗸 | 🗸 |  | 🗸 | 🗸 |  |
| **12-Emergency Response** | 🗸 | 🗸 |  | 🗸 |  | 🗸 |  | 🗸 | 🗸 |  |  | 🗸 |  |
| **13-Booking** |  |  |  |  | 🗸 | 🗸 | 🗸 | 🗸 | 🗸 | 🗸 | 🗸 |  |  |
| **14-Finance** |  |  |  |  | 🗸 |  |  |  | 🗸 |  |  | 🗸 |  |
| **15-Data Man.** |  |  | 🗸 | 🗸 | 🗸 | 🗸 | 🗸 |  | 🗸 | 🗸 | 🗸 | 🗸 | 🗸 |
| **16-Weather** |  | 🗸 |  |  |  |  |  | 🗸 |  |  |  |  |  |
| **17-Rating** |  |  | 🗸 |  |  |  |  |  | 🗸 |  |  | 🗸 |  |
| **18-Lost/Found** |  |  |  |  |  |  |  | 🗸 | 🗸 |  |  | 🗸 |  |
| **19-Offline** |  |  |  |  |  |  | 🗸 | 🗸 | 🗸 |  |  |  |  |
| **20-GPS** | 🗸 | 🗸 |  |  |  |  | 🗸 | 🗸 | 🗸 |  |  |  |  |
| **21-Disapilities** |  |  |  |  |  |  |  |  | 🗸 |  |  |  |  |
| **22-User Req.** |  |  |  |  |  |  |  | 🗸 | 🗸 |  |  | 🗸 | 🗸 |
| **23-Heatmap** | 🗸 |  |  |  |  |  |  |  | 🗸 |  | 🗸 | 🗸 |  |
| **24-Auth** |  |  |  |  |  | 🗸 |  |  | 🗸 |  |  |  |  |
| **25-Role Man.** |  |  |  |  |  |  |  |  | 🗸 |  |  |  |  |
| **26-Schedule** |  |  |  |  |  | 🗸 | 🗸 |  | 🗸 |  | 🗸 | 🗸 | 🗸 |
| **27-Favorites** |  |  |  |  |  | 🗸 | 🗸 |  | 🗸 |  |  |  |  |
| **28-Ai** |  |  |  |  |  |  |  |  | 🗸 |  | 🗸 | 🗸 |  |
| **29-QR CODE** |  |  |  |  | 🗸 |  |  |  | 🗸 | 🗸 |  | 🗸 |  |
| **30-Help** |  |  |  |  |  |  |  |  | 🗸 |  |  |  |  |
| **31-Wallet** |  |  |  |  | 🗸 | 🗸 |  |  | 🗸 |  |  | 🗸 |  |

Use Case Description

Ahmed Islam Farouk Abbas 22101008

**Use Case: Payment Management**

**Actors:** User(Passenger), University Admin, Banking System

**Type:** Primary and Essential

**Description:** The User logs into the system and navigates to Purchase a subscription (Whether daily, weekly, monthly or by semester) and then navigates to the payment section in order to complete the purchase. The User can choose from various payment methods like credit cards, PayPal, master cards, Vodafone cash, instapay or through the payment system wallet. Upon choosing their payment method the system directs them to a secure gateway system for secure transactions.

The payment system processes the transaction with the Banking system and the university admins depending on the user’s chosen method. Once payment is confirmed the system updates the subscription status then the system issues a subscription confirmation and payment confirmation through email and notification. Also Generating a virtual pass in the shape of a QR code that the user can use to confirm his subscription with the supervisors

**Preconditions:**

* + 1. The user must be logged in and authenticated into the system
    2. The user must have an active subscription ready to confirm
    3. The payment system and banking system must be online and running without issues and available to support chosen payment method

**Postconditions:**

* 1. The user subscription state is confirmed or pending
  2. The user receives the confirmation email if the payment is successful and his QR code

**Alternate Scenarios**

* + 1. The user has insufficient funds to the payment doesn’t go through
    2. An error massage pops up if the system is down or the user got timed out and payment can’t go through telling the user to try again later
    3. The user’s account is blocked or on hold not allowing him to subscribe to a plan

|  |  |
| --- | --- |
| **Interaction Scenario** | **System Responsibilities** |
| 1-User logs into the tansportation system | 2-Authenticates User and displays the dashboard |
| 3-User reviews their subscription details then initiates the payment process | 4-Retrives the subscription details to display for user checking the availability of the subscription and provides a proceed to payment option |
| 5-The user clicks on proceed payment | 6-Establishes secure payment environment and redirects the user to the payment gateway |
| 7-The user chooses the preferred payment method | 8-Display payment options and enables the selection of it also calculating the transaction amount |
| 9-The user proceeds with the payment details | 10-Processes payment details and verifies transaction with the banking system |
| 11-User completes Payment and subscription is made | 12-Recives the confirmation updating the status of the subscription generating a confirmation receipt |
| 13-Confirmation email and notification is sent to the user also accessing the QR code that was generated | 14-Sends the user their confirmation through email and notifications and also sends them their QR code |

**Alternative Courses**

**Step 1:** If the user’s login attempt fails the system displays an error massage

**Step 8:** If a specific payment method is temporarily unavailable the system notifies the user and shows only the available options

**Step 10:** If payment fails due to issues like insufficient funds or declined transaction the user is notified immediately

**Test Requirements**

1-Validate that the user is Authenticated before allowing him to access subscriptions and payment method

2-Validate that if the account is blocked or on hold the user cannot proceed with any transactions

3-validate that the user was redirected to a secure payment gateway

4-Validate that the system has only the available payment options available for the user

5-Validate that the user has valid credentials according to the chosen payment method

6-Validate that payment is rejected if the user doesn’t have valid credentials or sufficient funds

7-Validate that payment process is valid for processing if user has valid credentials and sufficient funds

8-Validate Accurate Deduction for Payment Amount Confirm that the transaction amount calculated matches the subscription’s pricing including any discounts or applicable taxes.

9-Validate that the system only updates the subscription status to confirmed after successful payment

10-Validate that the user receives his receipt and his email notification

11-Validate that the user receives their QR code

12-Validate that the system logs all transactions with appropriate details

13-Validate Data Security ensure that all user data and transaction details are encrypted and no sensitive payment information is stored on the system after processing.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Case ID** | **Test Scenario** | **Test Steps** | **Test Data** | **Expected Result** |
| TC\_PAY\_1\_1 | Validate Authentication with valid credentials | 1-Open Login/Signup page 2-Use University ID 3-Use university email and password | ID: validID email:Valid Email password:validpass | 1-Set Auth to approved 2-Payment Authorized |
| TC\_PAY\_1\_2 | Validate Authentication with invalid credentials | 1-Open Login/Signup page 2-Use University ID 3-Use invalid university email and password | ID: validID email:INVALID Email password:validpass | 1-Set Auth to a declined 2-Payment UnAuthorized |
| TC\_PAY\_2\_1 | Validate User Account is not blocked | 1-Check Account status | Status : Access granted | 1-Payment Authorized |
| TC\_PAY\_2\_2 | Validate User Account is blocked | 1-Check Account status | Status : Access blocked | 1-Payment UnAuthorized |
| TC\_PAY\_3\_1 | Validate Secure Gateway | 1-User picks a subscription plan and proceeds to pay 2-User redirected to payment page | Valid Payment Gateway details | 1-User directed to secure payment gateway |
| TC\_PAY\_4\_1 | Validate Payment Options | 1-Check avaliable payment options 2-show only avaliable options | Valid Payment options details | 1-User was shown the  avaliable payment methods |
| TC\_PAY\_5\_1 | Validate User Payment with invalid Credentials | 1-Entre invalid Credentials  according to each  payment method | inValid Credentials | User payment rejected |
| TC\_PAY\_5\_2 | Validate User has insufficent Funds | 1-Check if user has enough funds | Insufficent Funds | User payment rejected |
| TC\_PAY\_6\_1 | Validate User Payment with valid Credentials | 1-Entre valid Credentials  according to each  payment method | Valid Credentials | User payment Accepted |
| TC\_PAY\_6\_2 | Validate user has  sufficent funds | 1-Check if user has enough funds | sufficent funds | User payment Accepted |
| TC\_PAY\_7\_1 | Validate Accurate Deduction | 1-Check subscription amount 2-Check User Funds | 1-Subscription value  2-User Funds | Accurate Deduction  from User Funds |
| TC\_PAY\_8\_1 | Validate Subscription  update post payment | 1-Check for payment  confirmation  2-Check For subscription  status | 1-Payment Confirmation 2-Subscription status | All Subscriptions  were updated |
| TC\_PAY\_9\_1 | Validate User recives  reciept | 1- Send confirmation  to the user | Users Email | Confirmation was sent |
| TC\_PAY\_10\_1 | validate QR code receival | 1-send qr code to user | user email | qr code was sent |
| TC\_PAY\_11\_1 | Validate Logs | 1- Process Payment  2-Update Logs | Payment Details | Transaction has been saved |

Belal Fathy 22101311

**Actor(s):** Student, Admin, Bank System

**Type:** Primary and Essential

**Description:**This use case enables authenticated students to select a subscription plan (either daily subscribe or semester subscribe and process payments securely through an integrated payment gateway. After logging in, the student can view available subscription options configured by the Admin. Once a subscription type is selected, the student proceeds to the payment gateway, where they can choose from various payment methods, such as credit cards, debit cards, or other supported payment services, depending on the Bank System’s integration. Upon successful payment, the system activates the selected subscription, updates the student’s subscription status, and records the transaction details. Admins manage the subscription options, configuring and modifying them as needed. They also handle payment records, monitor transactions, and generate detailed financial reports on subscription usage and payments to ensure accurate tracking of subscription metrics.After a successful transaction, the system provides the student with a confirmation notification and receipt through email. The admin can also generate reports based on the subscription records and payment data to assess usage patterns, manage finances, and adjust subscription offerings as required.

**Preconditions:**

* The student must be authenticated within the system.
* Subscription options (daily or semester) must be configured by the Admin.
* Payment processing integration with the Bank System must be active.

**Postconditions:**

* The selected subscription (daily or semester) is activated for the student upon successful payment.
* Subscription records and payment status are updated in the system.
* Reports on subscription usage and payment transactions are generated for the Admin.

**Alternative Scenarios:**

* **Insufficient Funds**: If the student has insufficient funds or a payment is declined, an error message will inform the student, prompting them to try a different payment method or check their balance.
* **Bank System Down**: If the Bank System or payment gateway is temporarily unavailable, the student receives a notification to try again later, and the transaction remains pending until resolved.
* **Subscription Option Unavailable**: If an Admin temporarily disables a subscription option (e.g., for system maintenance), the student is notified and only enabled options are shown.
* **Student Account Issue**: If the student’s account is suspended or flagged, they cannot proceed with a subscription, and a message informs them to contact support for resolution.
* **Partial Payment Support**: If the student attempts a payment with a payment method not fully supported (e.g., installment payments if offered in the future), they are informed that only full payments are accepted at this time.

**Test Requirements:**

* Validate that students can view and choose between daily and semester subscription options.
* Validate that the system displays the chosen subscription details correctly.
* Validate that the payment processing is compatible with the bank system.
* Verify that the subscription will be activated after payment.
* Validate that the student can pay with different payment methods.
* Validate that the admin can add subscription option.
* Validate that the admin can modify subscription option.
* Validate that the admin can delete subscription option.
* Validate that the admin can generate financial reports.
* Validate that the generated reports cover all details.
* Validate that only authenticated Students can view subscription options.
* Validate that unauthorized Students can not view subscription options.

|  | **Alternative Scenario** | **System Response** |
| --- | --- | --- |

|  |  |
| --- | --- |
| **1.Login Failure**: Student login attempt fails due to incorrect credentials. | 2.Displays an error message prompting the student to re-enter correct login details. |

|  |  |
| --- | --- |
| **3.Unavailable Payment Method**: The selected payment method is temporarily unavailable. | 4.Notifies the student of unavailability and displays only active, available payment methods. |

|  |  |
| --- | --- |
| **5.Insufficient Funds**: The student’s payment is declined due to insufficient funds. | 5.Shows a notification indicating insufficient funds and prompts the student to try a different payment method or check their balance. |

|  |  |
| --- | --- |
| **6.Bank System Unavailable**: The Bank System or payment gateway is temporarily down. | 7.Displays a message indicating that payment processing is temporarily unavailable and advises the student to try again later. |

|  |  |
| --- | --- |
| **8.Subscription Option Disabled**: Admin has disabled the selected subscription option. | 9.Informs the student that the option is currently unavailable and shows only active subscription options. |

|  |  |
| --- | --- |
| **10.Account Issue**: Student’s account is suspended or flagged, preventing subscription. | 11.Notifies the student of the issue and advises them to contact support for resolution. |

|  |  |
| --- | --- |
| **12.Subscription Activation Failure**: System error prevents activation of subscription after payment. | 13.Sends a notification to the student about the issue and provides support contact information for assistance. |

|  |  |
| --- | --- |
| **14.Partial Payment Not Supported**: Student attempts partial payment with unsupported method (e.g., installment). | 15.Informs the student that only full payment options are supported at this time and prompts selection of a full payment method. |

**Alternative Courses**

* **Step 1**: If the student's login attempt fails, the system displays an error message.
* **Step 4**: If a specific payment method is unavailable, the system notifies the student and shows only active options.
* **Step 5**: If payment fails due to insufficient funds or a declined transaction, the system notifies the student immediately.

**Step 6**: If the subscription fails to activate due to a system error, the student is notified, and support contact information is provided

**Mazen Ahmed Samir 22100369**

**Use Case: Route Management**

* **Actors**:
  + **Primary Actor**: Administrator
  + **Secondary Actor**: Bus Company, Moderator
* **Description**:

The Admin or User accesses the system to manage Bus Routes and Stops.

, The system access Bus Routes and Stops data from database and displays it.

, The system allows the admin to add, edit, or remove Bus Routes and Stops and allows the User to View Bus Routes and Stops.

, The admin selects an option and submits the request.

, The system processes the request and updates the bus company details.

, The admin receives confirmation of the successful update.

* **Cross-References**:
  + Related to **Station Management**, **Trip Management**, and **Real-Time Tracking Management**.
* **Preconditions**:
  + Administrator must be logged in with permissions to manage route information.
* **Postconditions**:
  + Route details are updated in the system, enabling proper scheduling, tracking, and route assignment for buses.

| **Actor Intentions** | **System Responsibility** |
| --- | --- |
| 1. Administrator logs in with permissions to manage routes. | 2. System authenticates the Administrator and grants access to Route Management. |
| 3. Administrator creates a new route or selects an existing route to update. | 4. System displays route details and available buses for assignment. |
| 5. Administrator assigns buses and specifies stops, schedules, and other route details. | 6. System validates route information and checks for conflicts. |
| 7. Administrator confirms route details to finalize updates. | 8. System saves the route, updates schedules, and notifies relevant users if needed. |

**Alternative Courses**

* **Step 6**: If there is a scheduling or capacity conflict, the system alerts the Administrator to resolve it.
* **Step 5**: If there are insufficient buses for the route, the system prompts the Administrator for alternate schedules or bus assignments.
* Test Requirements:
* 1. Validate that admins can successfully add a new bus route.
* 2. Validate that admins can remove an existing route from the system.
* 3. Validate that admins can edit the details of an existing route.
* 4. Validate that admins can add bus stops for any route.
* 5. Validate that admins can modify bus stops for any route.
* 6. Validate that admins can remove bus stops for any route.
* 7. Validate that users can view all available bus routes.
* 8. Validate that users can view all stops associated with a selected route.
* 9. Admin and User can Search Bus Routes
* 10. Admin and User can Search Bus Stops within selected bus route

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Case ID** | **Test Scenario** | **Test Steps** | **Test Data** | **Expected Result** |
| Add\_BR01 | Validate that admins can successfully add a valid bus route. | 1. Login as admin 2. Select to "Add Route" 3. Enter route details 4. Click Confirm | 1.Route to be added 2.Valid bus route info 3.Admin account | New Bus Route added and  confirmation message  displayed |
| Add\_BR02 | Validate that admins can't add an invalid bus route. | 1.select "add bus route" 2.enter invalid bus info 3.click confirm | 1.Route to be added 2.Invalid bus info 3.Admin account | Bus Route isn't added and error message is displayed |
| Remove\_BR03 | Validate that admins can remove an existing route from the system. | 1. Login as admin 2. Select an existing route 3. Click "Delete" 4. Click Confirm | 1.Route ID: 101 2.Admin Account | Bus Route removed and  confirmation message  displayed |
| Remove\_BR04 | Validate that admins can remove an existing route from the system. | 1.select existing Bus Route 2.select "Remove Bus Route" 3.click confirm | 1.Admin account 2.Rounte ID: 110 | Bus Route removed and  confirmation message  displayed |
| Edit\_BR05 | Validate that admins can edit the details of an existing route. | 1. Login as admin 2. Select an existing route 3. Edit Bus Routes details  4. Click Confirm | 1.New Route details 2.Admin account 3.Rounte ID: 109 | Route details are updated in  the system |
| Edit\_BR06 | Validate that admin can't edit route with an invalid route ID | 1. Login as admin 2. Select an existing route 3. Edit Bus Routes details  4. Click Confirm | 1.New Route details 2.Admin account 3.Rounte ID: -109 | Error message displayed: "Route not found" |
| Add\_BS07 | Validate that admins can add bus stops for any route. | 1. Login as admin 2. Select route 3. Add new stops 4.Click Confirm | 1.Route ID: 108 2.bus stop name: deep mall 3.Admin account | New stop is added to the  selected route |
| Add\_BS08 | Validate that admins can't add duplicate bus stops for any route. | 1. Login as admin 2. Select route 3. Add new stops 4.Click Confirm | 1.Route ID: 108 2.bus stop name: deep mall 3.Admin account | Error message displayed: "Stop already exists" |
| Add\_BS09 | Validate that admins can't add bus stops for any route with missing stop name. | 1. Login as admin 2. Select route 3. Add new stops 4.Click Confirm | 1.Route ID: 108 2.bus stop name: 3.Admin account | Error message displayed:  "Stop name is required" |
| Edit\_BS10 | Validate that admins can modify bus stops for any route | 1. Login as admin 2. Select route 3. Select stop 4. Edit stop details and save | 1.Route ID: 106 2 stop ID : 206 3.Updated stop name: gleem 3.Admin account | Stop details are updated |
| Edit\_BS11 | Validate that admins can't edit bus stops for any route with invalid stop name. | 1. Login as admin 2. Select route 3. Select stop 4. Edit stop details and save | 1.Route ID: 106 2 stop ID : 206 3.Updated stop name: @#$%^ 3.Admin account | Error message displayed:  "Invalid stop name" |
| Remove\_BS12 | Validate that admins can remove bus stops for any route | 1. Login as admin 2. Select route 3. Select stop 4. Click "Delete" and confirm | 1.Route ID: 105 2. stop ID: 201 3.Admin account | Selected stop is removed  from the route |
| Remove\_BS13 | Validate that admins can't remove bus stops for any route with invaid ID. | 1. Login as admin 2. Select route 3. Select stop 4. Click "Delete" and confirm | 1.Route ID: 105 2. stop ID: -201 3.Admin account | Error message displayed: "Stop not found" |
| View\_BS14 | Validate that users can view all stops associated with a route | 1. Login as user 2. Select route 3. View associated stops | 1.User account 2.Route ID: 102 | All stops for the selected route are displayed |
| View\_BR16 | Validate that users can view all available bus routes | 1. Login as user/admin 2. Navigate to "View Routes" section | 1. User/admin account | All available routes are  displayed |
| View\_BR17 | Validate viewing routes with no routes available | 1. Login as user/admin 2. Navigate to "View Routes" section | 1. User/admin account | Error message displayed: "No routes available" |
| Search\_BR18 | Validate Admin and User can search bus routes | 1. Login as admin/user 2. Use "Search Route" feature 3. Enter search criteria | 1. User/ admin account 2.search criteria | Routes matching search criteria are displayed |
| Search\_BR19 | Validate Admin and User can't search bus routes with empty search criteria | 1. Login as admin/user 2. Use ""Search Route"" feature | 1. User/ admin account | Error message displayed: "Please enter search criteria" |
| Search\_BS20 | Validate Admin and User can search stops within a bus route | 1. Login as admin/user 2. Select route 3. Use "Search Stop" within selected route 4. Enter search criteria | 1. User/ admin account 2.search criteria 3.route ID: 107 | Stops matching search criteria are displayed |
| Search\_BS21 | Validate Admin and User can't search bus stop with empty search criteria | 1. Login as admin/user 2. Select route 3. Use "Search Stop" within selected route | 1. User/ admin account 2.route ID: 107 | Error message displayed:  "Stop not found" |
|  |  |  |  |  |

**Use Case: Account Management Eyad Elnakib 22100757**

* **Actors**:
  + **Primary Actor**: Passenger
  + **Secondary Actors**: Administrator, Moderator
* **Description**:
  + Manages user accounts within the system, including creating, updating, and deleting accounts, as well as managing permissions and access levels.
* **Cross-References**:
  + Related to **Authentication Management** and **Role Management**.
* **Preconditions**:
  + User must have valid credentials and appropriate permissions for account actions (e.g., Admin or Moderator for account creation and role management).
* **Postconditions**:
  + User account details are updated, and changes are reflected in other relevant areas (e.g., permissions, chat access).

| **Actor Intentions** | **System Responsibility** |
| --- | --- |
| 1. Admin or Moderator logs into the system with permissions. | 2. System authenticates the user and grants access to the Account Management module. |
| 3. Admin or Moderator selects a user account to create, update, or delete. | 4. System retrieves or displays relevant account details for editing. |
| 5. Admin or Moderator enters or edits account information (e.g., name, role, permissions). | 6. System validates and updates the account details in the database. |
| 7. Admin or Moderator assigns or modifies user permissions. | 8. System applies and saves permissions to the account. |
| 9. Admin or Moderator confirms changes to finalize the action. | 10. System confirms the action, logs it, and reflects changes in connected modules. |

**Alternative Courses**

**Step 2**: If the Admin or Moderator lacks sufficient permissions, the system displays an error message and restricts access.

**Step 7**: If the account is inactive, the system prompts the Admin or Moderator for reactivation options.

**Test Requirements**

1. **Validate Account Creation**
   * Validate that the system allows a new user to create an account by filling out all required fields, including Name, Email, and Password.
   * Upon successful creation, validate that the system sends a confirmation to the new user.
2. **Validate Profile Update**
   * Validate that the system allows a logged-in user to update their profile details, such as Name and Address.
   * Validate that changes are reflected immediately after saving.
3. **Validate Account Deletion**
   * Validate that the system provides an option for a logged-in user to delete their account.
   * Validate that once deleted, the account no longer exists, and the user is unable to log in with that account.
4. **Validate Viewing of Account Details**
   * Validate that the system allows a logged-in user to view their account details.
   * Validate that the displayed details match the data stored in the database for the specific user ID.
5. **Validate Account Creation with Missing Fields**
   * Validate that the system checks the input on the signup page to ensure all required fields are completed.
   * If any required fields are missing, validate that an error message is displayed, indicating the missing fields.
6. **Validate Profile Update with Invalid Data (Email)**
   * Validate that the system checks the format of the email address when updating the profile.
   * If an invalid email format is entered, validate that an error message "Invalid email format" is displayed.
7. **Validate Update of User ID**
   * Validate that the system allows a logged-in user to update their User ID.
   * Validate that the new User ID is unique and in a valid format.
   * Upon successful update, validate that a success message is displayed.
8. **Validate Update of User ID with Invalid Data**
   * Validate that the system checks the format of the User ID upon updating.
   * If an invalid User ID format is entered, validate that an error message "Invalid User ID format" is displayed.
9. **Validate Update of Phone Number**
   * Validate that the system allows a logged-in user to update their phone number.
   * Validate that the new phone number is saved and reflected in the user's profile.
10. **Validate Update of Phone Number with Invalid Data**
    * Validate that the system checks the phone number format upon updating.
    * If an invalid phone number format is entered, validate that an error message "Invalid phone number format" is displayed.
11. **Validate Update of Station**
    * Validate that the system allows a logged-in user to update their assigned station.
    * Validate that the station information is updated successfully and displayed correctly.
12. **Validate Update of Station with Invalid Data**
    * Validate that the system checks the station information format upon updating.
    * If invalid station data is entered, validate that an error message "Invalid station format" is displayed.
13. **Validate Update of Email**
    * Validate that the system allows a logged-in user to update their email address.
    * Upon entering a valid email, validate that the change is saved successfully and the new email is reflected in the user's profile.
14. **Validate Update of Email with Invalid Format**
    * Validate that the system checks the format of the email address upon updating.
    * If an invalid email format is entered, validate that an error message "Invalid email format" is displayed.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Case ID** | **Test Scenario** | **Test Steps** | **Test Data** | **Expected Result** |
| #-tc\_add\_01 | Validate creation of a new user account | 1. Navigate to the signup page 2. Enter user details 3. Submit form | Name, Email, Password | Account is created, and user receives confirmation |
| #-tc\_add\_02 | Validate update of user profile information | 1. Log in 2. Navigate to profile page 3. Update details and save | New Name, New Address | Profile is updated and reflects new details |
| #-tc\_add\_03 | Validate deletion of user account | 1. Log in 2. Navigate to delete account 3. Confirm deletion | Existing User ID | Account is deleted, and user can no longer log in |
| #-tc\_add\_04 | Validate viewing of user account details | 1. Log in 2. Navigate to account details page | Existing User ID | Correct user details are displayed |
| #-tc\_add\_05 | Validate error when creating an account with missing fields | 1. Navigate to signup page 2. Submit form without entering all required fields | Incomplete information | Error message indicating required fields are missing |
| #-tc\_add\_06 | Validate error when updating profile with invalid data | 1. Log in 2. Navigate to profile page 3. Enter invalid email format 4. Submit | Invalid email format | Error message "Invalid email format" is displayed |
| #-tc\_add\_07 | Validate successful update of User ID | 1. Log in 2. Navigate to profile page 3. Update User ID 4. Save | Valid New User ID | User ID is updated successfully |
| #-tc\_add\_08 | Validate error when updating User ID with invalid data | 1. Log in 2. Navigate to profile page 3. Enter invalid User ID format 4. Save | Invalid User ID format | Error message "Invalid User ID format" is displayed |
| #-tc\_add\_09 | Validate successful update of Phone Number | 1. Log in 2. Navigate to profile page 3. Update Phone Number 4. Save | Valid New Phone Number | Phone Number is updated successfully |
| #-tc\_add\_10 | Validate error when updating Phone Number with invalid data | 1. Log in 2. Navigate to profile page 3. Enter invalid Phone Number format 4. Save | Invalid Phone Number format | Error message "Invalid phone number format" is displayed |
| #-tc\_add\_11 | Validate successful update of Station | 1. Log in 2. Navigate to profile page 3. Update Station 4. Save | Valid New Station | Station is updated successfully |
| #-tc\_add\_12 | Validate error when updating Station with invalid data | 1. Log in 2. Navigate to profile page 3. Enter invalid Station data 4. Save | Invalid Station data | Error message "Invalid station format" is displayed |
| #-tc\_add\_13 | Validate successful update of Email | 1. Log in 2. Navigate to profile page 3. Update Email 4. Save | Valid New Email | Email is updated successfully |
| #-tc\_add\_14 | Validate error when updating Email with invalid format | 1. Log in 2. Navigate to profile page 3. Enter invalid Email format 4. Save | Invalid Email format | Error message "Invalid email format" is displayed |

Use Case: Notification Management Abdallah Basem Abdallah Zain 22100848

Actors: Admin, User

Type: Primary (initiator), and essential

Description: The system sends timely notifications to the user according to the preference of the customer either via e-mail or with in-app messaging. Examples include confirmation of booking, reminders of reservations, and related updates. Likewise, the application also extends notices for support needs and other interactions like responding to customer queries, notification of lost and found items, emergency notifications, feedback and rating requests, booking confirmations, response to user specific requests.

Preconditions:

- The customer must be authenticated.

- The customer must have enabled notification preferences in the system.

- Triggering events or updates are recorded in the system.

Postconditions:

- The customer receives the notifications.

Alternate Scenarios:

- If the delivery via the main channel should, for any reason, fail, the system tries to deliver by a different means.

- if the system fails to send the confirmation notification, it logs the error and tries to send the message after some short time. After trying three times and failing, the system alerts for customer support.

- In case the customer has already disabled notifications, the system will skip sending and update only the Notification History Log.

Interaction Scenarios

|  |  |
| --- | --- |
| **Actor Intentions** | **System Responsibility** |
| 1. Event triggers a notification | 1. The system verifies the user's notification preferences. |
| 1. System prepares notification content | 1. Sends the notification with details like booking confirmation, reminders, or alerts |
| 1. User receives the notification tracking and troubleshooting purposes | 1. 6. Logs the notification status in the database for tracking and troubleshooting purposes. |
| 1. User reviews notification content | 1. Ensures data integrity and accuracy across all sent notifications. |

Alternative Courses

- 2: If the primary notification channel is unavailable, the system switches to an alternative channel.

- 4: If the notification fails to send, the system prompts the customer to check notification settings and preferences.

- 8: If the customer identifies incorrect information or a discrepancy, they can report it to customer support for correction.

User Requirement:

Notifications: Users should receive timely notifications relevant to their reservations and other activities.

Test Requirements:

1. Verify that the user receives notifications through all configured channels [email and app].

2. Verify data accuracy across notification content and system records.

3. Verify notifications are sent immediately after the triggering event.

4. Verify data consistency between app notifications and email.

5. Verify that all relevant details are included in each notification.

6. Verify notifications are directed to the correct email address or account.

7. Verify there are no duplicate notifications for the same event.

8. Verify system retry logic works as expected after a failed notification attempt.

9. Verify user can disable notifications and that the system respects this setting.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Case ID | Test Scenario | Test Steps | Test Data | Expected Result |
| NOTIF\_1\_1 | Verify notifications through all channels | 1.auth user account.  2. Trigger notification event. 2. Check if notification is sent via email. 3. Check if notification is sent via app. | Email and app channels configured | Notification received via both email and app |
| NOTIF\_2\_1 | Verify data accuracy in notification | 1. Trigger notification event 2. Compare notification content with system record | System data for notification content | Notification content matches system record |
| NOTIF\_3\_1 | Verify immediate notification | 1. Trigger notification event 2. Check timestamp of sent notification | Event timestamp | Notification sent immediately after event |
| NOTIF\_4\_1 | Verify data consistency between channels | 1. Trigger notification event 2. Compare app and email notification content | App and email notifications | App and email notification content match |
| NOTIF\_5\_1 | Verify inclusion of all relevant details | 1. Trigger notification event 2. Review notification content | Notification content structure | Notification contains all relevant details |
| NOTIF\_6\_1 | Verify correct recipient address/account | 1. Trigger notification event 2. Check email and app account used for notification | User email and app account details | Notification sent to correct email and app account |
| NOTIF\_7\_1 | Verify no duplicate notifications | 1. Trigger notification event 2. Monitor system for duplicate notifications | Notification triggering event | No duplicate notifications for the same event |
| NOTIF\_8\_1 | Verify retry logic for failed notifications | 1. Trigger notification event 2. Simulate failure in sending 3. Verify retry attempt | Simulated notification failure | Notification sent successfully after retry |
| NOTIF\_9\_1 | Verify disable notifications functionality | 1. User disables notifications 2. Trigger notification event 3. Check if notification is sent | User notification settings staue | No notification sent when user has disabled them |

Use Case: Bus Management Mahmoud Eid 22100680

Actors:

• Admin (Initiator)

• User

Type:

• Primary and Essential

Description:

1. The Admin or User accesses the system to manage buses.

2. The system accesses bus data from the database and displays it.

3. The system allows the Admin to add, edit or remove buses, and allows the User to

view bus details.

4. The Admin selects an option and submits the request.

5. The system processes the request and updates the bus details.

6. The Admin receives confirmation of the successful update.

Postconditions:

1. The bus details are successfully updated.

Preconditions:

1. The User or Admin must be logged in.

Alternate Scenarios:

• Step 1: If the Admin's or User's session expires, they are prompted to log in again.

• Step 2: If no buses are available for management or viewing, the system displays a

message indicating this.

• Step 4: If there is an issue with the update, the system displays an error message.

Functional Requirements:

• Bus Management:

o Admins can add, remove, and update buses.

o Users can view bus details.

o Admins can assign buses to routes.

o Admins can track bus locations in real-time.

Test Requirements:

1. Validate that admins can successfully add a new bus.

2. Validate that admins can remove an existing bus from the system.

3. Validate that admins can edit the details of an existing bus.

4. Validate that admins can assign a bus to a specific route.

5. Validate that admins can track the real-time location of a bus.

6. Validate that users can view the details of a bus.

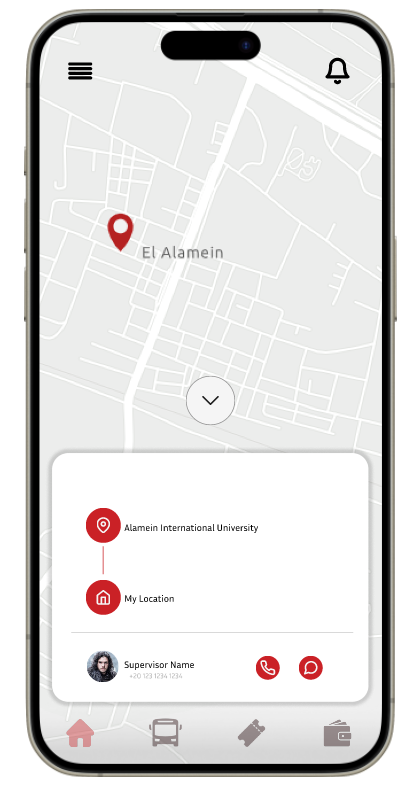
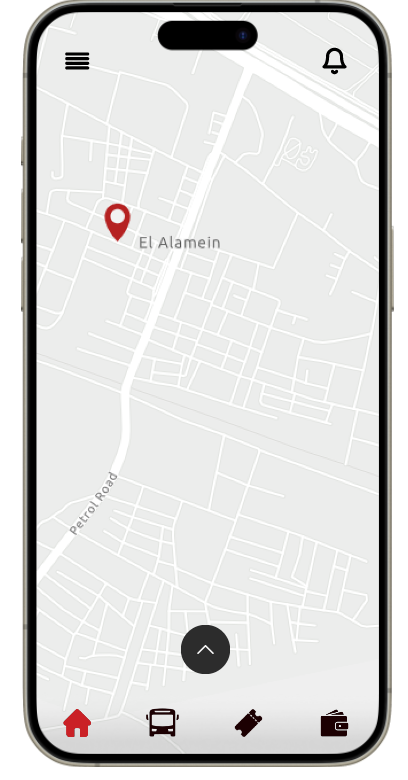
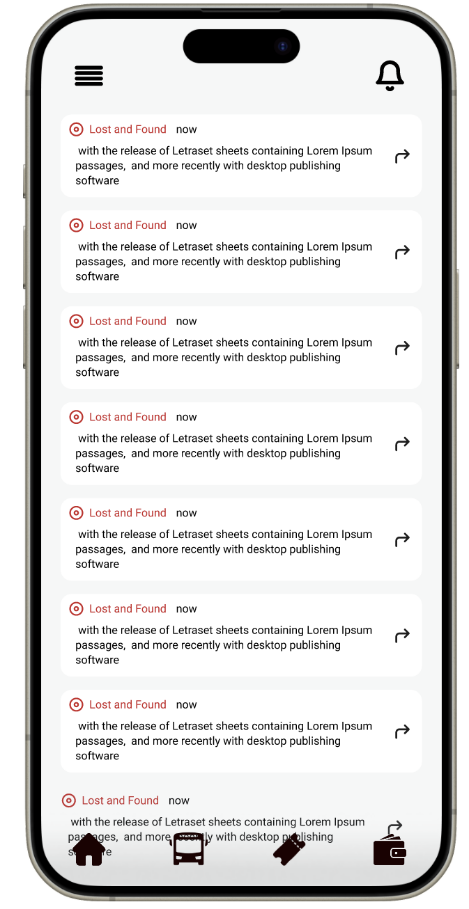
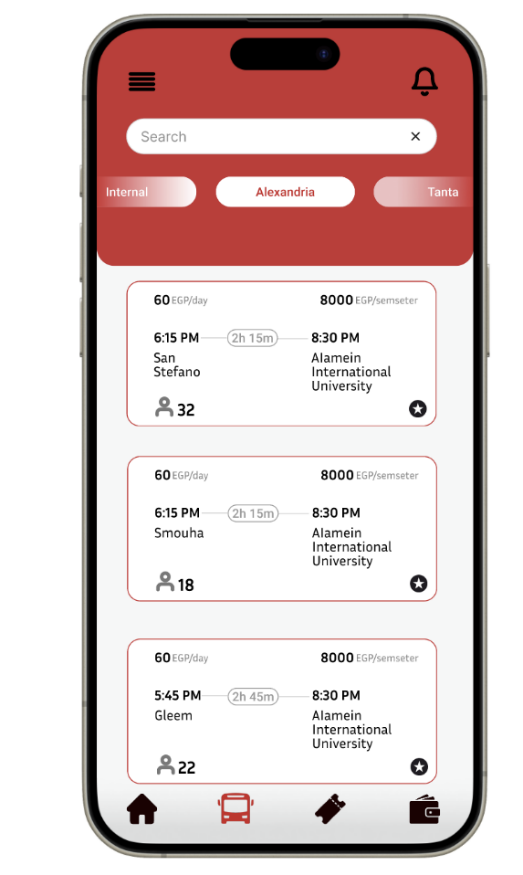
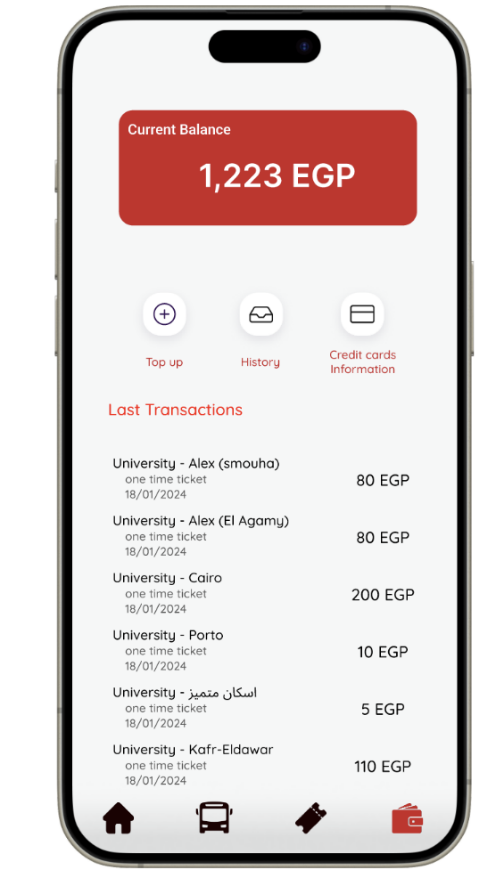
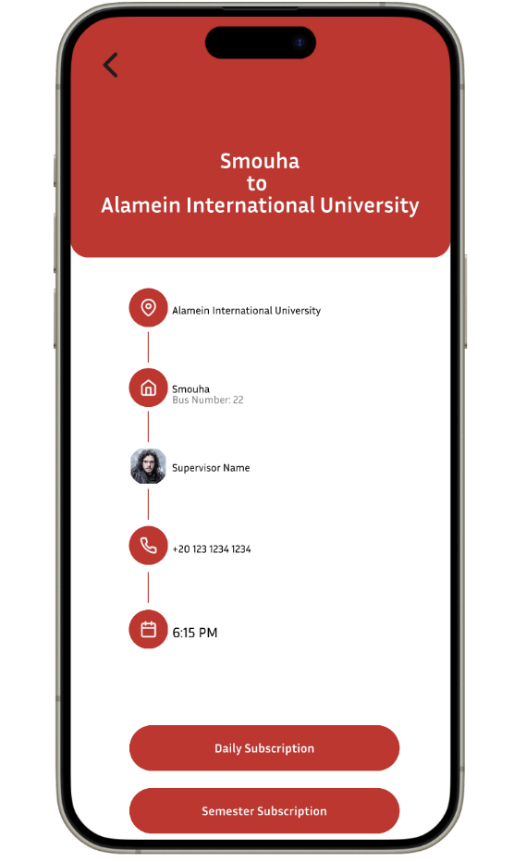
7. Validate that users can view the route assigned to a bus.

8. Validate that admins and users can search for buses.

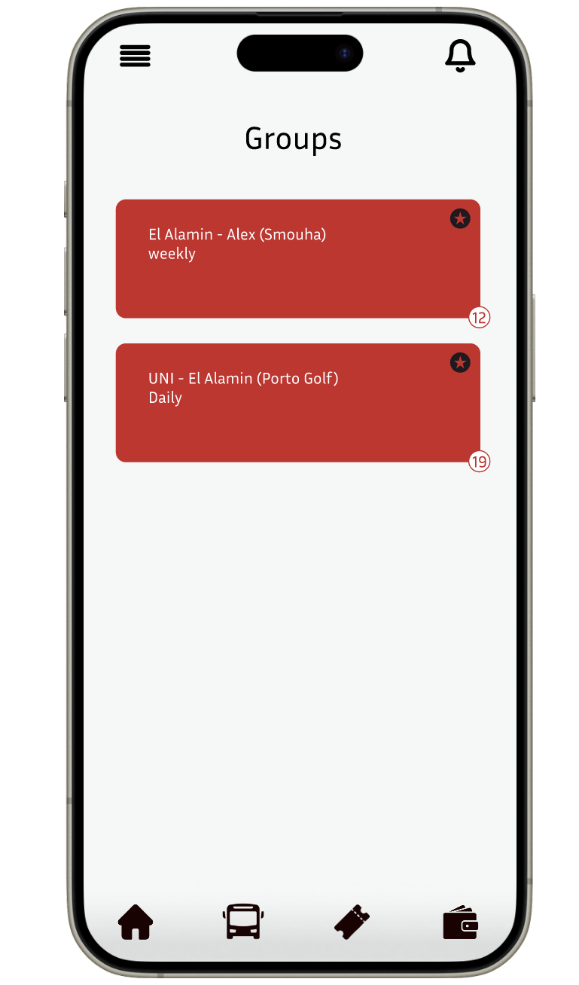
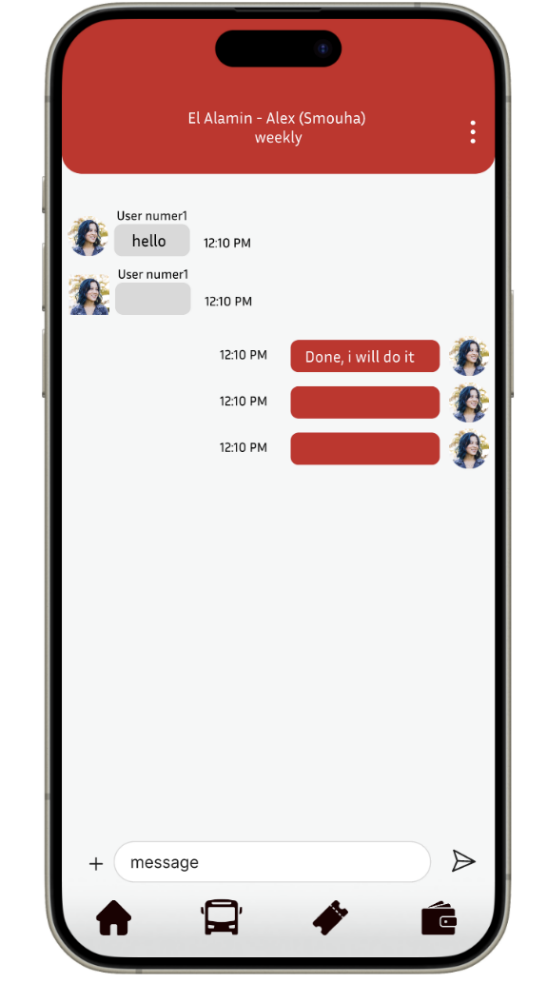
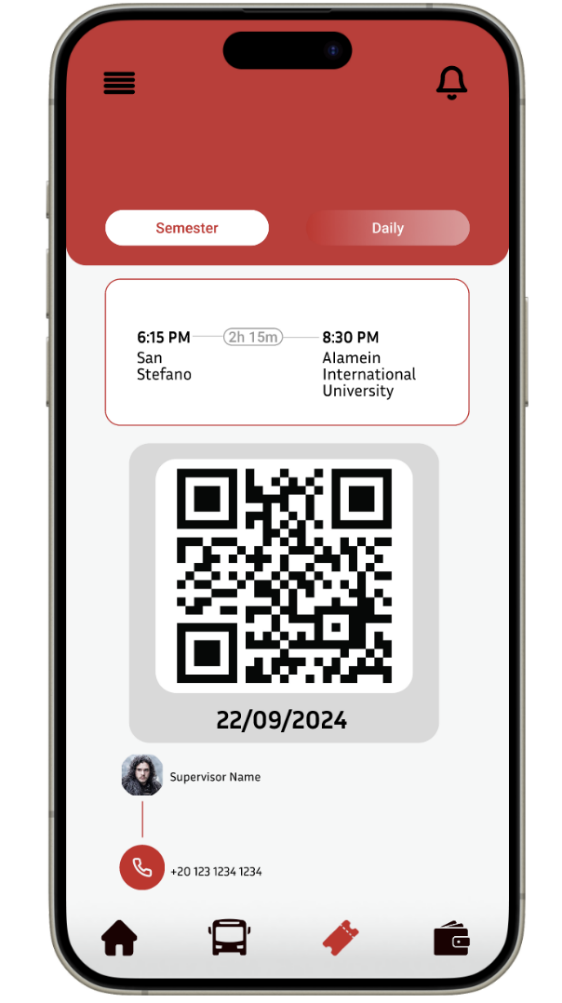
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Case ID** | **Test Description** | **Pre-Conditions** | **Test Steps** | **Expected Result** |
| TC\_001 | Validate that admins can successfully add a new bus | Admin login | 1. Navigate to 'Add Bus' page. 2. Enter all required details. 3. Click 'Save' | The new bus is added to the system and is visible in the bus list. |
| TC\_002 | Validate that admins can remove an existing bus from the system | Admin login, at least one bus added | 1. Navigate to the bus list page. 2. Select an existing bus. 3. Click 'Delete' and confirm the action | The bus is removed from the system and no longer appears in the list. |
| TC\_003 | Validate that admins can edit the details of an existing bus | Admin login, at least one bus added | 1. Navigate to the bus list page. 2. Select an existing bus. 3. Click 'Edit' and modify details. 4. Click 'Save' | The bus details are updated and reflect the edited information. |
| TC\_004 | Validate that admins can assign a bus to a specific route | Admin login, at least one bus and route exist | 1. Navigate to 'Assign Route' page. 2. Select a bus and route. 3. Click 'Assign' | The bus is assigned to the selected route and is visible in its details. |
| TC\_005 | Validate that admins can track the real-time location of a bus | Admin login, tracking system active | 1. Navigate to the tracking page. 2. Select a bus to view real-time location. | The bus's real-time location is displayed accurately. |
| TC\_006 | Validate that users can view the details of a bus | User login, at least one bus added | 1. Navigate to the bus list page. 2. Select a bus to view details. | The bus details are displayed to the user. |
| TC\_007 | Validate that users can view the route assigned to a bus | User login, bus assigned to a route | 1. Navigate to the bus list page. 2. Select a bus. 3. View assigned route details. | The assigned route is displayed to the user. |
| TC\_008 | Validate that admins and users can search for buses | Admin or User login | 1. Use the search functionality on the bus list page. 2. Enter search criteria and execute search. | The search results display matching buses based on criteria entered. |

**GUI**

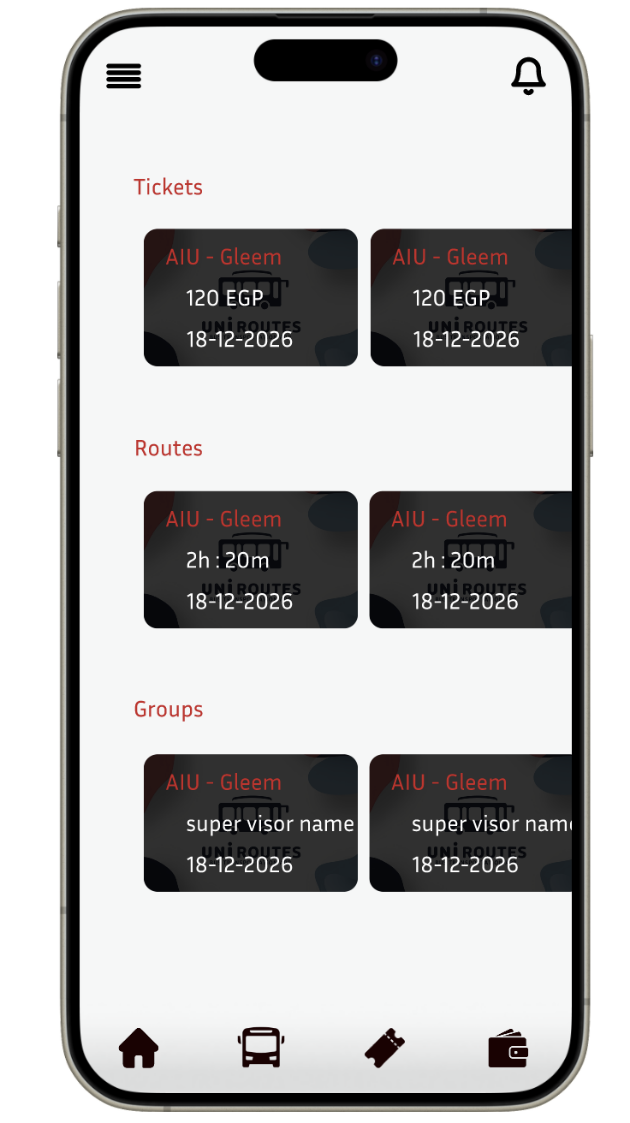
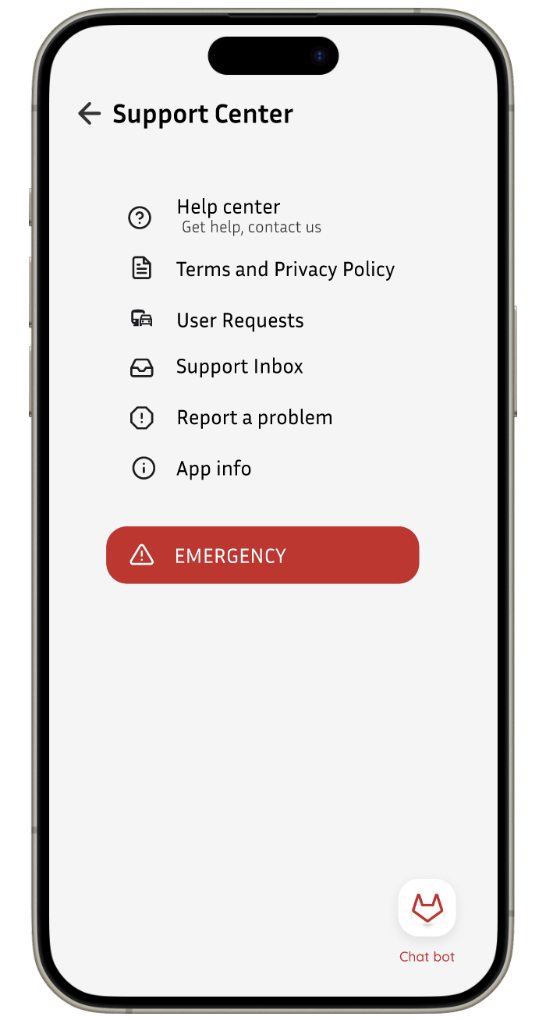
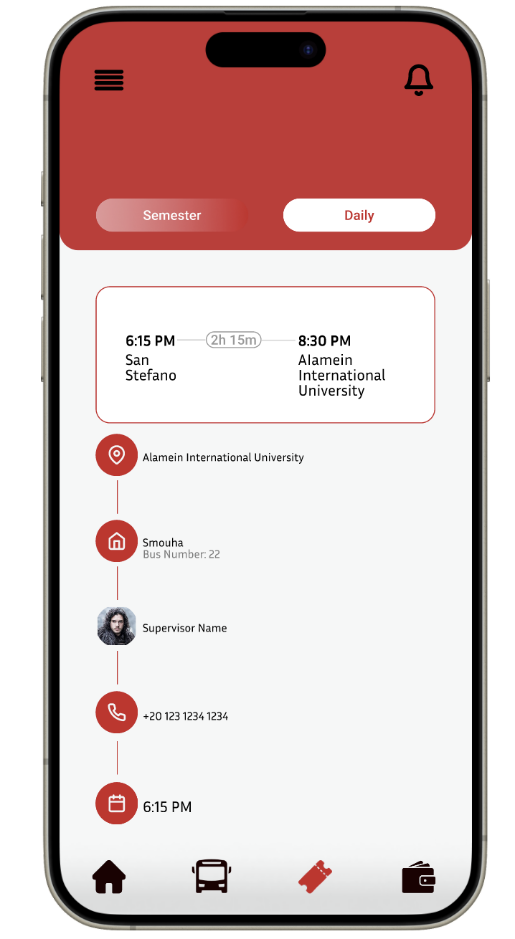
This is the home screen for the user with the bus details and notifications

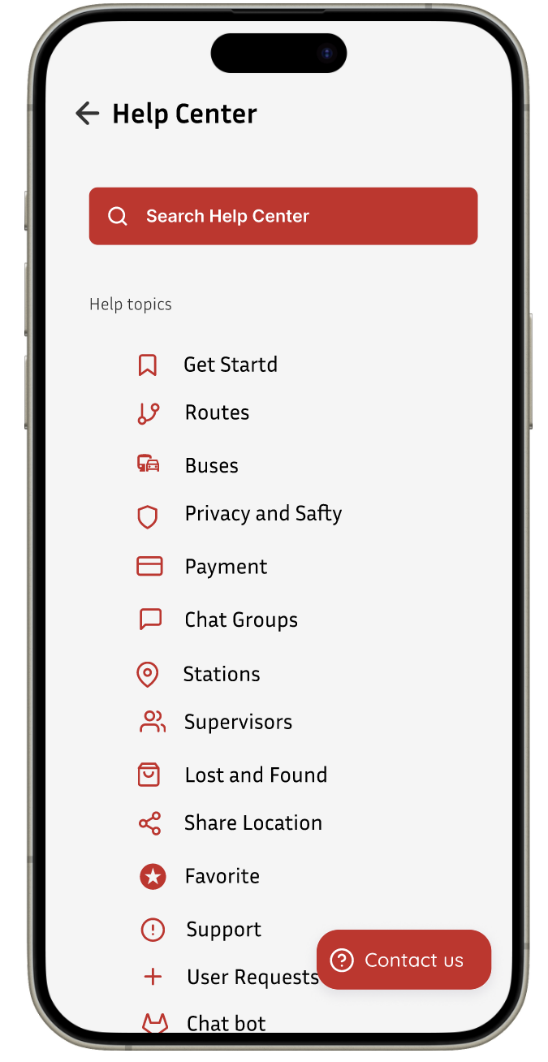
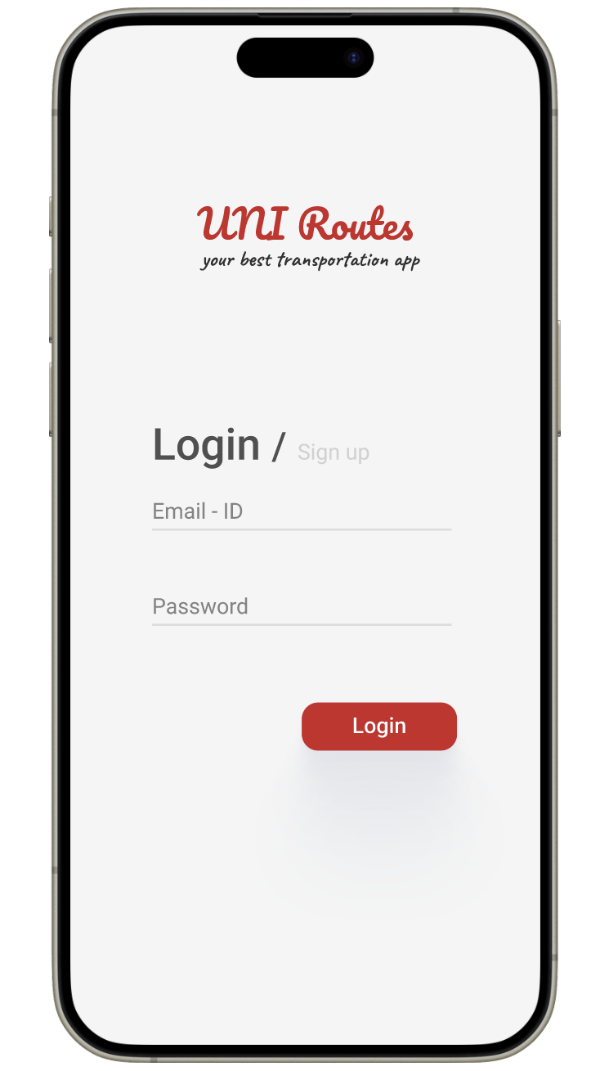
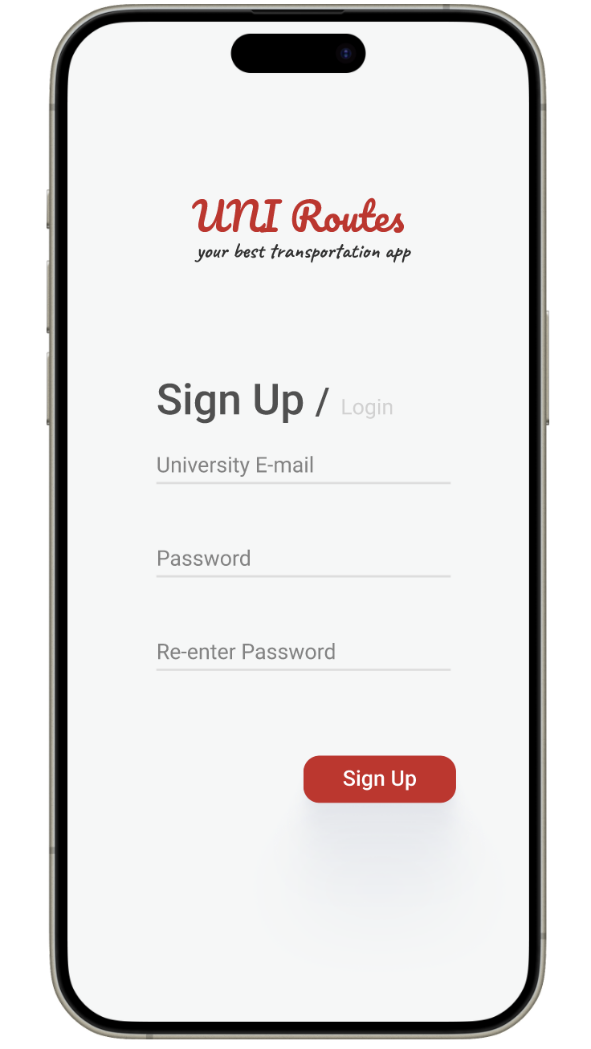
****

Subscriptions and the users wallet

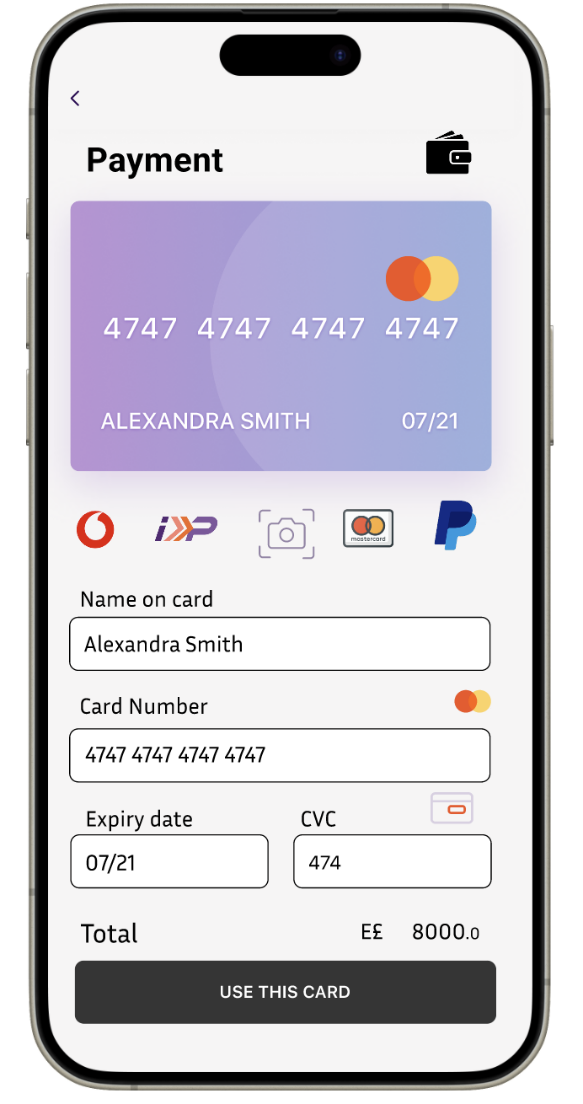
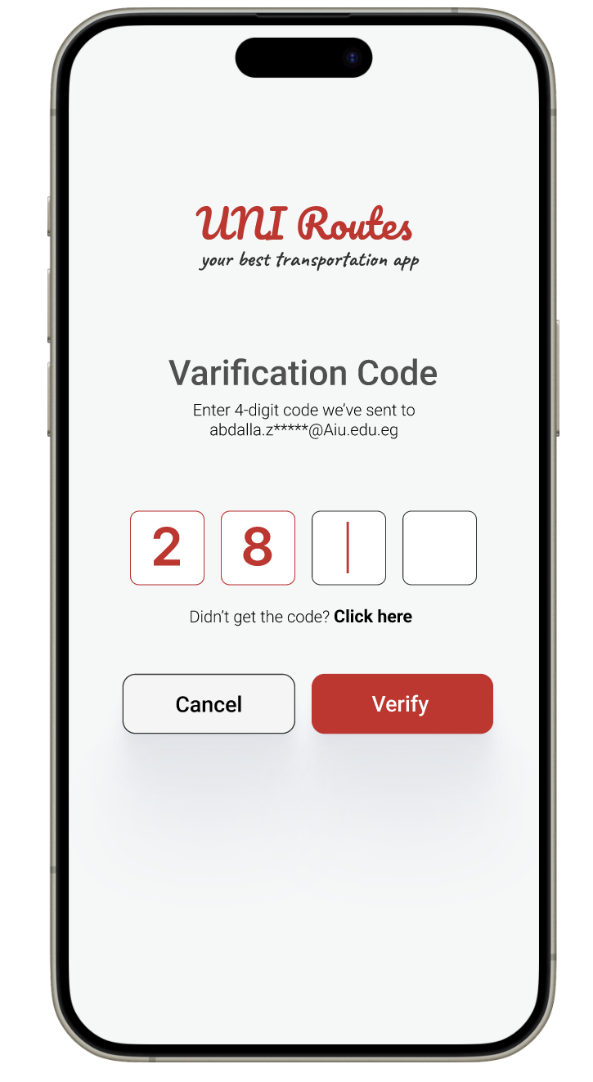
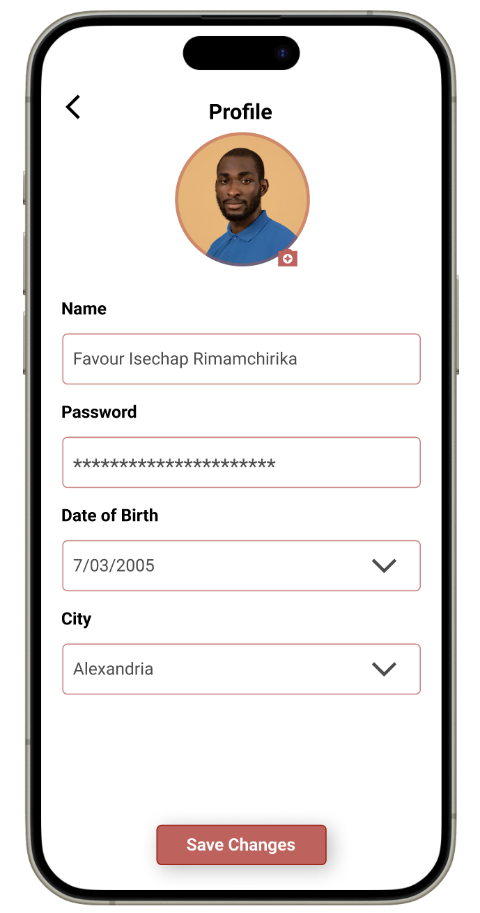
****

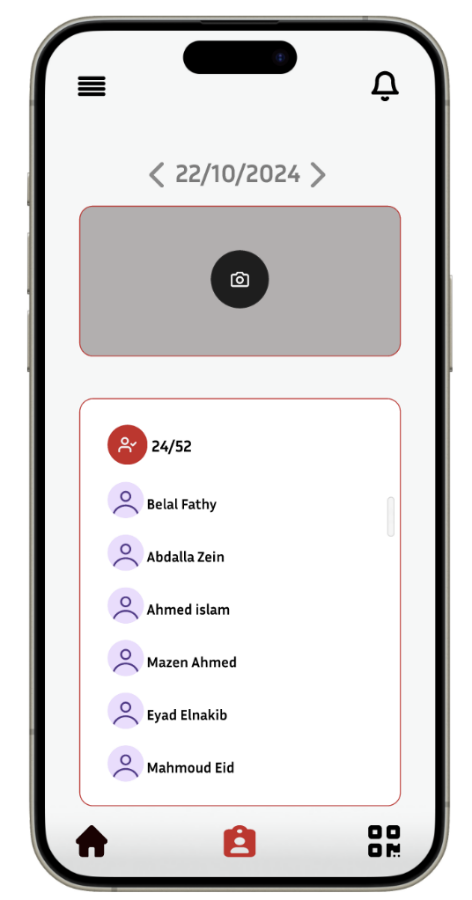
Group Chat For communication along with the ticketing system and the support center

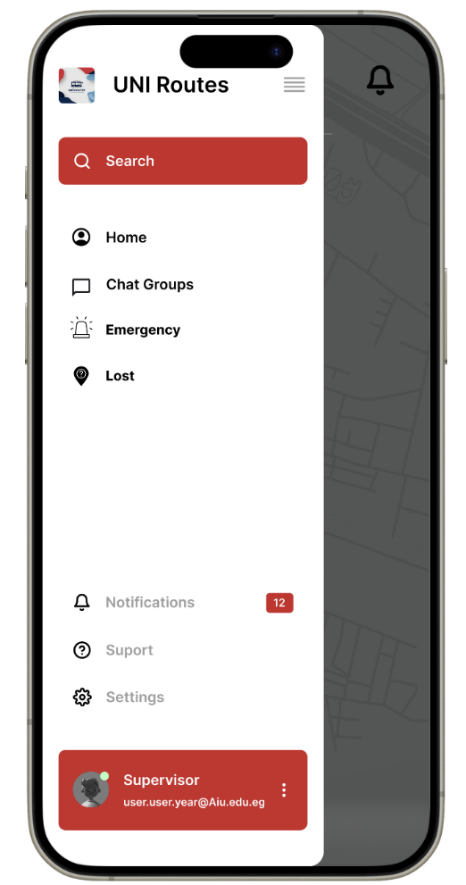
****



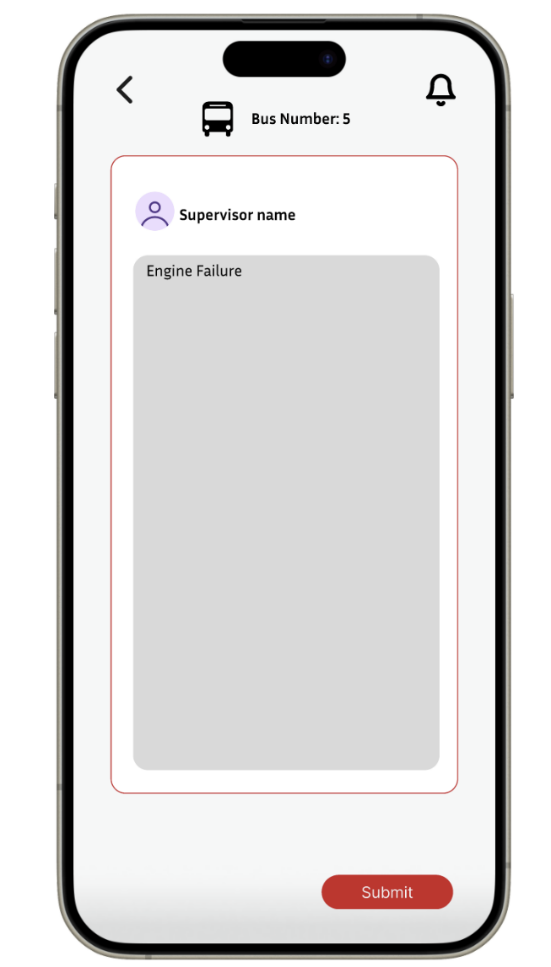
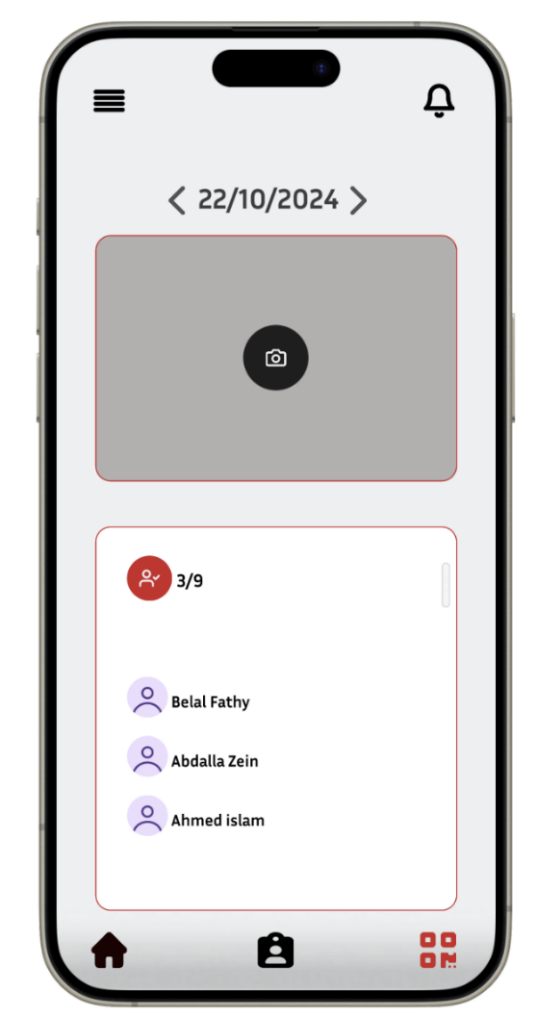
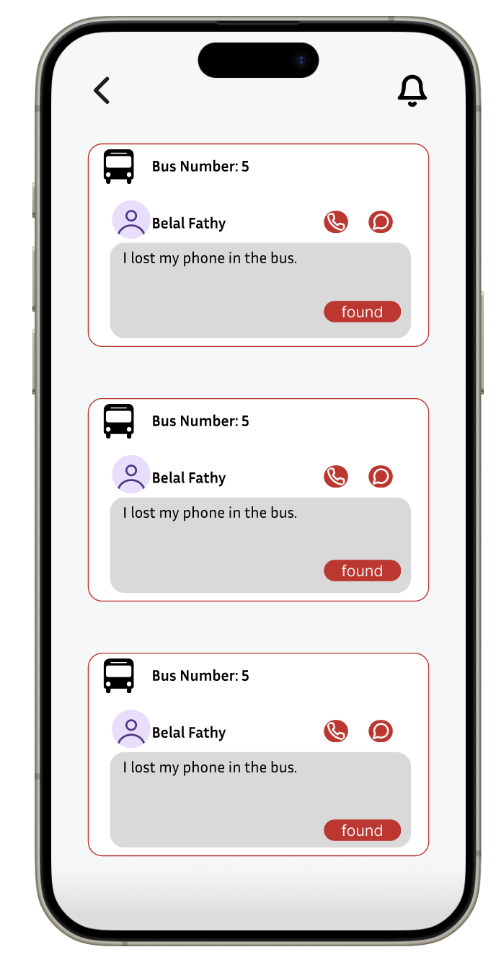
Help center and the login and sign-up page along with payment gateway profile and verification



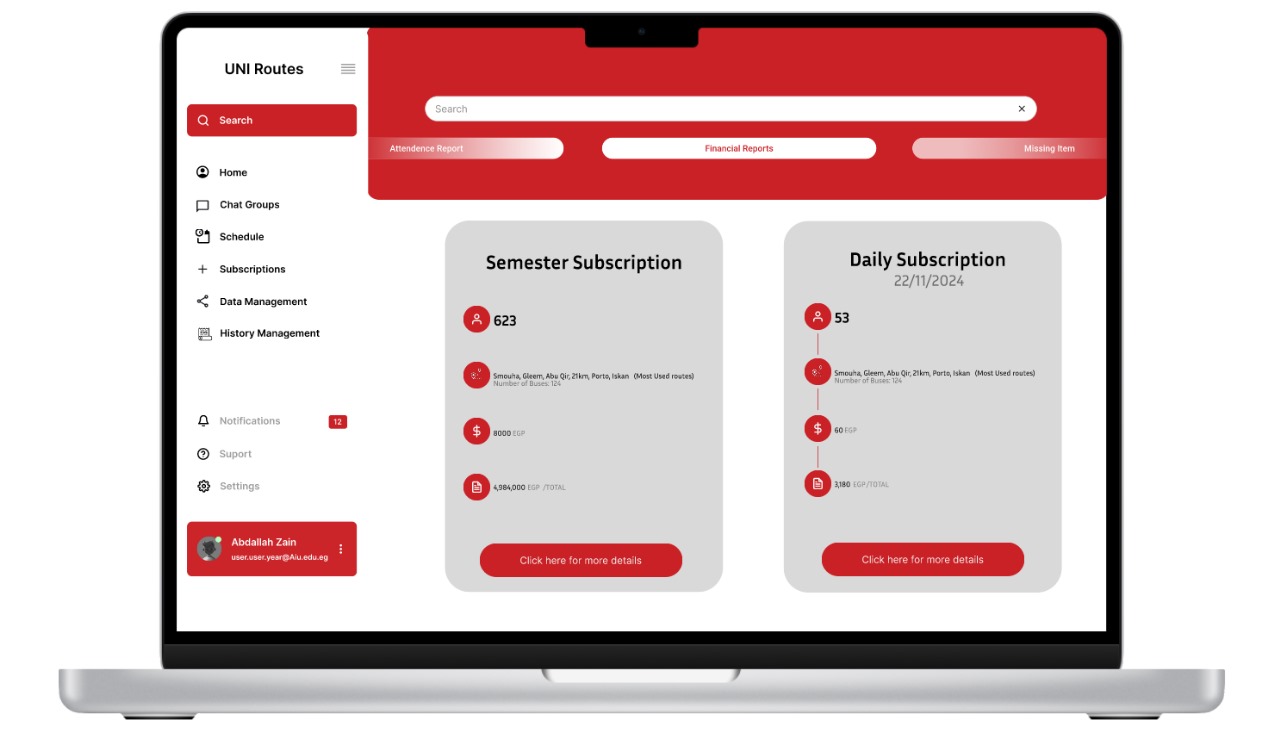
A phone with a map and a location on it

Description automatically generated

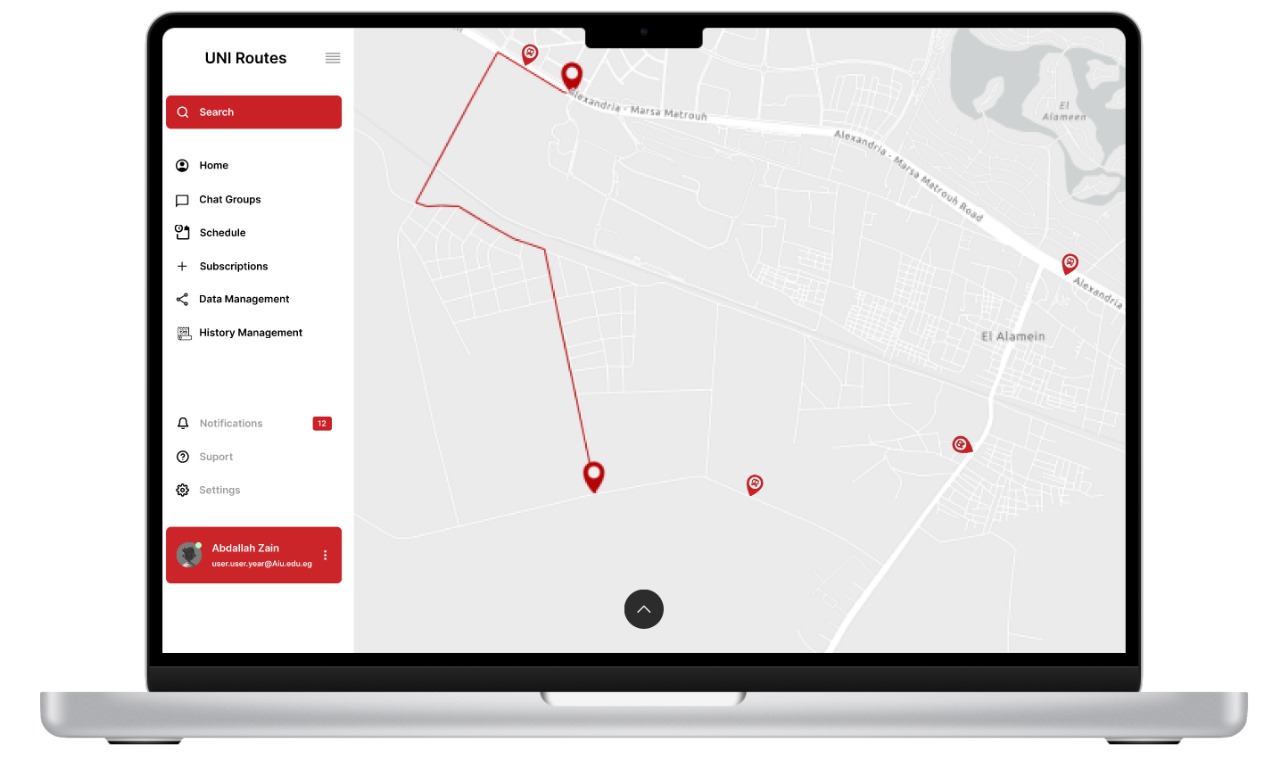
Rating system along with the menu of the supervisor his attendance and qr reader report and feedback access

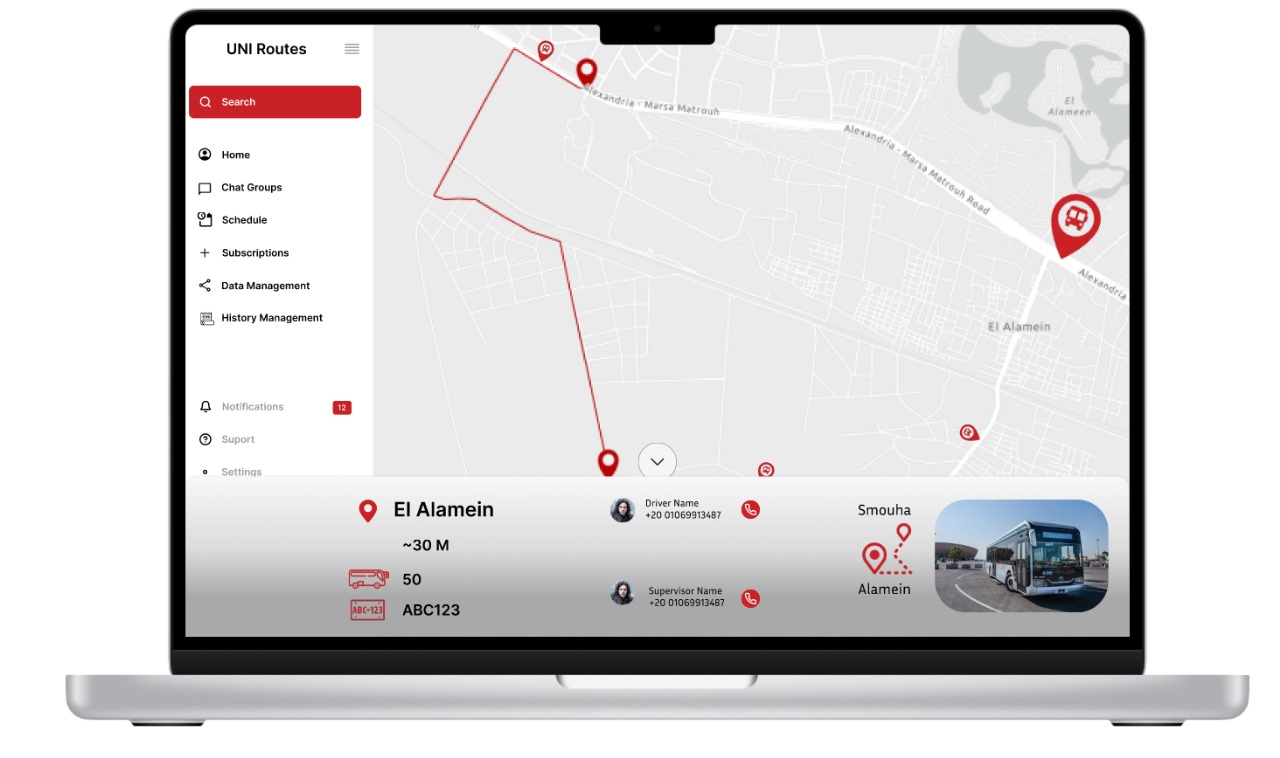


A screen shot of a cell phone

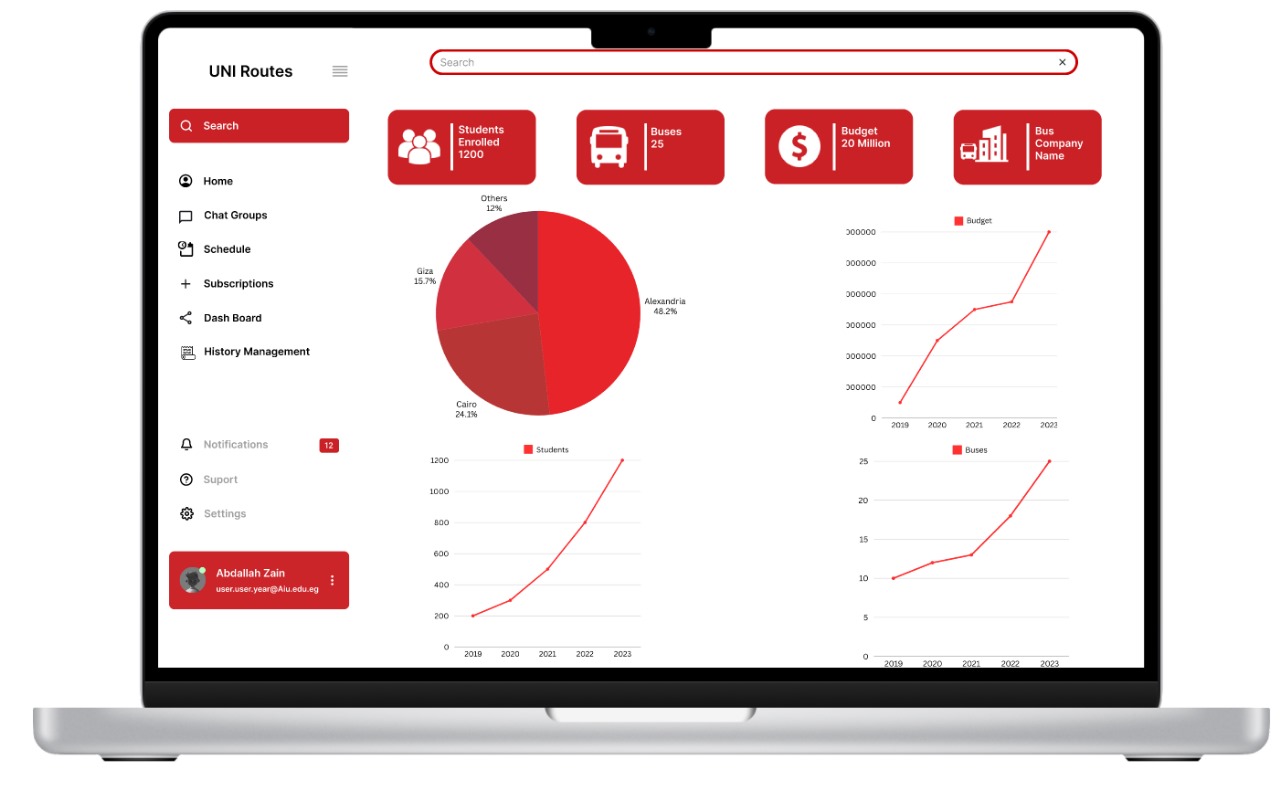
Description automatically generated

Users menu and the total subscription dashboard along with the admin home

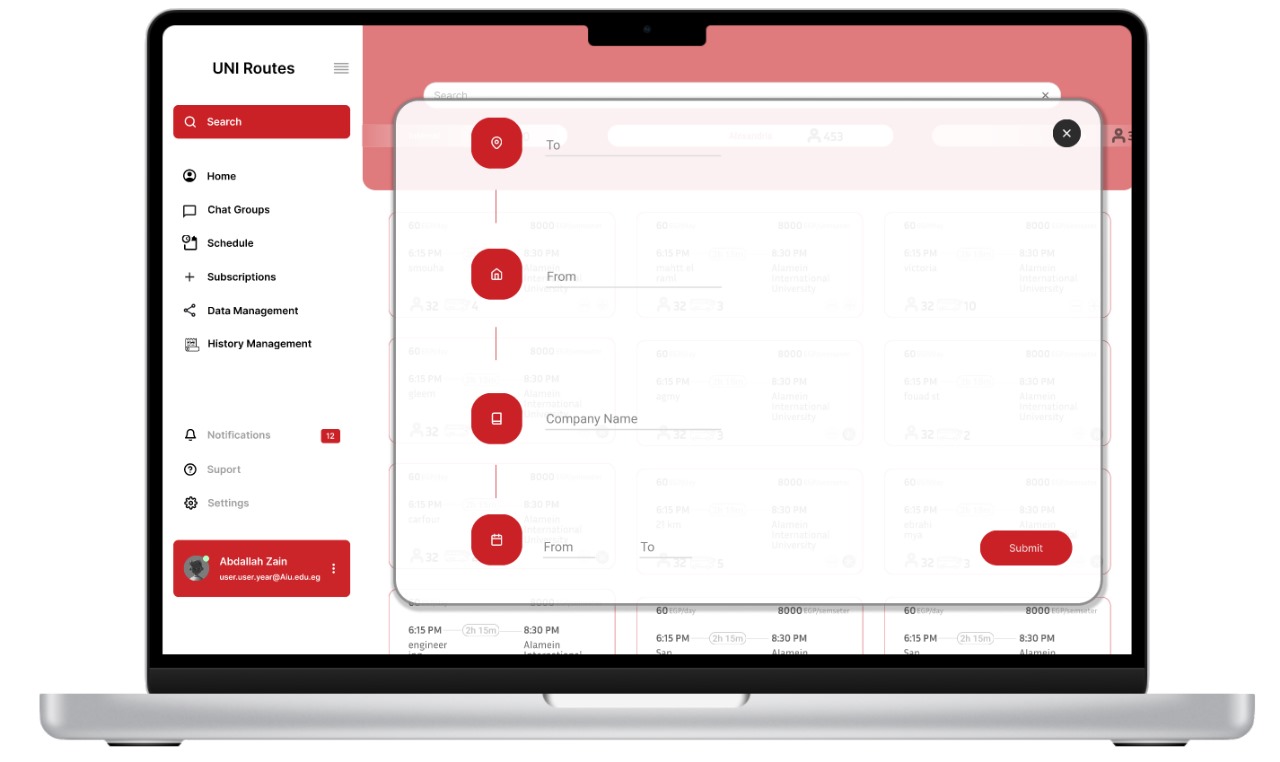




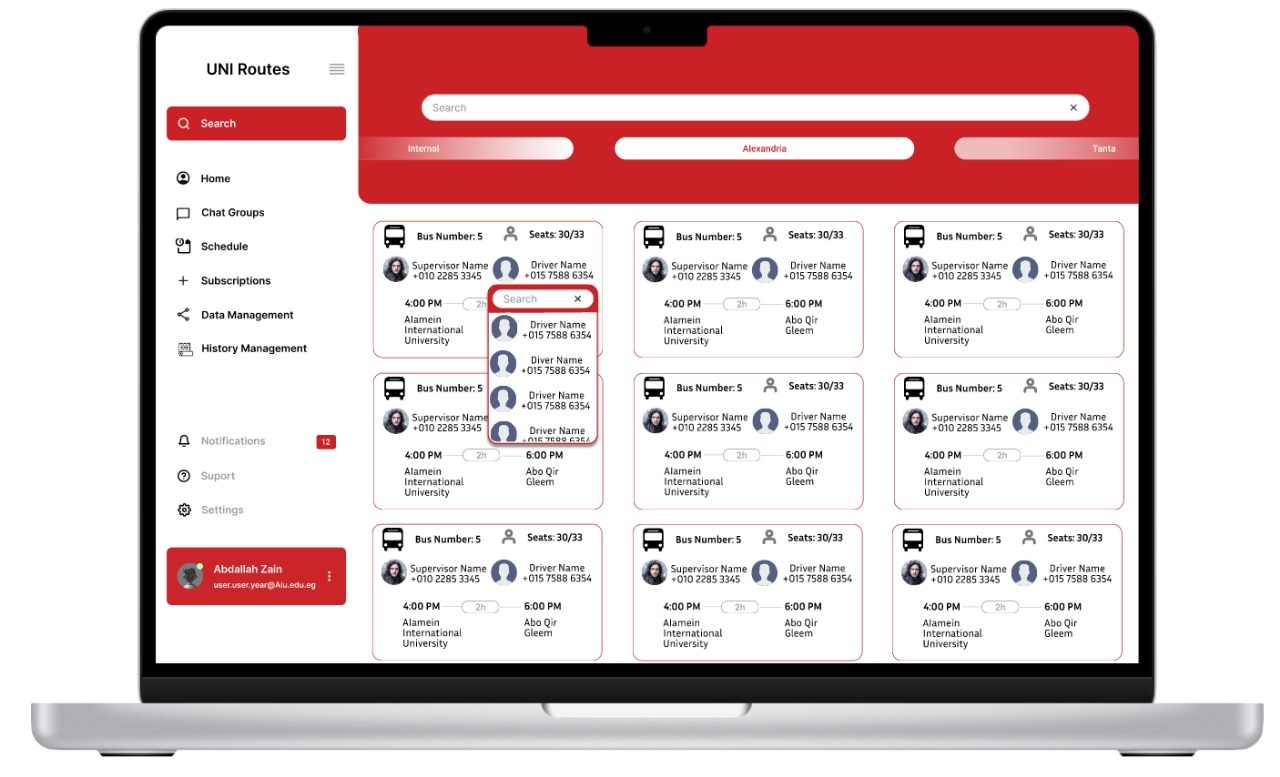
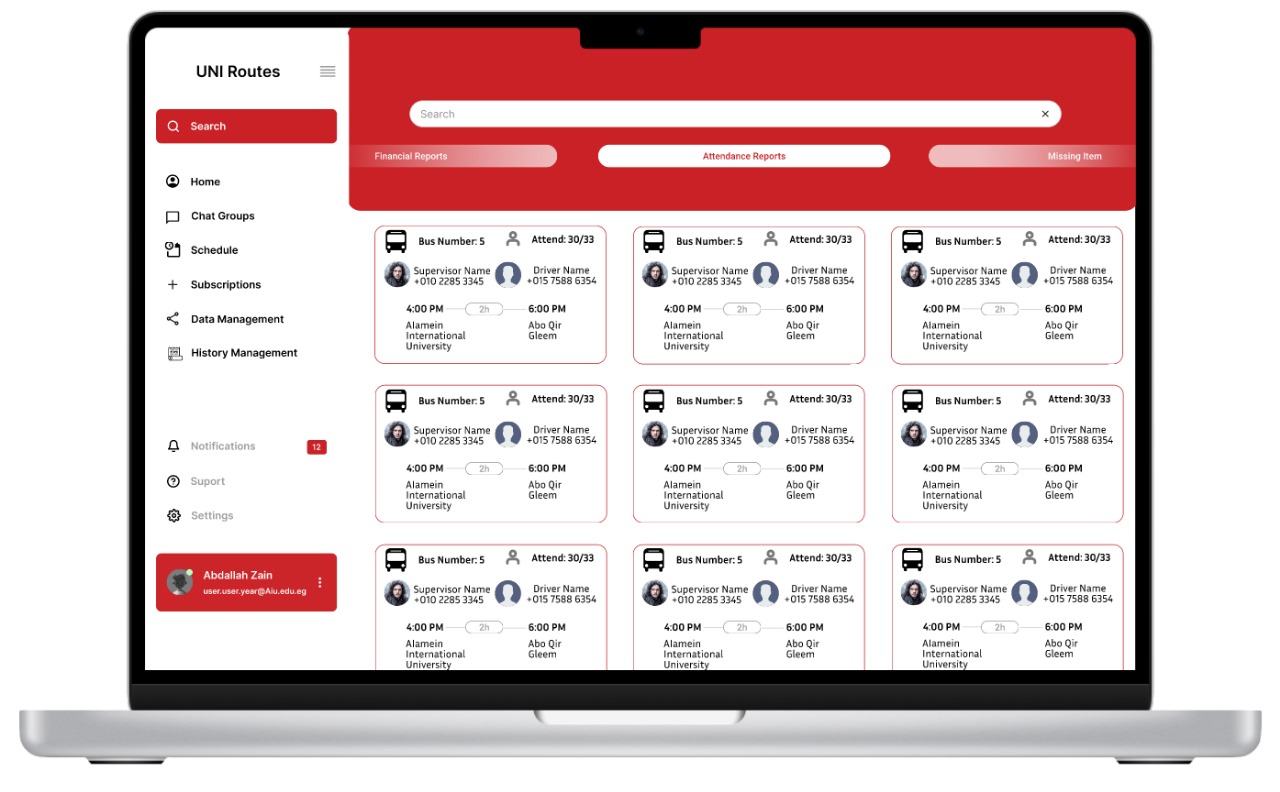
Details home screen for admins and their dashboard



A computer with a red and white screen

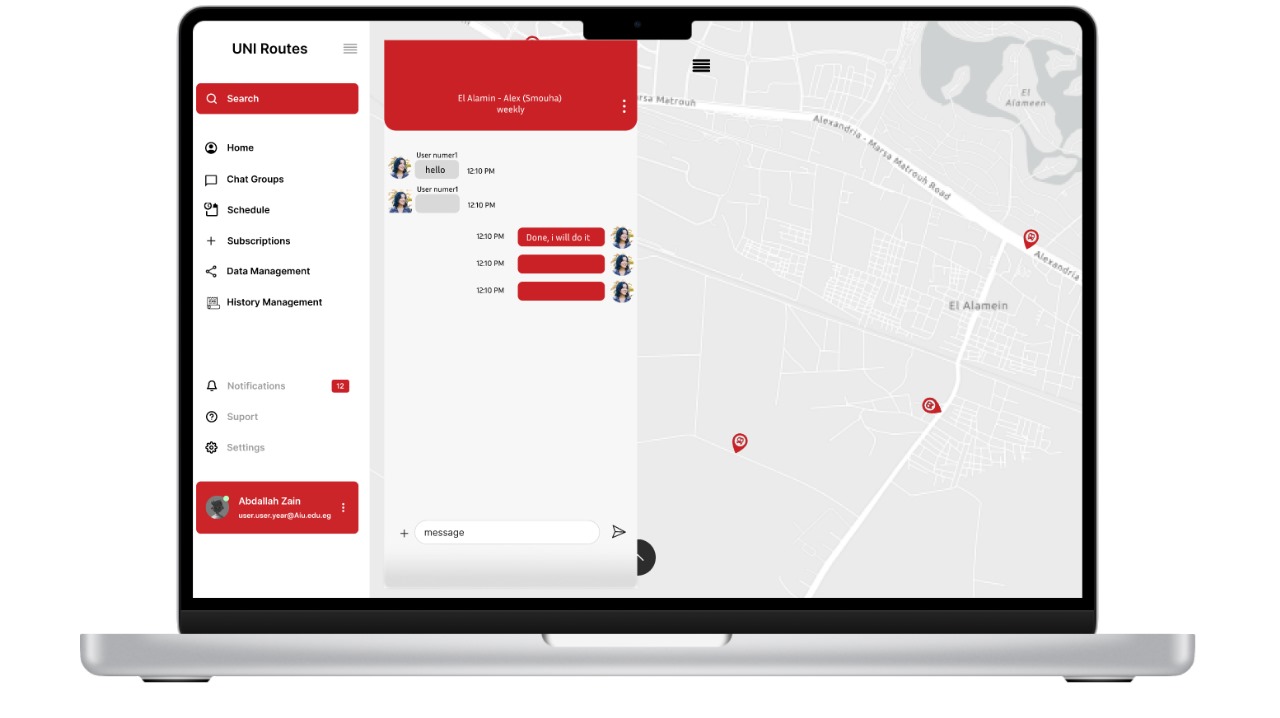
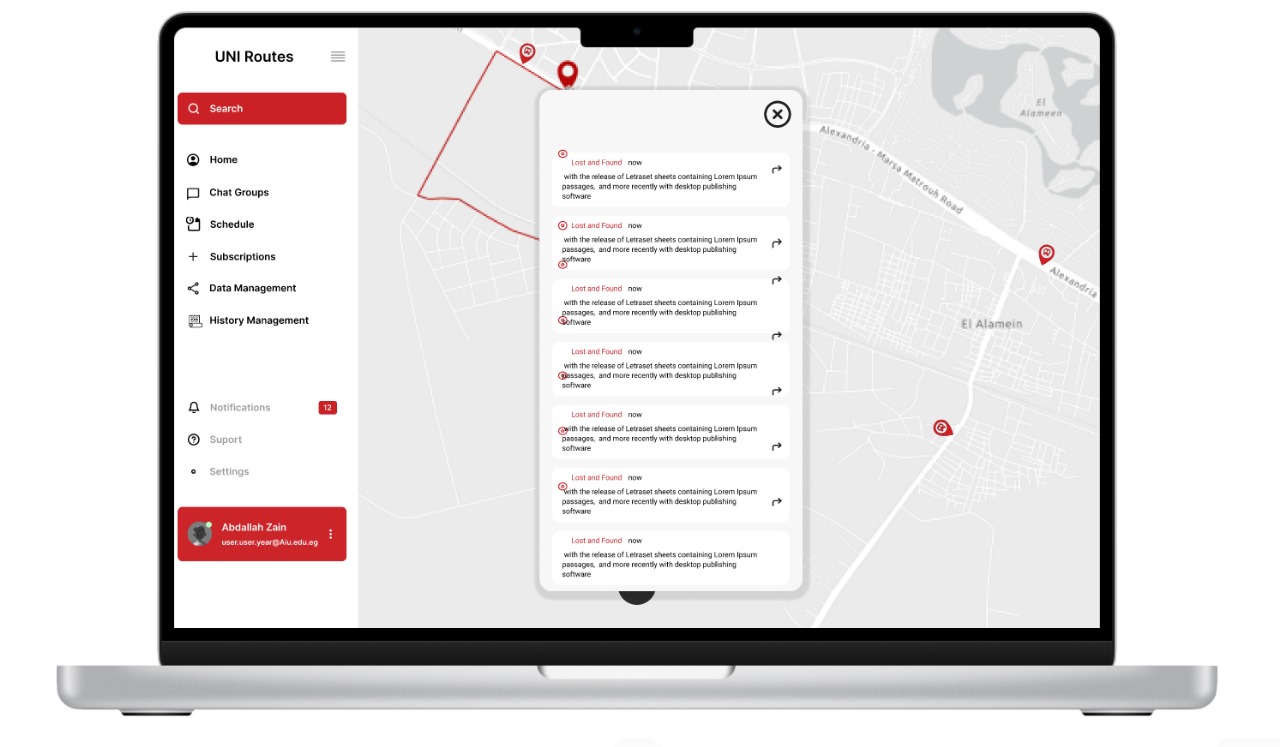
Description automatically generated

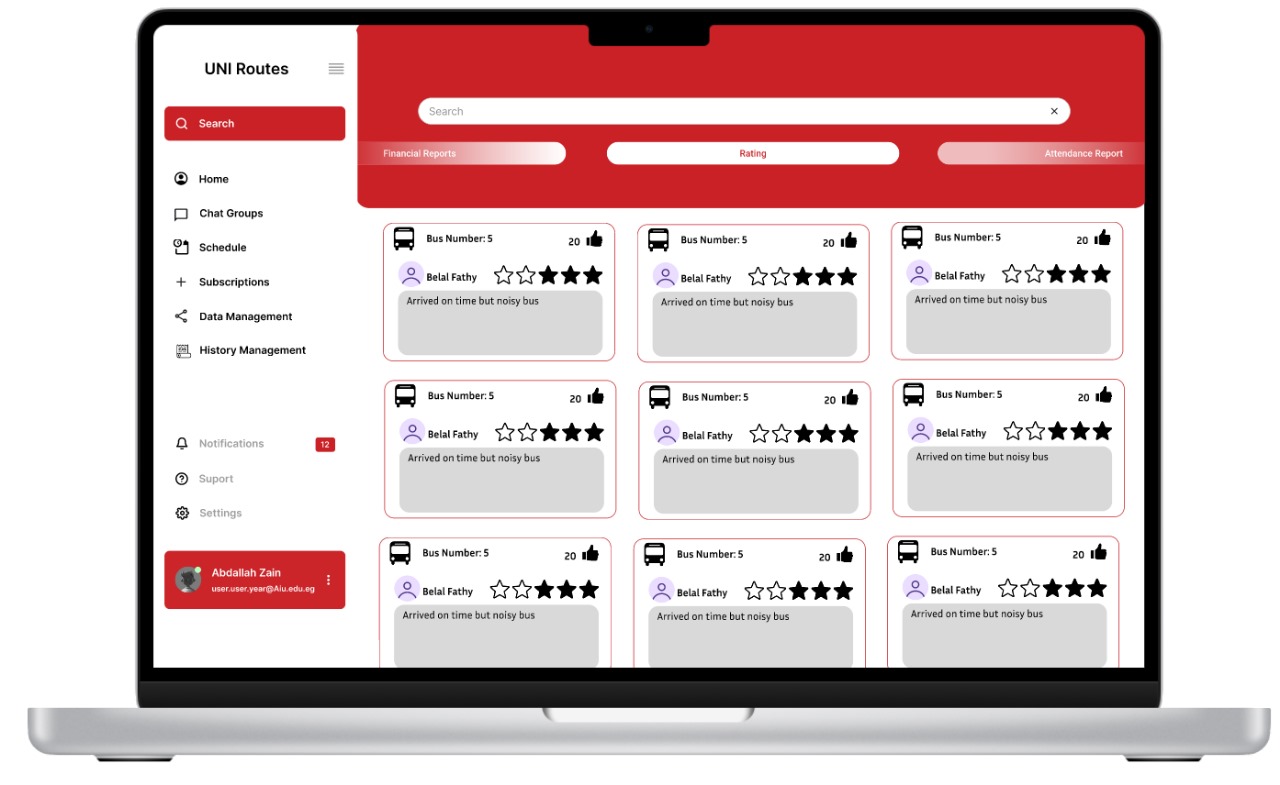
Subscription management for admis



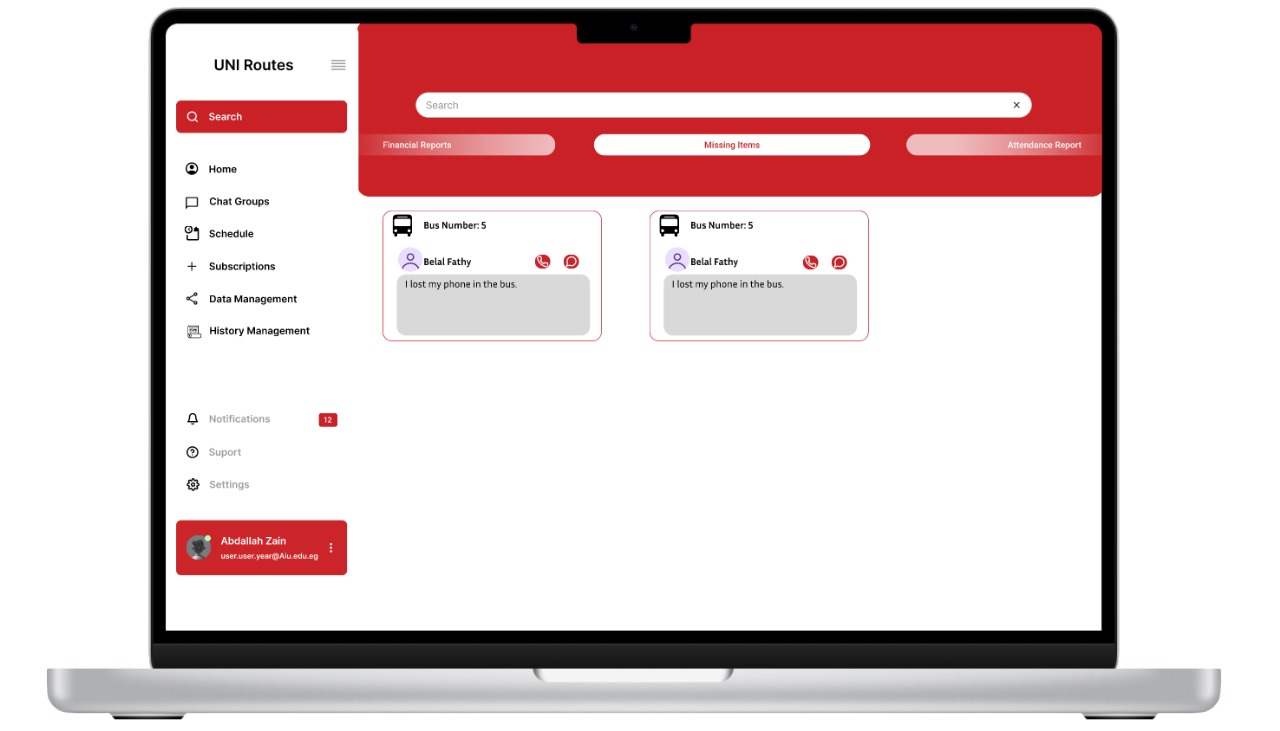
Schedule management and staff management fort admin

Notification and group chat management for admins





Rating system management for admins



Users have a mobile application while admins have a desktop application

Figma Link : <https://www.figma.com/design/nPvBAwzCdIXNFPMUD6mi9F/Untitled?node-id=43-74&t=JZ1o4OAl1m0Zev0n-1>

GitHub : <https://github.com/Abdallah4Z/UNIRoutes>

Abdallah Basem Zain:

•Works on test cases and requirements for Notification Management.

•Involved in the Traceability Matrix, subsystem,4 functional requirements, high-level use cases, low-level use cases, and a 9-frame GUI design.

Ahmed Islam Abbas:

•Focuses on Payment Management, including test cases and requirements.

•Covers the subsystem, Traceability Matrix, 4 functional requirements, high-level use cases, low-level use cases, and a 9-frame GUI design.

Belal Fathy Abdelfatah:

•Assigned to test cases and requirements for Subscription Management.

•Works on the Traceability Matrix, 4 functional requirements, low-level use cases, and an 11-frame GUI design.

•Non-functional requirement

Eyad Metwally Elnakib:

•Assigned to Account Management with test cases and requirements.

•Focuses on 4 functional requirements, low-level use cases.

•4 functional requirements.

Mahmoud Eid Khamis:

•Responsible for creating test cases and defining test requirements specifically for Bus Management, low-level use cases.

•4 functional requirements.

Mazen Ahmed Samir:

•Handles test cases and requirements for Routes and Stations Management.

•Contributes to the Traceability Matrix, 4 functional requirements, low-level use cases, an 8-frame GUI design, and non-functional requirements.

•Non- functional requirement