FACULTY OF
BUSINESS INFORMATICS

جامعة مصر للمعلوماتية
EGYPT UNIVERSITY
OF INFORMATICS

UniTracker

Main Supervisor

Dr. Anas Ismail

Submitted by

#	Student Name	Student ID
1	Jana Ahmed	21-101052
2	Maya Yakout	21-101019
3	Nouran Ahmed	21-101027
4	Christine Nagy	21-101058
5	Zeina Sherif	21-101176

Capstone Project

2024-2025

In Memory of Dr. Reem Bahgat

President of Egypt University of Informatics

We humbly dedicate this work to Allah Almighty, the Most Gracious and Merciful. His boundless mercy, divine guidance, and infinite wisdom have been our strength throughout this journey. Every step taken and every challenge overcome has been by His will and grace.

This project is also lovingly dedicated to the memory of **Prof. Reem Bahgat**, the Founding President of EUI. Her visionary leadership, unwavering dedication, and profound belief in the power of education continue to inspire and guide us. Though she is no longer with us, her legacy lives on in the values she instilled and the foundation she built. May her soul rest in eternal peace.

Lastly, we extend our deepest gratitude to our families. Their endless support, encouragement, and sacrifices formed the cornerstone of our perseverance and achievement. Without their love and belief in us, this accomplishment would not have been possible.

Thank you, Dr. Reem Bahgat.

In honor of all you gave us.

Abstract

The UniTracker system is a comprehensive university transportation management platform designed to modernize and optimize the operation of campus shuttle services. Built in response to widespread issues in university transport, such as lack of visibility into bus schedules, overcrowding, inefficient routing, and student dissatisfaction, UniTracker addresses these problems through a mobile-first, real-time, and data-driven solution. It empowers students, faculty, and staff to track buses live, reserve seats in advance, receive timely notifications, and plan their commutes with confidence. Simultaneously, it offers administrators robust backend tools for fleet monitoring, schedule optimization, and demand-based resource allocation.

The application was developed using **Flutter** for a seamless cross-platform user experience across Android, iOS, and web, while **Supabase** handles authentication, storage, and backend services with real-time database capabilities. It integrates **Google Maps API** for geolocation and live bus tracking, offering intuitive map views and dynamic route information. Security and scalability were primary considerations in the system architecture, ensuring the platform is suitable for large multi-campus institutions while remaining responsive and secure for daily users.

UniTracker stands out for its dual benefit: it enhances user experience and operational efficiency while contributing to broader institutional goals such as sustainability and digital transformation. It reduces fuel waste, mitigates unnecessary trips, and decreases carbon emissions by allowing institutions to allocate buses based on real-time demand. The system's design includes support for multilingual interfaces, role-based dashboards (for students, drivers, and admins), and advanced analytics to inform long-term planning.

Furthermore, the platform is fully scalable and modular, making it adaptable to the unique needs of other universities or even entire educational networks across Egypt and beyond. With a flexible revenue model based on subscription tiers and a strong return on investment demonstrated in financial feasibility studies, UniTracker is not just a technological solution, it is a sustainable business opportunity for transforming public university transportation into the digital age.

Acknowledgement

In the name of Allah, the Most Gracious, the Most Merciful, we begin by expressing our sincere gratitude to Allah Almighty for granting us the strength, patience, and wisdom to undertake and complete this capstone project. His divine guidance illuminated our path during moments of difficulty and uncertainty, and it is by His will that we were able to bring UniTracker to life.

We extend our heartfelt appreciation to our esteemed supervisor, **Dr. Anas Ismail**, for his continuous support, insightful feedback, and unwavering belief in our capabilities. His mentorship provided us with both technical direction and motivation, which were critical throughout the design, development, and testing phases of this project.

This work is especially dedicated to the late **Prof. Reem Bahgat**, Founding President of Egypt University of Informatics. Her pioneering vision and dedication to academic excellence remain a source of inspiration. Her legacy lives on in every line of code, every research milestone, and every student she empowered through her leadership. May her soul rest in peace.

We are also immensely thankful to all faculty members, lab technicians, and university staff who contributed their knowledge and resources to support our research and technical implementation. Their encouragement and expertise helped turn our vision into reality.

To our families, we owe a debt of gratitude that words cannot adequately express. Their constant support, love, sacrifices, and prayers were the foundation of our resilience. Every late night, missed event, and challenge we overcame was made possible because of their unconditional belief in us.

Lastly, we would like to recognize our fellow students, testers, and transportation stakeholders who participated in interviews and usability testing. Your real-world insights shaped the final product and made it more user-centric and practical.

This project is not only a technological achievement but a testament to collaboration, dedication, and the shared goal of building a better future for university communities.

Table of Contents

Abstract 3

Acknowledgement 4

List of Tables 14

List of Figures 15

List of Abbreviations 16

Chapter 1 17

Project Proposal 17

1.1 Introduction 17

1.2 Problem & Gap Field 17

1.3 Importance & Objectives 18

1.4 Competitive Advantage 19

1.5 Target Users 19

1.6 Inputs & Outputs 19

Inputs 19

Outputs 20

1.6.1 Immediate Outcomes 20

1.6.2 Long-term Outcomes 20

1.7 System Functions 21

1.8 System Features 21

1.9 SWOT Analysis 21

Strengths: 21

Weaknesses: 22

Opportunities: 22

Threats: 23

1.10 Scope of Implementation **23**

1.10.1 Key Features 23

1.10.2 Who's It For? 24

1.10.3 Technical Details 24

1.10.4 Final Deliverables 24

1.10.5 Who's Not Included? 24

Chapter 2 25

Project Planning **25**

2.1 Project planning introduction **25**

Phase 1: Analysis and Requirements Gathering (Weeks 1–3) 25

Phase 2: System Design and Planning (Weeks 4–6) 25

Phase 3: Development and Implementation (Weeks 7–12) 25

Phase 4: Deployment and Monitoring (Weeks 13–16) 26

2.2 Task table **26**

First Semester: Planning Phase 26

Second Semester: Implementation Phase 27

Gantt Chart **27**

Gantt Chart Analysis for Project Timeline **27**

X-Axis (Horizontal): 27

Y-Axis (Vertical): 28

Task Breakdown with Timelines 28

2.3 Marketing **29**

1. Target Audience 29

2. Marketing Messages 29

3. Marketing Materials 29
4. University Outreach Strategy 30
5. Online Marketing 30
6. Engage Students 30
7. After Successful Pilots or Adoption, Gather: 30
8. Plan a Multi-Media Launch Campaign 31
2.4 Organization Chart **31**
2.4.1 Executive Level 31
2.4.2 Management Level 32
University Administrator (Campus Operations) 33
Transportation Manager (Fleet Operations) 33
IT Support Manager (Technical Operations) 33
2.4.3 Operational Level 34
Bus Drivers (Field Operations) 34
Reservation Coordinator (Booking Management) 35
Location Coordinator (Pickup Point Management) 35
2.4.4 End-User Level 36
Students (System Users) 36
Guest Users (Visitors & Temporary Users) 36
2.4 Project Overview **36**
2.4.1 Project Burndown 37
2.4.2 Cost Overview 38
Chapter 3 39
Feasibility Study **39**
3.1 Technical Feasibility **39**

3.1.1 Technology Requirements 39
3.1.2 Resource Availability 39
3.1.3 Hardware Requirements 39
3.2 Revenue Model 40
3.2.1 Revenue Projections per Semester 40
3.2.2 Break-Even Analysis 40
3.2.3 Payback Period Calculation 40
3.3 Operational Feasibility 41
3.3.1 Integration 41
3.3.2 Staffing Needs 41
3.4 Legal Feasibility 42
3.5 What is the Importance of a Feasibility Study? 44
3.6 Software Pricing Table 45
3.6.1 Dart & Flutter 45
3.6.2 Supabase 45
3.6.3 Google Maps API 46
3.6.4 Provider (State Management) 46
3.6.5 Other Tools 46
3.7 Cash Flow 46
3.7.1 Year 1 46
3.7.2 Year 2 46
3.7.3 Year 3 47
Chapter 4 48
System Analysis & Design 48
4.1 What is System Analysis? 48

4.2 What is the Importance of System Analysis? 48

4.3 Context Diagram 49

4.3.1 External Entities 49

4.3.2 Central System 50

4.3.3 Key Features 50

4.4 DFD 52

1. Entities: 52

2. Major Processes: 52

3. System-Wide Interactions: 53

4.4.1 Data Flow (0-Level Diagram) 54

4. Inflows and Outflows for Each Process: 55

5. Updates from Data Flows to Data Stores: 60

4.4.2 Data Flow (1-Level Diagram) 62

a. 1.0 Select University 65

b. 2.0 Sign Up/Login 66

c. 3.0 Reservation 68

d. 4.0 Bus Route & Schedule 69

e. 5.0 Bus Tracking 70

f. 6.0 Manage Route & Schedule 71

g. 7.0 View Route & Schedule (Driver) 73

h. 8.0 Bus Status 74

i. 9.0 Medical & License Status 75

j. 10.0 Feedback 77

4.5 Use Case Diagram 79

4.5.1 Actors (Who Uses the System) 80

1. Student/User 80

2. Driver 80

3. University (Facilities Department) 81

4.5.2 What does the System Does? 81

1. Login 81

2. View Bus Route & Schedule 81

3. Real-Time GPS Tracking 82

4. Make a Reservation 82

5. Send Reservation Confirmation 83

6. Monitor & View Passenger Count 83

7. Modify Bus Route & Schedule 83

8. Update Bus Location 84

10. Track Bus Location 84

4.5.3 Interactions 85

4.6 Entity Relationship Diagram (ERD) 86

4.6.1 Entities & Relationships 87

4.7 UML Class Diagram 96

1. User 96

2. Reservation 97

3. Bus 97

4. Schedule 97

5. Driver 98

6. Admin 99

7. Location 99

4.8 Story Board 100

Chapter 5 103

Data Dictionary 103

5.1 Process Logic Description 103

5.2 Structured English 105

5.3 Logic Decision Table 110

5.4 Data Dictionary 113

5.5 Data Element 113

5.5.1 Entity: User 113

5.5.2 Entity Reservation 114

5.5.3 Entity: PaymentHistory 115

5.5.4 Entity: UserPaymentCards 117

5.5.5 Entity: Driver 118

5.5.6 Entity: Bus 119

5.5.7 Entity: Route 120

5.5.8 Entity: BusStop 121

5.5.9 Entity: Schedule 122

5.5.10 Entity: TripHistory 122

5.5.11 Entity: Admin 124

5.6 Data External Entity 125

5.7 Data Process 126

5.8 Data Stores 126

5.9 Data Flow 126

Chapter 6 127

System Implementation 127

6.1 Overview 127

6.2 Front-End Implementation 127

- **Flutter** 127
- **Dart** 127

6.3 Back-End Implementation 128

- **Supabase** 128
- **PostgreSQL (via Supabase)** 128

6.4 Real-Time Bus Tracking 128

- **Google Maps API** 128

6.5 Notifications and Alerts 128

- **OneSignal** 128

6.6 UI/UX Design and Prototyping 129

- **Figma** 129

6.7 Development Environment 129

- **Android Studio** 129

6.8 API Testing and Integration 129

- **Postman** 129

6.9 Database Design and Reporting 129

- **Database Report** 129

Chapter 7 131

UI Manual 131

6.1 Student Mobile App 131

6.2 Driver Mobile App 146

6.3 Admin Web App 149

Chapter 8 161

Conclusion and Recommendations and Future Improvements 161

8.1 Conclusion 161

8.2 Recommendations and Future Improvements 161

List of Tables

Table 1: Table of Abbreviations 16
Table 2: Task Table 26
Table 3: Project Burndown Phases 37
Table 4: Cost Overview 38
Table 5: Cash Flow 46
Table 6: Use Case Interactions 85
Table 7: Logic Decision Tables 110

List of Figures

Figure 1: Gantt Chart 27
Figure 2: Organizational Chart 31
Figure 3: Project Burndown 37
Figure 4: Context Diagram 49
Figure 5: Data Flow (0-Level Diagram) 54
Figure 6: Data Flow (1-Level Diagram) 62
Figure 7: Use Case Diagram 79
Figure 8: Entity Relationship Diagram (ERD) 86
Figure 9: UML Class Diagram 96
Figure 10: Story Board 102
Figure 11: Mobile App UI Manual 131

List of Abbreviations

Table 1: Table of Abbreviations

Abbreviation	Full Term
UBTRS	University Bus Tracking and Reservation System
GPS	Global Positioning System
API	Application Programming Interface
EUI	Egypt University of Informatics
UI	User Interface
UX	User Experience
DFD	Data Flow Diagram
ERD	Entity Relationship Diagram
UML	Unified Modeling Language
LLC	Limited Liability Company
MCIT	Ministry of Communications and Information Technology
MFA	Multi-Factor Authentication
GAFI	General Authority for Investment and Free Zones

Chapter 1

Project Proposal

1.1 Introduction

The University Bus Tracking and Reservation System (UBTRS) is a smart mobile application designed to solve common transportation challenges faced by university students and staff. Many institutions still rely on outdated methods to manage campus buses, leading to confusion, long wait times, overcrowded buses, or underutilized trips. UBTRS addresses these issues by offering real-time bus tracking through a live map, seat reservation functionality, and timely notifications about delays or schedule changes. The app is easy to use and compatible with both Android and iOS devices. It also provides valuable data to university administrators, enabling better planning and resource management. The system benefits students by helping them plan trips efficiently, assists drivers with clear scheduling, and supports transport offices with real-time operational insights. Its main goals include saving time, improving safety, enhancing efficiency, and offering a scalable solution that can be adopted by other universities. Once downloaded, users can log in, choose their university, view schedules, reserve seats, and track buses live. UBTRS brings together technology and transportation to make campus commuting more reliable, organized, and stress-free.

1.2 Problem & Gap Field

The current transportation systems at many universities face several challenges that impact the efficiency and convenience of students, staff, and faculty. The lack of real-time tracking makes it difficult for users to determine the exact location of university buses, leading to prolonged waiting times and uncertainty about arrival and departure schedules. Additionally, the absence of a clear, accessible platform for bus schedules and routes complicates journey planning and often results in missed buses or overcrowding at certain stops. This issue is particularly noticeable during peak travel times, such as the afternoon shuttle buses returning from campus often experiencing overcrowding, especially during busy periods like exam times when a higher volume of students relies on the service at the same time. On the other hand, during

quieter periods, such as midday or non-peak days, buses may operate with low occupancy and waste resources. Without a seat reservation system or a more dynamic scheduling approach to better manage these fluctuations in demand, the transportation system struggles with inefficiencies, failing to effectively allocate resources based on the actual needs of the students and staff.

Implementing a mobile app for university bus systems aims to address current challenges by providing real-time tracking, clear schedules, and a seat reservation system. All students will be able to take the bus, but the reservation system will help track the number of students planning to use the service. This will allow the university to assign the right buses with the appropriate number of seats, ensuring efficient transportation and avoiding overcrowding or underutilization. The solution aligns with the university's commitment to technology, offering a more reliable and sustainable transportation option. It also provides an opportunity to apply academic knowledge in a practical context, while the system's scalability could benefit other institutions facing similar issues.

1.3 Importance & Objectives

- **Real-Time Tracking:** To enable users to track the location of university buses in real time for better planning and reduced wait times.
- **Accessible Schedules and Routes**: To provide easily accessible bus schedules and routes on the app, allowing users to plan their journeys efficiently.
- **Seat Reservation System**: To implement a system that tracks student reservations, enabling the university to assign the right number of buses with the appropriate seating capacity.
- **Improved User Experience**: To offer an intuitive, easy-to-use mobile app that enhances the overall transportation experience for students and staff.
- **Operational Efficiency**: To optimize bus usage by ensuring efficient allocation based on reservations, improving resource utilization.
- **Sustainability**: To promote shared transportation, reducing the environmental impact of individual travel on campus.

1.4 Competitive Advantage

The proposed system provides a competitive edge by combining several features not currently available in traditional university transport systems. Real-time tracking, seat reservations, and adaptive scheduling give users a reliable, tech-driven experience that rivals commercial ride-sharing services. The system also supports the university's green initiatives, positioning the institution as a leader in sustainable campus solutions. Additionally, the ability to scale and integrate the system with other services provides long-term strategic benefits, setting the university apart from competitors.

1.5 Target Users

- University students
- Faculty members
- Administrative and support staff
- Transportation management personnel

1.6 Inputs & Outputs

Inputs

- User registration and authentication
- Bus location data via GPS
- User-selected pickup and drop-off locations
- Reservation data (seats, times, dates)
- Admin data entry (routes, schedules)
- Feedback and user ratings

Outputs

- Real-time bus location on map
- Bus arrival/departure notifications
- Confirmed or waitlisted reservations
- Optimized scheduling suggestions for operators
- Usage analytics and reports for management

1.6.1 Immediate Outcomes

1. **Reduced Waiting Times:**
    a. Users can see bus locations in real-time, allowing them to arrive at stops just before the bus does.
2. **Improved Journey Planning:**
    a. Easy access to schedules and routes empowers users to plan their day effectively.
3. **Enhanced Communication:**
    a. Real-time notifications inform users of delays, cancellations, or schedule adjustments.

1.6.2 Long-term Outcomes

1. **Increased User Satisfaction:**
    a. Reliable transportation improves the daily experience of students, staff, and faculty.
2. **Operational Efficiency:**
    a. Data insights on bus usage patterns help optimize routes and schedules, reducing costs and enhancing resource allocation.
3. **Scalability:**
    a. The system can accommodate future growth, such as more buses or additional routes, and integrate with other campus services (e.g., parking or ride-sharing).
4. **Sustainability:**

a. Efficient route planning reduces fuel consumption and environmental impact.

1.7 System Functions

- Real-time bus tracking using GPS
- Seat reservation and cancellation system
- Schedule and route management
- Push notifications for updates or delays
- Admin dashboard for monitoring and data analysis
- Feedback and rating mechanism

1.8 System Features

- Intuitive mobile app interface
- Interactive map showing buses in motion
- One-click seat reservation
- Dynamic scheduling based on demand
- Cross-platform support (iOS, Android)
- Secure cloud-based data storage
- Multilingual support for diverse campus populations

1.9 SWOT Analysis

Strengths:

- **Real-Time Tracking:** The system provides accurate, live updates on bus locations, ensuring timely commutes for students and staff and reducing uncertainty.

- **Enhanced Safety and Accountability:** Enables parents, students, and administrators to monitor bus movements, improving safety and fostering accountability for drivers.
- **Alignment with Universities' Vision:** Demonstrates the universities' commitment to leveraging technology for practical, innovative solutions, aligning with its mission as a leading institution in informatics.
- **Customizable Design:** Can be tailored to accommodate each university's unique schedules, routes, and specific requirements, enhancing usability and effectiveness.
- **Improved Reputation:** Positions universities as a forward-thinking, tech-driven, attracting prospective students and stakeholders.

Weaknesses:

- **High Development and Maintenance Costs:** Significant investment is required for software development, GPS hardware installation, server management, and regular updates.
- **Reliance on Internet Connectivity:** The system's functionality depends heavily on stable internet connections; any disruption can affect usability.
- **Ongoing Maintenance Requirements:** Regular updates, troubleshooting, and equipment maintenance are essential to ensure system reliability, which may strain resources.

Opportunities:

- **Improvement in Campus Transportation:** Encourages the use of university-provided transportation, reducing reliance on private vehicles and minimizing traffic congestion on campus.
- **Expansion to Other Institutions:** Once implemented successfully, the system can be marketed to other institutions across Egypt.
- **Data-Driven Optimization:** Analysis of collected data can improve route planning, fuel efficiency, and scheduling, contributing to cost savings and operational efficiency.

- **Partnership Opportunities:** The project can attract collaboration with government transport authorities, private bus operators, or tech companies, enhancing its scalability and functionality.

Threats:

- **Technical Challenges:** Issues such as GPS inaccuracies, server downtime, or software bugs could undermine the system's reliability and user trust.
- **Increased Competition:** Similar systems developed by other universities or private entities could reduce the uniqueness of the system's offering.
- **Data Privacy Concerns:** Tracking systems may raise concerns about the security and ethical use of location and personal data, potentially deterring users.
- **Regulatory Barriers:** The project must comply with local transport regulations and data protection laws, which could pose implementation and operational challenges.

1.10 Scope of Implementation

This project will create a simple and efficient mobile app to help students, university staff, and bus drivers manage and track university bus services.

1.10.1 Key Features

- **Bus Reservations:** Students can easily book seats for their preferred routes and times.
- **Live Bus Tracking:** The app will show the real-time location of buses on a map, thanks to GPS integration.
- **Route & Schedule Info:** Clear and detailed information about bus routes, stops, and schedules will be available.
- **Notifications:** Students will get alerts about delays, changes in schedules, or when their bus is nearby.
- **Personalized Accounts:** Students can log in with their university credentials to manage bookings and see their history.

1.10.2 Who's It For?

- **Students:** Making daily commutes easier and more predictable.
- **University Administration:** Helping streamline bus operations and improve efficiency.
- **Bus Drivers:** Enabling smoother coordination with real-time updates.

1.10.3 Technical Details

- **GPS Integration:** Track buses in real-time for accurate location updates.
- **Responsive Design:** The app will work seamlessly on both Android and IOS.
- **Secure Database:** A reliable system to store user accounts, reservations, schedules, feedback, and medical checkup updates for the driver.
- **Analytics Dashboard:** Useful data insights to help the university monitor and optimize bus services.

1.10.4 Final Deliverables

- A fully functional and easy-to-use bus tracking app.
- Step-by-step user guides for students and staff.
- Training sessions to ensure smooth adoption by the university team.

1.10.5 Who's Not Included?

- **University-Specific Only:** This system is designed solely for universities' bus services, not for city transport.
- **Internet Dependency:** Real-time features will require a stable internet and GPS connection.
- **Limited Scope Initially:** The first phase will cover key routes, with expansion planned later based on demand.

Chapter 2

Project Planning

2.1 Project planning introduction

Phase 1: Analysis and Requirements Gathering (Weeks 1–3)

1. **Conduct Stakeholder Interviews:**
   - Interview students, staff, faculty, and bus drivers to understand their pain points and expectations.
2. **Assess Current System:**
   - Analyze existing transportation schedules, routes, and user engagement with available information.
3. **Define Requirements:**
   - List required features for real-time tracking, user notification systems, and route optimization.

Phase 2: System Design and Planning (Weeks 4–6)

1. **Develop System Architecture:**
   - Plan a GPS-based real-time tracking solution integrated with a mobile/web application.
2. **Prepare Schedule and Route Database:**
   - Digitize and optimize current bus schedules and routes for integration.
3. **Create Prototypes:**
   - Design prototypes of the user-facing platform for feedback.

Phase 3: Development and Implementation (Weeks 7–12)

1. **Develop Real-Time Tracking System:**
   - Implement GPS devices and integrate them with a central database.
2. **Build User Platforms:**
   - Develop and test mobile and web applications with functionalities like schedule access, live tracking, and notifications.

3. **Integrate Data and Perform Testing:**
    - Test end-to-end functionality, including accurate GPS updates and user-friendly interfaces.

Phase 4: Deployment and Monitoring (Weeks 13–16)

1. **Launch Pilot Program:**
    - Roll out the system for selecting routes to gather real-world performance data.
2. **User Training and Awareness Campaigns:**
    - Train users on how to use the platform effectively.
3. **Monitor and Optimize:**
    - Collect user feedback and optimize based on performance data.

2.2 Task table

First Semester: Planning Phase

This phase focuses on analysis, requirements gathering, and design. It ensures a solid foundation for implementation.

Table 2: Task Table

Task	Start Date	End Date	Duration (Weeks)	Dependencies
Stakeholder Interviews	2024-12-01	2024-12-14	2	None
Assess Current System	2024-12-01	2024-12-14	2	None
Define Requirements	2024-12-15	2024-12-21	1	Previous
Develop System Architecture	2024-12-22	2025-01-04	2	Previous
Prepare Schedule Database	2024-12-22	2025-01-11	3	Previous

Second Semester: Implementation Phase

This phase involves building, testing, and deploying the system.

Task	Start Date	End Date	Duration (Weeks)	Dependencies
Develop Tracking System	2025-01-12	2025-02-08	4	Planning
Test and optimize	2025-02-09	2025-02-22	2	Previous
Pilot Launch	2025-02-23	2025-03-07	2	Testing
Full Rollout and Monitoring	2025-03-08	2025-03-21	2	Pilot

Gantt Chart

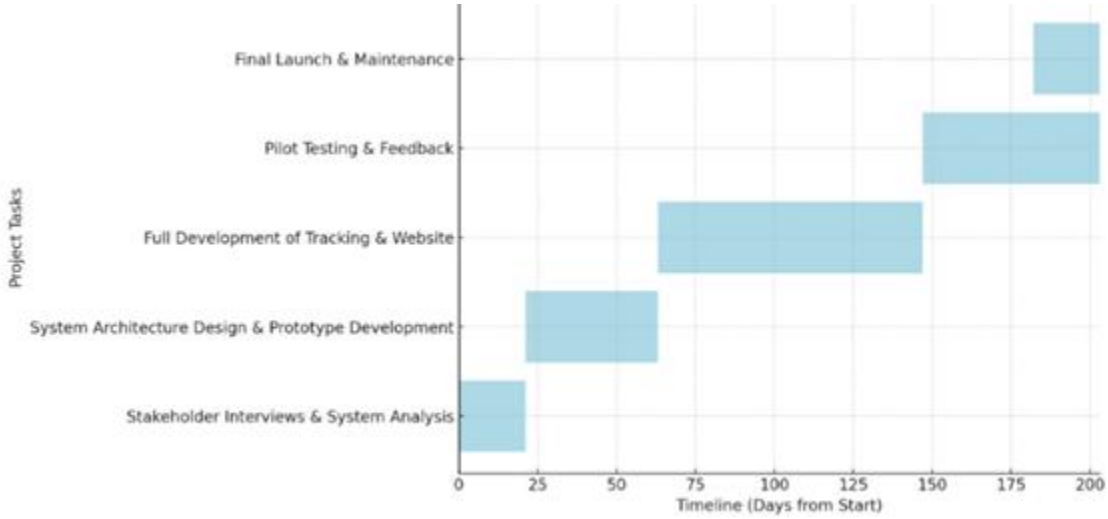


Figure 1: Gantt Chart

Gantt Chart Analysis for Project Timeline

X-Axis (Horizontal):

- Represents **time in days** from the start of the project.
- The scale goes from **0 to 200 days**.

Y-Axis (Vertical):

- Lists the **key project tasks** in sequence from bottom to top, showing their execution order

Task Breakdown with Timelines

1. **Stakeholder Interviews & System Analysis:**
    a. **Start:** Day 0
    b. **End:** ~Day 30
    c. **Duration:** ~30 days
    d. **Description:** Involves meeting with stakeholders (e.g., students, transport administrators) to gather requirements and analyze current transportation challenges.
2. **System Architecture Design & Prototype Development:**
    a. **Start:** ~Day 25
    b. **End:** ~Day 60
    c. **Duration:** ~35 days
    d. **Overlap:** Begins before the previous task ends (parallel execution for faster delivery).
    e. **Description:** Includes designing the app's technical architecture and developing initial wireframes or mockups.
3. **Full Development of Tracking & Website:**
    a. **Start:** ~Day 65
    b. **End:** ~Day 145
    c. **Duration:** ~80 days
    d. **Description:** Actual implementation of frontend (mobile app), backend systems, APIs, and Google Maps integration. Development covers features like live tracking, reservations, and user management.
4. **Pilot Testing & Feedback:**
    a. **Start:** ~Day 145
    b. **End:** ~Day 185
    c. **Duration:** ~40 days

d. **Description:** A soft launch to a limited audience (e.g., a few buses or departments). Collects feedback and identifies bugs or improvement areas.

5. **Final Launch & Maintenance:**
   a. **Start:** ~Day 180
   b. **End:** Day 200
   c. **Duration:** ~20 days
   d. **Overlap:** Overlaps with the end of the pilot phase.
   e. **Description:** Marks the official deployment for all users. Includes final bug fixes, performance optimization, and preparation for ongoing maintenance.

2.3 Marketing

1. Target Audience

a. **Primary Customers:** University administration, especially transportation/logistics departments.
b. **End Users:** Students, faculty, bus drivers
c. **Secondary Stakeholders:** IT departments, university boards, government education ministries (for public universities)

2. Marketing Messages

Key benefits:

a. Reduce student wait times
b. Prevent overcrowding
c. Optimize bus use and save operational costs
d. Real-time tracking and notifications
e. Scalable and cross-platform

3. Marketing Materials

a. **Professional Presentations:** Include demo videos, use cases, pricing plans, and testimonials.
b. **Pitch Deck:** Tailored for university board presentations.
c. **Explainer Video (1–2 mins):** Showing how the app works.

d. **Printable Brochures & Posters:** For on-campus placement.
e. **Social Media Content:** Share benefits, sneak peeks, student testimonials, and launch updates.

4. University Outreach Strategy

a. **Email Campaigns:** Contact deans, IT directors, and transportation managers.
b. **Cold Calls/Visits:** Schedule meetings or demos with university officials.
c. **Offer Free Pilot Program:** Let one university use it free for 3–6 months.
d. **Attend Educational/Tech Fairs:** Set up booths at university expos or education summits.

5. Online Marketing

a. **LinkedIn:** Target university professionals and education leaders
b. **YouTube Ads:** Short clips before videos related to campus life or transportation
c. **Google Ads:** For keywords like “university transportation system”, “bus tracking app for schools”
d. **SEO Blog Content:** Topics like "How to modernize university transport" or “Benefits of student bus reservation systems”

6. Engage Students

a. **Ambassador Program:** Let students promote it in return for perks or certificates
b. **In-App Referral Rewards:** "Get a free ride credit if you invite 5 friends to sign up"
c. **Social Challenges:** Run giveaways or contests like “Snap your Bus Ride & Tag Us”
d. Collect Testimonials & Case Studies

7. After Successful Pilots or Adoption, Gather:

a. Student feedback
b. Admin reviews
c. Performance data (e.g., reduced wait times, increased ridership)
d. Use this to build trust for future university clients.

e. Launch Plan

f. Choose a university semester start (September) for your big rollout

8. Plan a Multi-Media Launch Campaign

a. App launch event on campus

b. Email blasts

c. In-class announcements

d. Collaborations with student unions

2.4 Organization Chart

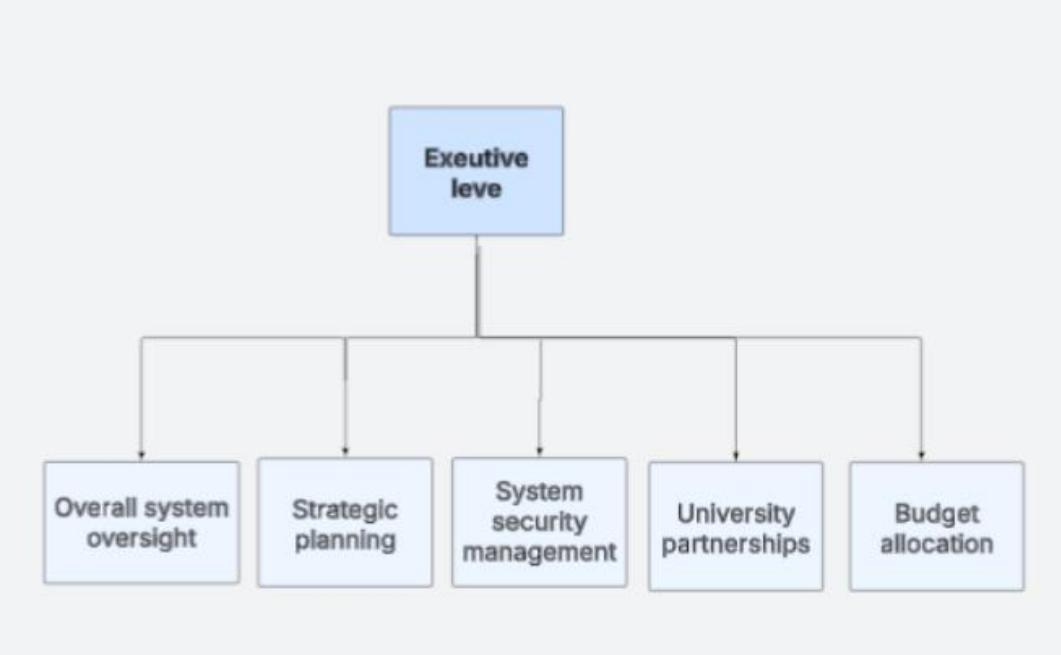


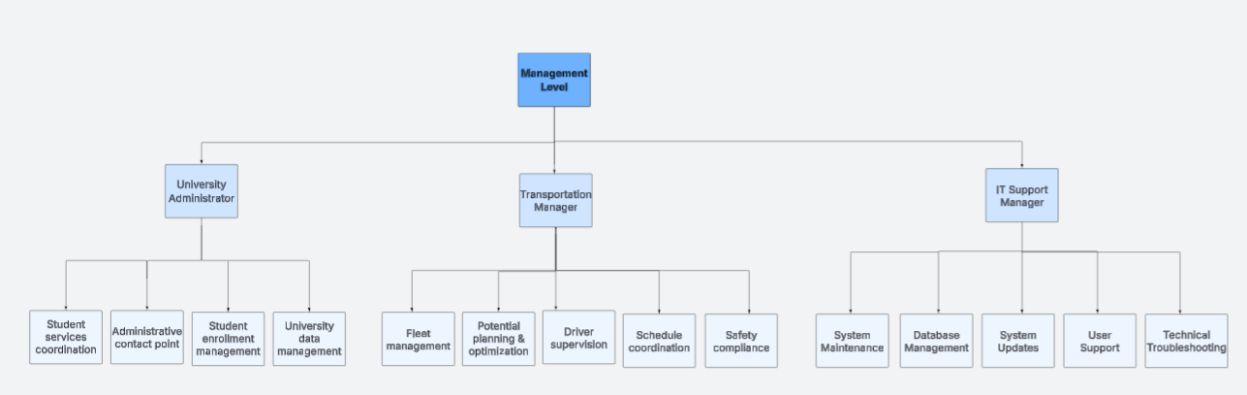
Figure 2: Organizational Chart

2.4.1 Executive Level

Roles: System Administrator, Chief Technology Officer (CTO)

At this highest tier, leadership is responsible for the **big picture** of the entire UniTracker platform:

- **Overall System Oversight:**
  Ensures all components (hardware, software, networks) work together reliably and at scale.
- **Strategic Planning:**
  Defines long-term goals. e.g., expanding partnerships, adopting emerging technologies, or entering new markets.
- **System Security Management:**
  Sets security policies, oversees risk assessments, and ensures compliance with data-protection regulations.
- **University Partnerships:**
  Cultivates relationships with university stakeholders, negotiates service agreements, and aligns system functionality with institutional needs.
- **Budget Allocation:**
  Decides how funds are distributed across development, operations, maintenance, and future initiatives.



2.4.2 Management Level

This middle layer translates executive strategy into day-to-day programs. There are three distinct managerial roles:

University Administrator (Campus Operations)

- **Student Enrolment Management:**
  Coordinates how student data flows into the system, ensuring accuracy of registrations.
- **Administrative Contact Point:**
  Acts as the liaison between campus leadership and the UniTracker team.
- **University Data Management:**
  Maintains the integrity of university-specific datasets (e.g., student IDs, course schedules).
- **Student Services Coordination:**
  Aligns bus tracking with other student services (health, counselling, events).

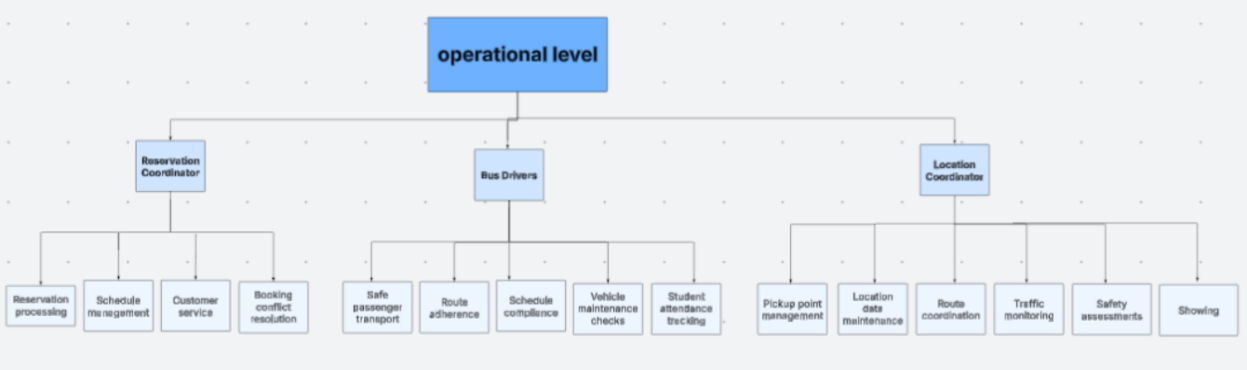
Transportation Manager (Fleet Operations)

- **Fleet Management:**
  Oversees bus assignments, maintenance schedules, and vehicle availability.
- **Route Planning & Optimization:**
  Uses ridership data to refine routes for efficiency and on-time performance.
- **Driver Supervision:**
  Hires, trains, and evaluates bus drivers to maintain service quality.
- **Schedule Coordination:**
  Aligns bus timetables with academic calendars, exam periods, and special events.
- **Safety Compliance:**
  Ensures all operations meet regional transport safety standards.

IT Support Manager (Technical Operations)

- **System Maintenance:**
  Schedules regular checks, handles backups, and applies patches.
- **Database Management:**
  Optimizes database performance and reliability.

- **User Support:**
  Runs helpdesk services, FAQs, and troubleshooting guides.
- **System Updates:**
  Manages version control, rollouts of new features, and change-control processes.
- **Technical Troubleshooting:**
  Quickly diagnoses and resolves outages or performance bottlenecks.



2.4.3 Operational Level

Front-line staff who execute the core functions:

Bus Drivers (Field Operations)

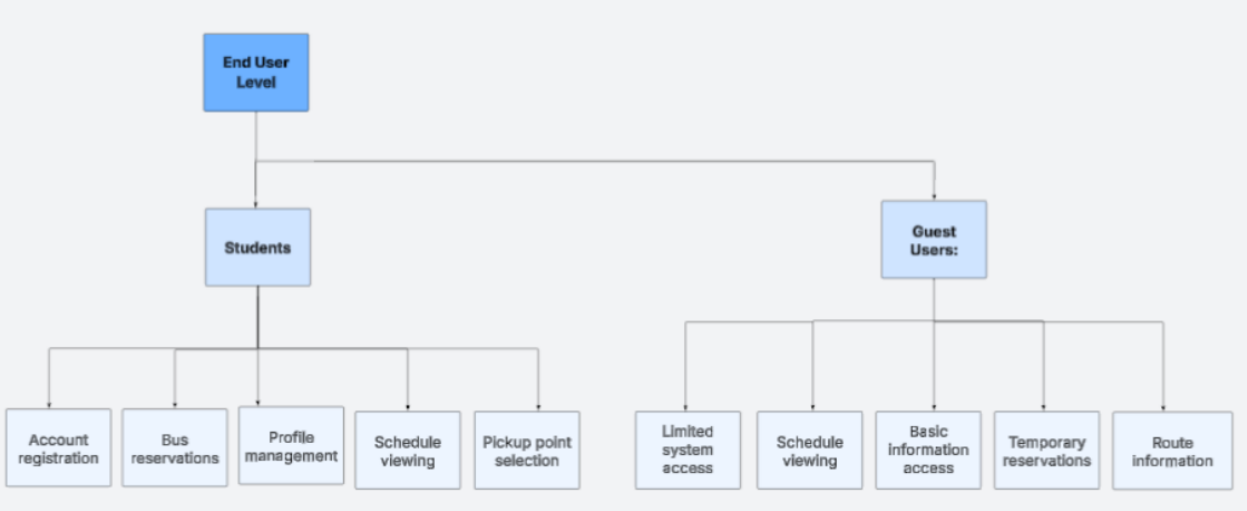
- **Safe Passenger Transport:**
  Adheres to traffic laws and university safety protocols.
- **Route Adherence & Schedule Compliance:**
  Follows assigned routes and keeps to the timetable.
- **Vehicle Maintenance Checks:**
  Conducts pre- and post-shift inspections to spot issues early.
- **Student Attendance Tracking:**
  Records who boards and disembarks for capacity planning and security.

Reservation Coordinator (Booking Management)

- **Reservation Processing:**
  Confirms and logs each student's seat reservation.
- **Schedule Management:**
  Updates availability as buses fill or cancel.
- **Customer Service:**
  Handles student inquiries, changes, and cancellations.
- **Booking Conflict Resolution:**
  Manages overbookings or system errors swiftly to minimize disruption.

Location Coordinator (Pickup Point Management)

- **Pickup Point Management:**
  Defines and updates geolocations for safe and convenient stops.
- **Location Data Maintenance:**
  Ensures map data stays current (e.g., new gates, construction detours).
- **Route Coordination:**
  Works with the Transportation Manager to integrate stops into optimized routes.
- **Traffic Monitoring & Safety Assessments:**
  Surveys pickup areas to minimize hazards and delays.



2.4.4 End-User Level

The system's ultimate beneficiaries:

Students (System Users)

- **Account Registration & Profile Management:**
  Sign up, update personal info, and manage login credentials.
- **Bus Reservations & Schedule Viewing:**
  Search routes, book seats, and check live bus locations.
- **Pickup Point Selection:**
  Choose the most convenient stop from a campus map.

Guest Users (Visitors & Temporary Users)

- **Limited System Access:**
  View public schedules without full account privileges.
- **Temporary Reservations:**
  Make one-time bookings (e.g., campus tours, events).
- **Route Information Access:**
  Check bus routes and times without creating an account.

2.4 Project Overview

The system helps universities manage their bus services for students who need rides to and from campus. It works like a simple app where students can book seats on university buses, choose where they want to be picked up, and see bus schedules. The system also helps bus drivers know their routes and keeps track of all the buses, including how many seats are available and where each bus is located. University staff can use it to manage everything - from adding new bus routes to keeping driver information up to date. The project covers multiple schools at once, so it can handle buses going to different universities in the same area. Students just need to sign up with their student ID, pick their pickup location, and reserve a seat for the day they need it. The system makes sure everything runs smoothly by organizing schedules, tracking which buses are where,

and making sure students get reliable transportation to their classes. It's basically designed to solve the everyday problem of getting students safely and efficiently to and from their universities using organized bus services.

2.4.1 Project Burndown

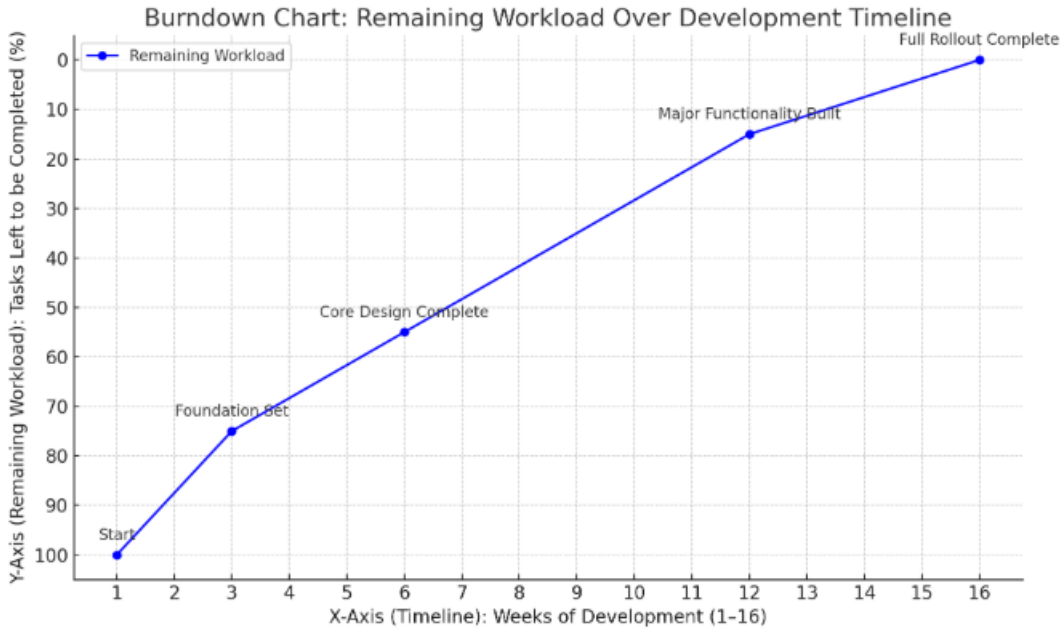


Figure 3: Project Burndown

Table 3: Project Burndown Phases

Phase	Tasks	Timeline	Expected Work Completion (%)
Analysis & Requirements Gathering	Finding the gap in the market & doing research	1 to 3 weeks	25% foundation is set
System Design & Planning	Architecture development, database setup & prototyping	4 to 6 weeks	45% core design complete
Development & Implementation	GPS integration, app interface creation & testing	7 to 12 weeks	85% major functionality built
Deployment & Monitoring	Pilot launch, training, feedback, optimization	13 to 16 weeks	100% full rollout complete

2.4.2 Cost Overview

Table 4: Cost Overview

Category	Item	Estimated Cost (EGP)
Personnel Costs	Frontend/Backend Developer	45,000
	UI/UX Designer	30,000
	Project Manager	30,000
	Quality Assurance Tester	9,000
	SUBTOTAL	**114,000**
Software & Tools	Node-red, AWS Hosting, APIs, Testing	60,000
	Real-Time Tracking API	90,000/Year
	Google Maps API, Domain, SSL, etc.	12,000
	SUBTOTAL	**162,000**
Miscellaneous	Training, Documentation, Legal, etc.	24,000
	TOTAL ESTIMATED COST	**300,000**

Chapter 3

Feasibility Study

3.1 Technical Feasibility

3.1.1 Technology Requirements

The project leverages modern, cross-platform development technologies to ensure broad accessibility and maintainability. The frontend application is built using **Flutter** and **Dart**, enabling a single codebase for both Android, iOS, and web platforms. For backend services, the project utilizes **Supabase**, an open-source Backend-as-a-Service (BaaS) that provides authentication, real-time database, storage, and serverless functions. **Google Maps API** is integrated for real-time bus tracking and route visualization. State management within the app is handled using the **Provider** package, ensuring scalable and maintainable architecture. The application is fully responsive and optimized for both mobile devices and tablets. Hosting and backend infrastructure are managed through Supabase and cloud platforms, ensuring scalability, security, and reliability. All chosen technologies are well-documented, widely adopted, and align with the team's expertise.

3.1.2 Resource Availability

All required software tools and platforms (Flutter, Dart, Supabase, Google Maps API) are open-source or offer generous free tiers for development. The development team has experience with these technologies, and there is extensive community support and documentation available to address any knowledge gaps.

3.1.3 Hardware Requirements

Development and testing can be performed on standard laptops or desktop computers with a stable internet connection. No specialized hardware is required, making the project accessible and cost-effective for the team.

3.2 Revenue Model

The platform can generate revenue through:

- **Subscription Fees:** Universities pay a fee per semester to use the service.
- **3 Packages Available:**
  - Trial Package: 0 to 5 buses = EGP 50,000
  - Classic Package: 20 to 35 buses = EGP 175,000
  - Premium Package: 35 to unlimited = EGP 250,000
- These prices will help in terms of scalability and reach our target, which is 200 buses subscribed to us.

3.2.1 Revenue Projections per Semester

Semester-Based Revenue

- **Number of Buses:** 20 (Scaling to 200 in 3 years).
- **Number of Universities:** 1 (Scaling to 6 in 3 years).
- **Average Subscriptions per Semester:**
  - **(Trial Package):** EGP 50,000 per Semester / 5 Months = EGP 10,000
  - **(Classic Package):** EGP 175,000 per Semester / 5 Months = EGP 35,000
  - **(Premium Package):** EGP 250,000 per Semester / 5 Months = EGP 50,000

3.2.2 Break-Even Analysis

Fixed Costs (Annually)

- Launching Cost (App Development & Testing): 190,000 EGP
- Subscriptions 100,000 EGP ($4000)

Break-Even Revenue / Break-Even Package (Semester)

- 2 of the Classic Package = EGP 175,000 x 2 = 350,000 EGP

3.2.3 Payback Period Calculation

1. **Initial Investment:**
   - 400,000 EGP
2. **Cumulative Cash Flow for Year 1 and Year 2:**

Year 1:

- **Total Net Cash Flow for Year 1:**
  EGP (50,000) + EGP 65,000 = EGP 15,000

Year 2:

- **Net Cash Flow in Year 2:**
  Starting from EGP 15,000 (First Semester)

- **In Year 1**, First Semester, the Trial Package was sold to one university for **50,000 EGP/Semester.**
- **In Year 1**, Second Semester, the Classic Package was sold to one university for **175,000 EGP/Semester.**
- **In Year 2**, First Semester, we added two universities to the Trial Package and continued with the Classic Package with the Launching University
  = (50,000 x 2) = 100,000 + 175,000 = EGP 275,000
- **In Year 2**, Second Semester, the three universities upgraded to the Classic Package **= 175,000 x 3 = EGP 525,000** Which **Exceeds the Initial Investment of EGP 400,000**

3.3 Operational Feasibility

3.3.1 Integration

The system will be designed to integrate with existing university processes easily. Universities will need to provide bus schedules, routes, and event details for setup. No major changes to the current infrastructure will be required.

3.3.2 Staffing Needs

The system will be designed to be user-friendly, and no significant training is needed. University staff can operate it with minimal preparation.

3.4 Legal Feasibility

1. **Limited Liability Company (LLC):**

   The app will be registered as a limited liability company, which is suitable for startups. The registration process will include:

   - **Prepare the Required Documentation:**

     The necessary documents, including founders' IDs, Articles of Incorporation, and initial capital proof, will be gathered. These will outline the company's purpose, ownership, and management roles.

   - **Reserve the Company Name:**

     A name reservation request will be submitted to GAFI, ensuring the name is unique and aligned with the app's purpose. Once approved, we'll pay the reservation fee.

   - **Draft the Company Contract:**

     With legal assistance, the company contract will be drafted, detailing the company structure, roles, and ownership. It will be authenticated by a Notary Public.

   - **Register the Company with GAFI:**

     All required documents will be submitted to GAFI and pay registration fees. After approval, we will receive the Commercial Registration Certificate.

   - **Open a Corporate Bank Account:**

     We will open a corporate bank account using the registration certificate and deposit the starting capital, obtaining proof of the deposit.

   - **Obtain Tax and Social Insurance Numbers:**

     We will register the company with the Egyptian Tax Authority for a Tax ID and with the Social Insurance Authority as we will hire employees.

   - **Obtain Any Required Licenses or Permits:**

     We will apply for any necessary licenses related to transportation services to ensure full compliance with local regulations.

2. **Registration with GAFI (General Authority for Investment and Free Zones):**

   - Prepare required documents
   - Application submission

- Company registration certificate

3. **Tax and Commercial Registration:**
    - **Tax Registration:** Obtain a **Tax Card** by registering with the **Egyptian Tax Authority**.
    - **Commercial Registry Certificate**: Register the company with the **Commercial Registry Office** under the Ministry of Supply and Internal Trade.
4. **Intellectual Property (IP) Protection:**
    - **Trademark:** Register the Brand and App Technology, file a trademark application with the **Egyptian Patent Office** to protect the app's name, logo, and unique features.
    - **Copyright:** Register copyrights for the app's software code and design.
5. **Compliance with Data Protection Laws:**

    Adhere to the Egyptian Data Protection Law (Law No. 151 of 2020)
    - Establish robust systems to protect user data, including GPS tracking and personal information.
    - Draft a **Privacy Policy** for the app, clearly explaining how data is collected, stored, and processed.
    - Submit the data protection policy to the **Data Protection Center** under the Ministry of Communications and Information Technology (MCIT) for compliance review.
6. **App Security Compliance:**
    - **Implement Security Protocols**: **Use SSL/TLS encryption** for secure data transmission between users and servers.
    - **Regular Security Audits**: **Conduct regular security assessments** to identify and mitigate vulnerabilities.
    - **User Authentication**: **Incorporate multi-factor authentication (MFA)** for secure user logins.
    - **Data Encryption**: **Encrypt sensitive data**, including personal information and GPS tracking, both at rest and in transit.
    - **Incident Response Plan**: **Develop a plan to manage potential breaches**, ensuring quick recovery and legal compliance.

7. **Sector-Specific Licensing:**

   Coordinate with Transportation Authorities

   - Seek permissions from local transportation departments for providing services related to bus tracking.
   - Partner with universities by formalizing contracts for transportation support and data-sharing agreements.

8. **Social Insurance and Labor Law Compliance:**

   - **Register Employees**: Ensure compliance with Egypt's Labor Law No. 12 of 2003, covering working hours, contracts, and employee benefits.
   - **Issue Employment Contracts**: Draft contracts specifying roles, responsibilities, and rights in line with Egyptian labour regulations.

9. **Bank Account Creation:**

   - Open a corporate bank account in the company's name to facilitate business transactions, including receiving payments from universities.

3.5 What is the Importance of a Feasibility Study?

The feasibility analysis is **crucial** because it provides a structured, evidence-based evaluation of whether the Bus Tracking and Reservation System should move forward into development and implementation. It is important for the following key reasons:

1. **Reduces Risk:**

   By examining technical capabilities, financial expectations, operational logistics, and legal obligations, the analysis identifies potential risks early and proposes solutions. This helps avoid costly mistakes and delays later in the project.

2. **Builds Stakeholder Confidence:**

   Investors, university partners, and development teams can make informed decisions based on real data, revenue projections, and compliance strategies. This transparency builds trust and increases the likelihood of support and funding.

3. **Ensures Smart Resource Allocation:**

   The analysis ensures that time, money, and human resources are invested in a project

that is viable and scalable. It helps prioritize what technologies, personnel, and infrastructure are truly needed.

4. **Confirms Market Demand and Profitability:**
   The economic analysis proves that the system can generate revenue through well-structured subscription packages. The break-even and payback period analysis shows that the investment can be recovered quickly with a strong potential for long-term growth.
5. **Supports Strategic Planning:**
   With detailed cash flow projections, a phased rollout plan, and legal requirements mapped out, the feasibility analysis functions as a **roadmap** for the project's development, launch, and expansion.
6. **Demonstrates Legal and Regulatory Readiness:**
   The legal feasibility section ensures that the project aligns with national laws and industry regulations, avoiding legal obstacles and penalties that could jeopardize the business.

3.6 Software Pricing Table

The project uses a range of open-source and affordable tools, enabling development and deployment with minimal software costs:

3.6.1 Dart & Flutter

Both are open-source and completely free to use for developing, testing, and deploying the application.

3.6.2 Supabase

Supabase is used for backend services, including authentication, database, storage, and serverless functions. It provides a generous free tier that covers our current needs. As the project grows, we can upgrade to paid plans with predictable, usage-based pricing.

3.6.3 Google Maps API

The app integrates Google Maps for location features. Google offers a free monthly usage quota, which covers our current usage. Additional usage is charged based on map loads and API requests.

3.6.4 Provider (State Management)

We use the Provider package for efficient state management. It is open-source and free to use.

3.6.5 Other Tools

We also use packages like shared_preferences for local storage and intl for internationalization. All these tools are open-source and free.

3.7 Cash Flow

3.7.1 Year 1

Table 5: Cash Flow

Semester	Revenue (EGP)	Expenses (EGP)	Net Cash Flow (EGP)	Cumulative Cash Flow (EGP)
First	50,000	100,000	(50,000)	(50,000)
Second	175,000	110,000	65,000	15,000

3.7.2 Year 2

Semester	Revenue (EGP)	Expenses (EGP)	Net Cash Flow (EGP)	Cumulative Cash Flow (EGP)
First	275,000	120,000	155,000	170,000
Second	525,000	120,000	405,000	575,000

3.7.3 Year 3

Semester	Revenue (EGP)	Expenses (EGP)	Net Cash Flow (EGP)	Cumulative Cash Flow (EGP)
First	675,000	130,000	545,000	1,120,000
Second	1,050,000	130,000	920,000	2,040,000

Chapter 4

System Analysis & Design

4.1 What is System Analysis?

System analysis is the process of studying a system in detail to understand how it works and how its components interact. It involves examining the system's inputs, processes, outputs, and environment to identify problems or opportunities for improvement. This process helps in designing or improving information systems to meet business goals effectively. System analysts use various tools and techniques to gather data, model system processes, and propose optimized solutions.

4.2 What is the Importance of System Analysis?

System analysis is important because it helps organizations understand their current systems thoroughly, allowing for better decision-making and system improvements. It ensures that new or modified systems meet user requirements, are cost-effective, and operate efficiently. By identifying inefficiencies and proposing enhancements, system analysis minimizes errors, saves resources, and supports the development of reliable and scalable systems that align with business needs.

4.3 Context Diagram

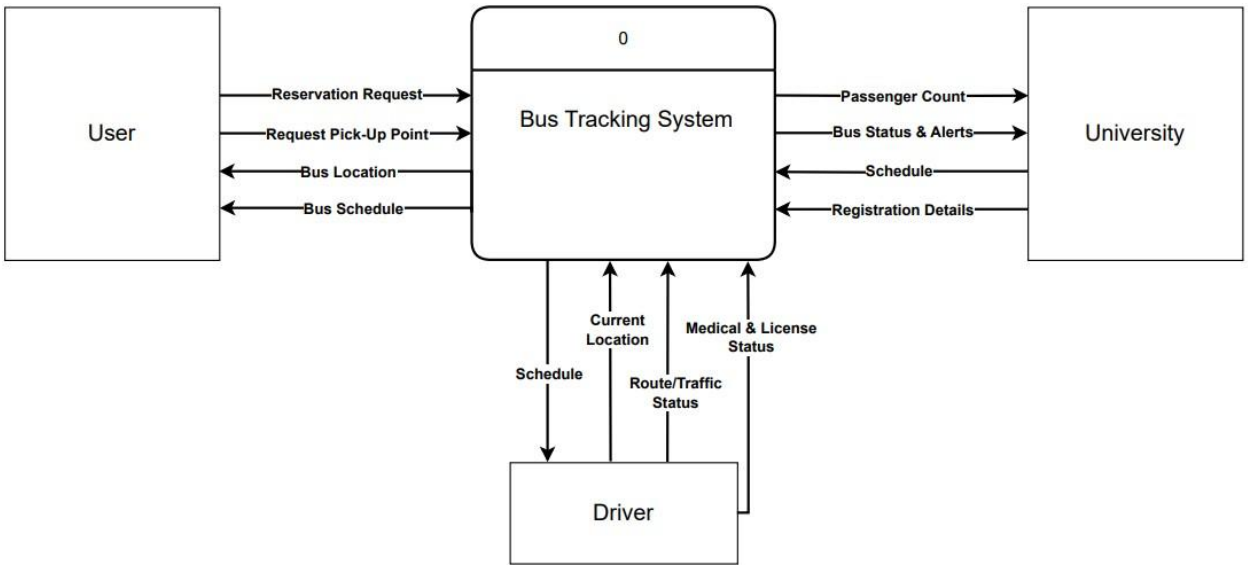


Figure 4: Context Diagram

The context diagram provides an overview of the **Bus Tracking System** and its interactions with external entities: **Users, Universities, and Drivers.** The user is anyone eligible to use the bus, such as students, faculty members, staff, or guests. It highlights the major data flows between the system and these entities and clearly explains the system's scope and boundaries.

4.3.1 External Entities

a. User:

- Interacts with the system to send a **Reservation Request** specifying their intention to book a seat on a university bus.
- Provide their **Pick-Up Point** to facilitate scheduling.
- Receives key information from the system, including:
    - **Bus Location:** The real-time position of the bus.
    - **Bus Schedule:** Timetable details to plan their journey.

b. University:

- Provides the system with **Registration Details** for authorized users and buses.
- Provides the **Schedule** of buses for integration into the system.
- Receives **Passenger Count** data, helping the university monitor bus utilization.
- Receives **Bus/Route Status** for updates on bus/route conditions, delays, or incidents.

c. **Driver:**

- Updates the system with the bus's **Current Location**, enabling real-time tracking for users and the university.
- Communicates **Bus/Route Status**, which helps the system calculate accurate arrival times and optimize routes.
- Updates the system with its License information, to verify driver's credentials.
- Receives the **Schedule** from the system, detailing their assigned routes and timings.
- Receives **Medical & License** checkup date from the system, in order to update it.

4.3.2 Central System

a. The **Bus Tracking System** serves as the core entity, processing input from the external entities and generating output to meet their requirements.
b. Facilitates seamless communication between all parties, ensuring effective tracking, scheduling, and management of university bus services.

4.3.3 Key Features

a. The system acts as a central hub for data collection and dissemination, ensuring real-time updates and efficient operations.

b. Its design ensures all stakeholders, users, the university, and drivers are informed and synchronized.

4.4 DFD

1. Entities:

The DFD shows the following entities that interact with the system:

- **User:** Represents universities' students, faculty, staff, and guests who utilize the system to access bus information, track buses, view bus schedules, and make reservations. This is the passenger entity in the system.
- **Location:** Represents the source of real-time location data.
- **University Admin:** Represents the administrative personnel or facilities department responsible for managing bus routes, schedules, and the overall system at the university.
- **Driver:** Represents the bus drivers who interact with the system for route information, location updates, and medical & License status.

2. Major Processes:

The DFD identifies these ten major processes:

- **1.0 Select University:** Primary users need to identify which university they relate to as the system will serve many universities.
- **2.0 Sign up/Login:** manages user registration and authentication for each university using the system.
- **3.0 Reservation:** allows users to book seats on buses. The system uses this information to know how many people plan to use each route and adjust bus availability accordingly.
- **4.0 Bus Route & Schedule:** enable the user to view when and where the buses travel, so they can plan their trips.
- **5.0 Bus Tracking:** It shows users exactly where the buses are on a map and the remaining time for the bus to get to each stop, so they know when to expect them.
- **6.0 Manage Route & Schedule:** University admin uses this to set the bus routes and times, ensuring the system is up to date.

- **7.0 View route & Schedule:** enables the drivers to see their assigned routes and times. It ensures they're aware of their duties.
- **8.0 Bus status:** This process receives and manages the live location of each bus, updating the data store so users can see where the buses are.
- **9.0 Medical & License Status:** This is for maintaining the medical & license information of each driver to make sure they are fit to drive.
- **10.0 Feedback:** Collects feedback from users and drivers, if any of them want to report a problem about their travel/bus.

3. System-Wide Interactions:

a. Real-Time Data Handling:

- The "Bus Tracking" and "Bus Status" processes are central to the system's real-time capabilities, integrating with Location updates and GPS data.

b. User Interaction Points:

- The "User" entity interacts primarily with processes such as Select University, Sign up/Login, Reservation, and Bus Route & Schedule, to access information and perform actions within the system.

c. Administrative Control:

- The “University Admin" entity has direct control over the "Manage Route & Schedule" process, reflecting the administrative role in the system.

d. Driver Role:

- The "Driver" entity interacts with the "Bus Route & Schedule," "Bus Status," and "Medical & License Status" processes for updating and retrieving relevant details.

e. Continuous Improvement:

- The "Feedback" process is designed to enable continuous improvement to the system with the data collected from users and drivers.

4.4.1 Data Flow (0-Level Diagram)

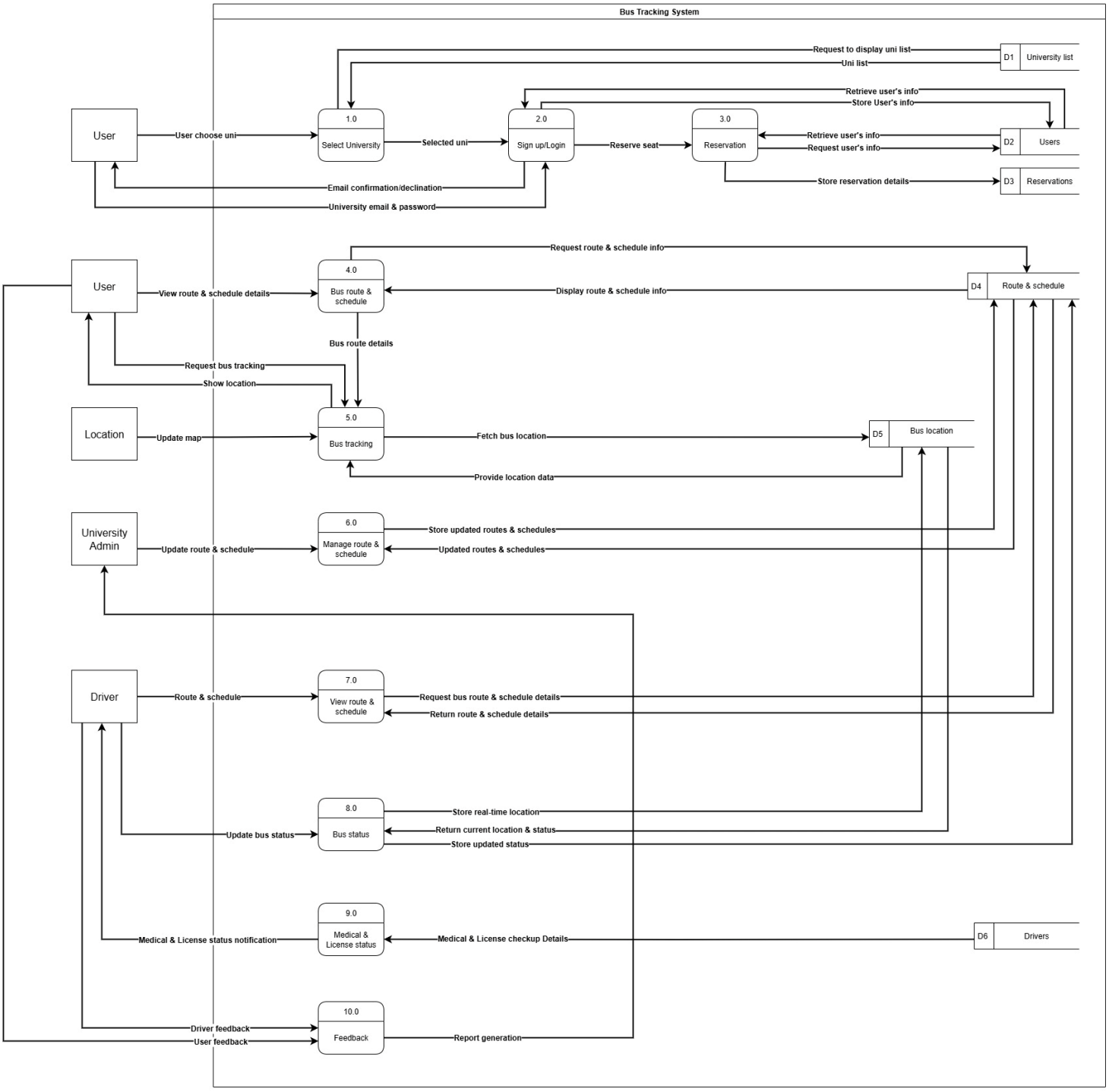
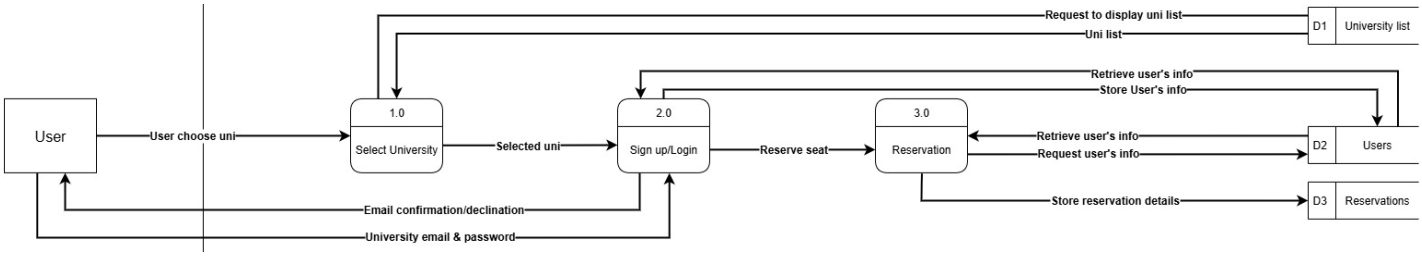


Figure 5: Data Flow (0-Level Diagram)

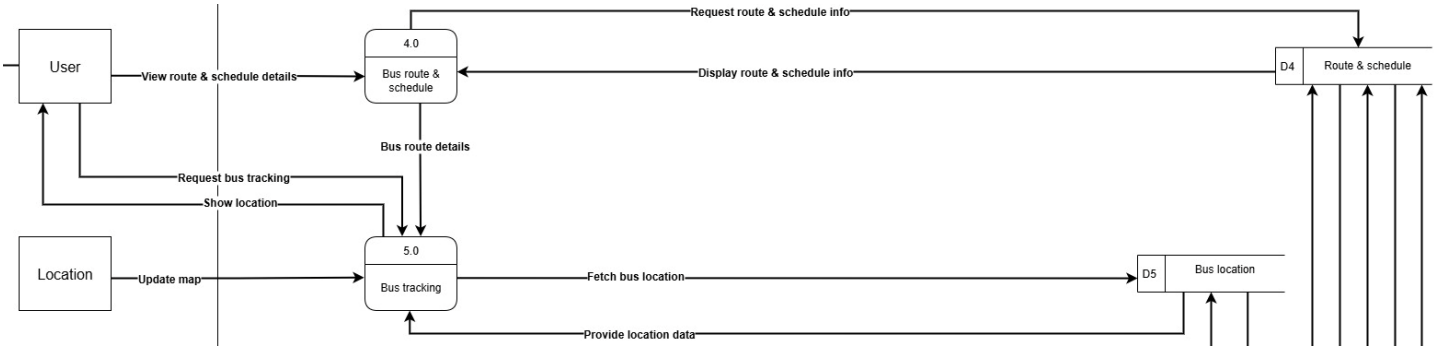
This level 0 DFD provides a high-level view of UniTracker. The diagram illustrates the primary entities and their interactions with the system, highlighting the entities, major processes, their inflows and outflows, updates to data stores, and system-wide interactions without detailing internal process logic.

4. Inflows and Outflows for Each Process:

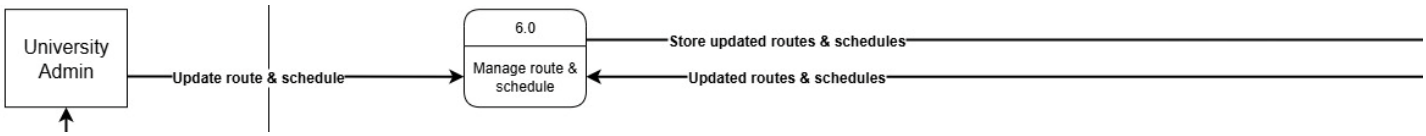


- **Select University:**
  - Inflows:
    - **"User choose uni"** the very first step, User input selecting their university from the displayed list.
    - **"Uni List"** the data store provides the list of universities, so the user can choose which one he wants to sign for.
  - Outflows:
    - **"Selected Uni"** (the chosen university) This data flow carries the specific university chosen by the user and informs the rest of the system of which university the user has chosen.
    - **"Request to display uni list"** A request to retrieve the list of available universities from the database, to present it to the user.

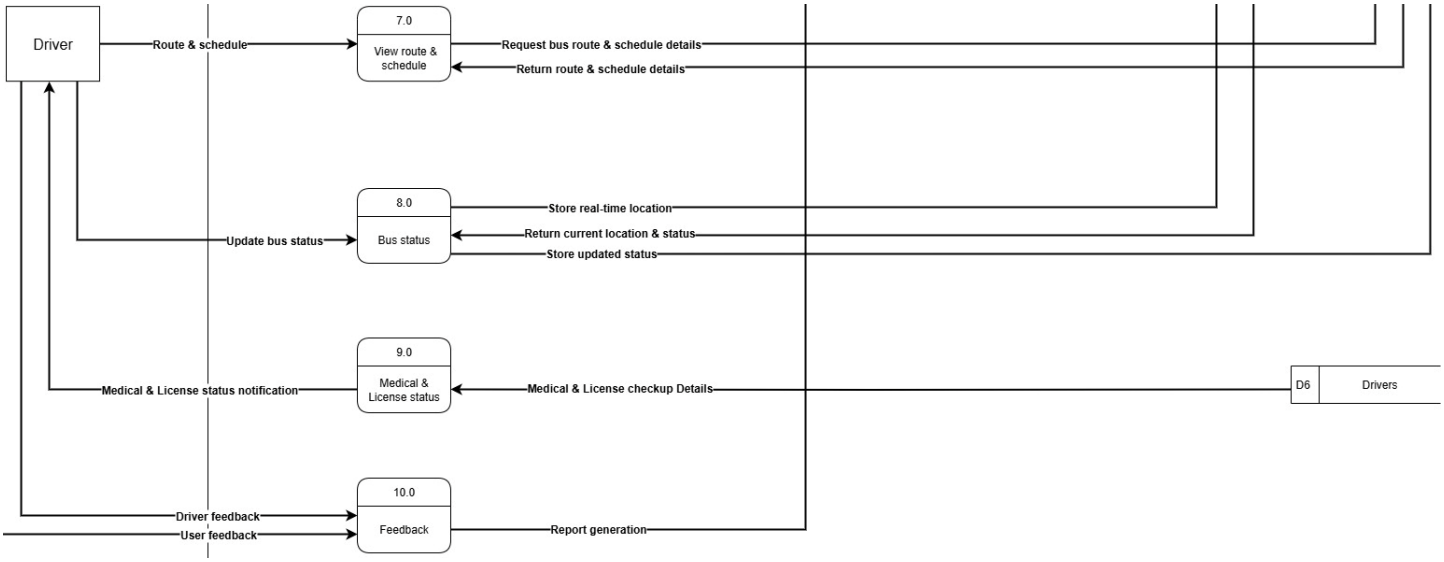
- **2.0 Sign up/Login:**
    - Inflows:
        - **“Selected uni”** The university the user chose, which is needed to process the login
        - **“University email & password”** User's credentials for their chosen university. It's how the system verifies who they are and if they have access to the university, they're from.
        - **“Retrieve user’s info”** the system retrieves a returning user's details for login.
    - Outflows:
        - **"Email confirmation/declination"** System's response to registration or login attempt. It's the system telling you, "Yes, you're logged in" or "No, your credentials aren't correct”.
        - **"User's info"** This is information the user provides while signing up to be stored in the system, for the next time login.
- **3.0 Reservation:**
    - Inflows:
        - **“Reserve seat”** is the next step after the user logs in successfully, he will reserve a seat on the bus he wants.
        - **"User's info"** Information about the user making the reservation.
    - Outflows:
        - **“Request user’s info”** The need for extra details. The system needs to ask the user for more information if it is needed at the time of booking.
        - **"Store reservation details"** The final booking details. This is the process that saves the details of the reservation in the system's database.



- **4.0 Bus Route & Schedule:**
  - Inflows:
    - **"View route & schedule details"** This is the user's request to see the bus schedules and routes, after signing for his university.
    - **"Display route & schedule info"** This is the displaying of the route information on the user's device.
  - Outflows:
    - **"Bus route details"** This is the user's request to see the bus schedules and routes for the chosen university.
    - **"Request route & schedule info"** requests to bring the route and schedule info from the data store.
- **5.0 Bus Tracking:**
  - Inflows:
    - **"Request bus tracking"** user's request to see the current location of buses.
    - **"Update map"** The location of the bus will be updated on the map.
    - **"Provide location data**" actual location data coming from the buses.
  - Outflows:
    - **"Show location"** displaying the bus location on the user's map.
    - **"Fetch bus location"** retrieving the location of buses from the system.



- **6.0 Manage Route & Schedule:**
  - Inflows:
    - **"Update route & schedule"** information an administrator provides to change bus schedules and routes
    - **"Updated routes & schedules"** route and schedule information after admin updates.
  - Outflows:
    - **"Store updated routes & schedules"** the process of saving the changes from the admin



- **7.0 View route & schedule:**
  - Inflows:
    - **"Route & schedule"** driver's requests to see their routes and schedules.
    - **"Return route & schedule details"** system providing the driver with their route and schedule.

- Outflows:
  - **“Request bus route & schedule details”** process request to bring the routes and schedules from the data store.

- **8.0 Bus Status:**
  - Inflows:
    - **“Update bus status”** driver's update about the bus location.
    - **“Return current location & status**” system providing the location of a particular bus.
  - Outflows:
    - **“Store real-time location”** is the saving of the bus location into the system.

- **9.0 Medical & License Status:**
  - Inflows:
    - **“Medical & License checkup details”** information about a driver's medical checkup date and license date.
  - Outflows:
    - **“Medical & License status notification”** the drivers get notified about their medical and license status.
    - **“Store updated status”** system saving the status of the bus.

- **10.0 Feedback:**
  - Inflows:
    - **“Driver feedback”** the complaints, comments, and suggestions from drivers.
    - **“User feedback”** the complaints, comments, and suggestions from users.
  - Outflows:
    - **“Report generation”** system creating reports of users and driver feedback, sending them to the administrator.

5. Updates from Data Flows to Data Stores:

a. D1 University List:

- Interacts with the process **(1.0 Select University)** to fetch the list of available universities for the user to choose from. This is done in two steps, first, the process sends a "Request to display uni list", which is a request to access the data store D1, then the data store sends back a "Uni list" to process 1.0, which is the data retrieved from the data store.

b. D2 Users: interacts with the process **(2.0 Sign up/Login)** when:

- New users register in the system. The data flow that performs the update is "User's info".
- a user logs in or when their account data is modified. The data flow that performs the update is "User's info".
- Also, interacts with process (3.0 Reservation) to fetch user details needed for booking. The data flow used for the retrieval is "User's info."

c. D3 Reservations:

- Updated by Process (3.0 Reservation) when a user successfully makes a reservation. The data flow that performs the update is "Reservation details."

d. D4 Route & Schedule:

- Updated by process (4.0 Bus Route & Schedule) to retrieve the routes and schedule details to be shown to the user. It does this through the data flow "Bus route details."
- Also, updated by process (6.0 Manage Route & Schedule) when a University Admin updates the bus routes or schedules for their specific university. The data flow that performs the update is "Store updated routes & schedules."
- The data store interacts with process (7.0 View Route & Schedule) to read the route and schedule information from D4, to provide the details to the drivers.
- Also, interacts with the process (8.0 Bus Status) to read routes and schedule information to store updates on bus status.

e. D5 Bus Location:

- Updated by process **(8.0 Bus Status)** when a driver's bus has an updated location. The data flow that performs the update is "Store real-time location". Additionally, the process updates the bus status using the flow "Store updated status."
- Also, updated by process **(5.0 Bus Tracking)** to retrieve the real-time bus locations for display. It uses a data flow called "Fetch bus location" to get the data.

f. D6 Drivers:

- Updated by process **(9.0 Medical & License Status)** when a driver has a new medical checkup date or it's time to update the license info. The data flow that performs the update is "Medical & License status notification."

4.4.2 Data Flow (1-Level Diagram)

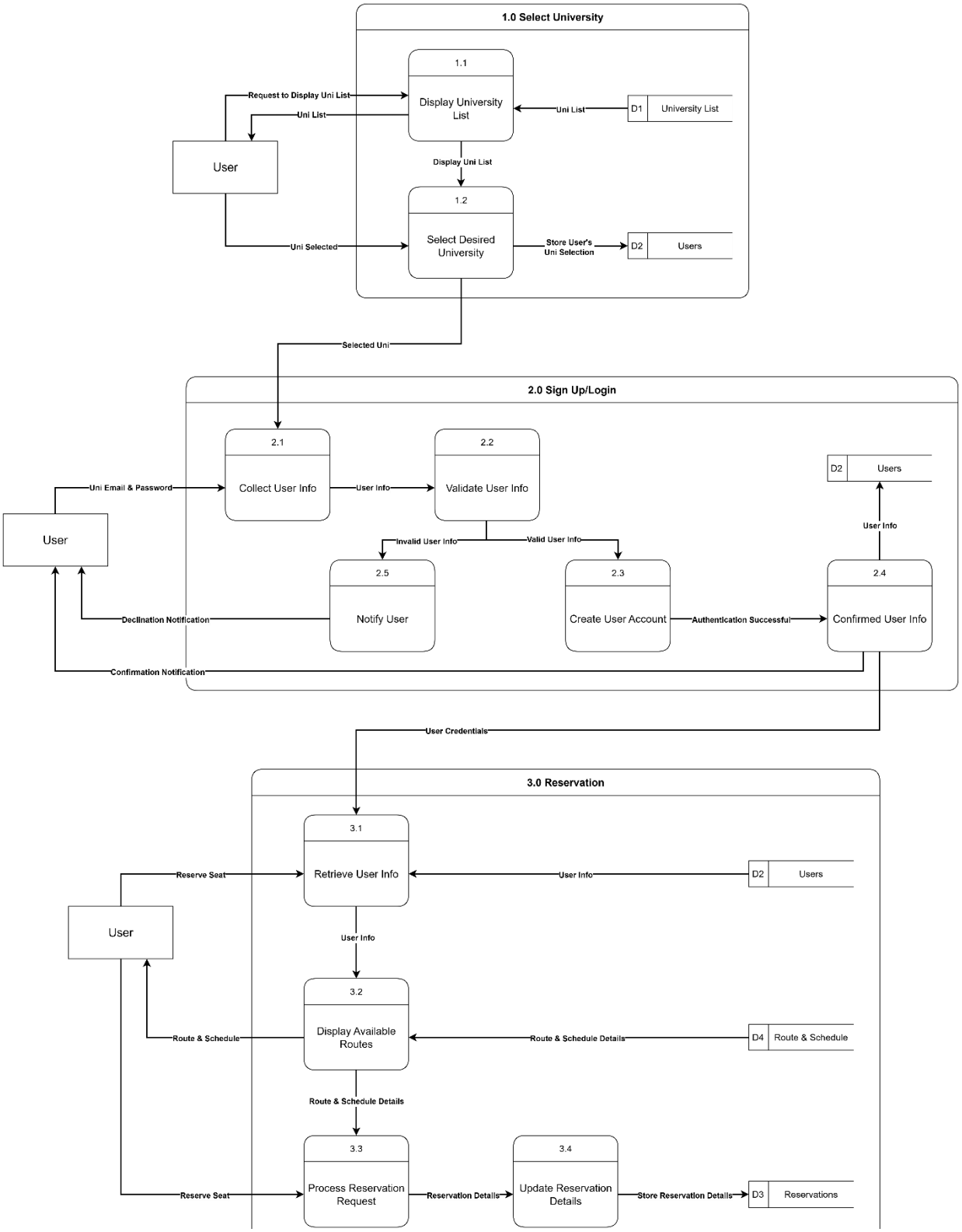
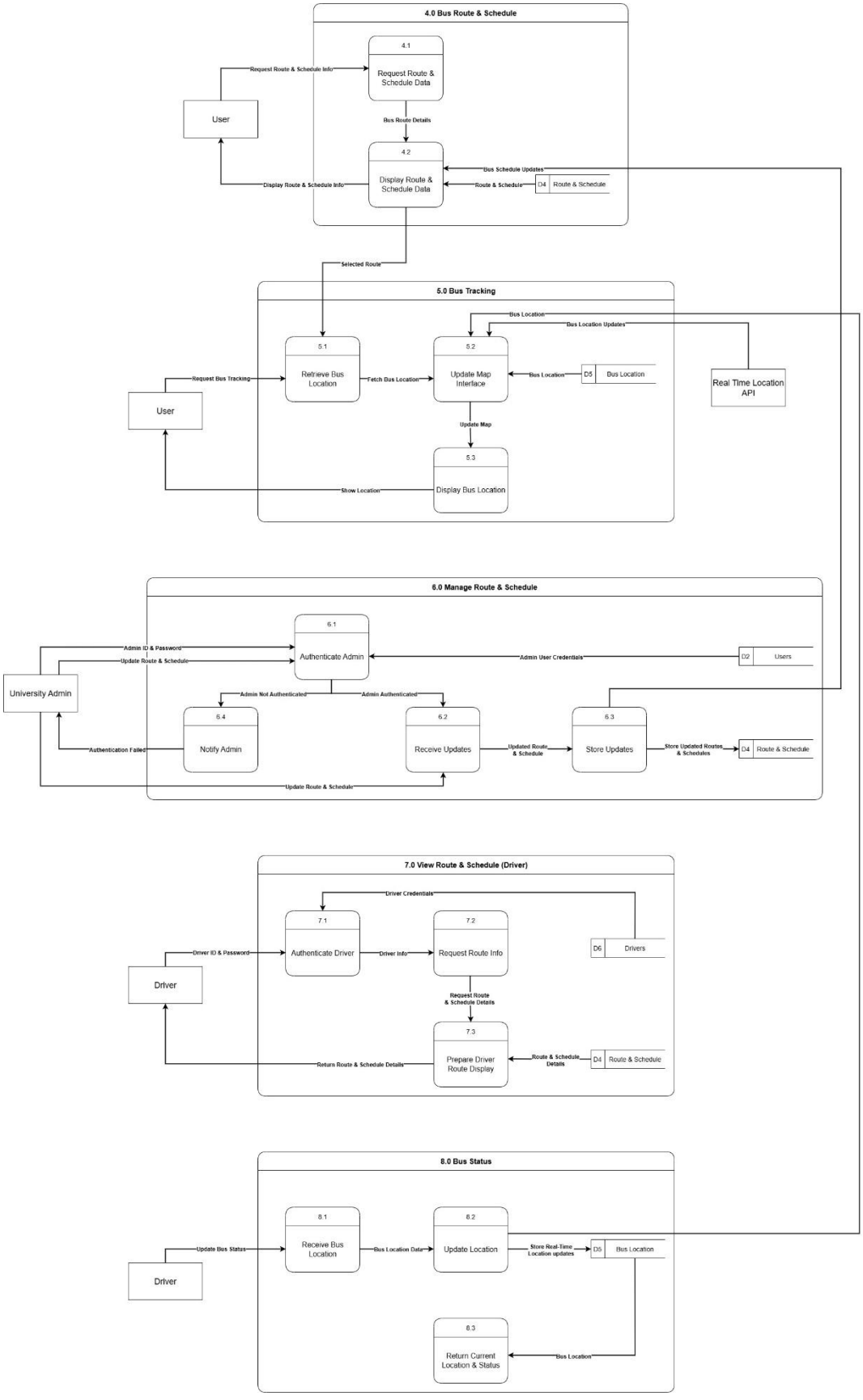
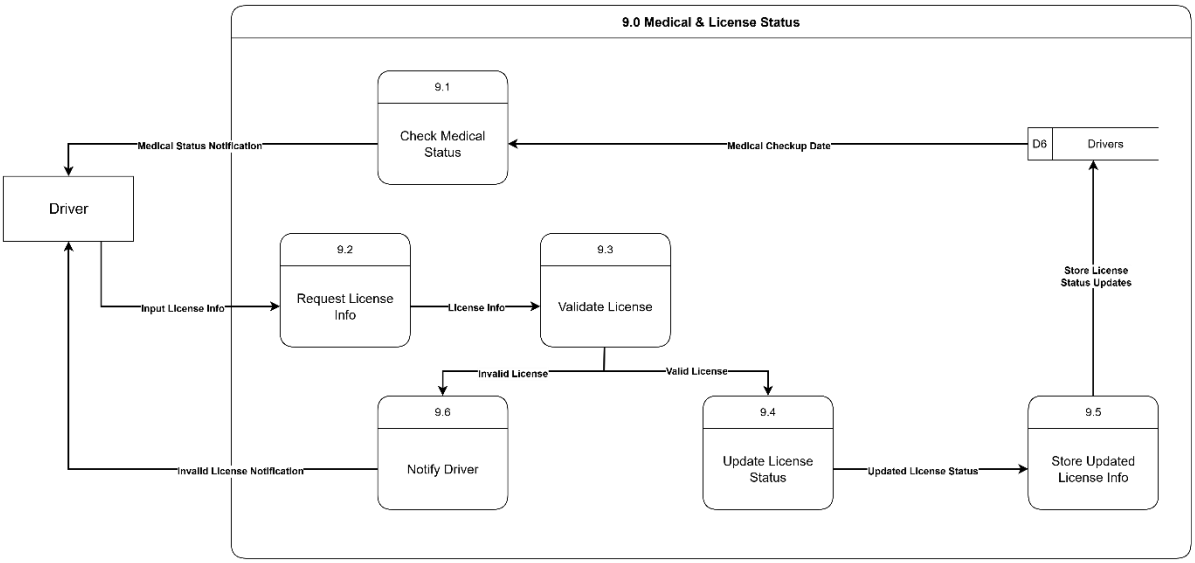
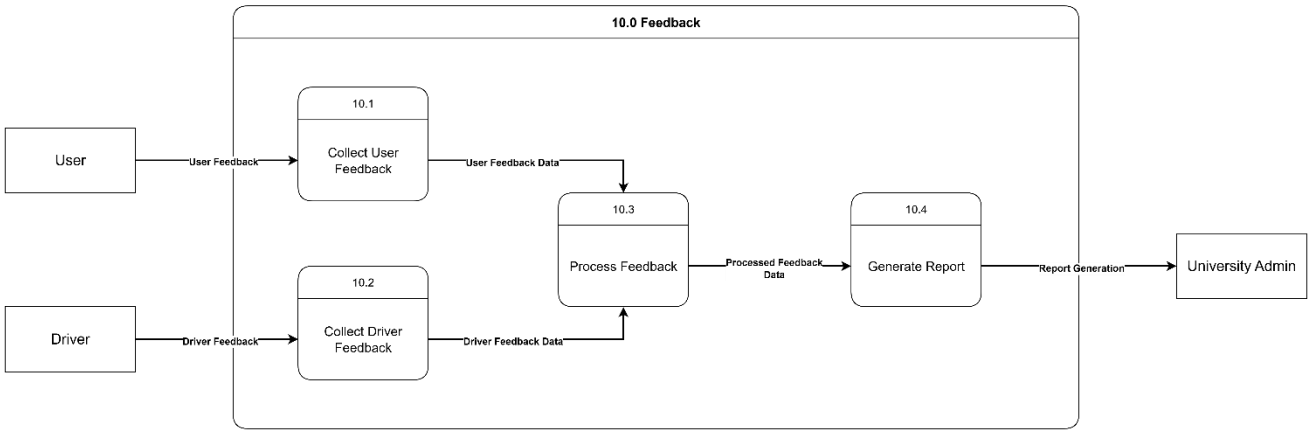


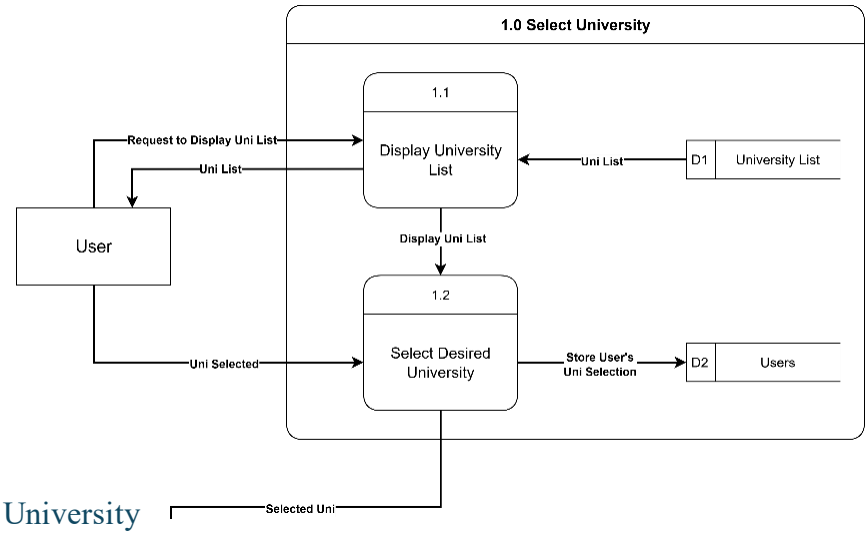
Figure 6: Data Flow (1-Level Diagram)


4.0 Bus Route & Schedule
4.1
Request Route & Schedule Data
Request Route & Schedule Info
User
Bus Route Details
4.2
Display Route & Schedule Data
Bus Schedule Updates
Display Route & Schedule Info
Route & Schedule
D4
Route & Schedule
Selected Route
5.0 Bus Tracking
Bus Location
Bus Location Updates
5.1
Retrieve Bus Location
5.2
Update Map Interface
Request Bus Tracking
Fetch Bus Location
Bus Location
D5
Bus Location
Real Time Location API
User
Update Map
5.3
Display Bus Location
Show Location
6.0 Manage Route & Schedule
6.1
Authenticate Admin
Admin ID & Password
Update Route & Schedule
Admin User Credentials
D2
Users
University Admin
Admin Not Authenticated
Admin Authenticated
6.4
Notify Admin
6.2
Receive Updates
6.3
Store Updates
Authentication Failed
Updated Route & Schedule
Store Updated Routes & Schedules
D4
Route & Schedule
Update Route & Schedule
7.0 View Route & Schedule (Driver)
Driver Credentials
7.1
Authenticate Driver
7.2
Request Route Info
Driver ID & Password
Driver Info
D6
Drivers
Driver
Request Route & Schedule Details
7.3
Prepare Driver Route Display
Route & Schedule Details
D4
Route & Schedule
Return Route & Schedule Details
8.0 Bus Status
8.1
Receive Bus Location
8.2
Update Location
Update Bus Status
Bus Location Data
Store Real-Time Location updates
D5
Bus Location
Driver
8.3
Return Current Location & Status
Bus Location


9.0 Medical & License Status
9.1
Check Medical Status
Medical Status Notification
Medical Checkup Date
D6
Drivers
Driver
9.2
Request License Info
9.3
Validate License
Input License Info
License Info
Store License Status Updates
Invalid License
Valid License
9.6
Notify Driver
9.4
Update License Status
9.5
Store Updated License Info
Invalid License Notification
Updated License Status


10.0 Feedback
10.1
Collect User Feedback
User
User Feedback
User Feedback Data
10.3
Process Feedback
10.4
Generate Report
Processed Feedback Data
Report Generation
University Admin
10.2
Collect Driver Feedback
Driver
Driver Feedback
Driver Feedback Data

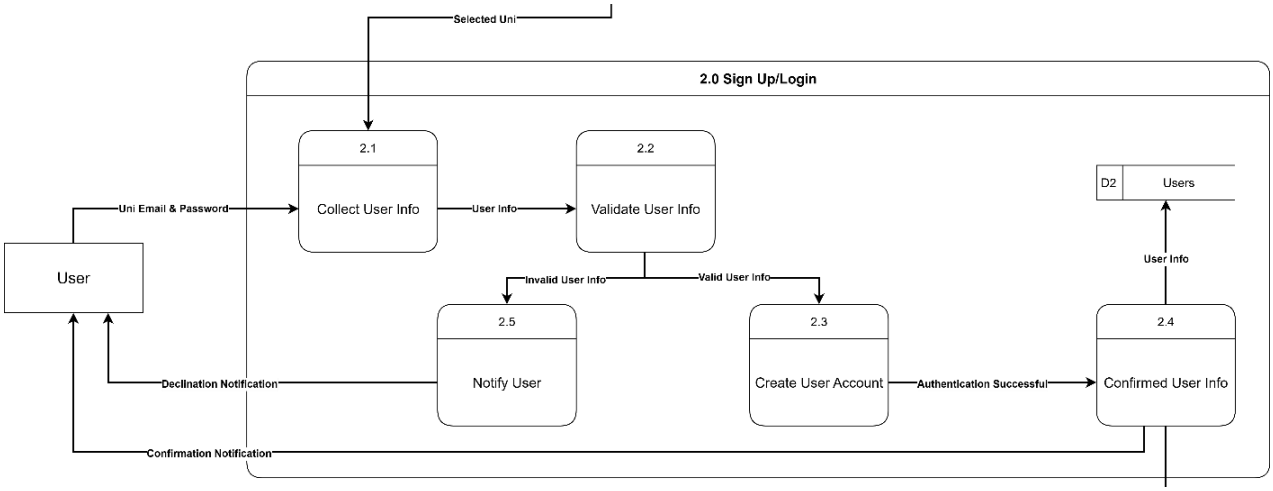
This Data Flow Diagram (DFD) Level 1 illustrates the major processes and data flows within the University Bus Management System. It provides a high-level overview of the system's functionality, showing how different actors (Users, Drivers, and University Admin) interact with the system and the key data stores involved. The diagram breaks the overall system into ten primary processes:



a. 1.0 Select University

- **1.1 Display University List:**
  - **Description:** Retrieves the list of universities from the University List datastore (D1)
  - **Inflows:** "Request to display uni list", A request from the User to view the available universities.
  - **Outflows:** "Uni List" & “Display Uni List” The list of universities is displayed to the user so that they can select their desired university.
  - **Datastore:** Reads from D1 (University List).
- **1.2 Select Desired University:**
  - **Description:** Takes the list and allows the user to select a single entry.
  - **Inflows:** "Display Uni List", The list of universities displayed by 1.1.
  - **Outflows:** "Uni Selected", The university the user has selected.

- **Data Store:** Stores the selected university information to D2 (Users)



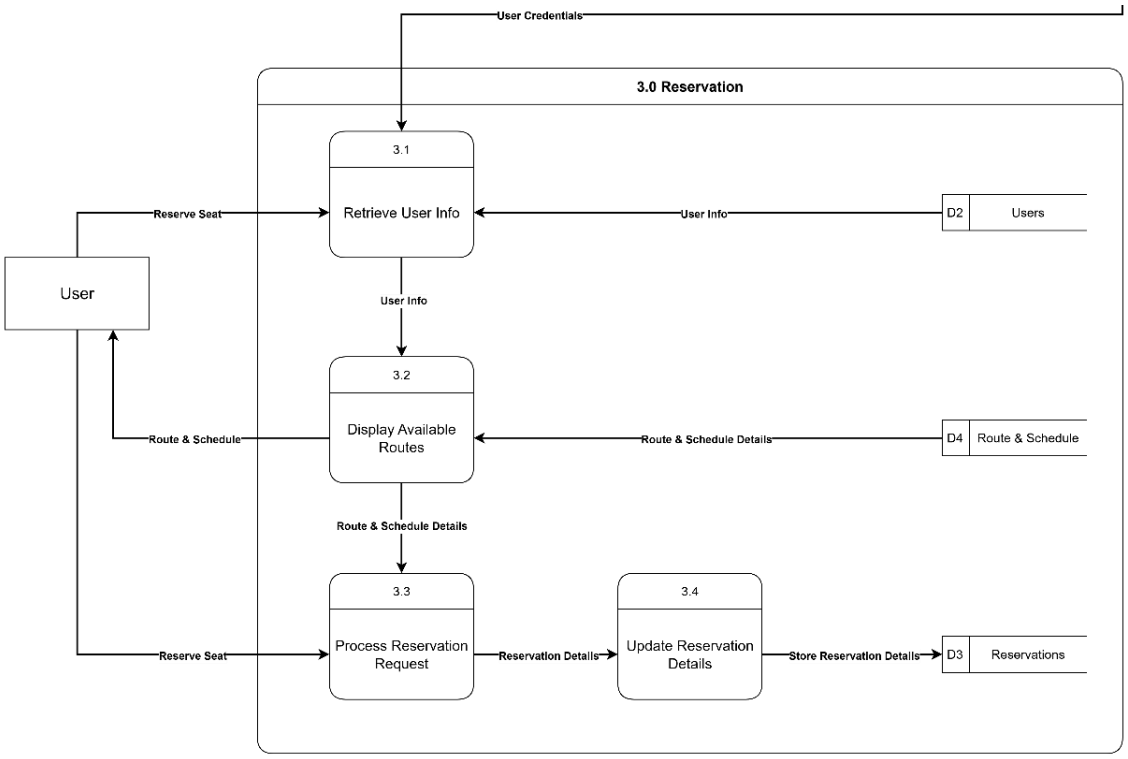
b. 2.0 Sign Up/Login

- **2.1 Collect User Info:**
  - **Description:** Receives the uni and user input and stores the information as User Info.
  - **Inflows:** "Selected Uni", The selected university from process 1.2, "Uni Email & Password", The user's entered email and password.
  - **Outflows:** "User Info", Collected user information (email, password, etc.)
  - **Datastore:** None
- **2.2 Validate User Info:**
  - **Description:** Checks the information against defined validation rules (e.g., email format, password complexity).
  - **Inflows:** "User Info", The collected user information from process 2.1
  - **Outflows:** "Valid User Info" or "Invalid User Info", Indicates whether the information passed validation.
  - **Datastore:** Potentially reads from D2 (Users) to check existing accounts during sign-up.
- **2.3 Create User Account:**

- **Description:** Creates a new record in the User datastore (D2) with the validated information.
- **Inflows:** "Valid User Info", The validated user information from process 2.2
- **Outflows:** "Authentication Successful", Confirms that the account was created successfully.
- **Datastore:** None

- **2.4 Confirmed User Info:**
  - **Description:** Creates a record of the user's details.
  - **Inflows:** "Authentication Successful", the signal that the user has been created successfully.
  - **Outflows:** "User Info", to store the info that the user provided in the datastore users (D2), "Confirmation Notification", which sends the user a notification of a successfully created account, "User Credentials", Confirms the user info to continue to the next step.
  - **Datastore:** stores user info in D2 Users.
- **2.5 Notify User:**
  - **Description:** Formats and sends a notification to the user indicating the rejection.
  - **Inflows:** "Invalid User Info", Signal from 2.2 that the user info is invalid.
  - **Outflows:** "Declination Notification" is a message informing the user of the invalid input.

- **Datastore:** None

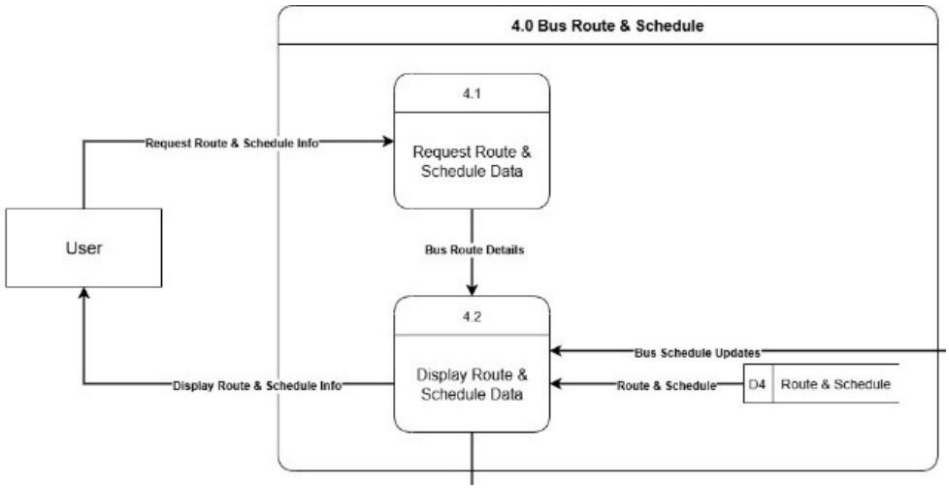


c. 3.0 Reservation

- **3.1 Retrieve User Info:**
  - **Description:** queries D2 (Users) using the user's credentials.
  - **Inflows:** "Reserve Seat", Request from the User to Reserve a seat.
  - **Outflows:** User Info", The retrieved user information.
  - **Datastore:** Reads from D2 (Users).
- **3.2 Display Available Routes:**
  - **Description:** Retrieves the list of routes and their corresponding schedules from the Route & Schedule datastore (D4).
  - **Inflows:** "User Info", User Information to find the routes for the same university.
  - **Outflows:** "Route & Schedule Details", The displayed route and schedule information to the user to continue to the next step to complete the reservation process.
  - **Datastore:** Reads from D4 (Route & Schedule).
- **3.3 Process Reservation Request:**

- **Description:** Validates the request (e.g., checks for seat availability) and prepares the reservation details.
- **Inflows:** "Route & Schedule", The chosen Route & Schedule from 3.2, and "Reserve Seat", the reservation request the user made.
- **Outflows:** "Reservation Details", The reservation details.
- **Datastore:** None

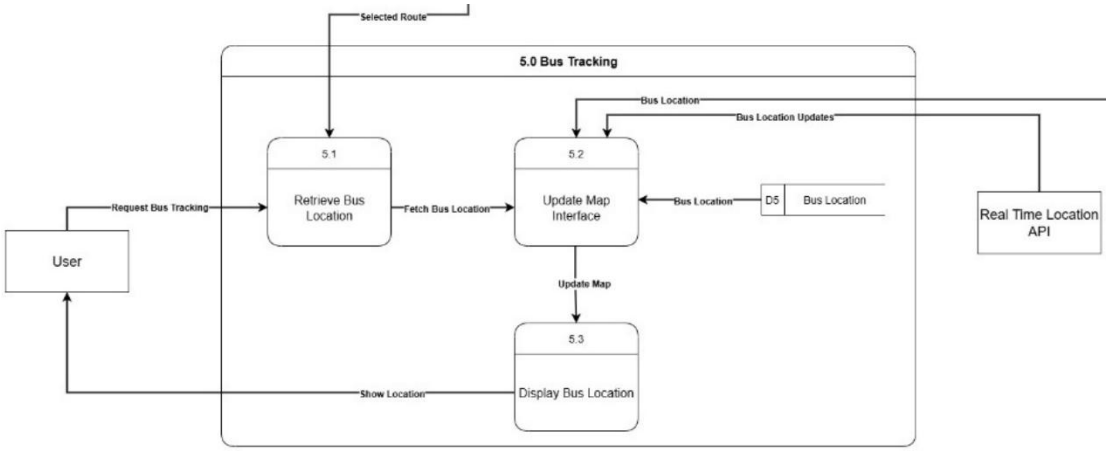
- **3.4 Update Reservation Details:**
  - **Description:** Stores the reservation record in D3.
  - **Inflows:** "Reservation Details", The details of the reservation from 3.3
  - **Outflows:** "Store Reservation Details", Store the data in the data store.
  - **Datastore:** Writes to D3 (Reservations).



d. 4.0 Bus Route & Schedule

- **4.1 Request Route & Schedule Data:**
  - **Description:** It processes the request and sends data from the Route & Schedule Datastore.
  - **Inflows:** "Request Route & Schedule Info", The User requests the information from the data store.
  - **Outflows:** "Bus Route Details", Returns details for the Bus Route.
  - **Datastore:** None

- **4.2 Display Route & Schedule Data:**
    - **Description:** It processes the output and displays it for the User.
    - **Inflows:** "Bus Route Details", The Bus Route Details to display from the data store, and “Bus Schedule Updates”, when the admin make updates to the schedule, it displays to the user.
    - **Outflows:** "Display Route & Schedule Info", Returns details for the Bus Route that will be displayed to the User and “Selected Route”, after the user selects a route, the system will be able to display the bus location.
    - **Datastore:** Reads from D4 (Route & Schedule).

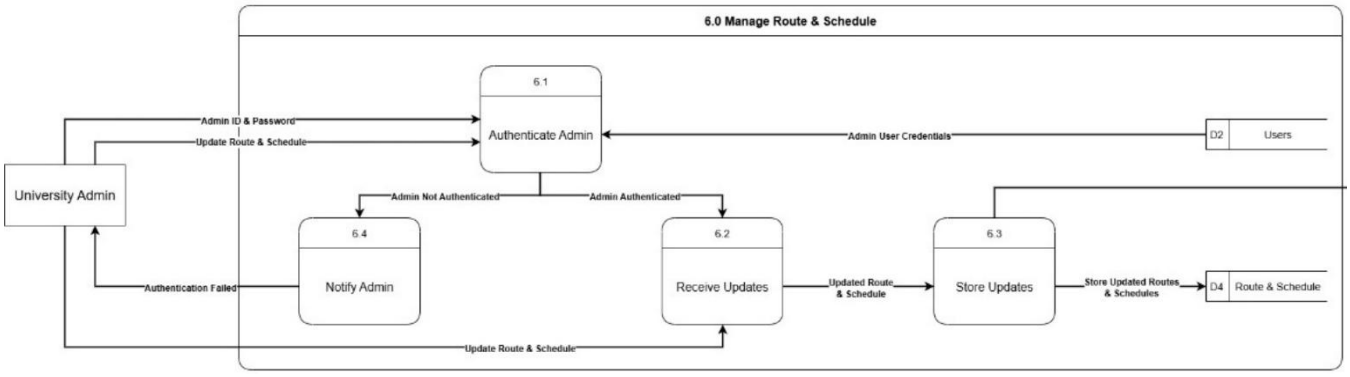


e. 5.0 Bus Tracking

- **5.1 Retrieve Bus Location:**
    - **Description:** Requests the bus location from the Real Time API.
    - **Inflows:** "Request Bus Tracking", Signal that the User wants to track a bus, and “Selected Route”, after the user selects a route, then the app displays the bus location.
    - **Outflows:** "Fetch Bus Location", Fetch the data from the API location.
    - **Datastore:** Interacts with the Real Time Location API.
- **5.2 Update Map Interface:**
    - **Description:** Processes of the data for the interface

- **Inflows:** "Fetch Bus Location", Retrieved bus location and "Bus Location Updates", New information from Real Time API, and “Bus Location”, from the updated bus location that is tracked by the driver.
- **Outflows:** "Update Map", Sends location information to the interface.
- **Datastore:** Writes to D5 (Bus Location).

- **5.3 Display Bus Location:**
  - **Description:** Takes in the map information and presents it to the user in real time.
  - **Inflows:** "Update Map”, Information for the map.
  - **Outflows:** "Show Location", Bus Location Information to be shown to the user.
  - **Datastore:** None

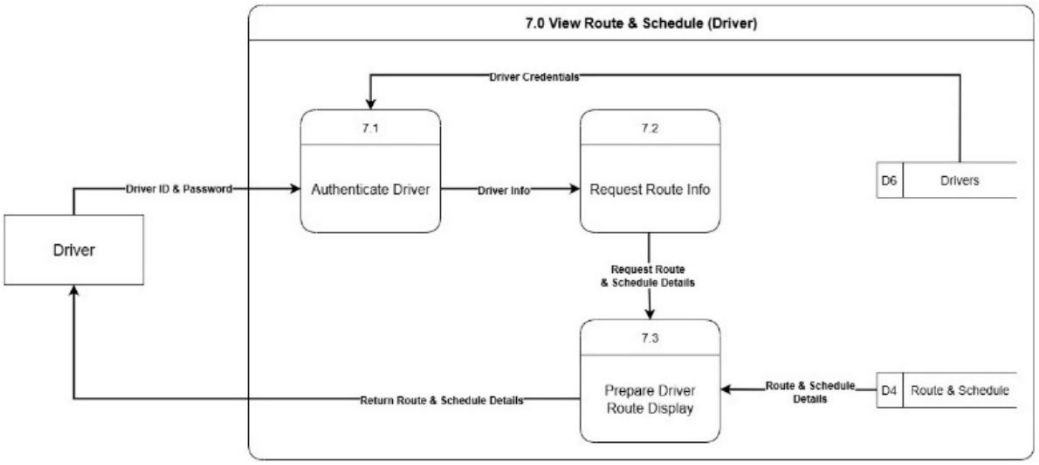


f. 6.0 Manage Route & Schedule

- **6.1 Authenticate Admin:**
  - **Description:** Retrieves the admin account and validates credentials.
  - **Inflows:** "Admin ID & Password", Log-in information to be validated, and “Update Route & Schedule”, the admin request to update the schedule.
  - **Outflows:** "Admin Authenticated" or "Admin Not Authenticated", Authenticate admin, otherwise reject.
  - **Datastore:** Reads from D2 (Users).
- **6.2 Receive Updates:**

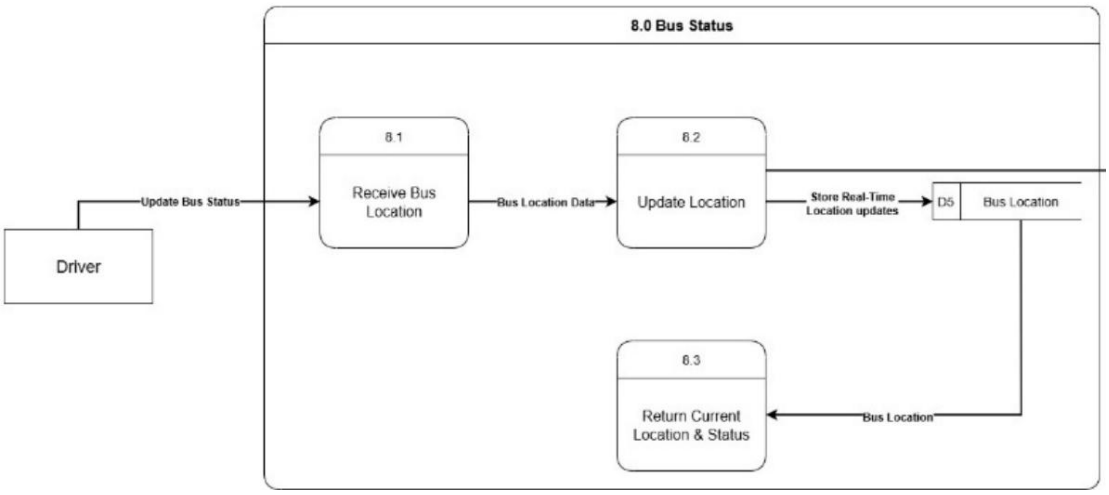
- **Description:** Receives the new changes to be implemented.
- **Inflows:** "Admin Authenticated", Authentication Data for the Admin User, "Update Route & Schedule", Data to update.
- **Outflows:** "Updated Route & Schedule", Data to be updated to the route.
- **Datastore:** None

- **6.3 Store Updates:**
  - **Description:** Updates the data to the datastore D4.
  - **Inflows:** "Updated Route & Schedule", The new information.
  - **Outflows:** "Store Updates", Updated Route and Schedule to the Datastore, and "Bus Schedule Updates", update the schedule to the user.
  - **Datastore:** Writes to D4 (Route & Schedule).
- **6.4 Notify Admin:**
  - **Description:** Message to show the admin to update new data.
  - **Inflows:** "Admin Not Authenticated", Signal that Authentication has failed.
  - **Outflows:** "Authentication Failed", To notify admin for a failed login.
  - **Datastore:** None



g. 7.0 View Route & Schedule (Driver)

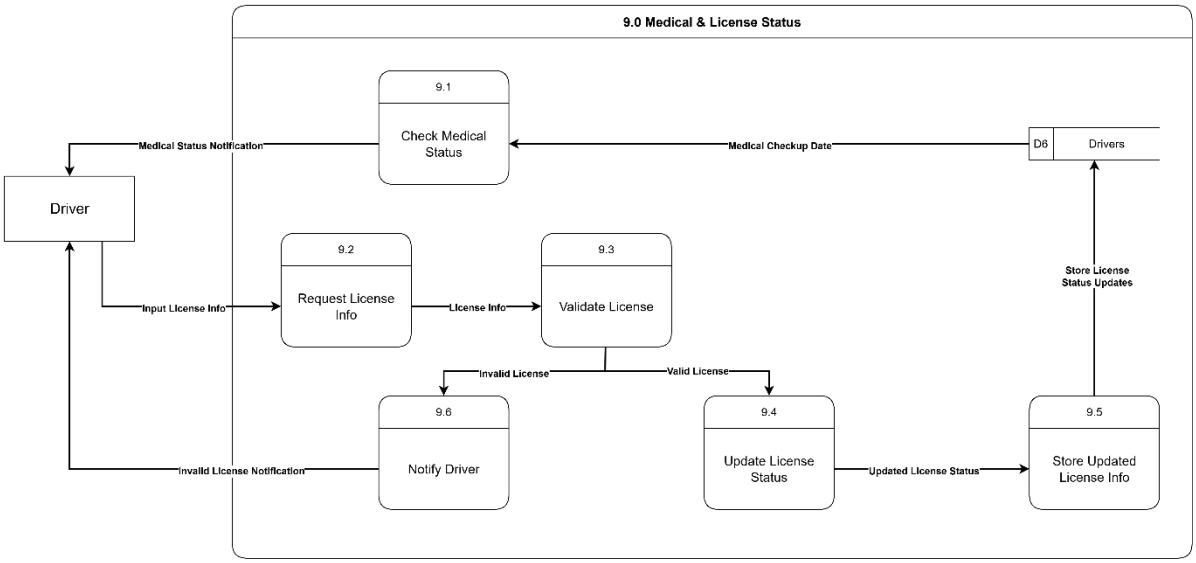
- **7.1 Authenticate Driver:**
  - **Description:** Retrieve their ID and password from D6 (Drivers).
  - **Inflows:** "Driver ID & Password", Log in for the driver.
  - **Outflows:** "Driver Info", Authenticate the driver with their information from the database.
  - **Datastore:** Reads from D6 (Drivers).
- **7.2 Request Route Info:**
  - **Description:** Check their schedule
  - **Inflows:** "Driver Info", Authenticated data for the driver
  - **Outflows:** "Request Bus Route & Schedule Details", The details for the route.
  - **Datastore:** None
- **7.3 Prepare Driver Route Display:**
  - **Description:** Read the Schedule data from D4.
  - **Inflows:** "Request Bus Route & Schedule Details", The schedule to be displayed.
  - **Outflows:** "Return Route & Schedule Details", The formatted schedule for the driver.
  - **Datastore:** Reads from D4 (Route & Schedule).



h. 8.0 Bus Status

- **8.1 Receive Bus Location:**
  - **Description:** Get the data of where the bus is currently at.
  - **Inflows:** "Update Bus Status", Information of where the bus is currently at.
  - **Outflows:** "Bus Location Data", Bus Location Information.
  - **Datastore:** None
- **8.2 Update Location:**
  - **Description:** Data for the Bus is updated in real-time.
  - **Inflows:** "Bus Location Data", The data for the bus, "Driver Update".
  - **Outflows:** "Location Updates", Update the real-time location.
  - **Datastore:** Writes to D5 (Bus Location).
- **8.3 Return Current Location & Status:**
  - **Description:** Read data and give current location.
  - **Inflows:** "Location Updates", The Data, and location information.
  - **Outflows:** "Bus Location", The Current Bus Location.

- **Datastore:** Writes to D5 (Bus Location).

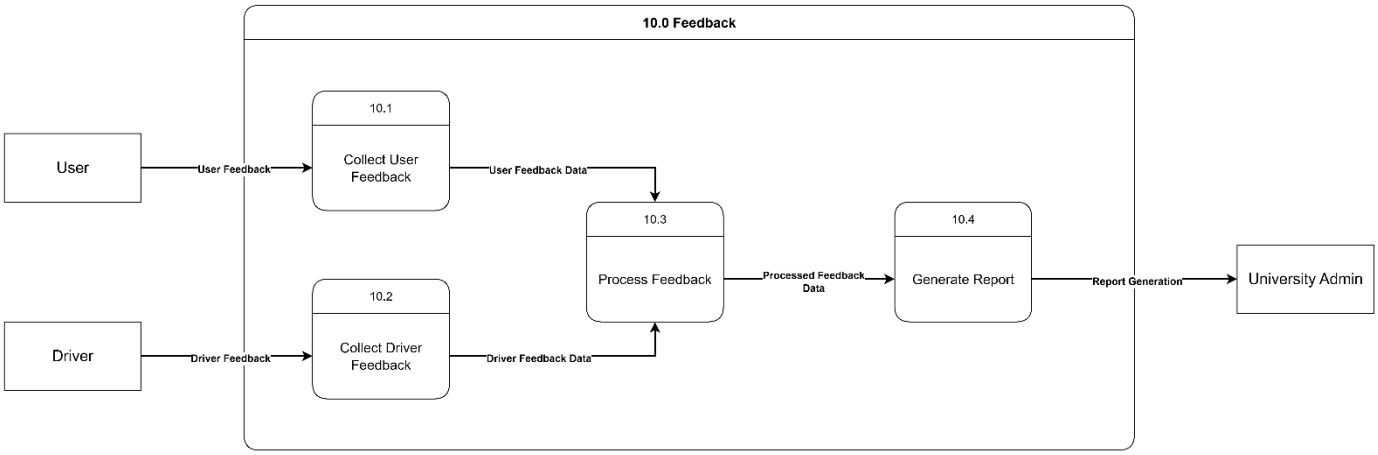


i. 9.0 Medical & License Status

- **9.1 Check Medical Status:**
  - **Description:** Request the medical checkup date.
  - **Inflows:** “Medical Checkup Date”, the date to see if the driver is safe to drive.
  - **Outflows:** "Medical Status Notification", sends a notification to the driver to notify him about their medical status and checkup date.
  - **Datastore:** Reads from D6 (Drivers).
- **9.2 Request License Info:**
  - **Description:** Request the license from the driver.
  - **Inflows:** “Input License Info”, the driver will provide their driving license info.
  - **Outflows:** “License Info”, driver input license info to check for the validity of the license.
  - **Datastore:** None
- **9.3 Validate License:**
  - **Description:** check the License and send it to the database.

- **Inflows:** “License Info”, collected license info from the driver.
- **Outflows:** "Valid License" or "Invalid License".
- **Datastore:** None

- **9.4 Update License Status:**
  - **Description:** Updates the status of the database.
  - **Inflows:** "Valid License", The license is valid.
  - **Outflows:** "Updated License Status", Updated data.
  - **Datastore:** Writes to D6 (Drivers).
- **9.5 Store Updated License Status:**
  - **Description:** Store and update the data.
  - **Inflows:** "Updated License Status", The updated status.
  - **Outflows:** "Store License Status Updates", Update, and Store Data.
  - **Datastore:** Writes to D6 (Drivers).
- **9.6 Notify Driver:**
  - **Description:** Show the Driver a prompt to fix their invalid License (expiration date, etc.)
  - **Inflows:** "Invalid License", The license is invalid.
  - **Outflows:** "Invalid License Notification", Notify the driver of this notification.
  - **Datastore:** None



j. 10.0 Feedback

- **10.1 Collect User Feedback:**
  - **Description:** Get feedback and update the system from the user.
  - **Inflows:** "User Feedback", Feedback for the User.
  - **Outflows:** "User Feedback Data", all the feedback data.
  - **Datastore:** None
- **10.2 Collect Driver Feedback:**
  - **Description:** Get feedback and update the system from the driver.
  - **Inflows:** "Driver Feedback", Feedback for the driver.
  - **Outflows:** "Driver Feedback Data", all the feedback data.
  - **Datastore:** None
- **10.3 Process Feedback:**
  - **Description:** Process data to the system.
  - **Inflows:** "User Feedback Data" and "Driver Feedback Data", All the Feedback.
  - **Outflows:** "Processed Feedback Data", Process Data.
  - **Datastore:** None
- **10.4 Generate Report:**
  - **Description:** Generate a report to the University Admin.

- **Inflows:** "Processed Feedback Data", Information for the University Admin.
- **Outflows:** "Report Generation", Send information to the admin to implement.
- **Datastore:** Creates a report for review and analysis.

4.5 Use Case Diagram

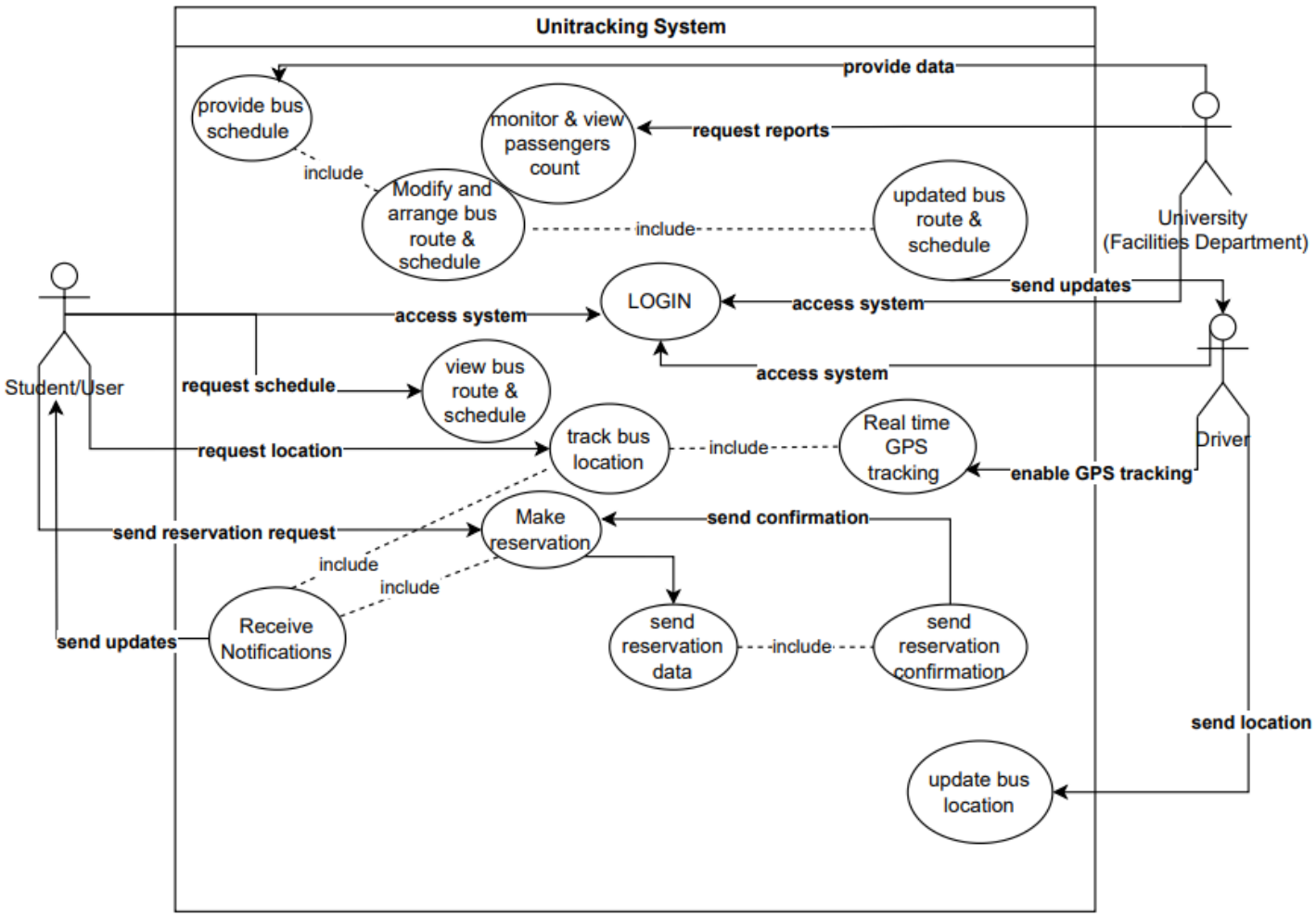


Figure 7: Use Case Diagram

4.5.1 Actors (Who Uses the System)

1. **Student/User**

They are the main users (students, faculty, or guests) who want to:

- Access the system (via a mobile app or web app).
- Login using their university credentials.
- View bus route & schedule to plan their trip.
- Track bus location in real time using GPS (to see where the bus is).
- Make a reservation to book a seat on a bus.
- Receive confirmation after booking.
- Receive notifications for:
    - Bus delays
    - Seat availability
    - Reservation updates

2. **Driver**

The driver uses the system to:

- Access the system using their login.
- Send updates about:
    - Bus location (real-time GPS tracking).
    - Medical and license status if required.
- Request their schedule – to know their assigned routes.
- Send confirmation after completing a task (like finishing a route).
- Enable GPS tracking so users can see the bus on the map.

3. **University (Facilities Department)**

They manage the bus service. They:

- Access the system as admins.
- Provide and update bus schedules.
- Modify routes and arrange new ones.
- Track bus location for all buses.
- Monitor passenger count – how many people are on the bus.
- Request reports to make data-driven decisions.
- Send data related to routes, schedules, and usage.

4.5.2 What does the System Does?

1. **Login**

   All users (students, drivers, admins) must log in to access features.

   - **Actors involved:** Student, Driver, University Admin
   - **Interactions:**
     - Must verify identity using **username/email + password**.
     - Once logged in:
       - Students can access reservation, tracking, and schedules.
       - Drivers can update location and check schedules.
       - Admins can modify routes, check passenger count, etc.

2. **View Bus Route & Schedule**

   - Students can see:
     - Bus stops
     - Timings
     - Which bus to take
   - **Actor:** Student, includes:
     - **Request Schedule** (user asks system to fetch schedule).

- **Display Schedule** (system shows result).

- **Interactions:**
  - Student selects a university and chooses a route.
  - The system pulls route and timing from database.
  - Displays pickup points and arrival times.

3. **Real-Time GPS Tracking**

   Shows the **live location of the bus** on a map using GPS.

   - Drivers share their location.
   - Students see where the bus is.
   - **Actors:** Student, Driver. Includes:
     - **Update Bus Location** (from driver via GPS)
     - **Request Location** (by student or admin)
     - **Display Live Location**
   - **Interactions:**
     - Driver's device constantly sends **live GPS coordinates**.
     - The system processes and updates the **bus marker on the map**.
     - Students/admins request location, and the system shows it.

4. **Make a Reservation**

   - Students can **book a seat**.
   - System checks availability.
   - Sends a confirmation if successful.
   - **Actor:** Student, includes:
     - **Send Reservation Request**
     - **Send Confirmation**
   - **Interactions:**
     - Student selects a bus, date, and time.
     - The system checks:
       - If a seat is available.

        - If the user is authorized.
    - If successful, it sends back a **reservation confirmation**.
    - Admins receive updated **passenger count**.

5. **Send Reservation Confirmation**

   After booking, the system sends a message: "Your seat is reserved."

6. **Monitor & View Passenger Count**

   - Admin can see how many people:
     - Reserved seats
     - Are on the bus
   - **Actor:** University Admin, includes:
     - **Send Reservation Data**
     - **Generate Reports (optional use case)**
   - **Interactions:**
     - Reservation data is stored in the database.
     - Admin can view:
       - How many users booked each bus.
       - Who reserved which route.
     - Useful for planning bus size and routes.

7. **Modify Bus Route & Schedule**

   - University staff can:
     - Change routes
     - Add/remove buses
     - Update timing
   - **Actor:** University Admin, includes:
     - Update Route & Schedule
     - Send Updates to System
     - Notify Drivers and Users
   - **Interactions:**
     - Admin changes time/route info in the system.

- System saves it in the database.
- Drivers are notified of new assignments.
- Students see the updated schedule next time they log in.

8. Update Bus Location

Drivers' GPS updates the location in real-time.

9. Send Notifications

- System sends alerts or messages to:
  - Inform users about bus status
  - Notify admins of issues
- **Actors:** Student, Driver. Includes:
  - Notification Trigger
  - Send Notification
- **Interactions:**
  - System sends push/email alerts for:
    - Bus delay
    - Seat reservation status
    - Route changes
    - Admin announcements

10. Track Bus Location

- **Actor:** University Admin. Includes:
  - Request Bus Location
  - Enable GPS Tracking (by driver)
- **Interactions:**
  - Admin can view where buses are in real-time.
  - Uses same GPS feed as student view.
  - Useful for monitoring performance, delays, or emergencies.

4.5.3 Interactions

Table 6: Use Case Interactions

Main Use Case	Includes (Sub-Use Case)	Why It's Included
Make Reservation	Send Reservation Request → Send Confirmation	Needed to complete reservation
View Bus Route & Schedule	Request Schedule → Show Schedule	Data must be fetched before display
Real-Time GPS Tracking	Update Location (Driver) → Request Location (User)	GPS must be live
Modify Route & Schedule	Update Info → Notify System	Reflects changes to users
Monitor Passengers	Send Reservation Data → Generate Reports	Tracks usage
Receive Notifications	Notification Trigger → Send Notification	Keeps users informed

4.6 Entity Relationship Diagram (ERD)

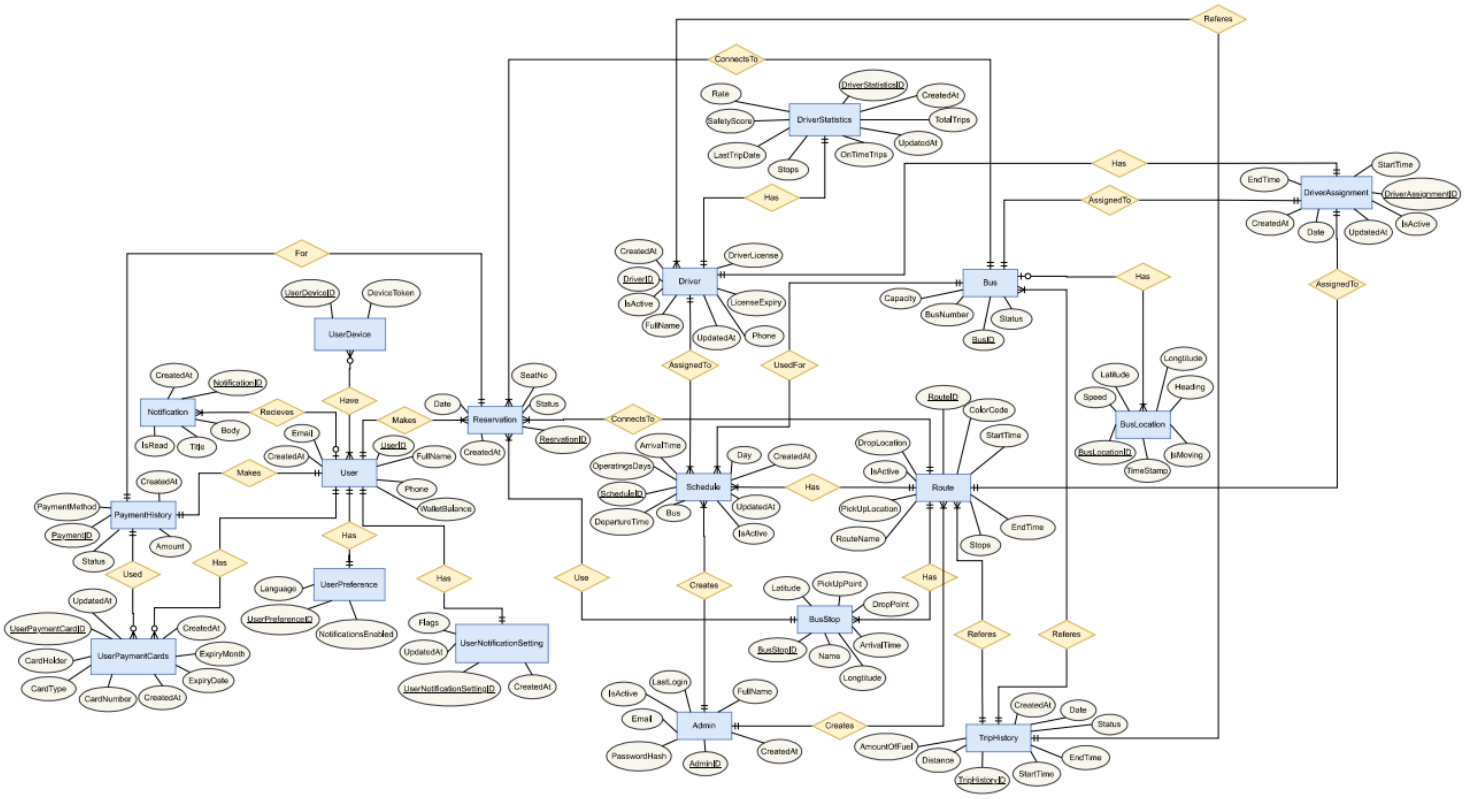
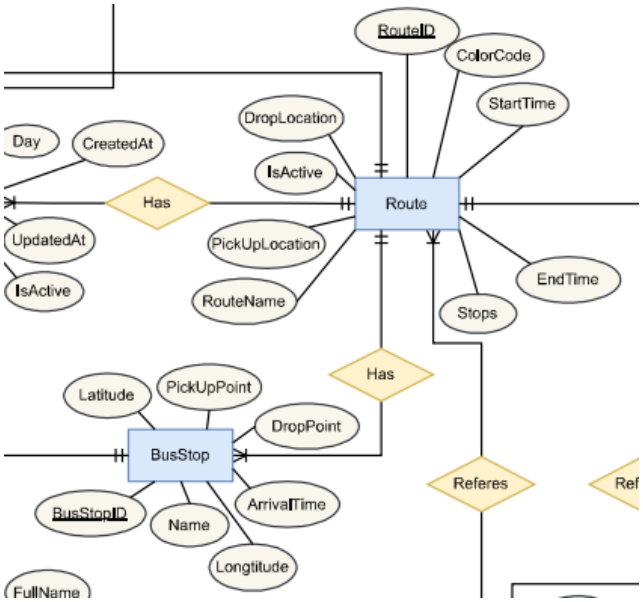


Figure 8: Entity Relationship Diagram (ERD)

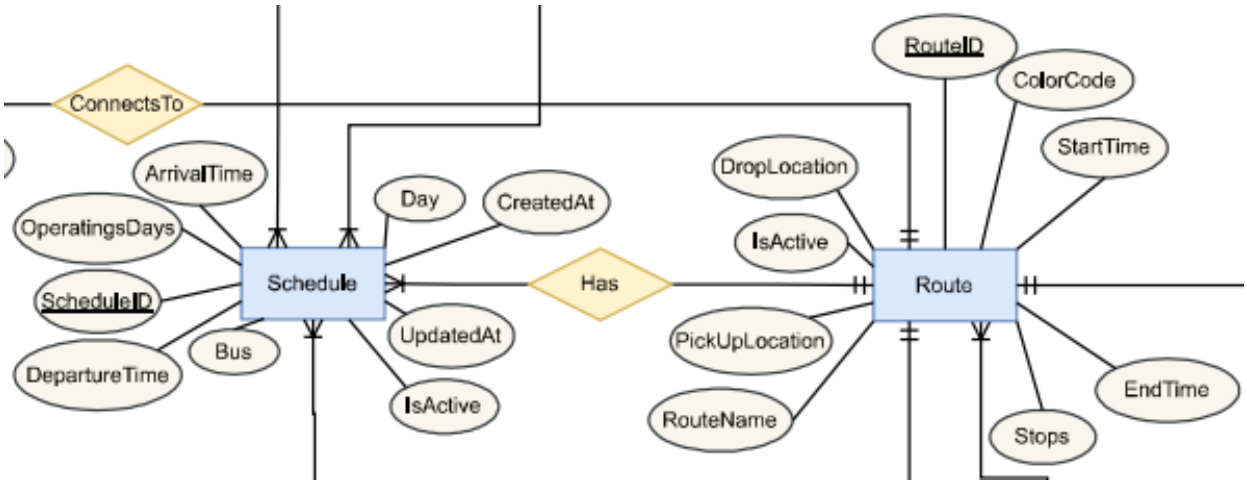
(The ERD Diagram is already normalized to the third form)

4.6.1 Entities & Relationships

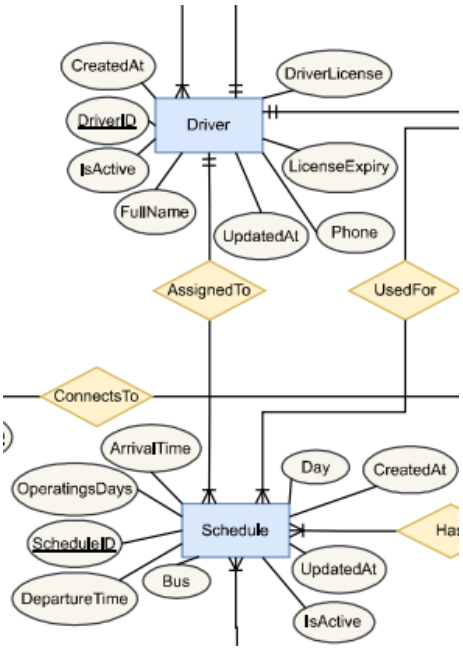
1. **Routes and Bus Stops:**
    - **Relationship:** One Route has Many Bus Stops.
    - **Cardinality:** One-to-Many
    - **Details:** A single route is composed of multiple, sequential bus stops. Each bus_stops record belongs to exactly one route.
    - **Foreign Key:** route_id



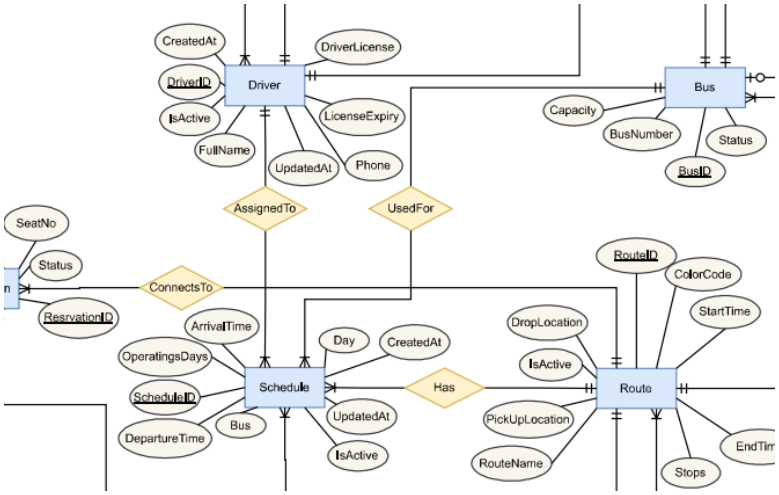
2. **Routes and Schedules:**
    - **Relationship:** One Route has Many Schedules.
    - **Cardinality:** One-to-Many
    - **Details:** A route can have many scheduled trips throughout the week. Each schedule entry is for a single route.
    - **Foreign Key:** route_id



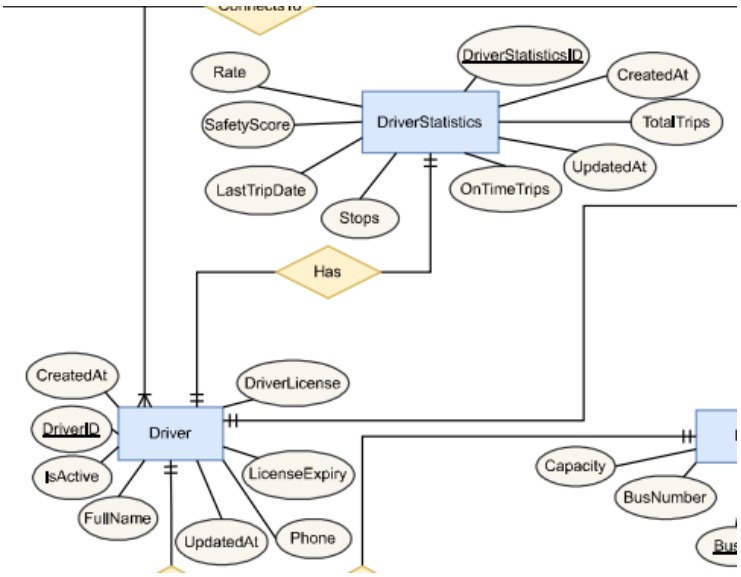
3. **Drivers and Schedules:**
    - **Relationship:** One Driver can be assigned to Many Schedules.
    - **Cardinality:** One-to-Many
    - **Details:** A single driver can be responsible for multiple scheduled trips. Each specific schedule is assigned to one driver.
    - **Foreign Key:** driver_id



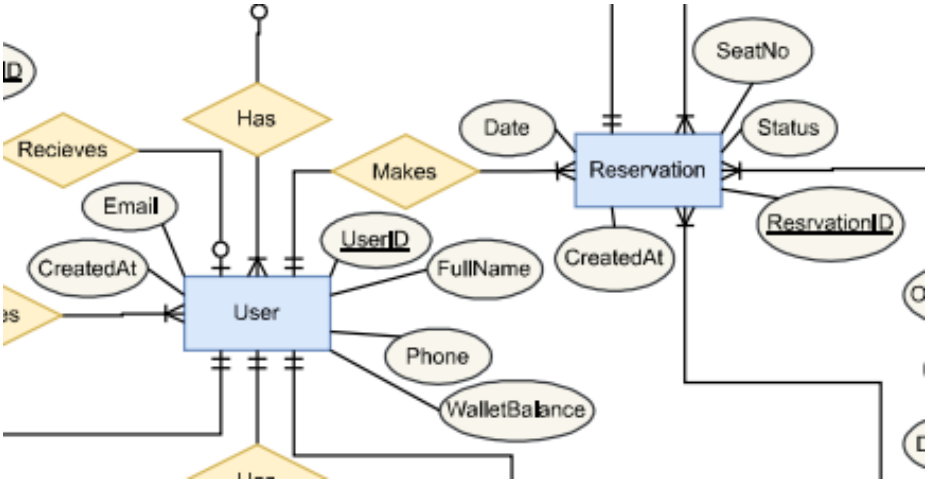
4. **Buses and Schedules:**
    - **Relationship:** One Bus can be used for Many Schedules.
    - **Cardinality:** One-to-Many
    - **Details:** A physical bus vehicle can be used to run multiple trips as defined in the schedules. Each schedule entry specifies exactly one bus.
    - **Foreign Key:** bus_id



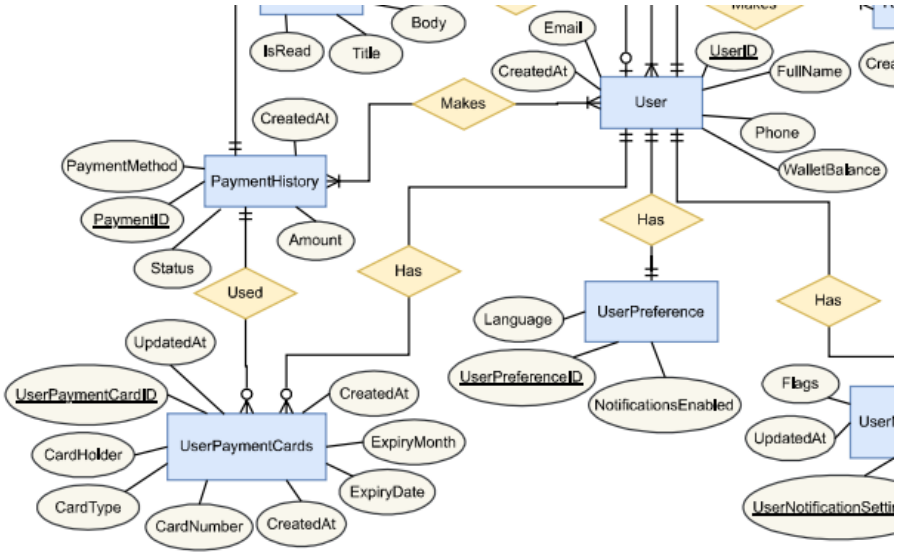
5. **Drivers and driver_statistics:**
    - **Relationship:** One Driver has One Statistics Record.
    - **Cardinality:** One-to-One
    - **Details:** Each driver has a single, corresponding record in driver_statistics that aggregates their performance data.
    - **Foreign Key:** driver_id



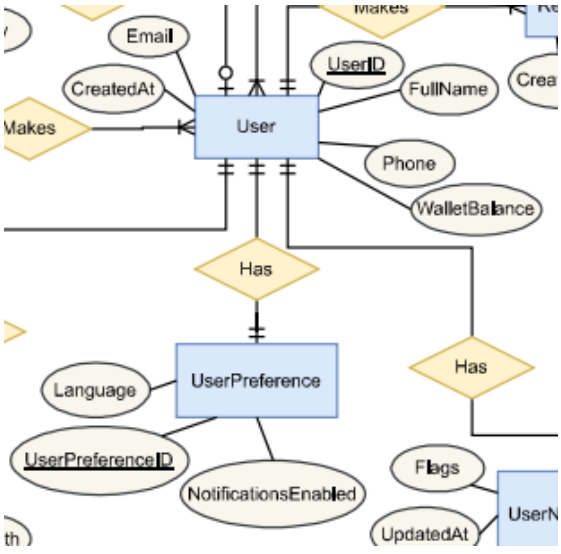
6. **Users and Reservations:**
    - **Relationship:** One User makes Many Reservations.
    - **Cardinality:** One-to-Many
    - **Details:** A user can book multiple bus trips. Each reservation is made by a single user.
    - **Foreign Key:** user_id



7. **Users and user_payment_cards:**
    - **Relationship:** One User has Many Payment Cards.
    - **Cardinality:** One-to-Many
    - **Details:** A user can store multiple credit/debit cards in their account for payment.
    - **Foreign Key:** user_id

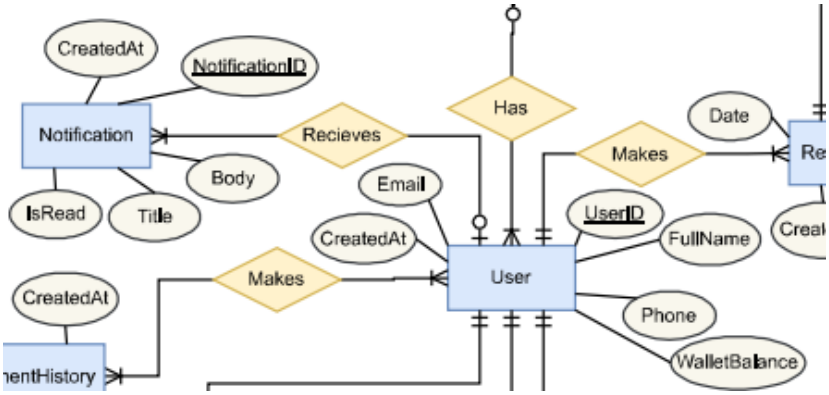


8. **Users and user_preferences:**
    - **Relationship:** One User has One set of Preferences.
    - **Cardinality:** One-to-One
    - **Details:** Each user has a unique set of application preferences
    - **Foreign Key:** user_id



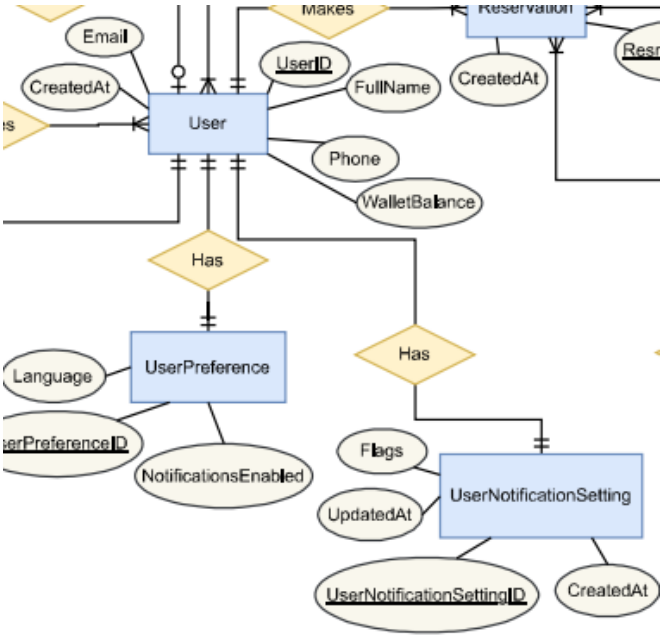
11. Users and Notifications:

- **Relationship:** One User receives Many Notifications.
- **Cardinality:** One-to-Many
- **Details:** The system can send multiple notifications (e.g., trip reminders, delay alerts) to a user.
- **Foreign Key:** user_id



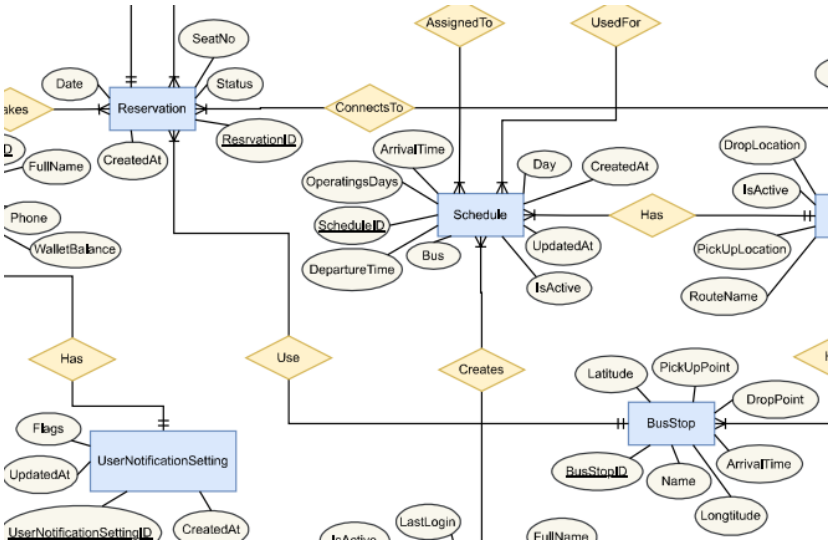
12. Users and user_notification_settings:

- **Relationship:** One User has One set of Notification Settings.
- **Cardinality:** One-to-One
- **Details:** Similar to preferences, each user has a unique set of toggles for which notifications they want to receive.
- **Foreign Key:** user_id



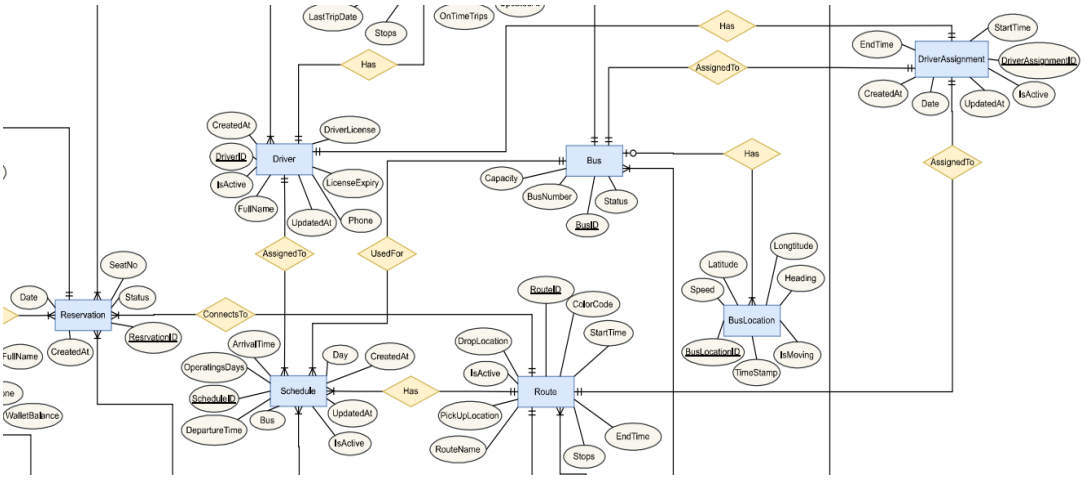
13. Reservations and bus_stops:

- **Relationship:** Many Reservations can use the same Bus Stop.
- **Cardinality:** Many-to-One
- **Details:** A reservation has exactly one pickup stop and one drop-off stop. A single bus stop can serve as the pickup point for many different reservations and as the drop-off point for many others.
- **Foreign Keys:** busstops_id



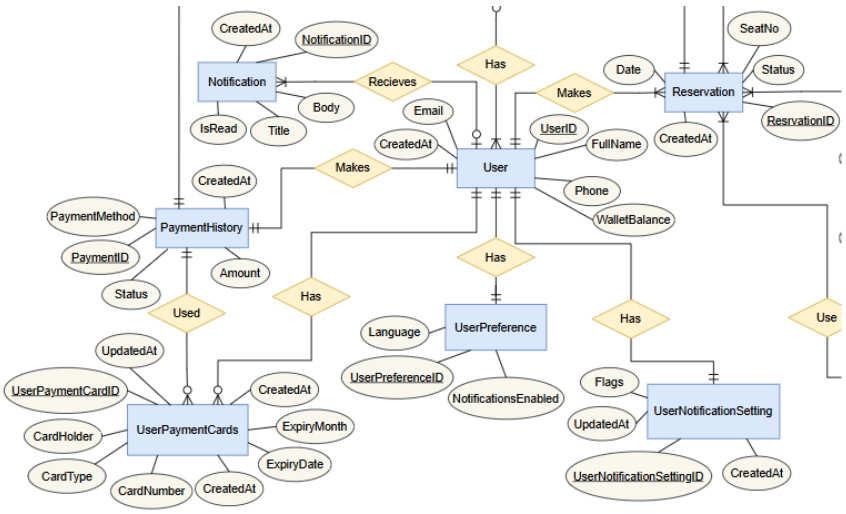
14. Reservations and routes/buses:

- **Relationship:** A reservation connects to one Route and one Bus.
- **Cardinality:** Many-to-One (N:1)
- **Details:** Many reservations can be made for the same route or the same bus.
- **Foreign Keys:** route_id and bus_id



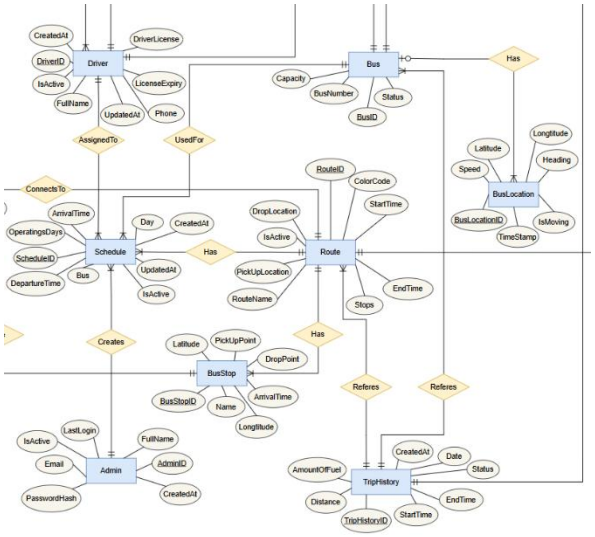
15. Payment_history:

- **payment_history & users:** Many-to-One (a user has many payments).
- **payment_history & user_payment_cards:** Many-to-One (a card can be used for many payments).
- **payment_history & reservations:** One-to-One (a payment is for one reservation).
- **Foreign Keys:** user_id, usercard_id, reservation_id.



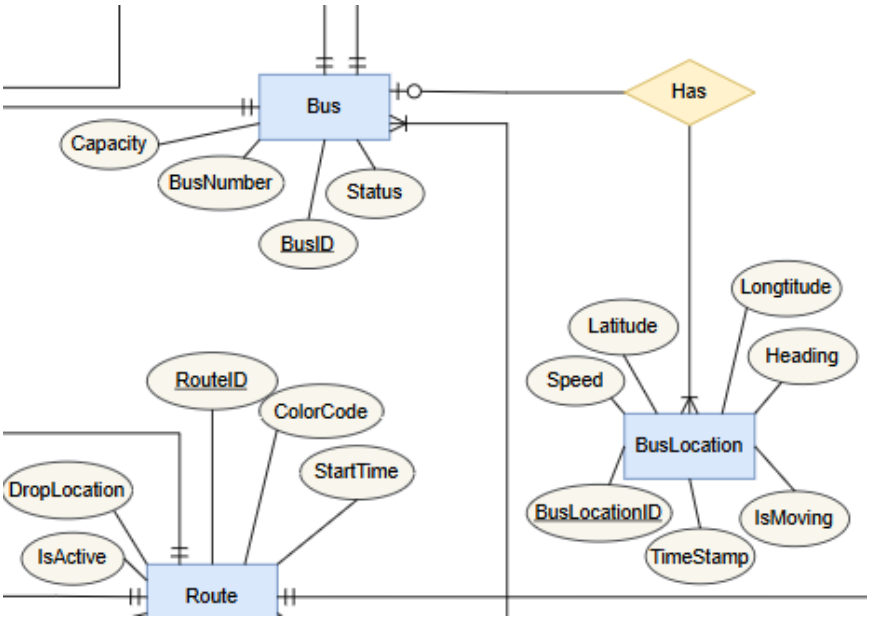
16. Trip_history:

- **Relationship:** A trip history record logs a completed trip.
- **Cardinality:** Many-to-One for each core entity.
- **Details:** Many trip history records can be associated with the same route, bus, or driver. Each record in trip_history refers to a single instance of each.
- **Foreign Keys:** route_id, bus_id, driver_id.



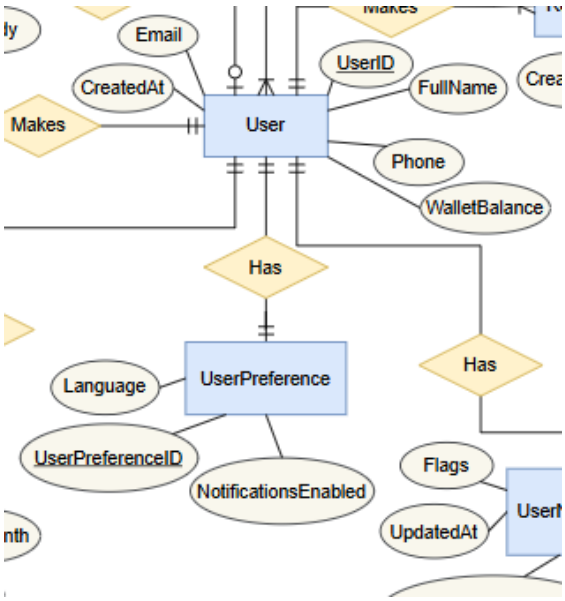
17. Bus Locations and Buses:

- **Relationship:** One Bus has Many Location logs.
- **Cardinality:** One-to-Many
- **Details:** The system continuously logs the location of a bus over time, creating many entries for a single vehicle.
- **Foreign Key:** bus_id



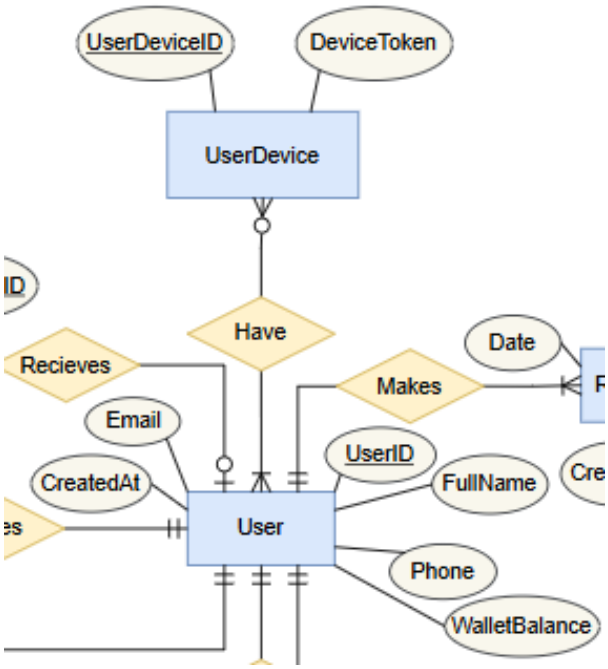
18. User and UserPrefernce:

- **Relationship:** One User has One Preference.
- **Cardinality:** One-to-One
- **Foreign Key:** user_id



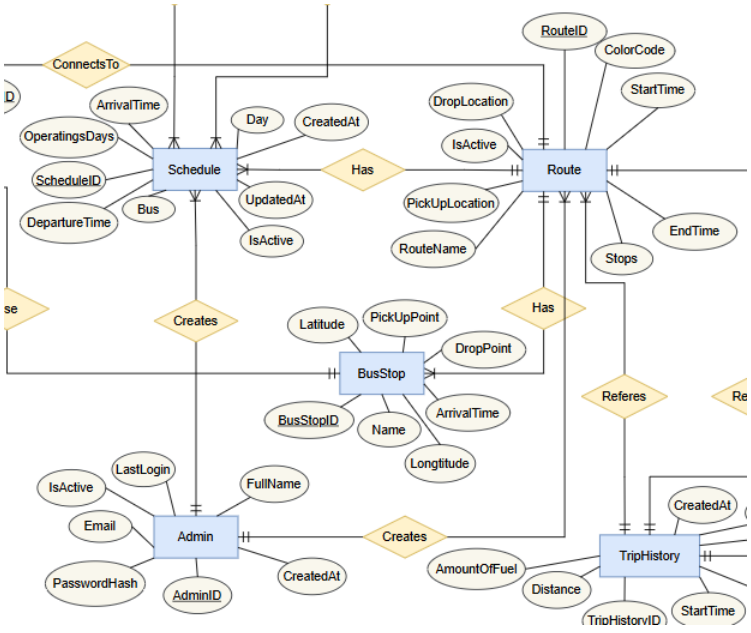
19. User and UserDevice:

- **Relationship:** Many Users can have many devices
- **Cardinality:** Many-to-Many
- **Foreign Key:** user_id



20. Admin, Schedule, and Route:

- **Relationship:** One Admin can create many routes and schedules
- **Cardinality:** One-to-Many
- **Foreign Key:** route_id, schedule_id



4.7 UML Class Diagram

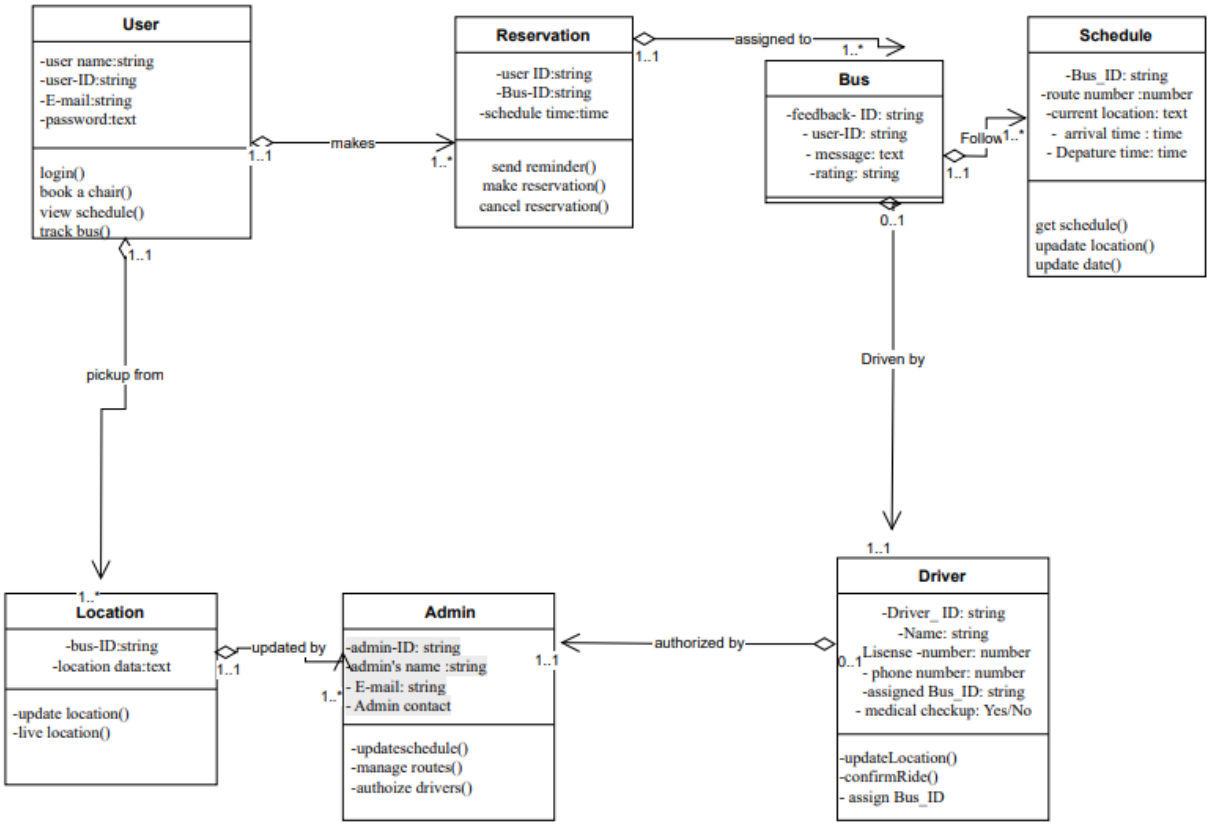


Figure 9: UML Class Diagram

1. User

- **Purpose:** Represents a customer or passenger who uses the service.
- **Attributes:**
    - user name: string: The user's display name.
    - user-ID: string: A unique identifier for the user.
    - E-mail: string: The user's email address, likely for communication and login.
    - password: text: The user's password for authentication.
- **Methods:**

- login(): Authenticates the user.
- book a chair(): Initiates the process of making a reservation.
- view schedule(): Allows the user to see available bus schedules.
- track bus(): Enables the user to see the live location of a bus.

2. Reservation

- **Purpose:** Represents a booking made by a user for a specific bus journey.
- **Attributes:**
  - user ID: string: Links the reservation to the user who made it.
  - Bus-ID: string: Links the reservation to a specific bus.
  - schedule time: time: The scheduled time of the journey for this reservation.
- **Methods:**
  - send reminder(): Sends a notification to the user about the upcoming trip.
  - make reservation(): Finalizes the booking process.
  - cancel reservation(): Cancels a confirmed booking.

3. Bus

- **Purpose:** Represents a physical bus in the fleet.
- **Attributes:**
  - feedback-ID: string, user-ID: string, message: text, rating: string: These attributes seem to describe user feedback. It's an unconventional design to place them directly in the Bus class
- **Methods:** No methods are listed for this class.

4. Schedule

- **Purpose:** Defines the details of a specific route or trip that a bus undertakes.
- **Attributes:**
  - Bus_ID: string: Identifies the bus assigned to this schedule.

- route number: number: The unique number for the route.
- current location: text: The last known location of the bus on this route.
- arrival time: time: The estimated time of arrival at the destination.
- Depature time: time (Typo for Departure): The time the bus starts its journey.

- **Methods:**
  - get schedule(): Retrieves schedule details.
  - upadate location() (Typo for update): Updates the bus's current location.
  - update date(): Updates the date of the schedule.

5. Driver

- **Purpose:** Represents the person driving the bus.
- **Attributes:**
  - Driver_ID: string: A unique identifier for the driver.
  - Name: string: The driver's full name.
  - Lisense -number: number (Typo for License): The driver's license number.
  - phone number: number: The driver's contact number.
  - assigned Bus_ID: string: The ID of the bus the driver is currently assigned to.
  - medical checkup: Yes/No: Indicates the status of the driver's medical clearance.
- **Methods:**
  - updateLocation(): Allows the driver to update the bus's current location.
  - confirmRide(): Confirms the start of a scheduled trip.
  - assign Bus_ID: Assigns the driver to a specific bus.

6. Admin

- **Purpose:** Represents a system administrator with privileged access to manage the system.
- **Attributes:**
  - admin-ID: string: A unique identifier for the administrator.
  - admin's name: string: The administrator's name.
  - E-mail: string: The administrator's email.
  - Admin contact: The administrator's contact information.
- **Methods:**
  - updateschedule(): Modifies or creates bus schedules.
  - manage routes(): Adds, removes, or edits bus routes.
  - authorize drivers() (Typo for authorize): Approves and manages driver accounts.

7. Location

- **Purpose:** Stores the geographical location data of a bus.
- **Attributes:**
  - bus-ID: string: The ID of the bus being tracked.
  - location data: text: The coordinates or address of the bus.
- **Methods:**
  - update location(): Updates the stored location data.
  - live location(): Provides the most recent location data.

4.8 Story Board

Scene 1: App Opening & Login

- The user opens the app.
- They see two buttons: **"Login"** and **"Sign Up."**
- The user enters their university email and password.
- The app checks if the information is correct and takes them to the main screen.

Purpose: Make sure only real users from the university can use the app.

Scene 2: Choose University

- The app asks the user to **choose their university** from a list.
- The user picks their university name.
- The app now shows only the bus routes and schedules for that university.

Purpose: Show information that matches the user's university.

Scene 3: See Bus Routes & Times

- The user clicks on **"View Bus Schedule."**
- They see a list of bus stops, routes, and times.
- The user chooses when and where they want to take the bus.

Purpose: Help the user plan when and where to catch the bus.

Scene 4: Reserve a Seat

- The user clicks **"Reserve a Seat."**
- The app checks if there are seats available.
- If yes, the user gets a **confirmation message**.
- If not, they are added to a **waiting list**.

Purpose: Make sure buses are not overcrowded.

Scene 5: Track the Bus Live

- The user taps **"Track My Bus."**
- A **map opens**, showing where the bus is right now.
- The app also shows how many minutes are left until the bus arrives.

Purpose: Let users know exactly where the bus is so they don't wait too long.

Scene 6: Get Notifications

- The app sends **alerts** like:
    - "Your bus is 5 minutes away."
    - "Your seat is confirmed."
    - "Bus is running late."

Purpose: Keep users updated in real-time.

Scene 7: Driver Screen

- Drivers log in to see their **assigned route and time**.
- They also **update the bus location** while driving.
- This helps the app show the **live location** to students.

Purpose: Help drivers share updates with the app.

Scene 8: Admin Control Panel

- University staff logs into their control panel.
- They can:
    - See how many people booked buses.
    - Change or update bus routes and times.

- Read feedback from users.

Purpose: Help the university manage the system.

What This App Does

- Makes traveling to and from university easier.
- Saves time by showing where buses are.
- Avoids full buses by letting students book seats.
- Helps the university improve bus service using real data.

Figure 10: Story Board

Chapter 5

Data Dictionary

5.1 Process Logic Description

1. User Registration:

When a user wants to register, the system collects their personal details such as full name, email, phone number, university, and department. Upon submission, the system validates the input and stores the user information in the database. The user's account is marked as active, a digital wallet is initialized with a zero balance, and a creation timestamp is recorded. If any required information is missing, the system prompts the user to complete their registration.

2. Admin Creates Route & Schedules:

This process allows an authenticated admin to define a complete bus route and its operational schedules. The process begins with the creation of a route, including key details such as route name, operating hours, and map color coding. The system records the admin's ID as the creator for audit and management purposes.

Next, the admin defines a sequence of bus stops that belong to the route. For each stop, geographic coordinates and the stop order are stored in the database and linked to the corresponding route.

After the stops are configured, the admin sets up multiple schedules for the route. Each schedule defines departure and arrival times, days of operation (stored as a bitmask), and assignments of available buses and drivers. This ensures the route is not only planned but also ready for operational deployment.

This process ensures proper planning, structured data input, and full traceability of the route and schedule setup.

3. Driver Assignment:

Admin users can assign drivers to specific routes and buses. Before assigning, the system checks that both the driver and the bus are active. If valid, a driver assignment record is created with the shift's start and end times and the assignment date. If either the driver or bus is inactive, the system notifies the admin and prevents the assignment.

4. User Makes Reservation:

A logged-in user begins by selecting their university. The system then displays all available routes and schedules for that university. The user chooses a specific schedule and selects a pickup and drop-off stop. The system checks the availability of seats for the chosen bus and time. If available, a reservation record is created and payment is initiated. Upon successful payment, the reservation is confirmed and a notification is sent to the user. If the trip is full, the user is either added to a waitlist or shown an error message.

5. Trip Logging:

Once a scheduled trip is either completed or cancelled, the system logs the trip details including route, bus, driver, start and end times, passenger count, distance traveled, fuel consumed, and any driver notes. It also calculates a punctuality score based on on-time performance. This information is stored for reporting and analytics.

6. Payment Processing:

When a user attempts to pay for a reservation, the system verifies the payment method (card details). If the card is valid, the system encrypts and stores the payment information, links it to the corresponding reservation, and updates the payment status. If the transaction fails, the system informs the user and halts the reservation confirmation.

7. Notifications:

Notifications are triggered by system events such as reservation confirmations, route changes, or delays. If the user's notification settings allow it, the system creates and

sends a notification via push or email. Each notification includes a title, message, type, and read status. If notifications are disabled for the user, no alert is sent.

5.2 Structured English

5.2.1 User Makes a Reservation

FOR a user who wants to make a reservation:

1. SELECT a desired Date.
2. SELECT a desired Route.
3. SYSTEM: DISPLAY available Schedules for the selected Route on that Date.
4. USER: SELECT a specific Schedule (which implies a Bus, Driver, DepartureTime, and ArrivalTime).
5. USER: SELECT a PickupPoint and a DropPoint from the Route's list of BusStops.
6. IF the selected Bus for that Schedule has available Capacity

   THEN

   a. CREATE a new Reservation record with the UserID, ScheduleID, Date, and Status 'Confirmed'.
   b. IF a seat number is assigned

      THEN

      i. RECORD the SeatNo in the Reservation.

      END IF

   c. INITIATE the payment process (see Process 4).
   d. IF payment is successful

      THEN

      i. CONFIRM the reservation.
      ii. SEND a Notification to the User's Device.

      ELSE

      i. CANCEL the reservation attempt.

ii. DISPLAY a "Payment Failed" message to the User.

END IF.

ELSE

a. DISPLAY an error message "Trip is full."
b. END the reservation process.

END IF.

5.2.2 Admin Sets Up a New Route and Schedule

1. CREATE a new Route record with RouteName, Start/End Times, ColorCode, etc.
    a. ASSOCIATE the AdminID with the new Route as the creator.
2. FOR EACH stop on the new Route:
    a. CREATE a new BusStop record with Name, Latitude, and Longitude.
    b. CONNECT the BusStop record to the Route.

END FOR.

3. FOR EACH scheduled trip on the new Route:
    a. CREATE a new Schedule record.
    b. SET the DepartureTime, ArrivalTime, and OperatingDays.
    c. ASSIGN an available Bus to the Schedule.
    d. ASSIGN an available Driver to the Schedule.

END FOR.

5.2.3 Driver Completes a Trip

WHEN a driver starts a scheduled trip:

1. CREATE a new TripHistory record with a 'In Progress' status.
    a. CONNECT the record to the Driver, Bus, and Route for this trip.
    b. RECORD the actual StartTime.
2. WHILE the trip is active:
    a. PERIODICALLY get the current Latitude, Longitude, Speed, and Heading from the bus.

b. CREATE a new BusLocation record with the live data and a Timestamp.

END WHILE.

3. WHEN the trip ends:
    a. UPDATE the TripHistory record:
        i. SET Status to 'Completed'.
        ii. RECORD the actual EndTime.
        iii. RECORD the total Distance and AmountOfFuel used.
    b. UPDATE the associated DriverStatistics record:
        i. INCREMENT TotalTrips.
        ii. UPDATE OnTimeTrips based on actual arrival.
        iii. UPDATE LastTripDate.

5.2.4 System Handles a Payment Transaction

WHEN a payment is required for a reservation:

1. RETRIEVE the User's default UserPaymentCard OR prompt for a new one.
2. CREATE a new PaymentHistory record with an Amount and 'Pending' Status.
    a. CONNECT the PaymentHistory to the User and the Reservation.
3. PROCESS the transaction using the selected PaymentMethod.
4. IF the transaction is successful

    THEN

    a. UPDATE the PaymentHistory Status to 'Succeeded'.

    b. DEDUCT the Amount from the User's selected card.

    ELSE

    a. UPDATE the PaymentHistory Status to 'Failed'.

END IF.

5.2.5 User Manages Profile and Settings

FOR a user managing their account:

1. TO change personal information:
   a. UPDATE the FullName or Phone in the User record.
2. TO manage payment methods:
   a. ADD a new UserPaymentCards record.
   b. DELETE an existing UserPaymentCards record.
   c. SET one UserPaymentCards record as the default.
3. TO change preferences:
   a. UPDATE the UserPreference record with the selected Language.
4. TO change notification settings:
   a. UPDATE the Flags in the UserNotificationSetting record (e.g., enable/disable 'NotificationsEnabled').

5.2.6 Driver Assignment

FOR admin assigning a driver to a route and bus:

1. IF driver and bus are active

   THEN

      a. STORE assignment details (shift times, date)
      b. SET assignment as active

   ELSE

      a. DISPLAY "Driver or bus is not active"

   ENDIF

5.2.7 Trip Logging

FOR any completed or cancelled trip:

1. IF trip is completed OR cancelled
   THEN
      a. STORE trip details (route, bus, driver, time, passenger count, distance)

b. CALCULATE on-time performance
c. STORE fuel consumption and notes
d. SAVE trip history

ELSE

a. DISPLAY "Trip not completed"

ENDIF

5.2.8 Payment Processing

FOR a user making a payment:

1. IF user initiates payment
   THEN
   a. ENCRYPT card number
   b. IF card is valid
      THEN
      i. STORE payment information and amount
      ii. LINK payment to reservation
      iii. SET payment status
      iv. STORE creation timestamp

      ELSE

      i. DISPLAY "Invalid card details"

   ENDIF

   ELSE

   a. DISPLAY "Payment not initiated"

ENDIF

5.2.9 Notifications

FOR a system-triggered notification:

1. IF a system event triggers a notification
   THEN
   a. IF user notifications are enabled
      THEN
      i. CREATE notification record (title, message)
      ii. SET is_read to FALSE
      iii. SEND push or email notification

      ELSE

      i. DISPLAY "Notifications disabled"

   ENDIF

   ELSE

   a. DISPLAY "No notification triggered"

ENDIF

5.3 Logic Decision Table

1. User Registration

Table 7: Logic Decision Tables

Rule	All Required Info Provided	Email Valid	User Already Exists	Store User Info	Activate Account	System Response
R1	Yes	Yes	No	Yes	Yes	User registered successfully
R2	No	-	-	No	No	Show "Please fill all fields"
R3	Yes	No	-	No	No	Show "Invalid email format"
R4	Yes	Yes	Yes	No	No	Show "User already exists"

2. Process: Admin Creates Routes & Schedules

Rule	Admin Authenticated	Route Info Provided	Bus Stops Provided	Schedules Provided	Create Route	Create Bus Stops	Create Schedules	System Response
R1	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Route and schedule created
R2	No	Yes	Yes	Yes	No	No	No	Show "Admin access required"
R3	Yes	No	Yes	Yes	No	No	No	Show "Missing route information"
R4	Yes	Yes	No	Yes	Yes	No	Yes	Show "Please define bus stops"
R4	Yes	Yes	Yes	No	Yes	Yes	No	Show "Please add schedule details"

3. Process: Driver Assignment

Rule	Driver Selected	Bus Selected	Both Are Active	Assigned Driver	System Response
R1	Yes	Yes	Yes	Yes	Driver assigned successfully
R2	Yes	Yes	No	No	Show "Driver or Bus is inactive"
R3	No	Yes	-	No	Show "Driver not selected"
R4	Yes	No	-	No	Show "Bus not selected"

4. Process: User Makes Reservation

Rule	User Logged In	University Selected	Route & Time Selected	Seats Available	Make Reservation	System Response
R1	Yes	Yes	Yes	Yes	Yes	Reservation confirmed
R2	Yes	Yes	Yes	No	No	Show "Trip is full – added to waitlist"
R3	Yes	Yes	No	-	No	Show "Please select route and time"
R4	Yes	No	-	-	No	Show "Please select your university"
R5	No	-	-	-	No	Show "Please login first"

5. Process: Trip Logging

Rule	Trip Completed	Trip Cancelled	Log Trip Details	System Response
R1	Yes	-	Yes	Trip log saved
R2	-	Yes	Yes	Cancelled trip log saved
R3	No	No	No	Show "Trip not finalized yet"

6. Process: Payment Processing

Rule	Payment Initiated	Card Valid	Payment Success	Store Payment	System Response
R1	Yes	Yes	Yes	Yes	Payment confirmed
R2	Yes	Yes	No	No	Show "Payment failed"
R3	Yes	No	-	No	Show "Invalid card details"
R4	No	-	-	No	Show "Payment not started"

7. **Process: Notifications**

Rule	Event Occurred	Notifications Enabled	Send Notification	System Response
R1	Yes	Yes	Yes	Notification sent
R2	Yes	No	No	Show "Notifications disabled"
R3	No	-	No	No notification triggered

5.4 Data Dictionary

The data dictionary is a centralized repository of information, serving as the definitive guide for all data objects identified during systems development. A core principle of its use is that every object, along with each of its individual components, must have a corresponding and complete definition documented within it. The primary purpose of a data dictionary is to catalogue and communicate the precise structure and content of data, providing meaningful and unambiguous descriptions for all individually named data objects. To ensure it remains accurate and relevant, business analysts and domain experts must work together to create and maintain it as a living document throughout the entire lifecycle of the project.

5.5 Data Element

5.5.1 Entity: User

- **Alias:** UserID
    - **Description:** A unique system-generated identifier for the user. This is the Primary Key.
    - **Values:** A unique string or number (e.g., UUID).
    - **Range:** N/A.
- **Alias:** FullName

    - **Description:** The full name of the user.
    - **Values:** Text string.
    - **Range:** e.g., 1-255 characters.
- **Alias:** Email
    - **Description:** The user's unique email address, likely used for login and notifications.
    - **Values:** A valid email address format.
    - **Range:** e.g., 1-255 characters.
- **Alias:** Phone
    - **Description:** The user's contact phone number.
    - **Values:** A valid phone number format.
    - **Range:** e.g., 10-20 characters.
- **Alias:** WalletBalance
    - **Description:** The current monetary balance in the user's digital wallet.
    - **Values:** Numeric decimal value.
    - **Range:** 0.00 or greater.
- **Alias:** CreatedAt
    - **Description:** The date and time when the user account was created.
    - **Values:** A valid timestamp.
    - **Range:** N/A.

5.5.2 Entity Reservation

Represents a booking made by a User for a specific trip.

- **Alias:** ReservationID

    - **Description:** A unique identifier for the reservation. This is the Primary Key.
    - **Values:** A unique string or number.
    - **Range:** N/A.
- **Alias:** Date
    - **Description:** The specific date for which the reservation is valid.
    - **Values:** A valid date.
    - **Range:** N/A.
- **Alias:** Status
    - **Description:** The current status of the reservation.
    - **Values:** Enumerated text: 'Confirmed', 'Cancelled', 'Completed', 'No-Show'.
    - **Range:** Limited to the set of defined values.
- **Alias:** SeatNumber
    - **Description:** The assigned seat number for the reservation, if applicable.
    - **Values:** Alphanumeric text string (e.g., "A1", "14B").
    - **Range:** e.g., 1-5 characters.
- **Alias:** CreatedAt
    - **Description:** The date and time when the reservation was made.
    - **Values:** A valid timestamp.
    - **Range:** N/A.

5.5.3 Entity: PaymentHistory

Represents a financial transaction, usually linked to a User or Reservation.

- **Alias:** PaymentID

- **Description:** A unique identifier for the payment transaction. This is the Primary Key.
- **Values:** A unique string or number.
- **Range:** N/A.

- **Alias:** Amount
  - **Description:** The monetary amount of the transaction.
  - **Values:** Numeric decimal value.
  - **Range:** Greater than 0.00.
- **Alias:** Status
  - **Description:** The status of the payment transaction.
  - **Values:** Enumerated text: 'Succeeded', 'Failed', 'Pending', 'Refunded'.
  - **Range:** Limited to the set of defined values.
- **Alias:** PaymentMethod
  - **Description:** The method used for the payment (e.g., wallet, credit card).
  - **Values:** Text string.
  - **Range:** e.g., 'Wallet', 'Card'.
- **Alias:** CreatedAt
  - **Description:** The date and time when the payment was initiated.
  - **Values:** A valid timestamp.
  - **Range:** N/A.
- **Alias:** UpdatedAt
  - **Description:** The date and time when the payment status was last updated.
  - **Values:** A valid timestamp.

- **Range:** N/A.

5.5.4 Entity: UserPaymentCards

Represents a payment card saved by a user.

- **Alias:** UserPaymentCardID
  - **Description:** A unique identifier for the saved card. This is the Primary Key.
  - **Values:** A unique string or number.
  - **Range:** N/A.
- **Alias:** CardNumber
  - **Description:** The user's credit/debit card number (should be encrypted).
  - **Values:** Encrypted text string.
  - **Range:** N/A.
- **Alias:** CardHolder
  - **Description:** The name of the person as it appears on the card.
  - **Values:** Text string.
  - **Range:** e.g., 1-255 characters.
- **Alias:** ExpiryDate / ExpiryMonth
  - **Description:** The expiration date of the card.
  - **Values:** A valid date (MM/YYYY).
  - **Range:** Future dates only.
- **Alias:** CardType
  - **Description:** The brand of the card.
  - **Values:** Enumerated text: 'Visa', 'Mastercard', 'Amex', etc.
  - **Range:** Limited to supported card types.

- **Alias:** CreatedAt
    - **Description:** The date and time when the card were added.
    - **Values:** A valid timestamp.
    - **Range:** N/A.

5.5.5 Entity: Driver

Represents a bus driver employed by the service.

- **Alias:** DriverID
    - **Description:** A unique identifier for the driver. This is the Primary Key.
    - **Values:** A unique string or number.
    - **Range:** N/A.
- **Alias:** FullName
    - **Description:** The full name of the driver.
    - **Values:** Text string.
    - **Range:** e.g., 1-255 characters.
- **Alias:** Phone
    - **Description:** The driver's contact phone number.
    - **Values:** A valid phone number format.
    - **Range:** e.g., 10-20 characters.
- **Alias:** DriverLicense
    - **Description:** The driver's license number.
    - **Values:** Alphanumeric text string.
    - **Range:** e.g., 5-30 characters.
- **Alias:** LicenseExpiry

    - **Description:** The expiration date of the driver's license.
    - **Values:** A valid date.
    - **Range:** Future dates only.
- **Alias:** IsActive
    - **Description:** A flag indicating if the driver is currently active and eligible for assignments.
    - **Values:** Boolean (True, False).
    - **Range:** N/A.
- **Alias:** CreatedAt / UpdatedAt
    - **Description:** Timestamps for record creation and last update.
    - **Values:** A valid timestamp.
    - **Range:** N/A.

5.5.6 Entity: Bus

- **Alias:** BusID
    - **Description:** A unique identifier for the bus. This is the Primary Key.
    - **Values:** A unique string or number.
    - **Range:** N/A.
- **Alias:** BusNumber
    - **Description:** The public-facing number of the bus (e.g., "07", "12A").
    - **Values:** Alphanumeric text string.
    - **Range:** e.g., 1-10 characters.
- **Alias:** Capacity
    - **Description:** The maximum number of passengers the bus can hold.

    - **Values:** Positive integer.
    - **Range:** e.g., 1-100.
- **Alias:** Status
    - **Description:** The current operational status of the bus.
    - **Values:** Enumerated text: 'Active', 'In Maintenance', 'Out of Service'.
    - **Range:** Limited to the set of defined values.

5.5.7 Entity: Route

Represents a defined bus route with a start, end, and stops.

- **Alias:** RouteID
    - **Description:** A unique identifier for the route. This is the Primary Key.
    - **Values:** A unique string or number.
    - **Range:** N/A.
- **Alias:** RouteName
    - **Description:** The name of the route (e.g., "Campus Loop A").
    - **Values:** Text string.
    - **Range:** e.g., 1-255 characters.
- **Alias:** PickUpLocation / DropLocation
    - **Description:** A general text description of the route's start and end areas.
    - **Values:** Text string.
    - **Range:** N/A.
- **Alias:** StartTime / EndTime
    - **Description:** The typical start and end times for service on this route.
    - **Values:** A valid time format (HH:MM).

    - **Range:** N/A.

- **Alias:** ColorCode
    - **Description:** A hex color code used to represent the route on a map.
    - **Values:** Hex color string (e.g., "#FF5733").
    - **Range:** 7 characters, starting with '#'.
- **Alias:** IsActive
    - **Description:** A flag indicating if the route is currently in service.
    - **Values:** Boolean (True, False).
    - **Range:** N/A.

5.5.8 Entity: BusStop

- **Alias:** BusStopID
    - **Description:** A unique identifier for the bus stop. This is the Primary Key.
    - **Values:** A unique string or number.
    - **Range:** N/A.
- **Alias:** Name
    - **Description:** The name of the bus stop (e.g., "Library Entrance").
    - **Values:** Text string.
    - **Range:** e.g., 1-255 characters.
- **Alias:** Latitude / Longitude
    - **Description:** The precise geographic coordinates of the bus stop.
    - **Values:** Numeric decimal value.
    - **Range:** Latitude: -90 to +90. Longitude: -180 to +180.
- **Alias:** ArrivalTime

- **Description:** The estimated time of arrival at this stop for a given schedule.
- **Values:** A valid time format (HH:MM).
- **Range:** N/A.

5.5.9 Entity: Schedule

- **Alias:** ScheduleID
  - **Description:** A unique identifier for the schedule entry. This is the Primary Key.
  - **Values:** A unique string or number.
  - **Range:** N/A.
- **Alias:** DepartureTime / ArrivalTime
  - **Description:** The scheduled departure and arrival times for the trip.
  - **Values:** A valid time format (HH:MM).
  - **Range:** N/A.
- **Alias:** OperatingDays
  - **Description:** The days of the week this schedule is active.
  - **Values:** A list or bitmask representing days (e.g., "Mon, Wed, Fri").
  - **Range:** Any combination of the 7 days of the week.
- **Alias:** IsActive
  - **Description:** A flag indicating if this specific schedule is currently active.
  - **Values:** Boolean (True, False).
  - **Range:** N/A.

5.5.10 Entity: TripHistory

- **Alias:** TripHistoryID

    - **Description:** A unique identifier for the historical trip record. This is the Primary Key.
    - **Values:** A unique string or number.
    - **Range:** N/A.
- **Alias:** Date
    - **Description:** The date the trip occurred.
    - **Values:** A valid date.
    - **Range:** N/A.
- **Alias:** StartTime / EndTime
    - **Description:** The actual start and end times of the trip.
    - **Values:** A valid timestamp.
    - **Range:** N/A.
- **Alias:** Distance
    - **Description:** The actual distance traveled during the trip.
    - **Values:** Positive numeric decimal value (e.g., in km or miles).
    - **Range:** 0 or greater.
- **Alias:** AmountOfFuel
    - **Description:** The amount of fuel consumed during the trip.
    - **Values:** Positive numeric decimal value (e.g., in liters or gallons).
    - **Range:** 0 or greater.
- **Alias:** Status
    - **Description:** The final status of the trip.
    - **Values:** Enumerated text: 'Completed', 'Cancelled'.

- **Range:** Limited to the set of defined values.

5.5.11 Entity: Admin

- **Alias: :** AdminID
  - **Description:** A unique identifier for the administrator. This is the Primary Key.
  - **Values:** A unique string or number.
  - **Range:** N/A.
- **Alias:** FullName
  - **Description:** The full name of the administrator.
  - **Values:** Text string.
  - **Range:** e.g., 1-255 characters.
- **Alias:** Email
  - **Description:** The admin's unique email address for login.
  - **Values:** A valid email address format.
  - **Range:** e.g., 1-255 characters.
- **Alias:** PasswordHash
  - **Description:** The hashed password for the admin account.
  - **Values:** A hashed string.
  - **Range:** N/A.
- **Alias:** LastLogin
  - **Description:** The timestamp of the admin's last login.
  - **Values:** A valid timestamp.
  - **Range:** N/A.
- **Alias:** IsActive

- **Description:** A flag indicating if the admin account is active.
- **Values:** Boolean (True, False).
- **Range:** N/A.

5.6 Data External Entity

a. User:

- Interacts with the system to send a **Reservation Request** specifying their intention to book a seat on a university bus.
- Provide their **Pick-Up Point** to facilitate scheduling.
- Receives key information from the system, including:
  - **Bus Location:** The real-time position of the bus.
  - **Bus Schedule:** Timetable details to plan their journey.

b. University:

- Provides the system with **Registration Details** for authorized users and buses.
- Provides the **Schedule** of buses for integration into the system.
- Receives **Passenger Count** data, helping the university monitor bus utilization.
- Receives **Bus/Route Status** for updates on bus/route conditions, delays, or incidents.

c. Driver:

- Updates the system with the bus's **Current Location**, enabling real-time tracking for users and the university.
- Communicates **Bus/Route Status**, which helps the system calculate accurate arrival times and optimize routes.
- Updates the system with its License information, to verify driver's credentials.
- Receives the **Schedule** from the system, detailing their assigned routes and timings.
- Receives **Medical & License** checkup date from the system, in order to update it.

5.7 Data Process

- **Select University** – User selects a university to access its bus data.
- **User Registration/Login** – Authenticates users (students, drivers, admins).
- **Make Reservation** – Allows users to book bus seats.
- **View Route & Schedule** – Displays available routes and timings.
- **Track Bus (Live Location)** – Retrieves and displays real-time GPS bus location.
- **Manage Routes & Schedules** – Admins can update or assign new bus schedules.
- **Driver Route Viewing** – Enables drivers to check their routes and schedules.
- **Update Bus Status** – Receives and stores real-time location from drivers.
- **License & Medical Validation** – Checks and updates driver compliance info.

5.8 Data Stores

- **D1: University List** – Stores available universities.
- **D2: Users** – Holds user credentials and profiles.
- **D3: Reservations** – Stores reservation records.
- **D4: Route & Schedule** – Contains bus route and schedule data.
- **D5: Bus Location** – Stores real-time location data from GPS.
- **D6: Drivers** – Stores driver profiles, license, and medical data.

5.9 Data Flow

- **User Credentials** → from user to Login Process.
- **Reservation Request** → from user to Reservation Process.
- **Selected University** → from user to system after university choice.
- **Bus Location Data** → from driver's device to Bus Tracking Process.
- **Feedback Submission** → from user/driver to Feedback Process.
- **Updated Schedule** → from Admin to Manage Routes Process.
- **License and Medical Checkup Status** → from driver to validation process.
- **Real-time Map Updates** → from Bus Tracking Process to User Interface.

Chapter 6

System Implementation

6.1 Overview

To build UniTracker, we selected a modern and efficient technology stack tailored to the needs of a university bus tracking system. Our goal was to create a scalable, cross-platform, real-time application with a strong user experience. We used Flutter for the frontend and Supabase for the backend, supported by tools like Google Maps API, OneSignal, Postman, and Figma. Below is a detailed description of the tools we used and how we applied them throughout the implementation phase.

6.2 Front-End Implementation

- **Flutter**

  We developed the entire user interface using Flutter. This allowed us to build a single codebase that works seamlessly on Android, iOS, and web platforms. Flutter's widget library made it easy to create a modern, intuitive, and responsive interface, including map displays, reservation screens, bus details, and user profiles. Features like hot reload significantly sped up our testing and iteration process.

- **Dart**

  All frontend logic was written in Dart, Flutter's native language. We used Dart's asynchronous features to handle data fetching and real-time updates, especially for the live bus tracking component. The learning curve was manageable, and it allowed us to maintain clean and modular code throughout the project.

6.3 Back-End Implementation

- **Supabase**

  We used Supabase as our Backend-as-a-Service platform. It provided us with everything we needed: authentication, a real-time database, API services, and storage. We integrated Supabase to handle user sign-in/sign-up processes, manage roles (students, drivers, admins), and sync data in real-time between users and the system.

- **PostgreSQL (via Supabase)**

  All structured data, such as user accounts, bus schedules, routes, driver information, and reservation history was stored in PostgreSQL, managed by Supabase. We carefully designed the schema to ensure normalization, reduce redundancy, and enable fast queries for future analytics like trip optimization.

6.4 Real-Time Bus Tracking

- **Google Maps API**

  To visualize bus locations and routes, we integrated the Google Maps API within our Flutter application. We used google_maps_flutter to display interactive maps. Real-time bus coordinates were plotted using live data streamed from Supabase, providing users with up-to-date location tracking and smooth navigation features.

6.5 Notifications and Alerts

- **OneSignal**

  To enhance communication with users, we implemented OneSignal for push notifications. We configured it to notify students of upcoming bus arrivals, delays, or cancellations, even when the app is closed. This real-time alert system significantly improved the overall experience and reduced uncertainty for students relying on bus services.

6.6 UI/UX Design and Prototyping

- **Figma**

  Before development, we created detailed wireframes and mockups using Figma. This helped us visualize user journeys, define screen layouts, and plan consistent UI elements. We used these designs as a blueprint during implementation, ensuring the final application matched our visual and functional goals. We chose the Poppins font and a modern color palette to keep the app visually appealing and accessible.

6.7 Development Environment

- **Android Studio**

  We used Android Studio as our main development environment. With Flutter and Dart plugins, it supported code writing, real-time debugging, and device emulation. We tested the app on various Android Virtual Devices (AVDs) to ensure compatibility across different screen sizes and Android versions. We also configured native Android files for permissions like location access and push notifications.

6.8 API Testing and Integration

- **Postman**

  Throughout the development process, we used Postman to test our Supabase API endpoints. This helped us ensure that data sent from the frontend was correctly handled by the backend. We tested various scenarios such as login, reservation requests, and live location updates. This proactive testing helped us catch and fix issues early.

6.9 Database Design and Reporting

- **Database Report**

  We documented the backend schema in a structured database report. It includes:

  - An **Entity Relationship Diagram (ERD)** showing the relationships between tables like Users, Buses, Schedules, Reservations, and Feedback.

- **Normalization steps** applied to reduce redundancy and maintain data integrity.
- **Indexing strategies** to improve performance.
- **Backup and recovery** procedures provided by Supabase to ensure data safety.

This documentation ensures our system remains maintainable and scalable in the future.

Chapter 7

UI Manual

6.1 Student Mobile App

1. Home Page:

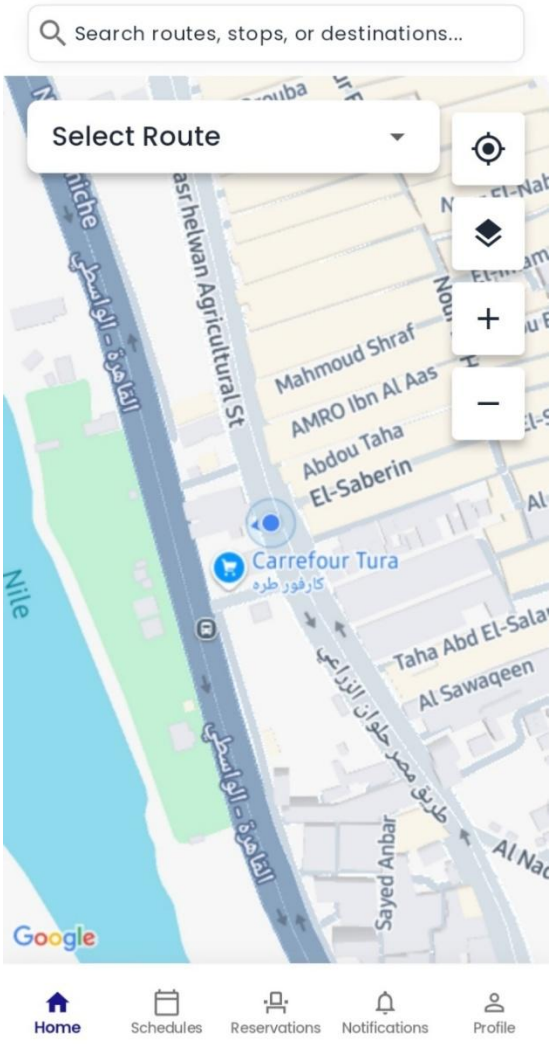
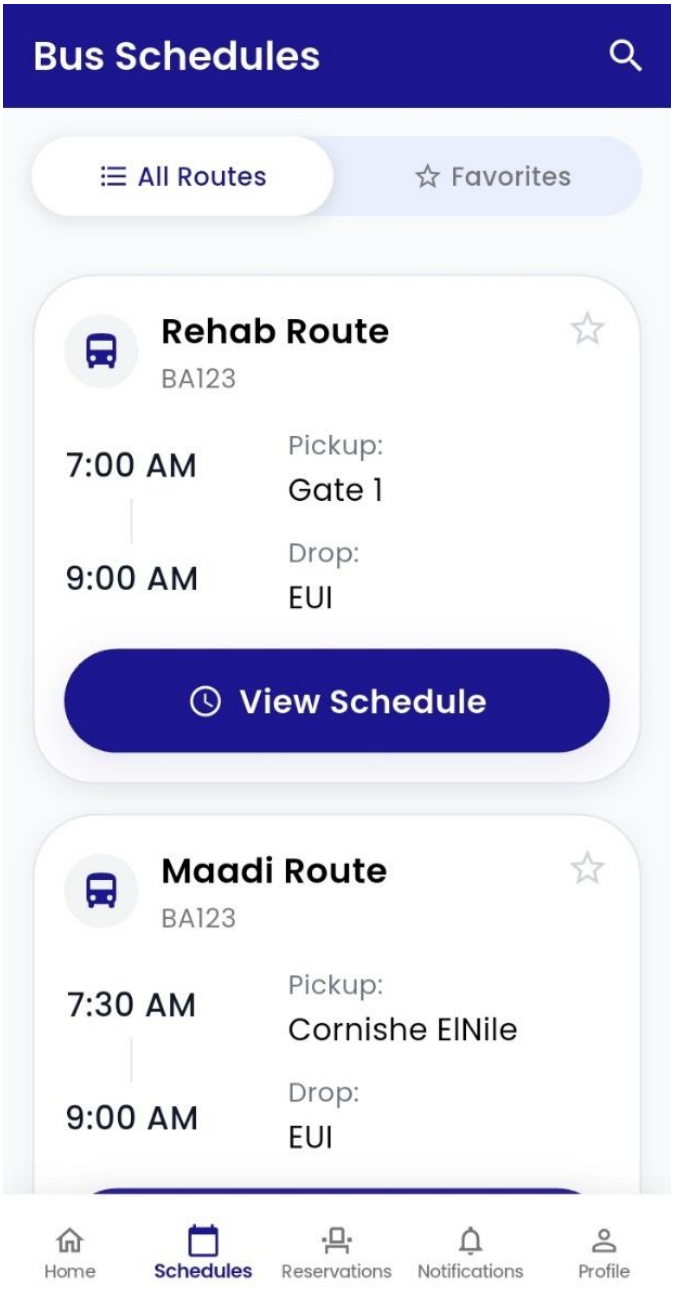


Figure 11: Mobile App UI Manual

Description: The Home Page serves as the main entry point, displaying an interactive map. Users can immediately search for routes and destinations or select a specific bus line to view its path.

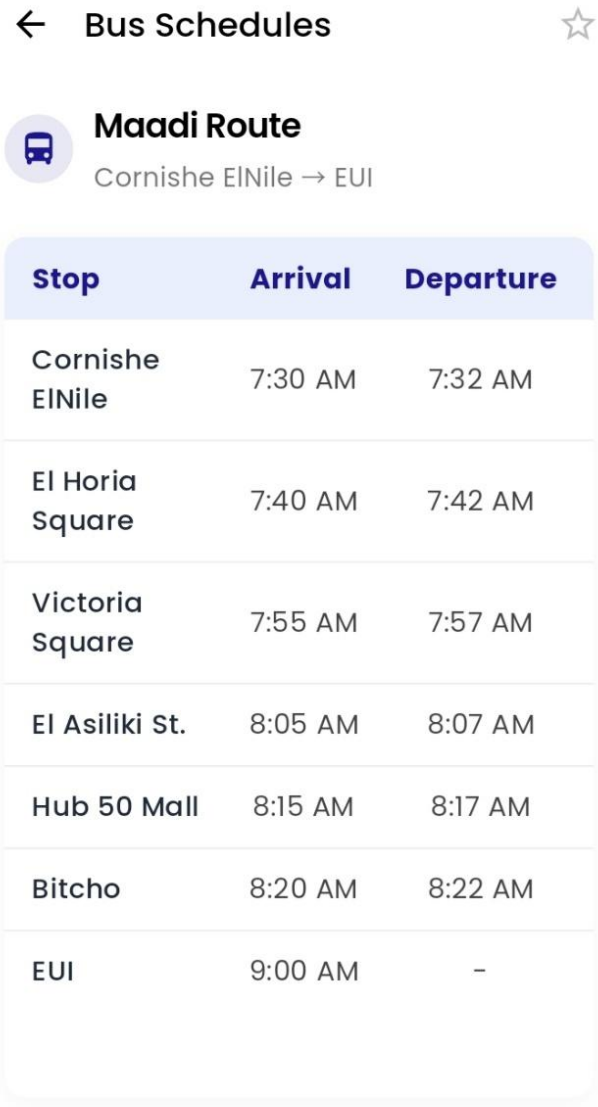
2. Schedules Page:

a. Bus Schedules Page:



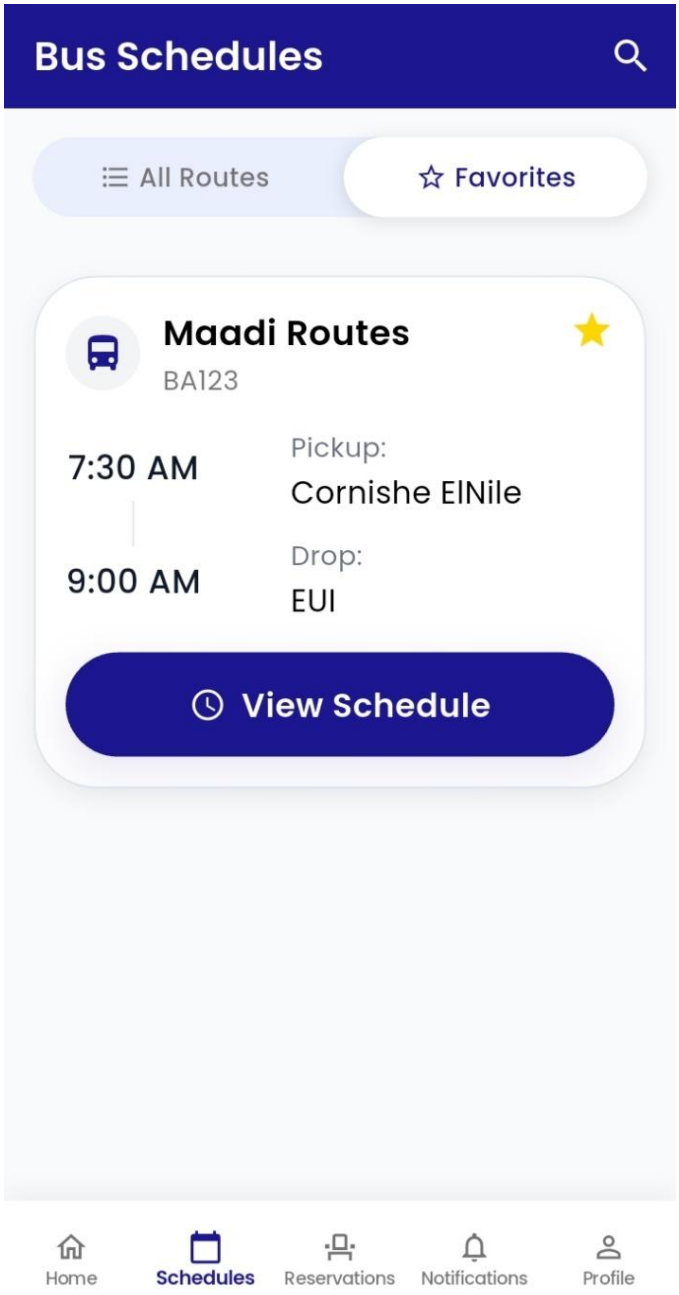
Description: The Bus Schedules page provides a comprehensive list of all available routes. Each entry summarizes key information, including pickup/drop-off times and locations, with an option to view the full schedule or mark a route as a favorite for quick access.

b. Bus Stops Page:



Description: This screen provides a detailed timetable for a selected bus route. It clearly lists all designated stops along the route, along with their scheduled arrival and departure times, allowing users to plan their journey precisely.

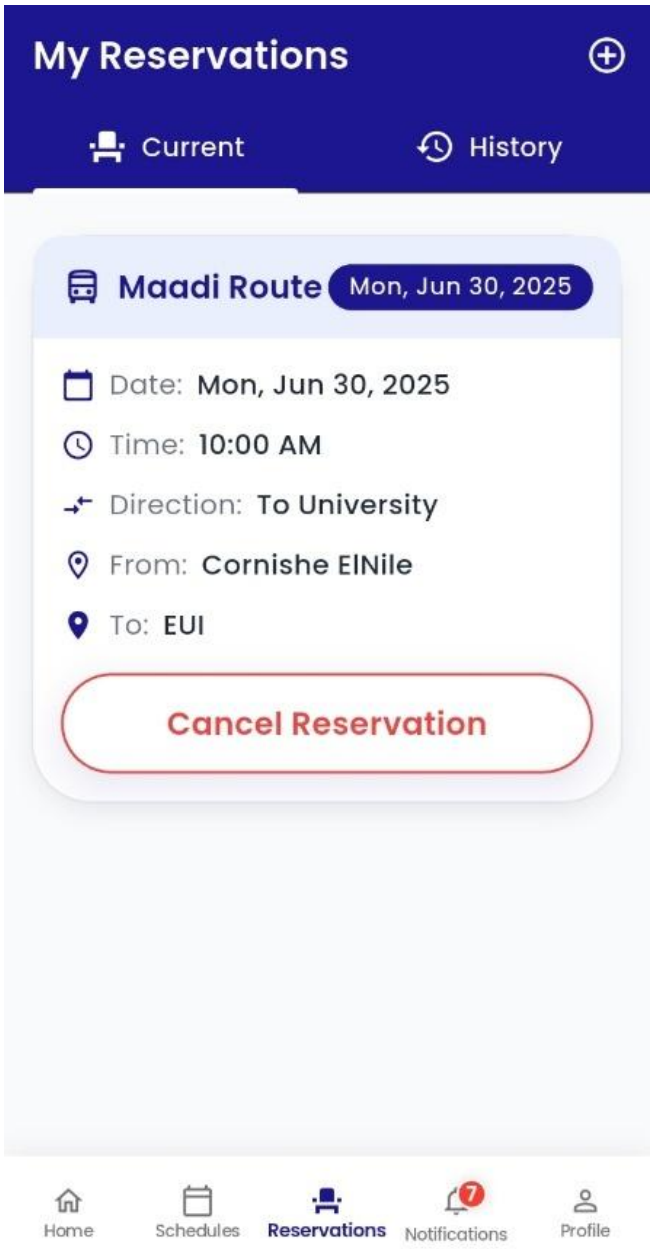
c. Favourites Page:



Description: The Favourites page offers a personalized view, displaying only the bus routes that the user has previously saved. This feature enables quick and convenient access to the schedules they use most often.

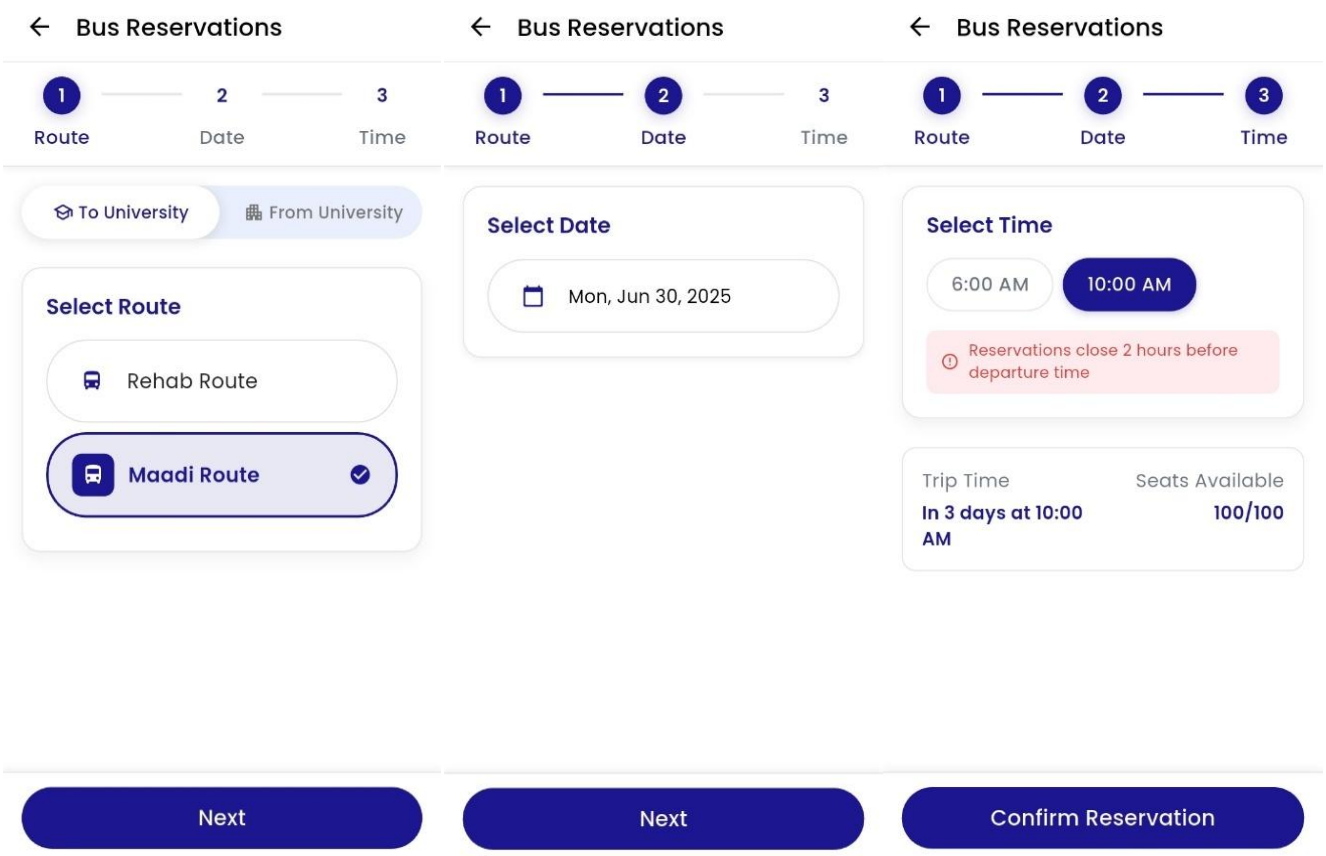
3. Reservations Screen:

a. Current Reservations Page:



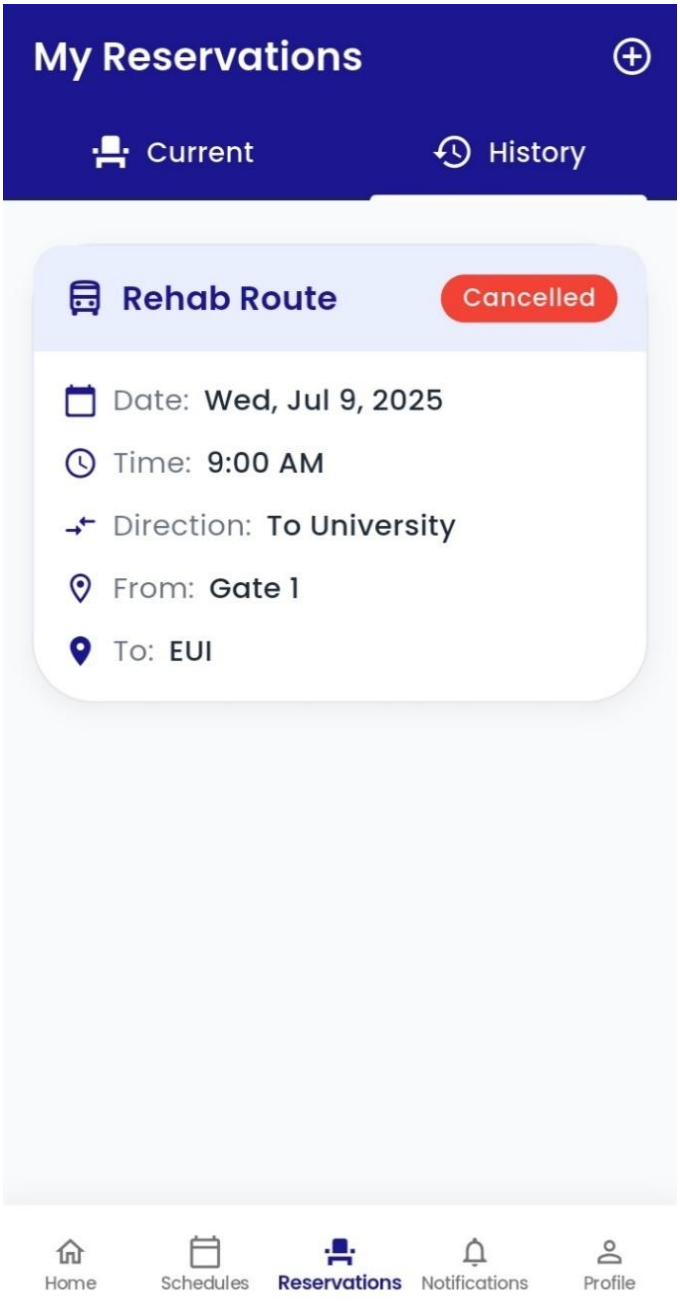
Description: The Reservations screen displays a user's upcoming trips. Each reservation card details the route, date, time, and direction. Users can easily manage their bookings from this page, with an option to cancel if needed.

b. Select Route & Select Date & Select Time Pages:



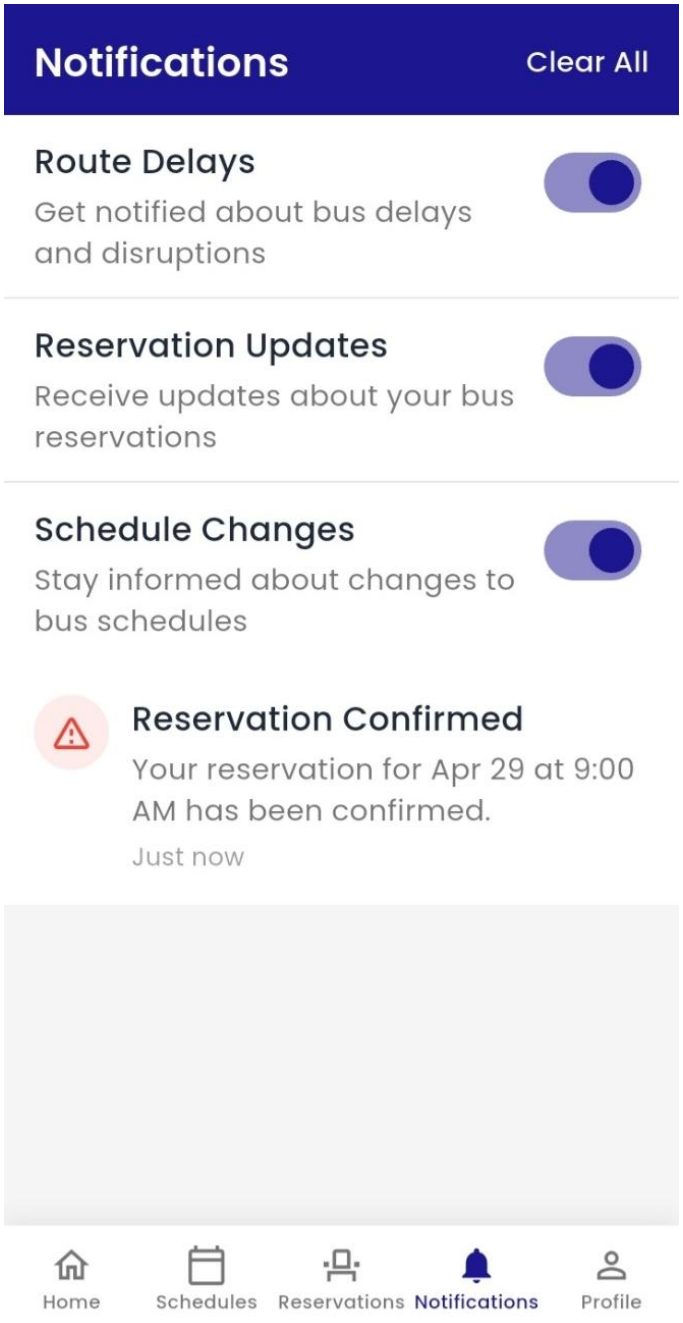
Description: This sequence showcases the guided, three-step process for creating a new bus reservation. A progress indicator guides the user as they first select their desired route and direction, then choose a date, and finally pick an available trip time before confirming their booking.

c. History Reservations Page:



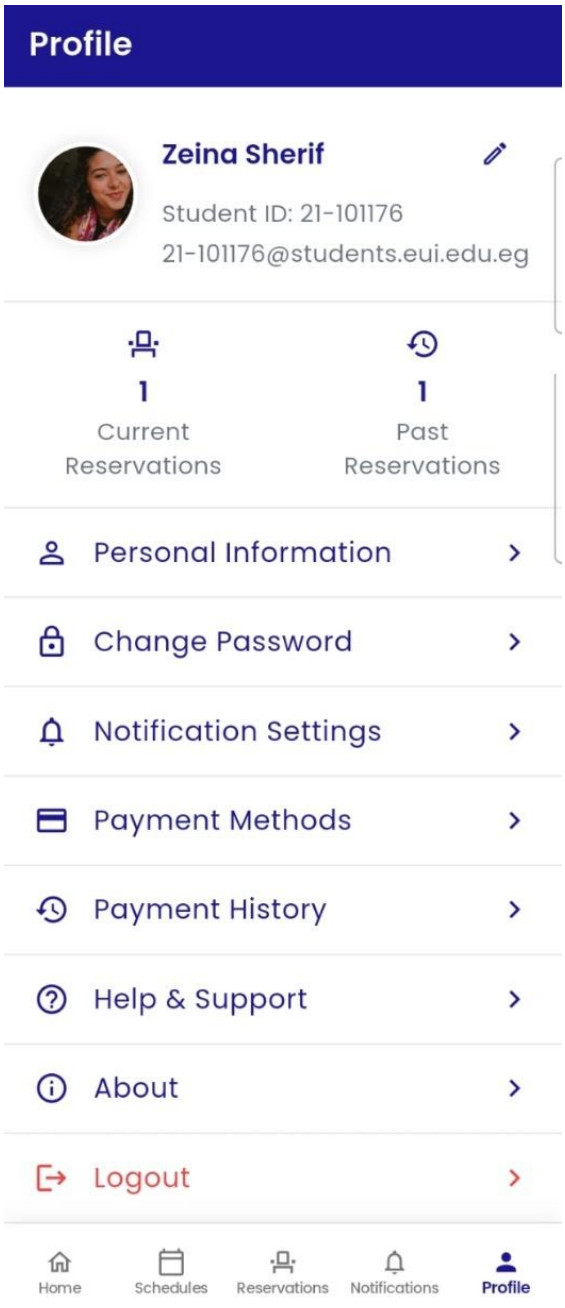
Description: The History tab provides a log of all past reservations. Users can review the details of their previous trips, including those that were completed or cancelled, offering a complete record of their travel activity.

4. Notifications Screen:



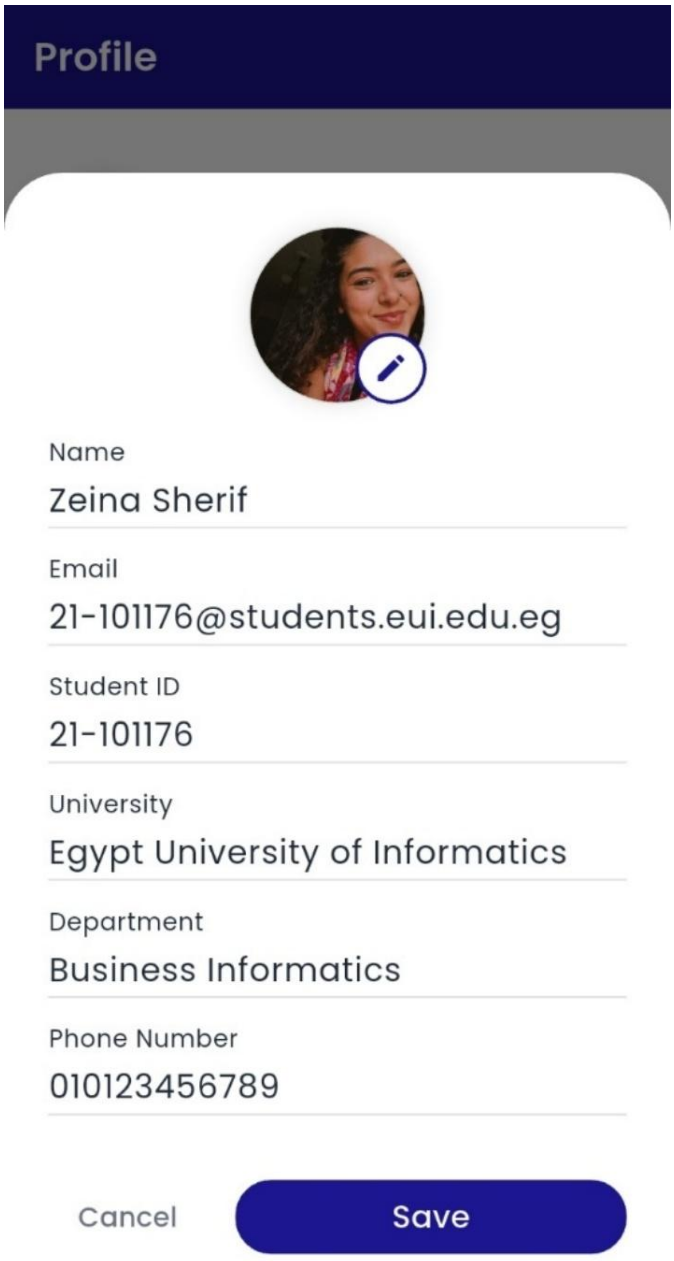
Description: The Notifications screen acts as a central hub for all app-related alerts. Users can manage their notification preferences for different event types and view a feed of important updates, such as reservation confirmations and schedule changes.

5. Profile Screen:



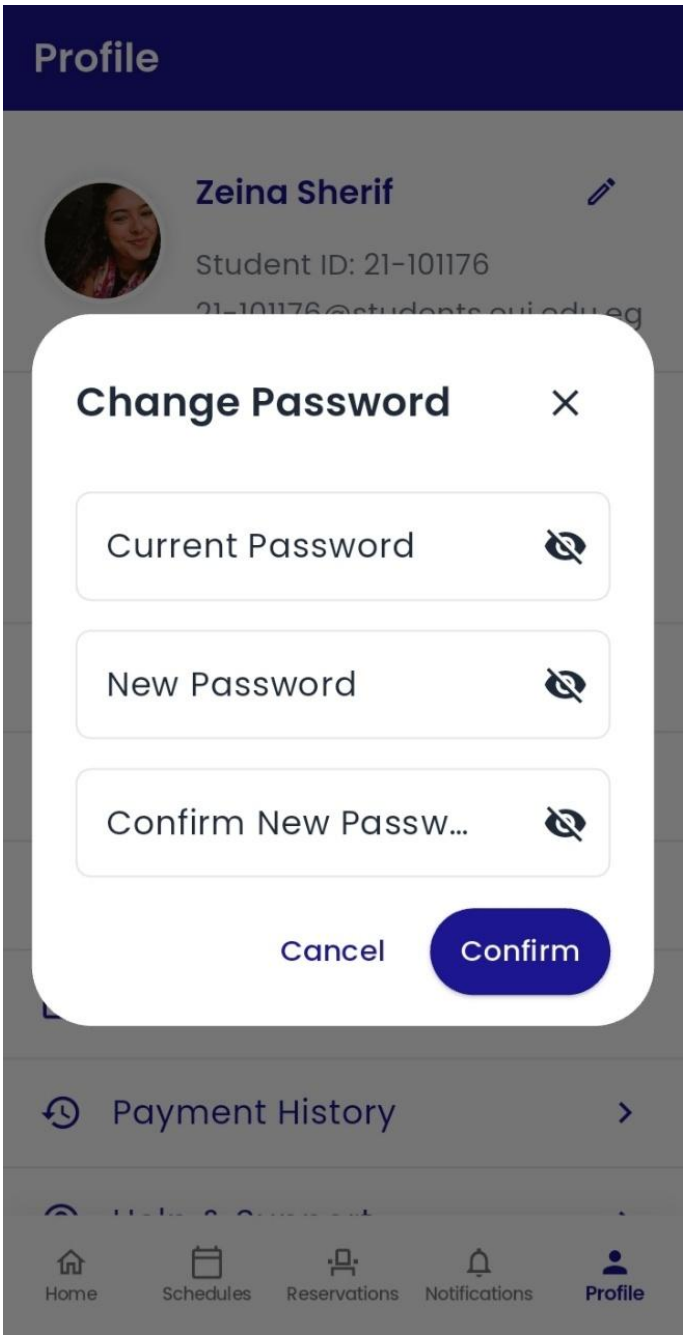
Description: The Profile screen serves as the main dashboard for user account management. It provides an overview of the user's reservation history and offers access to various settings, including personal information, password changes, and payment methods.

a. Edit Profile & Personal Information Page:



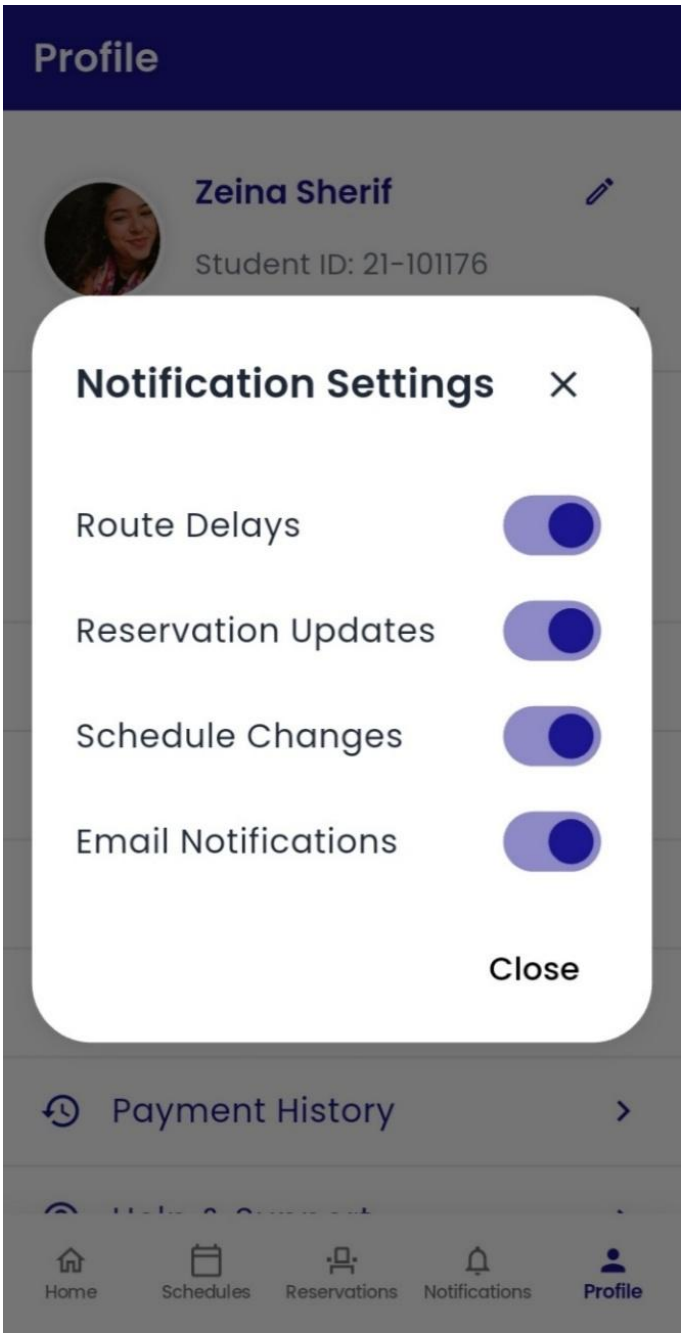
Description: This screen allows users to view and update their personal information. All key details, such as name, email, student ID, and phone number, can be modified here to ensure the user's profile is always accurate.

b. Change Password Page:



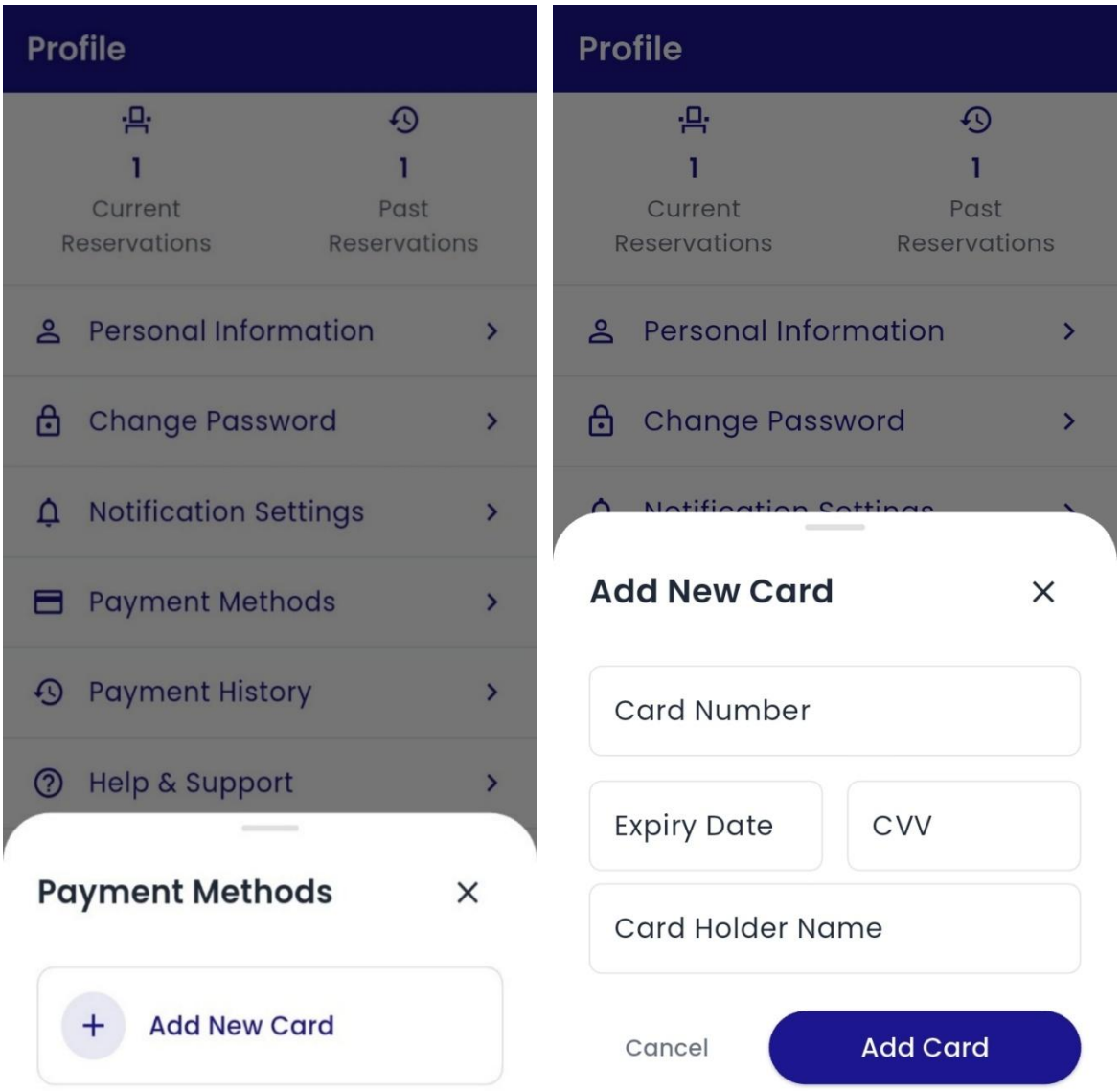
Description: This screen provides a secure interface for users to update their account password. The pop-up modal requires the user to verify their current password before setting and confirming a new one, ensuring account integrity.

c. Notification Settings Page:



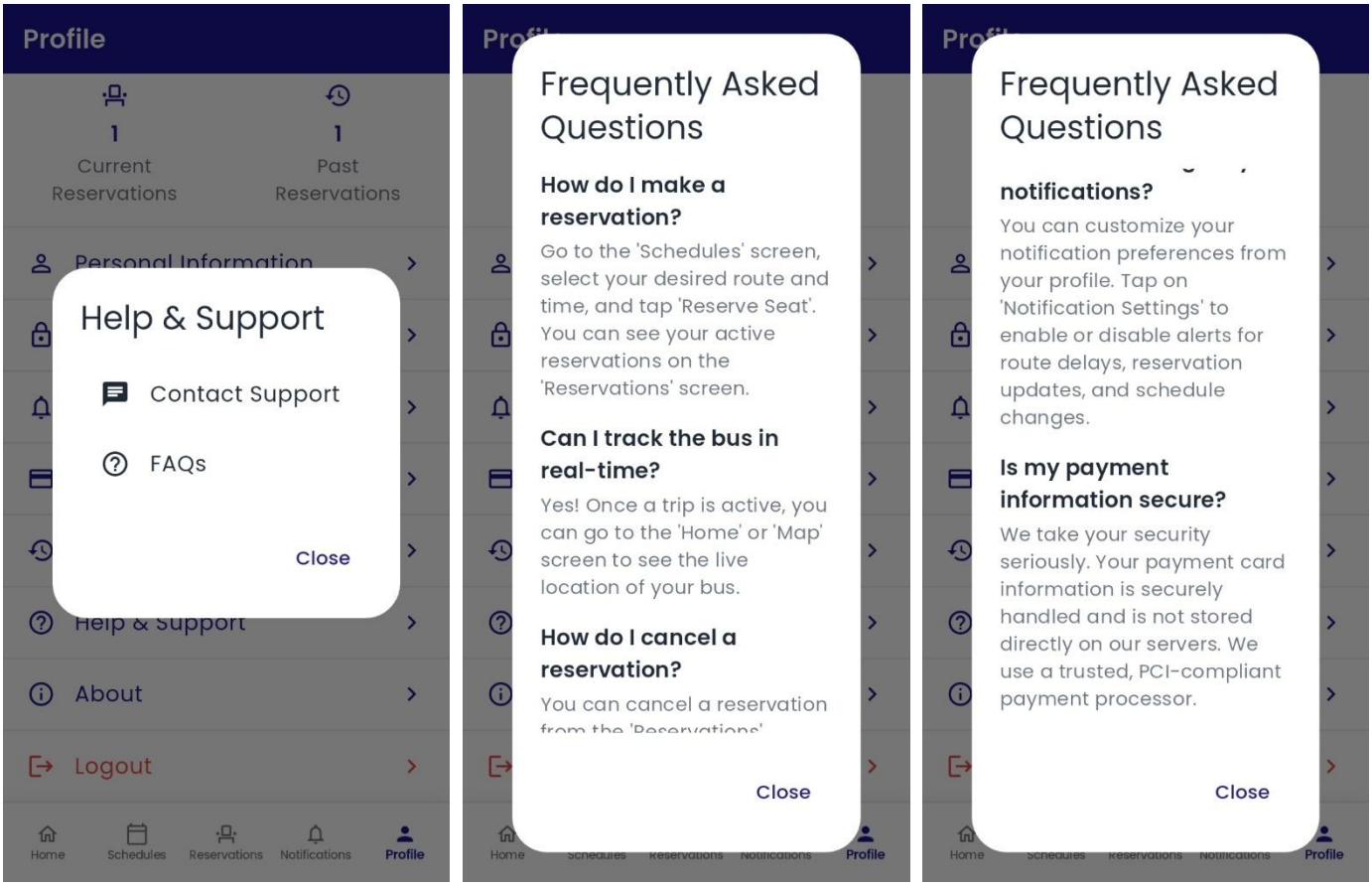
Description: The Notification Settings modal allows users to personalize their alert preferences. They can individually toggle notifications for route delays, reservation updates, schedule changes, and general email communications to stay informed.

d. Payment Methods Page:



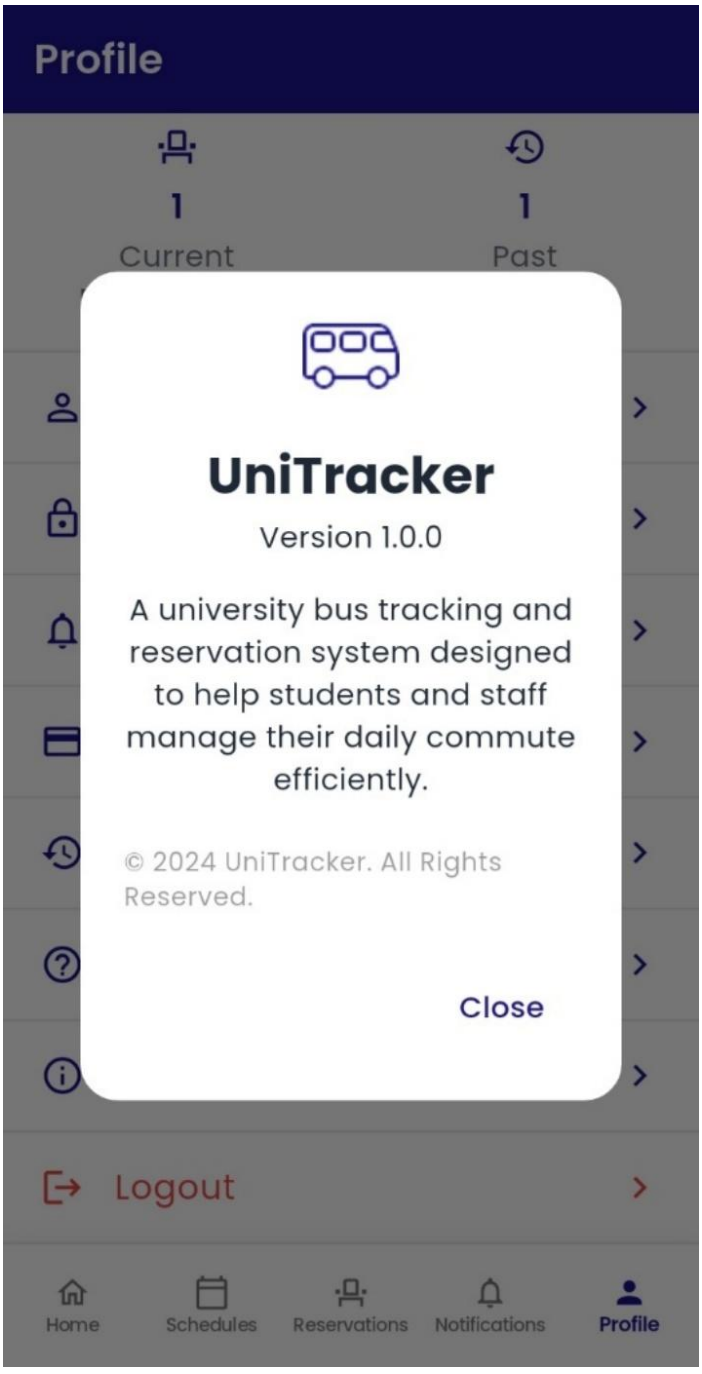
Description: This screen enables users to manage their payment options. It features a function to add a new credit or debit card via a secure pop-up form, streamlining the process for future reservation payments.

e. Help & Support Page:



Description: This page offers user assistance, featuring a list of Frequently Asked Questions (FAQs) for common issues and a "Contact Support" option for direct inquiries.

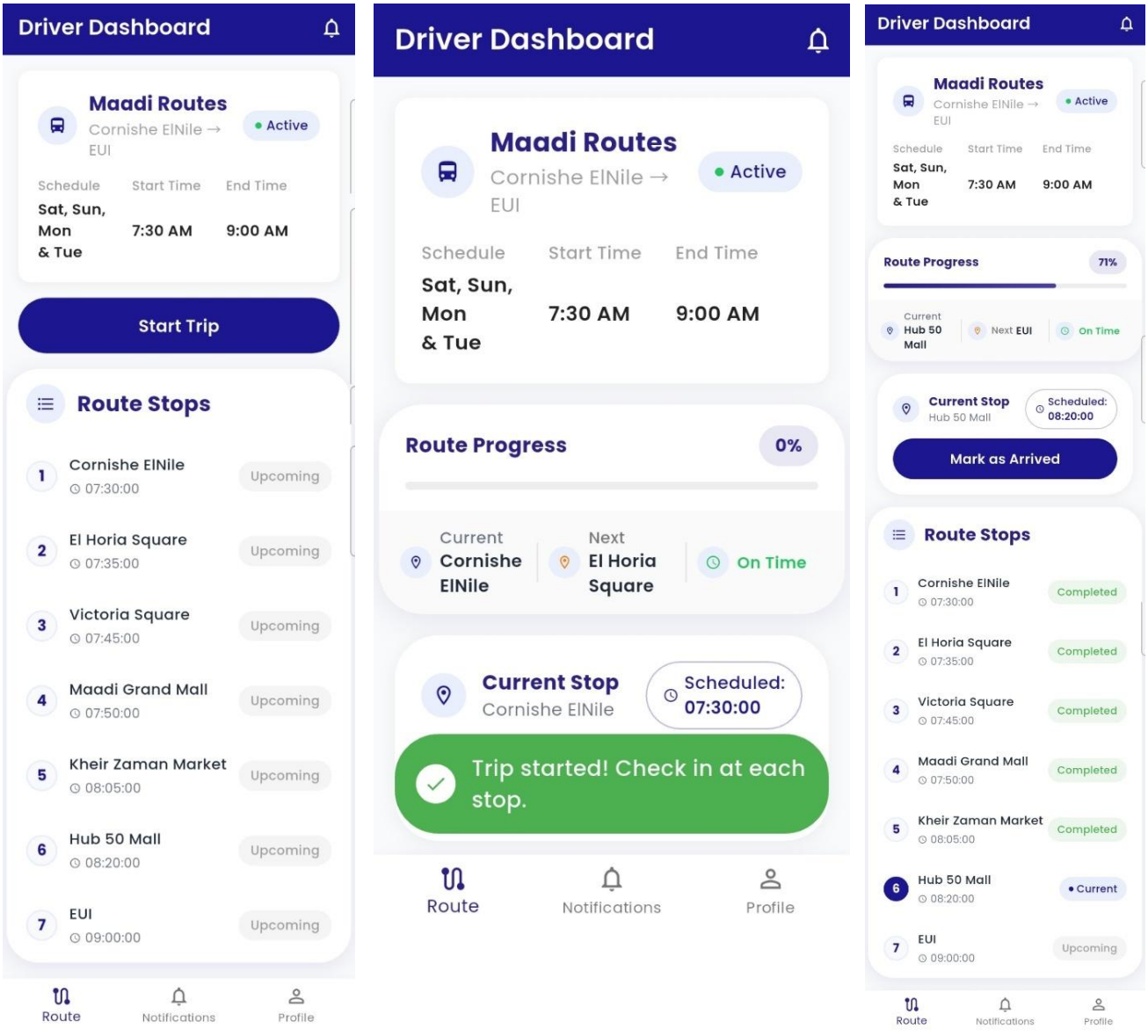
f. About App Page:



Description: The About page displays key information regarding the application, including the "UniTracker" name, version number, and a brief summary of its purpose.

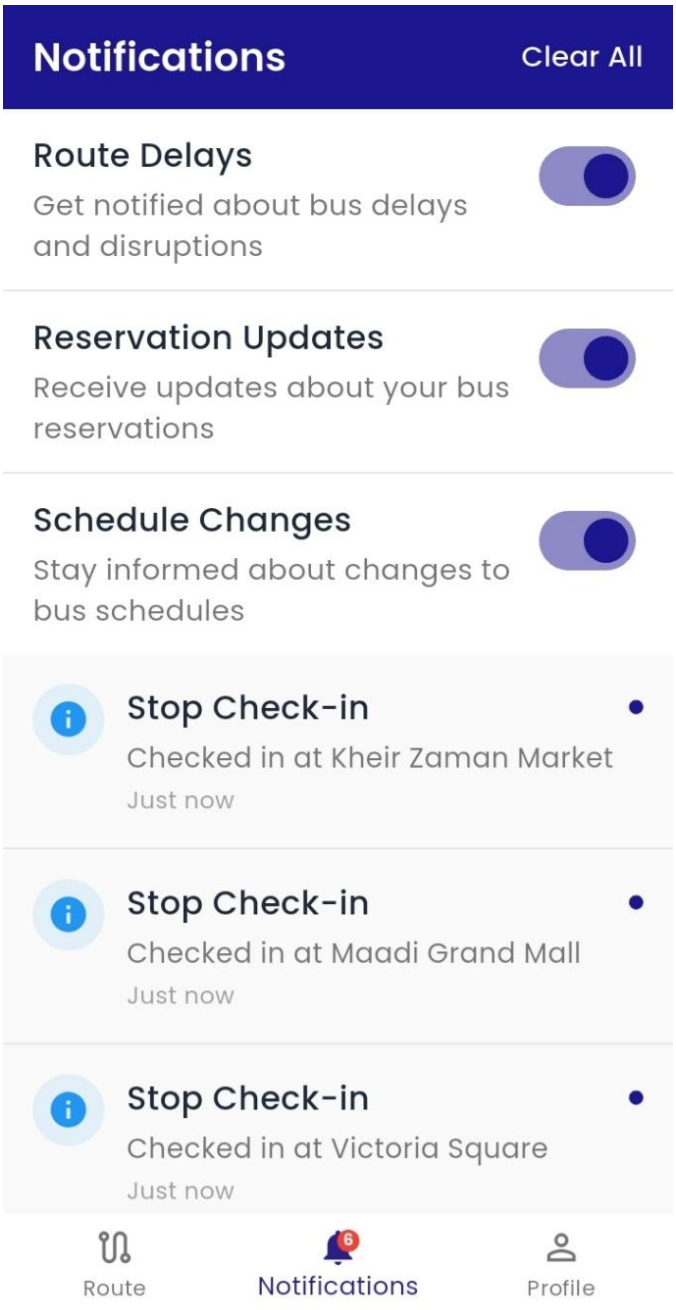
6.2 Driver Mobile App

1. Routes Screen:



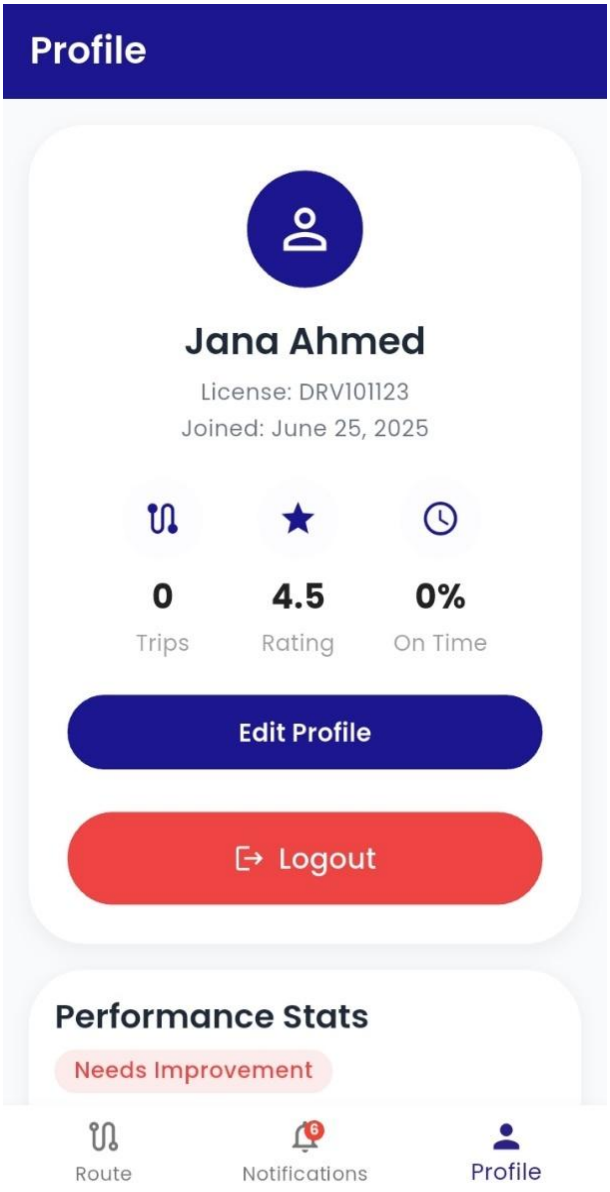
Description: The driver's dynamic Home Screen displays their assigned route and daily schedule. It guides them through their trip with functions to start the journey, view route progress, and manage student check-ins at each stop.

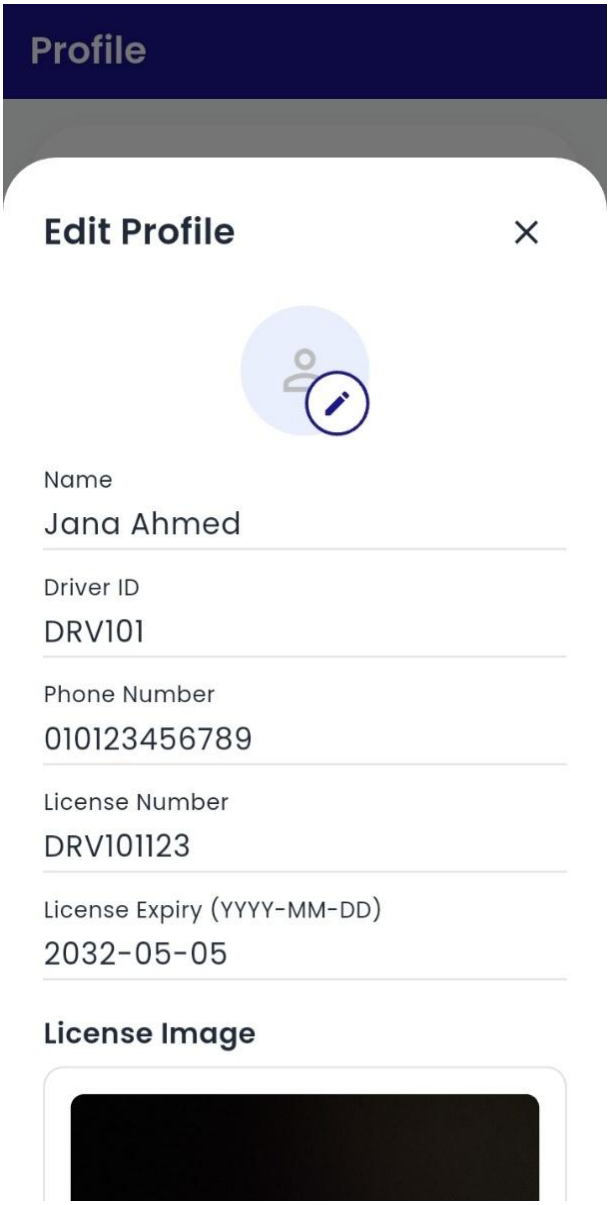
2. Notifications Screen:



Description: This screen provides drivers with essential real-time alerts, including updates on route delays, schedule changes, and confirmations of student check-ins at their stops.

3. Profile Screen:

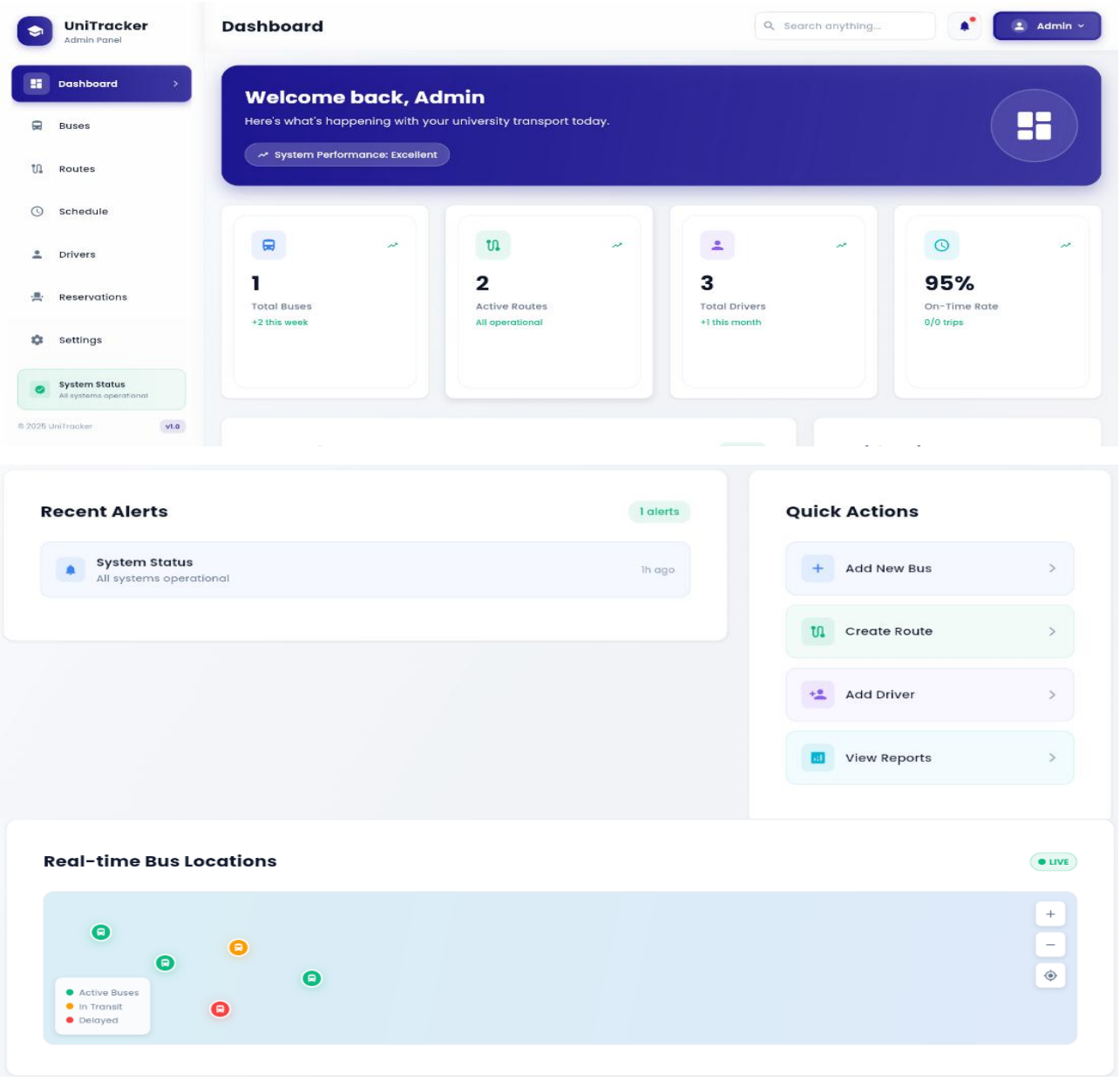




Description: The Profile screen contains the driver's professional information, such as license number and performance stats. It also includes an option to edit their personal details.

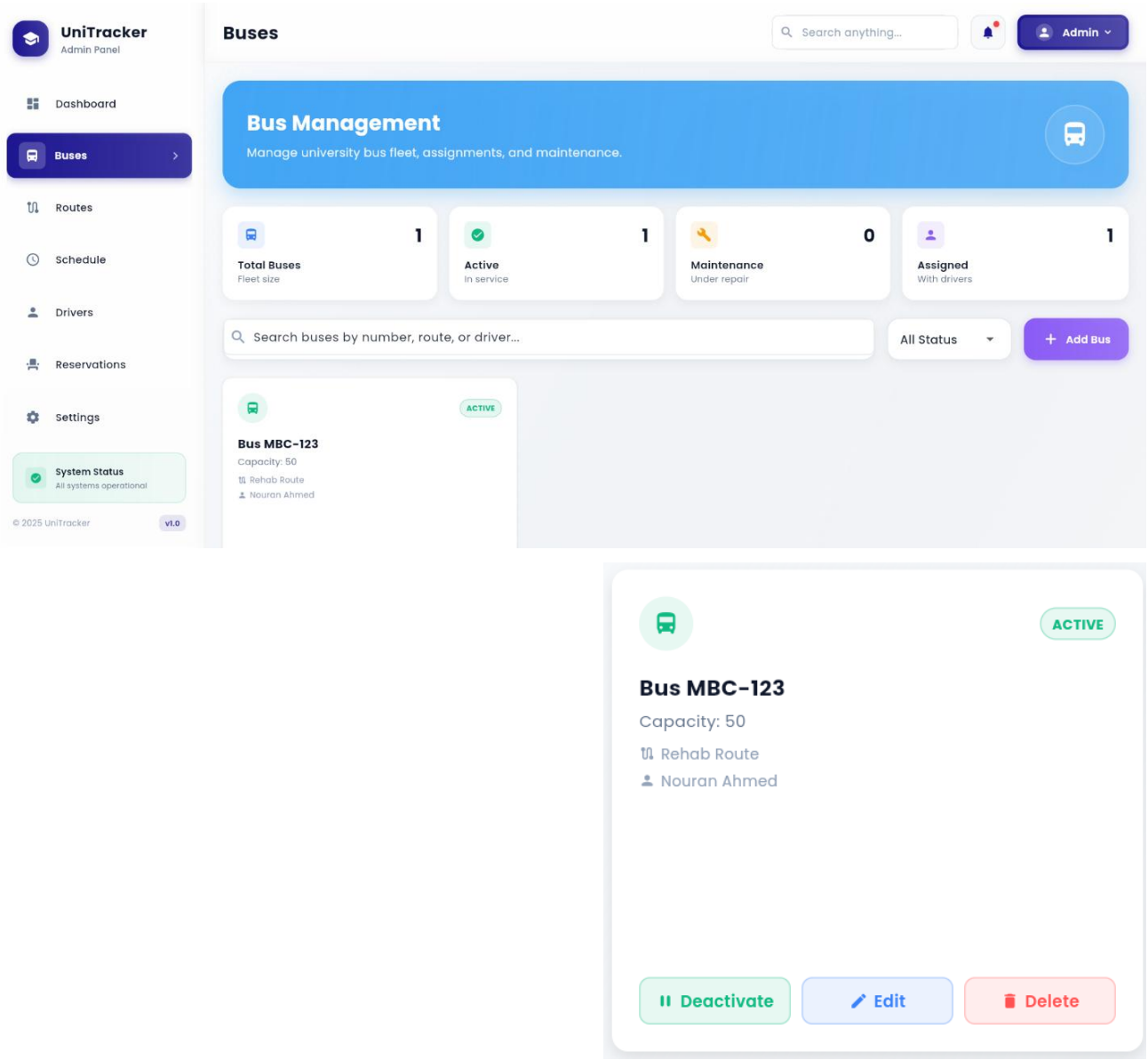
6.3 Admin Web App

1. Dashboard Screen:



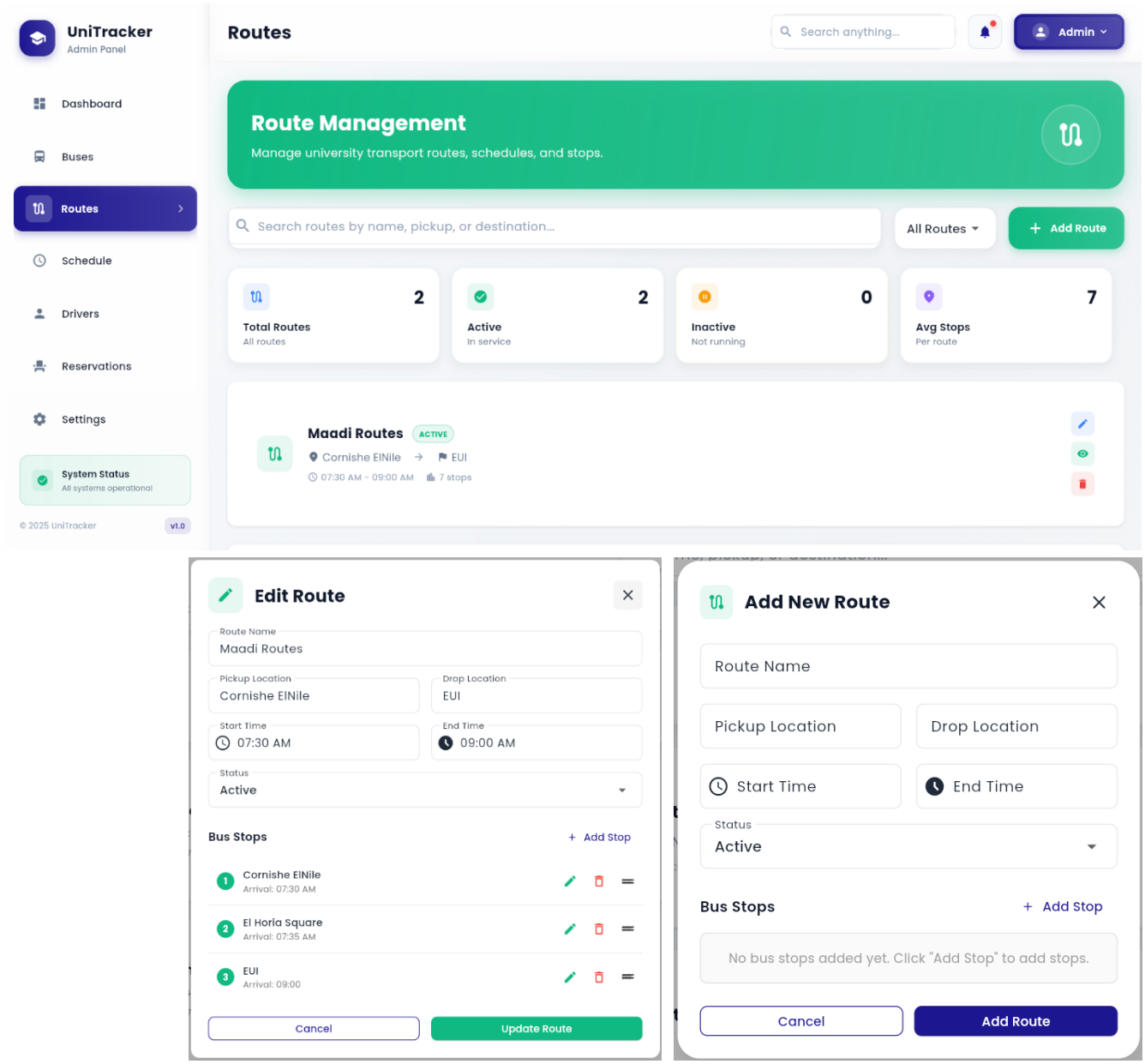
Description: The main Dashboard provides administrators with a comprehensive, at-a-glance overview of system activity, featuring key metrics, recent trips, and a map of active buses.

2. Buses Screen:



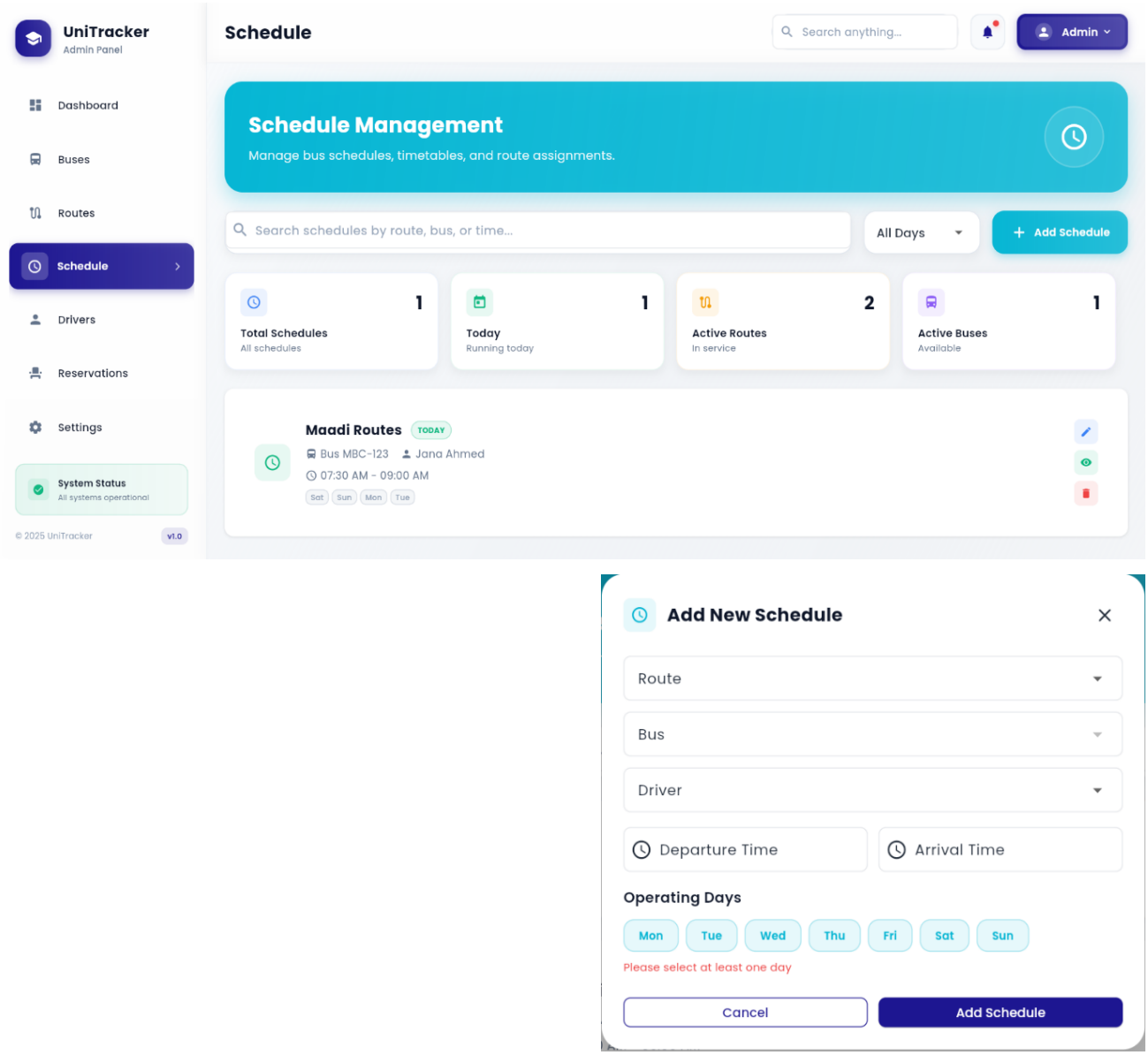
Description: The Bus Management screen allows administrators to oversee the vehicle fleet. It displays a list of all registered buses and provides tools to add, edit, or remove vehicles.

3. Routes Screen:



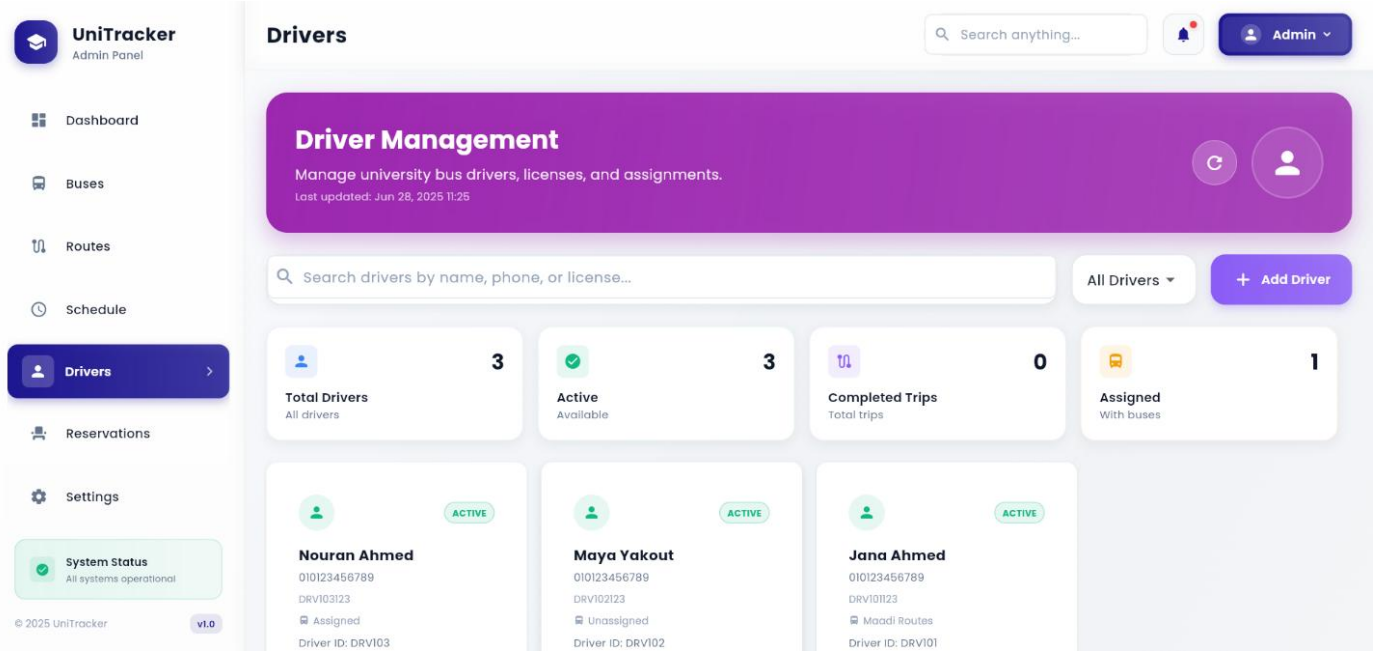
Description: This screen enables the creation and management of bus routes. Administrators can define stops, set paths, and edit details for all routes in the system.

4. Schedule Screen:



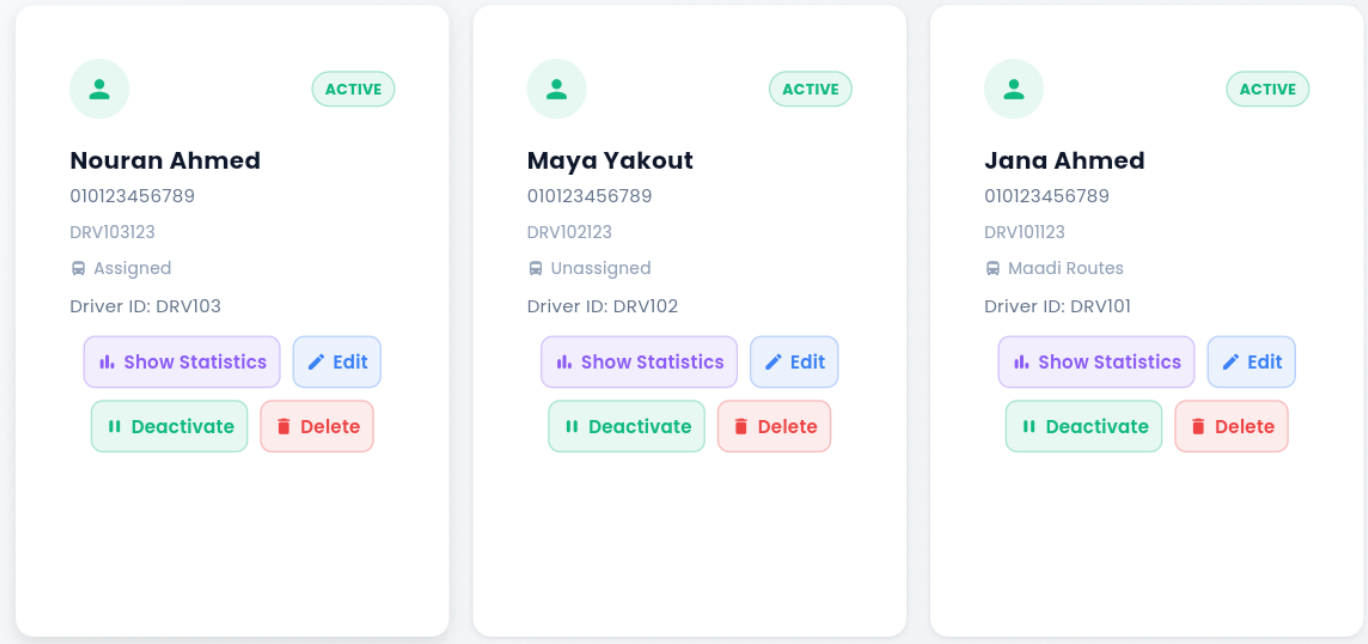
Description: The Schedule Management screen is used to create and assign timetables. Admins can link drivers and buses to specific routes and departure/arrival times.

5. Drivers Screen:



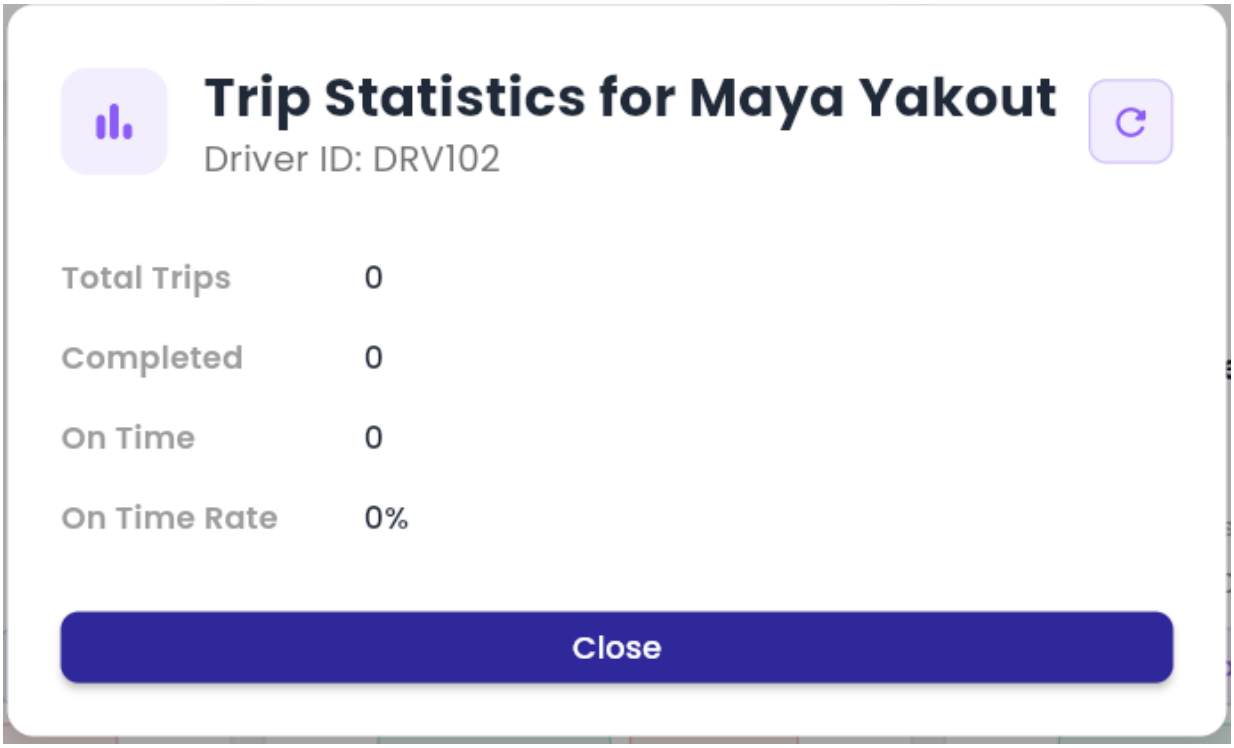
Description: This screen provides a complete overview of all driver accounts. Administrators can view driver details, manage their information, and monitor their status.

a. **Drivers Cards:**



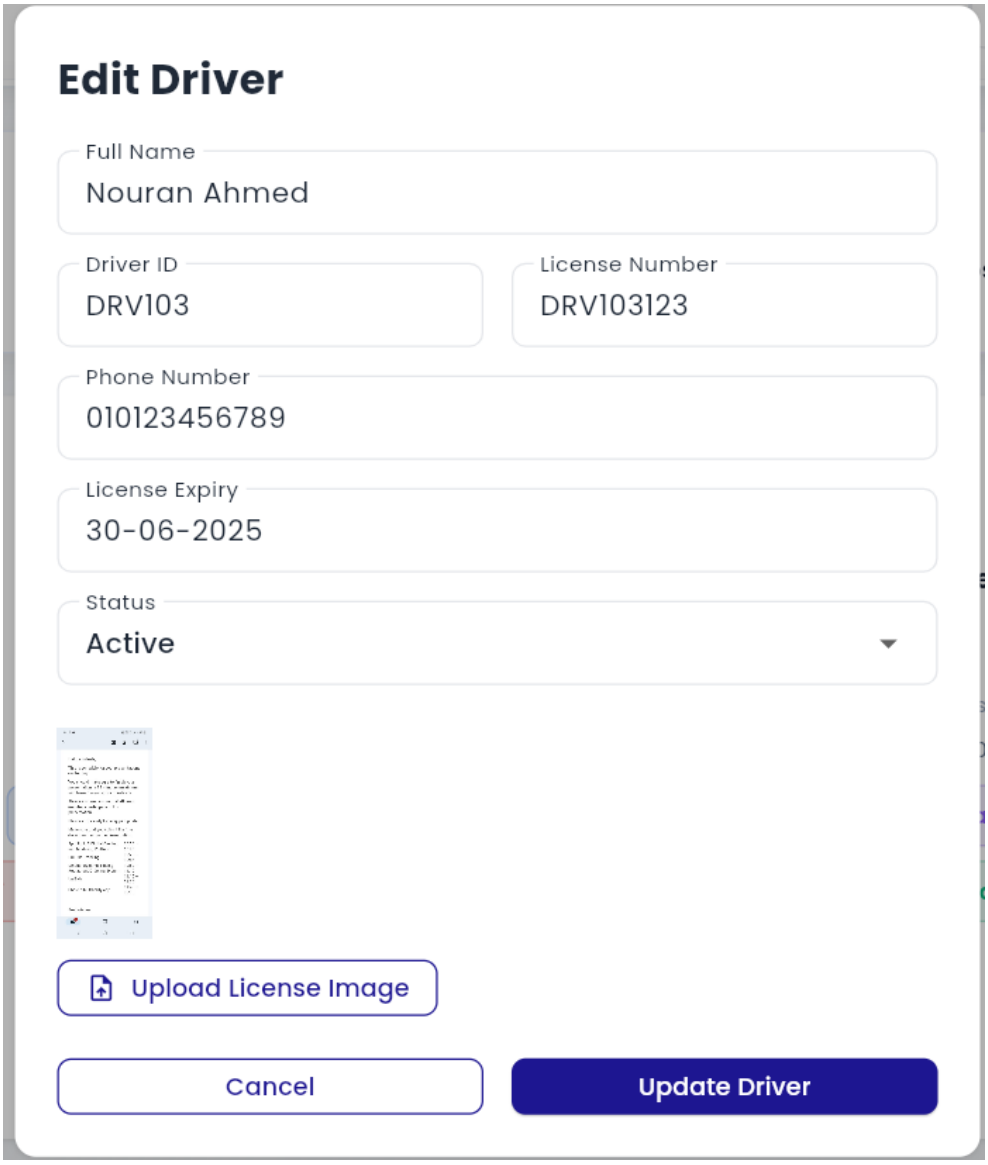
Description: This view provides a detailed log of driver activity, allowing administrators to monitor check-ins, trip status, and access individual driver performance statistics.

b. Show Statistics Page:



Description: This modal displays a performance summary for a selected driver, providing key statistics such as the total number of completed trips and their on-time performance rate.

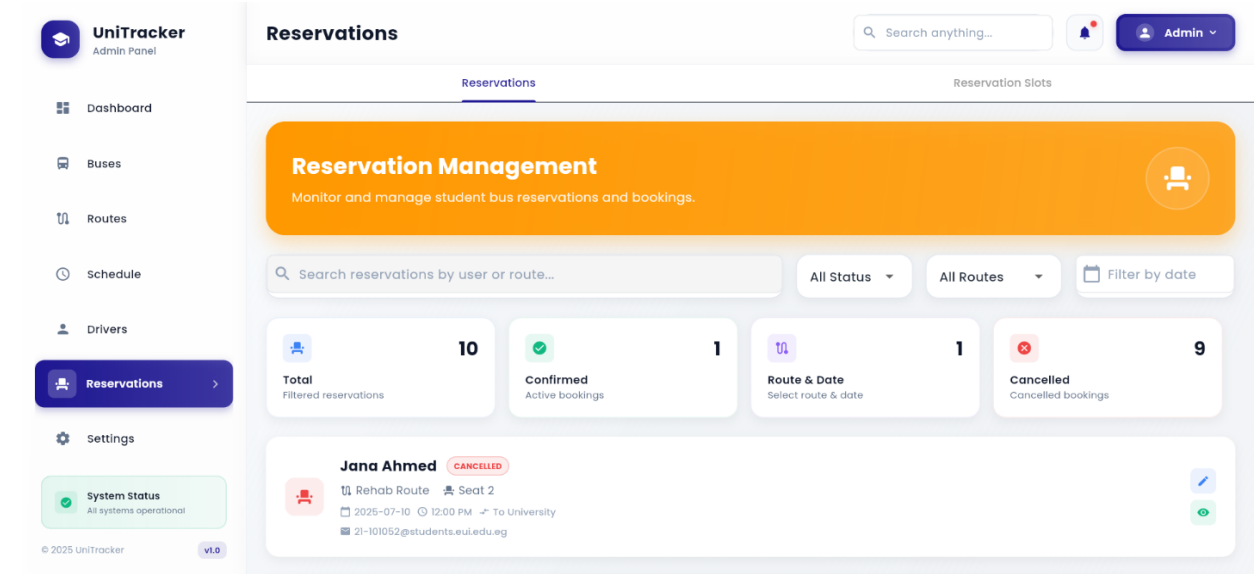
c. Edit Driver Page:

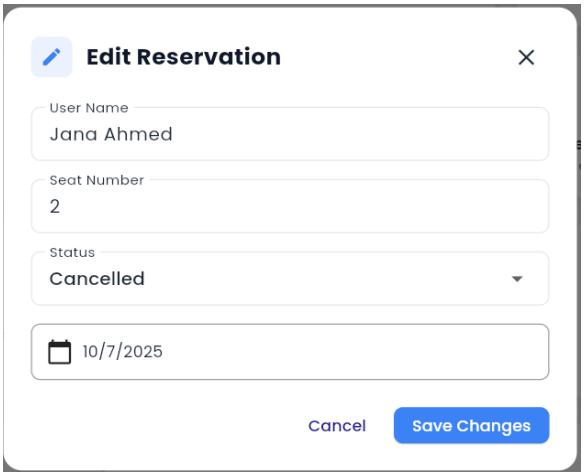


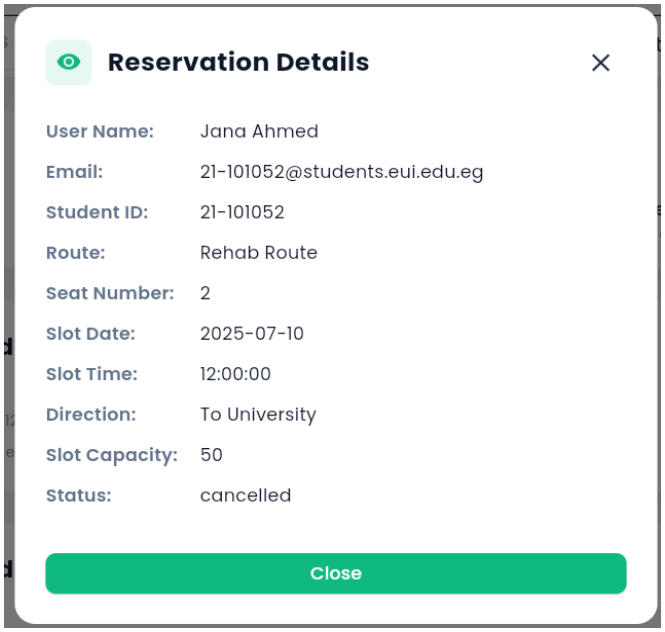
Description: The Edit Driver form enables administrators to update a driver's personal and professional information, including their status and license details, ensuring all records remain current and accurate.

6. Reservations Screen:

a. Reservations Page:

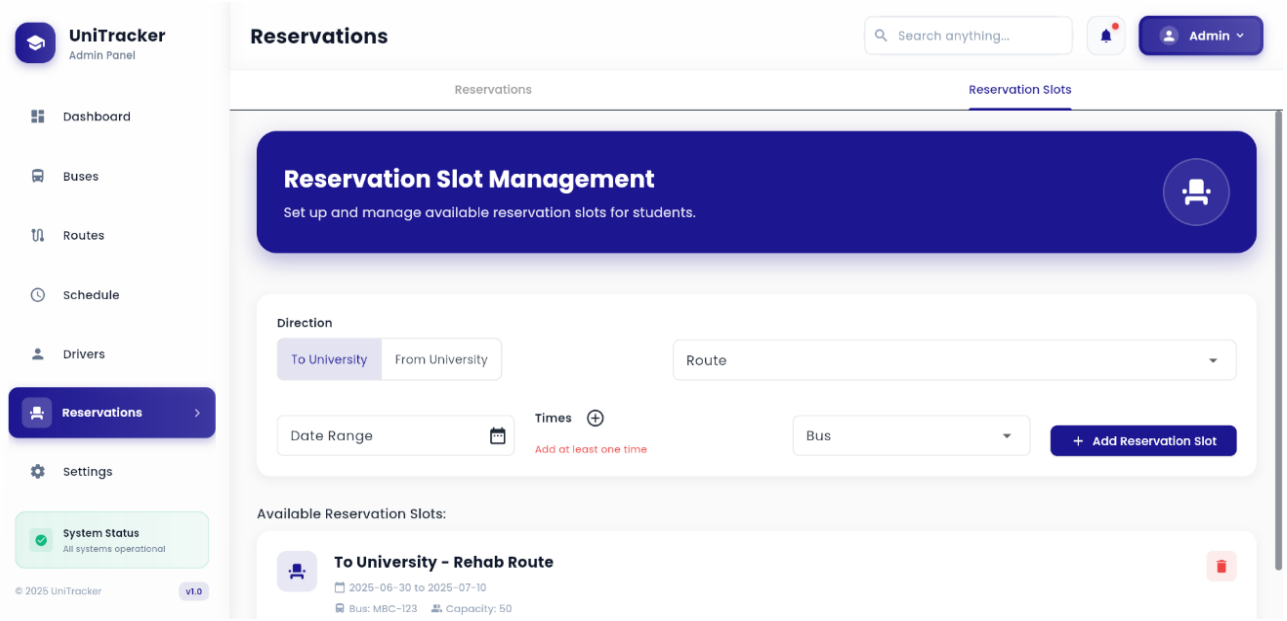


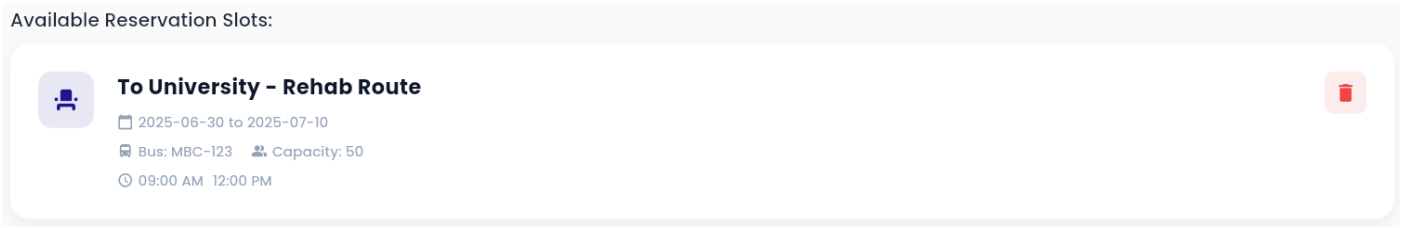




Description: The Reservations screen provides a comprehensive management interface for all bookings. Administrators can filter reservations and view or edit specific details, such as student information, route, and booking status, via an integrated modal.

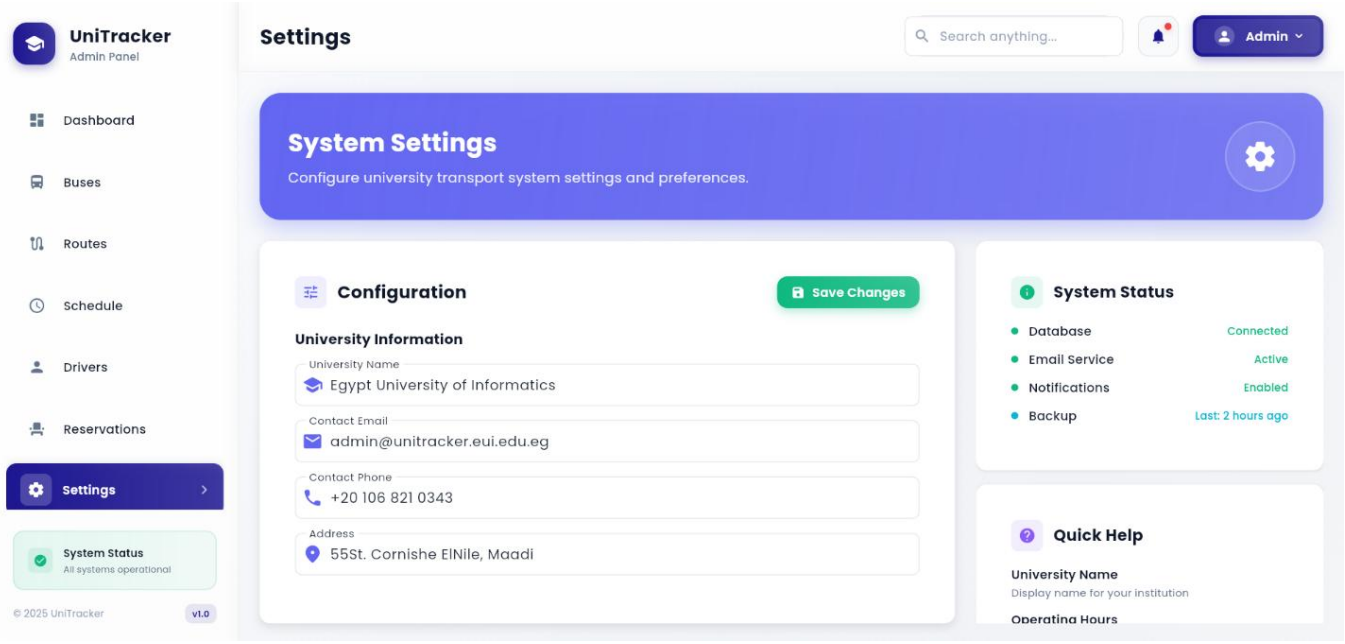
b. Reservation Slots Page:





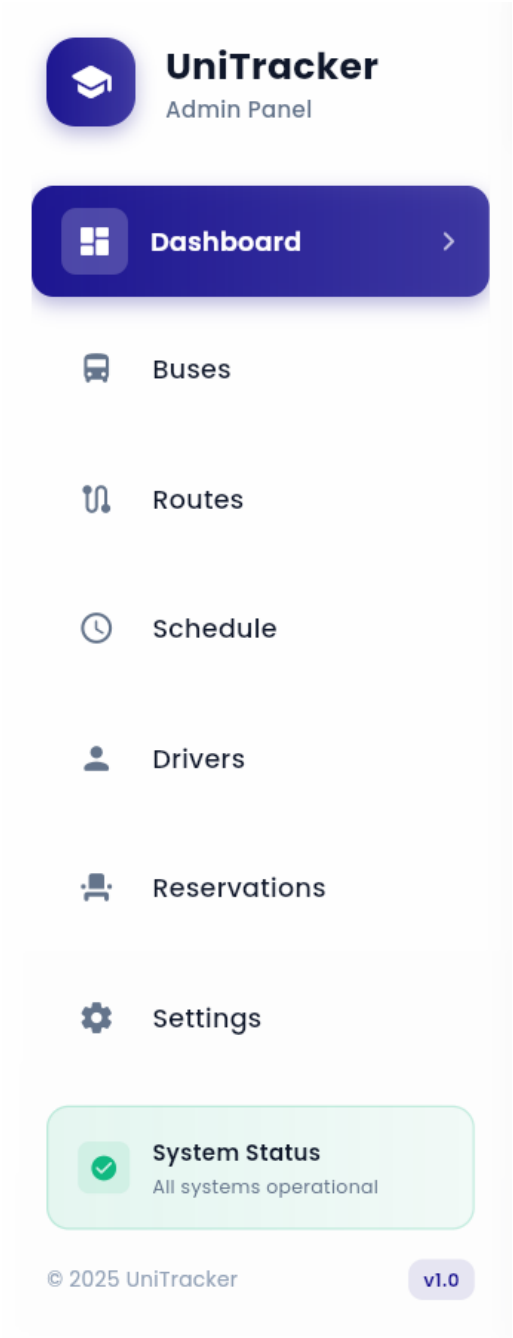
Description: This page allows administrators to create and manage system-wide notes or alerts related to reservations. This feature can be used to communicate important information about specific services to users or drivers.

7. Settings Page:



Description: The Settings page serves as the central control panel for the application. It allows administrators to manage global system settings, configure operational parameters, and update primary administrative information.

8. Side Navigation Bar:



Description: The Side Navigation Bar is the primary navigational component of the admin panel. It provides quick and structured access to all major management sections and displays the current system operational status.

Chapter 8

Conclusion and Recommendations and Future Improvements

8.1 Conclusion

To conclude, The University Bus Tracking and Reservation System (UBTRS) is a smart solution that helps students, staff, and university transport teams. It solves many problems like long waiting times, crowded buses, and confusing schedules by offering easy features such as:

- Real-time bus tracking
- Seat reservations
- Clear schedules and routes
- Notifications and updates

The system is easy to use, works on mobile phones, and supports both students and university staff. It helps the university manage buses better and gives users a more organized and stress-free way to travel to and from campus. The project is also cost-effective, secure, and ready to grow and be used in other universities.

8.2 Recommendations and Future Improvements

Based on the development and testing of UniTracker, several recommendations and future enhancements have been identified to improve the system's functionality, scalability, and user satisfaction. These insights will guide future iterations of the application as it expands to serve more universities and students.

1. **Expand to More Universities:**

   After successfully testing the system at one campus, we recommend offering UniTracker to other universities facing similar transportation challenges. This will allow for broader validation, increased impact, and continuous improvement based on diverse user feedback.

2. Enhance GPS Accuracy:

For more reliable tracking, future versions should integrate higher-precision GPS tools or external tracking hardware in buses. This will ensure accurate and smooth live location updates, even in areas with weak signals.

3. Enable Offline Access:

Adding offline support would allow users to view saved data such as schedules, routes, and previous reservations without an active internet connection. This feature is especially useful in low-connectivity zones.

4. Implement QR Code Check-In:

Integrating QR code scanning for boarding will streamline the check-in process, reduce manual errors, and improve security. Each student can scan their code when entering the bus for attendance verification.

5. Introduce In-App Payment Features:

Allowing users to make payments for special services, such as reserved seats or premium trips adds convenience. Future integration with mobile wallets or university payment systems could also support tuition or transport fee payments.

6. Strengthen Data Privacy and Security:

To protect sensitive user data, the system should be updated regularly with the latest security patches. All information should be encrypted and handled in compliance with local data protection regulations.

7. Provide Driver and Admin Training:

Training sessions should be conducted for drivers and university administrators to ensure proper use of the system. For students, visual guides and tutorial videos can be developed to simplify onboarding and usage.

8. Gather Continuous User Feedback:

Implementing short surveys or feedback forms within the app will help collect user insights and identify issues early. This continuous feedback loop is essential for refining the app based on real user needs.

9. Develop a Dedicated Driver App:

A separate mobile interface for drivers will help them manage their routes, receive live updates, confirm schedules, and report issues, all while staying focused on the road.

10. Support Smart Cancellations:

Future versions can introduce cancellation policies with optional online payments. For example, small fees could apply for last-minute cancellations to improve booking reliability and bus seat management.

11. Enable University Collaboration:

The system can be extended to support shared transport networks between universities, especially during inter-university events or joint programs. This encourages resource sharing and regional scalability.

12. Apply AI-Based Route Optimization:

As the system grows, AI can be used to analyze historical data and suggest optimized routes and schedules. This will help reduce delays, balance loads, and improve overall efficiency.