بسم الله الرحمن الرحيم

AN-NAJAH NATIONAL UNIVERSITY

FACULTY OF ENGINEERING AND INFORMATION TECHNOLOGY

Computer Engineering Department

Software Graduation Project

Driving School Management System

Students
Mohamad Salha
Ahmad Awad

Advisor
Dr. Oula Mardawi

June, 2026

Abstract

This graduation project presents a Driving School Management System designed to automate driving school operations. Traditional driving schools rely on manual paper-based processes for student management, scheduling, and payments, leading to inefficiencies and errors. Our system addresses these challenges through a web and mobile application that manages the complete student journey from enrollment to license certification.

The system was developed using Node.js with Express for the backend, MongoDB for data storage, Next.js for the web application, and React Native with Expo for the mobile app. It supports four user roles: Administrators, Teachers, Trainers, and Students. Key features include intelligent booking management with conflict prevention, dual payment processing through Stripe and cash methods, real-time push notifications via Firebase, in-app messaging, and comprehensive progress tracking.

The system was tested through various workflows including concurrent booking prevention, payment processing, and the complete student journey. Results show successful prevention of scheduling conflicts and maintained data integrity. This project demonstrates how modern web technologies can improve traditional service operations by automating administrative tasks and providing transparent digital interfaces for users.

Dedication

To our families for their support and encouragement throughout this project.

Acknowledgment

We gratefully acknowledge the support and guidance of **Dr. Oula Mardawi** throughout this project. We also thank our families and friends for their constant motivation and encouragement.

Disclaimer

This report is submitted in partial fulfillment of the requirements for the Bachelor degree in Computer Engineering at An-Najah National University. The opinions and conclusions expressed herein are those of the authors and do not necessarily reflect the views of the university or the supervisor.

Contents

Abstract **i**

Dedication **ii**

Acknowledgment **iii**

Disclaimer **iv**

1 Introduction **1**
1.1 Project Overview 1
1.2 Problem Statement 1
1.3 Project Objectives 1
1.4 System Landing Pages 2

2 Literature Review and Related Work **5**
2.1 Overview 5
2.2 Existing Driving School Systems 5
2.2.1 Ahlia Driving School Website (Palestine) 5
2.2.2 Drive-Smart System (An-Najah University Project, 2025) 6
2.3 Comparison Analysis 7
2.4 Research Gap and Our Contribution 7
2.5 Why Our System is Needed 8

3 Requirements Analysis **9**
3.1 Overview 9
3.2 Functional Requirements 9
3.2.1 Administrator Requirements 9
3.2.2 Student Requirements 9
3.2.3 Teacher Requirements 10
3.2.4 Trainer Requirements 10
3.3 Non-Functional Requirements 11
3.3.1 Performance Requirements 11
3.3.2 Security Requirements 11
3.3.3 Usability Requirements 11
3.3.4 Reliability Requirements 12
3.3.5 Scalability Requirements 12
3.4 System Constraints 12
3.5 Use Cases 13
3.5.1 Use Case 1: Student Registration and Enrollment 13
3.5.2 Use Case 2: Booking Practical Session 13
3.5.3 Use Case 3: Marking Attendance 13
3.5.4 Use Case 4: Scheduling and Grading Exam 14

4 Development Methodology **15**
4.1 Development Approach . . . 15
4.2 Standards and Specifications . . . 15
4.2.1 Software Engineering Standards . . . 15
4.2.2 Web Standards . . . 16
4.2.3 Mobile Development Standards . . . 16
4.2.4 Database Design Standards . . . 17
4.2.5 Security Standards . . . 17
4.3 Tools and Technologies . . . 17
4.3.1 Development Tools . . . 17
4.3.2 Development Frameworks and Libraries . . . 17
4.4 Development Standards . . . 18

5 System Architecture **19**
5.1 Technology Stack . . . 19
5.2 Authentication & Authorization . . . 19
5.3 Database Schema . . . 24
5.4 System Architecture and Data Flow . . . 25
5.5 Security and Data Protection . . . 26
5.6 External Services Integration . . . 27

6 Administrator Module **29**
6.1 Admin Dashboard . . . 29
6.2 Student Management . . . 31
6.3 Staff Management . . . 35
6.3.1 Teachers Management . . . 35
6.3.2 Trainers Management . . . 38
6.4 Exam Management . . . 40
6.4.1 Exam Scheduling . . . 40
6.4.2 Exam Attempts and Grading . . . 45
6.5 Vehicle Management . . . 51
6.6 Payment and Revenue Management . . . 54
6.7 Reports and Analytics . . . 58

7 Student Module **59**
7.1 Student Enrollment Journey . . . 59
7.1.1 License Selection . . . 59
7.1.2 Instructor Selection . . . 60
7.2 Student Dashboard . . . 63
7.3 My Course and Progress . . . 66
7.4 Schedule and Booking System . . . 73
7.4.1 Theoretical Schedule . . . 73
7.4.2 Practical Session Booking . . . 73
7.5 Payment System . . . 81
7.6 Examination System . . . 87

7.7 Notifications and Profile 92

8 Teacher Module **96**
8.1 Teacher Dashboard 96
8.2 My Students 99
8.3 Schedule Management 102
8.4 Attendance Management 104

9 Trainer Module **107**
9.1 Trainer Dashboard 107
9.2 My Students 109
9.3 Schedule Management 113
9.4 Attendance and Session Management 116

10 Key Features and Implementation **120**
10.1 Credit Balance and Refund System 120
10.2 Intelligent Session Booking 120
10.3 Progress Tracking 120
10.4 Payment Integration 120
10.5 Notification System 121
10.6 In-App Messaging System 121

11 Challenges and Solutions **125**
11.1 Technical Challenges 125
11.2 Design Challenges 125

12 Future Enhancements **126**
12.1 Planned Features 126
12.2 AI-Powered Features 126
12.3 Technical Improvements 127

13 Conclusion and Discussion **128**
13.1 Project Summary 128
13.2 Key Achievements and Technical Contributions 128
13.3 Learning Outcomes and Development Experience 129
13.4 Impact and Practical Value 130
13.5 Challenges Overcome and Lessons Learned 131
13.6 Future Directions and Recommendations 131
13.7 Final Remarks 132

A Technology Stack and Dependencies **134**
A.1 Core Technologies 134
A.2 External Services Integration 134
A.3 Development Tools 135

B Database Schema Overview **135**

C Glossary of Terms **136**

List of Figures

1 Landing Page - Hero Section 2
2 Landing Page - Features Overview 2
3 Landing Page - How It Works 3
4 Landing Page - Services Section 3
5 Landing Page - Testimonials 4
6 Landing Page - Contact Section 4
7 Web Application - Login Screen 19
8 Web Application - Sign Up Screen 20
9 Mobile Application - Login Screen 21
10 Mobile Application - Create Account (Step 1) 22
11 Mobile Application - Create Account (Step 2) 23
12 Database Entity-Relationship Diagram - 17 MongoDB Collections 24
13 High-Level System Architecture - Three-Tier Design 25
14 Admin Dashboard - Overview (Web) 29
15 Admin Dashboard - Statistics (Web) 29
16 Admin Dashboard - Mobile View (Part 1) 30
17 Admin Dashboard - Mobile View (Part 2) 31
18 Student Management - Overview (Web) 32
19 Student Management - Mobile View 33
20 Student Approval - Mobile Interface 34
21 Staff Management - Overview (Web) 35
22 Teachers List - Mobile View 36
23 Staff Management - Mobile Interface 37
24 Trainers Management - Web View 38
25 Trainers List - Mobile View 39
26 Schedule New Exam - Web Interface 40
27 Exam Search and Filter - Web View 40
28 Exams Management - Mobile View 41
29 Create Theoretical Exam (Step 1) - Mobile 42
30 Create Theoretical Exam (Step 2) - Mobile 43
31 Schedule Practical Exam - Mobile 44
32 Exam Attempts - Web View 45
33 Exam Reports - Web View 45
34 Exam Management - Mobile Interface 46
35 Exam Attempts - Mobile View 47
36 Manage Exam Attempts - Mobile 48
37 Grade Theoretical Exam - Mobile 49
38 Grade Practical Exam - Mobile 50
39 Vehicle Fleet Management - Web View 51
40 License Types Management - Web View 51
41 Add New License Type - Web Interface 52
42 Vehicle Management - Mobile View 53
43 Revenue Dashboard - Web View 54

44 Payment Statistics - Web View 54
45 Revenue Reports - Web View 55
46 Payment Management - Mobile View 56
47 Administrative Management - Mobile Dashboard 57
48 Printable Reports - Web Interface 58
49 License Selection - Mobile Interface 59
50 Choose Theoretical Instructor - Mobile 60
51 Choose Practical Trainer - Mobile 61
52 Trainer Selection Interface - Mobile 62
53 Student Dashboard - Overview (Web) 63
54 Student Dashboard - Progress View (Web) 63
55 Student Dashboard - Mobile View 64
56 Student Dashboard - Progress Overview (Mobile) 65
57 My Course - Web View 66
58 Progress Tracking - Web View 66
59 Theoretical Training Complete - Mobile 67
60 Theoretical Exam Passed - Mobile 68
61 Ready for Theoretical Exam - Mobile 69
62 Ready for Practical Training - Mobile 70
63 Ready for Practical Exam - Mobile 71
64 License Obtained - Congratulations Screen 72
65 Theoretical Class Schedule - Web View 73
66 Available Practical Sessions - Web View 73
67 Session Booking Interface - Web View 74
68 Booking Confirmation - Web View 74
69 Book Session - Mobile Interface 75
70 Session Booking Details - Mobile 76
71 Book with Trainer (Step 1) - Mobile 77
72 Book with Trainer (Step 2) - Mobile 78
73 Confirm Booking - Mobile 79
74 Booking Confirmation Screen - Mobile 80
75 Payment Interface - Web View 81
76 Payment Dashboard - Mobile 82
77 Payment History - Mobile 83
78 Transaction History - Mobile 84
79 Card Payment Processing - Mobile 85
80 Post-Booking Confirmation - Mobile 86
81 Exam Registration - Web View 87
82 Exam Details and Results - Web View 87
83 Exams Overview - Mobile 88
84 Theoretical Exam Registration - Mobile 89
85 Register for Theoretical Exam - Mobile 90
86 Practical Exam Registration - Mobile 91
87 Notifications Center - Mobile 92
88 Notification Details - Mobile 93

89 Approval Notification - Mobile 94
90 Student Profile - Mobile 95
91 Teacher Dashboard - Web View 96
92 Teacher Dashboard - Mobile View 97
93 Teacher Dashboard - Statistics (Mobile) 98
94 My Students List - Web View 99
95 My Students - Mobile View 100
96 Students in Progress - Mobile 101
97 Class Schedule - Web View 102
98 My Schedule - Mobile View 103
99 Mark Attendance - Web Interface 104
100 Mark Student Ready for Exam - Web 104
101 Attendance Tracking - Mobile 105
102 Mark Attendance Interface - Mobile 106
103 Trainer Dashboard - Overview (Web) 107
104 Trainer Dashboard - Statistics (Web) 107
105 Trainer Dashboard - Mobile View 108
106 All Students - Web View 109
107 Students in Training - Web View 109
108 Students Ready for Exam - Web View 110
109 My Students - Mobile View 111
110 Students in Progress - Mobile 112
111 My Schedule - Web View 113
112 Create New Schedule - Web Interface 113
113 Schedule Overview - Mobile 114
114 Create Schedule - Mobile Interface 115
115 Mark Attendance - Web View 116
116 Attendance Actions - Web Interface 116
117 Session Cancellation - Web View 117
118 Attendance Tracking - Mobile 118
119 Mark Attendance Interface - Mobile 119
120 Choose Conversation - Messaging Interface 122
121 Start New Conversation 123
122 Messaging Conversation View 124

List of Tables

1 Feature Comparison with Existing Systems 7

1 Introduction

1.1 Project Overview

The Driving School Management System is a full-stack web and mobile application designed to modernize and automate the complete operations of a driving school. The system manages the entire student lifecycle from initial enrollment, through theoretical and practical training, to final examination and license issuance.

1.2 Problem Statement

Traditional driving schools rely heavily on manual processes, paper-based record keeping, and face-to-face scheduling, leading to inefficiencies, scheduling conflicts, and limited transparency for students regarding their progress. This project addresses these challenges by providing a comprehensive digital solution.

1.3 Project Objectives

- Automate student enrollment and license selection process
- Digitize theoretical class attendance and progress tracking
- Implement intelligent practical session booking with vehicle assignment
- Integrate secure online payment processing
- Provide real-time notifications and updates
- Generate comprehensive reports and analytics for administrators
- Support both web and mobile platforms for accessibility

1.4 System Landing Pages

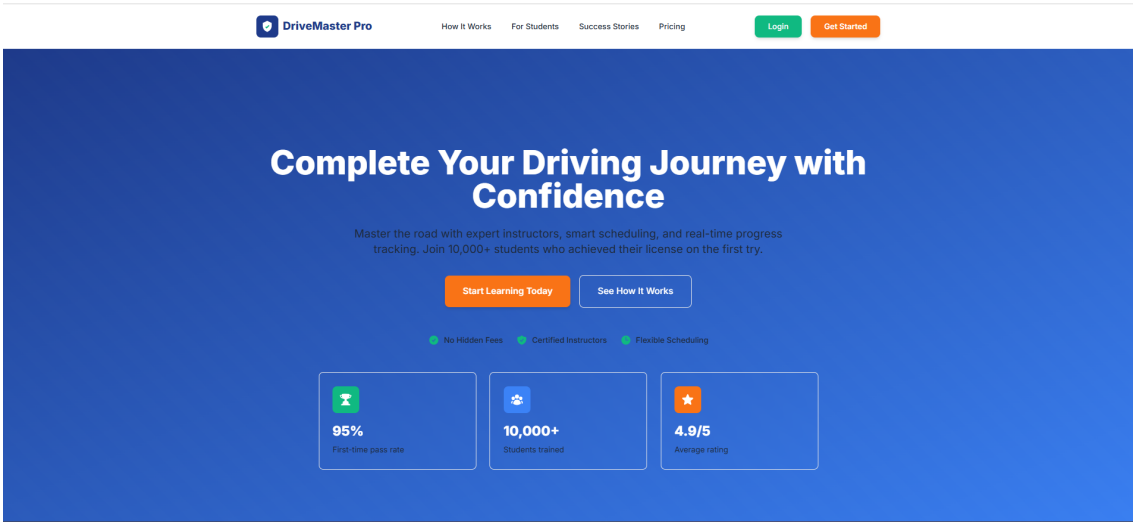

Figure 1: Landing Page - Hero Section

Figure 2: Landing Page - Features Overview

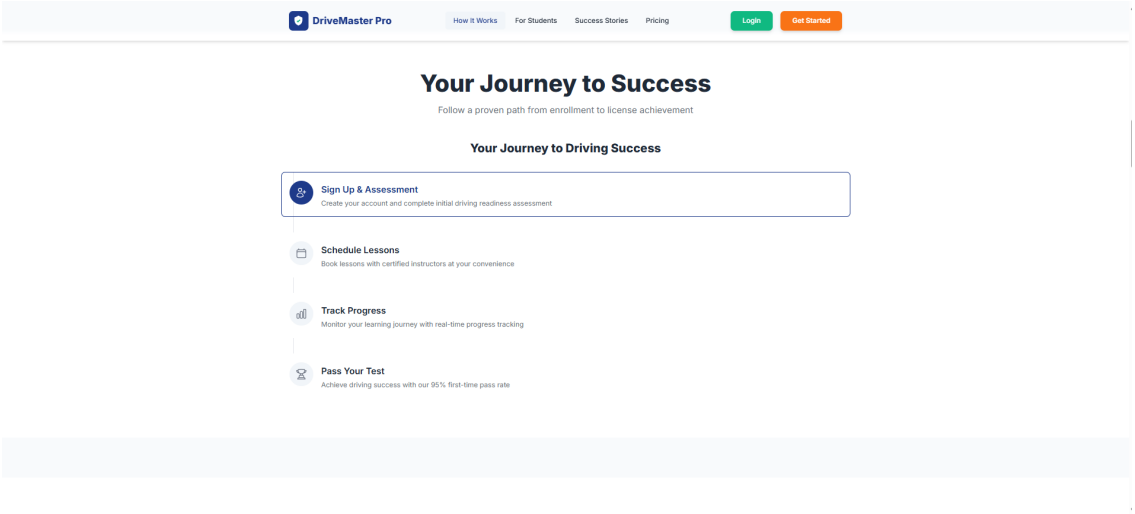

Figure 3: Landing Page - How It Works

Figure 4: Landing Page - Services Section

Figure 5: Landing Page - Testimonials

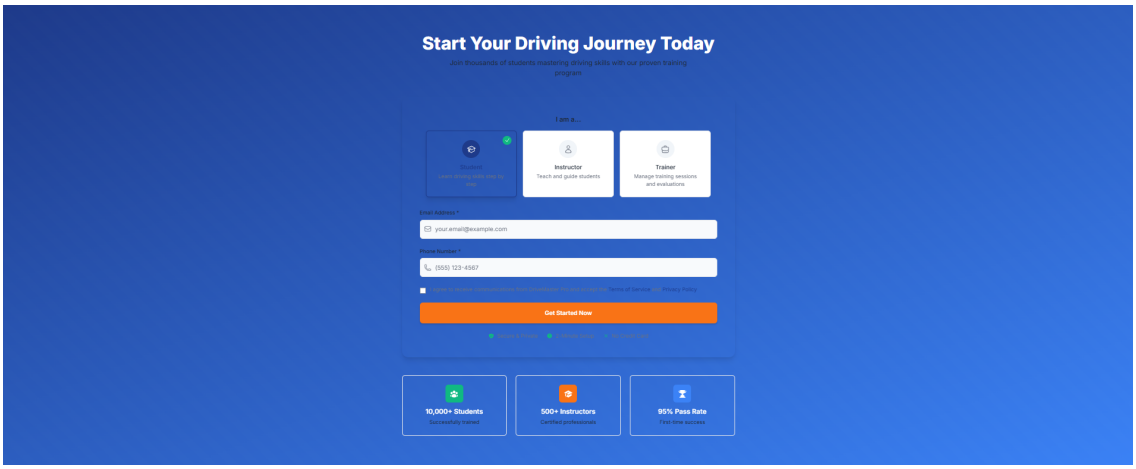

Figure 6: Landing Page - Contact Section

2 Literature Review and Related Work

2.1 Overview

Before developing our system, we conducted research on existing driving school management platforms to understand current solutions, identify gaps, and learn from their strengths and weaknesses. This research focused primarily on solutions available in Palestine and similar markets, as well as academic projects from other universities.

2.2 Existing Driving School Systems

2.2.1 Ahlia Driving School Website (Palestine)

Ahlia is one of the major driving schools operating in Nablus, Palestine. We analyzed their website (`https://www.ahliads.com/`) as a representative example of current digital presence among Palestinian driving schools.

What they offer:

- Static informational website showing services
- Contact information and location details
- Social media presence (Facebook page)
- Phone-based inquiry system

What they lack:

- No online student registration system
- No instructor profiles or credentials display
- No vehicle information or fleet management
- No detailed theoretical materials or resources
- No online booking or scheduling system
- No progress tracking for students
- No digital payment options
- Mobile-only presence through Facebook

Key insight: Palestinian driving schools currently rely almost entirely on walk-in registrations and phone calls for all operations, creating inefficiencies for both staff and students.

2.2.2 Drive-Smart System (An-Najah University Project, 2025)

Drive-Smart is a similar graduation project completed by students at An-Najah National University in 2025. We studied their report to understand their approach and differentiate our work.

Their features:

- Web and mobile applications
- Four user roles (Student, Trainer, Manager, Admin)
- Quiz maker for theoretical exams
- Student resources library
- AI chatbot assistant ("Drive Master Bot")
- Multi-school platform support (manages multiple schools)
- Uses MySQL database

Our differences:

- We focus on single-school comprehensive management vs their multi-school platform
- We implemented real payment integration (Stripe) while they didn't include payments
- We have Firebase push notifications while they didn't mention notification system
- We have credit/refund system for missed sessions
- We use MongoDB for flexibility vs their MySQL
- We have separate Teacher and Trainer roles vs combined approach

2.3 Comparison Analysis

Table 1: Feature Comparison with Existing Systems

Feature	**Ahlia Website**	**DriveSmart**	**Drive-Pro (Ours)**
Online Registration	No	Yes	Yes
Mobile Application	No	Yes	Yes (iOS + Android)
Payment Integration	No	No	Yes (Stripe + Cash)
Push Notifications	No	Not mentioned	Yes (Firebase FCM)
In-App Messaging	No	Yes	Yes
Progress Tracking	No	Yes	Yes (Detailed)
Booking System	No	Yes	Yes (Smart constraints)
Theoretical Quizzes	No	Yes	No
Resources Library	No	Yes	No
AI Chatbot	No	Yes	No
Credit/Refund System	No	No	Yes
Multi-School Support	N/A	Yes	No (Single school)
Role Separation	N/A	Combined	Separate Teacher/-Trainer

2.4 Research Gap and Our Contribution

Based on our analysis, we identified several gaps in existing solutions for the Palestinian market:

Gap 1: Lack of Integrated Payment Systems

- Current Palestinian systems rely on cash-only, manual recording
- Our solution: Dual payment system (online credit card via Stripe + traditional cash) with automatic receipt generation

Gap 2: No Mobile-First Approach

- Most local schools have basic websites or Facebook pages only
- Our solution: Native mobile app for iOS and Android with full functionality

Gap 3: Limited Communication Tools

- Students must call or visit school for any inquiry
- Our solution: In-app messaging + push notifications for real-time updates

Gap 4: No Transparent Progress Tracking

- Students don't know their completion status until asking instructor
- Our solution: Real-time dashboard showing lectures attended, hours completed, exam readiness

Gap 5: Manual Scheduling Conflicts

- Phone bookings often lead to double-bookings or conflicts
- Our solution: Smart booking system with automatic conflict prevention and weekly limits

2.5 Why Our System is Needed

The driving school industry in Palestine is growing rapidly with increasing numbers of young people seeking driver's licenses. However, most schools still operate with paper-based systems, leading to:

- Long waiting times for registration and scheduling
- Lack of transparency in progress and payments
- Scheduling errors and conflicts
- Difficulty tracking student completion
- Limited accessibility (students must visit during office hours)
- No digital records or audit trails

Our system addresses all these issues while remaining practical for implementation in the Palestinian context, where factors such as internet reliability, smartphone adoption, and digital payment familiarity must be considered.

3 Requirements Analysis

3.1 Overview

This section defines what our system must do (functional requirements) and how it must perform (non-functional requirements). We gathered these requirements by interviewing driving school administrators, students, and instructors to understand their daily needs and pain points.

3.2 Functional Requirements

3.2.1 Administrator Requirements

- **FR-A1:** System shall allow admin to view and approve/reject new student registrations
- **FR-A2:** System shall allow admin to assign theoretical teachers to students
- **FR-A3:** System shall allow admin to assign practical trainers to students
- **FR-A4:** System shall allow admin to create and schedule theoretical exams
- **FR-A5:** System shall allow admin to create and schedule practical exams
- **FR-A6:** System shall allow admin to grade student exams (pass/fail)
- **FR-A7:** System shall allow admin to add, edit, and remove staff members (teachers, trainers)
- **FR-A8:** System shall allow admin to manage vehicle fleet (add, edit, assign to trainers)
- **FR-A9:** System shall allow admin to view all payments and revenue statistics
- **FR-A10:** System shall allow admin to generate reports on student progress, revenue, and operations
- **FR-A11:** System shall allow admin to view all system notifications and messages

3.2.2 Student Requirements

- **FR-S1:** System shall allow students to register with email, password, and personal details
- **FR-S2:** System shall allow students to select a license type during enrollment
- **FR-S3:** System shall allow students to view their current progress (lectures attended, hours completed)
- **FR-S4:** System shall allow students to view available theoretical class schedules
- **FR-S5:** System shall allow students to view their assigned teacher and trainer profiles

- **FR-S6:** System shall allow students to book practical driving sessions from available time slots
- **FR-S7:** System shall prevent students from booking more than 3 sessions per week
- **FR-S8:** System shall prevent students from booking more than 5 hours per week
- **FR-S9:** System shall allow students to pay for sessions via credit card or mark as cash payment
- **FR-S10:** System shall allow students to view their payment history
- **FR-S11:** System shall allow students to register for theoretical exams when ready
- **FR-S12:** System shall allow students to register for practical exams after passing theoretical exam
- **FR-S13:** System shall allow students to view their credit balance from refunds
- **FR-S14:** System shall allow students to receive push notifications on mobile devices
- **FR-S15:** System shall allow students to send messages to their teacher and trainer
- **FR-S16:** System shall allow students to cancel booked sessions

3.2.3 Teacher Requirements

- **FR-T1:** System shall allow teachers to create weekly theoretical class schedules
- **FR-T2:** System shall allow teachers to view list of assigned students
- **FR-T3:** System shall allow teachers to mark student attendance for each class
- **FR-T4:** System shall allow teachers to mark topics as completed for students
- **FR-T5:** System shall allow teachers to mark students as ready for theoretical exam
- **FR-T6:** System shall allow teachers to view student progress dashboards
- **FR-T7:** System shall allow teachers to send messages to students
- **FR-T8:** System shall automatically notify teachers when new students are assigned

3.2.4 Trainer Requirements

- **FR-TR1:** System shall allow trainers to create weekly practical schedules with time slots
- **FR-TR2:** System shall allow trainers to publish schedules for student booking
- **FR-TR3:** System shall allow trainers to view list of assigned students

- **FR-TR4:** System shall allow trainers to mark session attendance after completion
- **FR-TR5:** System shall automatically update student hours when attendance is marked
- **FR-TR6:** System shall allow trainers to mark students as ready for practical exam
- **FR-TR7:** System shall prevent trainers from marking readiness if student has pending payments
- **FR-TR8:** System shall allow trainers to view their assigned vehicle information
- **FR-TR9:** System shall allow trainers to send messages to students
- **FR-TR10:** System shall show trainers which time slots are booked vs available

3.3 Non-Functional Requirements

3.3.1 Performance Requirements

- **NFR-1:** System shall respond to user actions within 3 seconds under normal load
- **NFR-2:** System shall support at least 100 concurrent users without performance degradation
- **NFR-3:** Mobile application shall load initial dashboard within 4 seconds
- **NFR-4:** Database queries shall complete within 500 milliseconds for simple operations

3.3.2 Security Requirements

- **NFR-5:** System shall hash all passwords using bcrypt algorithm
- **NFR-6:** System shall use JWT tokens for user authentication
- **NFR-7:** System shall implement role-based access control to protect routes
- **NFR-8:** System shall validate all user inputs to prevent injection attacks
- **NFR-9:** System shall use HTTPS for all communications in production
- **NFR-10:** System shall not store credit card information (handled by Stripe)

3.3.3 Usability Requirements

- **NFR-11:** System interface shall be intuitive and require minimal training
- **NFR-12:** System shall be accessible on mobile devices (iOS and Android)
- **NFR-13:** System shall provide clear error messages when operations fail
- **NFR-14:** System shall have consistent navigation across all pages

3.3.4 Reliability Requirements

- **NFR-16:** System shall have 99% uptime during business hours
- **NFR-17:** System shall backup database data daily
- **NFR-18:** System shall handle network failures gracefully
- **NFR-19:** System shall log all errors for debugging purposes

3.3.5 Scalability Requirements

- **NFR-20:** System shall support up to 1000 registered students
- **NFR-21:** System database shall handle growth without restructuring
- **NFR-22:** System shall allow addition of new license types without code changes

3.4 System Constraints

Technical Constraints:

- Backend must be developed using Node.js (department requirement)
- Must use a NoSQL database (we chose MongoDB)
- Frontend must use React framework (we chose Next.js)
- Payment system must be in TEST MODE only for this academic project

Business Constraints:

- System designed for single driving school (not multi-tenant)
- Students must be approved by admin before full access
- Students can only have one active license at a time
- Theoretical exam must be passed before practical training begins

Environmental Constraints:

- Must work in Palestinian market with varying internet speeds
- Must accommodate both smartphone users and desktop users
- Must support traditional cash payments alongside online payments
- Must be developed within 6-month timeframe by 2-person team

3.5 Use Cases

3.5.1 Use Case 1: Student Registration and Enrollment

Actor: Student
Goal: Register in the system and enroll in a license program
Steps:

1. Student visits website/app and clicks "Sign Up"
2. Student enters personal details (name, email, password, phone, date of birth)
3. System validates information and creates account with "pending" status
4. Admin reviews and approves student
5. Student logs in and selects a license type
6. System creates enrollment record and student can now access full features

3.5.2 Use Case 2: Booking Practical Session

Actor: Student
Goal: Book a practical driving session with trainer
Preconditions: Student has passed theoretical exam and has assigned trainer
Steps:

1. Student navigates to "Book Session" page
2. System shows trainer's published schedule for current week
3. Student selects desired date and consecutive time slots
4. System validates booking limits (max 3 sessions/week, 5 hours/week, 1 per day)
5. Student confirms booking
6. System creates booking and payment record
7. Student pays via credit card or marks as cash
8. System sends confirmation notification to student and trainer

3.5.3 Use Case 3: Marking Attendance

Actor: Trainer
Goal: Mark student attendance after completed practical session
Preconditions: Session was booked and session time has passed
Steps:

1. Trainer views "My Schedule" and sees booked sessions

2. Trainer clicks "Mark Attendance" on completed session
3. System confirms attendance marking
4. System updates time slot status to "completed"
5. System adds session hours to student's total completed hours
6. If payment was cash, system automatically marks payment as paid
7. System sends notification to student confirming attendance

3.5.4 Use Case 4: Scheduling and Grading Exam

Actor: Admin
Goal: Schedule exam and grade students
Steps:

1. Admin navigates to "Exam Management"
2. Admin creates new exam with date, location, time, and type (theoretical/practical)
3. System publishes exam and notifies eligible students
4. Students register for exam through their dashboard
5. After exam date, admin views registered students
6. Admin enters pass/fail grade for each student
7. System updates student status and sends result notifications
8. If practical exam passed, system can generate license certificate

4 Development Methodology

4.1 Development Approach

We followed an iterative development approach similar to Agile methodology, but adapted for our 2-person student team. We divided the project into phases and completed features incrementally, allowing us to test each component before moving forward.

4.2 Standards and Specifications

To ensure our system meets professional quality standards and follows industry best practices, we adhered to several engineering standards and specifications:

4.2.1 Software Engineering Standards

RESTful API Design (REST):

- Our backend API follows RESTful architectural constraints for web services
- Utilized standard HTTP methods (GET, POST, PUT, DELETE) with appropriate semantics
- Implemented resource-based URL naming conventions
- Applied stateless communication between client and server

JSON Web Token (JWT) - RFC 7519:

- Implemented token-based authentication following JWT standard
- Tokens include standardized claims (iss, sub, exp, iat)
- Used HMAC SHA-256 for token signing
- Followed security best practices for token storage and transmission

Payment Card Industry Data Security Standard (PCI DSS):

- Delegated card data handling to Stripe (PCI-compliant provider)
- Never stored sensitive card information in our database
- Used HTTPS for all payment-related communications
- Implemented webhook signature verification for payment notifications

4.2.2 Web Standards

HTTPS/TLS Protocol:

- All communications encrypted using TLS 1.2+
- Secure transmission of authentication credentials and sensitive data
- Protection against man-in-the-middle attacks

WCAG 2.1 (Web Content Accessibility Guidelines):

- Implemented proper semantic HTML structure
- Ensured sufficient color contrast ratios
- Provided keyboard navigation support
- Added descriptive labels and alt text for images

HTTP Status Codes (RFC 7231):

- 200 OK - Successful requests
- 201 Created - Successful resource creation
- 400 Bad Request - Invalid client input
- 401 Unauthorized - Missing or invalid authentication
- 403 Forbidden - Insufficient permissions
- 404 Not Found - Resource not found
- 500 Internal Server Error - Server-side errors

4.2.3 Mobile Development Standards

React Native Best Practices:

- Followed official React Native documentation guidelines
- Implemented Material Design principles for Android (React Native Paper)
- Adhered to iOS Human Interface Guidelines for iOS platform
- Used platform-specific code where necessary for native behavior

Firebase Cloud Messaging (FCM):

- Implemented push notifications following FCM specifications
- Handled notification permissions properly for iOS and Android
- Implemented notification data payloads according to FCM format

4.2.4 Database Design Standards

MongoDB Schema Design Best Practices:

- Followed MongoDB data modeling patterns
- Implemented appropriate indexes for query optimization
- Used data validation rules with Mongoose schemas
- Applied normalization where appropriate while leveraging document embedding

4.2.5 Security Standards

OWASP Top 10 Security Principles:

- Protected against SQL/NoSQL injection through parameterized queries
- Implemented proper authentication and session management
- Used bcrypt for password hashing (10+ salt rounds)
- Applied input validation and sanitization
- Implemented role-based access control (RBAC)
- Protected sensitive data with encryption

4.3 Tools and Technologies

4.3.1 Development Tools

- **Code Editor:** Visual Studio Code with extensions for React, TypeScript
- **Version Control:** Git and GitHub for code management
- **API Testing:** Postman for testing backend endpoints
- **Database Management:** MongoDB Compass for viewing and managing database

4.3.2 Development Frameworks and Libraries

- **Backend:** Node.js v20, Express.js v4, Mongoose v8
- **Frontend Web:** Next.js 15, React 18, TypeScript, Tailwind CSS
- **Frontend Mobile:** React Native, Expo SDK 54, React Native Paper
- **Authentication:** JWT (jsonwebtoken), bcryptjs
- **Payments:** Stripe API
- **Notifications:** Firebase Cloud Messaging (FCM)
- **Image Storage:** Cloudinary

4.4 Development Standards

We established coding standards to maintain consistency:

Backend Standards:

- Layered architecture: Controllers → Services → Repositories
- camelCase for variables and functions
- Async/await for asynchronous operations
- Error handling with try-catch blocks
- Input validation on all API endpoints

Frontend Standards:

- Component-based architecture
- TypeScript for type safety
- Functional components with hooks
- Tailwind CSS for styling (web)
- React Native Paper components (mobile)

API Standards:

- RESTful endpoint naming conventions
- Consistent response format: `{success, data, message}`
- JWT tokens in Authorization header
- HTTP status codes: 200 (success), 400 (validation error), 401 (unauthorized), 500 (server error)

5 System Architecture

5.1 Technology Stack

Backend:

- Node.js with Express.js framework
- MongoDB database with Mongoose ODM
- JWT-based authentication
- Stripe payment integration
- Cloudinary for media storage

Web Frontend:

- Next.js 15 with React 18
- TypeScript for type safety
- Tailwind CSS for styling
- Axios for API communication

Mobile Application:

- React Native with Expo SDK 54
- TypeScript
- React Native Paper (Material Design)
- Expo Router for navigation

5.2 Authentication & Authorization

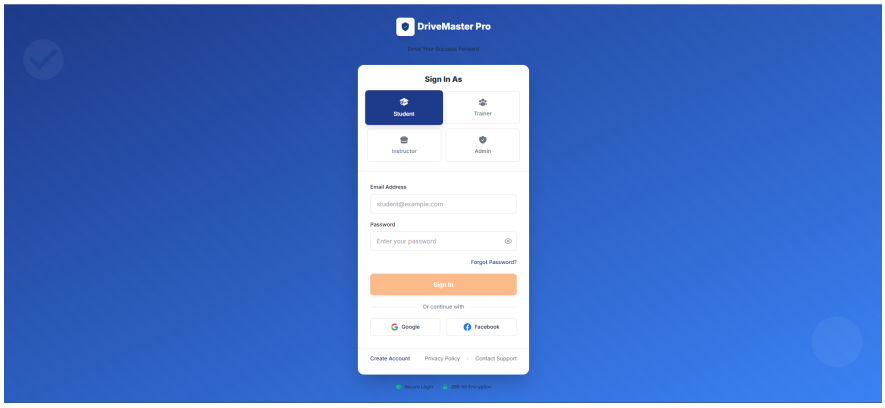

Figure 7: Web Application - Login Screen

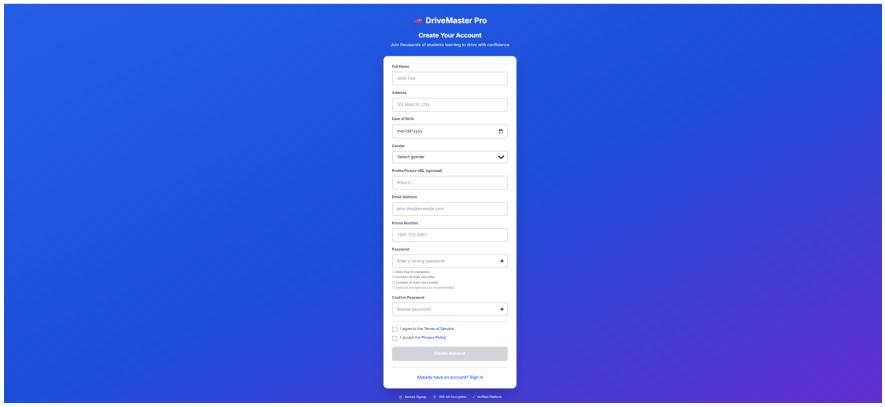

Figure 8: Web Application - Sign Up Screen

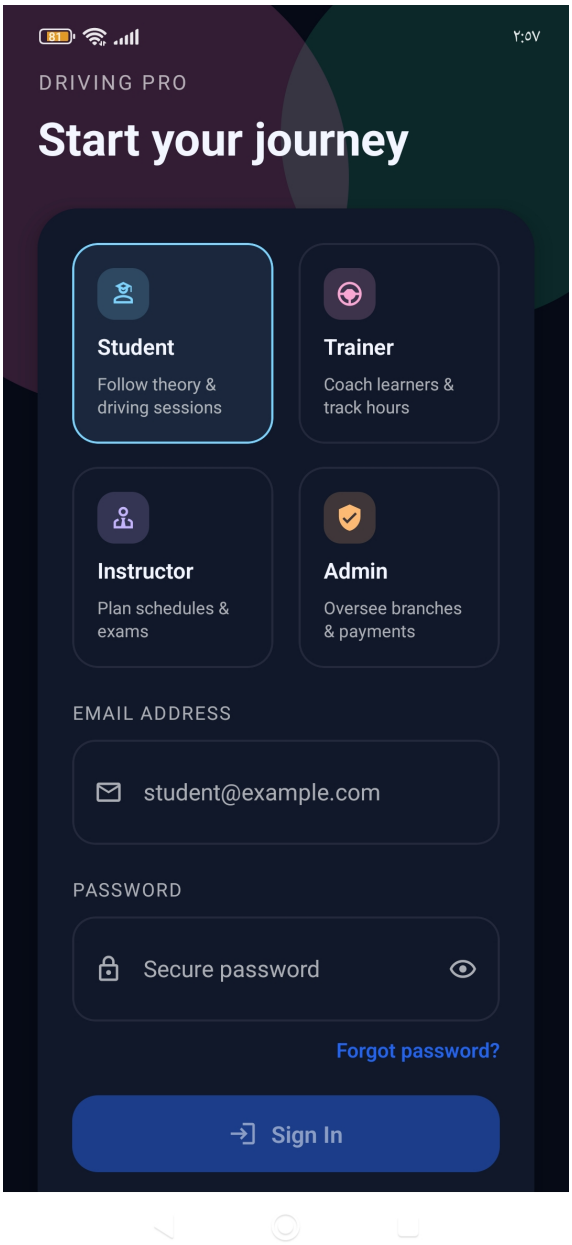

Figure 9: Mobile Application - Login Screen

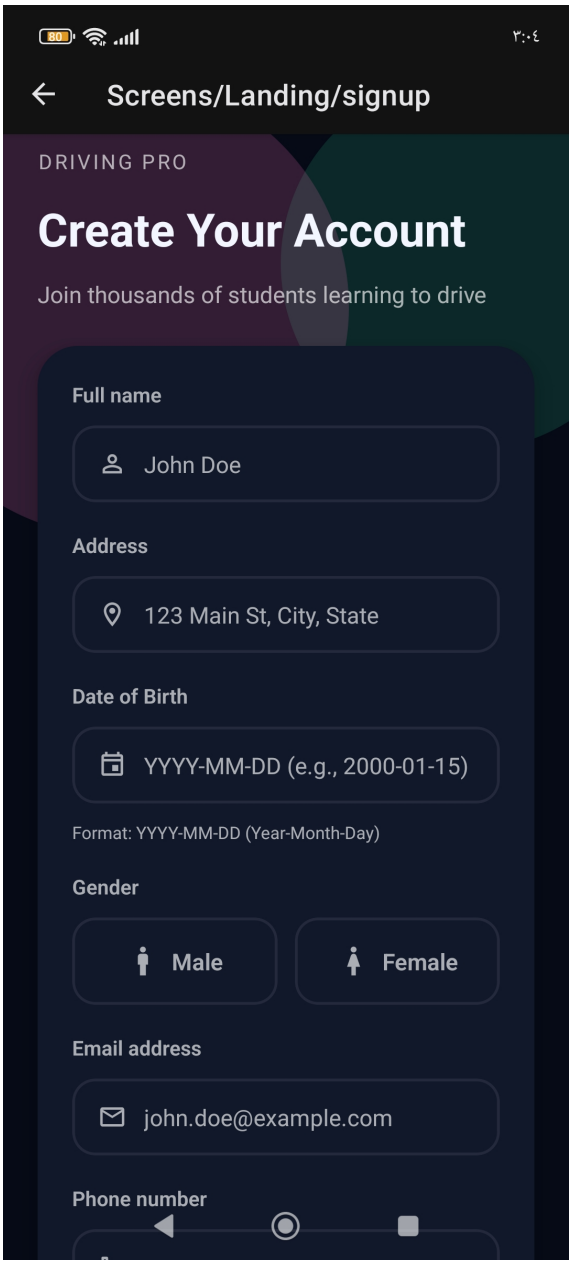

Figure 10: Mobile Application - Create Account (Step 1)

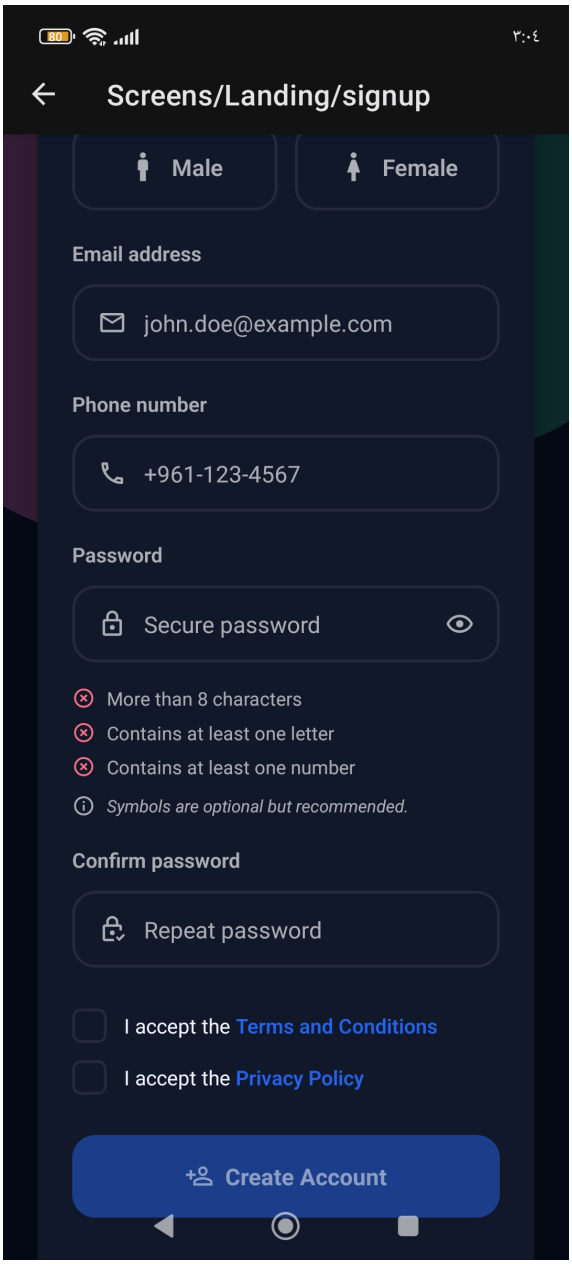

Figure 11: Mobile Application - Create Account (Step 2)

5.3 Database Schema

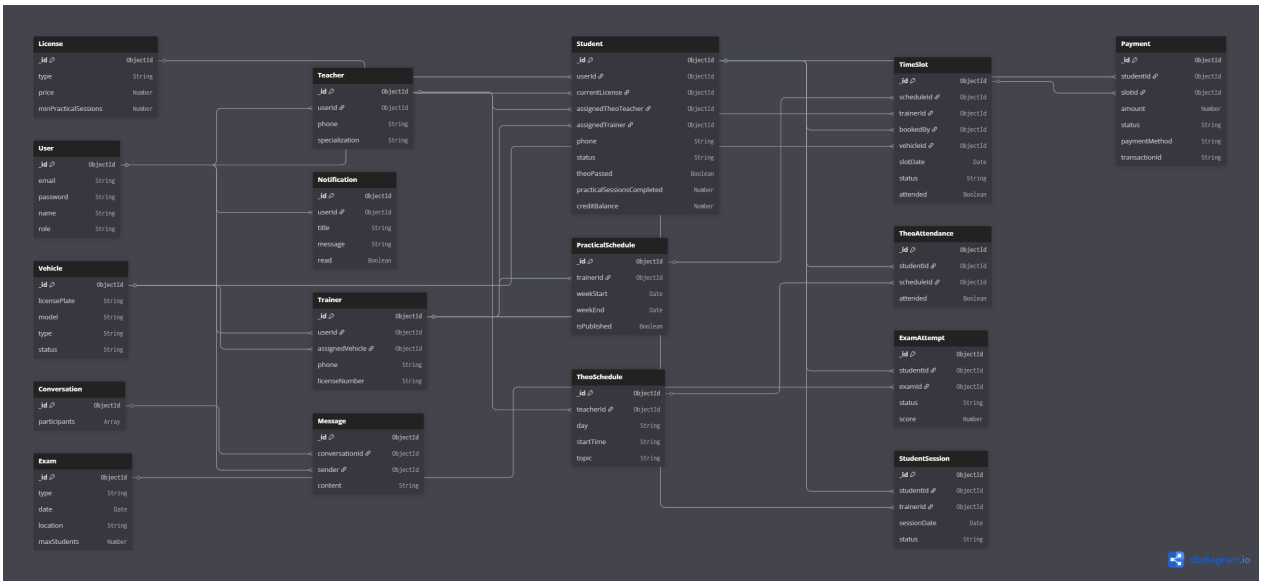

Figure 12: Database Entity-Relationship Diagram - 17 MongoDB Collections

The system uses MongoDB with the following main collections:

- **Users** - Base user authentication (email, hashed password, role)
- **Students** - Student profiles and progress (currentLicense, theoPassed, practicalSessionsCompleted)
- **Teachers** - Theoretical instructors (specialization, assignedStudents)
- **Trainers** - Practical driving instructors (licenseNumber, assignedVehicle, assignedStudents)
- **Licenses** - Available license types (type, price, minPracticalSessions)
- **PracticalSchedule** - Trainer weekly schedules (weekStart, weekEnd, location)
- **TimeSlots** - Bookable practical sessions (day, startTime, endTime, status, bookedBy)
- **Payments** - Transaction records (amount, status, paymentMethod, cardLast4)
- **Exams** - Scheduled examinations (type, date, location, maxStudents)
- **ExamAttempt** - Student exam registrations (studentId, examId, score, status)
- **TheoSchedule** - Theoretical class schedules (teacherId, day, startTime, topic)
- **TheoAttendance** - Theory attendance records (studentId, scheduleId, attended)
- **Vehicles** - Driving school fleet (licensePlate, model, type, status)

- **Notification** - Push notifications history
- **Conversation** - Messaging conversations
- **Message** - Individual messages
- **StudentSession** - Session booking records

5.4 System Architecture and Data Flow

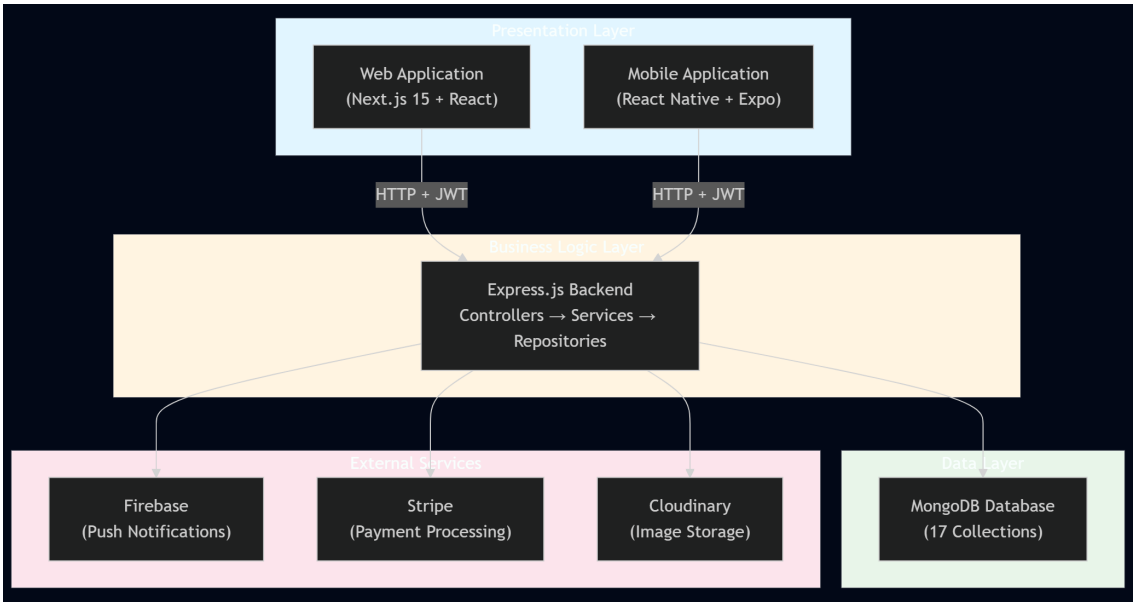

Figure 13: High-Level System Architecture - Three-Tier Design

High-Level Architecture:

Our system follows a three-tier architecture pattern:

1. **Presentation Layer** - Web (Next.js) and Mobile (React Native) applications that users interact with
2. **Business Logic Layer** - Node.js/Express backend that processes requests and implements business rules
3. **Data Layer** - MongoDB database that stores all system data

Request Flow Example (Student Booking Session):

1. Student clicks "Book Session" button on mobile/web app
2. Frontend sends HTTP POST request to `/api/student/book-slot` with JWT token in Authorization header
3. Express middleware verifies JWT token and extracts user information

4. Role middleware checks if user has "student" role
5. Request reaches StudentController which calls StudentService
6. StudentService validates booking (checks weekly limits, daily limit, slot availability)
7. StudentRepository queries MongoDB for available slots and creates booking
8. Payment record created in database
9. Notification sent via Firebase FCM to student's device
10. Success response sent back to frontend
11. Frontend updates UI with confirmation message

5.5 Security and Data Protection

Authentication Mechanism:

- Passwords are hashed using bcrypt with 10 salt rounds before storage
- Users never stored in plain text in database
- Upon successful login, server generates JWT (JSON Web Token) containing user ID, email, name, and role
- JWT expires after 24 hours, requiring re-login for security
- Token stored securely: localStorage on web, SecureStore on mobile

Authorization Strategy:

- Every protected API route requires JWT token verification
- `verifyToken` middleware extracts and validates token
- `requireRole` middleware checks if user has required role for the endpoint
- Example: Student routes reject requests from Teacher/Trainer/Admin roles
- Prevents unauthorized access to sensitive operations

API Security:

- Input validation on all endpoints prevents invalid data
- MongoDB injection prevented through Mongoose parameterized queries
- CORS configured to allow only trusted origins
- Rate limiting can be added to prevent abuse

- Sensitive operations (exam grading, student approval) restricted to admin only

Payment Security:

- No credit card information stored on our servers
- All card data handled by Stripe PCI-compliant servers
- We only store last 4 digits of card for display purposes
- Stripe webhooks use signature verification to prevent tampering
- Test mode ensures no real money transactions during development

Data Privacy:

- Student personal information (phone, address, date of birth) only accessible by admin and assigned instructors
- Payment history only visible to student owner and admin
- Messages only visible to conversation participants
- No data sharing with third parties (except Stripe for payments, Cloudinary for images)

5.6 External Services Integration

Firebase Cloud Messaging (FCM):

- Used for sending push notifications to mobile devices
- Admin configures Firebase project and downloads service account key
- Backend sends notification requests to FCM API
- Mobile app receives notifications even when app is closed
- Notifications include title, body, and relevant data (booking ID, exam date, etc.)

Stripe Payment Gateway:

- Handles online credit/debit card payments
- Backend creates payment intents via Stripe API
- Frontend displays Stripe's secure payment form
- Webhooks notify our system when payment succeeds/fails
- Currently in TEST MODE for academic purposes

Cloudinary Image Storage:

- Stores profile images for users
- Backend uploads images to Cloudinary via API
- Returns public URL for accessing image
- Reduces load on our server

6 Administrator Module

6.1 Admin Dashboard

The administrator dashboard provides a comprehensive overview of the driving school operations including student statistics, revenue metrics, upcoming exams, and recent activities.

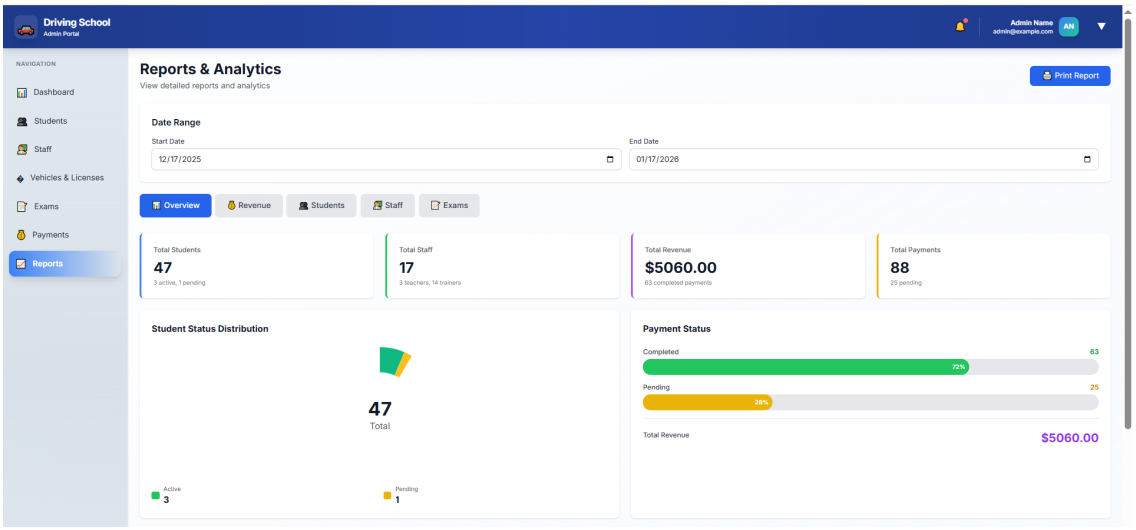

Figure 14: Admin Dashboard - Overview (Web)

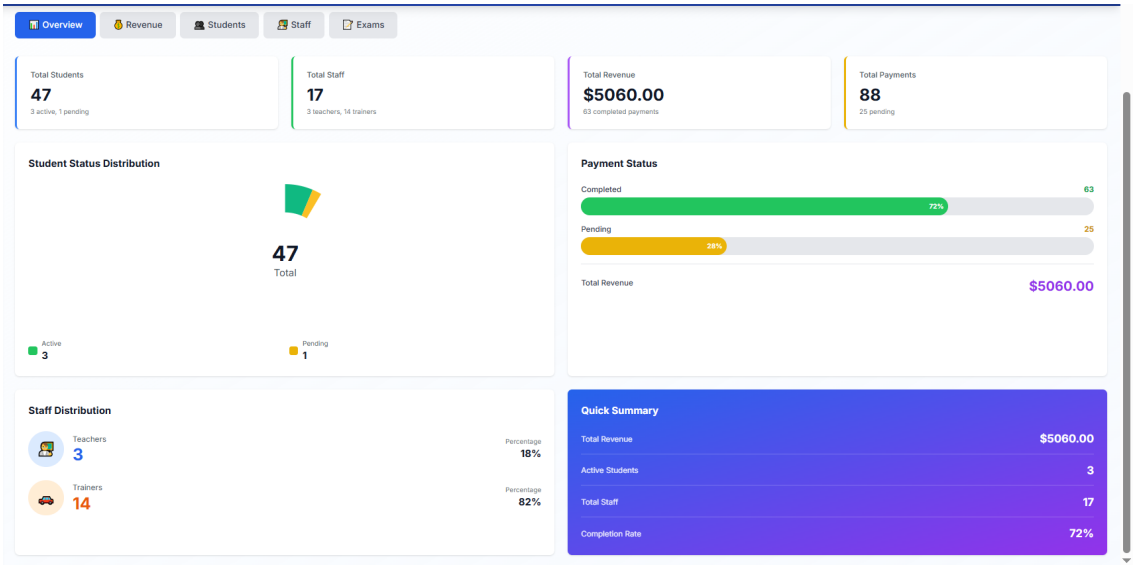

Figure 15: Admin Dashboard - Statistics (Web)

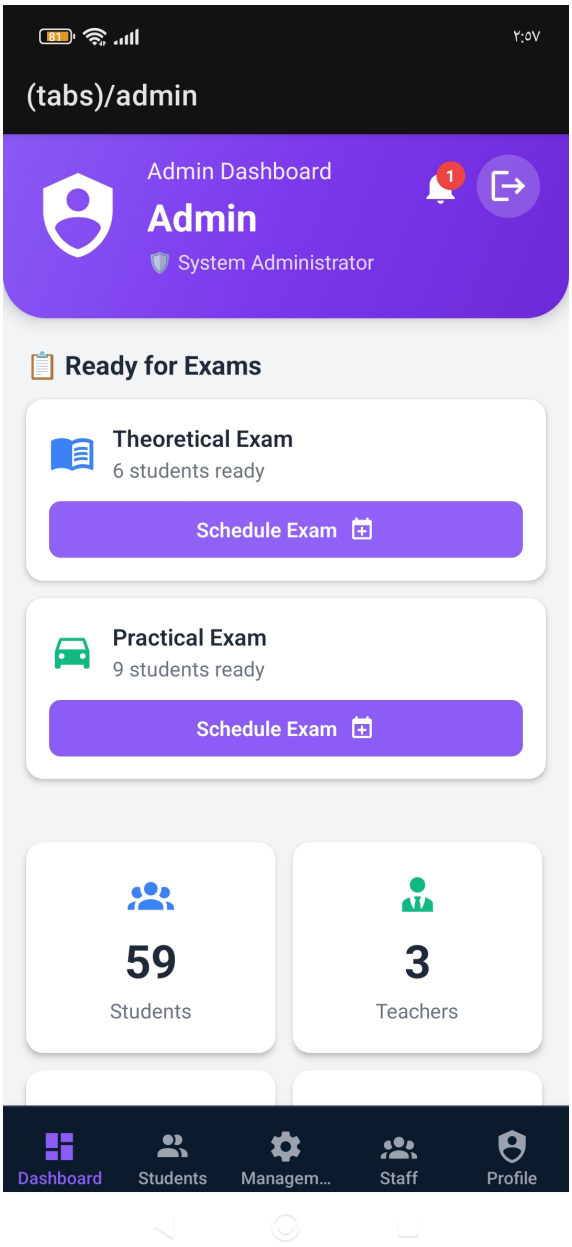

Figure 16: Admin Dashboard - Mobile View (Part 1)

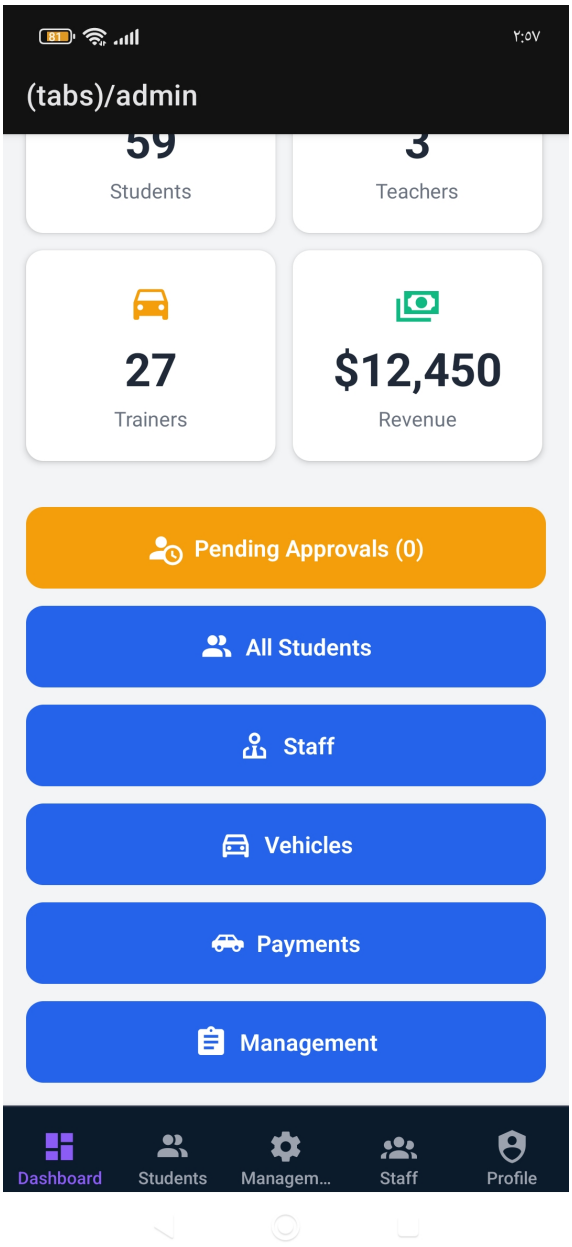

Figure 17: Admin Dashboard - Mobile View (Part 2)

6.2 Student Management

Administrators can view all registered students, approve new registrations, assign teachers and trainers, and track student progress throughout their journey.

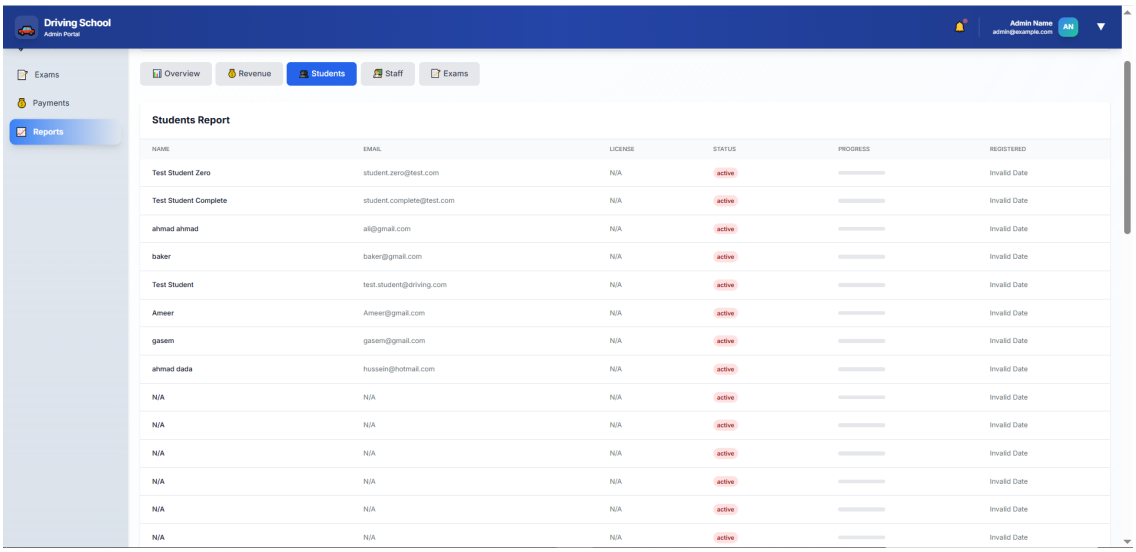

Figure 18: Student Management - Overview (Web)

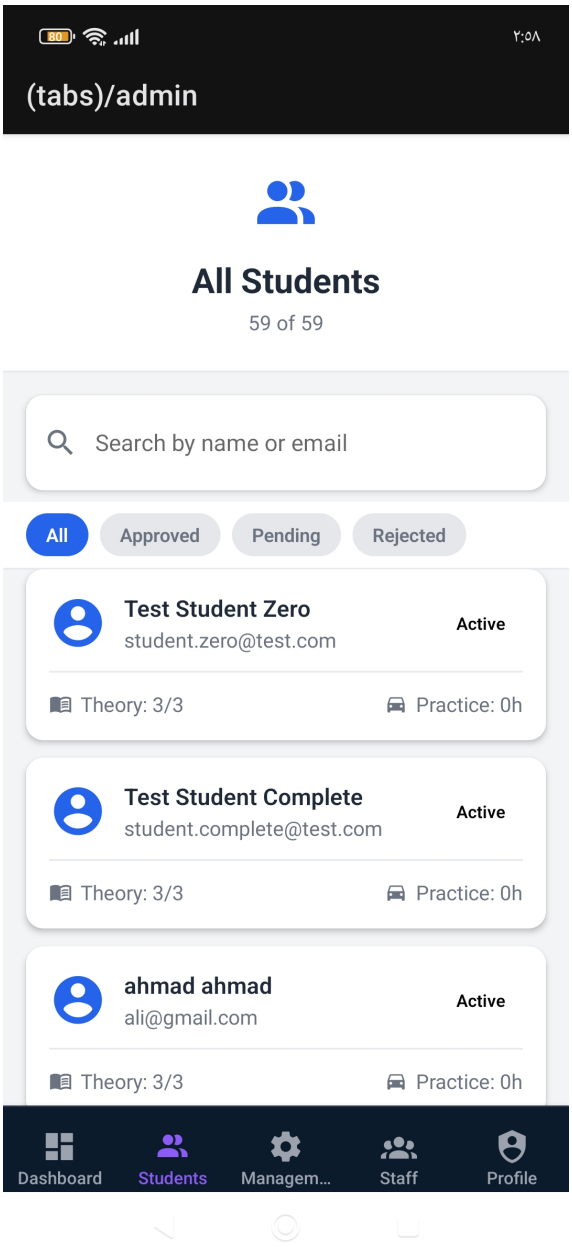

Figure 19: Student Management - Mobile View

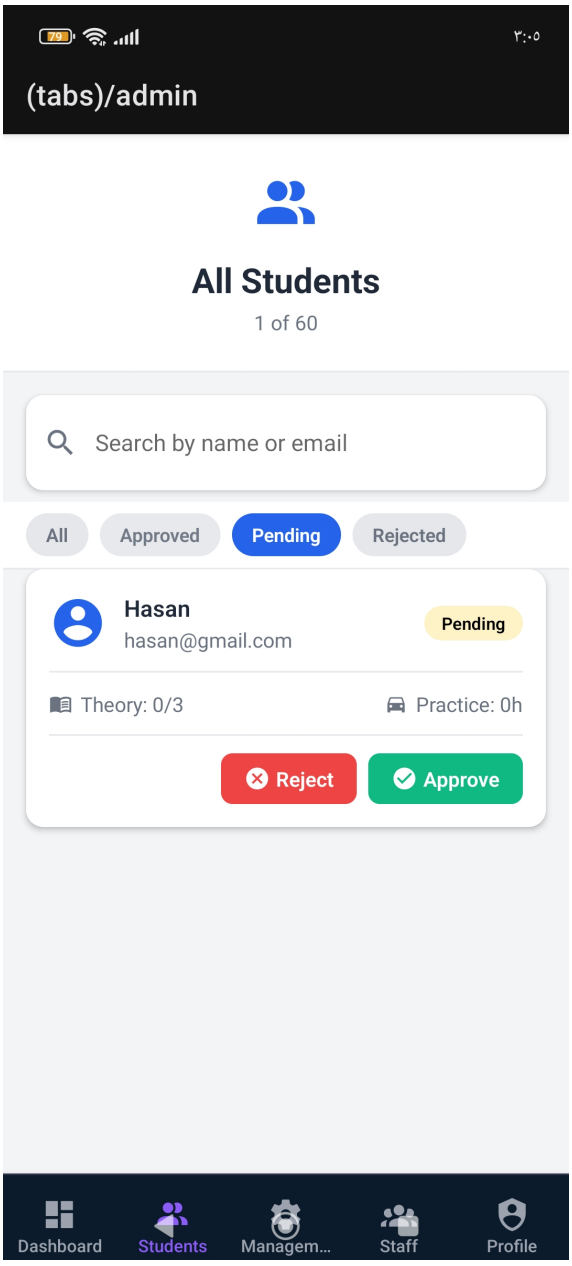

Figure 20: Student Approval - Mobile Interface

6.3 Staff Management

6.3.1 Teachers Management

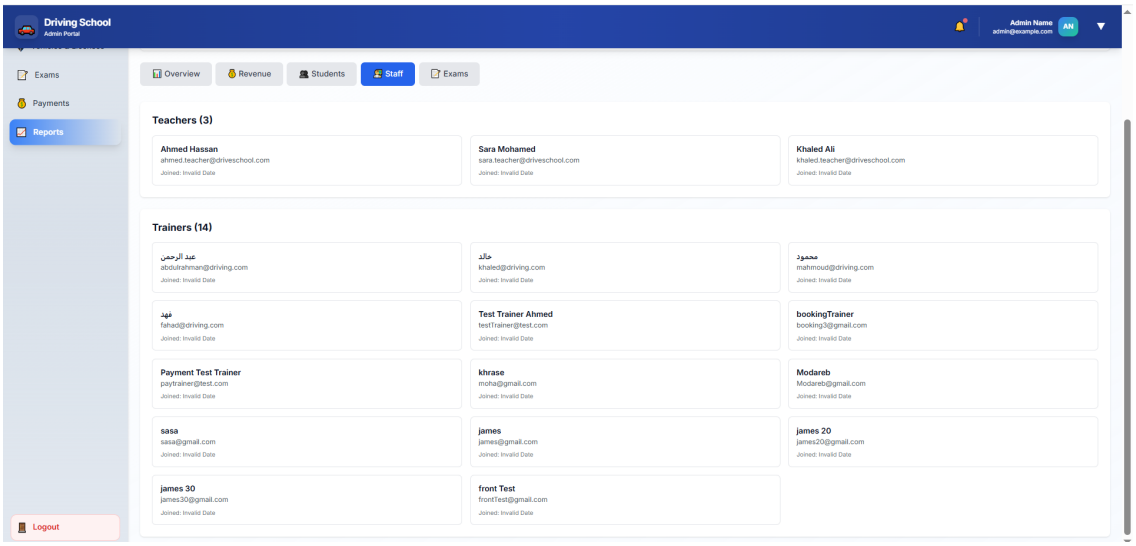

Figure 21: Staff Management - Overview (Web)

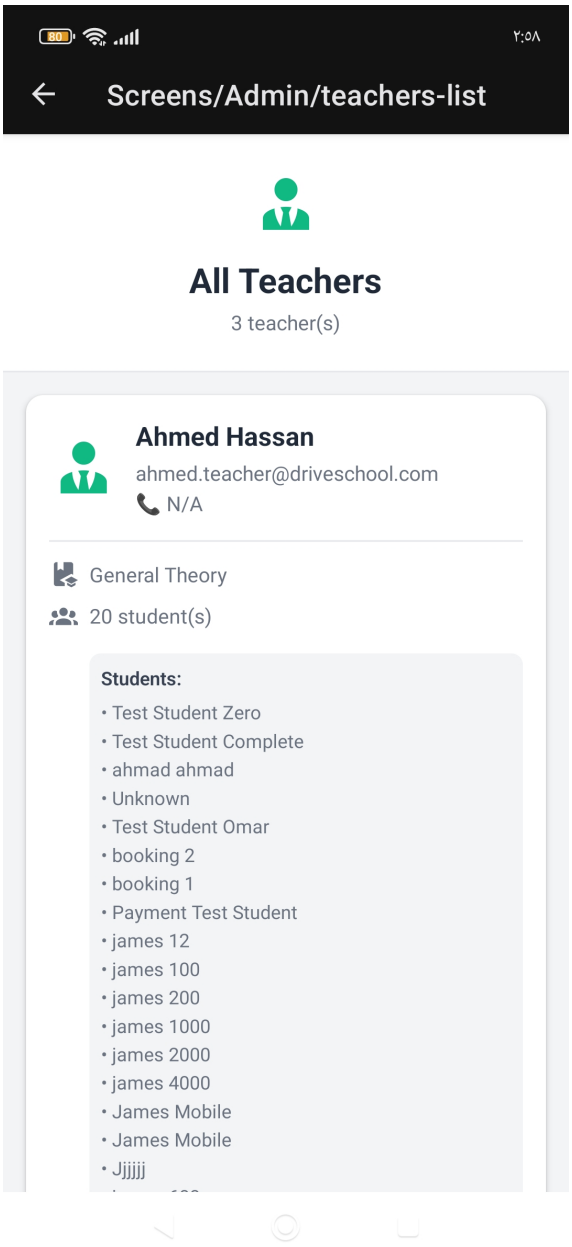

Figure 22: Teachers List - Mobile View

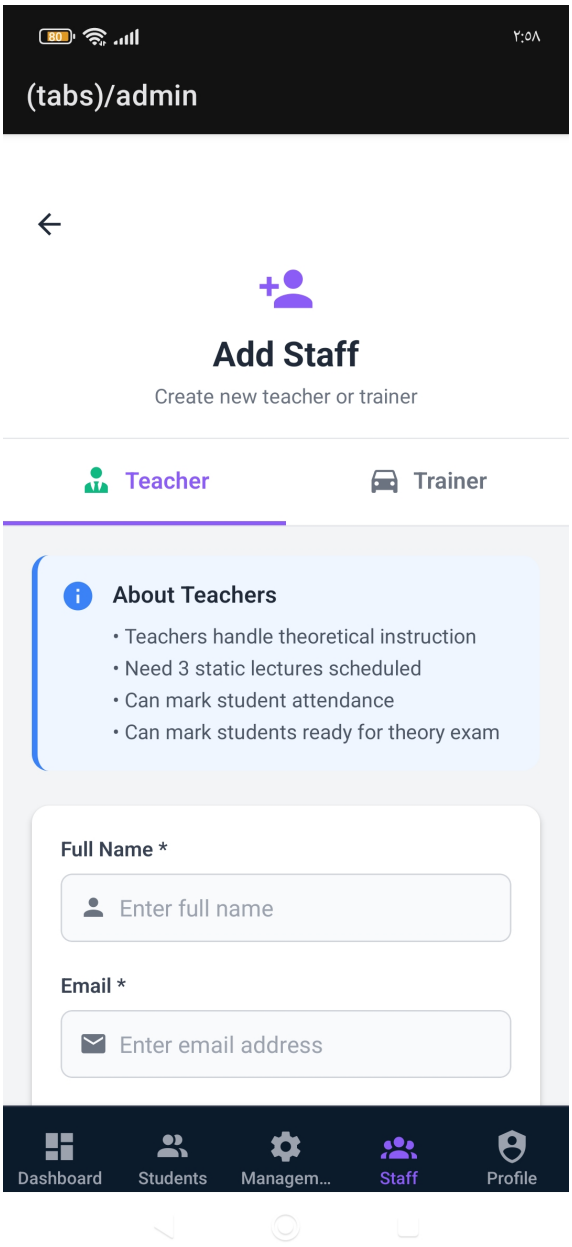

Figure 23: Staff Management - Mobile Interface

6.3.2 Trainers Management

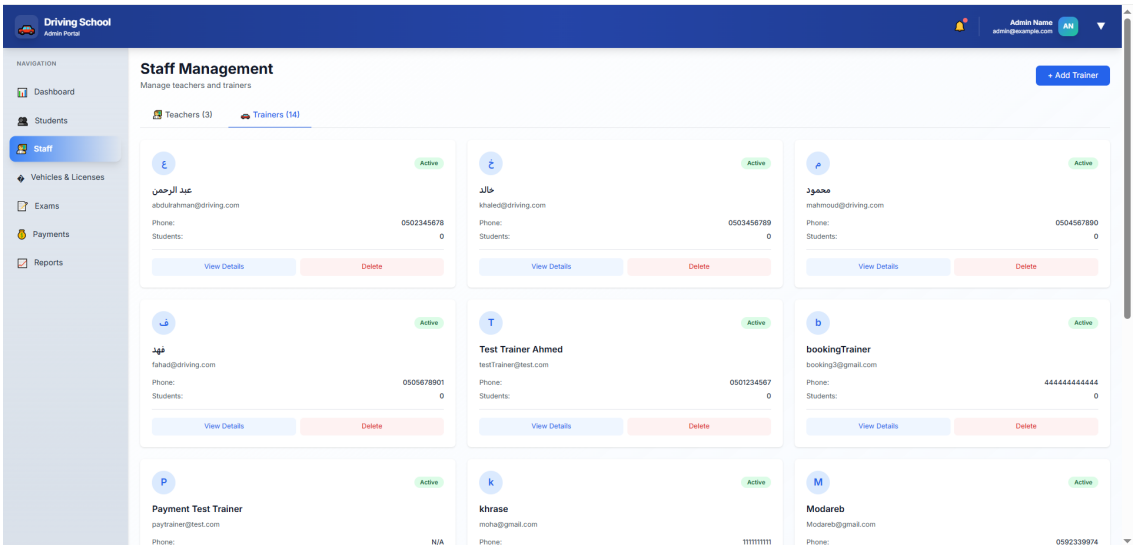

Figure 24: Trainers Management - Web View

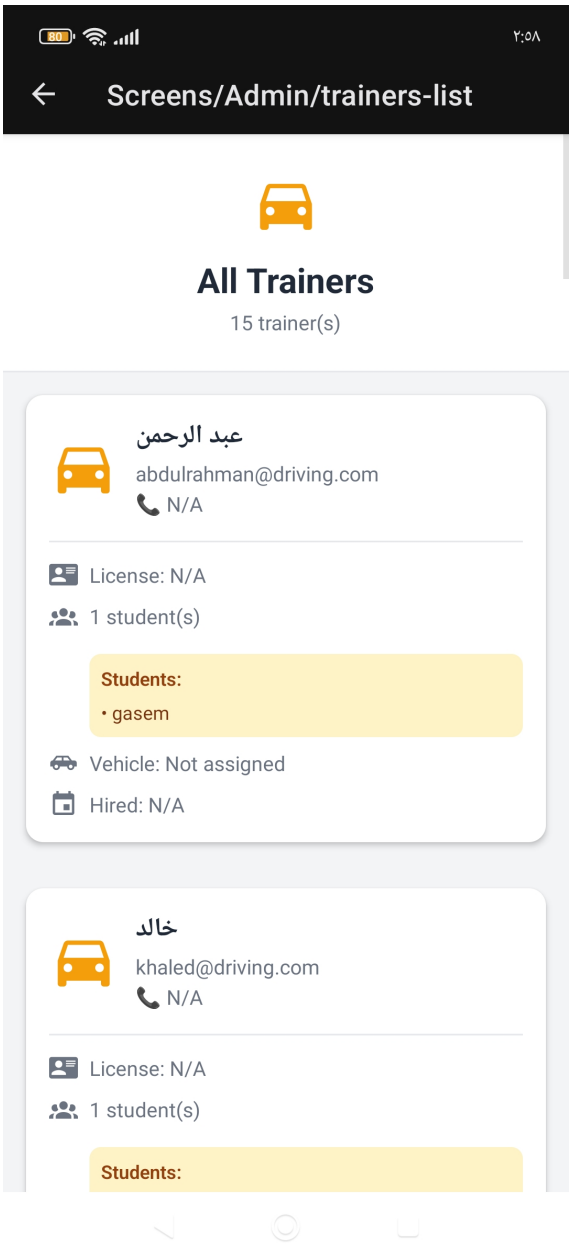

Figure 25: Trainers List - Mobile View

6.4 Exam Management

6.4.1 Exam Scheduling

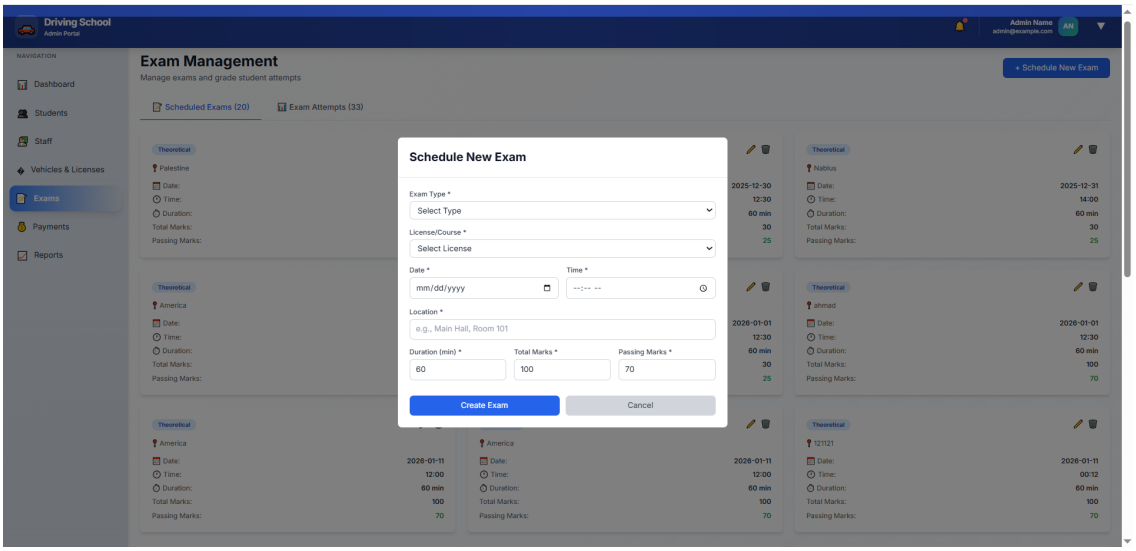

Figure 26: Schedule New Exam - Web Interface

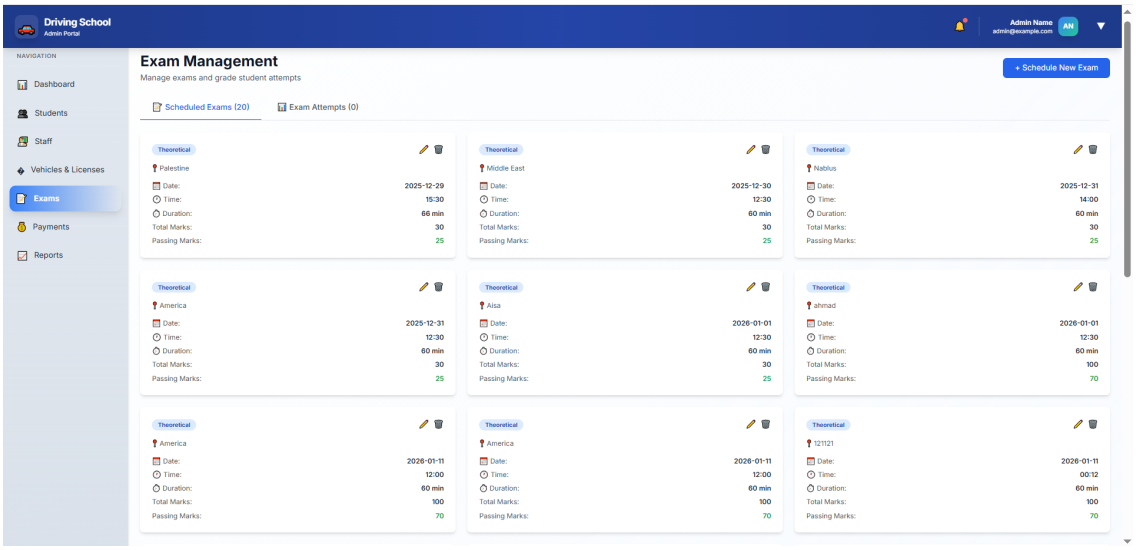

Figure 27: Exam Search and Filter - Web View

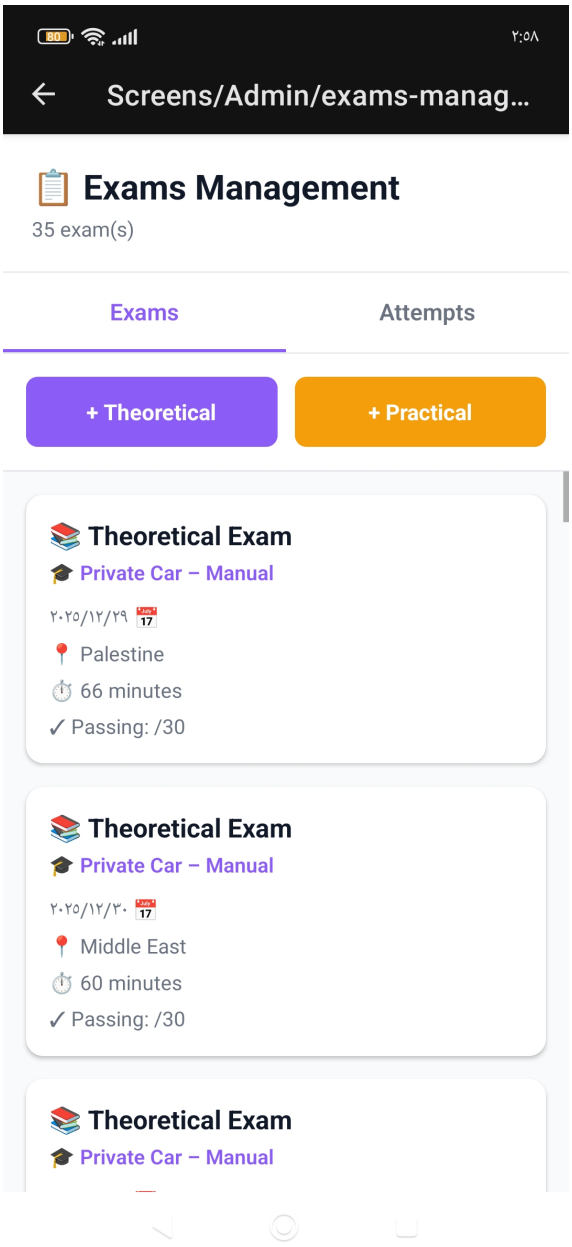

Figure 28: Exams Management - Mobile View

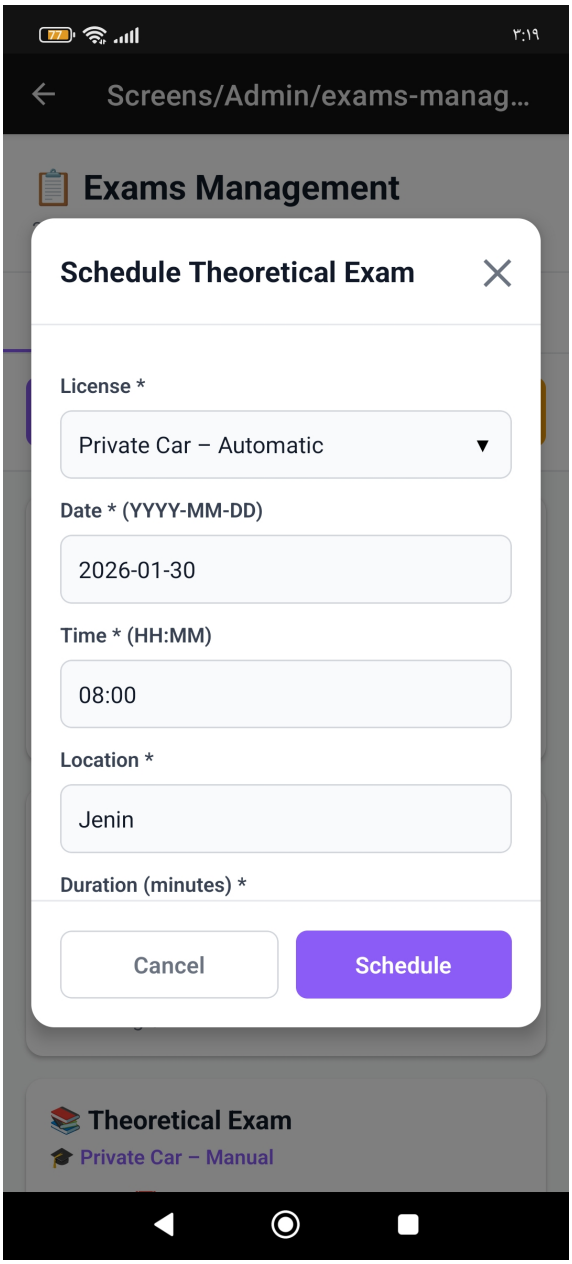

Figure 29: Create Theoretical Exam (Step 1) - Mobile

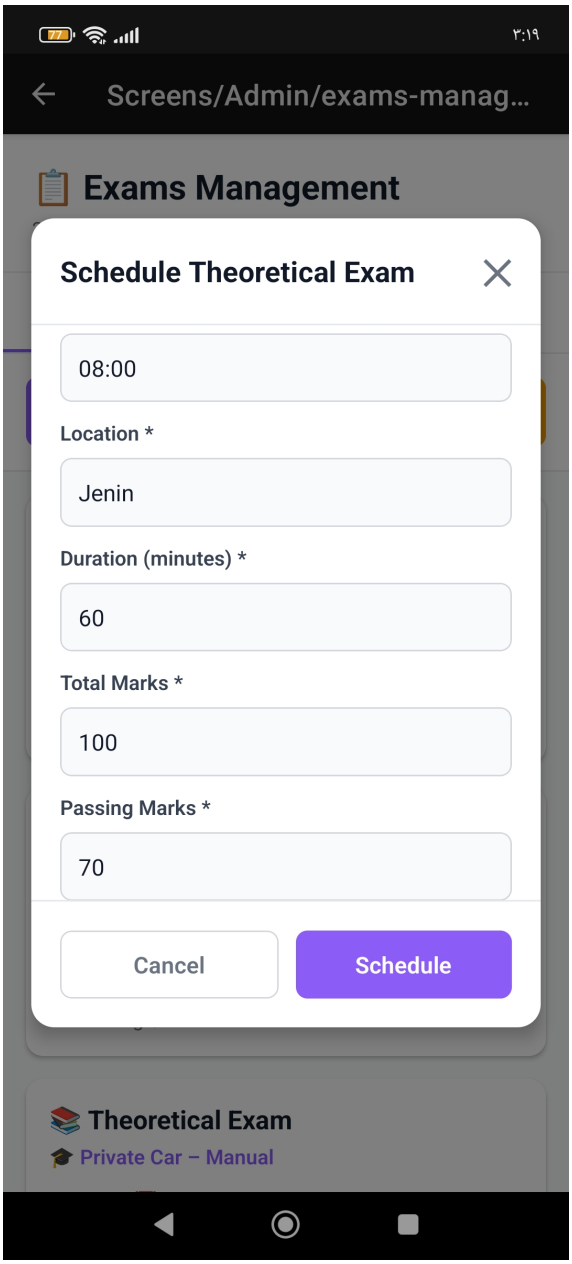

Figure 30: Create Theoretical Exam (Step 2) - Mobile

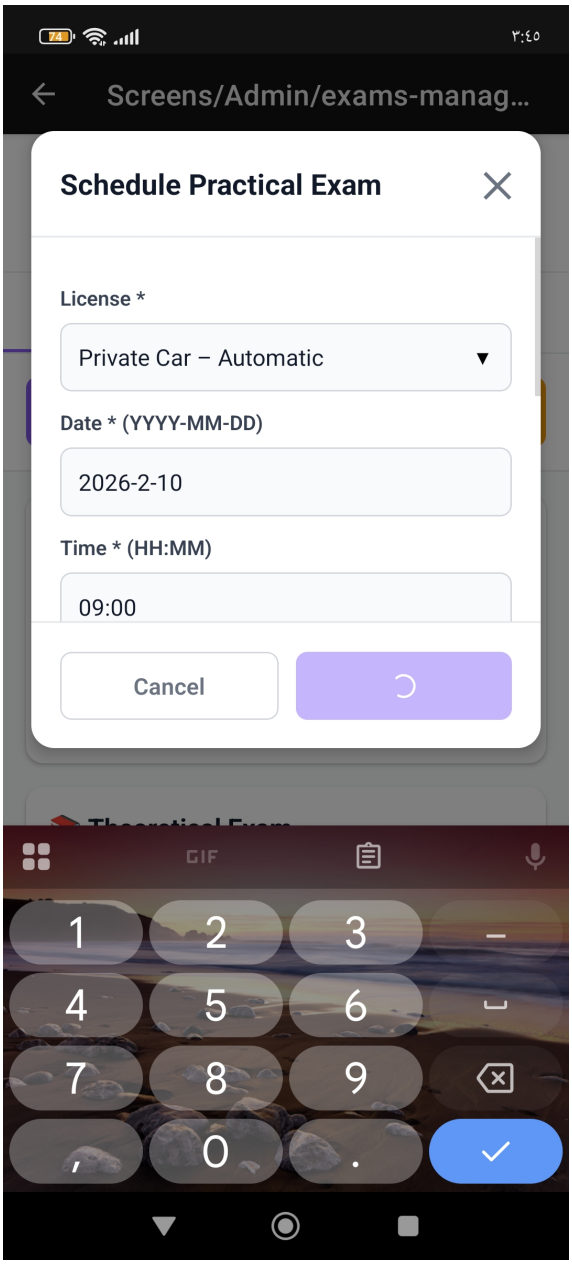

Figure 31: Schedule Practical Exam - Mobile

6.4.2 Exam Attempts and Grading

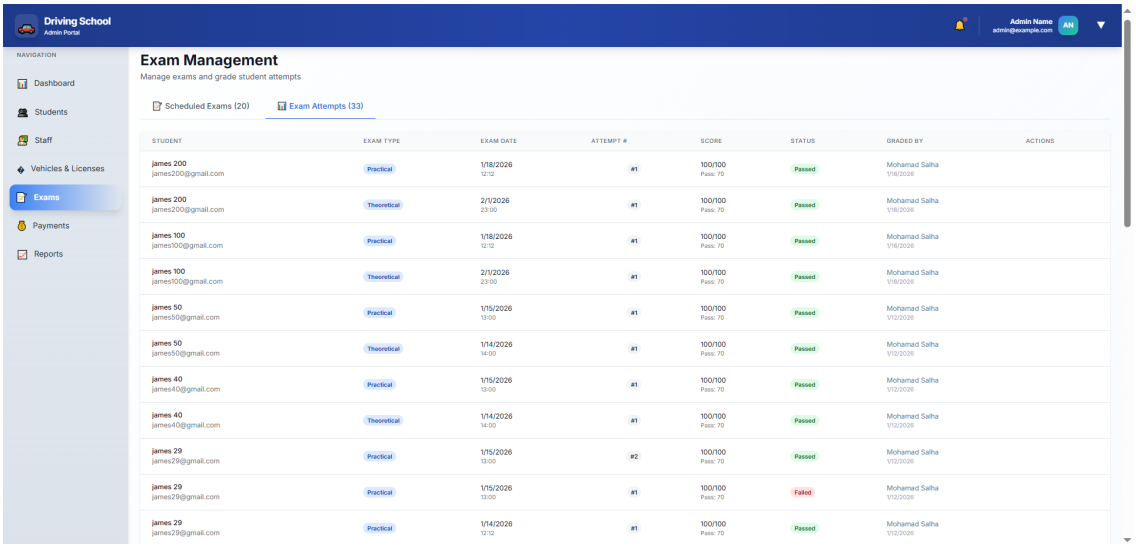

Figure 32: Exam Attempts - Web View

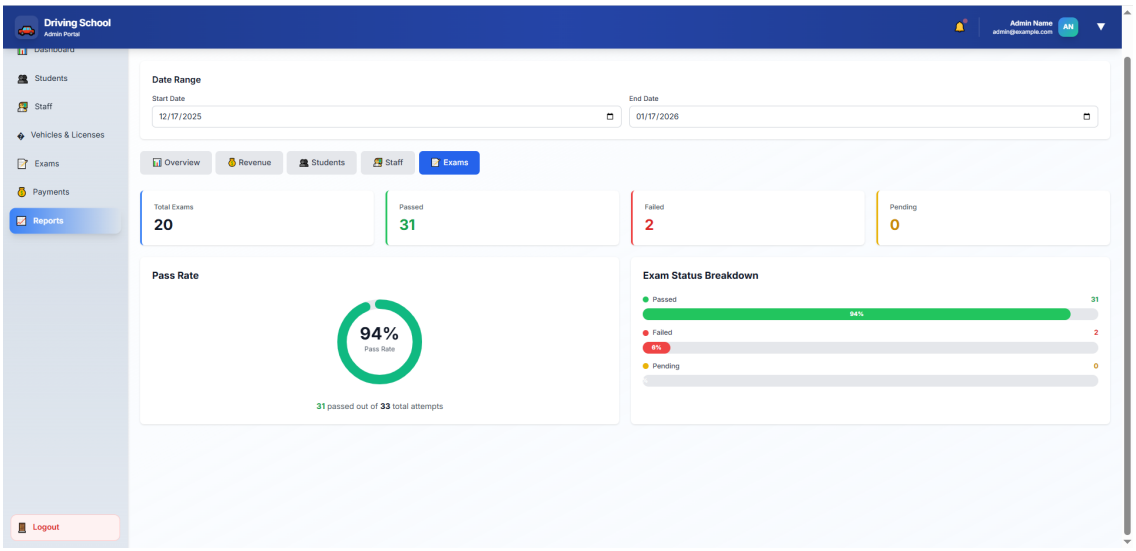

Figure 33: Exam Reports - Web View

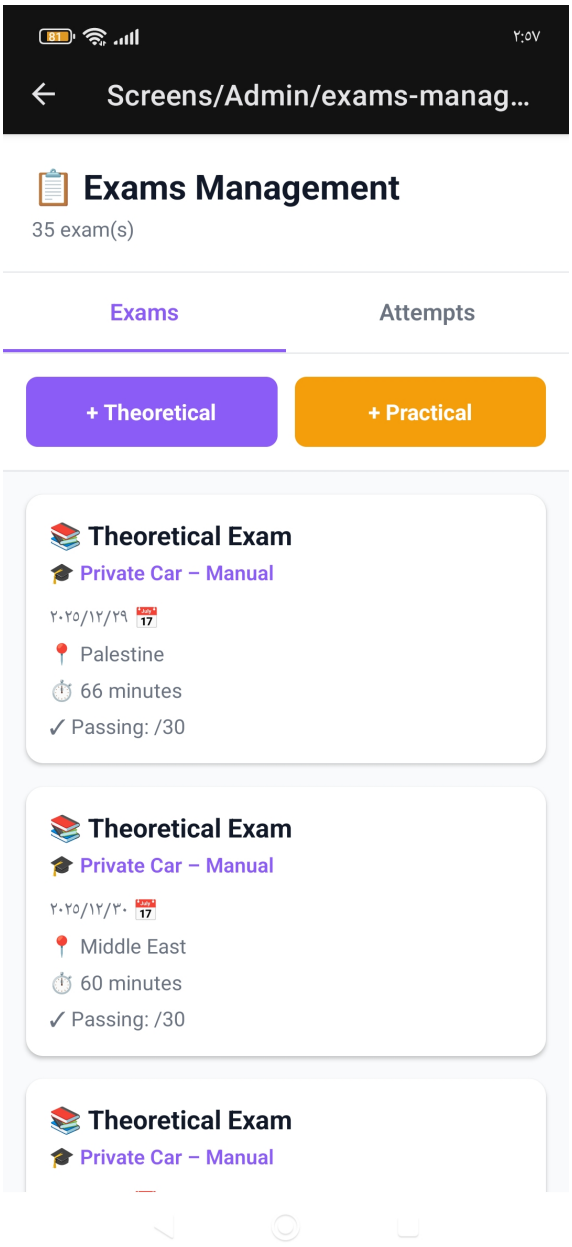

Figure 34: Exam Management - Mobile Interface

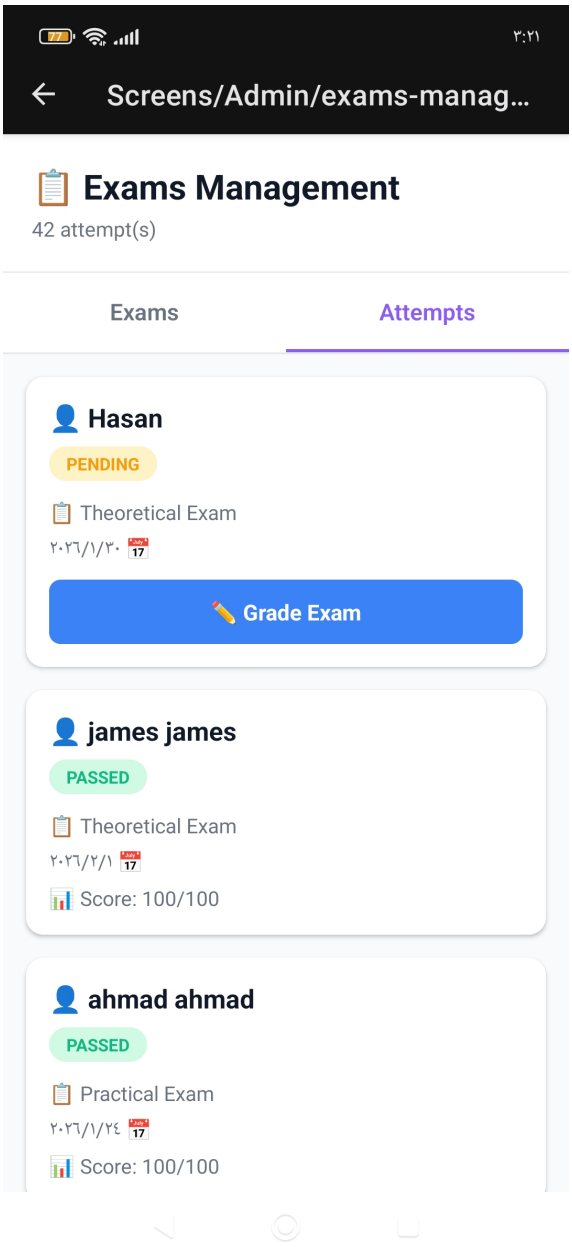

Figure 35: Exam Attempts - Mobile View

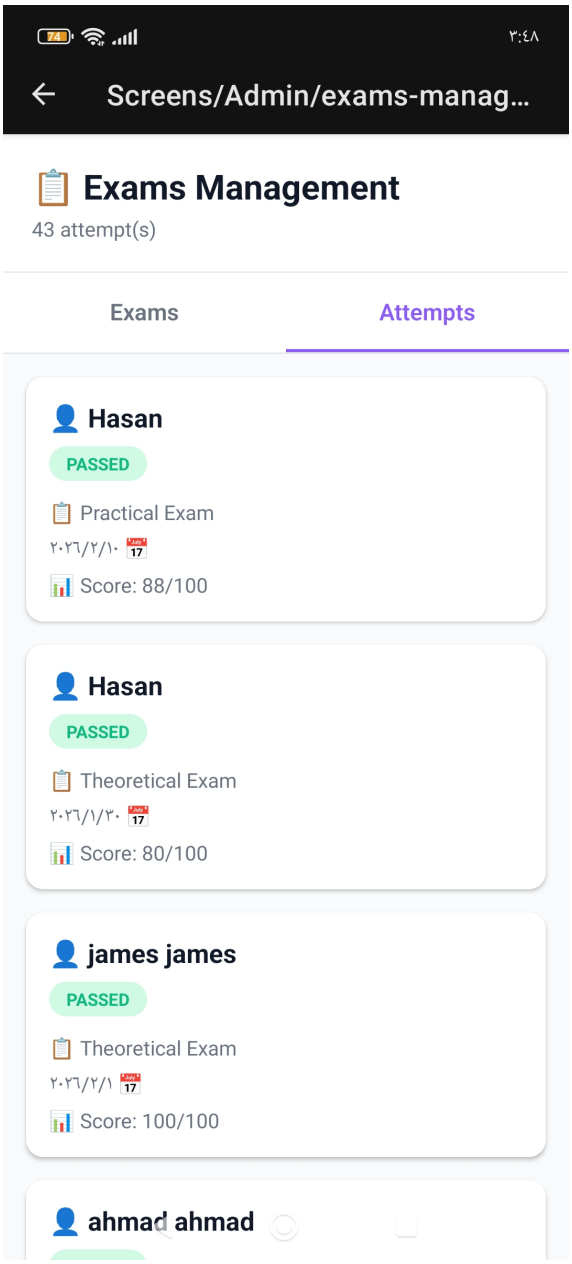

Figure 36: Manage Exam Attempts - Mobile

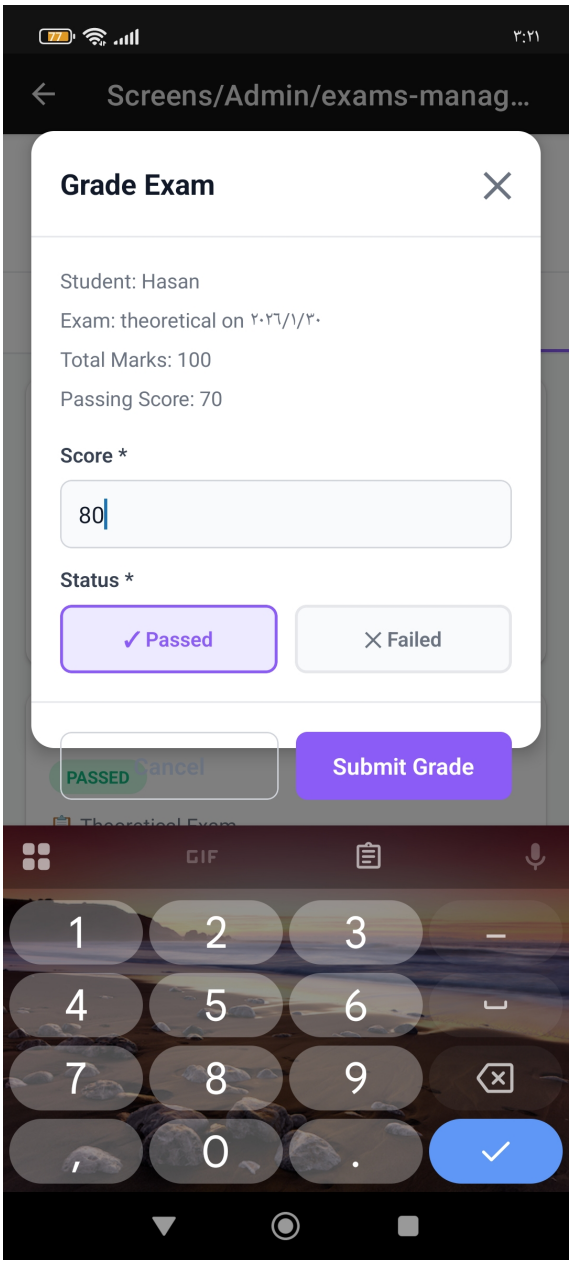

Figure 37: Grade Theoretical Exam - Mobile

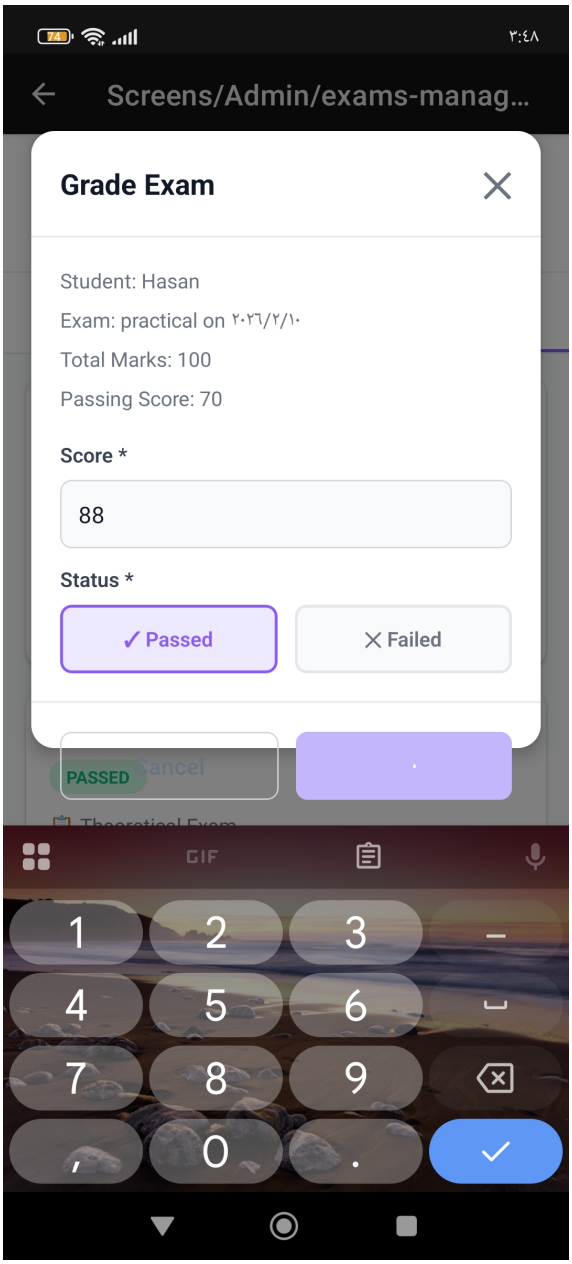

Figure 38: Grade Practical Exam - Mobile

6.5 Vehicle Management

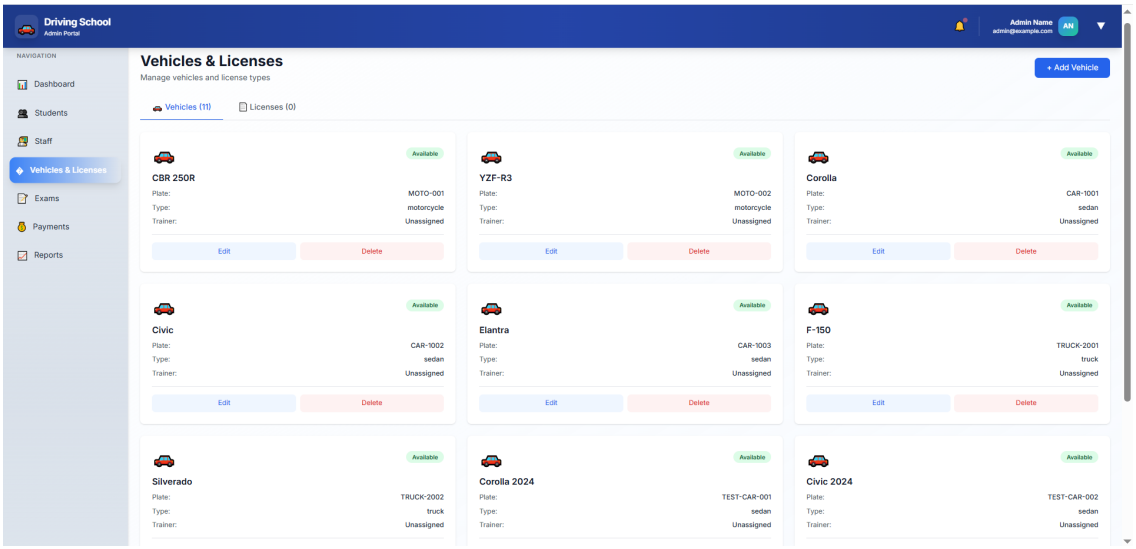

Figure 39: Vehicle Fleet Management - Web View

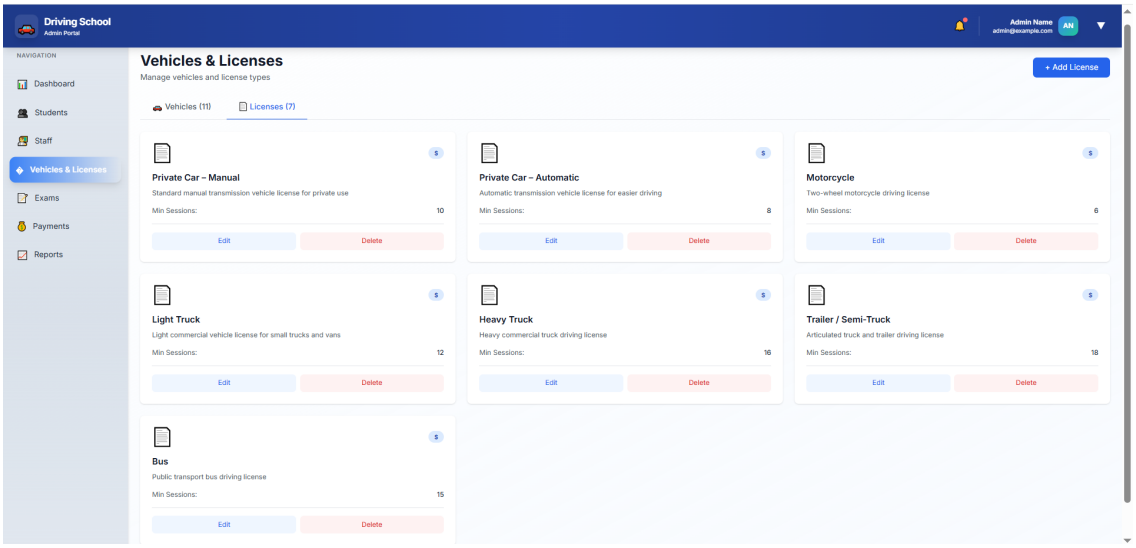

Figure 40: License Types Management - Web View

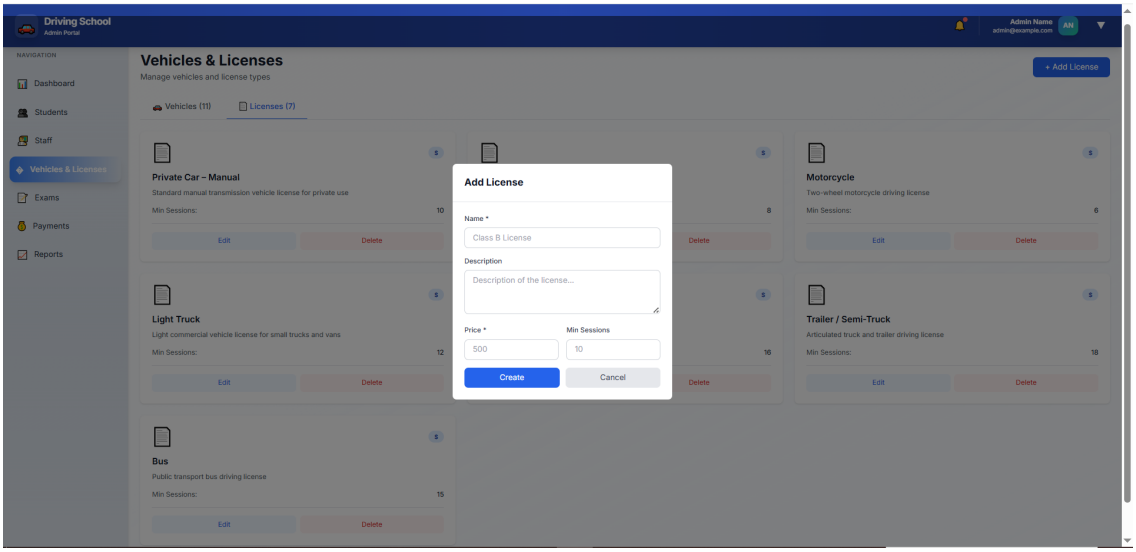

Figure 41: Add New License Type - Web Interface

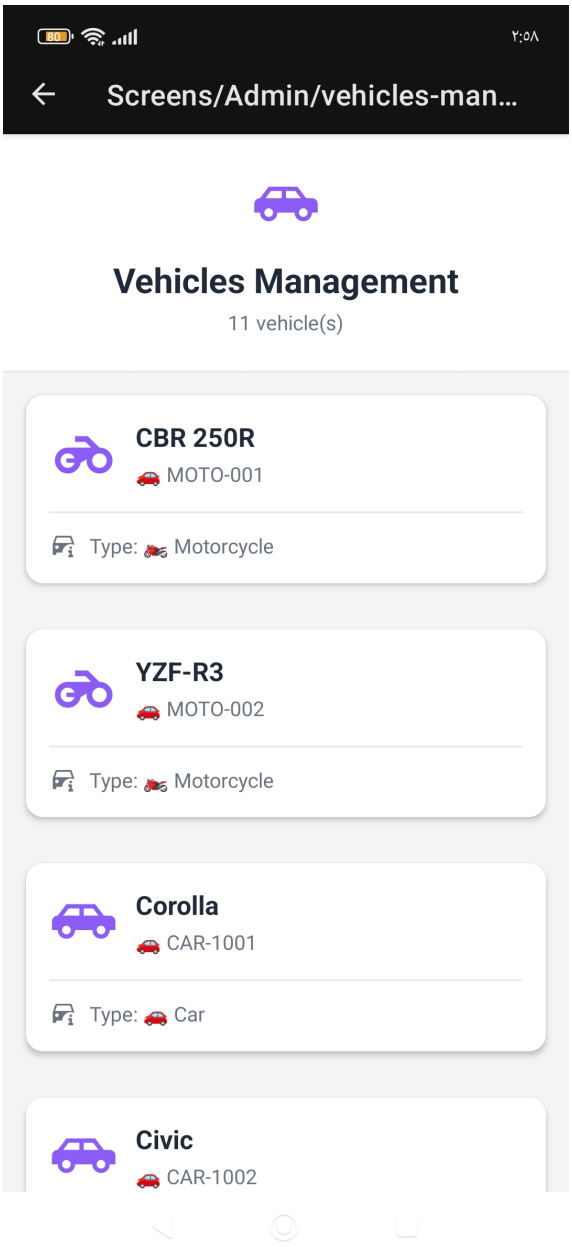

Figure 42: Vehicle Management - Mobile View

6.6 Payment and Revenue Management

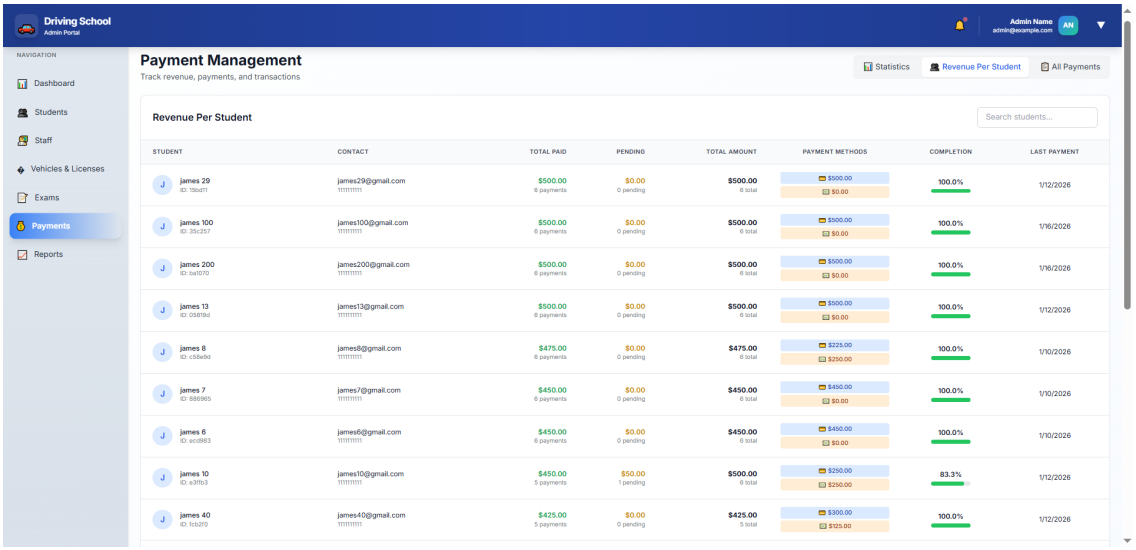

Figure 43: Revenue Dashboard - Web View

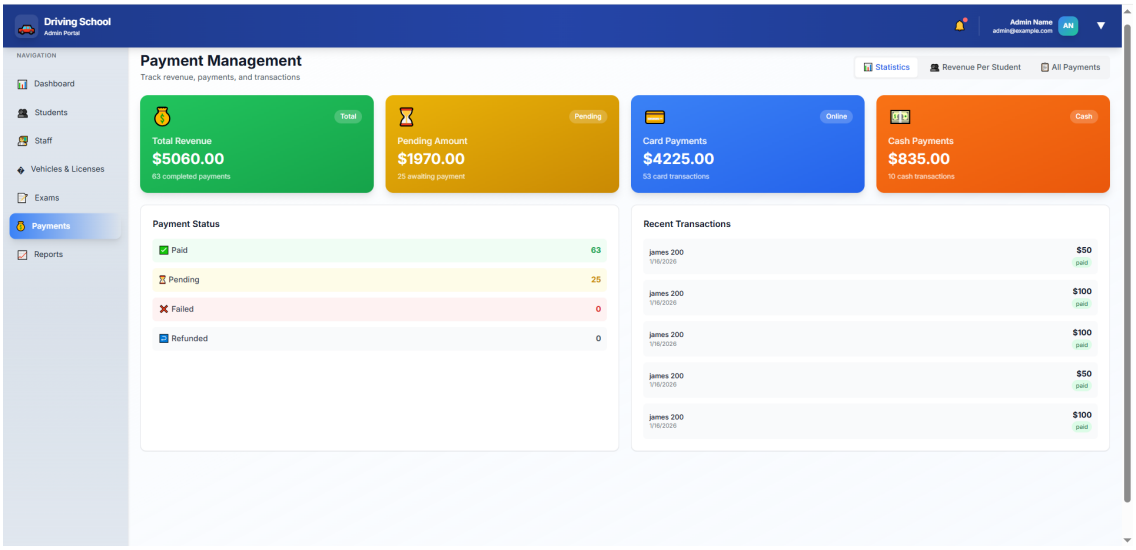

Figure 44: Payment Statistics - Web View

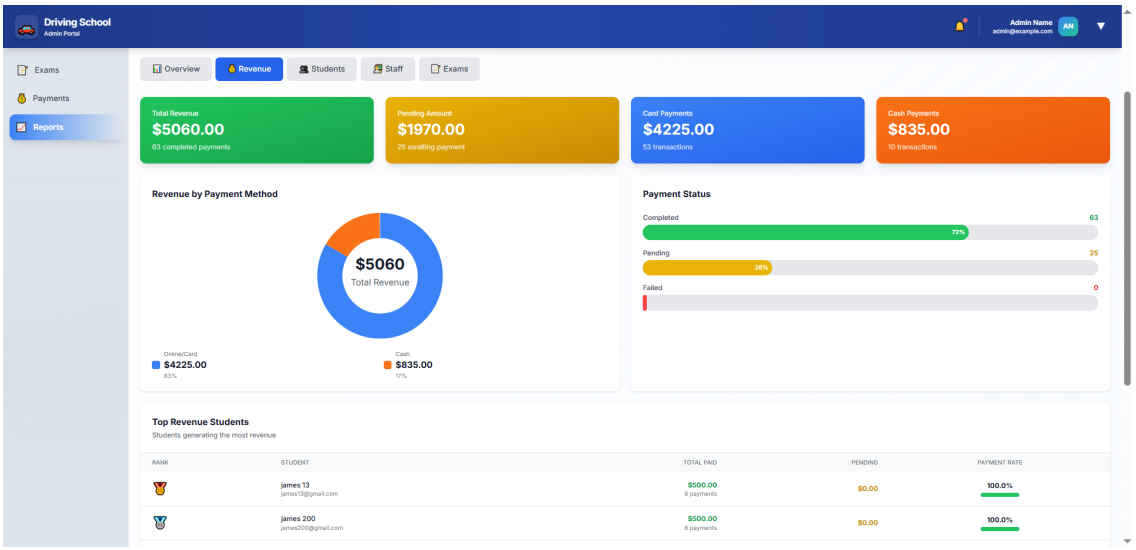

Figure 45: Revenue Reports - Web View

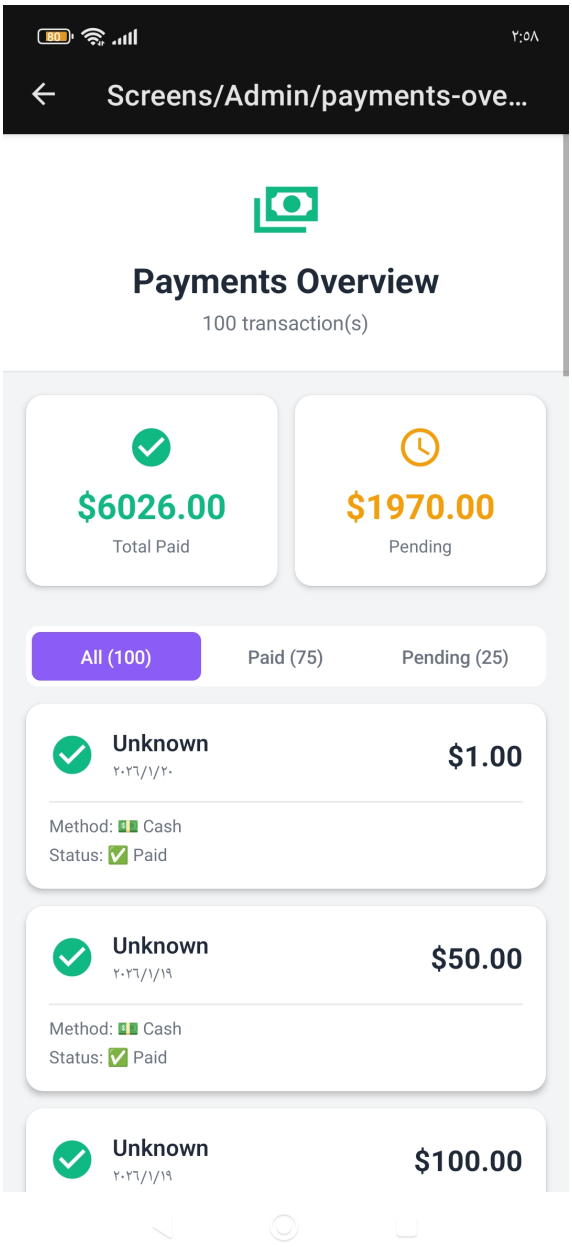

Figure 46: Payment Management - Mobile View

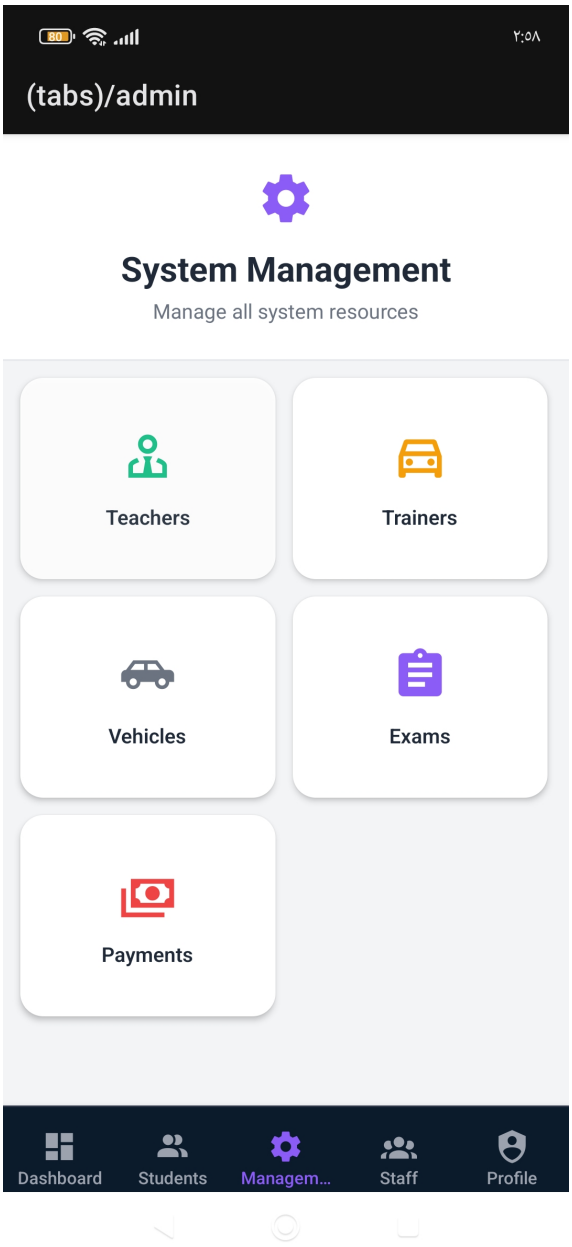

Figure 47: Administrative Management - Mobile Dashboard

6.7 Reports and Analytics

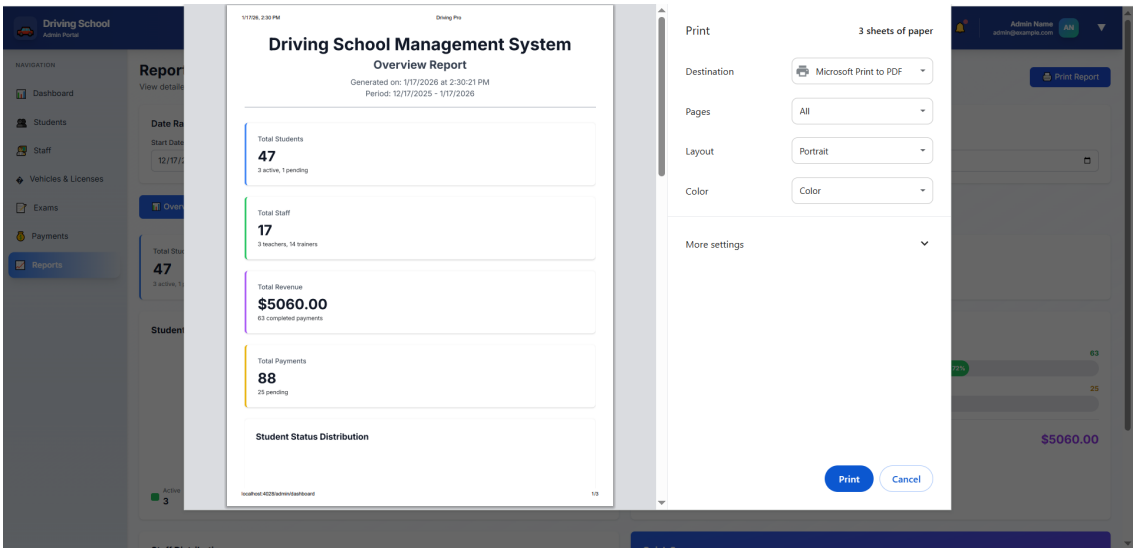

Figure 48: Printable Reports - Web Interface

7 Student Module

7.1 Student Enrollment Journey

7.1.1 License Selection

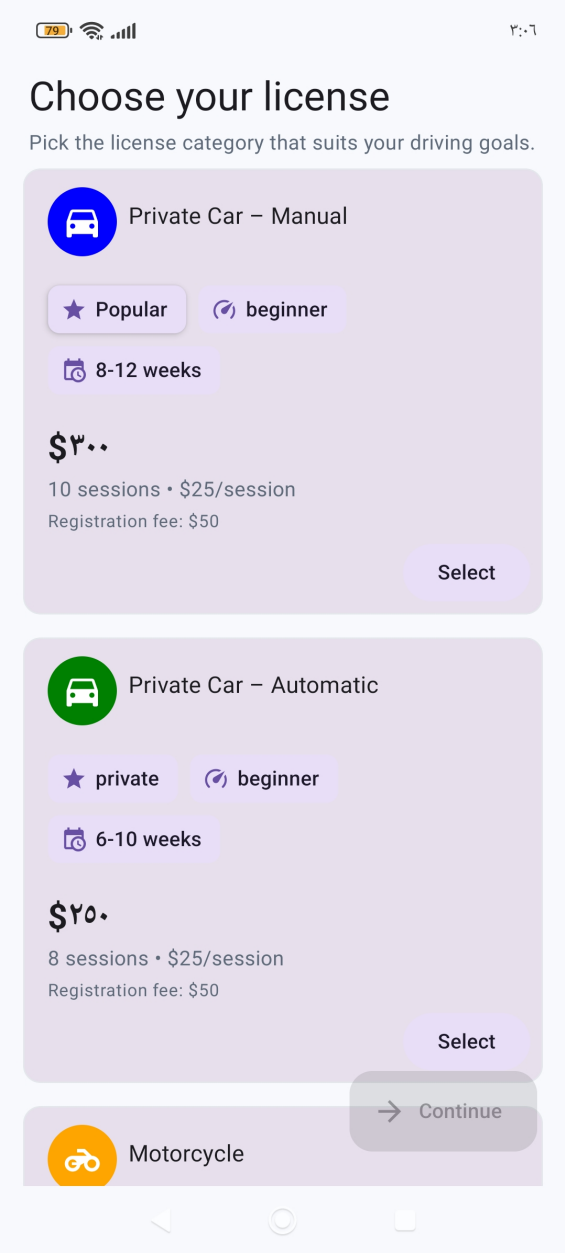

Figure 49: License Selection - Mobile Interface

7.1.2 Instructor Selection

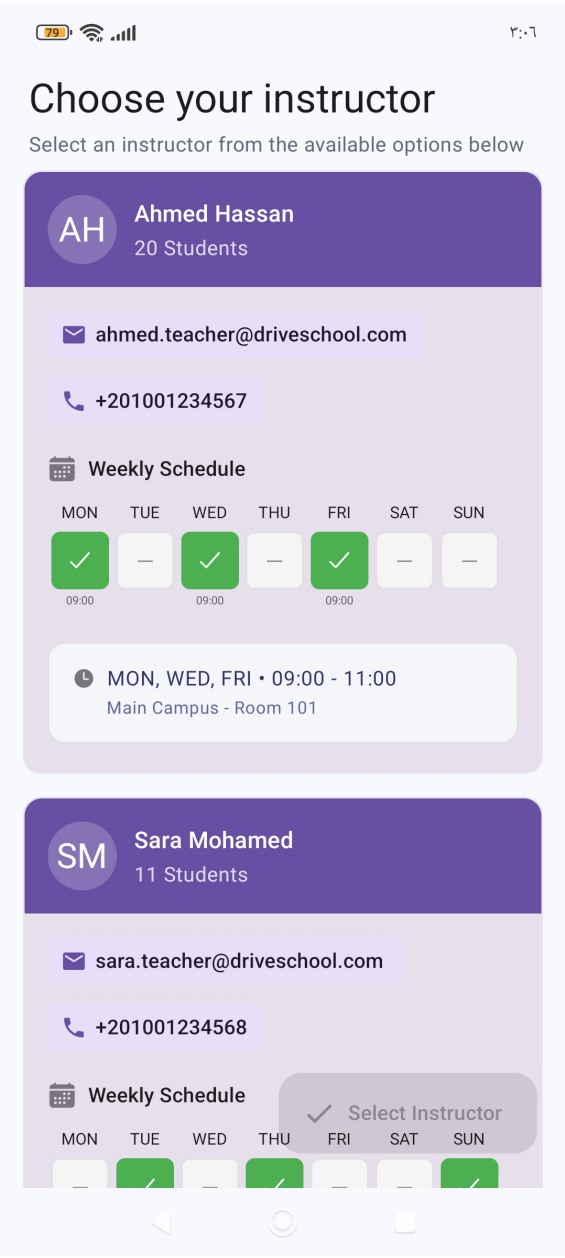

Figure 50: Choose Theoretical Instructor - Mobile

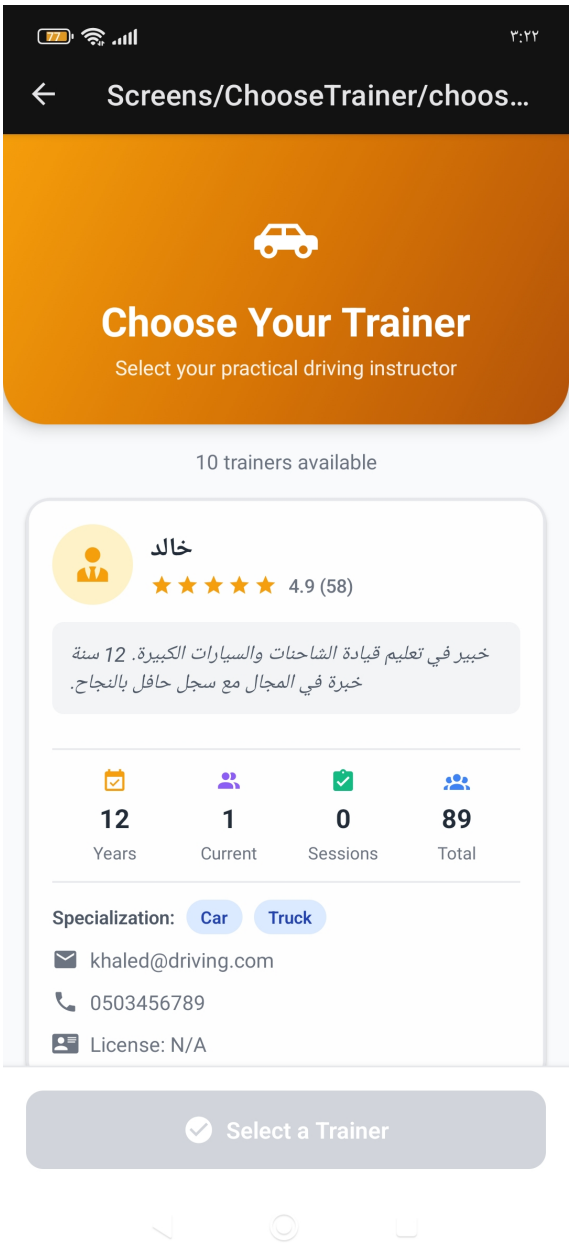

Figure 51: Choose Practical Trainer - Mobile

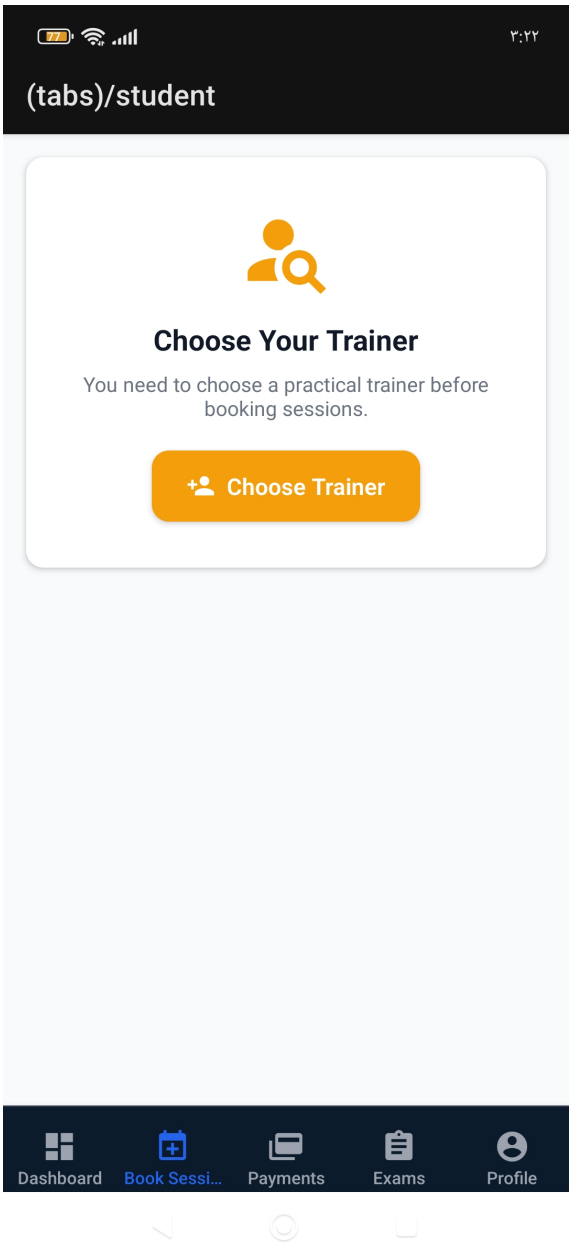

Figure 52: Trainer Selection Interface - Mobile

7.2 Student Dashboard

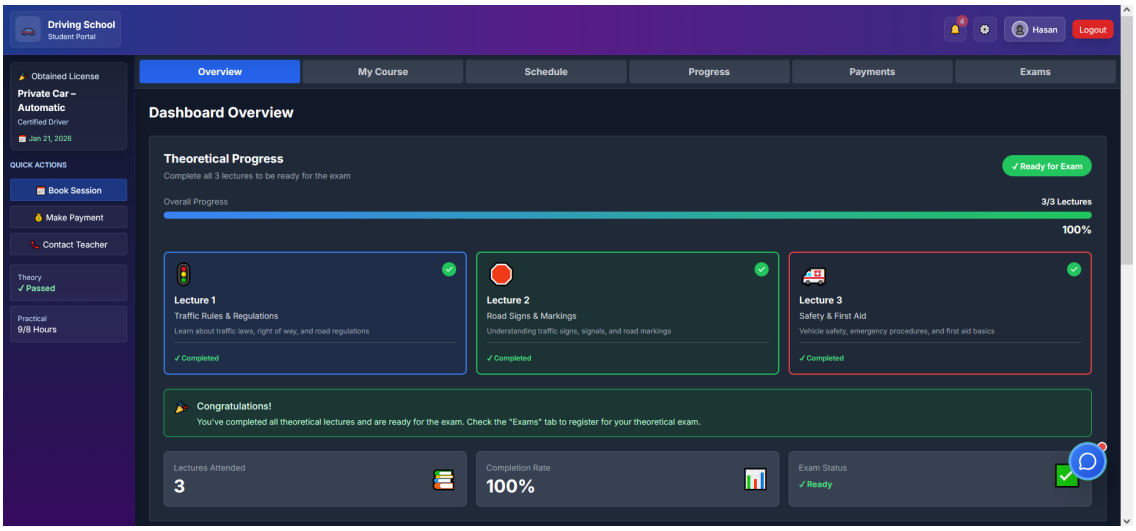

Figure 53: Student Dashboard - Overview (Web)

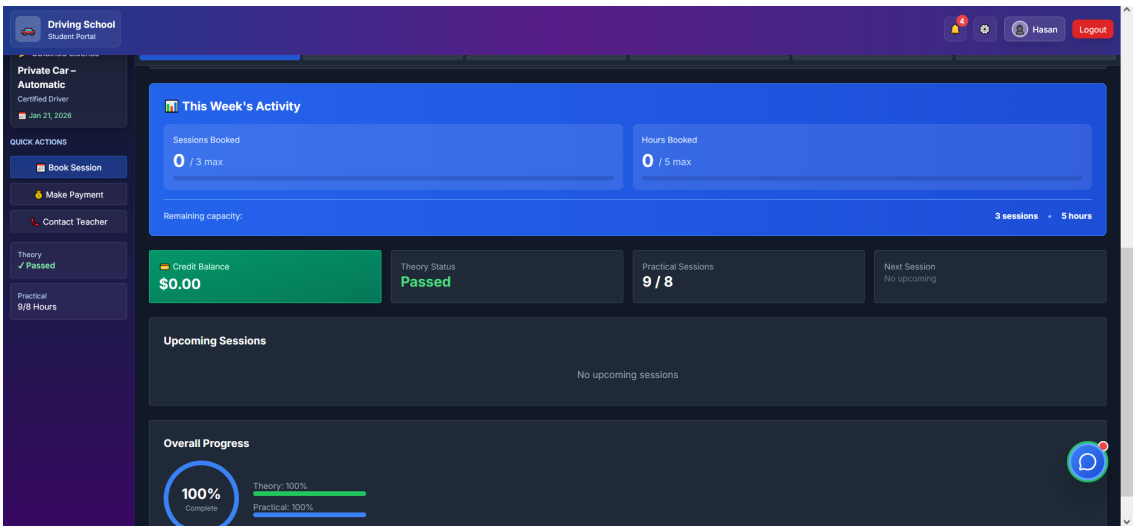

Figure 54: Student Dashboard - Progress View (Web)

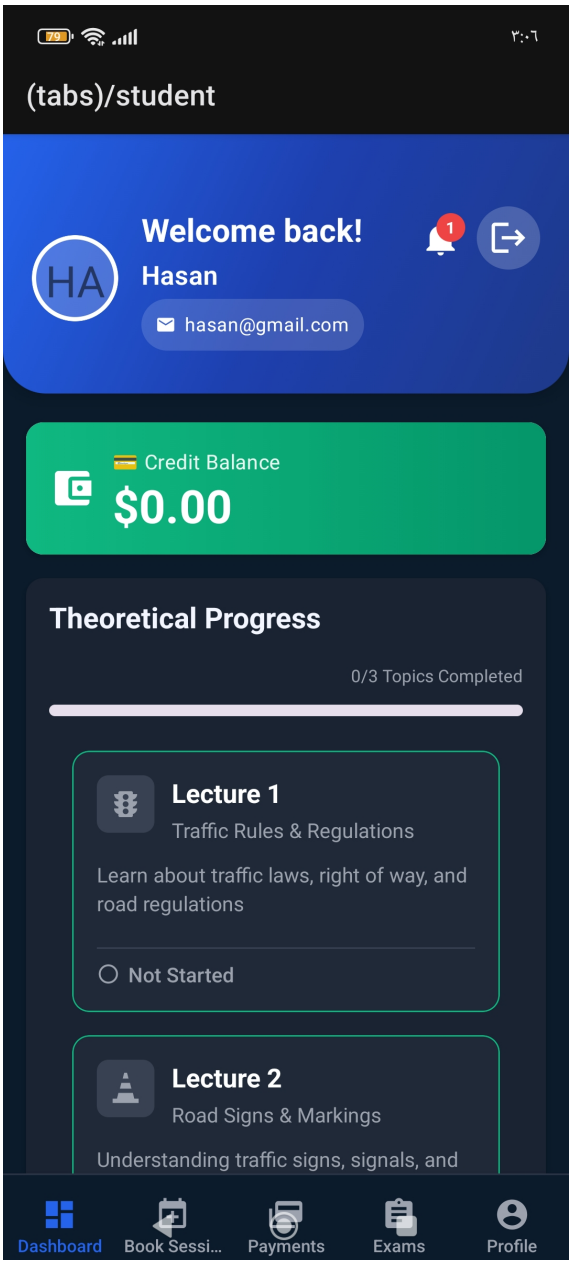

Figure 55: Student Dashboard - Mobile View

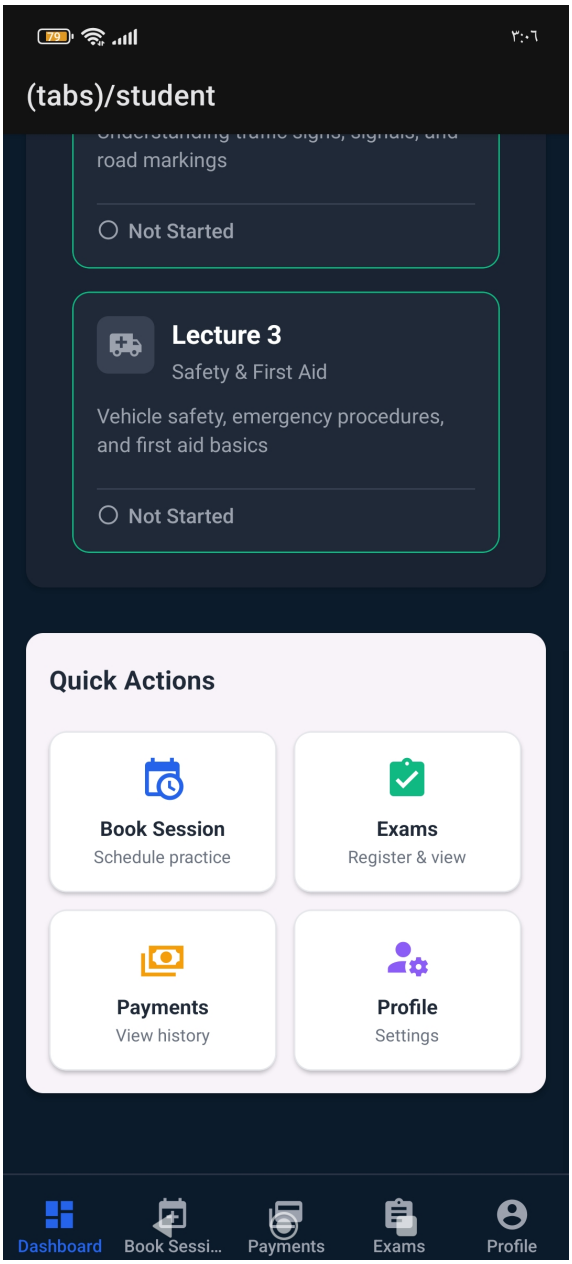

Figure 56: Student Dashboard - Progress Overview (Mobile)

7.3 My Course and Progress

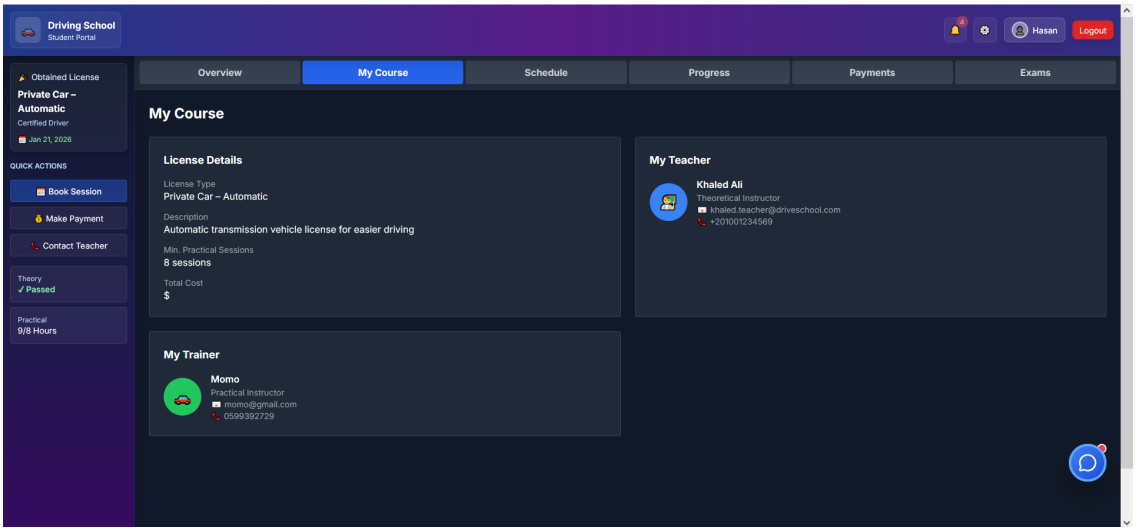

Figure 57: My Course - Web View

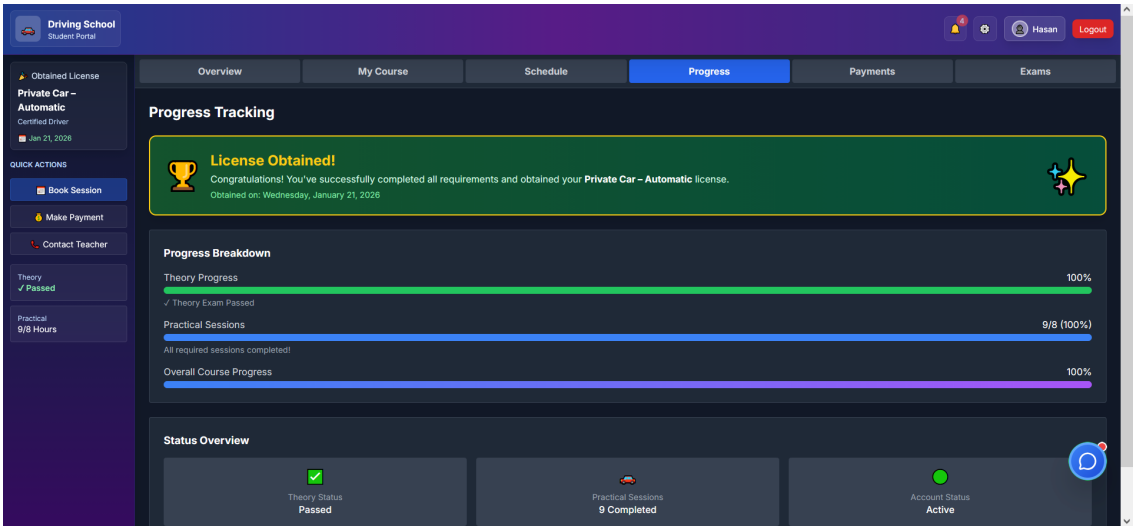

Figure 58: Progress Tracking - Web View

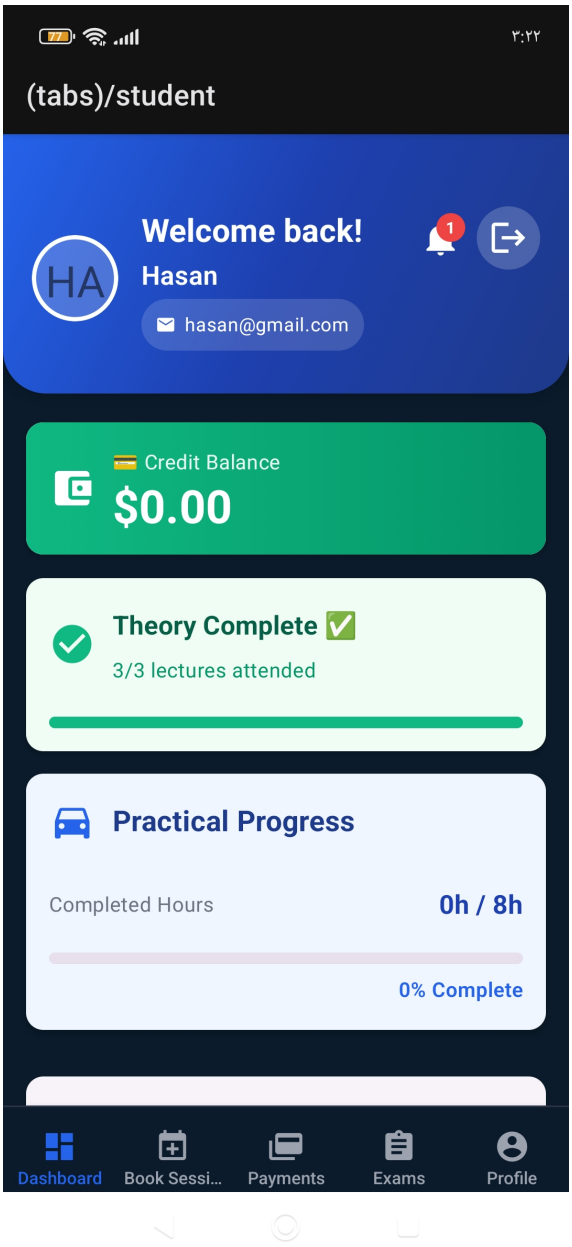

Figure 59: Theoretical Training Complete - Mobile

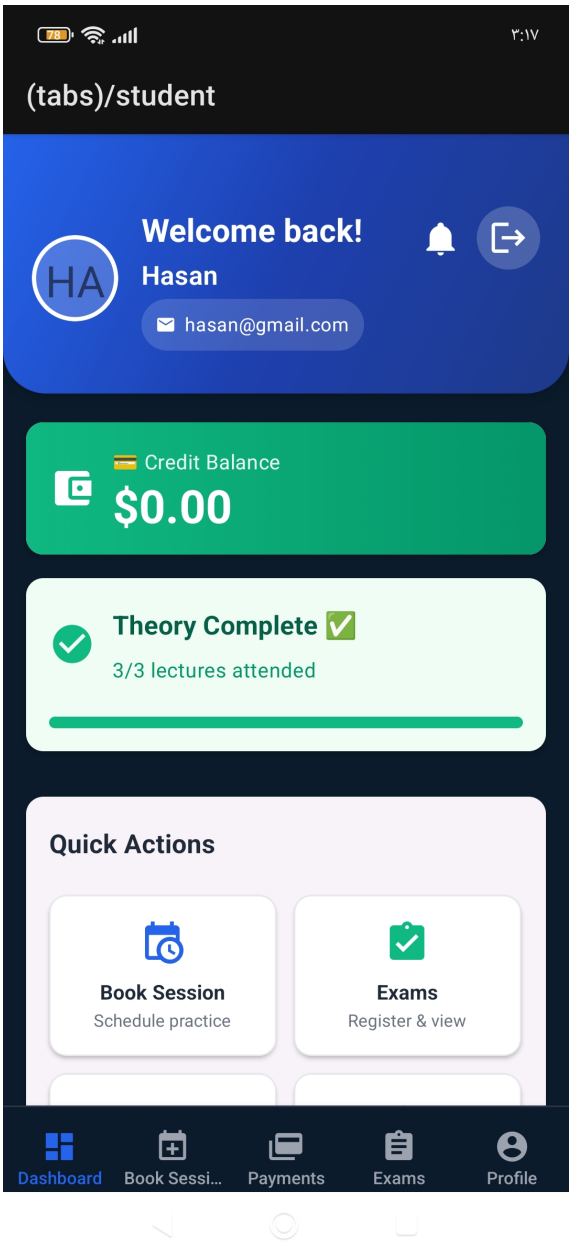

Figure 60: Theoretical Exam Passed - Mobile

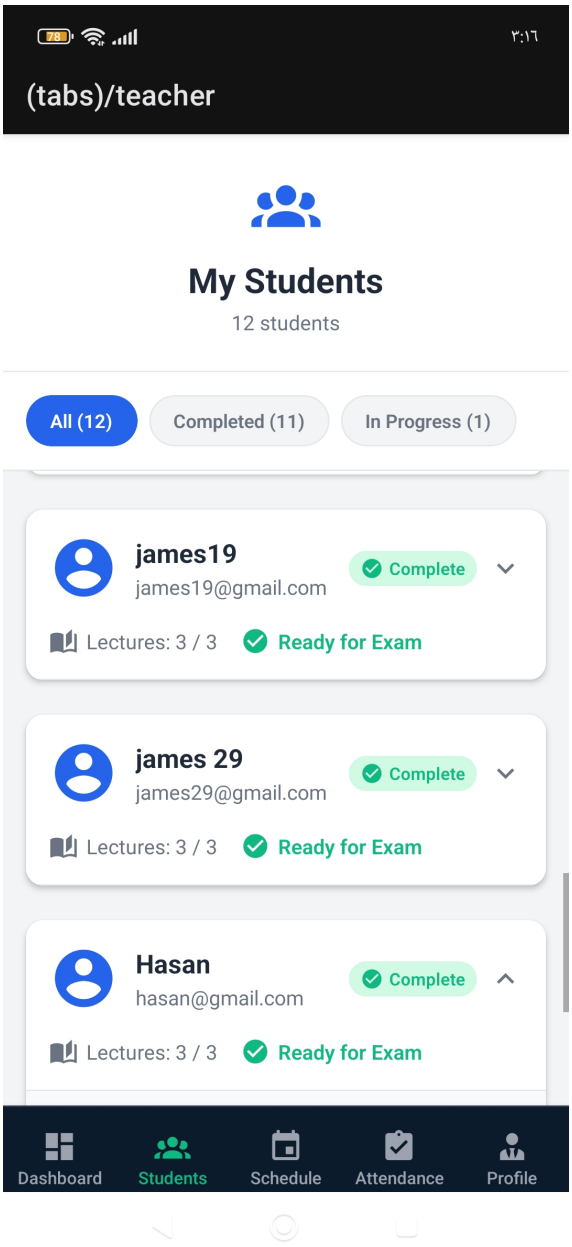

Figure 61: Ready for Theoretical Exam - Mobile

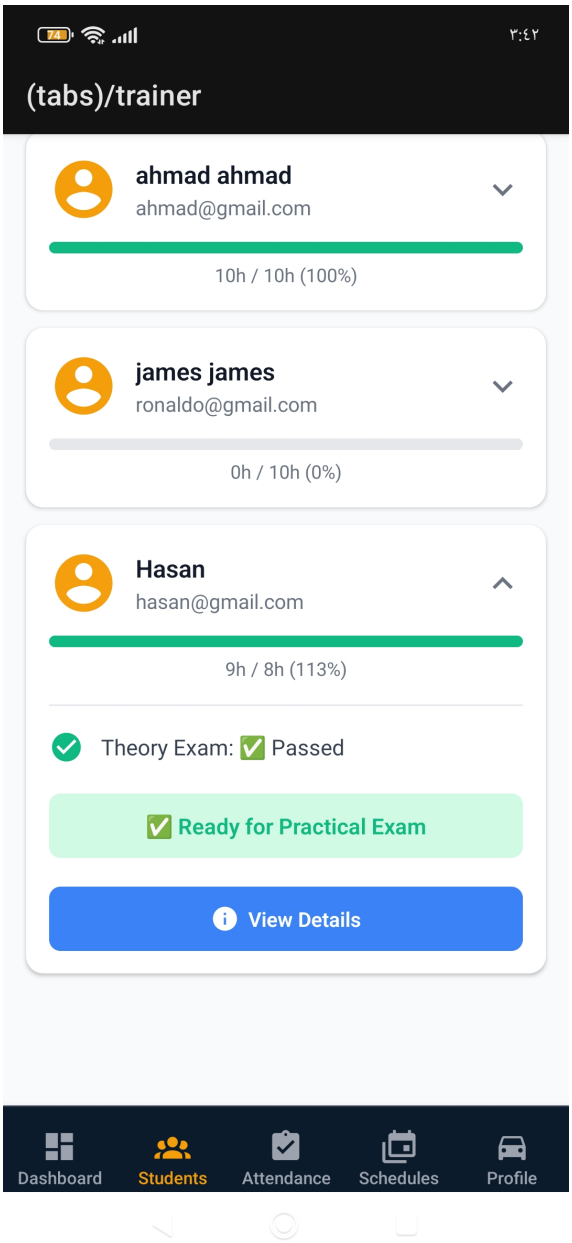

Figure 62: Ready for Practical Training - Mobile

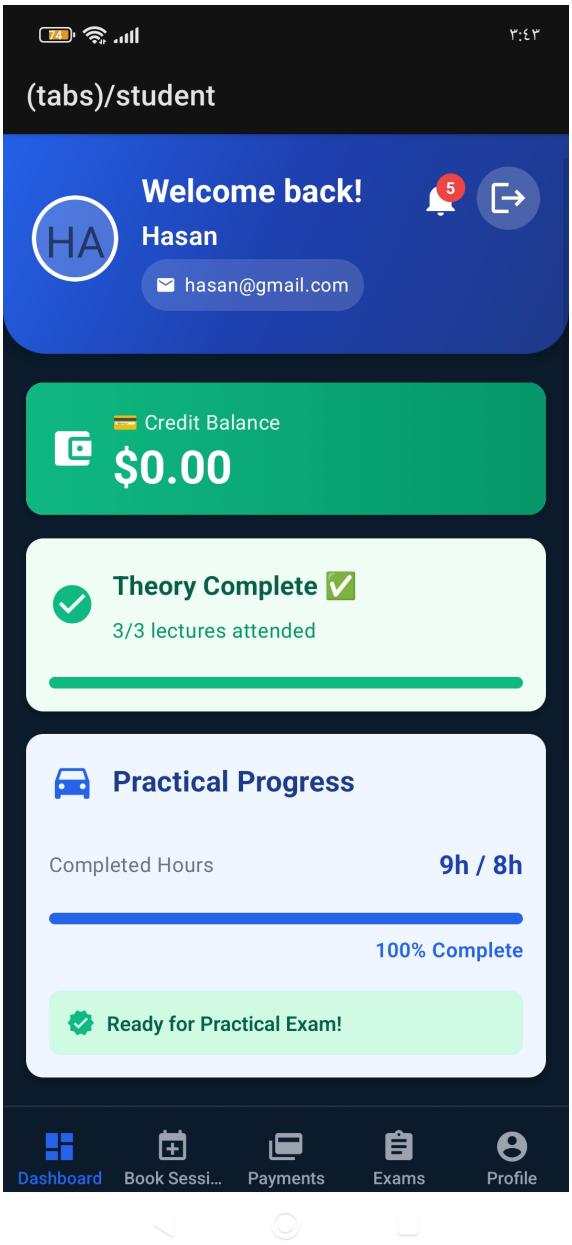

Figure 63: Ready for Practical Exam - Mobile

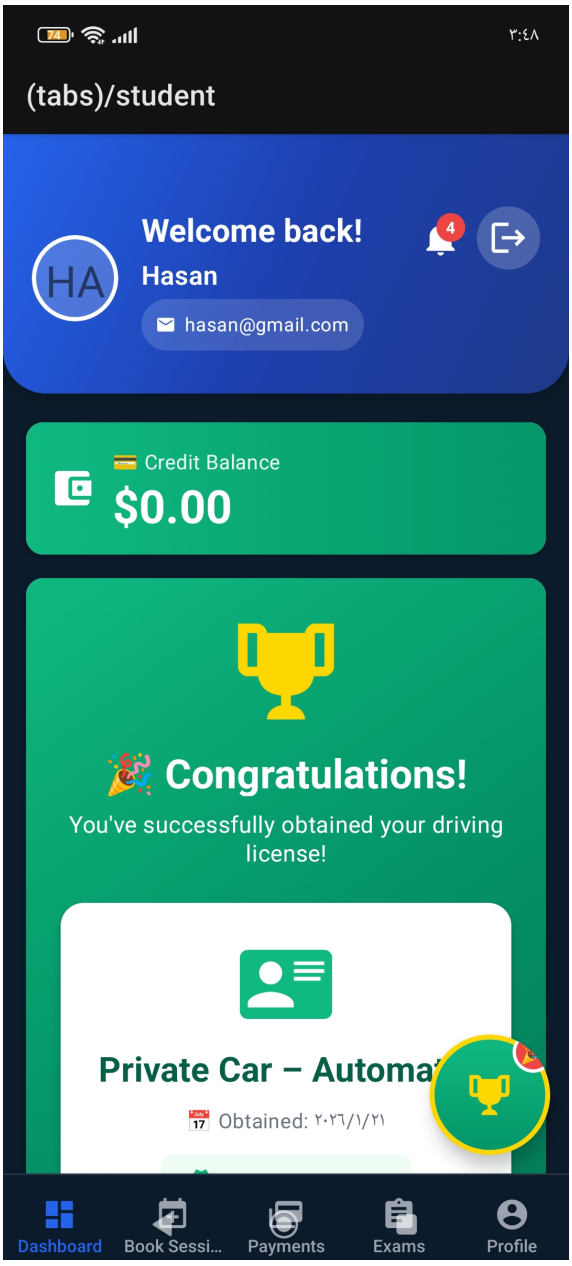

Figure 64: License Obtained - Congratulations Screen

7.4 Schedule and Booking System

7.4.1 Theoretical Schedule

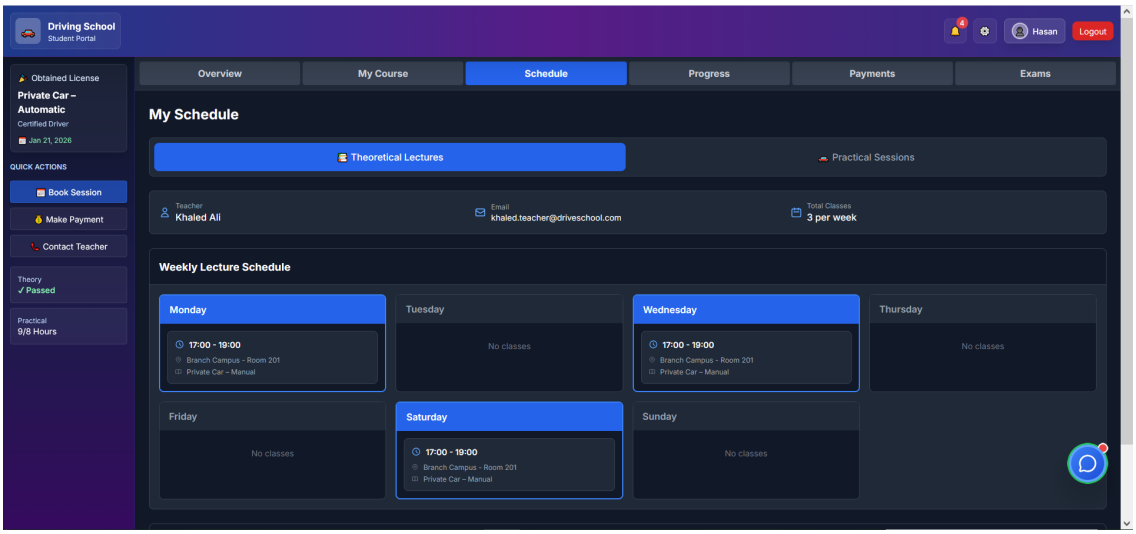

Figure 65: Theoretical Class Schedule - Web View

7.4.2 Practical Session Booking

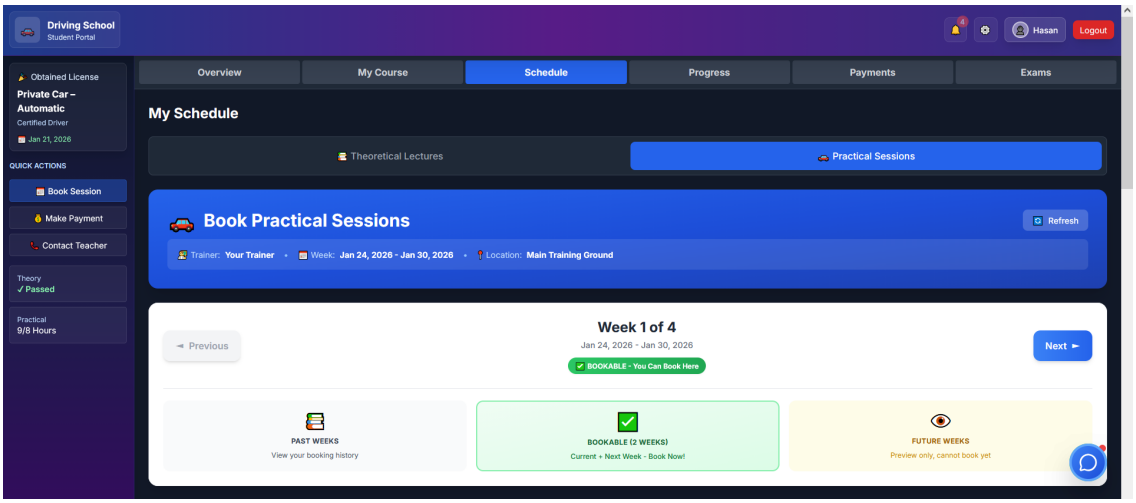

Figure 66: Available Practical Sessions - Web View

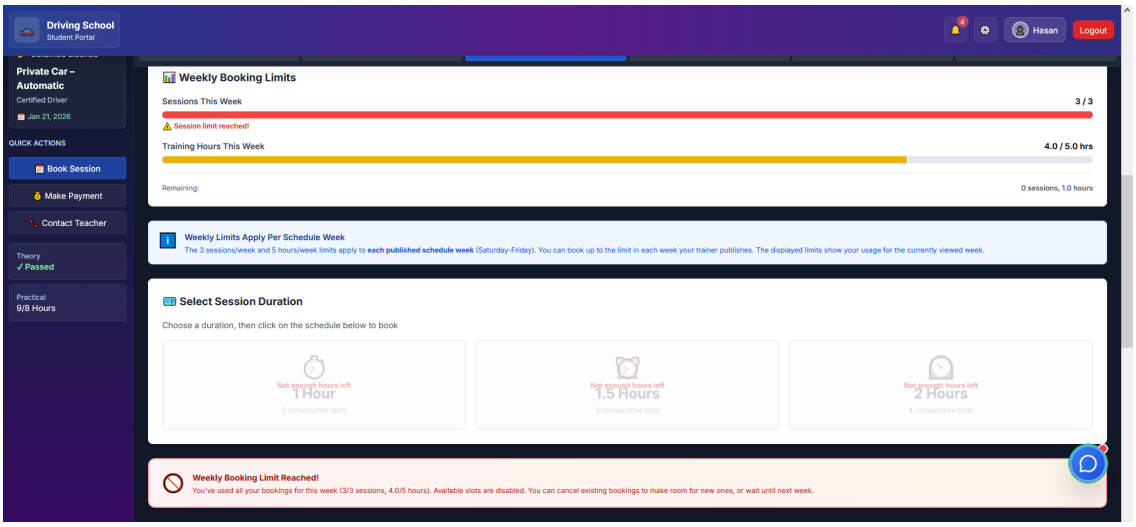

Figure 67: Session Booking Interface - Web View

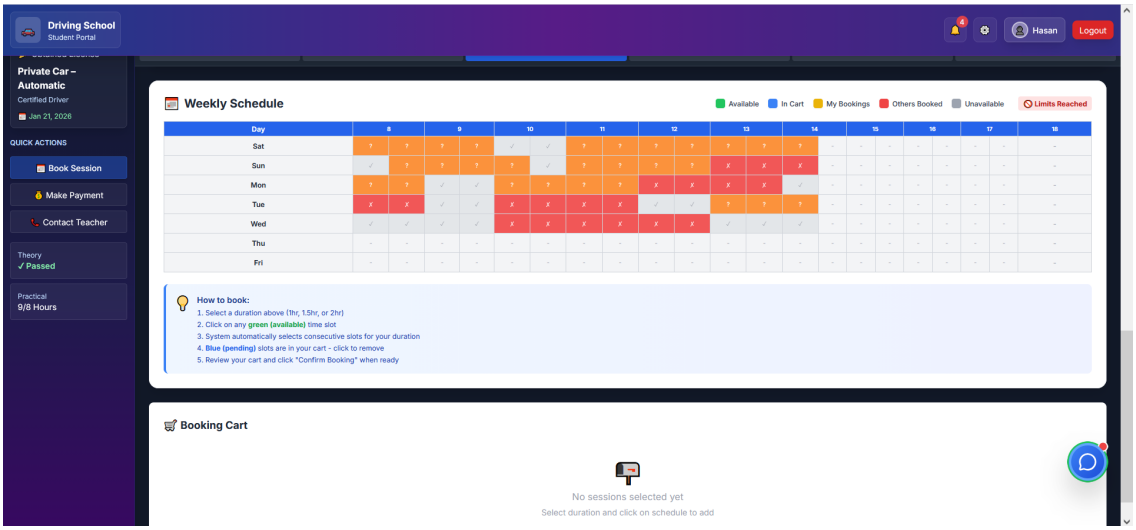

Figure 68: Booking Confirmation - Web View

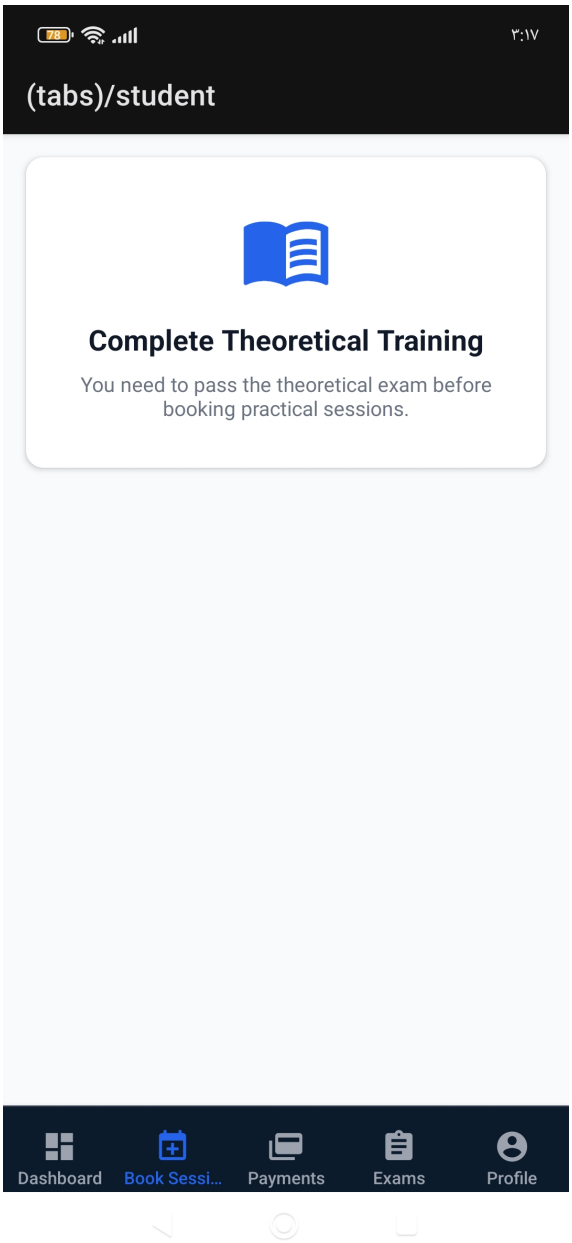

Figure 69: Book Session - Mobile Interface

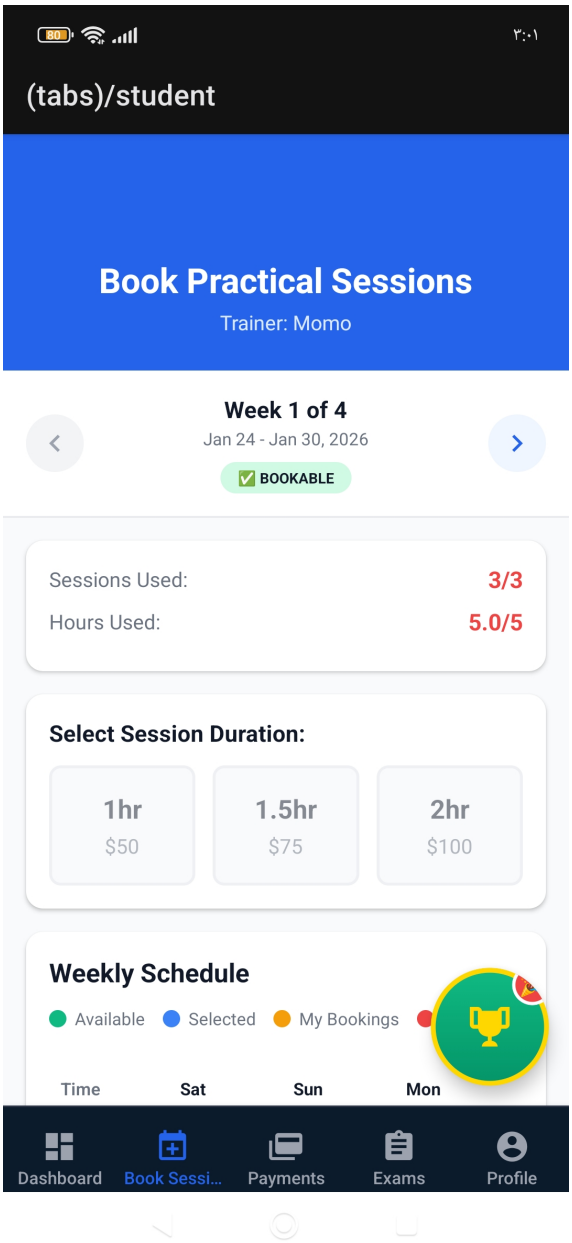

Figure 70: Session Booking Details - Mobile

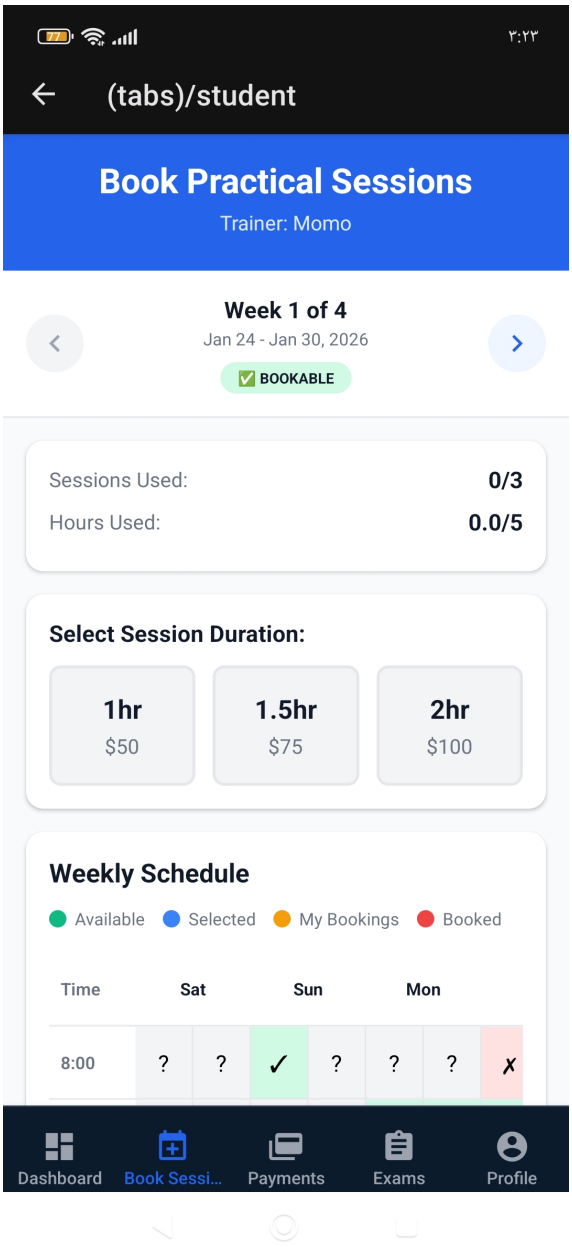

Figure 71: Book with Trainer (Step 1) - Mobile

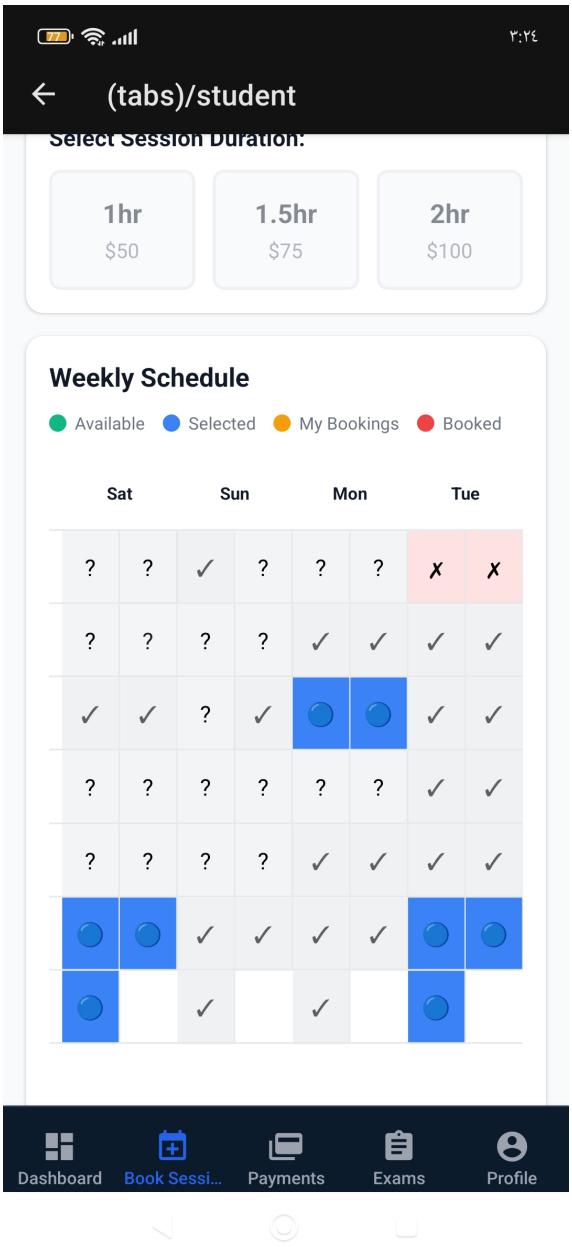

Figure 72: Book with Trainer (Step 2) - Mobile

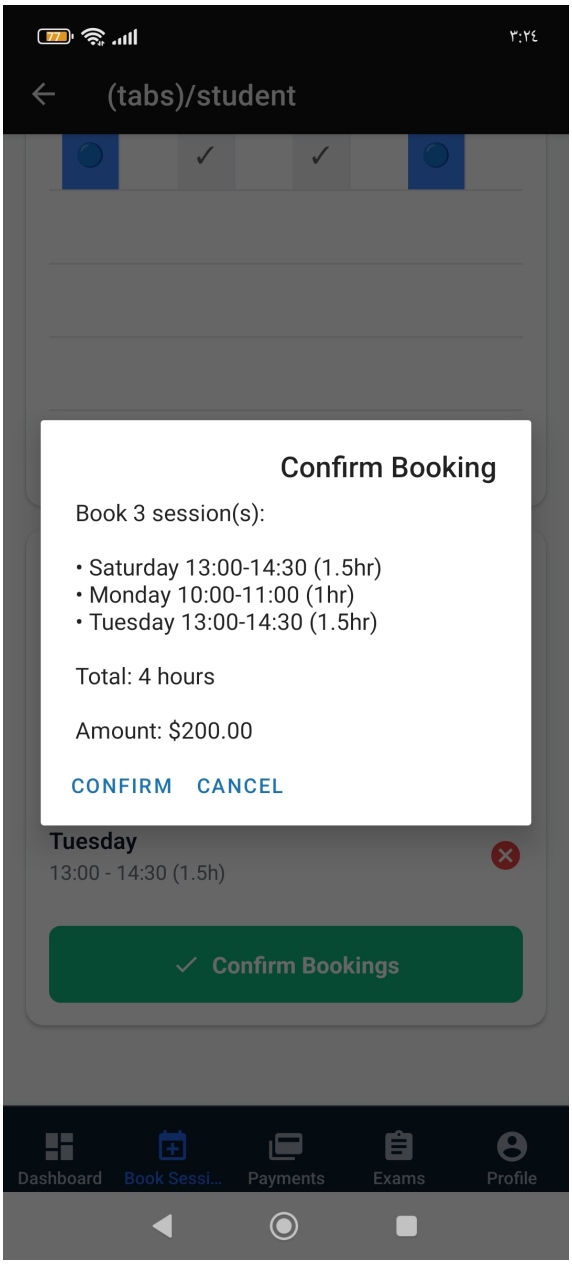

Figure 73: Confirm Booking - Mobile

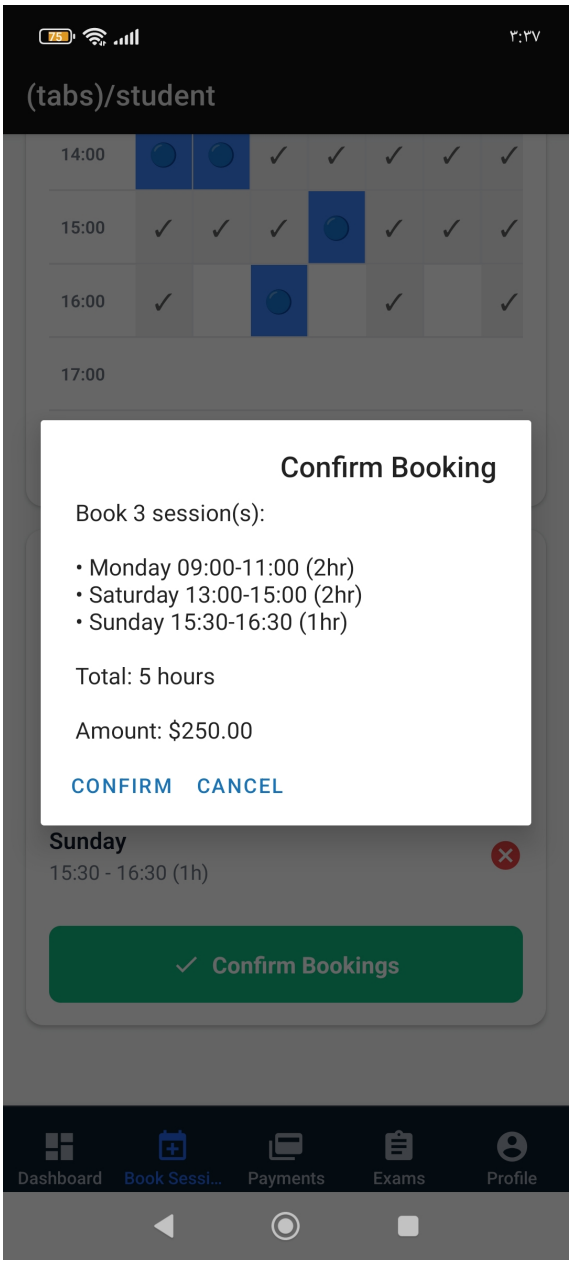

Figure 74: Booking Confirmation Screen - Mobile

7.5 Payment System

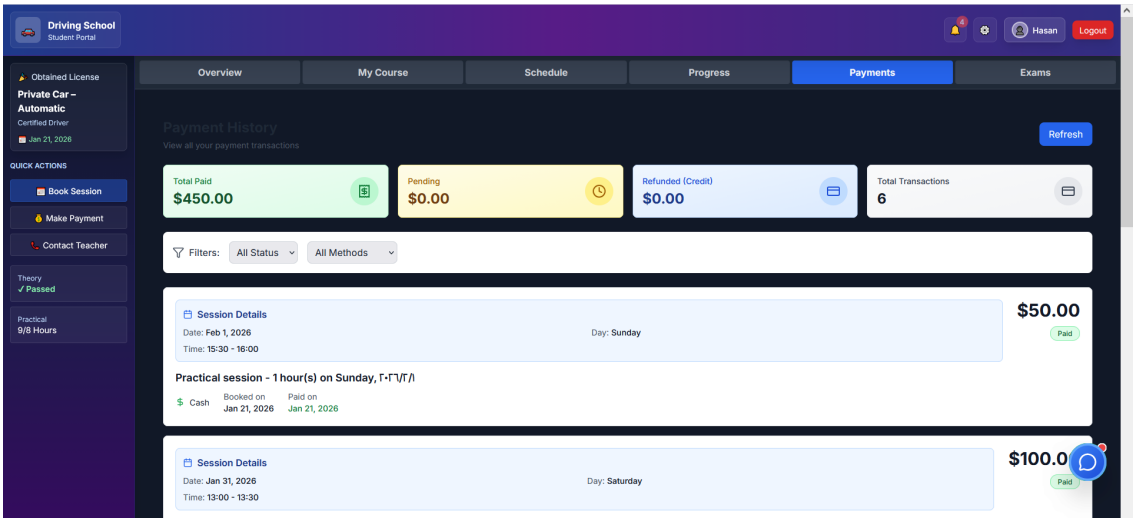

Figure 75: Payment Interface - Web View

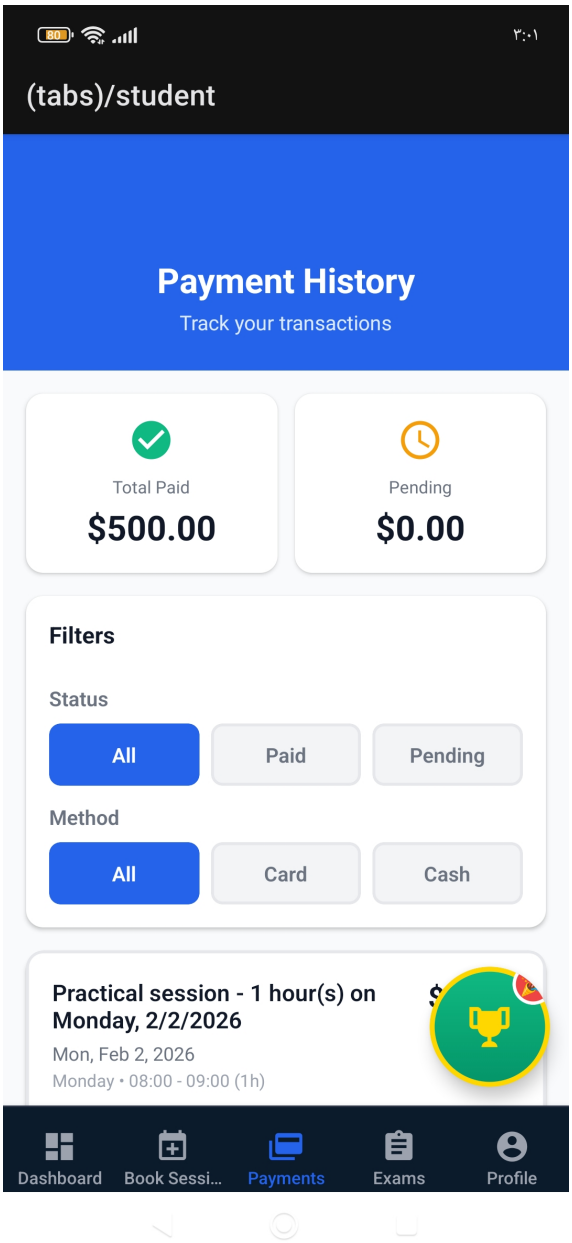

Figure 76: Payment Dashboard - Mobile

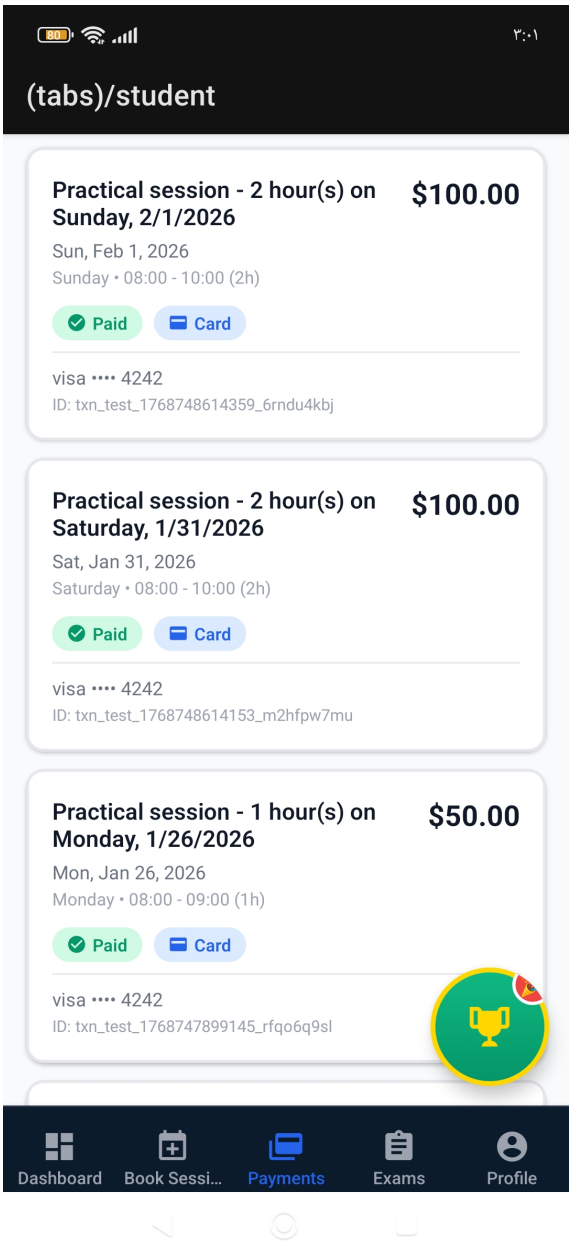

Figure 77: Payment History - Mobile

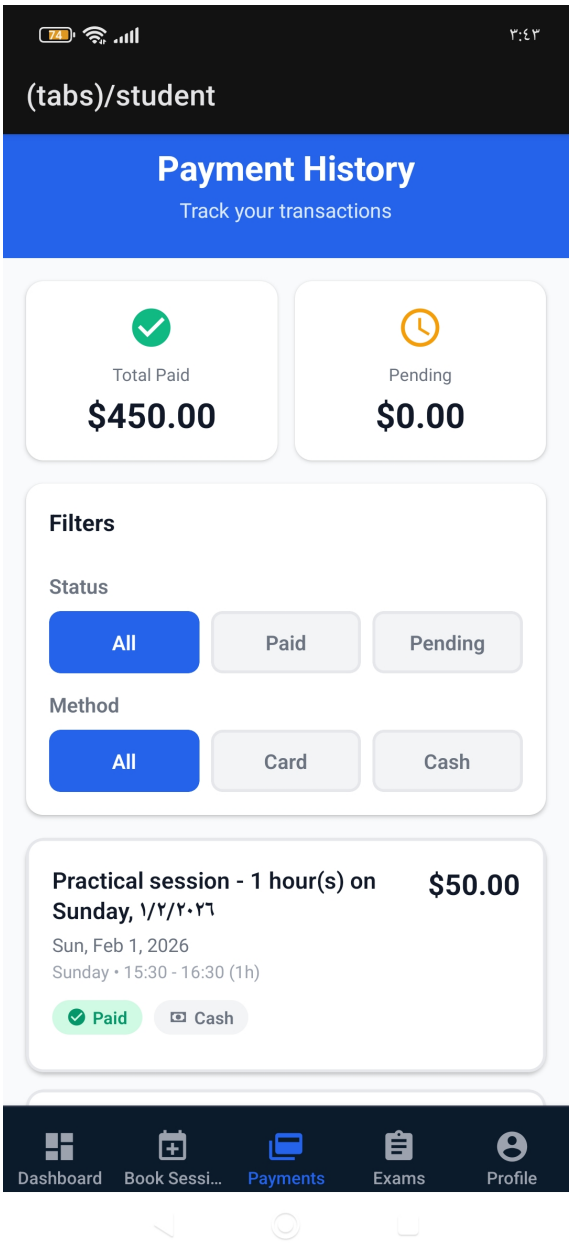

Figure 78: Transaction History - Mobile

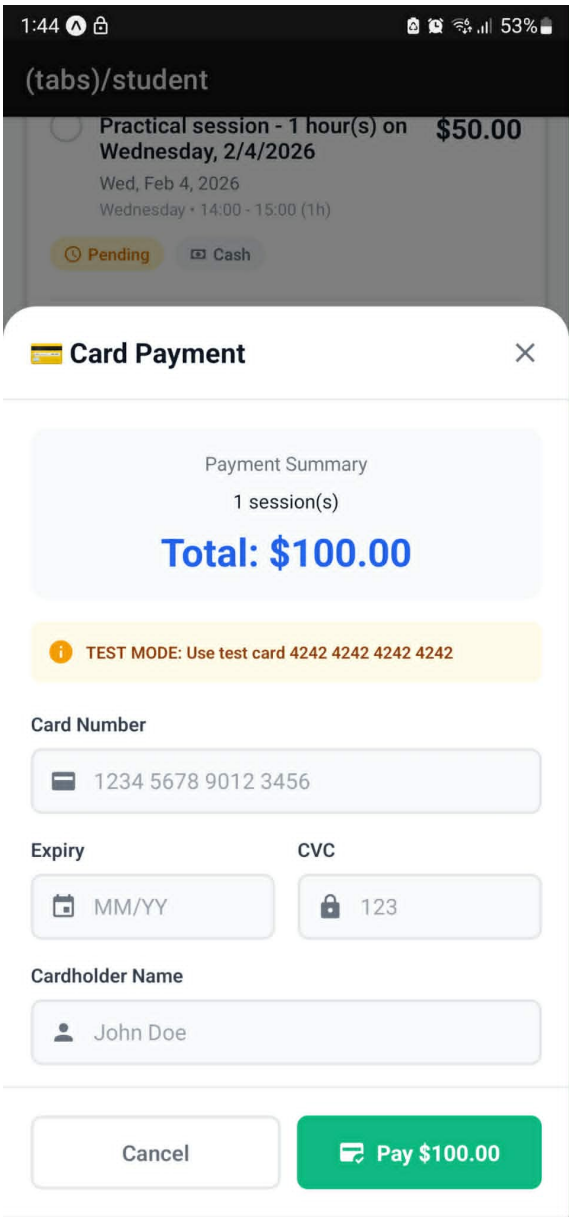

Figure 79: Card Payment Processing - Mobile

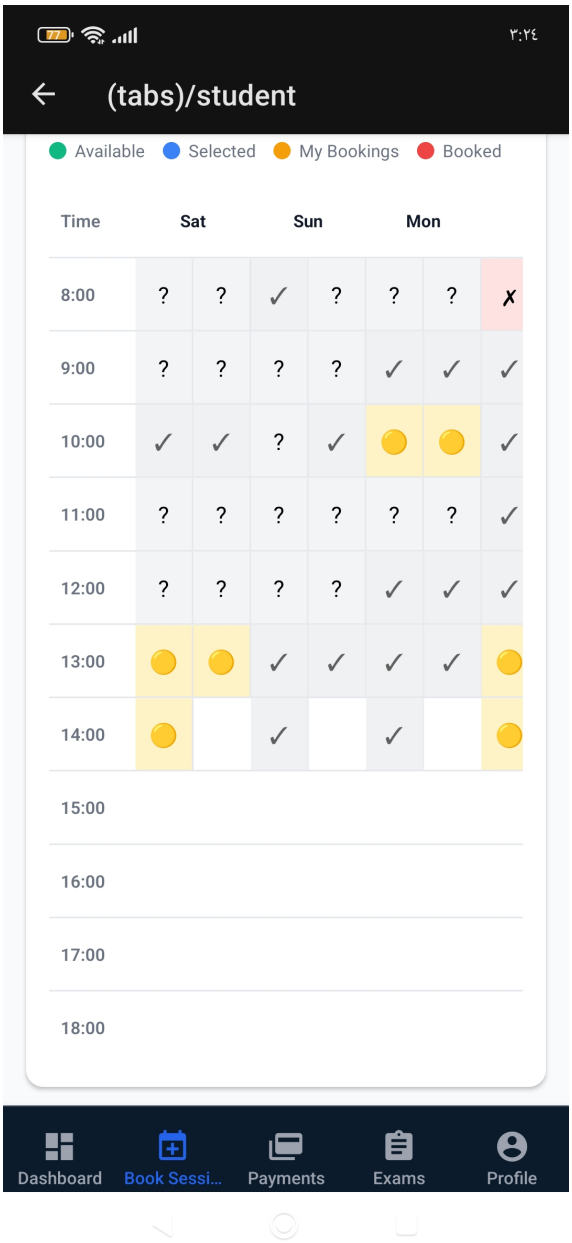

Figure 80: Post-Booking Confirmation - Mobile

7.6 Examination System

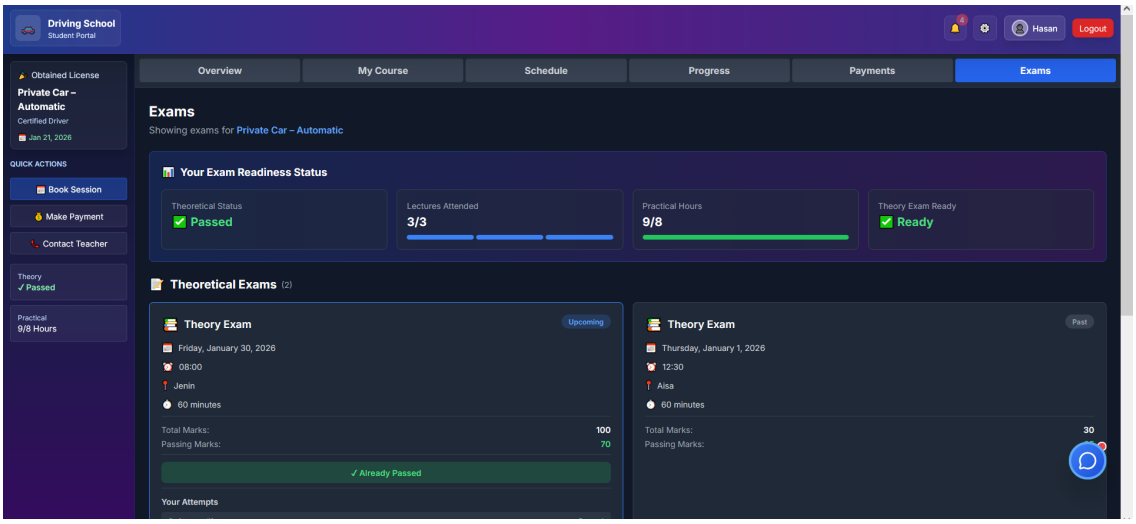

Figure 81: Exam Registration - Web View

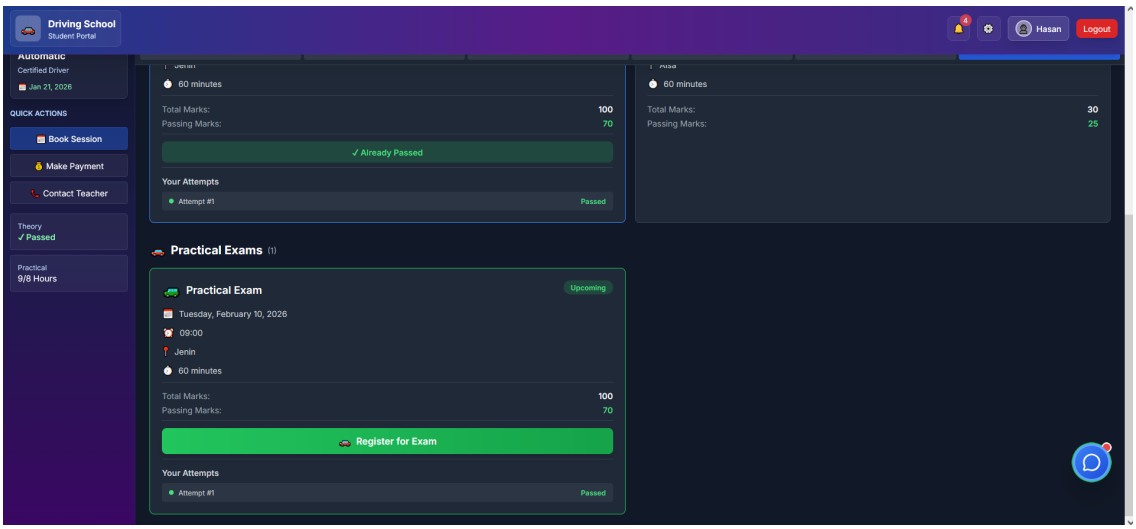

Figure 82: Exam Details and Results - Web View

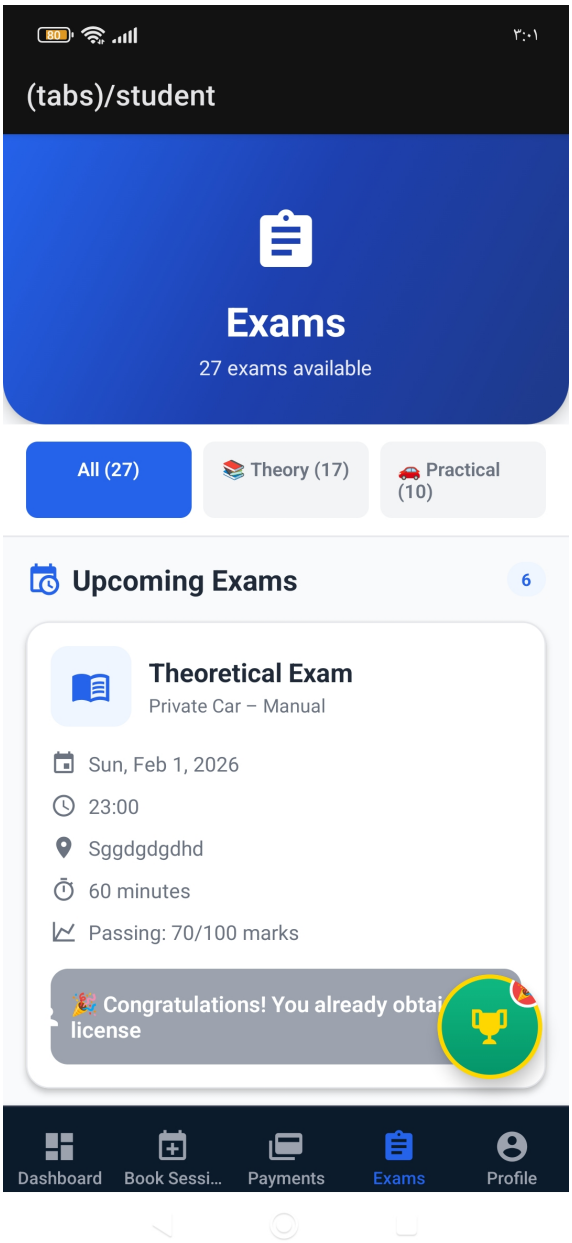

Figure 83: Exams Overview - Mobile

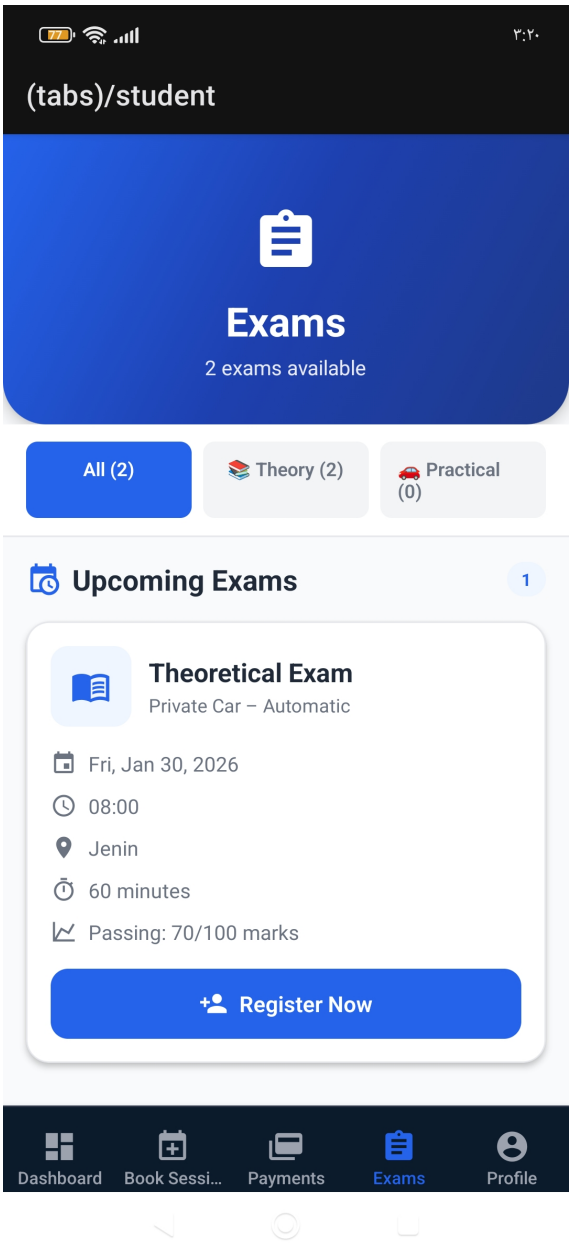

Figure 84: Theoretical Exam Registration - Mobile

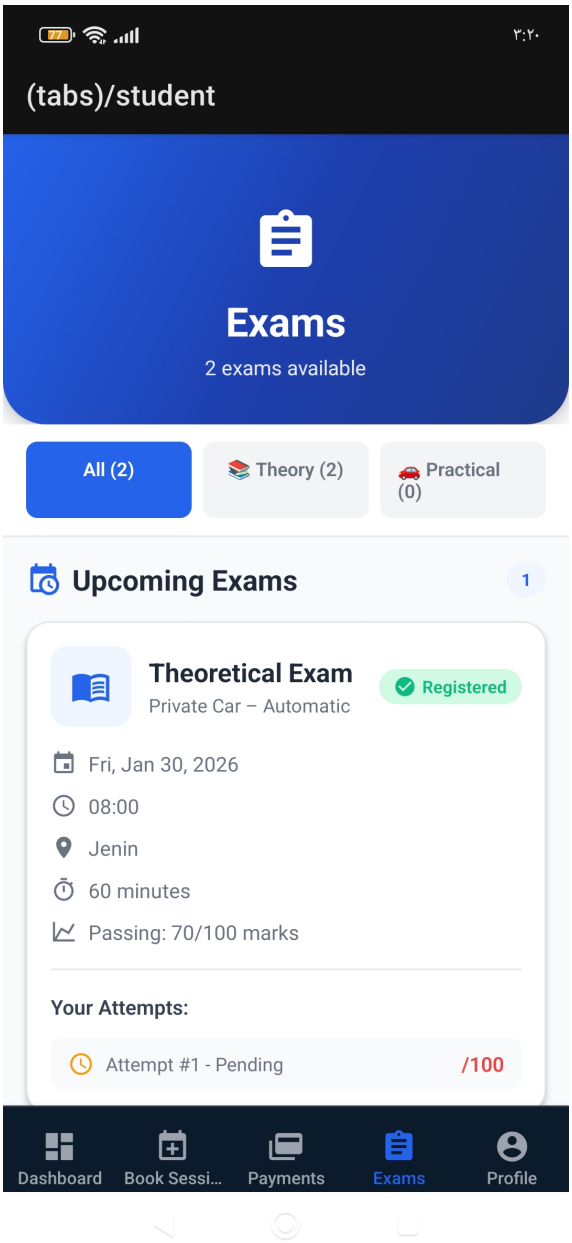

Figure 85: Register for Theoretical Exam - Mobile

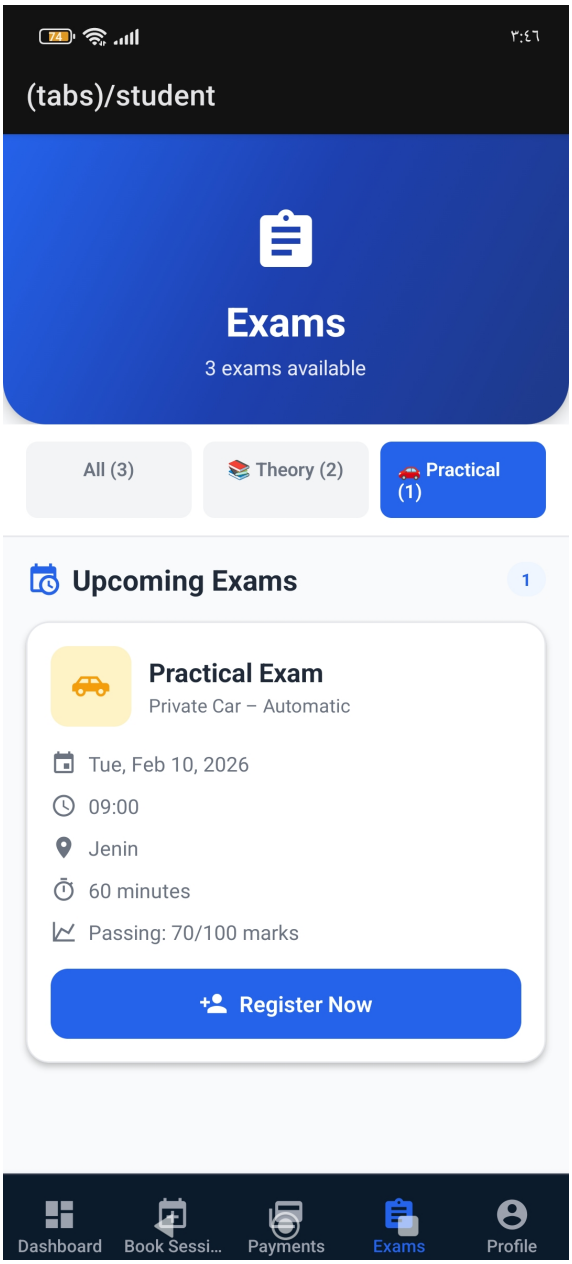

Figure 86: Practical Exam Registration - Mobile

7.7 Notifications and Profile

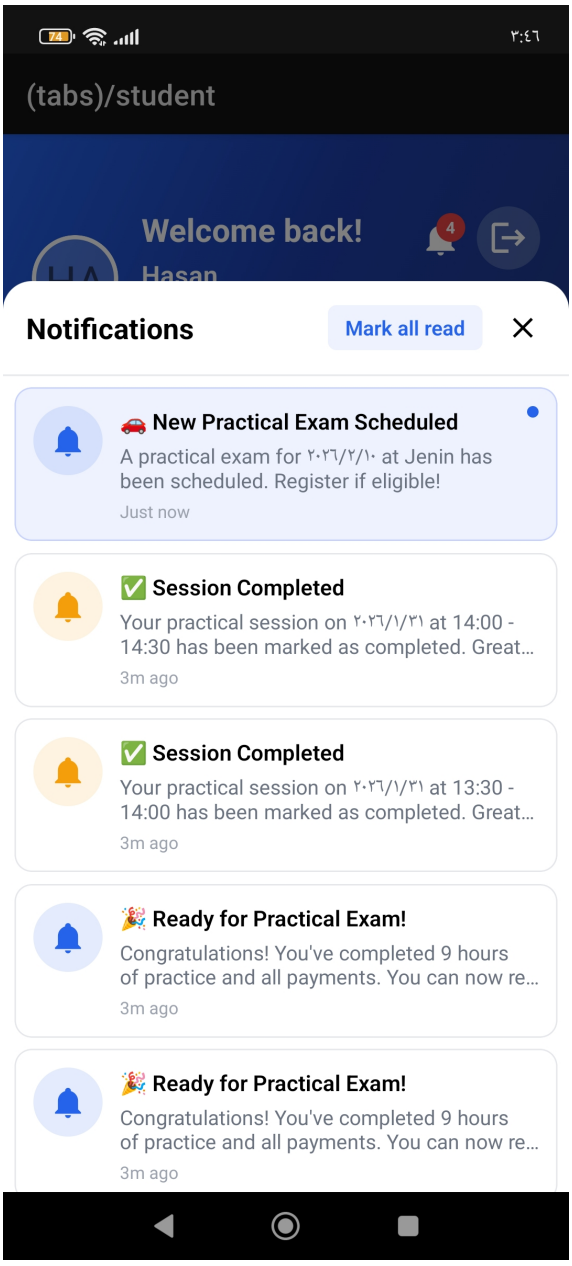

Figure 87: Notifications Center - Mobile

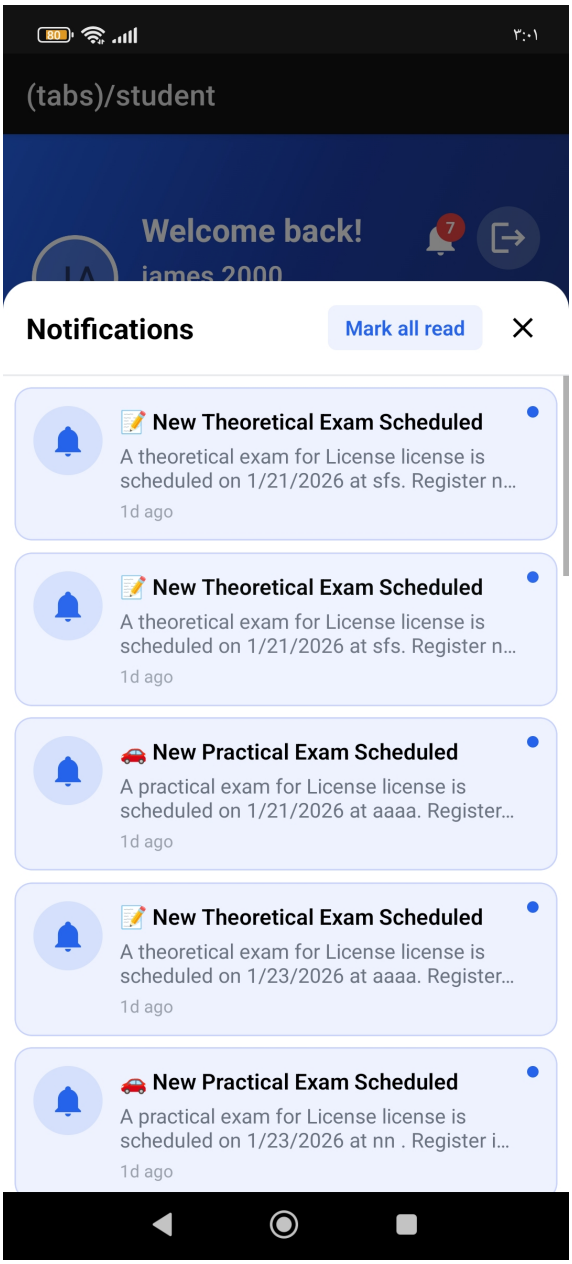

Figure 88: Notification Details - Mobile

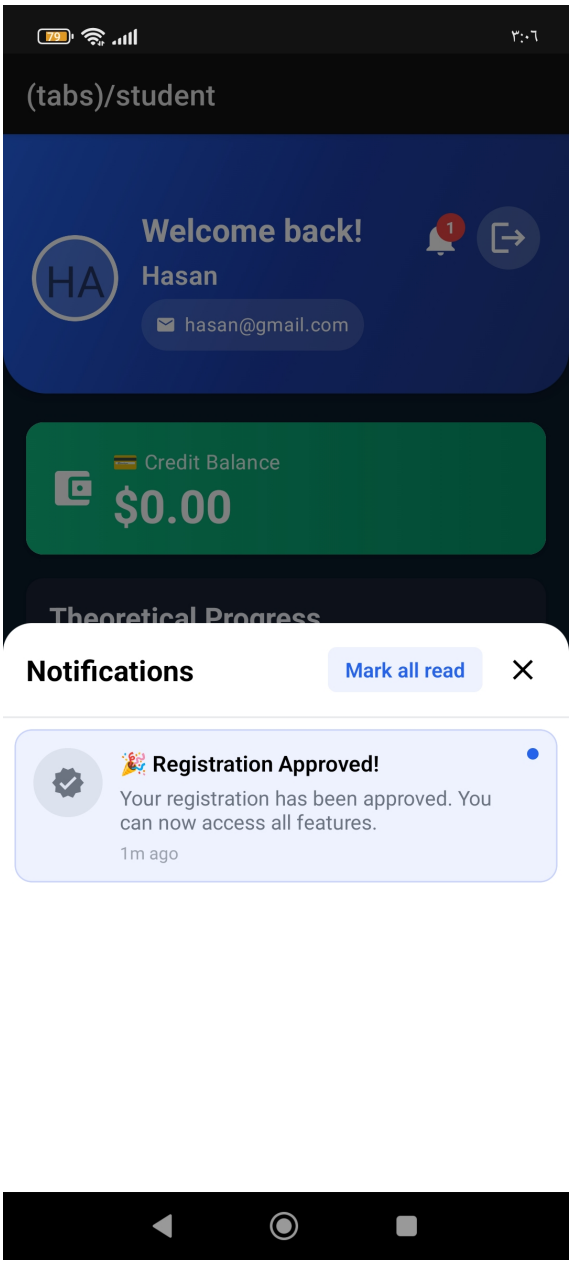

Figure 89: Approval Notification - Mobile

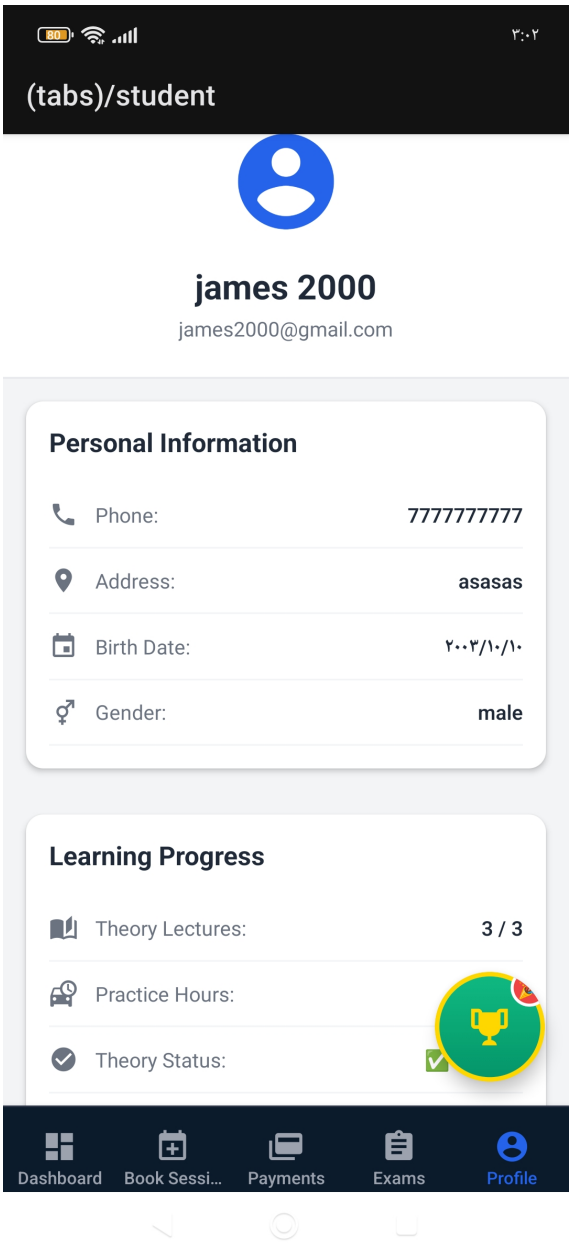

Figure 90: Student Profile - Mobile

8 Teacher Module

8.1 Teacher Dashboard

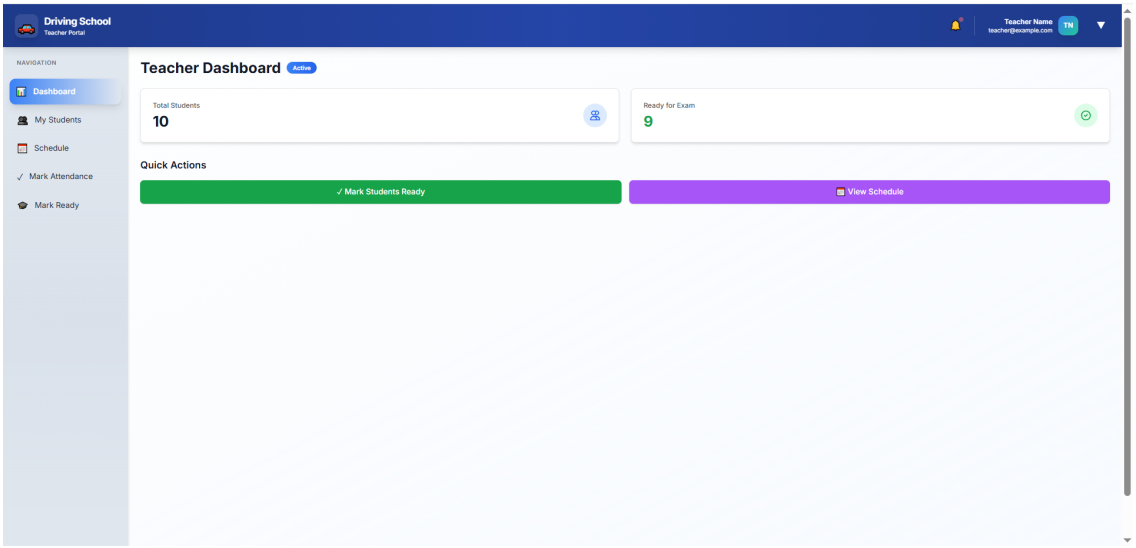

Figure 91: Teacher Dashboard - Web View

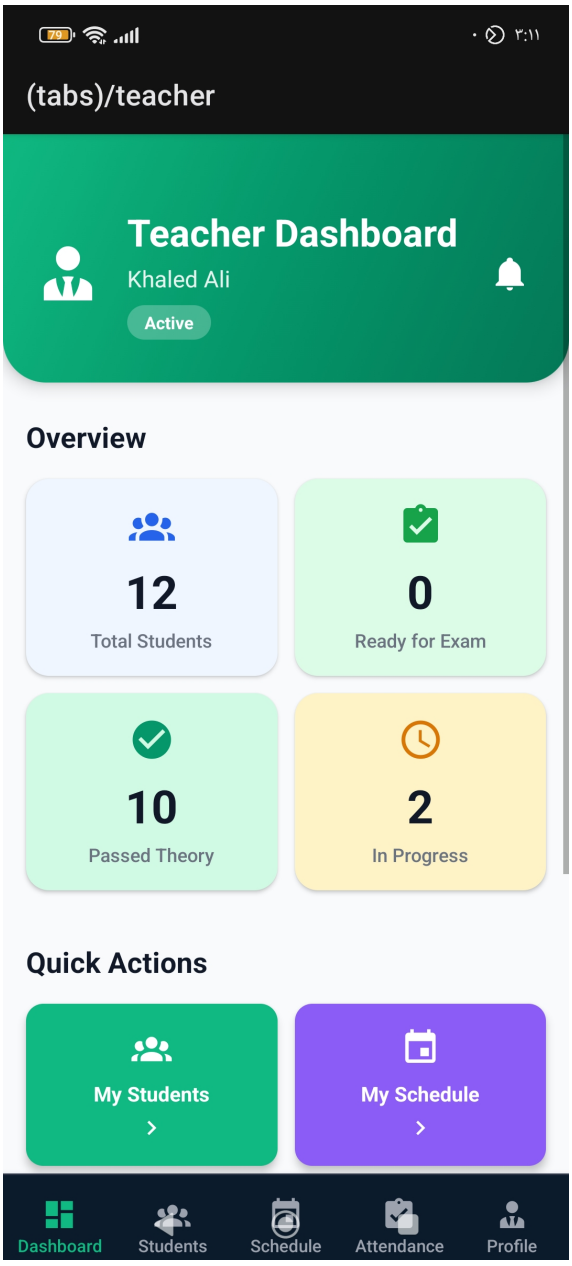

Figure 92: Teacher Dashboard - Mobile View

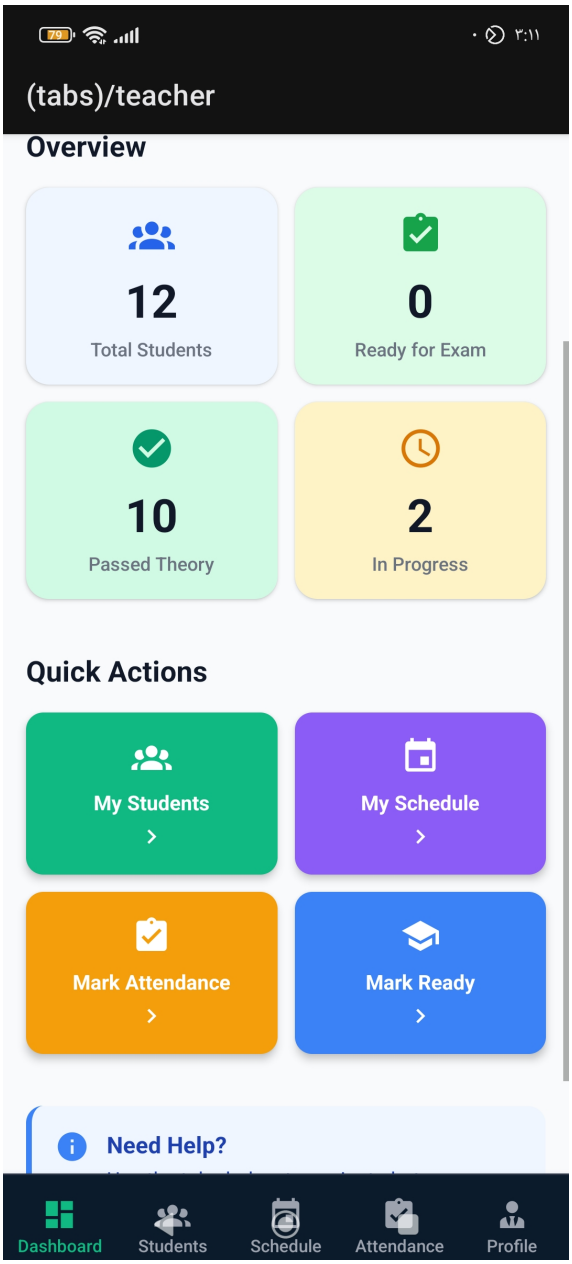

Figure 93: Teacher Dashboard - Statistics (Mobile)

8.2 My Students

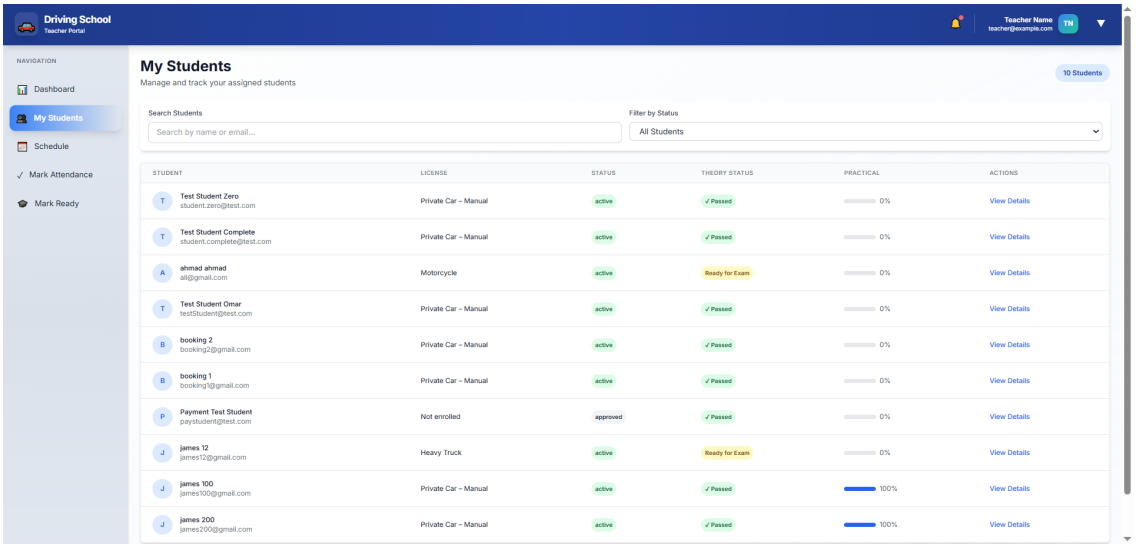

Figure 94: My Students List - Web View

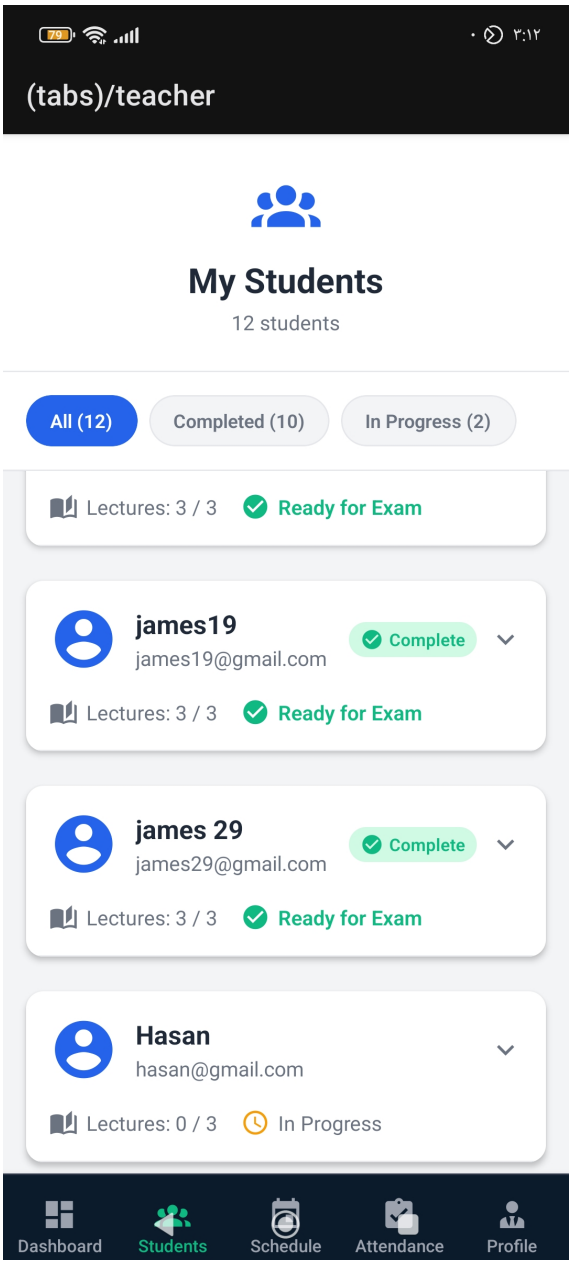

Figure 95: My Students - Mobile View

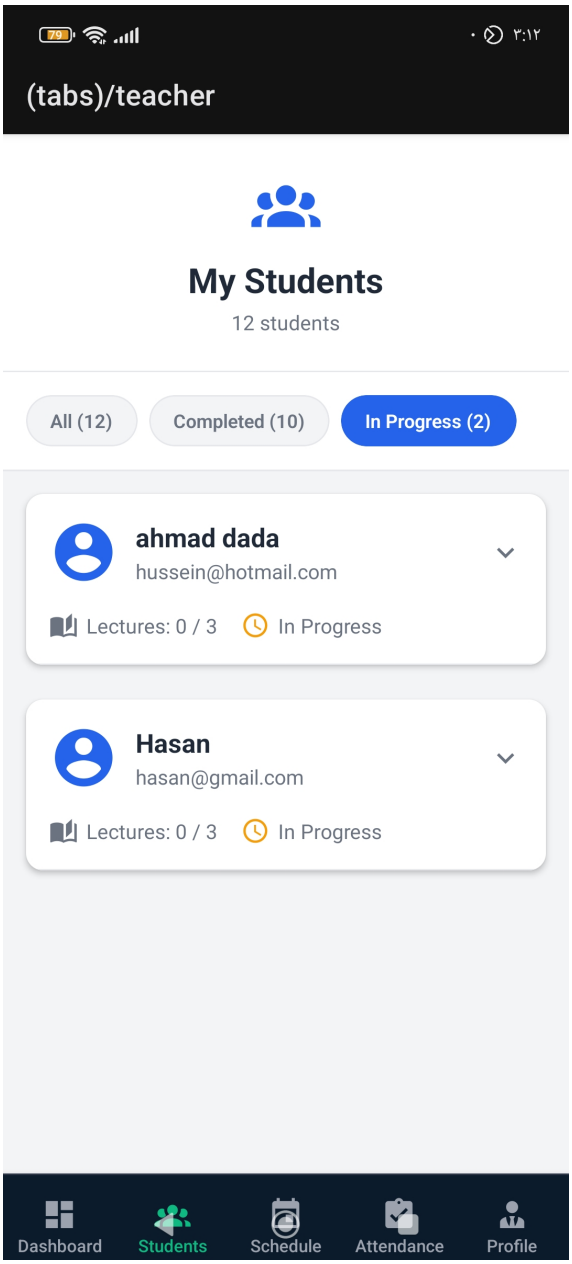

Figure 96: Students in Progress - Mobile

8.3 Schedule Management

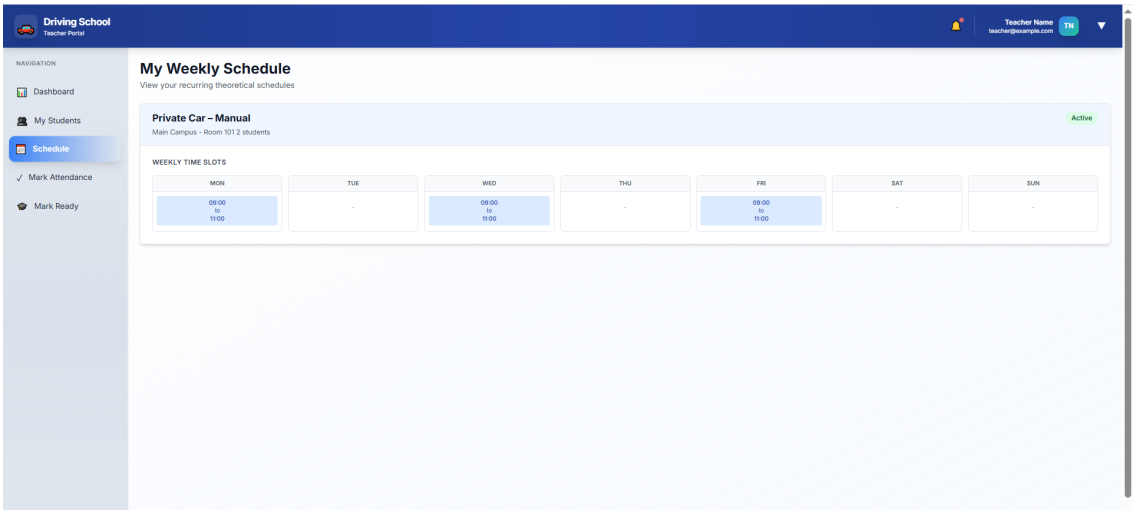

Figure 97: Class Schedule - Web View

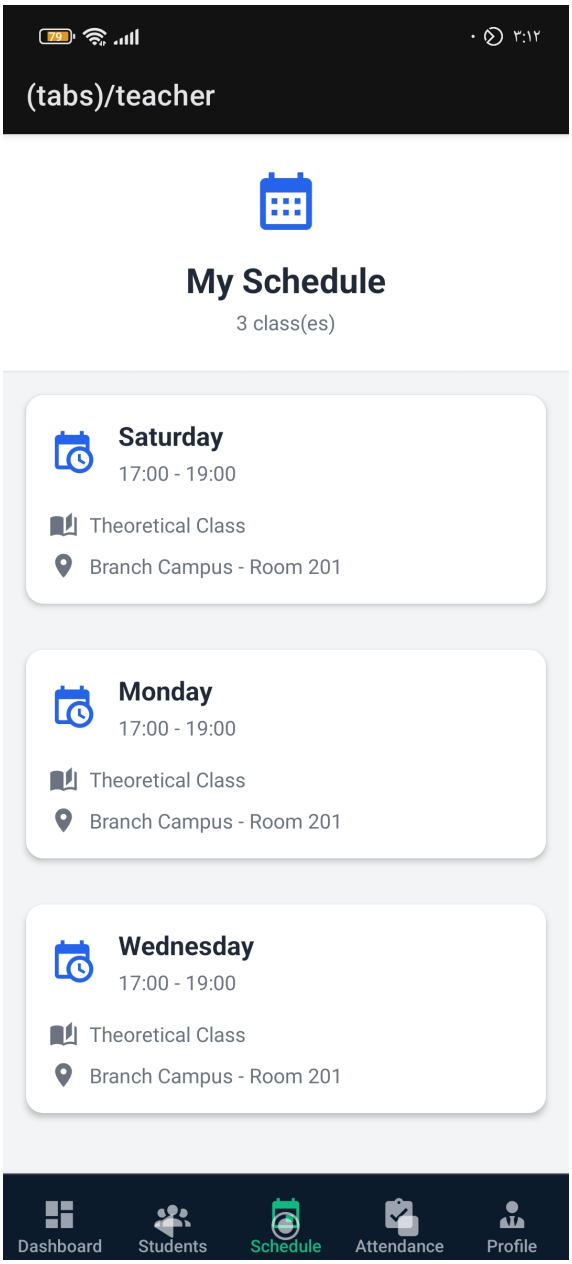

Figure 98: My Schedule - Mobile View

8.4 Attendance Management

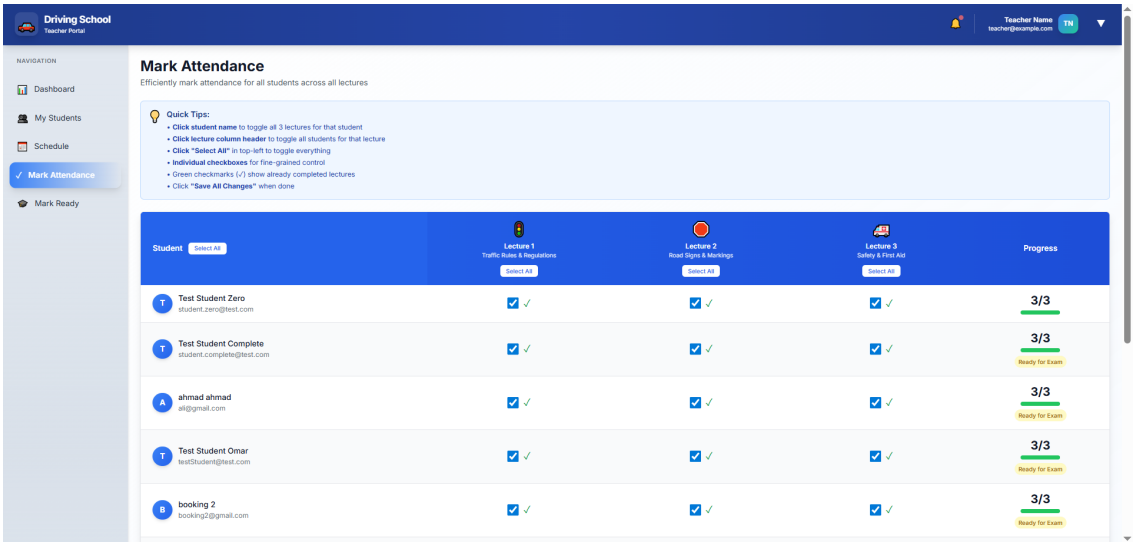

Figure 99: Mark Attendance - Web Interface

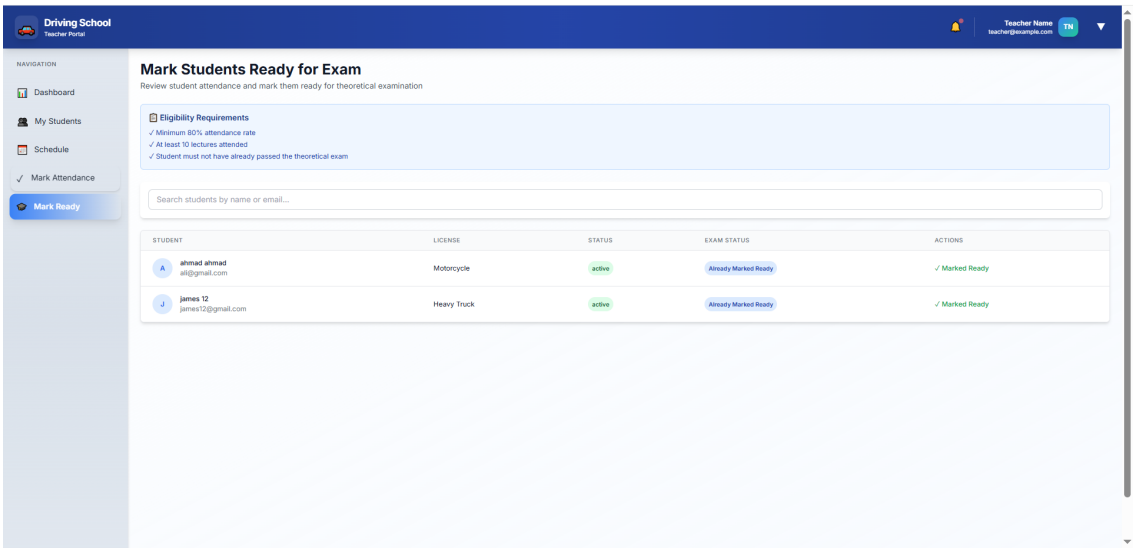

Figure 100: Mark Student Ready for Exam - Web

Figure 101: Attendance Tracking - Mobile

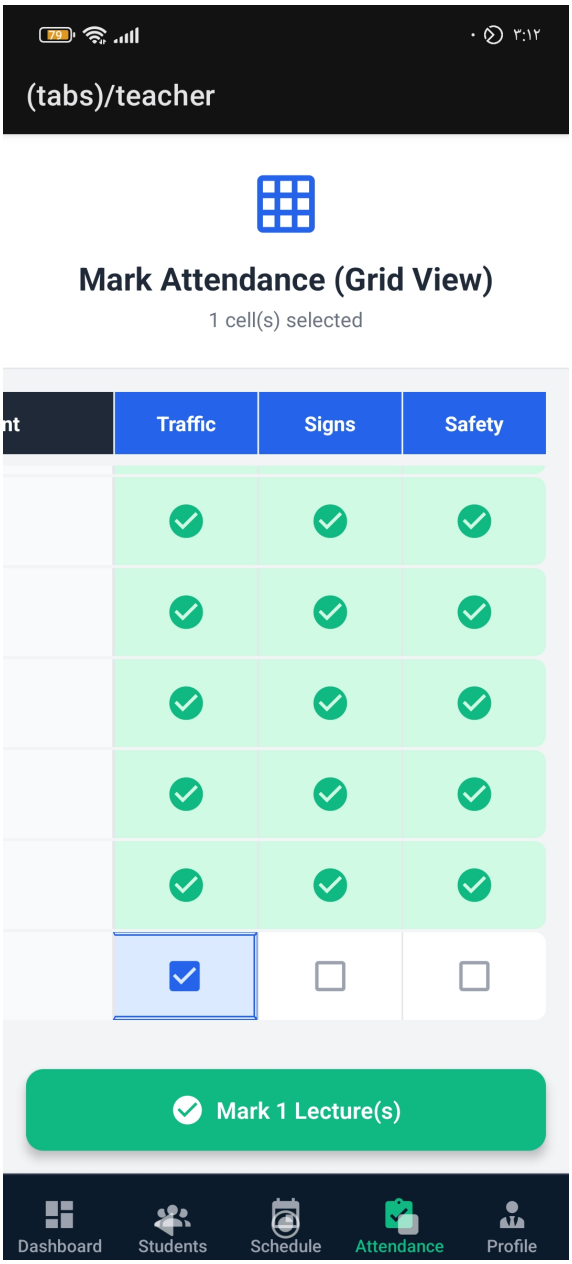

Figure 102: Mark Attendance Interface - Mobile

9 Trainer Module

9.1 Trainer Dashboard

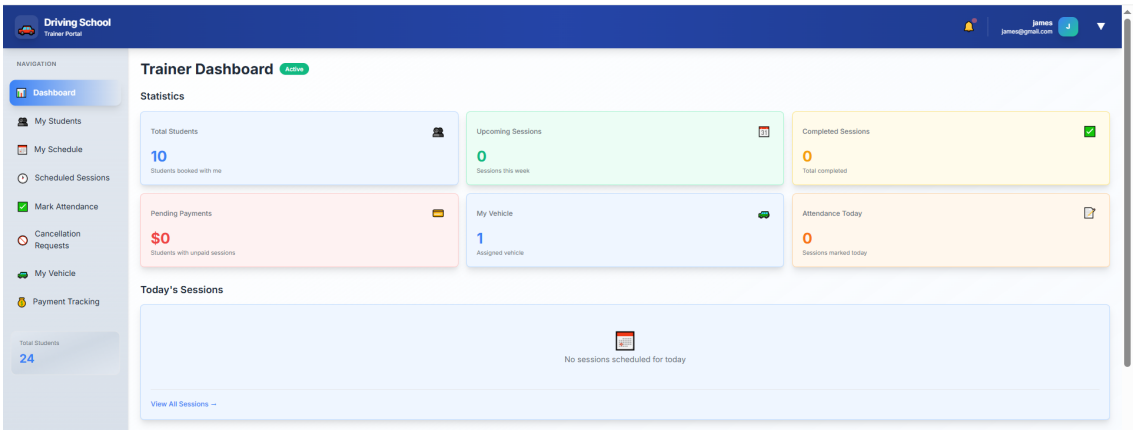

Figure 103: Trainer Dashboard - Overview (Web)

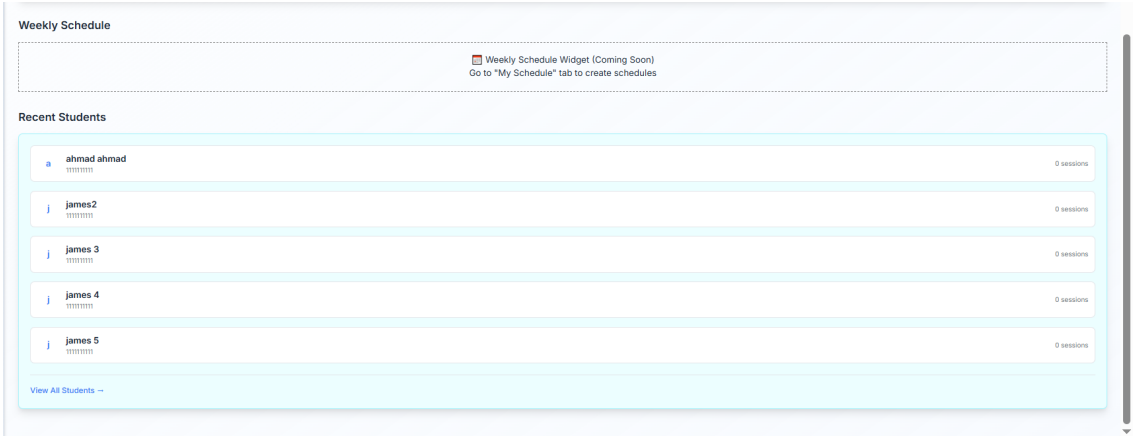

Figure 104: Trainer Dashboard - Statistics (Web)

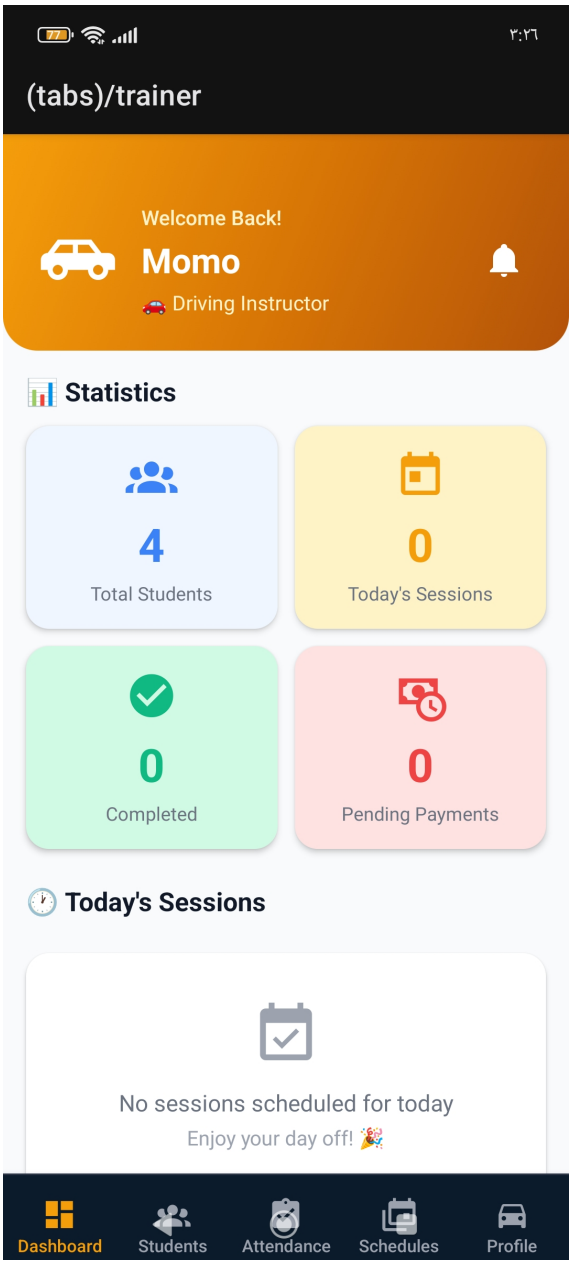

Figure 105: Trainer Dashboard - Mobile View

9.2 My Students

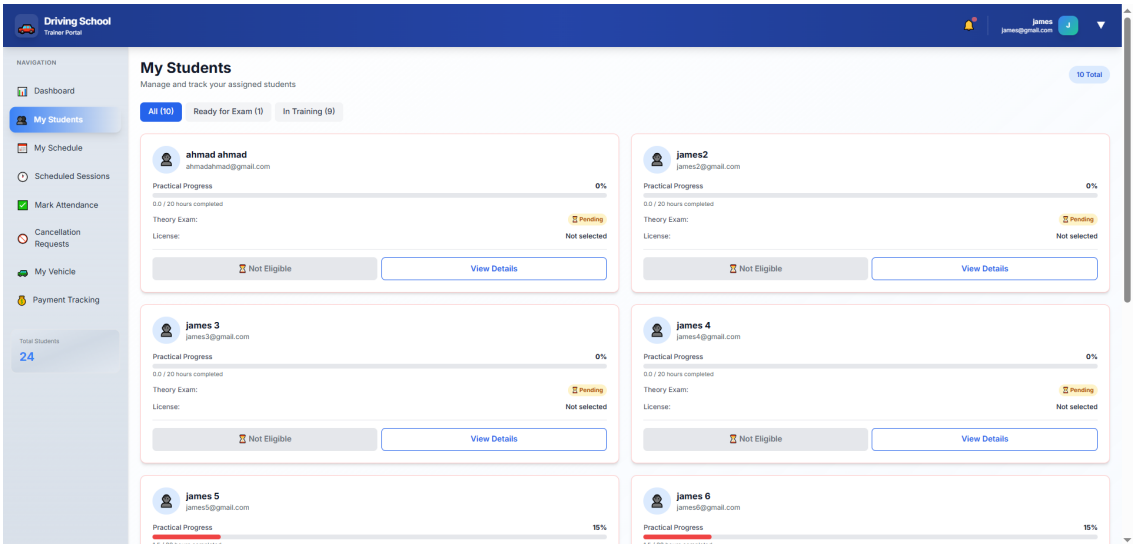

Figure 106: All Students - Web View

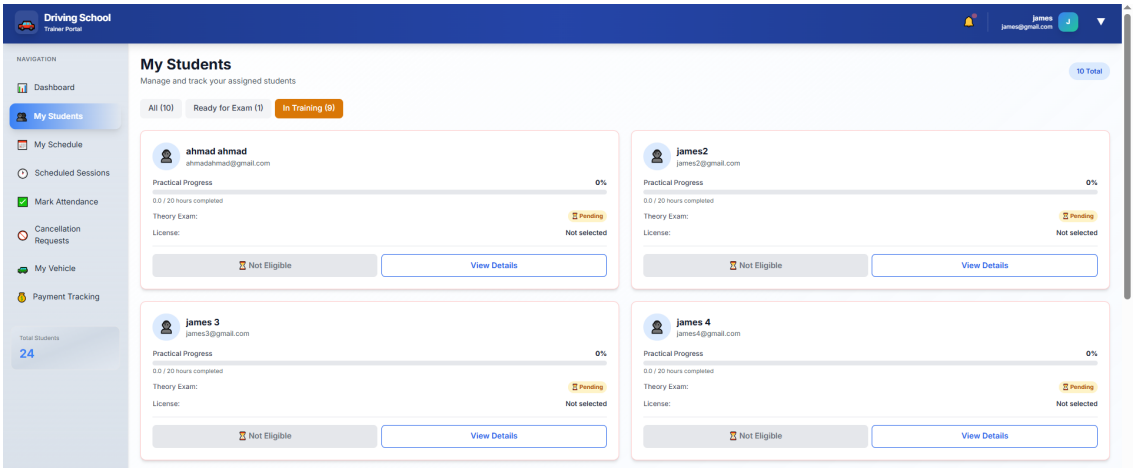

Figure 107: Students in Training - Web View

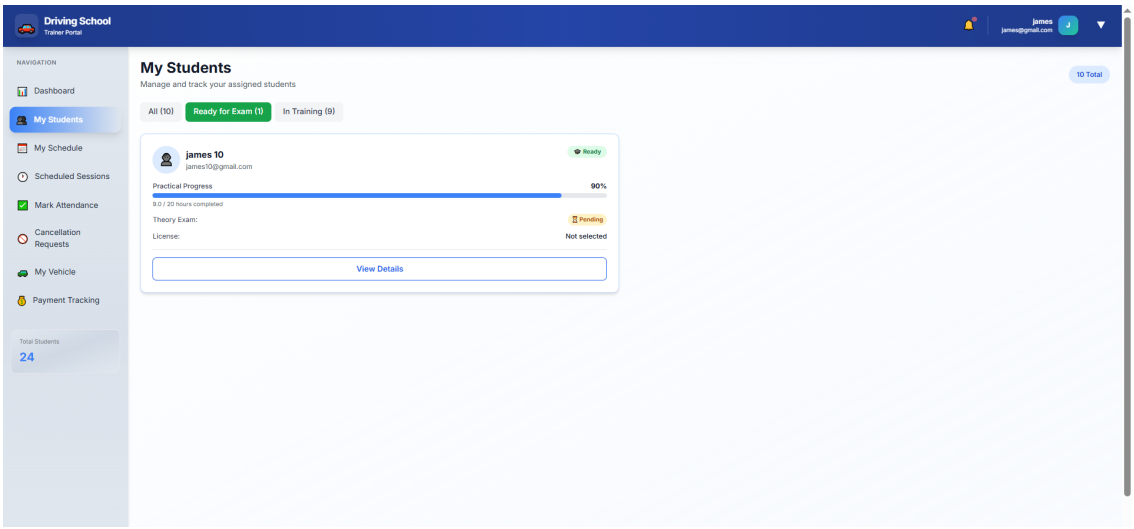

Figure 108: Students Ready for Exam - Web View

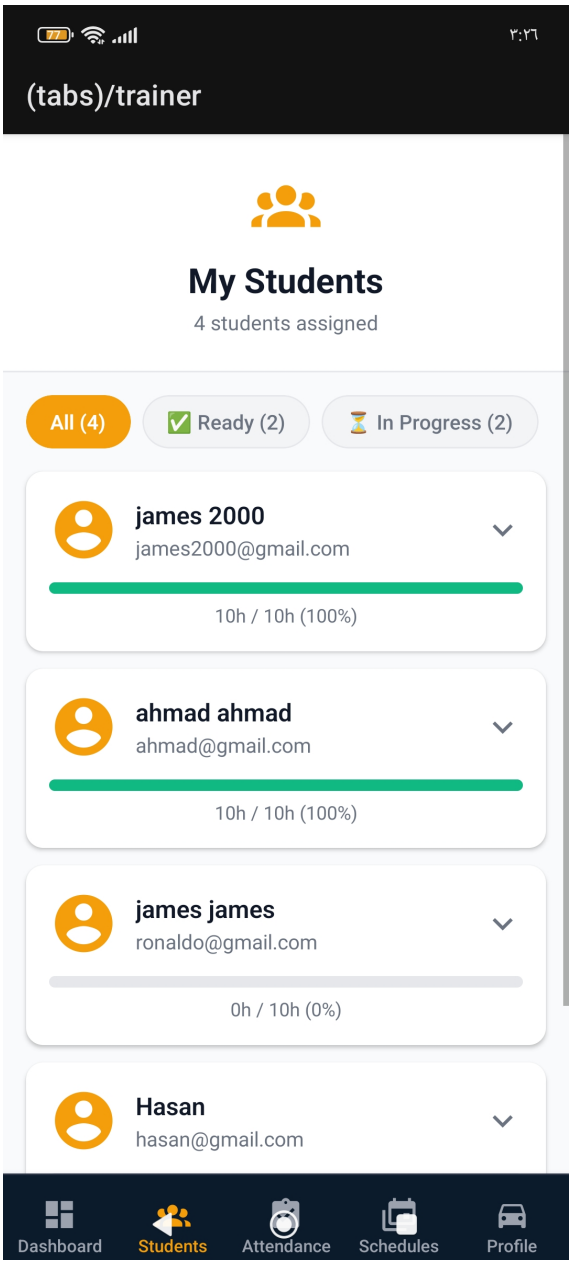

Figure 109: My Students - Mobile View

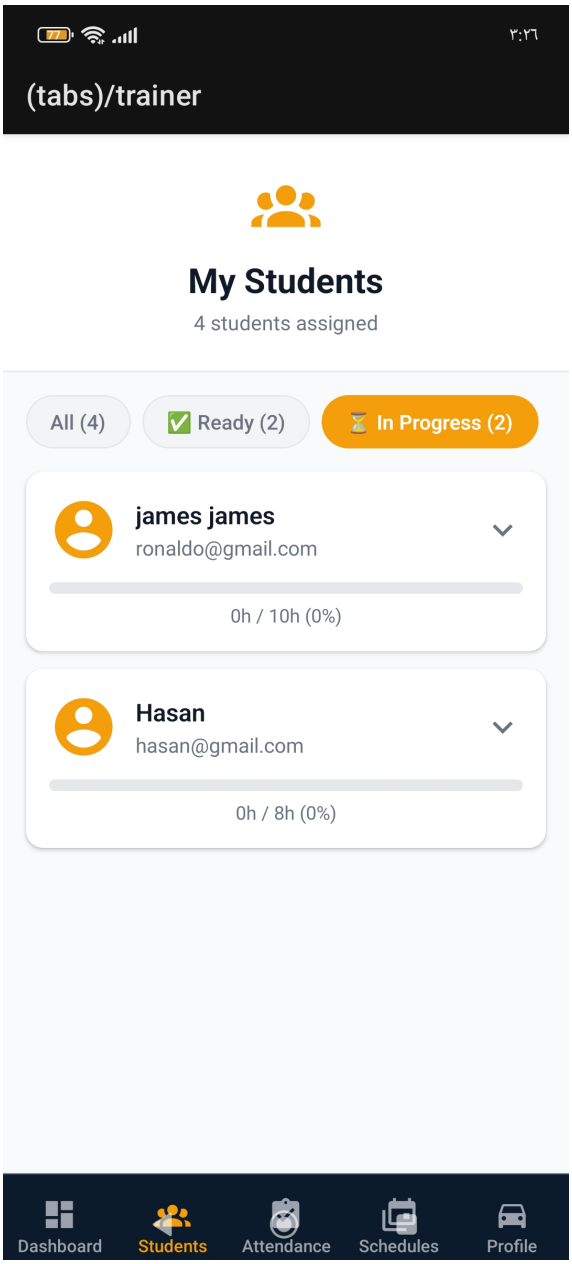

Figure 110: Students in Progress - Mobile

9.3 Schedule Management

Figure 111: My Schedule - Web View

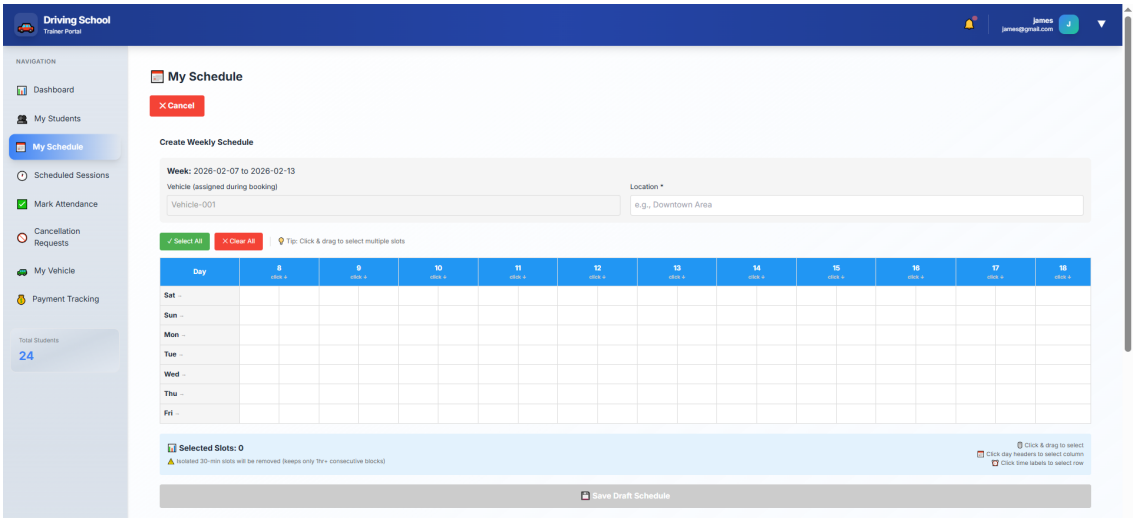

Figure 112: Create New Schedule - Web Interface

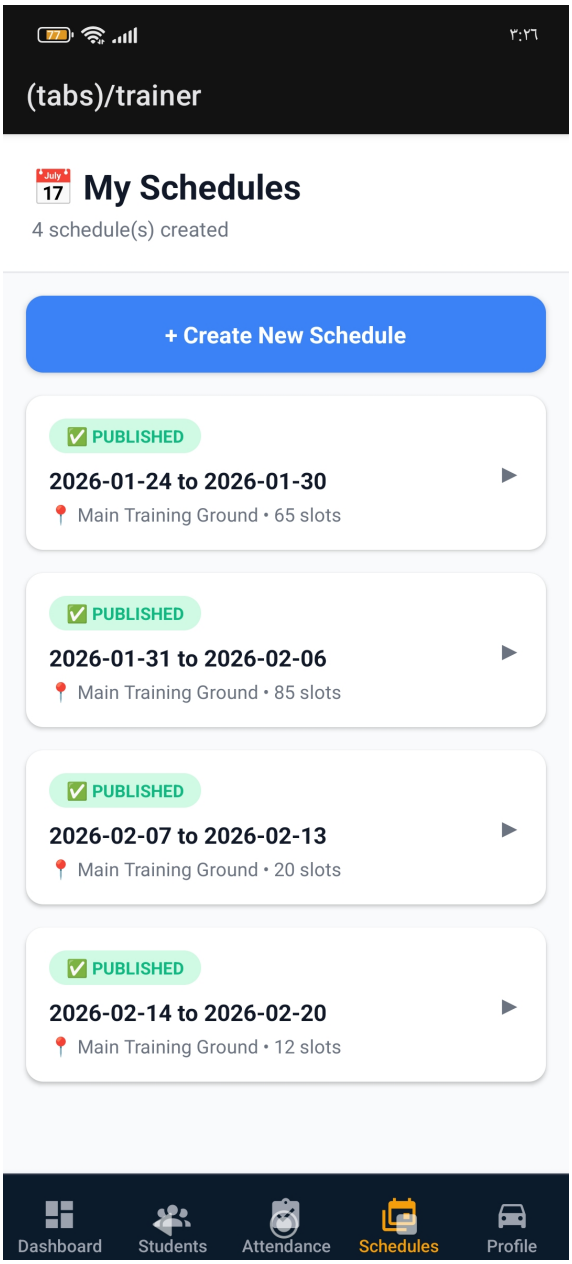

Figure 113: Schedule Overview - Mobile

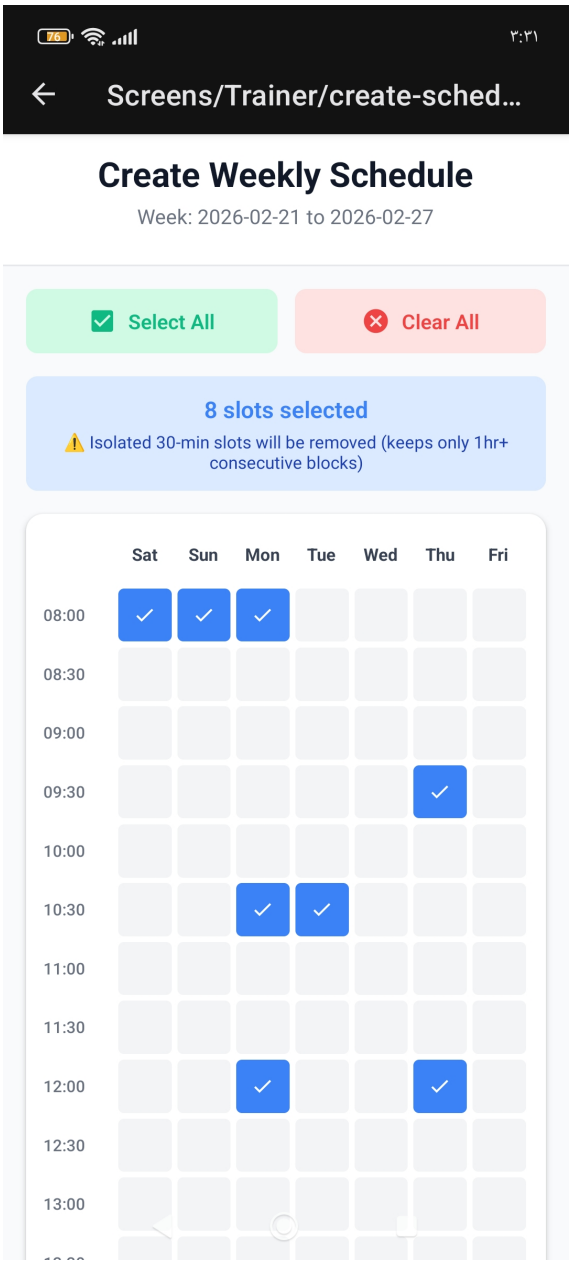

Figure 114: Create Schedule - Mobile Interface

9.4 Attendance and Session Management

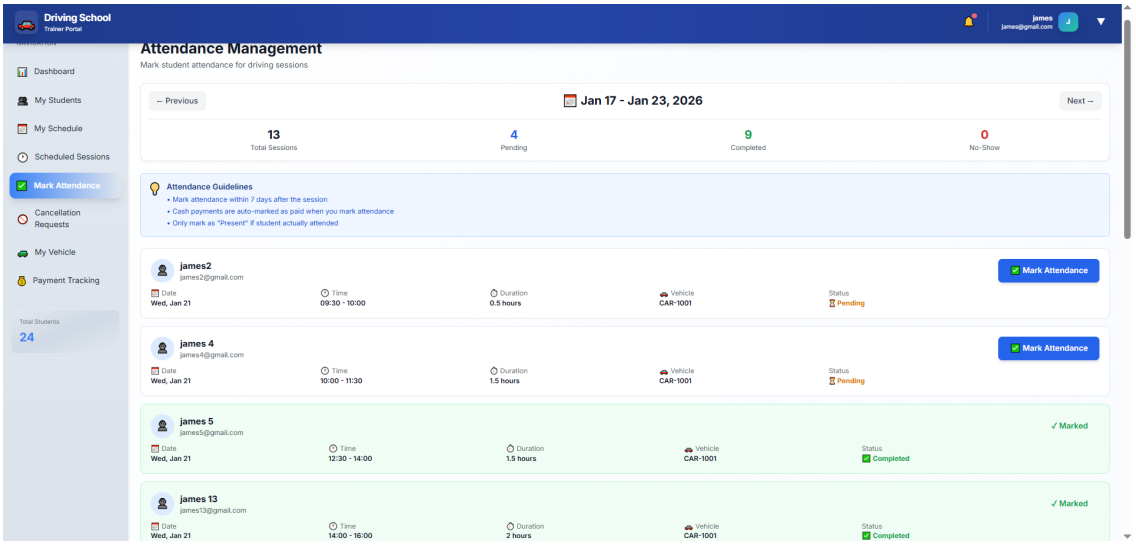

Figure 115: Mark Attendance - Web View

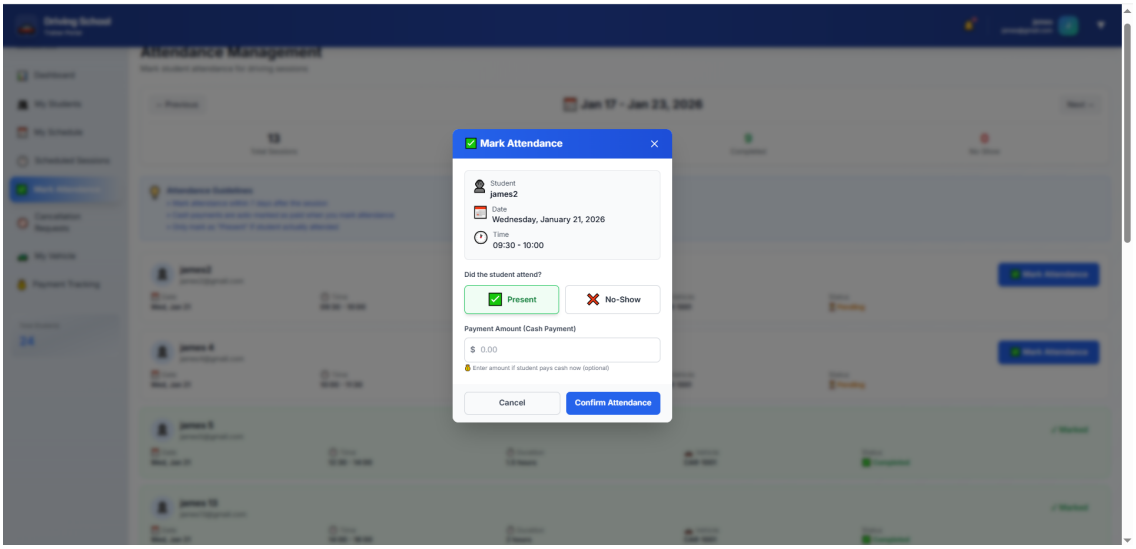

Figure 116: Attendance Actions - Web Interface

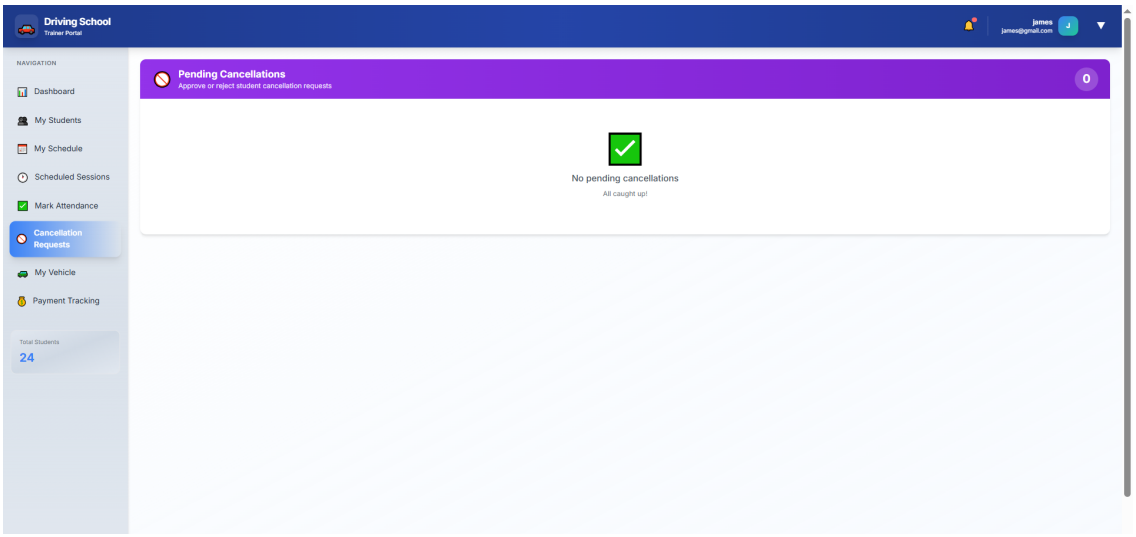

Figure 117: Session Cancellation - Web View

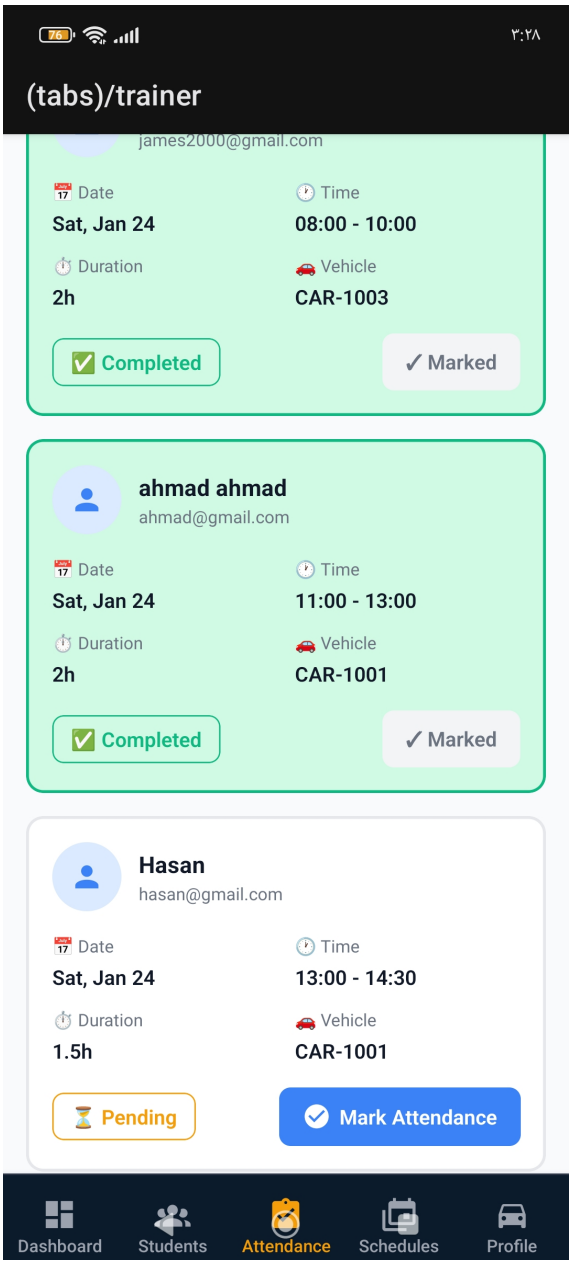

Figure 118: Attendance Tracking - Mobile

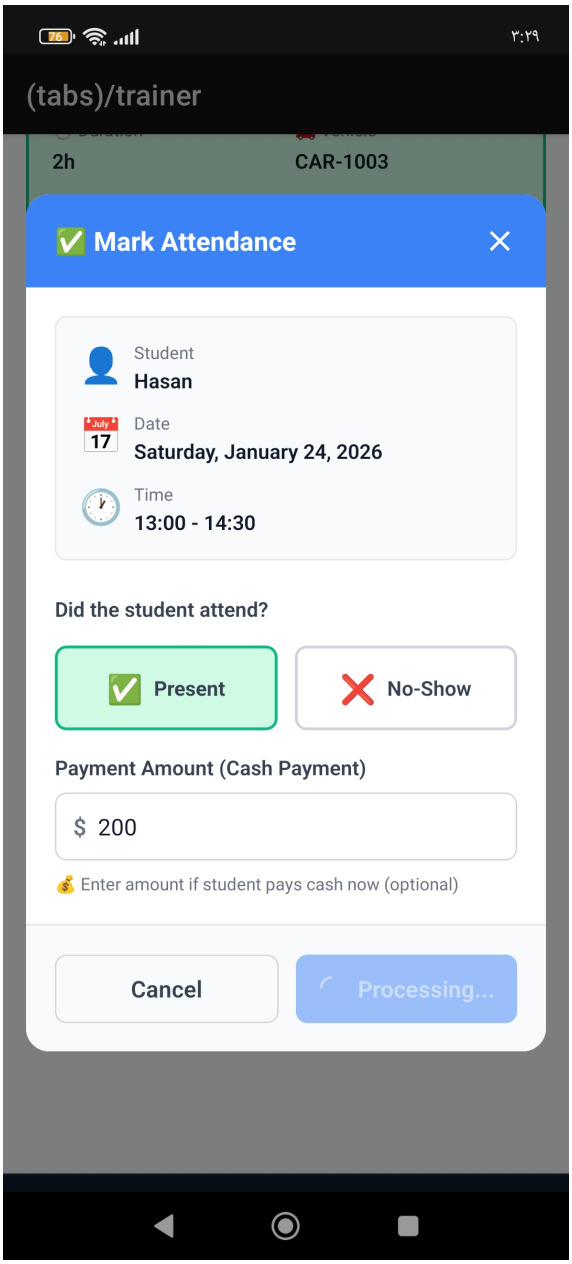

Figure 119: Mark Attendance Interface - Mobile

10 Key Features and Implementation

10.1 Credit Balance and Refund System

The system implements an automated refund mechanism where students who miss sessions after card payment receive automatic credit balance that can be used for future bookings. This feature ensures fairness and improves customer satisfaction.

10.2 Intelligent Session Booking

The booking system includes:

- Weekly booking limits (3 sessions, 5 hours max)
- One session per day restriction
- Automatic vehicle assignment based on license category
- Conflict detection and prevention
- Real-time availability updates

10.3 Progress Tracking

Comprehensive progress tracking for both theoretical and practical training:

- Lecture attendance tracking (3 lectures required)
- Practical hours completion monitoring
- Automatic readiness calculation for exams
- Visual progress indicators
- Milestone notifications

10.4 Payment Integration

Secure payment processing with Stripe integration:

- Credit/debit card payments (test mode)
- Cash payment tracking
- Credit balance system
- Payment history with detailed transactions
- Automatic receipt generation

10.5 Notification System

Real-time notifications for:

- Registration approval
- Session bookings and reminders
- Exam scheduling
- Payment confirmations
- Progress milestones
- Attendance marking
- Refund processing

10.6 In-App Messaging System

The system includes a built-in messaging feature allowing direct communication between students and their instructors (teachers and trainers):

- Students can message their assigned teacher or trainer
- Teachers and trainers can initiate conversations with their students
- Real-time message delivery
- Message history preserved for reference
- Supports text messages

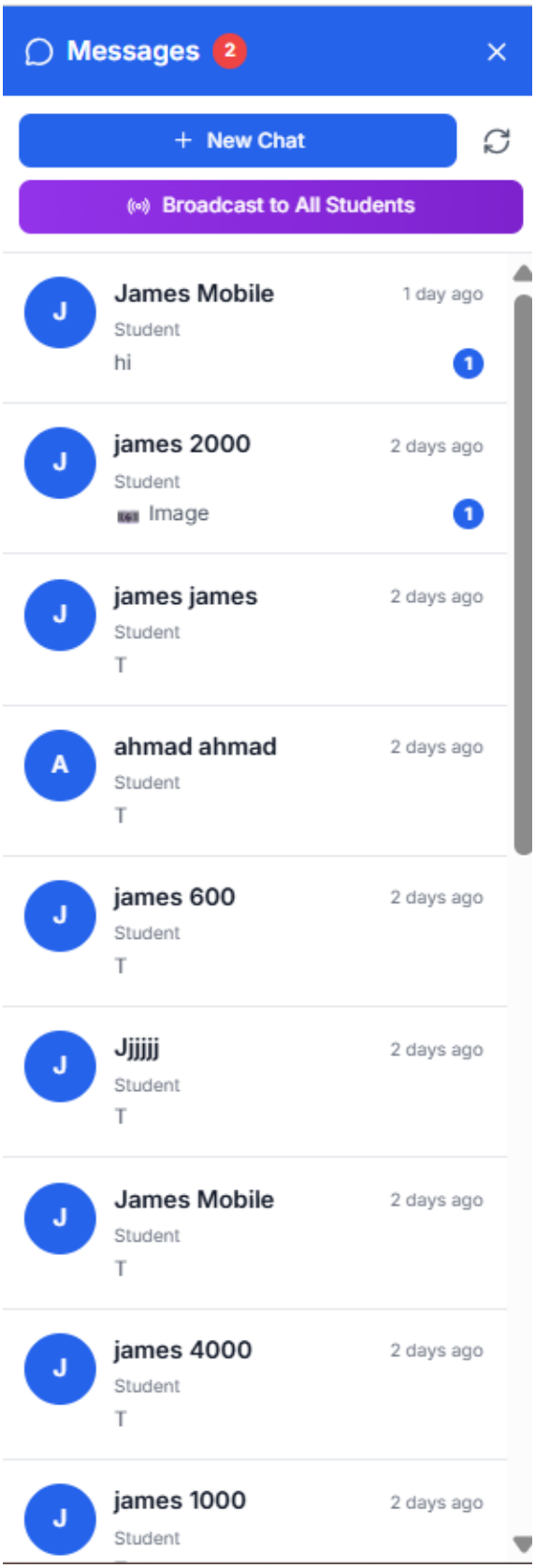

Figure 120: Choose Conversation - Messaging Interface

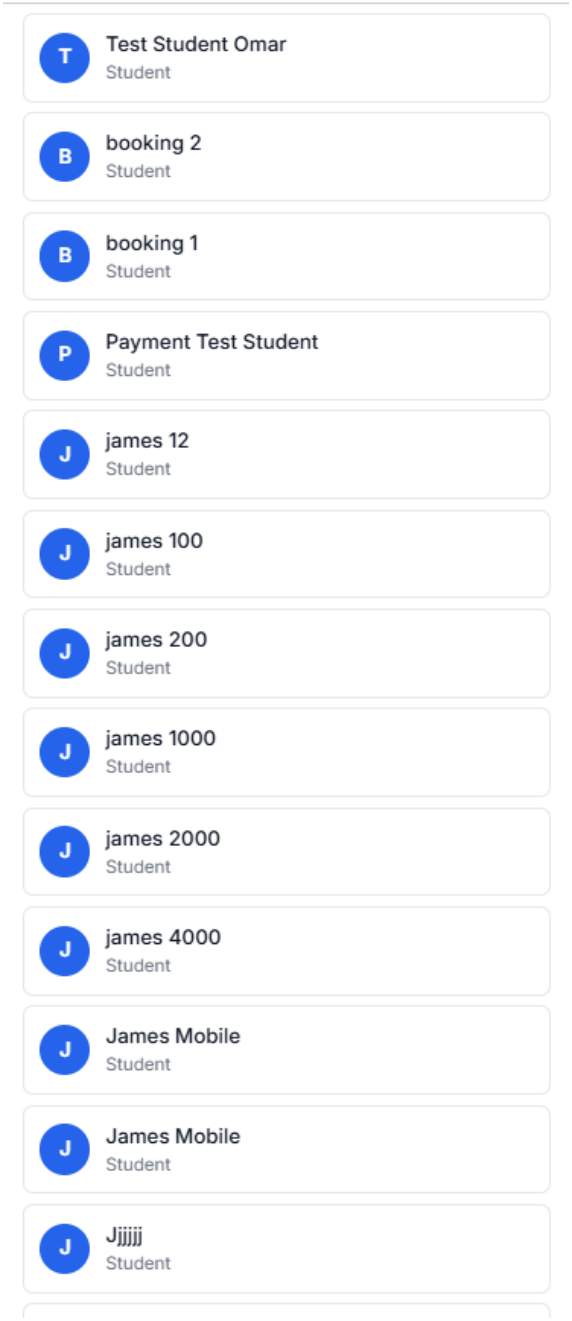

Figure 121: Start New Conversation

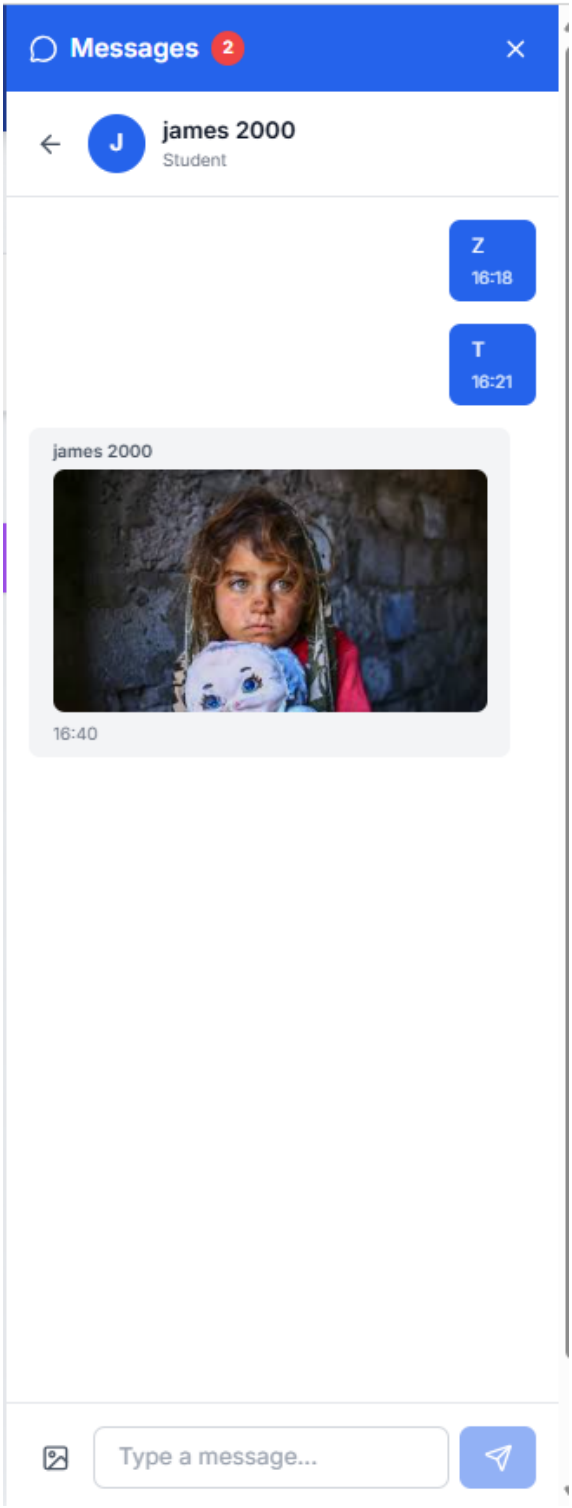

Figure 122: Messaging Conversation View

This messaging feature eliminates the need for external communication apps and keeps all driving school-related conversations within the system, making it easier to track student inquiries and instructor responses.

11 Challenges and Solutions

11.1 Technical Challenges

Challenge 1: Real-time Booking Conflicts

- *Problem:* Multiple students attempting to book the same slot simultaneously
- *Solution:* Implemented atomic database transactions with status checks

Challenge 2: Payment Processing Integration

- *Problem:* Handling both online and cash payments with different workflows
- *Solution:* Dual payment system with automatic status management

Challenge 3: Mobile Notification Delivery

- *Problem:* Expo Go limitations for push notifications
- *Solution:* Implemented polling-based notification system

11.2 Design Challenges

Challenge 1: Complex User Flows

- *Problem:* Managing different workflows for four user roles
- *Solution:* Role-based routing and conditional rendering

Challenge 2: Mobile Responsiveness

- *Problem:* Ensuring consistent UX across web and mobile
- *Solution:* Shared design system and component architecture

12 Future Enhancements

12.1 Planned Features

1. **Real-time Chat System**
    - Group chat for theoretical classes
    - File sharing capabilities
2. **Advanced Analytics**
    - Predictive analysis for student success
    - Revenue forecasting
    - Instructor performance metrics
3. **Multi-license Enrollment**
    - Allow students to enroll in multiple licenses
    - Track progress for each license separately
    - Discount system for multiple enrollments
4. **Video Integration**
    - Theoretical class recordings
    - Video tutorials library
    - Live streaming for remote classes
5. **Advanced Reporting**
    - Customizable report templates
    - Export to PDF/Excel
    - Scheduled report generation

12.2 AI-Powered Features

The integration of artificial intelligence and machine learning capabilities presents significant opportunities for enhancing the system's functionality and user experience. The following AI features are planned for future implementation:

1. **AI-Powered Chatbot Assistant**
    - 24/7 intelligent conversational support for students
    - Natural language processing for answering common queries
    - Context-aware responses based on user role and history
    - Multi-language support for diverse student base

- Reduce admin workload by 40-60% through automation

2. **Predictive Student Performance Analytics**
   - Machine learning models to predict exam success rates
   - Early identification of at-risk students requiring additional support
   - Personalized study recommendations based on learning patterns
   - Optimal training pace suggestions
   - Expected outcome: 15-25% increase in pass rates
3. **Smart Schedule Optimization**
   - AI-driven scheduling to optimize trainer availability
   - Intelligent trainer-student matching based on compatibility
   - Peak time prediction and dynamic availability adjustment
   - Weather-aware scheduling with automatic rescheduling
   - Route optimization for multiple consecutive sessions

12.3 Technical Improvements

- Migrate to native mobile builds for push notifications
- Implement caching strategies for better performance
- Add offline mode support for mobile app
- Enhance security with two-factor authentication
- Implement automated backup system

13 Conclusion and Discussion

13.1 Project Summary

The Driving School Management System represents a comprehensive solution designed to modernize and streamline the operations of traditional driving schools in Palestine. Throughout this project, we have successfully developed a full-stack application that addresses the core challenges faced by driving schools, including manual record-keeping, scheduling conflicts, payment tracking difficulties, and lack of transparent communication between students and instructors. The system provides an integrated platform that manages the complete student journey from initial enrollment through license certification, while simultaneously supporting the diverse needs of administrators, teachers, and trainers.

Our solution stands out by offering both web and mobile platforms, ensuring accessibility and convenience for all users regardless of their preferred device. The web application, built with Next.js and React, provides a powerful administrative interface with comprehensive dashboards and management tools. Meanwhile, the mobile application, developed using React Native and Expo, delivers a user-friendly experience optimized for students and trainers who need on-the-go access to booking, attendance marking, and real-time notifications. This dual-platform approach ensures maximum reach and usability across different user groups and scenarios.

The backend infrastructure, powered by Node.js, Express, and MongoDB, implements a robust three-tier architecture that separates concerns effectively. The Controller-Service-Repository pattern we adopted ensures maintainability and scalability, while MongoDB's flexible document-based structure accommodates the complex relationships between users, bookings, payments, and progress tracking. Integration with external services such as Stripe for payments, Firebase for notifications, and Cloudinary for media storage demonstrates our commitment to leveraging industry-standard solutions rather than reinventing the wheel.

13.2 Key Achievements and Technical Contributions

This project successfully delivered several significant technical achievements. We implemented a sophisticated booking system that prevents concurrent access conflicts through atomic database transactions and real-time status checking. This ensures data integrity even when multiple students attempt to book the same time slot simultaneously, a critical requirement for any reservation-based system. The dual payment system we designed elegantly handles both online card payments through Stripe and traditional cash payments, with automatic status management and refund processing when sessions are not attended.

The role-based access control system we built provides fine-grained security across four distinct user roles: administrators, students, teachers, and trainers. Each role has precisely defined permissions and access levels, implemented through JWT token authentication and middleware verification. This ensures that users can only access features and data appropriate to their responsibilities, maintaining both security and privacy throughout the system. The authentication mechanism uses industry-standard bcrypt hashing for password storage and JWT tokens for session management, providing a secure foundation for all system operations.

Progress tracking represents another major achievement, where the system automatically

monitors student advancement through theoretical classes, practical driving sessions, exam attempts, and payment completion. Students receive real-time visibility into their current status, upcoming requirements, and next steps, eliminating the confusion and uncertainty that often characterizes traditional driving school experiences. The notification system, implemented through Firebase Cloud Messaging, keeps students informed about important events such as exam scheduling, booking confirmations, and trainer assignments.

From a user experience perspective, we invested significant effort in creating intuitive interfaces that require minimal training. The web dashboard features clear navigation, informative visualizations of statistics and progress, and streamlined workflows for common tasks. The mobile application follows Material Design principles through React Native Paper components, ensuring familiar interactions and visual consistency. Both platforms feature responsive designs that adapt gracefully to different screen sizes and orientations.

13.3 Learning Outcomes and Development Experience

This graduation project provided invaluable hands-on experience in modern full-stack software development. Working with the MERN stack (MongoDB, Express, React, Node.js) gave us deep understanding of how contemporary web applications are architected and deployed. We learned to make informed decisions about technology selection, considering factors such as community support, documentation quality, performance characteristics, and long-term maintainability. The experience of building both web and mobile applications simultaneously taught us about code reuse strategies, API design for multiple clients, and the unique constraints and opportunities of each platform.

Database design and management formed a crucial learning component of this project. We gained practical experience in schema design for MongoDB, including decisions about embedding versus referencing documents, index creation for query optimization, and aggregation pipeline construction for complex analytics. Understanding the performance implications of different data modeling approaches and learning to use MongoDB's rich query language effectively will serve us well in future database-driven projects.

Payment integration with Stripe exposed us to the complexities of financial transaction processing, including webhook handling, idempotency considerations, error recovery strategies, and security best practices. We learned about PCI compliance requirements, test versus production environments, and the importance of comprehensive logging and monitoring for payment systems. Although we implemented the system in test mode, we gained conceptual understanding necessary for production payment processing in future projects.

Project management and collaboration skills developed significantly throughout this work. We used Git for version control, learning advanced workflows including branching strategies, pull requests, code reviews, and conflict resolution. Dividing responsibilities between team members, with Mohamad focusing on backend and API development while Ahmad concentrated on frontend and mobile interfaces, taught us about effective collaboration, clear communication, and integration testing across components built by different developers.

Debugging and problem-solving abilities improved dramatically as we encountered and resolved various technical challenges. From concurrent booking conflicts to payment webhook timing issues to mobile notification delivery complexities, each problem required systematic analysis, research, experimentation, and validation. Learning to read error messages effec-

tively, use debugging tools, write targeted test scripts, and validate fixes across different scenarios built confidence and competence in troubleshooting complex systems.

13.4 Impact and Practical Value

The Driving School Management System demonstrates substantial potential to transform how driving schools operate in Palestine and similar markets. By automating administrative tasks such as student enrollment processing, schedule creation, attendance tracking, and payment recording, the system can reduce administrative overhead by an estimated 60-70 percent. This efficiency gain allows driving school staff to focus on higher-value activities such as student support, quality instruction, and business development rather than manual paperwork and record-keeping.

Student satisfaction improves significantly through transparent progress tracking and convenient self-service capabilities. Students can view their completion status for theoretical lectures, practical hours driven, exam results, and payment history at any time through the mobile app or web portal. This transparency eliminates frustration caused by unclear requirements or lost records in traditional paper-based systems. The ability to book practical sessions online at their convenience, rather than calling during business hours or visiting the school in person, represents a major quality-of-life improvement aligned with modern expectations for digital services.

Scheduling conflict minimization through the intelligent booking system benefits both students and trainers. The system's built-in constraints prevent double-booking, enforce weekly session limits, and ensure students cannot book multiple sessions on the same day. Trainers benefit from automated schedule management that prevents overlapping commitments and provides clear visibility into upcoming sessions. The reduction in scheduling errors and the elimination of manual coordination reduces stress and improves overall operational smoothness.

Data-driven decision making becomes possible through the system's comprehensive data collection and reporting capabilities. Administrators can analyze metrics such as average time from enrollment to license completion, pass rates for theoretical and practical exams, trainer utilization and productivity, revenue patterns, and student satisfaction trends. These insights enable informed decisions about resource allocation, pricing adjustments, curriculum improvements, and strategic planning. The digital audit trail maintained by the system also provides accountability and facilitates resolution of disputes or questions about past events.

The 24/7 accessibility provided by web and mobile platforms removes temporal and geographical barriers to interaction with the driving school. Students working unusual hours or living far from the school location can manage their learning journey without visiting during limited business hours. Trainers can check schedules, mark attendance, and communicate with students from any location with internet connectivity. This flexibility accommodates modern lifestyles and work patterns, making driving school services more accessible to broader populations.

13.5 Challenges Overcome and Lessons Learned

Throughout development, we encountered and overcame several significant technical and design challenges. The real-time booking conflict prevention required careful implementation of database transactions and status checks to ensure data integrity under concurrent access. We learned the importance of atomic operations, optimistic locking strategies, and thorough testing with simultaneous users to verify correctness under load.

Integrating payment processing presented complexities around webhook reliability, race conditions between booking creation and payment confirmation, and proper error handling for declined transactions. We implemented retry mechanisms, idempotency keys, and comprehensive logging to create a robust payment system that handles edge cases gracefully. This experience taught us that external API integration requires defensive programming and cannot assume ideal network conditions or timing.

Mobile push notification delivery faced limitations in Expo Go development mode that required workarounds during development and highlighted the importance of understanding platform constraints early. We learned to carefully evaluate framework limitations against project requirements and plan for necessary migration paths to production environments with different capabilities.

Design challenges included managing complex user flows across four different roles with distinct interfaces and capabilities. We learned the value of comprehensive user flow diagrams, wireframing before implementation, and iterative refinement based on feedback. Creating consistent user experiences across web and mobile platforms while respecting platform-specific conventions required careful attention to design systems and component architecture.

These challenges taught us valuable lessons about planning, risk assessment, iterative development, and the importance of early testing and validation. We learned that complex systems cannot be fully specified upfront and that flexibility to adapt designs based on discovered requirements or technical constraints is essential for successful project completion.

13.6 Future Directions and Recommendations

While the current system provides comprehensive functionality for core driving school operations, several enhancements could further increase value. Multi-license enrollment support would allow students to pursue multiple license types simultaneously with independent progress tracking, appealing to students seeking commercial or motorcycle licenses in addition to standard vehicle licenses. Enhanced analytics and reporting capabilities, including predictive modeling for student success, revenue forecasting, and instructor performance evaluation, would provide even greater decision support for school management.

Video integration for theoretical class recordings, tutorial libraries, and potentially live streaming for remote classes would increase educational flexibility and support students with varying learning styles and schedules. Advanced messaging features such as group chat for classes, file sharing capabilities, and integration with external communication platforms could further improve collaboration and support. Implementation of automated backup systems, disaster recovery procedures, and production monitoring would increase system reliability and operational readiness for mission-critical deployment.

From a business perspective, evolution toward a multi-tenant architecture would allow the

system to serve multiple driving schools from a single deployment, enabling a software-as-a-service business model. This would require additional work on data isolation, customization capabilities, and per-organization configuration, but would dramatically increase the potential market and revenue opportunities. Partnership with actual driving schools for pilot deployments would provide valuable real-world feedback and validation, helping refine the system based on operational experience rather than assumptions.

13.7 Final Remarks

The successful completion of this graduation project demonstrates the practical application of computer engineering principles to address genuine real-world problems in the Palestinian market. We designed and implemented a complex, full-featured system that integrates modern web technologies, mobile development, payment processing, real-time notifications, and comprehensive data management. The system architecture we created is scalable, maintainable, and provides a solid foundation for future enhancements and evolution.

Beyond the technical artifacts produced, this project developed our capabilities as software engineers and problem solvers. We learned to research technologies, evaluate alternatives, make informed decisions, implement solutions, test thoroughly, and iterate based on feedback. The experience of seeing a project through from initial concept to working implementation taught lessons about perseverance, attention to detail, user-centered design, and the satisfaction of creating something useful and functional.

We believe this project makes a meaningful contribution to the ongoing digitalization of educational and training institutions in Palestine. As more services move online and users expect convenient digital interactions, systems like the Driving School Management System will become essential for institutions seeking to remain competitive and meet modern expectations. The technical approaches, architecture patterns, and integration strategies we developed apply broadly to many similar domain-specific management systems, and we hope our work can serve as a reference for future student projects addressing local market needs with modern technology solutions.

Ultimately, this project validates our education and preparation as software engineers ready to contribute to the Palestinian technology sector and global software industry. The combination of theoretical knowledge from our coursework and practical implementation experience from this project provides a strong foundation for our future careers and continued learning in this rapidly evolving field.

References

[1] React Documentation. (2024). *React - A JavaScript library for building user interfaces*. Retrieved from https://react.dev/

[2] Next.js Documentation. (2024). *Next.js by Vercel - The React Framework*. Retrieved from https://nextjs.org/docs

[3] Node.js Foundation. (2024). *Node.js Documentation*. Retrieved from https://nodejs.org/docs/

[4] MongoDB Inc. (2024). *MongoDB Manual*. Retrieved from https://docs.mongodb.com/

[5] Meta Platforms, Inc. (2024). *React Native Documentation*. Retrieved from https://reactnative.dev/docs/

[6] Expo. (2024). *Expo Documentation*. Retrieved from https://docs.expo.dev/

[7] Stripe, Inc. (2024). *Stripe API Documentation*. Retrieved from https://stripe.com/docs/api

[8] Google LLC. (2024). *Firebase Cloud Messaging Documentation*. Retrieved from https://firebase.google.com/docs/cloud-messaging

[9] Cloudinary Ltd. (2024). *Cloudinary Documentation - Image and Video Management*. Retrieved from https://cloudinary.com/documentation

[10] Tailwind Labs. (2024). *Tailwind CSS Documentation*. Retrieved from https://tailwindcss.com/docs

[11] Microsoft Corporation. (2024). *TypeScript Documentation*. Retrieved from https://www.typescriptlang.org/docs/

A Technology Stack and Dependencies

The Driving School Management System is built using modern web and mobile technologies. The following components form the foundation of the system:

A.1 Core Technologies

Backend Technologies:

- Node.js version 20 - JavaScript runtime for server-side execution
- Express.js - Web application framework for building RESTful APIs
- MongoDB version 7 - NoSQL database for flexible data storage
- Mongoose - Object Data Modeling (ODM) library for MongoDB

Frontend Web Technologies:

- Next.js version 15 - React framework with server-side rendering
- React version 18 - JavaScript library for building user interfaces
- TypeScript - Typed superset of JavaScript for improved code quality
- Tailwind CSS - Utility-first CSS framework for styling

Mobile Technologies:

- React Native - Framework for building native mobile applications
- Expo SDK version 54 - Toolchain and framework for React Native
- React Native Paper - Material Design component library
- TypeScript - For type-safe mobile application development

A.2 External Services Integration

The system integrates with several external services to provide enhanced functionality:

- **Stripe** - Online payment processing platform for handling card payments and transactions
- **Firebase Cloud Messaging (FCM)** - Push notification delivery service for real-time alerts
- **Cloudinary** - Cloud-based image and media management service for profile photos and documents
- **MongoDB Atlas** - Cloud database hosting option for production deployments

A.3 Development Tools

The following tools were used during development:

- Git - Version control system for source code management
- Visual Studio Code - Code editor with TypeScript and React support
- Postman - API testing and documentation tool
- MongoDB Compass - Graphical interface for MongoDB database management
- Expo CLI - Command-line tool for React Native development

B Database Schema Overview

- **User Model** - Connected to Student, Teacher, Trainer (1:1 relationships)
- **Student Model** - Connected to User, License, Payments, ExamAttempts, Messages (1:Many relationships)
- **Teacher Model** - Connected to User, Students, TheoSchedule (1:Many relationships)
- **Trainer Model** - Connected to User, Students, Vehicle, PracticalSchedule (1:Many relationships)
- **TimeSlot Model** - Connected to PracticalSchedule, Student, Vehicle (Many:1 relationships)
- **Payment Model** - Connected to Student, TimeSlot (Many:1 relationships)
- **Exam and ExamAttempt Models** - Connected to Students (1:Many relationships)

For detailed installation instructions, API endpoint documentation, and developer guidelines, please refer to the README.md files in the project repository:

- `backend/README.md` - Backend setup and API documentation
- `frontend/README.md` - Frontend setup instructions
- `mobile/README.md` - Mobile app setup and deployment

Source Code Repository:
The complete source code for this project is available on GitHub:
`https://github.com/Mohamad-Salha/Graduation-SW`

C Glossary of Terms

API (Application Programming Interface): Set of rules that allows different software applications to communicate with each other.

Backend: Server-side part of the application that handles business logic, database operations, and authentication.

bcrypt: Password hashing algorithm used to securely store passwords.

Frontend: Client-side part of the application that users interact with (web or mobile interface).

JWT (JSON Web Token): Secure method for transmitting information between parties as a JSON object, used for authentication.

MongoDB: NoSQL database that stores data in flexible, JSON-like documents.

Next.js: React framework for building web applications with server-side rendering.

React Native: Framework for building native mobile applications using React and JavaScript.

REST API: Architectural style for designing networked applications using HTTP requests.

Role-Based Access Control (RBAC): Security approach that restricts system access based on user roles.

Stripe: Online payment processing platform for handling credit card transactions.

Time Slot: Specific time period available for booking practical driving sessions.

Webhook: Automated message sent from one application to another when a specific event occurs.