People's Democratic Republic of Algeria
Ministry of Higher Education and Scientific Research
M'Hamed Bougara Boumerdes University



Faculty of Sciences
Computer Science Department

Domain : Computer and Mathematics
Sector : Computer Science
Specialization : Computer Systems

N ◦ of the specialty authorization decree: Decree No. 872 of 07/26/2016

Final graduation memory in view of obtaining the academic license diploma

Topic

Conception and Realization of a Website for a Driving School Resources Management

Submitted
AISSANI Manel
AZZEDDINE Lina
BENHAMMANA Rania
BOULELLOU Amira

Defended on 03/06/2026 before the jury composed of
SENOUSSI Nour El Houda **: Supervisor**
DJOUZI Kheyreddine **: Examiner**

Academic Year: 2025-2026

AKNOWLEDGMENTS

We would like, first, to thank “ALLAH”, the Most-Powerful and Merciful, for giving us the strength, patience, and courage to overcome the difficulties and obstacles that faced us during this long period of study, and throughout the realization of this work.

Secondly, we are deeply grateful to our Supervisor: **Ms. SENOUSSI Nour El Houda**, for believing in us. She guided us patiently, gave us her time generously, and shared with us precious advice. We benefited greatly from her knowledge during this enriching experience.

We want in third, to thank our strong thanks for the Examiner: **M. DJOUZI Kheyreddine,** it was an honour to have you as a member of the jury for our work.

We want to thank in fourth, the ensemble of our teachers from the Computer Science Department, for their valuable guidance throughout our studies.

In final, we want to thank sincerely our parents, for their support and presence during our study curriculum. Our Families and Friends, who provided us the moral support.

ABSTRACT

The goal of this final project is to create a website for driving school within a company. This platform will simplify the reservation of sessions, plan exams and many more operations. Three types of users are implemented: manager, monitor and can-candidate. Beside a general introduction and a conclusion, this report is divided into three chapters. The first chapter brings a general definition about what is a website (different types of architectures, their key characteristics and how they function). The second chapter discusses about the UML implementation (use case, sequence and class diagrams, in addition the logical model of data and the database). The third and last chapter presents the softwares and tools that we used to create our website, in addition to the resulting interfaces. The website we built is an experience for using our theoretical and coding skills to solve a problematic. It can be enriched in the future with more precise options.

Keywords: Driving license, Website, Platform.

Résumé

L'objectif de ce projet de fin d'études est de créer un site web pour une auto-école au sein d'une entreprise. Cette plateforme simplifiera la réservation des séances, la planification des examens, ainsi que de nombreuses autres opérations. Trois types d'utilisateurs sont implémentés : le gestionnaire, le moniteur et le candidat. Outre une introduction générale et une conclusion, ce rapport est divisé en trois chapitres. Le premier chapitre présente une définition générale de ce qu'est un site web, ses différentes architectures, leurs caractéristiques principales et leur fonctionnement. Le deuxième chapitre traite de l'implémentation UML (diagrammes de cas d'utilisation, de séquence et de classes), ainsi que du modèle logique de données et de la base de données. Le troisième et dernier chapitre présente les logiciels et outils que nous avons utilisés pour créer notre site, ainsi que les interfaces obtenues. L'application que nous avons construite est une expérience qui mobilise nos compétences théoriques et de codage pour résoudre des problèmes. Elle pourra être enrichie à l'avenir avec des options plus précises.

Mots-clés : Permis de conduire, Site web, Plateforme.

TABLE OF CONTENTS

AKNOWLEDGMENTS ..... **I**

ABSTRACT ..... **II**

TABLE OF CONTENTS ..... **III**

LIST OF FIGURES ..... **VIII**

LIST OF TABLES ..... **XI**

LIST OF ABBREVIATIONS ..... **XII**

GENERAL INTRODUCTION **1**

CHAPTER 1 WEBSITE DEFINITION **3**

1. Introduction 3

2. What is a Website 3

3. Key Characteristics of Website 3

3.1. Server-Based Processing 3

3.2. Easy Updates 3

3.3. Cross-Platform Compatibility 4

3.4. Network Dependency 4

4. How Website Function 4

5. Types of Websites 5

5.1. Static Website 5

5.2. Dynamic Website 5

6. The Architecture of Website 5

6.1. Core Components of the Architecture 5

6.1.1. Frontend (Client-Side)……..……...5

6.1.2. Backend (Server-Side)……..……...6

6.1.3. Database (Persistence Layer)……..……6

6.1.4. Web Server……..……..…6

6.1.5. API……..……6

6.2. Common Architectural Models and Types……..……..…6

6.2.1. One-Tier Architecture……6

6.2.2. Two-Tier Architecture……...7

6.2.3. Three-Tier Architecture……7

6.2.4. Monolithic Architecture……7

6.2.5. Microservices Architecture……...7

6.2.6. Serverless Architecture……8

7. Conclusion……..……...8

CHAPTER 2 ANALYSIS AND CONCEPTION 9

1. Introduction 9

2. UML Modelling Tool 9

3. System Actors……9

4. Use Case Diagram ..... .10

4.1. Candidate Use Case Diagram and it's Textual Description……..……...11

4.2. Monitor Use Case Diagram and it's Textual Description 12

4.3. Manager Use Case Diagram and it's Textual Description……..…13

5. Sequence Diagram 14

5.1. Add Candidate / Monitor Sequence Diagram…….…….15

5.2. Change Session Sequence Diagram……...15

5.3. Affect Worktime for Monitors Sequence Diagram……..16

5.4. Login Sequence Diagram…….....17

5.5. Evaluate Progression Sequence Diagram…….17

6. Class Diagram…….……18

7. Logical Data Model (MLD)……...19

8. Rules of Conversion from Class Diagram to MLD……..20

9. Conclusion……..20

CHAPTER 3 IMPLEMENTATION 21

1. Introduction 21

2. Web Technologies Used..... 21

2.1. Server-Side Development.....21

2.1.1. PHP……..…….…21

2.1.2. MYSQL…….22

2.2. Client-Side Development……...22

2.2.1. Bootstarp 5……22

2.2.2. Vanilla JavaScript (ECMAScript)……23

2.2.3. Chart.js……...23

3. Development Tools.....24

3.1. WAMP64 (WampServer)……..24

3.2. Visual Studio Code……..24

4. Application Description……...24

4.1. Home Page……...24

4.2. Authentication System Page……...25

4.3. User Profile Pages……...25

4.4. Dashboard Pages……26

4.5. Schedule Pages……27

4.6. Session Change Request Pages……...28

4.7. Candidate Management Pages……....29

4.8. Monitor Management Page……30

4.9. Vehicle Management Pages……30

4.10. Session Change Request Management Page……31

4.11. Candidate Progress Evaluation Page……..32

4.12. Candidate Progress Tracking Page……...32

4.13. Theoretical Lessons Page…….……...32

4.14. Practice Test Page……..33

4.15. Extra Candidates Pages……33

5. Conclusion.....34

6. GENERAL CONCLUSION 35

7. WEBOGRAPHY……..36

LIST OF FIGURES

2.1. The Global Use Case Diagram of our Driving School System…….....10

2.2. Viewer Use Case Diagram 11

2.3. Candidate Actions Use Case Diagram……...11

2.4. Monitor Actions Use Case Diagram……..12

2.5. Manager Actions Use Case Diagram……..13

2.6. Add Candidate / Monitor Sequence Diagram..... 15

2.7. Change Session Sequence Diagram 16

2.8. Affects Worktime for Monitors Sequence Diagram……...…16

2.9. Login Sequence Diagram……...17

2.10. Evaluate Progression Sequence Diagram…….17

2.11. The Class Diagram our System…….18

2.12. Logical Data Model..... 19

3.1. Home Page.....25

3.2. Login Page..... 25

3.3. Manger Profile Page..... 26

3.4. Monitor Profile Page……..26

3.5. Candidate Profile Page 26

3.6. Manager Dashboard.....27

3.7. Monitor Dashboard..... 27

3.8. Candidate Dashboard.....27

3.9. Manager Schedule Page..... 28

3.10. Monitor Schedule Page 28

3.11. Candidate Schedule Page..... 28

3.12. Monitor Request Session Change Page…….…...29

3.13. Candidate Request Session Change Page…….29

3.14. Manager Candidate Management Page……...29

3.15. Monitor Candidate Management Page……...……..30

3.16. Monitor Management Page……..30

3.17. Manager Vehicle Management Page……...….31

3.18. Monitor Vehicle Management Page…….31

3.19. Change Request Management Page.....31

3.20. Progress Evaluation Page 32

3.21. Candidate Progress Tracking Page 32

3.22. Theoretical Lessons Page 33

3.23. Practice Test Page.....33

3.24. Extra Candidates (No account).....34

3.25. My Extra Candidates 34

LIST OF TABLES

2.1. Textual Description of Candidate Use Case……..……...12

2.2. Textual Description of Monitor Use Case..... 13

2.3. Textual Description of Manager Use Case ……...14

LIST OF ABBREVIATIONS

API	Application Programming Interface
UML	Unified Modeling Language
MLD	Logical Data Model
PHP	Hypertext Pre-processor
SQL	Structured Query Language
UI	User Interface
UX	User Experience
RDBMS	Relational Database Management System
DOM	Document Object Model
VS Code	Visual Studio Code
IDE	Integrated Development Environment
HTTP	HyperText Transfer Protocol
CSS	Cascading Style Sheets

GENERAL INTRODUCTION

In recent years, digital technologies have transformed the way organizations manage their activities and interact with users. Many institutions are moving from traditional paper-based management to computerized systems in order to improve efficiency, organization, and accessibility of information.

This section introduces the general context of the project, explains the importance of digital technologies in improving the management of organizations and services, and presents the role of information systems in simplifying administrative tasks and improving communication between users.

Websites have become a fundamental part of modern computing. From online banking and e-commerce to collaborative work tools and social media platforms, websites serve millions of users daily across the globe. Their widespread adoption stems from several key advantages (users can access them from any device with a browser and an internet connection, they require no installation and updates are managed centrally by developers rather than being distributed to individual users) .

Presentation of the Entity

Driving schools are institutions responsible for training candidates to obtain a driving license. Their activities include theoretical courses (traffic rules), practical driving lessons, exam preparation, and administrative management of candidates. These institutions must manage several types of information such as candidate registration, monitor assignments, training schedules, examinations, and payments. Efficient management of these activities is essential to ensure a smooth learning process and proper organization of the driving school.

The Problematic

Many driving schools still rely on traditional management methods such as paper records, phone calls, and spreadsheets to manage candidates, schedule lessons, organize examinations, and track payments.

These traditional methods often lead to several problems

- Difficulty in managing candidate information.
- Poor organization of driving lesson schedules.
- Lack of centralized data management.
- Risk of data loss or errors.
- Difficulty tracking candidate progress.

These challenges highlight the need for a modern digital system that can simplify and automate the management process.

Proposed Solution

To solve these problems, the proposed solution is the development of a web-based platform for driving school management. This website aims to provide a centralized platform that allows managers to manage candidates, monitors, training sessions, exams, and payments more efficiently. The system will also facilitate communication between different users of the platform, (here is one of our inspirations, there are foreign websites [w3] [w2]).

Objectives of the Project

The main objective of this project is to design and develop a website that improves the management of a driving school. The specific objectives include

- Simplifying candidate registration and management.
- Organizing driving lesson schedules.
- Managing monitors.
- Tracking candidate progress during training.
- Providing an efficient and user-friendly system for managers.

CHAPTER 1 WEBSITE DEFINITION

1. Introduction

This chapter provides the foundational knowledge necessary for understanding the website developed as part of this project. Before presenting the specific design, implementation, and evaluation of that website, it is essential to establish a clear understanding of what websites are and how they are typically constructed.

2. What is a Website

A website [w1] is a collection of web pages that runs on a remote server and is accessed through a web browser over the internet. Unlike traditional desktop applications that must be downloaded and installed on a user's computer, website do not require installation. This makes them accessible from any device with a browser, including smartphones, tablets, and desktops, providing a consistent cross-platform experience.

3. Key Characteristics of Websites

Websites possess distinct characteristics that set them apart from traditional softwares

3.1. Server-Based Processing

Most of the heavy lifting happens on a server, not on your device. When you use a website, your computer mainly handles displaying the interface, while the server deals with tasks like saving data, processing information, and running calculations. This is why you can use a simple laptop to access powerful software running on remote servers.

3.2. Easy Updates

Because the website lives on a server, developers can update it in one place, and all users automatically get the latest version the next time they log in. There is no need to download and install updates manually, which is a huge convenience for both users and developers.

3.3. Cross-Platform Compatibility

Websites work on any device with a browser, regardless of its operating system. Whether you are using Windows, macOS, Linux, or even a Chromebook, the experience remains largely the same.

3.4. Network Dependency

Websites need an internet connection to function. They communicate using HTTP, which is the standard protocol for transferring data on the web.

4. How Websites Function

A website [w8] [w9] operates on a classic client-server model. The client (your web browser) sends requests to a server, which processes them and sends back a response. The process can be broken down into a cycle of steps

1- User Request the user performs an action in the browser, such as clicking a link or submitting a form. This action sends an HTTP request over the internet to the website's web server.

2- Server Processing the web server receives the request and forwards it to the web server. This is the "brain" of the operation, where the backend logic handles the request. This might involve checking user credentials (identification information), performing calculations, or interacting with other services.

3- Data Interaction if the request requires data (e.g., loading a user's profile), the server queries the database. It may also communicate with internal or external services via APIs (Application Programming Interfaces) to fetch or push data.

4- Response and Rendering the server packages the processed data and sends an HTTP response back to the browser. The browser then renders this response, typically in the form of HTML, CSS, and JavaScript showing you the updated information or confirming that your action was successful.

This entire cycle happens in seconds, creating a smooth experience for the user.

5. Types of Website

5.1. Static Websites

Static websites are built exclusively using front-end languages such as HTML and CSS. They consist of pre-built pages that are created once by the developer using an HTML editor and then stored directly on the web server. When a visitor requests a page, the server simply delivers that exact pre-existing file without any modification.

5.2. Dynamic Websites

Dynamic websites as we implemented in this driving school project are significantly more sophisticated than their static counterparts. In addition to standard HTML and CSS, they rely on server-side programming languages (such as PHP), database query languages (like SQL), and database management systems (for instance, MySQL).The core principle is that the HTML pages are not pre-created. Instead, they are generated on the fly each time a user visits the site. The server processes the request, fetches and filters relevant data from the database, and then assembles that information into a fully formatted page before sending it to the visitor's browser.

6. The Architecture of a Website

Website's architecture is the structural framework that defines how its client-side and server-side components interact, communicate, and work together to handle user requests and deliver responses. A well-designed architecture is crucial for a website's performance, security, and scalability.

6.1. Core Components of the Architecture

A dynamic website is built from several key components

6.1.1. Frontend (Client-Side)

The frontend [w6] is the presentation-tier that users see and with which they interact. It is built using foundational web technologies like HTML (structure), CSS (styling), and JavaScript (interactivity). Modern frontend development often uses frameworks like React, Angular, or Vue.js, which help developers build complex and responsive user interfaces more efficiently.

6.1.2. Backend (Server-Side)

The backend [w6] is the application-tier and founded in dynamic website. This layer contains the core business logic of the website. It handles user authentication, processes data, and manages server configurations. It is written in server-side languages such as Python, Java, Node.js, or PHP, often with the help of frameworks like Django, Spring Boot, Express.js, or Laravel.

6.1.3. Database (Persistence Layer)

The database [w5] is the data-tier and also founded in dynamic website. This layer is responsible for storing, organizing, and retrieving all website data, such as user profiles, transactions, and content. Databases can be SQL (e.g. PostgreSQL, MySQL) or NoSQL (e.g., MongoDB).

6.1.4. Web Server

The web server handles incoming client requests and delivers responses. It acts as a gatekeeper, managing traffic and serving static content like images and CSS files, while passing dynamic content requests to the server.

6.1.5. API

API (especially RESTful APIs or GraphQL) are the messengers that allow the frontend and backend to communicate in a dynamic website. They provide a defined set of rules that allow different software components to interact and share data.

6.2. Common Architectural Models and Types

The components can be organized in different ways.

6.2.1. One-Tier Architecture

This structure is applicable in static or dynamic website, it contain the presentation-tier (frontend), the application-tier (backend) and the data-tier (database) in same code space. The advantage is that it can be used in simple project. The disadvantage is in complex work, it can easily complicate the workflow and the advancement of the project because if we modify in a layer we should do it in other ones.

6.2.2. Two-Tier Architecture

This structure called client-server contain the presentation-tier (frontend) and application-tier (backend) in same code space (client-side), the data-tier (database) is in another space (server-side) .The communication happen between client and server with the backend. The advantage is that the client is separated from server side which maintain a certain level of debugging flexibility. The disadvantage is that if we change in application or presentation we are obliged to adapt the modification to the other part of the client.

6.2.3. Three-Tier or N-Tier Architecture

This structure is one of the most used structure, it's named the layered architecture. The project is separated into three logical-tiers the presentation-tier (frontend), the application-tier (backend), and the data-tier (database). For more complex websites, this can be extended to an n-tier architecture**,** adding layers like an integration tier to manage communication with external services.

Beyond layering, several high-level architectural patterns have emerged to address different needs.

6.2.4. Monolithic Architecture

In this approach, all parts of the dynamic website frontend, backend logic, and data access are bundled together into a single unit. This is simpler to develop initially but can become unwieldy and difficult to scale as the website grows.

6.2.5. Microservices Architecture

The dynamic website is broken down into a collection of small, independent services. Each service is responsible for a specific business function (e.g., user authentication, payment processing) and can be developed, deployed, and scaled independently. These services communicate with each other via APIs. This is ideal for large, complex websites but adds operational complexity.

6.2.6. Serverless Architecture

Despite the name, servers are still involved. However, developers do not have to manage them. Instead, they write code despite the type of website and after upload it to a cloud provider (like Amazon Web Services or Google Cloud), which automatically handles running the code and scaling it based on demand. This approach lets developers focus purely on writing code without worrying about server maintenance.

7. Conclusion

This chapter has established the foundational concepts necessary for understanding websites. We began by defining websites, how they function. Next, the chapter examined the core characteristics that define websites and the basic principles governing their operation, this included an explanation of the client-server model, which forms the foundation of almost all web-based systems, and the request-response cycle that occurs each time a user interacts with a website. Finally, the chapter explored website types and architectures, several architectural approaches have emerged over the years, each with its own strengths and trade-offs. By the end of this chapter, readers should have a solid understanding of websites fundamentals. With this foundation established, we are now prepared to examine the specific website developed for this project.

CHAPTER 2 ANALYSIS AND CONCEPTION

1. Introduction

In this chapter, we present the modelling of our system using the UML language [w7], based on the UP (Unified Process). Consequently, we will detail the following five steps. First, we will define the use case diagram. Then, the use cases will be detailed in several sequence diagrams. We will next, represent the class diagram, which describes the structure of the studied system. Following this, we will develop the MLD (Logical Data Model) to bridge the gap between the object-oriented design and the relational structure. Finally, we will conclude with the implementation of the database.

2. UML Modelling Tool

There are several UML modelling tools available as paid or free software, as well as online platforms. These include Power AMC, Describe, Model maker, Argo UML, and Draw.io; and in our drawing, we use the last tool “Draw.io”.

Draw.io (also known as diagrams.net) is a free, open-source online diagram software for making flowcharts, process diagrams, and UML models. It is highly regarded for its accessibility, as it requires no account registration and integrates seamlessly with cloud storage services like Google Drive, OneDrive, and GitHub. This tool provides a powerful and flexible alternative to proprietary software, offering a wide range of shape libraries specifically designed for software architecture and database schema design.

3. System Actors

The information flow analysis allowed us to determine three actors

Manager

Monitor

Candidate

4. Use Case Diagram

The use case diagram allows for the visual representation of a sequence of actions performed by a system, it answers ‘What this actor do? It is represented by

Rectangular box represent a single system (system boundaries),

Primary actor located on the left, represent the entity that initiates the system.

Secondary actor located on the right, it provides a service to the system,

Oval represent the use case themselves ‘Actions’

Lines / Arrows represent the relationships between actors and use cases.

Figure 2.1. the result of the candidate requirements analysis, obtained after several meetings and interviews regarding the school's operational workflow—specifically regarding candidate reservations and lesson scheduling—can be schematized in the form of a use case diagram.

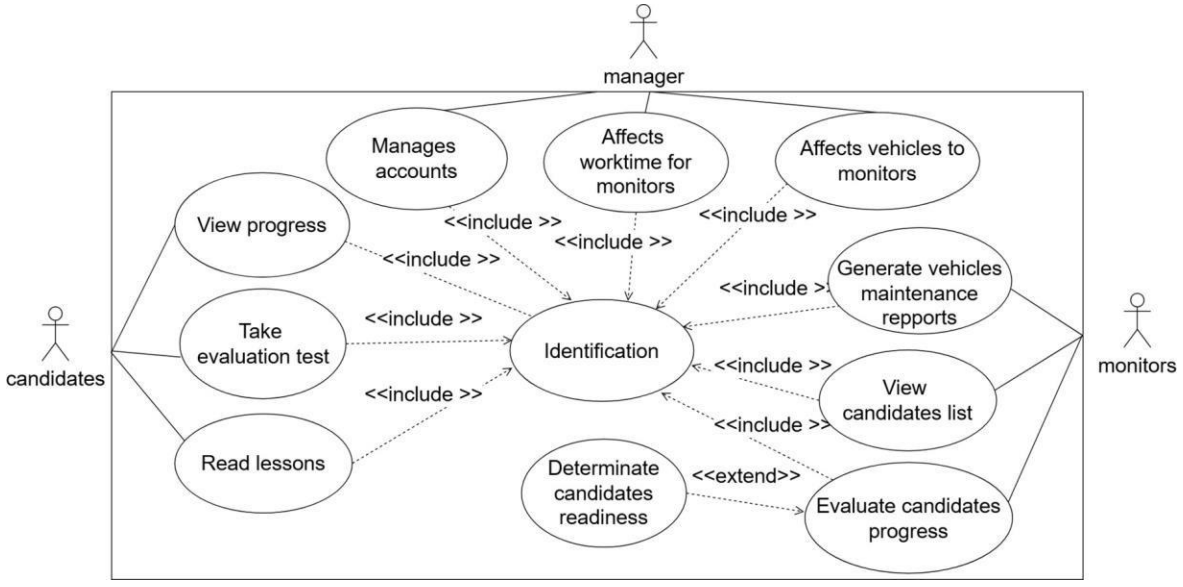


FIG 2.1. The Global Use Case Diagram of our Driving School System

All actors are inherited from “Viewer” actor where they all share the same viewer actions such as view profile, schedule, home page…etc. in the figure 2.2.

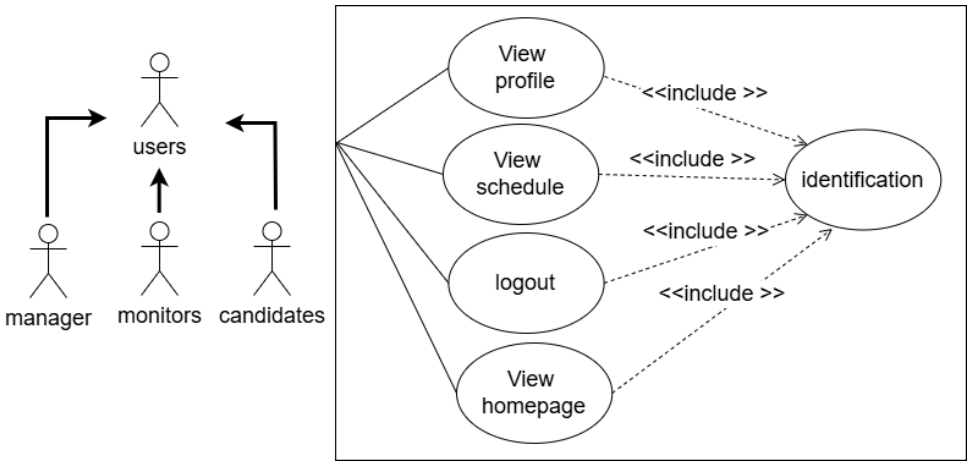


FIG 2.2. Viewer Use Case Diagram

4.1. Candidate Use case Diagram and it's Textual Description

Figure 2.3. represents a use case of all candidate`s actions within the system

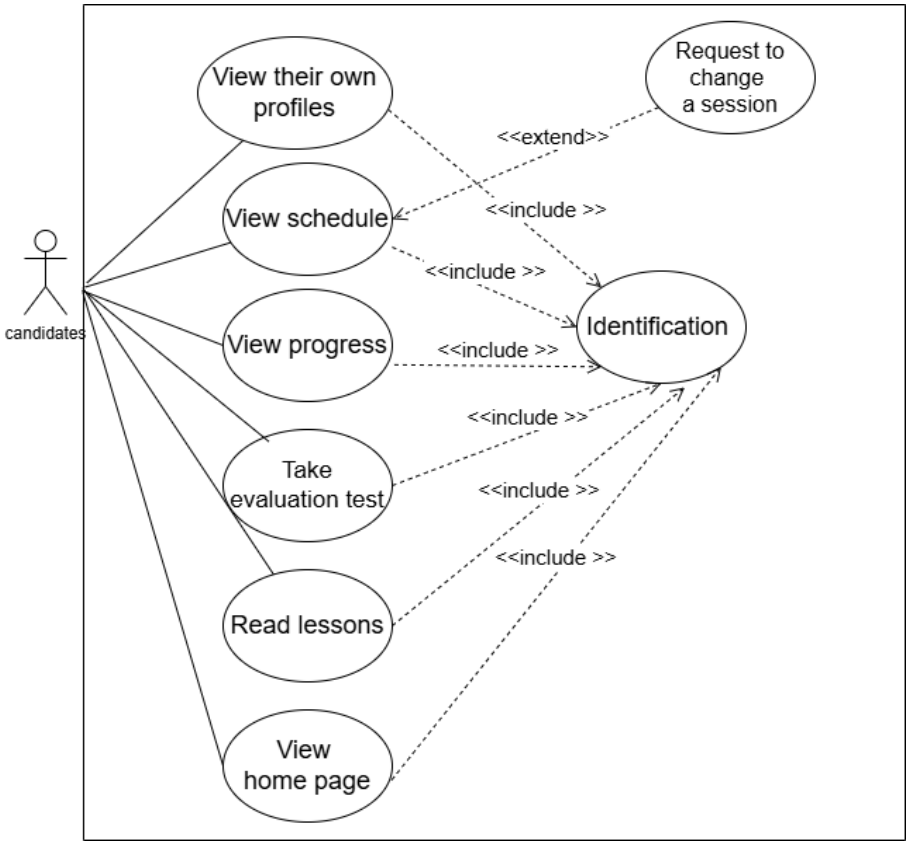


FIG 2.3. Candidate Actions Use Case Diagram

Table 2.1. represents the textual description of the candidate use case

Field	Description
Use case name	Candidate Actions Use Case
Objective	This use case allows the candidate to access and interact with the platform's functionalities such as viewing schedule, profile, progress, lessons, and taking evaluation tests.
Primary actor	candidates
Secondary actor	DB
Pre-condition	The candidate must be authenticated in the system.
Main Scenario	1- The candidate requests access to the system. 2- The system verifies authentication (Identification). 3- The system displays the home page. 4- The candidate selects an action: • View schedule • View profile • View progress • Read lessons • Take evaluation test 5- The system processes the request and displays the corresponding information.
Alternative Scenario	A1: Request session change – The candidate requests to change a session. – The system processes the request and updates the schedule if accepted.
Error Scenario	E1: Authentication failed – The system denies access and displays an error message.
Post-condition	The requested information is displayed or the action is completed successfully.

TAB 2.1. Textual Description of Candidate Use Case

4.2. Monitor Use case Diagram and its Textual Description

Figure 2.4. represents a use case of all monitor's actions within the system



FIG 2.4. Monitor Actions Use Case Diagram

Table 2.2. represents the textual description of the monitor use case

Field	Description
Use case name	Monitor Actions Use Case Diagram
Objective	This use case allows the monitors to manage candidates, evaluate their progress, generate reports, and access system functionalities.
Primary actor	monitors
Secondary actor	DB
Pre-condition	The monitor must be authenticated in the system.
Main Scenario	1- The monitor access the system. 2- The system verifies authentication (Identification). 3- The system displays the home page. 4- The monitor selects an action: • View schedule • View candidate List • View profile • Evaluate candidate progress • Generate vehicle maintenance report 5- The system processes the request and displays results.
Alternative Scenario	A1: Request Session Change – The monitor requests to reschedule a session. – The system updates the session if approved. A2: Determine candidate readiness – The monitor evaluates candidate performance. – The system determines whether the candidate is ready.
Error Scenario	E1: Authentication failed – Access is denied and an error message is displayed. E2: System error – The system fails to process the request.
Post-condition	The selected operation is completed and the system state is updated if necessary.

TAB 2.2. Textual Description of Monitor Use Case

4.3. Manager Use Case Diagram and its Textual Description

Figure 2.5. represents a use case of all manager's actions within the system

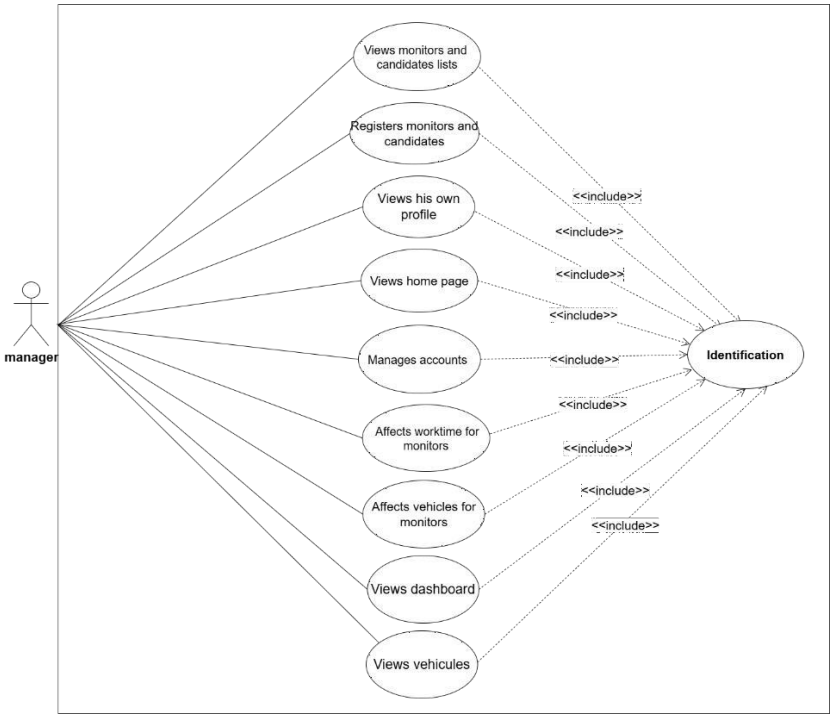


FIG 2.5. Manager Actions Use Case Diagram

Table 2.3. represents the textual description of the manager use case

Field	Description
Use case name	Manager Action Use Diagram
Objective	This use case allows a manager to manage personnel (monitors/candidates), assign resources, and monitor platform activity.
Primary actor	manager
Secondary actor	DB
Pre-condition	The manager must be authenticated in the system.
Main Scenario	1-The manager requests access to a specific management module (e.g. View dashboard or manage accounts). 2-The system triggers the **Identification** process (required for all actions). 3-The system displays the requested interface (e.g. monitors list, planning, or vehicle assignment). 4-The manager performs the desired action (viewing, signing, or assigning). 5-The system verifies and saves the changes in the database. 6-The system displays a confirmation message or the updated view.
Alternative Scenario	A1: Modification of assignment worktime/vehicle - The manager modifies an monitors schedule or vehicle assignment. -The system updates the records and notifies the relevant monitors.
Error Scenario	E1: Database connection failure. - The system fails to retrieve data from the DB. -The system displays an error message: "Data unavailable."
Post-condition	The manager's administrative actions are recorded, and the database is updated.

TAB 2.3. Textual Description of Manager Use Case

5. Sequence Diagram

A sequence diagram is a popular dynamic modelling solution in UML (Unified Modelling Language), it is designed to answer exactly ‘How the system behaves over time’ to complete a task, as they focus specifically on lifelines, processes, and objects that exist simultaneously, and the messages they exchange with each other to perform a function before the end of the lifeline. It is represented by

Lifelines (Vertical Dashed Lines) represent the existence of an object or actor over a period.

Actor Symbols (Stick Figures) usually located on the far left, represents the entity that initiates the interaction.

Activation Bars (Thin Rectangles) placed on top of the lifelines to show exactly when an object is "active" or performing a task.

Horizontal Arrows (Messages) represent the communication between objects, and there are

- Solid Arrow a synchronous call represents a request ‘we know the sender’.
- Dashed Arrow an asynchronous call represents a response ‘we know the receiver’.

Frames (Large Box labelled at top) used to group messages together to show

complex logic that a simple list of arrows cannot capture, and there are many frames type:

alt (Alternative) Chooses one path among several branches based on a condition.

opt (Optional) a single interaction that executes only if a specified condition is true.

loop (Loop) a repeated interaction sequence that continues while a condition is hold.

par (Parallel) multiple interaction fragments that execute concurrently or in any order.

break (Break) used when an interaction flow ends if a condition occurs.

ref (Reference) a placeholder that links to another sequence diagram for reuse.

neg (Negative) interactions that the system explicitly defines as invalid or not allowed.

sd (Sequence Diagram) Defines the overall boundary and scope of the sequence diagram.

5.1. Add Candidate / Monitor Sequence Diagram

Figure 2.6. represents a sequence of adding candidate or monitor by the manager within the system

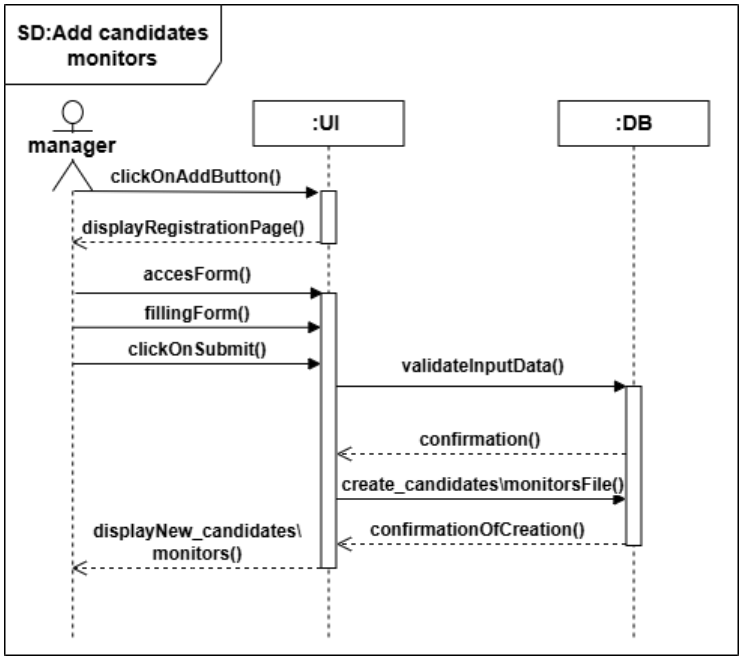


FIG 2.6. Add Candidate/Monitor Sequence Diagram

5.2. Change Session Sequence Diagram

Figure 2.7. represents a sequence of changing session by the candidate or monitor within the system

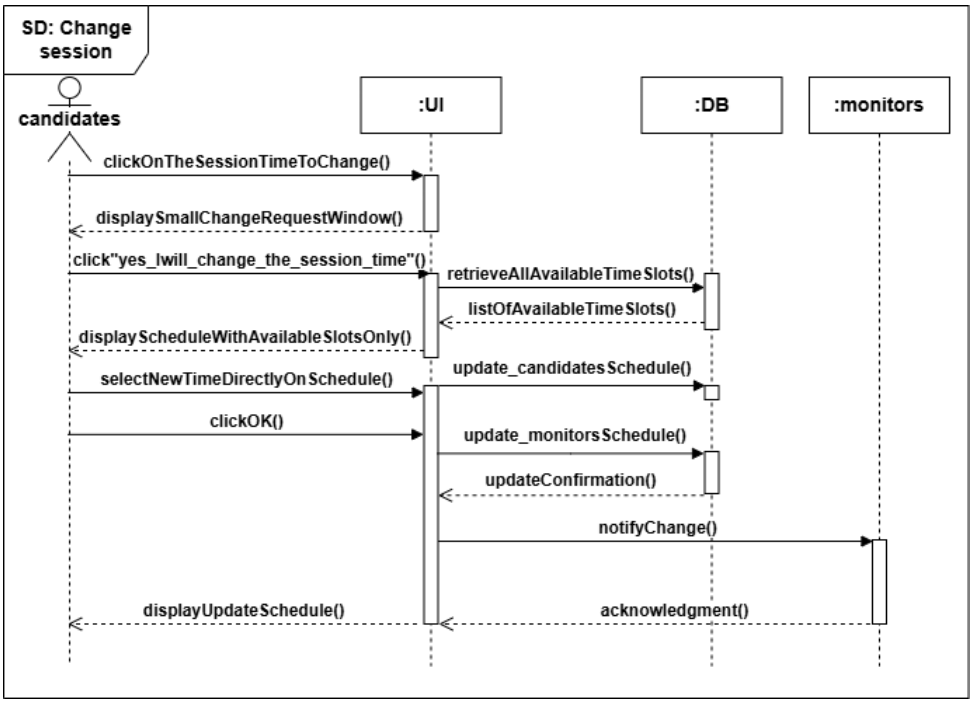


FIG 2.7. Change Session Sequence Diagram

5.3. Affects Worktime for Monitors Sequence Diagram

Figure 2.8. represents a sequence of Affects Worktime for Monitors by the manager within the system

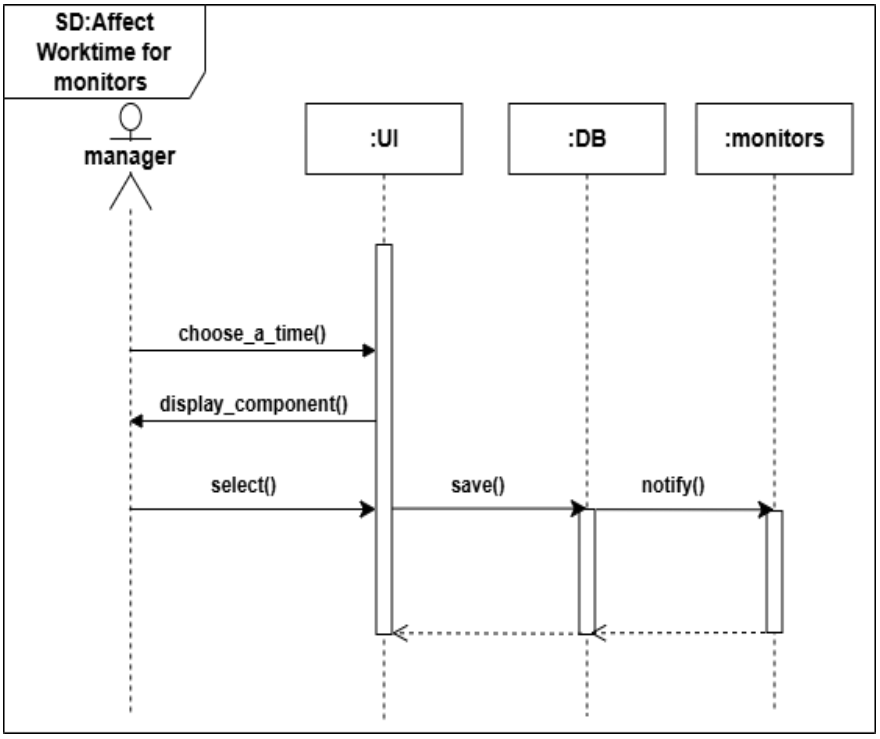


FIG 2.8. Affects Worktime for Monitors Sequence Diagram

5.4. Login Sequence Diagram

Figure 2.9. represents a sequence of login action within the system

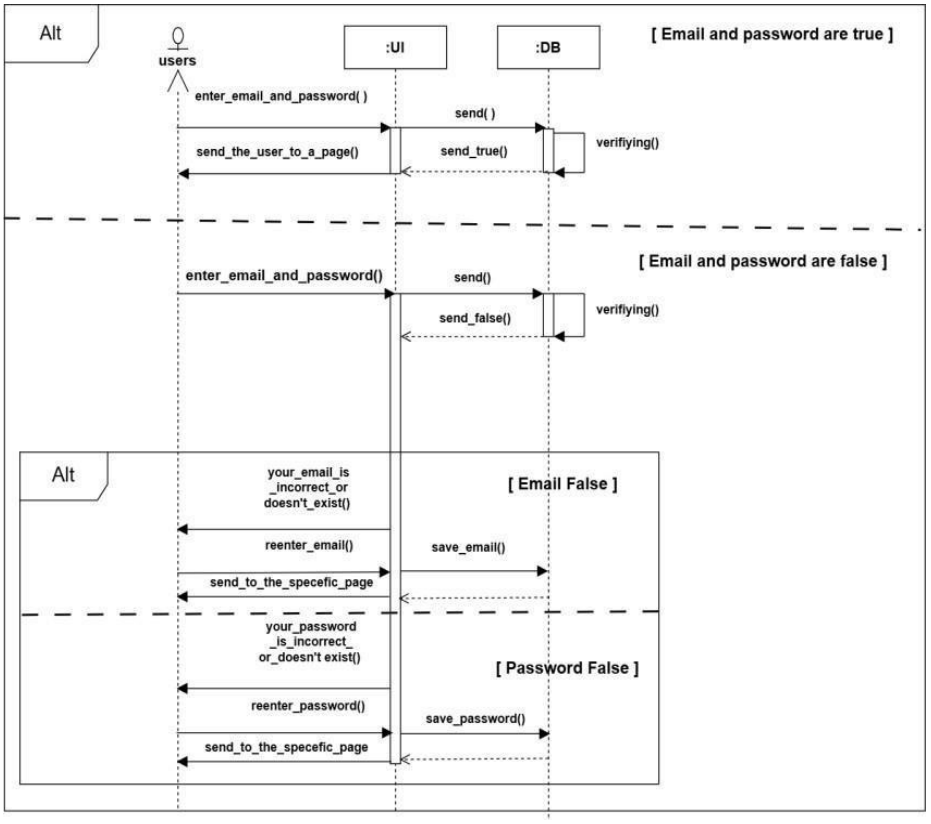


FIG 2.9. Login Sequence Diagram

5.5. Evaluate Progression Sequence Diagram

The Figure 2.10. represents a sequence of evaluate progression of candidate by the monitor

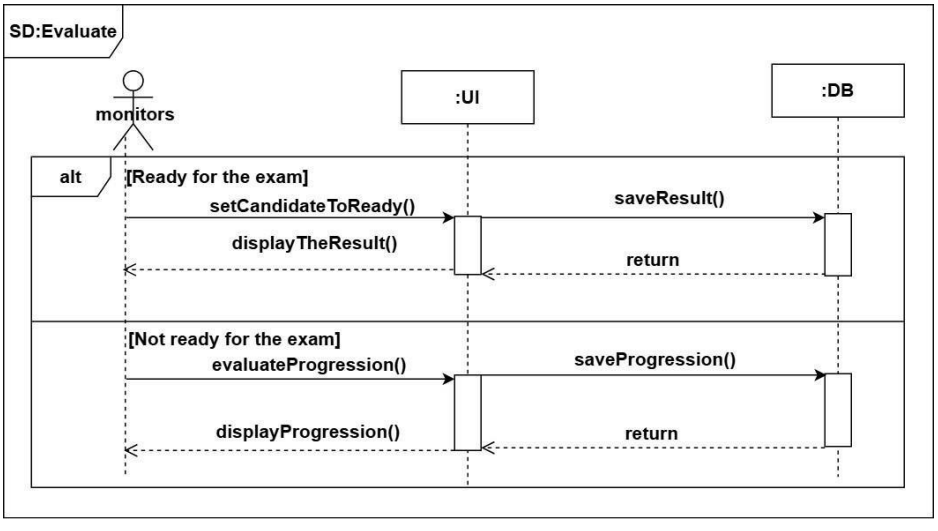


FIG 2.10. Evaluate Progress Sequence Diagram

6. Class Diagram

A class diagram is a schema used in software to present the classes and interfaces of the system and their attribute, methods and their relations. This diagram is part of the static part of UML and are not interested in the temporal and dynamic aspects. It is very helpful for visualize system design, plan object-oriented systems, show relationships clearly, and it can be easily translated into JAVA, which simplifies the development process. The class diagram of our system is presented below figure 2.11.

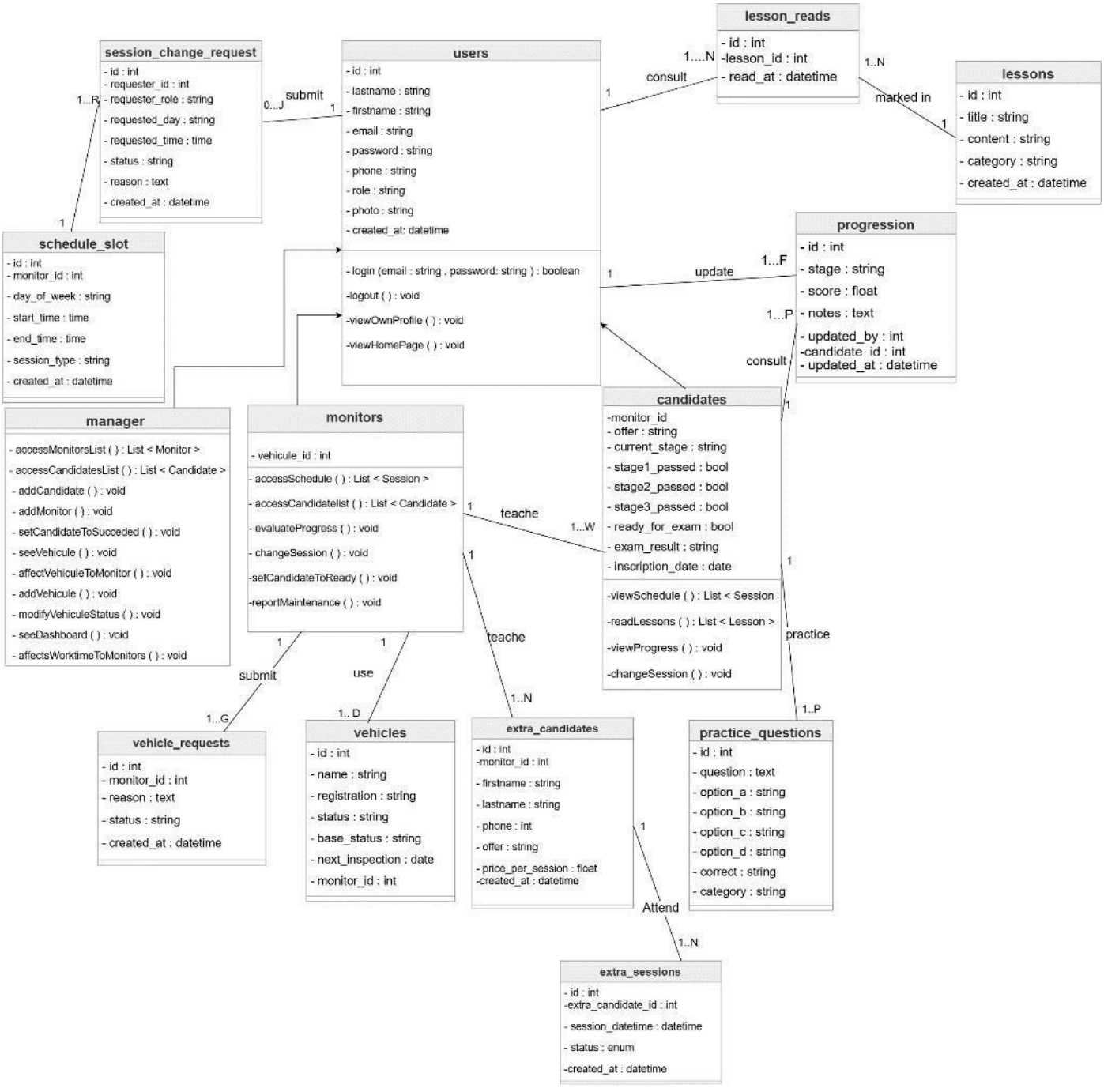


FIG 2.11. The Class Diagram of our System

7. Logical Data Model (MLD)

Logical data model is a structured representation of data that is ready to be implemented in a database, usually in the form of tables, columns and keys. It is important to prepare the system for database implementation, ensuring data consistency, help avoid redundancy and it is directly convertible into SQL tables. Below is the MLD corresponds to the class diagram figure 2.12.

users (id, lastname, firstname, email, password, phone, role, photo, created_at)
candidates (#user_id, #monitor_id, offer, current_stage, stage1_passed, stage2_passed, stage3_passed, ready_for_exam, exam_result, inscription_date)
monitors (#user_id, #vehicle_id)
manager (#user_id)
progression (id, #updated_by, #candidate_id, stage, score, notes, updated_at)
vehicles (id, #monitor_id, name, registartion, status, base_status, next_inspection)
schedule_slots (id, #monitor_id, day_of_week, start_time, end_time, session_type, created_at)
session_change_requests (id, #requester_id, #slot_id, requester_role, requested_day, requested_time, status, reason, created_at)
lessons (id, title, content, category, created_at)
lesson_reads (id, #lesson_id, read_at)
practice_questions (id, question, option_a, option_b, option_c, option_d, correct, category)
vehicle_requests (id, #monitor_id, reason, status, created_at)
extra_candidates (id, #monitor_id, firstname, lastname, phone, offer, price_per_session, created_at)
extra_sessions (id, #extra_candidate_id, session_datetime, status, created_at)

FIG 2.12. Logical Data Model

8. Rules of Conversion from Class Diagram to MLD

Rule 1: Mapping Classes

Every persistent class in the class diagram becomes a table in the relational model.
Each attribute of the class becomes a column in the corresponding table.
The identifier of the class becomes the primary key of the table.

Rule 2: Mapping Inheritance

One table for the parent class (superclass).
One table for each child class (subclass).
The child table takes the parent's attributes with the primary key of the parent class; it is a primary key for the child table and a foreign key in same time referencing the parent table. This primary key is for containing attributes of the parent class.

Rule 3: Mapping a 1-* Association

The primary key of the class, on the 1 side, is added as a foreign key in the table corresponding to the « *» it can be (N, P, R…) side, this created the link between the two tables.

9. Conclusion

UML modelling provides a standardized way to visualize, design, document software systems. Using diagrams such as use case, sequence, class diagram. It bridges the gap between conceptual understanding and implementation by structuring system components and their interactions. Through these models, it facilitates a smooth transition from design to code and database structure. Overall, UML improves communication, reduces errors, and makes system development more organized and efficient.

CHAPTER 3 IMPLEMENTATION

1. Introduction

This chapter is devoted to the practical implementation of our driving school management website, which is founded upon the conceptual modelling and design principles detailed in the preceding chapter. To carry out the implementation phase, we selected the PHP programming language as the primary server-side scripting language, alongside the MySQL relational database management system for efficient data storage and retrieval. Furthermore, to enable local testing and execution of the website, we employed the WAMP64 web server environment, which facilitates the hosting and access of web pages on local machines.

2. Web Technologies Used

The following web technologies were employed to develop the full-featured driving school platform.

2.1. Server-Side Development

Server-side programming involves writing code that executes on the web server, performing computations, retrieving data from databases or external APIs, and generating dynamic content that is sent to the client's browser. This content may also include client-side code such as HTML, CSS, or JavaScript.

2.1.1. PHP



PHP is a widely used, open source scripting language specifically designed for web development. It is executed on the server and is particularly well suited for creating dynamic web pages. In our project, PHP serves as the core language for handling user authentication, session management, database interactions, and business logic. PHP is commonly associated with database systems such as MySQL.

2.1.2. MYSQL



MySQL is a relational database management system (RDBMS) distributed under a dual license (GPL and proprietary). It is one of the most popular database systems worldwide, used by both general public web applications and professional enterprises. The name derives from "My," the daughter of co-founder Michael Widenius, while SQL stands for Structured Query Language. In our project, MySQL stores all application data, including users, candidates, monitors, sessions, vehicles, and change requests.

2.2. Client-Side Development

Client-side programming (front-end development) consists of producing code (HTML5, CSS3, JavaScript), that is interpreted by a web browser to produce a visual user interface (UI). This includes elements that enable navigation and interaction, thereby enhancing the user experience (UX).

2.2.1. Bootstrap 5



Bootstrap is a free, open-source front-end framework originally developed by Twitter and released under the MIT license. Bootstrap 5 provides a comprehensive collection of HTML, CSS, and JavaScript components for rapid development of responsive, mobile-first websites. The Bootstrap5 framework includes predefined styling for typography, forms, buttons, navigation bars, cards, modals, and a 12 column grid system. In this project, Bootstrap 5 is employed as the primary front-end framework to ensure a consistent, professional, and responsive interface across manager, monitor, and candidate roles.

2.2.2. Vanilla JavaScript (ECMAScript)



Vanilla JavaScript is a high-level, interpreted programming language that conforms to the ECMAScript specification, enabling client-side scripting for interactive web features. It facilitates event handling, Document Object Model (DOM) manipulation, form validation, asynchronous operations, and local storage management. JavaScript is natively supported by all modern web browsers without requiring additional plugins or libraries. In this project, Vanilla JavaScript is used for live form validation, progress circle rendering using conic gradients, modal dialog management, auto-save mechanisms utilizing local Storage, and client-side data filtering.

2.2.3. Chart.js



Chart.js is an open-source JavaScript library that enables creation of responsive, animated, and customizable HTML5 charts using the element. It supports various chart types including line, bar, radar, pie, and doughnut charts. Chart.js Chart.js is an open-source JavaScript library that enables creation of responsive, animated, and customizable HTML5 operates entirely client-side, automatically handles responsive resizing, and requires minimal configuration. Within this project, Chart.js is implemented to visualize statistical data and candidate progress trends across manager and candidate dashboards, facilitating data driven decision-making.

3. Development Tools

3.1. WAMP64 (WampServer)



WAMP64 [w4] is a free, open-source, Windows-based local web server environment that bundles three essential components for web development: Apache (HTTP server), MySQL (relational database management system), and PHP (server-side scripting language). The acronym stands for Windows, Apache, MySQL, PHP, while the "64" suffix indicates compatibility with 64-bit Windows operating systems.

3.2. Visual Studio Code



Visual Studio Code (VS Code) is a free, open-source code editor developed by Microsoft in 2015. It is available for Windows, Linux, and macOS, and is built using the Electron framework. VS Code is particularly well suited for web development projects involving JavaScript, Node.js, TypeScript, and PHP**.** In our project, VS Code served as the primary integrated development environment (IDE).

4. Application Description

In this section, we present the main pages and features of the application, organized by user role: manager, monitors, and candidates.

4.1. Home Page

This is the first page of the website (Home), it contains links to the other pages of the site.

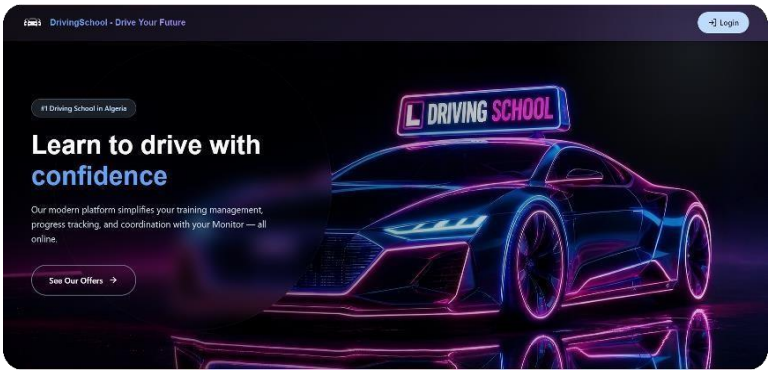


FIG 3.1. Home Page.

4.2. Authentication System Page

These pages manage user login and session handling. Upon successful authentication, the system checks the user's role (manager, monitors, or candidates) and redirects to the corresponding dashboard.

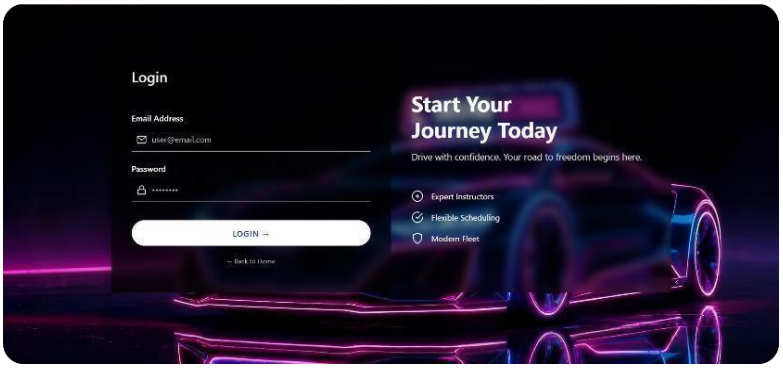


FIG 3.2. Login Page

4.3. User Profile Pages

Each user role has a profile page allowing them to view and edit personal information. The manager profile displays quick statistics about the school. The monitor profile shows assigned candidates and upcoming sessions. The candidate profile includes registration date, selected offer, and assigned monitor name.

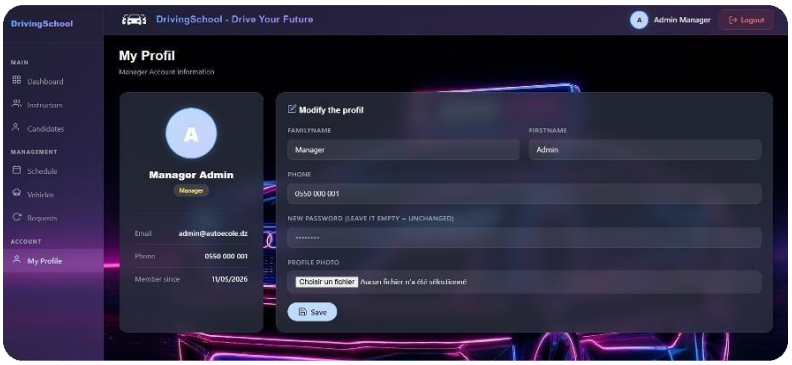


FIG 3.3. Manager Profile Page



FIG 3.4. Monitor Profile Page

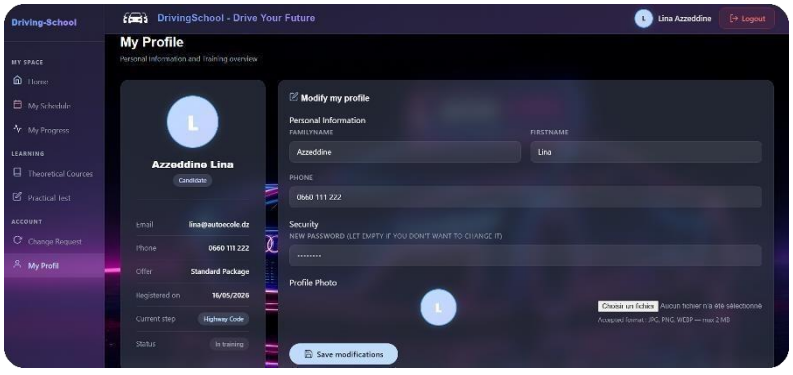


FIG 3.5. Candidate Profile Page

4.4. Dashboard Pages:

The dashboard serves as the main interface after authentication. The manager dashboard displays key performance indicators and quick management actions. The monitor dashboard shows assigned candidates, today's sessions, and current vehicle. The candidate dashboard presents overall progress, upcoming sessions, and quick links to learning resources.

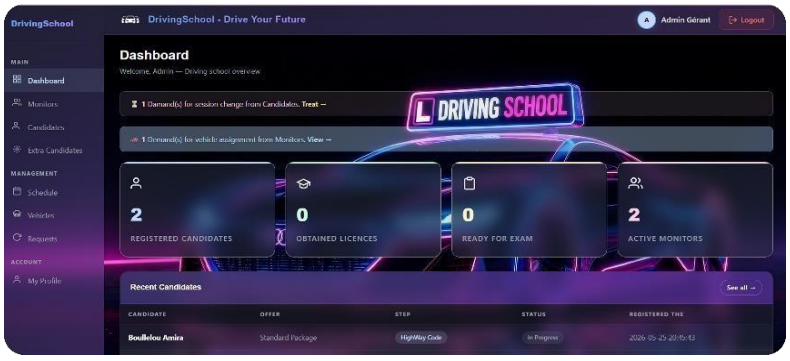


FIG 3.6. Manager Dashboard

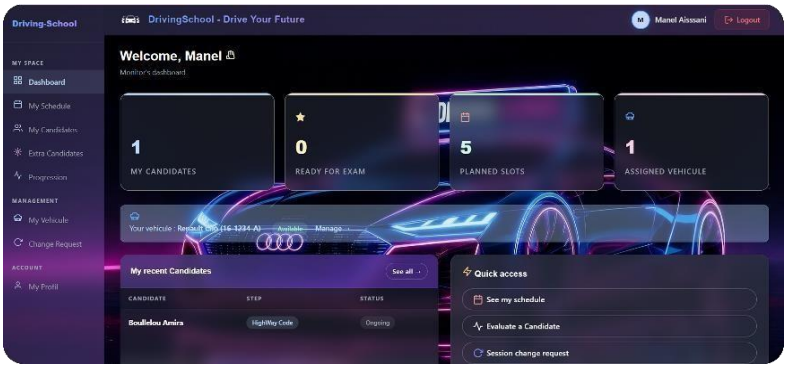


FIG 3.7. Monitor Dashboard

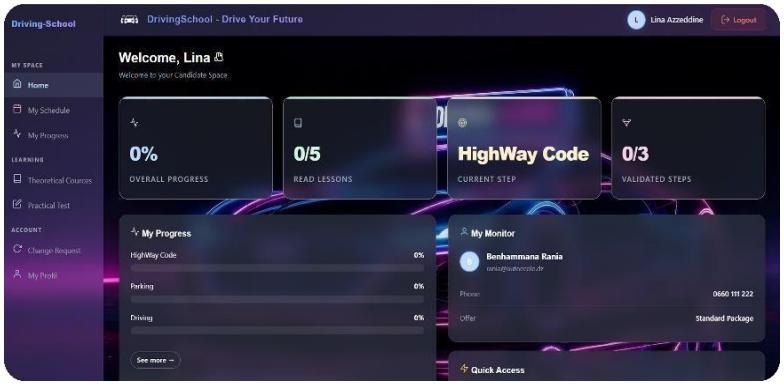


FIG 3.8. Candidate Dashboard

4.5. Schedule Pages:

Schedule pages display upcoming sessions grouped by day. The manager schedule allows creating new sessions. The monitor schedule shows assigned sessions with candidates and vehicles. The candidate schedule shows sessions with monitors and vehicles.

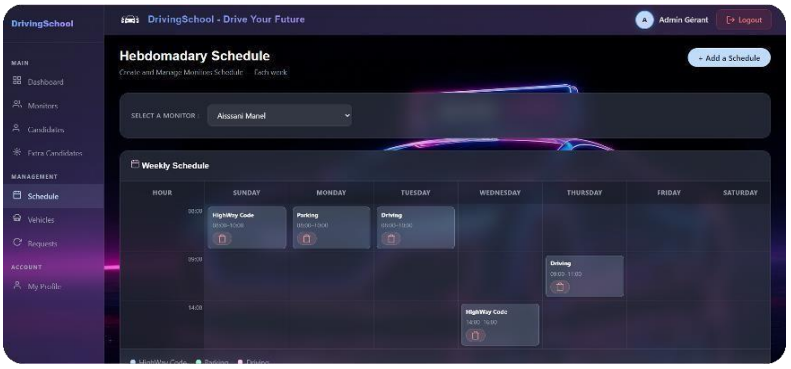


FIG 3.9. Manager Schedule Page

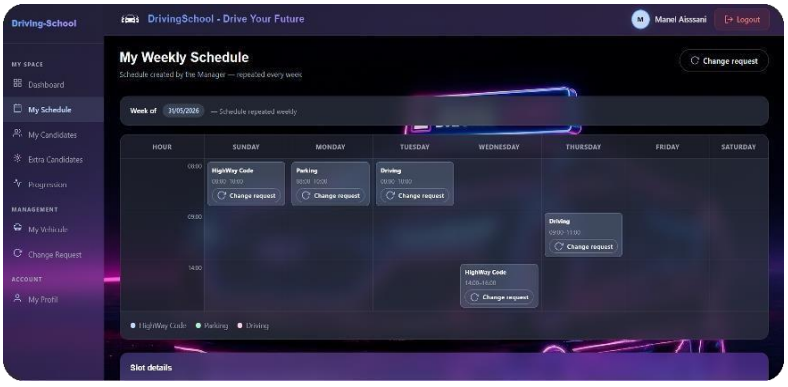


FIG 3.10. Monitor Schedule Page

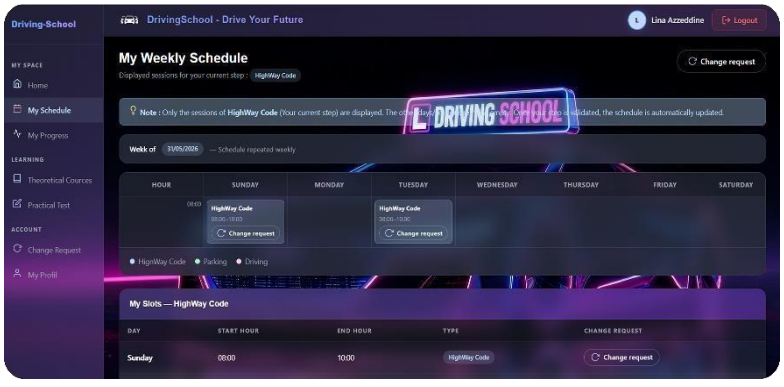


FIG 3.11. Candidate Schedule Page

4.6. Session Change Request Pages

These pages allow monitors and candidates to request changes to upcoming sessions by selecting a new date and time. Requests are stored with status "pending" for manager approval. Users can view their previous requests and cancel pending ones.

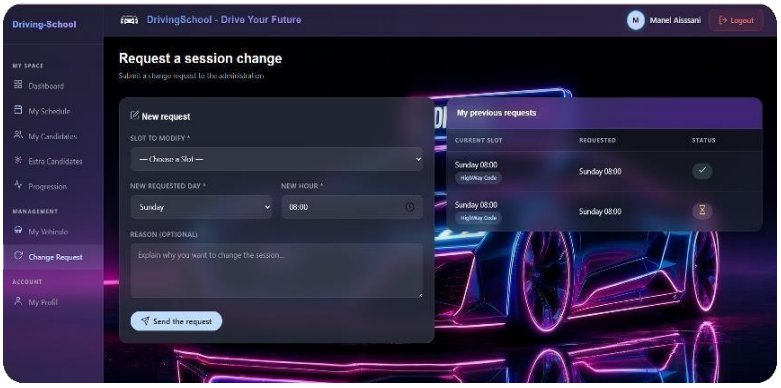


FIG 3.12. Monitor Request Session Change Page

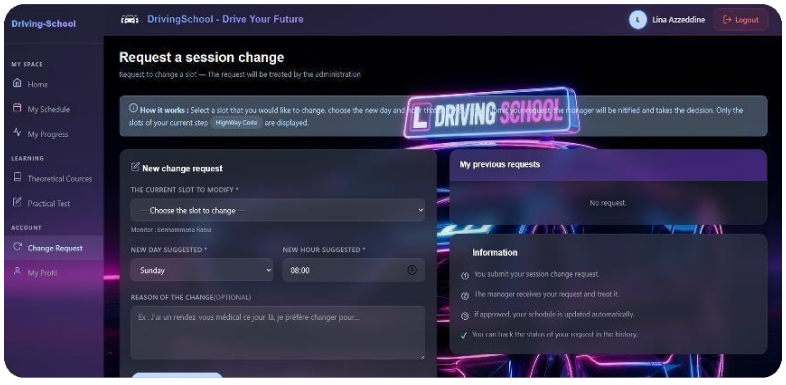


FIG 3.13. Candidate Request Session Change Page

4.7. Candidate Management Pages

The manager page displays all registered candidates with their file numbers, assigned monitors, learning stages, and exam readiness. Manager can assign monitors and update exam results. The monitor page displays only assigned candidates, allowing progress updates and note addition. If all progress areas reach 80%, the candidate is automatically marked as "Ready for Exam.



FIG 3.14. Manager Candidate Management Page.



FIG 3.15. Monitor Candidate Management Page

4.8. Monitor Management Page

This page is exclusive to manager. It displays the list of monitors, their personal information, hire dates, and number of assigned candidates. Manager can add, delete, or assign vehicles to monitors.



FIG 3.16. Monitor Management Page

4.9. Vehicle Management Pages

The manager page provides complete fleet management, allowing addition, status updates, and vehicle assignment to monitors. Weekly schedules and real-time usage status are displayed. The monitor page displays only the currently assigned vehicle, with options to request maintenance or change vehicles.

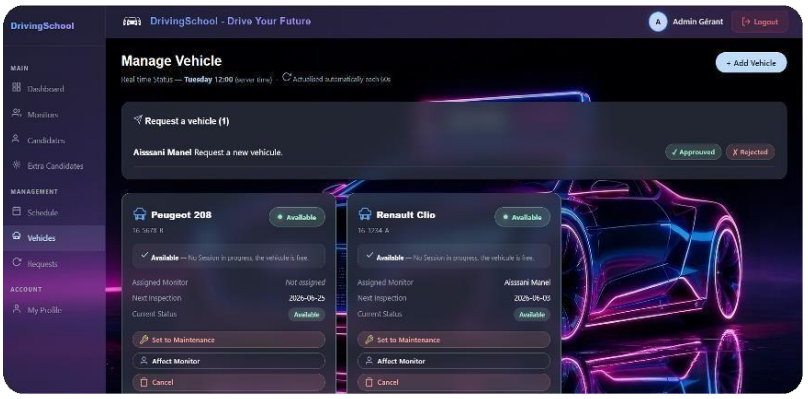


FIG 3.17. Manager Vehicle Management Page



FIG 3.18. Monitor Vehicle Management Page

4.10. Session Change Request Management Page

This page is exclusive to manager. It displays all pending session change requests submitted by monitors or candidates. The manager can approve or reject each request.

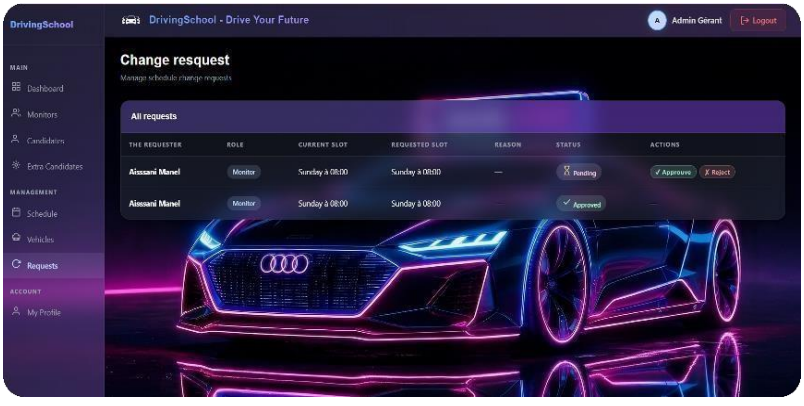


FIG 3.19. Change Requests Management Page

4.11. Candidate Progress Evaluation Page

This page is exclusive to monitors. It allows evaluating candidate progress by assigning scores (0–100) and adding notes for each session. Candidate progress is automatically updated based on the average of evaluated session

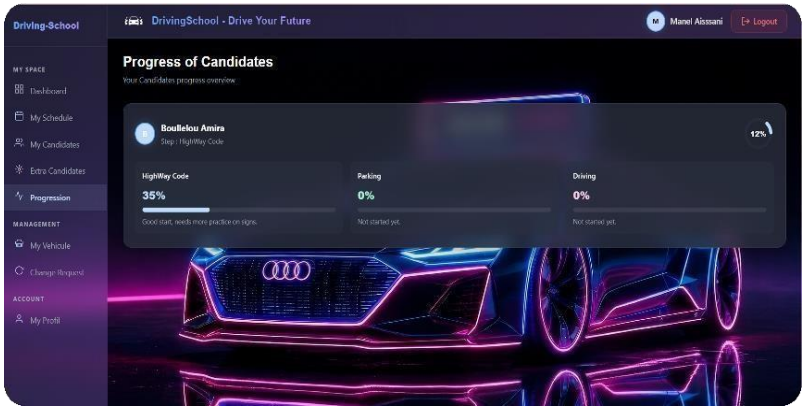


FIG 3.20. Progress Evaluation Page

4.12. Candidate Progress Tracking Page

This page is exclusive to candidates. It displays overall progress, detailed progress bars for Code, Parking, and Driving, a progress evolution chart, monitor notes, and exam results.



FIG 3.21. Candidate Progress Tracking Page

4.13. Theoretical Lessons Page

This page is exclusive to candidates. It allows viewing theoretical lessons categorized as Code, Driving, or Safety. Lessons can be marked as "read," and progress is saved in the database.

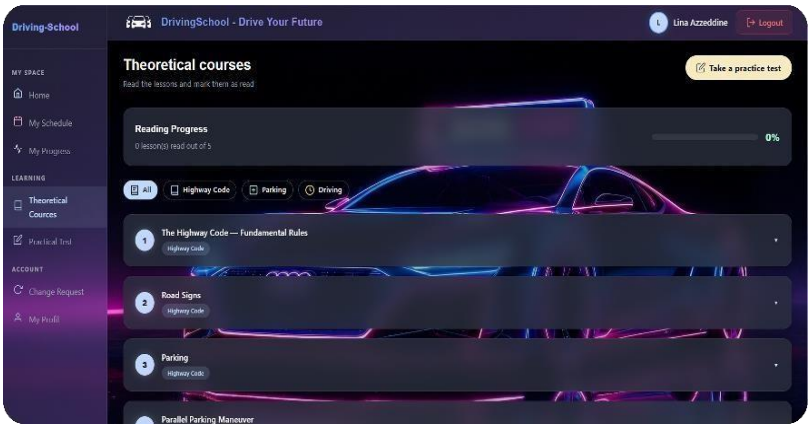


FIG 3.22. Theoretical Lessons Page

4.14. Practice Test Page

This page is exclusive to candidates. It offers a simulated code exam with multiple-choice questions covering traffic rules and regulations. The system automatically grades the test and provides feedback.

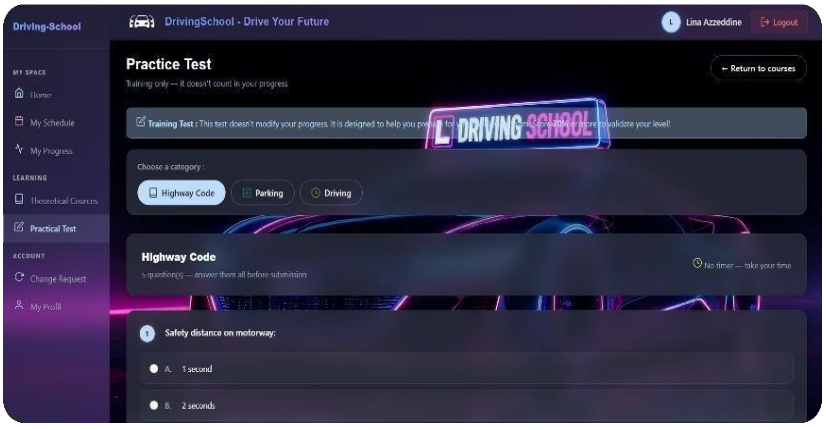


FIG 3.23. Practice Test Page

4.15. Extra Candidates Pages

The first page is exclusive to manager it displays all nonregistered candidates with the offer (Additional Session and Graduated Discount) it organize their information: name, phone, offer, price/session, monitor, date of registration and action .This table is to keep a track on them.

The second page is from monitors, it displays his affected nonregistered candidates with the offer (Additional Session and Graduated Discount) it organize their information: name, phone, offer, price/session, and upcoming sessions.

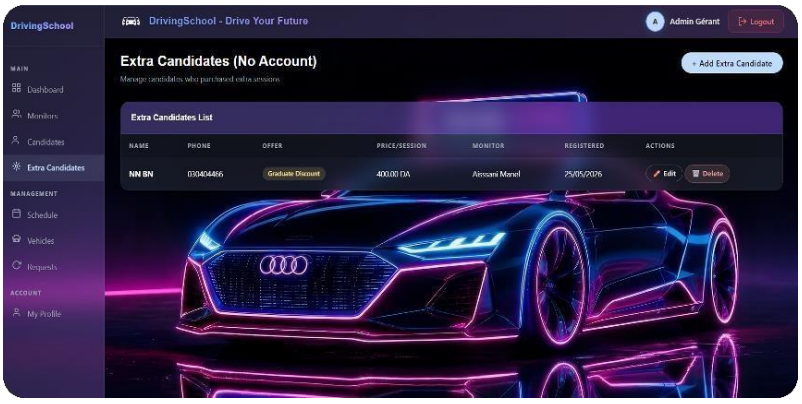


FIG 3.24. Extra Candidates (No Account)

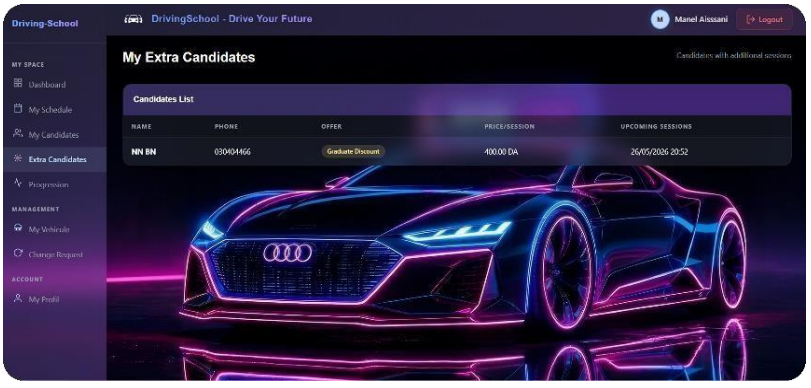


FIG 3.25. My Extra Candidates

5. Conclusion

In this chapter, we presented the complete implementation of our driving school management website using PHP, MySQL, and WAMP64. The website was structured around three user roles: manager, monitors, and candidate. For manager, we implemented monitor and candidate management, vehicle and session scheduling, and change request processing. For monitors, we developed schedule viewing, candidate progress evaluation with scores and notes, and vehicle management. For candidates, we provided dashboards, progress tracking, theoretical lessons, a practice test system, and session change requests. The database schema ensures data integrity through foreign keys and automated events.

GENERAL CONCLUSION

The launch of an Internet-based management system will respond directly to the problem of traditional management practices within the driving school industry. This new software project will address specific management needs, with one unified, efficient, easy to use website, which will improve all driving school core processes, such as registering candidate drivers, tracking driving school monitors, scheduling classes, and tracking payments.

By utilizing current web technology, driving schools will be able to eliminate the negative aspects of paper-based processes, reduce the likelihood of data loss, improve communication among users, and provide a means of tracking and monitoring candidate progress throughout their training programs as they train to become drivers. The proposed new website will automate repetitive routine administration tasks and provide all participants in the process with a structured and easily accessible manner in which to complete their responsibilities.

This project supports the broader trend of digital transformation in education and administration and demonstrates how successful websites can lead to noteworthy enhancements for daily organizational problems. Details regarding the driving school management system project's analysis, design, and implementation will be provided in the succeeding chapters of this report, along with verification that this innovative solution was developed to meet the needs of the driving school and all of its stakeholders.

The implementation successfully met all functional requirements identified during the analysis phase. The experience allowed us to apply theoretical knowledge in web development and overcome challenges such as automated vehicle status updates.

As future perspectives, the website could be enhanced with online payment integration, mobile versions for Android and iOS, real-time SMS or email notifications, and a workflow engine to automate information flows within the school.

WEBOGRAPHY

[w1] Definition of a Website

(What is a Website - GeeksforGeeks)

[w2] Inspiration Website1 “Klaxo driving school management website”

(https://share.google/mM6w7rLiJDvEbTrdU)

[w3] Inspiration Website2 “Glitz driving school management website”

(https://share.google/uqKVBvLfboziZ7JDr)

[w4] Tutorial Wamp
(https://www.wampserver.com/en/documentation)

[w5] SQL Database Course

(https://elearning.univ-boumerdes.dz/course/view.php?id=28130)

[w6] Frontend and Backend Development
(https://elearning.univ-boumerdes.dz/course/view.php?id=28133)

[w7] Software Engineering Course
(https://elearning.univ-boumerdes.dz/course/view.php?id=24464)

[w8] Definition and Components of Dynamic Websites and Applications
(Web Application Architecture: Guide 2025 | Implex)

[w9] Generalities about Websites
(https://www.godaddy.com/resources/skills/what-is-a-website)