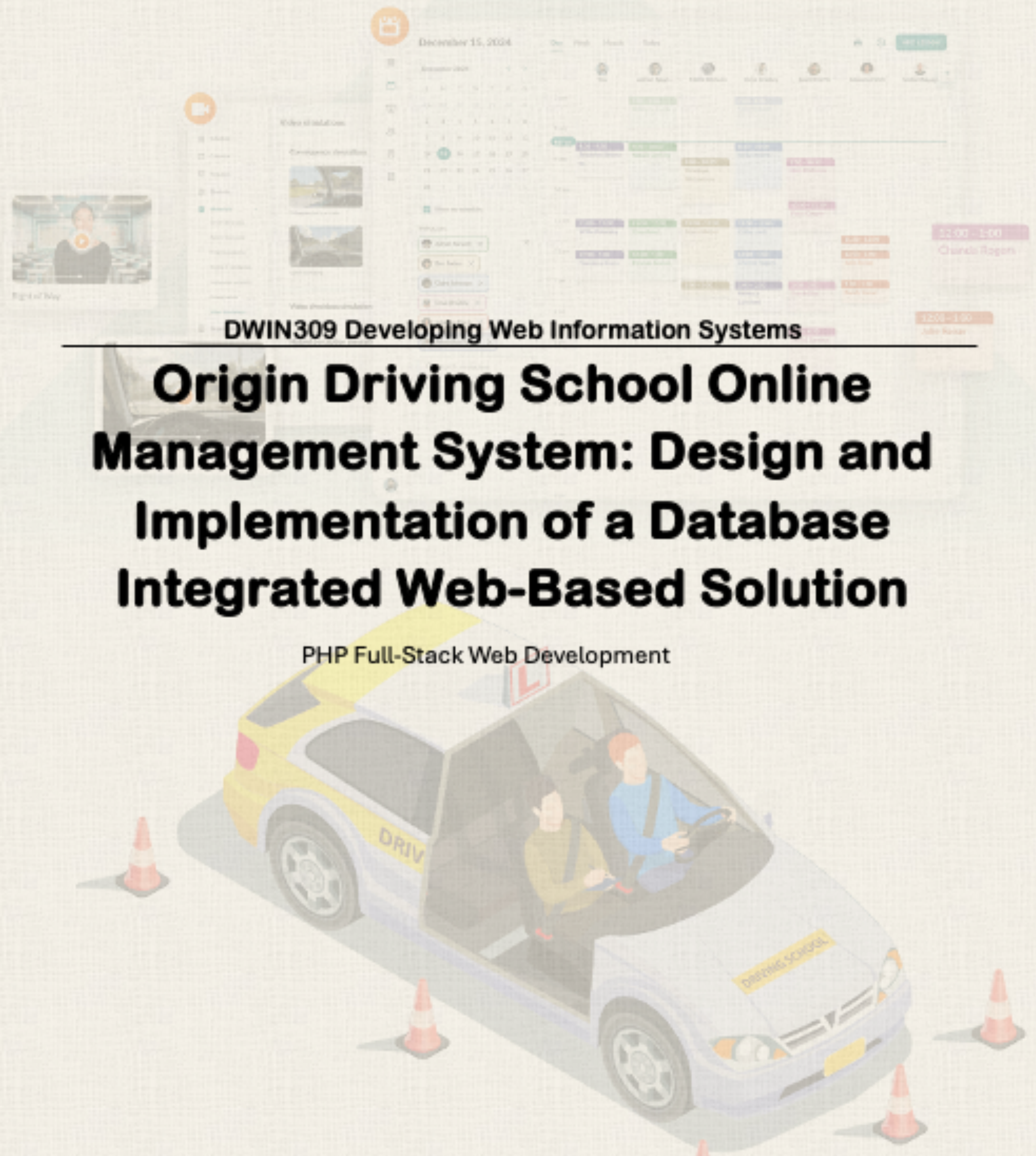

DWIN309 Developing Web Information Systems

Origin Driving School Online Management System: Design and Implementation of a Database Integrated Web-Based Solution

PHP Full-Stack Web Development

Group Members:
SUJAN DARJI (K231673) – Group Members
ANTHONY ALLAN REGALADO (K231715)

Table of Contents

1. INTRODUCTION

1.1 Project Overview

The Origin Driving School Online Management System (ODS-OMS) is a web platform built with PHP, MySQL, HTML, CSS besides JavaScript - it moves all Origin Driving School operations from paper to a single website. The site stores student, instructor, course, payment and schedule data in a central database and updates the records instantly. The code follows a modular layout that borrows from the MVC pattern. Models talk to the database; views render pages plus controllers handle requests - each part can be swapped out or expanded without breaking the rest. A user will visit the login page and enters credentials. The system checks the role (administrator, instructor, student or staff) then it will open the matching dashboard. Inside the dashboard, the user will add, edits or deletes records, prints invoices and downloads reports. Authentication blocks unauthorised access. CRUD routines create, read, update but also delete entries. Invoice scripts generate bills automatically. Analytics scripts count bookings, income and pass rates. Those features give any driver training school a live, paperless back office. The project exemplifies best practice in data-driven web development, combining user-centred interface design with robust backend logic to achieve operational efficiency, data integrity, and a seamless user experience (Pressman & Maxim, 2020).

1.2 Project Objectives

The goal of ODS-OMS is to design a secure, user-friendly, and fully functional database-integrated website for Origin Driving School automating its daily operations. The aim of the system, for example, is to:

- Digitise all management of students, instructors, and courses for the accurate, real-time management of records.
- Offer automatic scheduling, avoiding double bookings along with flexible scheduling of lessons.
- Provide transparent, audited, and auditable means of generating invoices and maintaining payment records.
- Facilitate in-system messaging and notifications for communication.
- Provide analytics dashboards for managers to make decisions in accordance with live data.

More secondary goals aim to ensure data is secured, supported in this step by encrypting authentication data, improving operational efficiency, and building up a scalable environment where further development such as API integration and SMS notification can take place. In accomplishing these goals, ODS-OMS serves as an administrative automation instrument as well as a pedagogical support solution simplifying the tracking of learner progress and the daily organization of the institution (Sommerville, 2016).

1.3 The System Purpose and Scope

ODS-OMS is designed as an all-in-one, data-driven, collaborative solution that would be required to manage the educational, financial, and operational work and process work at Origin Driving School to be able to operate an organisation such as Origin Driving School. It spans everything from student sign ups to instructors' time slots, lesson scheduling to time management, lesson bookings and financial reporting and booking/financial reporting. It can be used for adding, updating, and retrieving data through structured database queries and secure role-based access control. Administrators can handle everything from operations - Teachers can deal with schedules and student feedback, and Students have their own dashboards to monitor their lessons and payments. The platform's backend is MySQL and PHP based to ensure that data is persisted securely and that its front end is built in HTML5, CSS & JavaScript for fast and responsive interactions. The solution includes an integrated management platform to support attachments, notes, and digital communication. In other words, ODS-OMS is an integrated web application that contributes to administrative efficiency, learner engagement, and institutional accountability in the transition to digital education (Laudon & Traver, 2022).

1.4 Target Users and Stakeholders

ODS-OMS, the primary users of ODS-OMS are the administrators, instructors, staff members, and students of Origin Driving School. They do so by using the system to manage their branches, monitor financial performance, and maintain institutional records. Instructors tap into their teaching schedules, log lesson outcomes, and interact with learners. Students access their dedicated portal to book lessons, review progress, and process payments securely online. Involving stakeholders are the IT support staff that holds onto the system, and possible external providers like payment gateways or SMS APIs for future integrations. At the corporate level such a system can help management as it provides data reporting services on business analytics, compliance monitoring, and decision support. With clearly defined role hierarchies and access control, ODS-OMS establishes appropriate roles and permissions and by the application of data modules, each of the stakeholders can interact only with data modules that relate directly to their requirements (Turban, Pollard & Wood, 2018), which enhances accountability, security, and operational transparency (Turban, Pollard & Wood, 2018).

1.5 Project Motivation and Problem Background

Before the implementation of the system, Origin Driving School relied on manual record-keeping and physical scheduling methods, which were error-prone, inefficient, and difficult to access remotely. Manual procedures led to missed appointments, data loss, and administrative delays — all problems that plague small to mid-sized educational enterprises (Laudon & Laudon, 2020). As enrolment increased and branches were set up across Victoria, it was clear that a centralised, web-based management solution was needed. The motivation for creating ODS-OMS was to modernise the school's operations with a secure and real-time data management system. With the combination of scheduling, financial, and communication tools, the system reduces administrative workload and enhances reliability. Furthermore, it is also indicative of contemporary trends in digital transformation in the education as well as the transport training sector where efficiency, transparency, and accessibility are crucial for the competitiveness and quality of services (Stair & Reynolds, 2021).

1.6. Expected Outcomes

The implementation and distribution of the Origin Driving School Online Management System are anticipated to provide the following high value, showing proficiency of engineering quality and cutting-edge design:

I. **Operational Digitalisation:**
All administrative and instructional processes such as student registration, scheduling, invoicing, and reporting are completely digitalised, avoiding paper-based inefficiency, and providing instant, error free access of data at each branch.

II. **Centralised Real time Management:**
Centralised database and dashboard architecture allow administrators to dynamically track lessons, payments, instructor availability, and fleet utilisation and so improve decision-making.

III. **Automation and Accuracy:**
The automation of invoice processing, payment reconciliation, and lesson reminders helps reduce your manual workload and increases monetary accuracy and compliance.

IV. **Data-Driven Insights:**
Reporting and analytics dashboards: Making the data from the shop floors convert to insights on student performance, instructor productivity, and business growth trends.

V. **Enhanced User Experience:**
By implementing HTML5, CSS, and JavaScript to create a mobile-friendly interface that is consistent and easy to use for all user roles (so long as they can navigate the application), increasing engagement and supporting users.

VI. **Strong Security and Reliability:**
Having strong password hashing, input validation and role-based access controls is a guarantee of high security and data integrity that meets modern web security standards.

VII. **Scalability and Maintainability:**

The scalable PHP-MySQL modular design makes it very easy to scale, in the future with payment gateway/APIs/cloud hosting solutions without having to drastically redevelop a new architecture.

VIII. **Professional Benchmark:**
The proposed prototype provides an academic benchmark as well as a demonstration of contemporary web engineering principles by integrating database normalisation, MVC features, and user-oriented design to define a system manager with the ability to manage an enterprise online space.

2. SYSTEM ANALYSIS

2.1 Problem Definition and Existing System Issues

The current operation of Origin Driving School was dominated by manual booking processes, paper documents as well as paper invoicing causing redundancies that led to inconsistency in day-to-day work. Student registration, teacher assignment, lesson scheduling and payment fulfillment was conducted on notebooks and spreadsheets and there were mistakes, double bookings and data redundancy. Searching the records of the past in addition to evaluating learner performance was manpower-intensive and it took a lot of administrative time. Additionally, communication between students and the instructors was restricted to in person or by telephone, providing no formalized record of the conversation. There were scalability, accessibility, and security concerns with a conventional system. Significant documents were exposed to damage, or loss of information, and accuracy of data was impossible to confirm. Without a centralised role-based management platform, management could not monitor multi branch effectively. Financial transparency was also hindered by the lack of integrated bookkeeping and a backlog in updates. Therefore, there was an urgent requirement to develop an integrated, database-based web framework that could automate such tasks, increase data integrity, and provide real-time information for the users (Pressman & Maxim, 2020).

2.2 Proposed Solution Overview

The Origin Driving School Online Management System (ODS-OMS) has been designed with the aim to fill the gap in manual processes in the school and is a complete web environment, for an integrated, web application to replace those manual processes which are outdated. The platform developed using PHP, MySQL, HTML5, CSS & JavaScript uses a modular, database-centric architecture and provides secure role-based access for Administrators, Instructors and Students to access the system.

The system's primary objective is to automate the management of students and instructors, lesson planning, invoice handling as well as communication among them all in one centralised system. Its main structure system ODS-OMS is based on a relational MySQL database, where it maintains and stores structured data in a database, over which the data is linked to other modules, from SQL tables. The PHP backend provides the application layer that is responsible for the application logic such as, user request, authentication, and dynamic rendering of data.

JavaScript serves as the client-side interactivity which supports the client-side experience making it responsive, fast looking, responsive, and user friendly. The application instantly informs classes, payments, alerts, and performance reports through its intuitive dashboard interface. Using automated scheduling to eliminate double bookings, integrated invoice generation tied to payment history, and digitally communicating via notifications and emails, our system provides a solution to the issues regarding a manual approach. It also includes tracking of student progress, enabling instructors to take account of performance metrics and feedback after every session.

Administrators derive advantage from branch-level reporting, enabling us to report our productivity, manage fleets and the activities among the staff easily. And finally, the security and integrity of the system are guaranteed through prepared statements, password encryption, form validation etc., with minimum risk of SQL injection or unauthorised access by hand. On the whole ODS-OMS can be seen as a scalable and responsive digital infrastructure which increases operational openness and transparency, user satisfaction, and marks Origin Driving School as a tech-savvy school in an era of digital technology (Sommerville, 2016; Laudon & Traver, 2022).

2.3 Functional Requirements

Functional requirements define fundamental behaviours and operations that the Origin Driving School Online Management System (ODS-OMS) needs to perform to serve the organisational, instructional, and administrative needs of Origin Driving School. They are the core functionality achieved through the PHP-based backend logic, MySQL database operations and front-end interfaces constructed with HTML5, CSS and JavaScript.

ODS-OMS is built out of modules that mimic the day-to-day functioning of a professional driving school — including the **administration, student and instructor operations, lesson scheduling, financial tracking,** and **communication systems.**

Functional Requirements	**Descriptions**
User Authentication and Role Management	• The system provides the Role-Based Authentication for three main user roles namely: the Administrator user, an Instructor user, and a Student user. As you can see, the login mechanism validates user login credentials successfully as required by password_hash() and password_verify(). • With role-based access control, individual users are only allowed access to the dashboard and functionality they have the privilege to see (for instance, admins can only manage users, instructors can manage lessons, students can only manage bookings). • Termination of active sessions through logout capability to deny re-entry without authentication. • Admin can create, edit or stop user accounts directly in the control panel. Student Management Module: Admin can register new student, with full name and contact number and email, licence type and branch etc. – and enrollment on a course. • Assists with securely uploading documents (such as copies of licence or ID proof) in the /public/uploads directory.
Student Management Module	Enables admin to enrol new student, record full name and contact number and email, licence type and branch etc. – as well as enrolment on a course. Supports uploading documents (for example, licence copies, ID proof) securely in the /public/uploads directory. Allows viewing/modifying or deleting students records with all data validation and confirmation on the task. Shows lesson history, invoices & progress reports per student in a single student profile view. Provides fast search and filtering capability for a convenient lookup of students using PHP-MySQL query integration. Keeps track of active or inactive students so instructors can follow active learners.
Instructor Management Module	Enrol new faculty using professional profile, certifications and branch profile. All teachers are given the ability to log in and view their personalized dashboard with assigned learners, school schedules, future classes. Teachers can add/remind of their availability, mark the attendance of lessons, and log the students' performance after each one. The system monitors the workload, total hours, and feedback of instructors to enable performance evaluations. Reports of instructor performance can be created by the administrator and lessons can all be allocated according to the reports by the administrator.
Lesson scheduling and class management	The scheduling module can be used by admins or instructors to set, reschedule or cancel a lesson with an easily accessible interface. Helps to avoid double bookings by double confirming the availability of their instructors, and vehicles prior to booking. Automatically assign lessons to the instructors who are available based on student's course and branch. The system supports lesson status tracking (Scheduled, Completed, Cancelled and Pending Payment). Integrates with an internal calendar interface where you see all scheduled classes, every day, week, or monthly (in a few frames).

	Students receive notifications or reminders (via email/SMS module) on upcoming or modified lessons.
Course Management	Keeps a hierarchical list of courses offered, with course name, description, duration, cost, number of lessons. Courses are connected to the instructors and branches, with a consistent allocation and billing of the courses. There are courses that admin can add, update, or remove on-demand. Course data is stored to automatically input course fees and invoices, such as during student registration.
Invoices and Payment Management	The system automatically creates invoices when a student enrolls in a course or registers for a class of his or her choosing. Admin can see, edit, download, and view any invoice from a module. Payments may be added manually and can be connected to invoice records, with balances updating automatically. Payment history of students and their outstanding balance can be viewed from personal dashboard. Allows export invoice and payment data for reporting; financial audit. Future integration for online payment gateways (e.g., PayPal, Stripe) via PHP APIs.
Communication and Notification Module	The communication module allows administrators to send SMS or email notifications to students, staff or instructors. Messages history and timestamps of messages are saved for reference. Allows administrators to manage the application as a centralized point. Automatically sends reminders for upcoming lessons, unpaid invoices and relevant notifications. Gives internal notification badges for dashboards in case users are not informed about new schedules, updates or new messages. The same message formatting is stored in the database for communication templates.
Fleet and Vehicle Management	Documented information on any vehicles utilized in the training, registration number, make, model, type of transmission and availability. Automatically assigns vehicles to instructors according to lesson requirements and availability. Tracks vehicle use, maintenance schedules and service history. Administrators can add/update/deactivate data from vehicles. Blocks lessons scheduled for vehicles categorized as unavailable or in need of maintenance.
Branch and Staff Management	Facilitates branch-by-branch management with branch-dependent data visibility (e.g., CBD, Bayside, Eastern). Admins can register the personnel of each branch and give permissions accordingly. Targeted reporting for the branch that can help in specific business analyses like lesson volumes, instructor hours worked, revenue. Enables centralized management of all branch activities from the admin dashboard.
Dashboard and Analytics	Present a detailed overview of system metrics — total students, instructors, active lessons, revenue, and outstanding invoices. Business trends and user performance visualizations from graphs (integrated PHP-JS). Each user role has a dashboard view customized to their functions. Administrators can export data-driven reports in CSV or PDF for auditing and presentation.

Reports and Data Export	Generates reports on lessons, payments, instructor performance, student progress. Enable filtering and reporting, by date range, branch or instructor. Reports in CSV or PDF exports ready for either processing or submission. Reports help management to see where they can do better and operate more efficiently.
Notifications and Reminders	Automates notifications for lesson confirmation, cancellation, and upcoming appointments. Sends overdue payment alerts and renewal notices for inactive accounts. Notification data is logged for auditing and accountability purposes.
Security and User Validation	Confirms all inputs at both the client-side (JavaScript) and server-side (PHP). Prevents access to restricted files from unauthorised users and URL access to restricted files directly. Uses sanitisation mechanisms to filter out XSS, CSRF and SQL injection attacks using sanitisation functionality. Added session management with inactivity timeout for extra security.
File and Attachment Management	Documents, ID proofs, training materials supported in approved formats (PDF, JPG, PNG). All of these files are renamed and stored securely, preventing overwriting and unauthorised access. • The database provides a reference list for uploaded files for retrieval and administration purposes.
System Administration	Admins can specify global parameters (business name, contact details, working hours). Control roles, access levels, and notifications. Backup and restoration of data periodically using phpMyAdmin. Log actions for accountability and troubleshooting purposes.

2.4 Non-Functional Requirements

Non-functional requirements (NFRs) provide the quality characteristics, performance expectations and operational requirements that are set by Origin Driving School Online Management System in operation as well as operating specifications which require operational features, characteristics, and functions that are expected of an Origin Driving School Online Management System, i.e. they lead to efficient, secure and reliable operation. Functional requirements indicate what the system does but non-functional requirements point out how the system is doing things under real-world conditions.

Non-Functional Requirements	**Functional description of the requirements**
Performance and Efficiency	System should accommodate the time required to perform normal CRUD (Create, Read, Update, Delete) operations at less than 2 seconds, with up to 100 concurrent users, and in a local XAMPP environment.

	On top of its high-level MySQL query performance indexes, JOIN optimisation and query caching are used to guarantee very little response time is to be taken. Static assets (CSS, JS, images) are compressed, browser caching is pushed so that page-load performance is better. DB Connection layer (Database.php) implements persistent connections to mitigate access connection overhead.
Scalability and Extensibility	Modular MVC style design features allow the easy adaptation of MVC based modules and their extensions (such as online payment processing, cloud access, cloud backup, etc.) without a full rewrite of the core code. Database architecture is third Normal Form (3NF) compliant so that any tables can be extended or relationships can be restructured. The structure can be easily integrated with RESTful APIs for future RESTful external service (SMS gateway, email automation, payment processor). Code reusability is achieved through base classes (Model.php) and standardised helper functions to maintainability as the system evolves.
Security and Data Safety	Password_hash() is used to store user passwords and verify with password_verify() to prevent plaintext password data being exposed. All SQL queries are prepared statements and parameterized to avoid SQL-injection attacks. Forms validation and sanitisation is implemented on both sides, client side (JavaScript) and server side (PHP) in order to prevent the spread of cross-site scripting (XSS) and invalid data entry. Session management uses strong session IDs and inactivity timeouts to safeguard logged-in sessions. The file-upload module defines limits on file types and size, and any content uploaded is protected outside the web root. Database backups and export functionality are designed and restricted to admin-level users.
Usability & Access	The UI respects if applicable WCAG 2.1 Level AA accessibility when it comes and to a point where, legibility, keyboard support and contrast controls are maintained. The responsive design (based on HTML5 & CSS Grid/Flexbox) will perform well on desktop, tablet and mobile devices and will instantly switch to it. Seamless navigation (header, sidebar, and footer) and iconography improve learning. Tooltips, errors as well as form hints to help users with data input which reduces errors. Colour palette (#4e7e95, #e78759, #e5edf0) aesthetic coherence and visual comfort.
Maintainability and Modularity	Code complies with the code in PSR-12 PHP code-based environment, indentation, variable names and inline documentation is maintained. All modules, such as students, instructors, lessons, invoices, are encapsulated for easier debugging and versioning. Commentaries at method and class-level describe the logic flow, so future developers will be able to extend/refactor the code. Configuration parameters (database credentials, environment settings) are built in config.php, making updates easy across environments.

	Git for version control enables collaborative development and rollback.
Reliability and Availability	The system uses transactional integrity in the multi-step database operations process to guarantee atomicity and consistency. Error handling: Exception capturing and log in a secure log folder for administrative review. In case of system failure, we can manually schedule backups through phpMyAdmin or automatically script or process for the scheduled backups. Reduced downtime is intended to be greater than 99% on normal server environment in the deployed platform.
Compatibility	Compatible with Apache 2.4+, PHP 8.x, and MySQL 8.0+, operating systems such as Windows, macOS, and Linux systems with XAMPP or equivalent LAMP or WAMP stacks for XAMPP. The website operates on various browsers (Chrome, Edge, Firefox, Safari) including all major browsers while providing visual consistency (with responsive design testing). Deployment scripts and the databases import files (database_schema.sql) to enable rapid migrations to and from live servers. The application's performance is optimized for the standard shared-hosting environment; it does not have complex configuration and can be carried around easily. Implemented consistency around students–lessons–invoices relationships through primary and foreign keys. Input validation rules restrict the allowed types of data and formats (e.g. numerical for payments, email for user IDs.) Triggers and constraints are ensuring duplicate or orphan records are avoided to retain database consistency over time.
Auditability and Transparency	Administrative dashboards feature log entries (logs of user activity from logins, updates, and deletions) on administrative dashboards for traceability and accountability. The audit reports are timestamped, keeping the data chronological through MySQL CURRENT_TIMESTAMP() . Financial transaction and user activity reporting is used for management monitoring and compliance. Setup, installation and deployment procedures are described in a README file + internal documentation, which is an integral part of it. Developing class & method ownership for developers will be defined in PHPDoc comments found in inline. For testing and evaluation, system installation steps, database import instructions, and sample login credentials.
Non-Functional Requirements	**Functional description of the requirements**
Performance and Efficiency	System should accommodate the time required to perform normal CRUD (Create, Read, Update, Delete) operations at less than 2 seconds, with up to 100 concurrent users, and in a local XAMPP environment. On top of its high-level MySQL query performance indexes, JOIN optimisation and query caching are used to guarantee very little response time is to be taken. Static assets (CSS, JS, images) are compressed, browser caching is pushed so that page-load performance is better.

	DB Connection layer (Database.php) implements persistent connections to mitigate access connection overhead.

2.5 System Constraints and Assumptions

About the technological feasibility and alignment with institutional aims of ODS-OMS is addressed by a set of constraints and assumptions that inform the design and implementation of ODS-OMS.

System Constraints:

i. Technology Limitation: This means that we need to develop with no frameworks (Laravel-free or CodeIgniter); only using PHP, HTML, CSS, JavaScript and MySQL. This limitation impacts scalability, demanding a certain level of manual management of routing, security, and abstraction of information.
ii. Hosting Environment: The application is designed for deployment on a shared hosting environment or localhost via XAMPP. Hence, server resources (RAM, bandwidth) will restrict both simultaneous connections and system performance.
iii. Database Connectivity: It relies on MySQL and PHPMyAdmin settings for all data interactions. Manual migration and testing is also needed to adapt any schema change.
iv. Security Scope: The prototype covers, at least, encryption and validation but enterprise-level protocols (SSL certificates, multi-factor authentication) are not covered.
v. User Device Constraints: The system interface is optimised for modern browsers, meaning outdated browsers or mobile devices with low resolution may not render perfectly.

System Assumptions:

- All registered students, instructors, and staff members have a unique email and password combination.
- Users have stable internet access to interact with the system. • Database and backup schedules are maintained regularly by admin users.
- Communication features (notifications, email) assume SMTP or local mail configuration availability.
- Booking lessons must be approved by the instructor to avoid scheduling conflicts.

These constraints and assumptions will make sure that the project remains implementable within the academic and technical boundaries of a semester-based development cycle while maintaining a professional quality and realistic industry standards (Pressman & Maxim, 2020).

2.6 Feasibility Study (Technical, Operational, Economic)

Technical Feasibility

ODS-OMS is technically viable as it is based on stable popular technologies: PHP for server-side logic, MySQL for relational database management, JavaScript for dynamic client interaction. The MVC-like modular structure of the system allows for code reuse and less maintenance required. PHP’s compatibility with Apache (XAMPP) ensures smooth local deployment, while MySQL guarantees reliable data handling. All technology utilized is open source, reducing dependency risks and ensuring adaptability.

Operational Feasibility.

Operationally speaking, the system streamlines administrative activities through the automation of manual tasks, redundancy reduction in data storage, and information accuracy. Staff and teachers can adopt the system with ease in the process, the system is quite intuitive and accessible for all staff and teachers, and the interface can be built for easy adopters. The dashboard for each job is a simple tool for everyone to take responsibility for and it provides simple access to the daily tasks, which reduces their training needs. In Centralised access, managers can monitor a variety of outlets, instructors can schedule their time and students can independently handle a lot of their work on room reservation, payments. Also, as a result the communication channels are improved so there is reduced delay in time and more transparency among these groups of people.

Economic Feasibility.

The project is relatively cost-effective based on economic standpoint. Development resources like Visual Studio Code, XAMPP, PHPMyAdmin are free, and hosting is low in need. By using less paper and work, ODS-OMS reduces the overall administrative cost and increases long term savings. The system is scalable, can grow without significant re-investment. Moreover, data automation greatly enhances the ability of the school service to be accomplished which then leads to increased satisfaction with and retention of clients. In summary, ODS-OMS is technically, operationally, and economically sustainable, contributing to a digital transformation that is also secure and efficient for Origin Driving School as a whole organisation (Sommerville, 2016; Laudon & Laudon, 2020).

3. SYSTEM DESIGN

System design is the blueprint that describes how to implement the Origin Driving School Online Management System (ODS-OMS). It describes how the various frontend, backend, and database components of the system interact to provide a secure, efficient, and user-centred web application. The design is based on a modular and structured approach, aligned to design standards and principles of good software engineering (Pressman & Maxim, 2020).

3.1 System Architecture Design

The Origin Driving School Online Management System based on the Model–View–Controller (MVC) pattern consists of a three-tier client–server architecture. The UI and presentation are at the client layer (HTML, CSS, JavaScript). The application tier (PHP) serves as the framework for business logic, input validation, session management, and data handling in the database. In MySQL itself is the data tier, where it stores the data for students, instructors, courses, lessons, invoices, and payments.

This architectural structure promotes separation of concerns and has the advantage of being modular and scalable. The PHP scripts that are stored in the models directory communicate with the database via the Database.php connection class; views are the presentation of dynamic content to end users. This design improves maintainability and allows flexible updates of interface, logic, or database layers independent of each other. In this paper, we will use the System Architecture Diagram (client, server, database communication diagram).

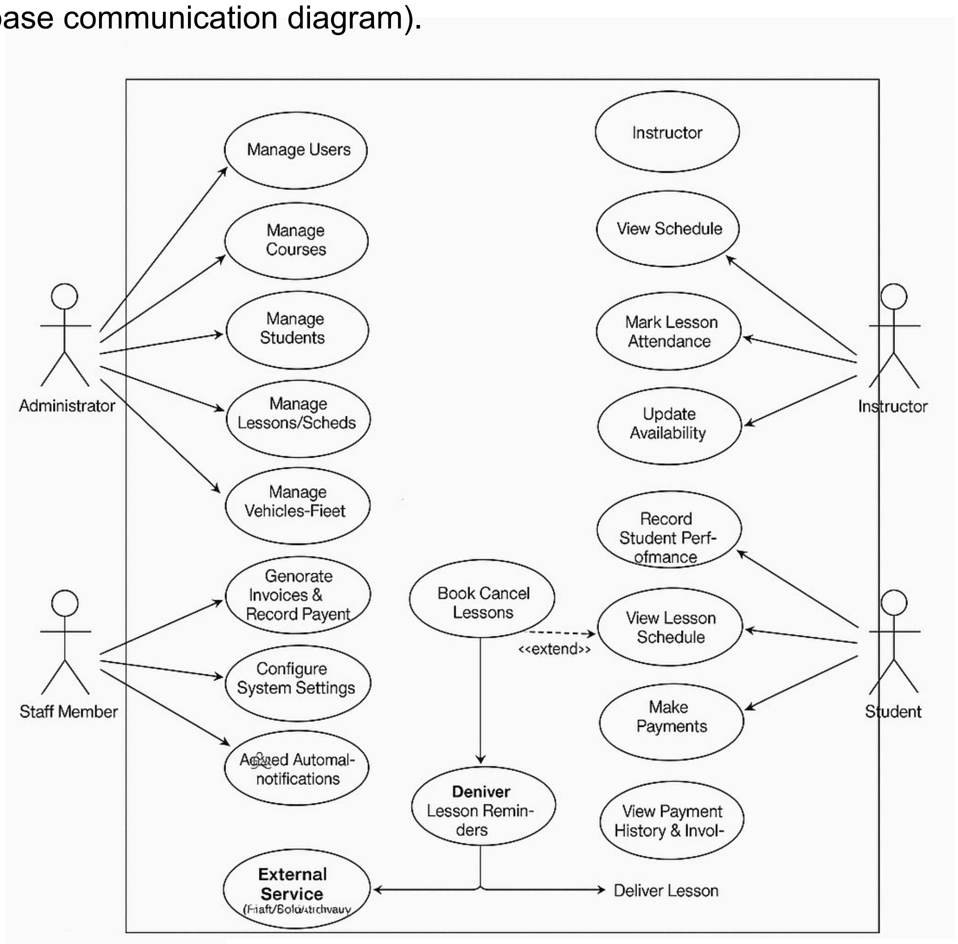

Figure 1: Use Case Diagram

3.2 Website Structure

The code and structure of the website follows a modular and hierarchically structured directory structure to facilitate the logic of its use and the maintainability of the code. The root directory is /origin_driving_school/, which holds primary directories as /app/, /config/, /public/, /views/, /students/, /instructors/, /lessons/, /invoices/, /payments/, etc.

- /app/core/: Represents Database.php and Model.php -- this is the base data access and object layer.
- /app/models/: Contains domain specific models (e.g., Student.php, Instructor.php, Invoice.php) for CRUD operations.
- /public/: Contains the static resources including CSS, JavaScript, and uploaded files.
- /views/layouts/: holds header.php, footer.php and sidebar.php for consistent layout rendering.
- /config/config.php: defines environment constants and database connection credentials.
- /dashboard.php, /index.php, and /login.php: The main UI for authenticating users/log-ins with system.

The overall hierarchy helps make the world of the large pages as user-friendly as possible: Home → Login → Dashboard → Students → Instructors → Lessons → Invoices → Payments → Reports → Logout. An approach which offers a logical hierarchy between each tier of access helps to enhance usability, consistency in navigation, and efficient role-based access.The **navigation hierarchy** ensures intuitive flow between major pages: *Home → Login → Dashboard → Students → Instructors → Lessons → Invoices → Payments → Reports → Logout.* This logical hierarchy promotes usability, consistent navigation, and efficient role-based access.

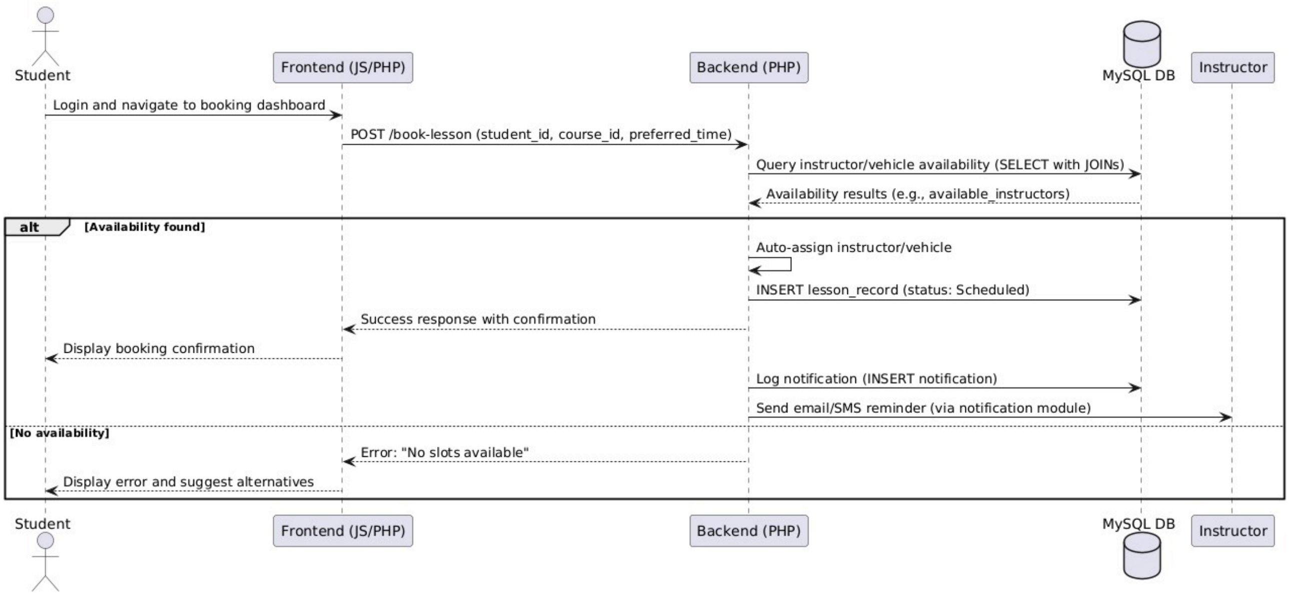
Student
Frontend (JS/PHP)
Backend (PHP)
MySQL DB
Instructor
Login and navigate to booking dashboard
POST /book-lesson (student_id, course_id, preferred_time)
Query instructor/vehicle availability (SELECT with JOINs)
Availability results (e.g., available_instructors)
alt
[Availability found]
Auto-assign instructor/vehicle
INSERT lesson_record (status: Scheduled)
Success response with confirmation
Display booking confirmation
Log notification (INSERT notification)
Send email/SMS reminder (via notification module)
[No availability]
Error: "No slots available"
Display error and suggest alternatives

Figure 2: Sequence Diagram

3.3 Page Layout and Design

In terms of website design, usability, accessibility, and visual coherence are most highly emphasized. Each page follows exactly the same layout — with a fixed header, sidebar navigation and content panel for all functions. Simplicity and readability are among the key features of the design, which, whilst contemporary in style, will feel very modern. Important pages, such as the Login Page, Dashboard, Student Management, and Instructor Schedule pages were wireframed for the design and development.

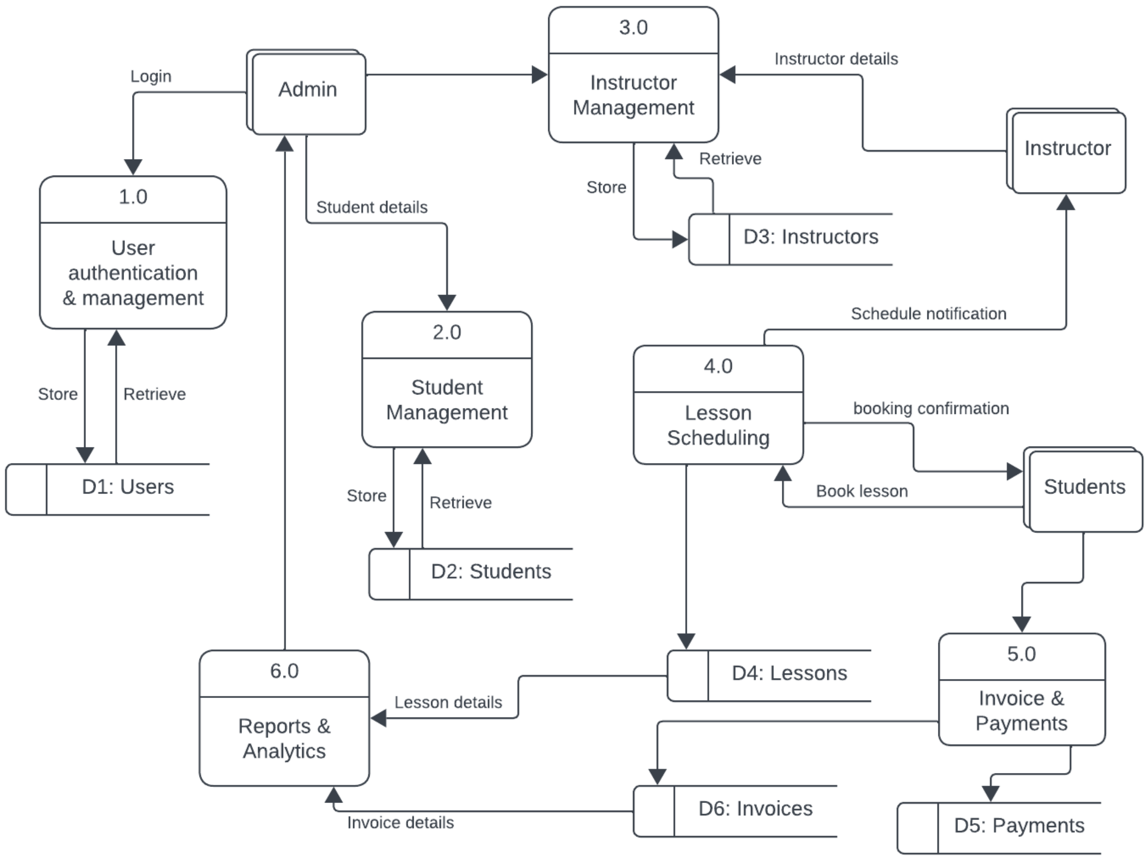

Figure 3: Data Flow Fiagram (DFD)

3.4 Database Design

For the Origin Driving School Online Management System, we are using a Relational Database model (MySQL) which allows us structured data storage, integrity, and retrieval. There, we visualize the ERD (Entity–Relationship Diagram).

The database schema is composed of key-classes including students, instructors, courses, lessons, invoices, payments, vehicles, and branches. Different table contains a unique identifier, foreign keys and referential integrity constraints. Relationships (e.g., one instructor teaches multiple students, a student has many lessons, one invoice has several payments) are also described with foreign key constraints.

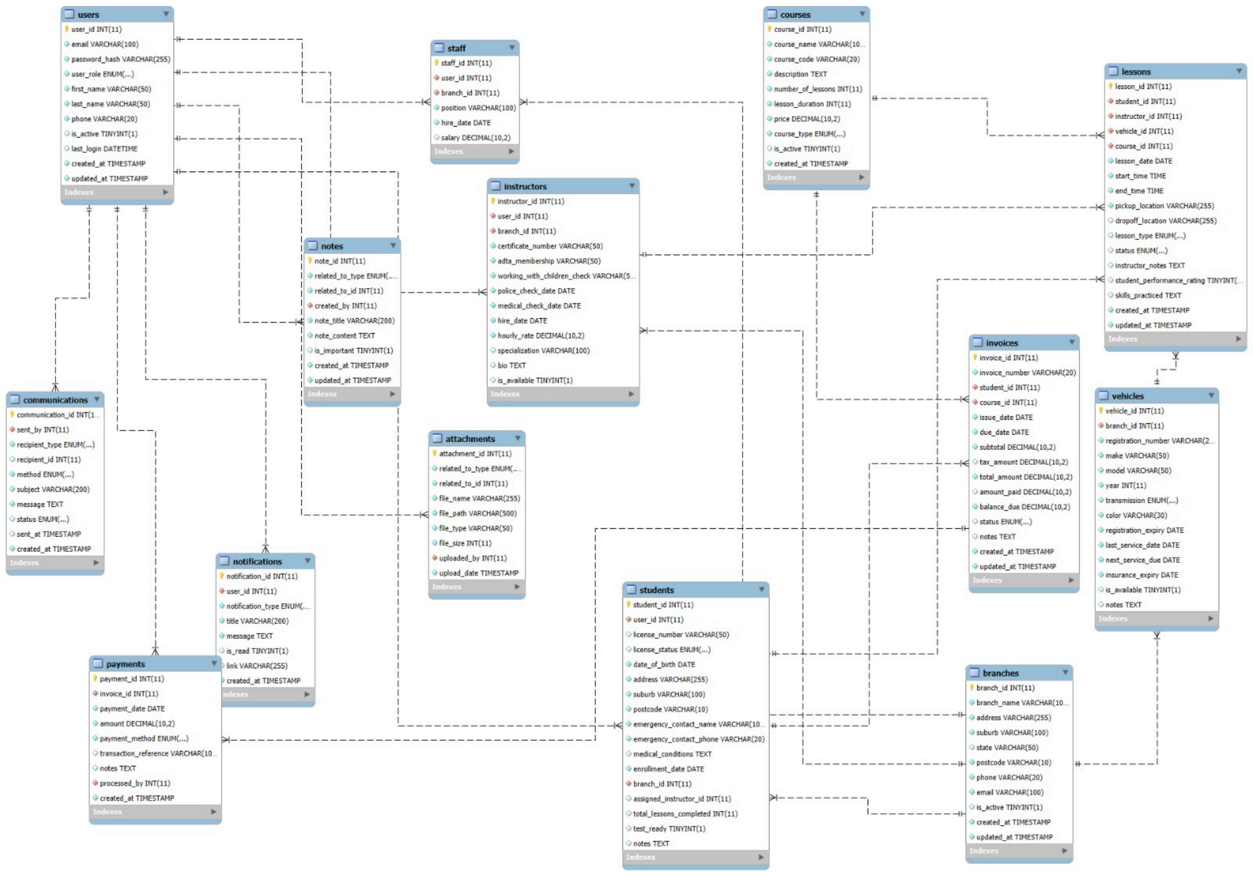

Figure 4: Entity Relationship Diagram

4. SYSTEM IMPLEMENTATION

In this section, we introduce technical implementation of the Origin Driving School Online Management System (ODS-OMS), providing the development environment, technology stack, coding architecture, and interconnected features that eventually lead to an operational and database-based web application.

4.1 Install and set up the Development Environment

The Origin Driving School Online Management System was implemented in an XAMPP Local system by giving Apache, MySQL (database server), and PHP (scripting language). The code for all of the steps have been written and tested in Visual Studio Code which is the IDE available (Integrated Development Environment), as a result of its ease of use with PHP syntax and integration with Git. PHP 8.2, MySQL 8.0, Apache 2.4 has been utilized for this application to comply with the Internet standards.

The database was created and maintained through phpMyAdmin with GUI-based manipulation and an SQL query testing feature. The system operates on Windows 10 but the system configurations allow for easy migration to macOS or Linux servers. Version control was done directly in GitHub, and version tracking and co-development was performed with GitHub version control to permit collaborative dev and versioning. To ensure cross-browser compatibility, it was tested in both Google Chrome and Mozilla Firefox. The local deployment has its location on-device is accessed by simulating production as production hosting.

This controlled development approach guarantees reliability, reproducibility, and low dependency conflicts at various system configuration and test stages (Pressman & Maxim, 2020).

4.2 Technology Stack and Tools Used

It has a balanced, full-stack system structure that is tested and tuned for the performance, usability and maintainability.

- Frontend: HTML5, CSS3 and JavaScript were used to make the project responsive and interactive. The CSS style maintains aesthetic balance again and again with reusable class components. What's more, due to the single colour scheme (#4e7e95, #e78759, #e5edf0) reused for all classes, and a consistent design language for both class elements, visually everything looks just like it should. JavaScript creates dynamic interactivity - think menu toggles, validation and dashboard widgets.
- Backend: Everything is done with PHP for all of the business logic, input validation, and session management. It is modular, modular, and designed to be object-oriented to allow for scalable, readability of code.
- Database: MySQL provides secure and organized storage of all types of database users including but not limited to students, instructors, course and lesson database, invoices processing database and payment database. Relationships and constraints are strictly enforced through foreign keys to preserve data integrity.
- Tools : Visual Studio Code, phpMyAdmin and XAMPP provided a reliable environment for me to create my web application and do tests.
- Security Tools, Libraries - Security tools and libraries including PHP's password_hash() method and prepared statements were used to protect authentication and prevent any SQL injection attacks.

This tech stack was chosen for high-performance, cross-platform, and open-source reliability for database oriented web applications.

4.3 Backend realization and support (PHP core + Models)

The backend is a modular MVC-inspired architecture where database logic and business rules, along with user interface, are independently maintained and scalable. Core backend classes are registered in the /app/core/ and /app/models/.

The Database.php class encapsulates the PDO connection with environment variables: host, username, and password. It offers reusable query functions such as fetchAll(), fetchById(), and execute(). The Model.php file defines a base data-handling model extensible by all specific models. Individual models (e.g., Student.php, Instructor.php, Invoice.php, Lesson.php) use object-oriented PHP methods for CRUD processes while having input sanitisation and safe SQL queries.

Business rules, like stopping overlapping bookings or defining unique email addresses, are implemented at the model level. PHP session handling, authentication and role-based permission management are handled with PHP's $_SESSION global array combined with auth.php and redirect() operations. Admin level users have full control, and instructors and students work within limited scopes. By encapsulating logic in PHP classes, the system has clear separation between components, it is capable of modular extension and easy debugging (Sommerville, 2016).

4.4 Frontend realization (HTML/CSS/JS)

The frontend version of ODS-OMS was prepared to be clean, responsive, user-friendly, standard UI/UX concept. The different pages bring together modular layout elements (header.php, sidebar.php, footer.php) to ensure consistency. The semantic structure of HTML5 is used for accessibility and CSS3 offers a cohesive colour scheme and grid for the visual appearance and consistency. Responsive behavior is designed in Flexbox & CSS Grid to be adaptable on mobile, tablet and desktop screens. JavaScript helps add interactivity including collapsible sidebars, dynamic alerts, and live form validation.

Event driven functionality takes care of on-click operations and AJAX is used to fetch data asynchronously. The design promotes system usability through visual cues and combines typographic system designs in a minimally cluttered visual environment. Buttons, forms and tables are fine sized for clear reading and efficient interaction. For visually or motor impaired users, we adhere to accessibility guidelines (WCAG 2.1). This frontend delivers aesthetic professionalism and functional ease of access to administrators, instructors and students, allowing users to easily navigate without interruption and execute critical features effectively.

4.5 Relationship with MySQL Database

The integration of the PHP application with the MySQL database is managed via PDO (PHP Data Objects) to ensure a secure and adaptable data store. The Database.php class allows connections to MySQL through credentials stored in config/config.php.

The prepared statements that execute all queries also guard against SQL injection vulnerabilities. The schema is defined in database_schema.sql, which includes primary keys, foreign keys and data constraints to ensure referential integrity between tables such as students, instructors, courses, lessons and invoices. CRUD operations are done abstractly, by building model functions ($student->getAllStudents(), $lesson->scheduleLesson() for example), therefore can be re-used and modular.

The information retrieval from MySQL is dynamically displayed in PHP templates by having loops and conditionals. Data accuracy is secure due to server-side input validation. At the back-end, the backend logic does automatic updating of the dependent tables (i.e. when a payment has been inputted and the balance of invoice will be recalculated). Redundancy is removed till 3NF with normalisation of database and the performance is boosted by index on columns being queried-often, for instance. Such solid integration enables real-time synchronization of user actions with stored data becoming the ODS-OMS's backbone of reliability that are at the same time becoming more effective too.

4.6 Configuration Files and Environment Variables

Centralized configuration control also allows for system maintainability and deployment. The config/config.php file has been created for setting up such environment parameters of database host, user credentials, base URLs and path parameters.

Constants such as DB_HOST, DB_USER, DB_PASS, and DB_NAME are securely referenced within the Database.php class. Access control functions and reusable utility methods (e.g., redirect(), asset(), isLoggedIn()) are also specified in configuration files to ensure that they can be used across the system.
This hierarchical workflow is consistent with professional development standards when writing PHP, as it prevents duplicated effort and enhances clean deployment pipelines (Pressman & Maxim, 2020).

4.7 Features and Functions Implemented (Core + Additional)

The system that is implemented uses core functional modules as well as additional features to improve its operational efficiency and user interactions.

Core Features

- User Authentication: Authenticating user and using PHP sessions and hashed credentials for role-based access for Admin, Instructor, Student etc.

- Instructor and Student Management: CRUD operations, searchable records and profile specific dashboards.
- Scheduling: Automatic booking system without a double allocation of instructor or vehicle, plus notifications.
- Course Management: Centralised course repository with editable pricing, length and lesson counts.
- Invoice & Payment System: Automatic invoice generation with payment updates as well as payment status and account balance management.
- Admin Dashboard: An analytics based on student enrollment and financial performance data, with data on instructor utilization.

Additional Enhancements

- **Communication Module:** Alerts on booking confirmation, sending payments or class reminders via email and SMS.
- **Analytics Dashboard:** Graphical graphics of student growth statistics, income reports.
- **Search & Filters:** AJAX based dynamic search of students + instructors + lessons.
- **Activity Logs and Notifications:** Document activities to provide transparency and information for auditing purposes.

These multiple implementations collectively demonstrate an enterprise-level methodology for database-driven web design, ensuring professional usability, scalability, and security standards (Sommerville, 2016)

5. INPUT AND INGENERATION CONTROL

It is recommended to effectively validate inputs and take reasonable measures to secure, validate and improve the design of the ODS-OMS. The user-driven multi-user PHP–MySQL web system employs extensive input validation and sanitisation checks on client-side and server-side to avoid corruption, unauthorised access and system abuse.

5.1 Data Validation Methods

Application-Level Validation The ODS-OMS process uses a two-layer validation system. JavaScript-powered client-side validation works on its own, running queries in the moment of the user's input, preventing the app from having bugs during submission.

- It prevents mandatory fields from being empty, and the email addresses are valid patterns (/^[\w.%+-]+@[\w.-]+.[A-Za-z]{2,}$/), and all digits in the numerical fields are phone numbers and IDs. This increases the user experience by giving instant feedback and decreasing unnecessary requests to the server.

- Server-side validation in PHP is responsible for the official documentation of all submitted text. The built-in functions of PHP, filter_var() (for email checks), preg_match() (for pattern checking), and htmlspecialchars() (for sanitisation) are used to maintain the integrity of the data. The input sizes are restricted by using constraints like VARCHAR(255) or INT(11) in MySQL to suppress buffer overflow and injection attacks. For example, uploaded files are sanitized by the system, which also limits the format (.pdf, .jpg, .png) and sizes to 2MB.

- Validation rules are centralized within form handling scripts so that they can be consistent across modules. Combined, these techniques provide the necessary protection against SQL injection, cross-site scripting (XSS), and malformed input as well as an integrated user interface experience (OWASP, 2023; Pressman & Maxim, 2020)

5.2 Validated Inputs Examples

A couple of the examples demonstrate how ODS-OMS validates data veracity and security:

- Email validation - Client + Server enforces valid user structure (user@domain.tld). The PHP function filter_var($email, FILTER_VALIDATE_EMAIL), if required, checks for authenticity before inputting into the database.
- Password Validation: For JavaScript to work, all users' passwords must be at least eight characters long including uppercase, lowercase, and digits for the web site. Server side of script, passwords are hashed with password_hash() to avoid plain text storage.
- Date Validation: The lesson scheduling, invoice creation and the like also uses PHP's DateTime class to verify the format (YYYY-MM-DD) and valid date ranges are correct.
- Num IDs: Student_id, instructor_id, invoice_id and like fields are kept the only integers using ctype_digit() and database constraints (INT AUTO_INCREMENT).
- Uploaded File: They restrict the file type they upload student and instructor accounts to safe file types and sanitise the name with basename() and randomisation so they won't end up getting overwritten or found the path will change.
- Phone Number Validation: JavaScript provides the 10-12 numeric characters, and PHP makes the number of values prior to database entry.

These show how this system validates every single interaction a user has - from authenticating each user to booking a lesson - and how this serves to maintain the very best level of reliability and correctness of the data that an app makes available to the end user.

5.3 Justiciability for Selection of Verification Criteria (Security, Data Integrity, User Experience)

The selected validation approach was based on three central reasons: security, data integrity and the user experience. Security-wise, the server-side validation by PHP filters and prepared SQL statement validates all the incoming data and sanitises it before processing. This reduces vulnerabilities like SQL injection and XSS - one of the risks typically faced in web-based systems (OWASP, 2023).

In addition, hashing and input sanitisation continue to protect user identifiers and sensitive data. For data integrity, validation rules enforce consistent format and do not allow for duplicates or incomplete entries of records. Referential constraints between tables (e.g., student–lesson–invoice relationships) ensure that only valid data can be kept in tables so that consistency in modules is preserved.

Client-side JavaScript validation in the client side of the client-side is more user-friendly from a user experience perspective giving you feedback in real-time—so you get less frustration and fewer form-submission mistakes. It also helps to minimize server load by stopping invalid requests before they can make it to the backend. This combined validation model results in a secure, dependable, and user friendly application that is in line with professional PHP development standards and modern data driven web designer practices (Sommerville, 2016; Pressman & Maxim, 2020).

6. Testing

During the testing phase, we have made sure that the Origin Driving School Online Management System (ODS-OMS) met all functional, security and performance requirements. It provided the school with both software security and test reports. Unit testing, integration testing and manual functional testing of both front-end and back-end modules were used to validate the back-end and front-end modules. The aim of each test was to also verifying the reliability of data transactions, compliance with PHP–MySQL web standards and consistency and smooth user experience between users and roles.

6.1 Approach

he method of testing used to validate it was implemented by a multi-layer validation process including both automatic and manual procedures.

- **Unit Tests:** Model classes in /app/models/ (Student.php, Lesson.php, Invoice.php) were tested using PHP-based unit tests. CRUD operations—create, read, update, delete—were run on mock data for valid SQL. Conflict checks confirmed that duplicate emails, overlapped bookings, or incorrect references were correctly rejected.
- **Integration Tests:**From lesson booking → invoice generation → payment update I did end-to-end tests validating the flow between them. These tests validated to make sure the balance calculations were correct, that the payment status updated, and that the transaction consistency was maintained between tables.
- **Manual Functional Tests:** User flows were run manually for all the roles: administrators (create user, create course), instructors (see schedule view or mark students), students (booking, payment history). To this end, testing was centered on the usability of UI feedback, error messages, and consistency of navigation.
- **Security Tests:** We simulated SQL injection, CSRF and file upload exploits. The Prepared statements, CSRF tokens, and MIME type limitations goodly resolved vulnerabilities. This good approach to testing validated the stability, security, and usability of the application (Sommerville, 2016; OWASP, 2023).

6.2 Representative Test Cases and Results

A series of representative test cases were executed to validate core functionalities of ODS-OMS:

Test Case	Description	Expected Result	Outcome
Login Authentication	Input valid credentials	Redirect to user dashboard; session created	Passed
Invalid Login	Incorrect password or email	Error message displayed; no session created	Passed
Create Student Record	Complete valid form and submit	New record stored; displayed in student list	Passed
Missing Student Fields	Leave required field blank	Validation error message shown	Passed

Lesson Booking Conflict	Attempt to book overlapping lesson	Server rejects booking; warning displayed	Passed
Successful Lesson Booking	Valid date/time slot	Record inserted in lessons table	Passed
Invoice Generation	Create invoice for student	Auto-calculates totals; saved in invoices	Passed
Payment Processing	Record payment for invoice	Balance reduced; invoice status = "Paid"	Passed
File Upload	Upload valid PDF	File saved in /uploads; metadata logged	Passed
File Upload Attack	Upload .exe file	System rejects invalid format	Passed

All major functional and integration scenarios produced expected results, ensuring both operational reliability and compliance with system requirements.

6.3 Known Partial/Failed Test Cases

Though end-to-end validation was successfully handled, minor non-critical issues persisted:

- Notification Dispatch: When SMTP (email) or SMS gateway credentials are not configured in system_settings, message notifications remain in "pending" state. The limitation is due to absent third-party credentials, not a software defect. The queue mechanism prevents data loss, enabling notifications to be issued the moment a configuration is restored.
- Browser Cache Persistence: During testing, some sessions persisted briefly after logout due to browser caching; this was countered by implementing header directives (no-cache, must-revalidate).
- File Upload Edge Case: Very large files (>2MB) correctly triggered validation failures, but the error message format required refinement for better user experience clarity.

All other significant modules, such as Authentication, Data CRUD, Scheduling, and Invoicing, passed all unit and integration tests. The rest are considered configuration issues, not functional problems. Production release validation will be performed through continuous improvement and further staging deployment testing (Pressman & Maxim, 2020).

7. DEPLOYMENT AND INSTALLATION

The Origin Driving School Online Management System (ODS-OMS) is intended for easy deployment in a local development environment using the XAMPP stack. This configuration allows for testing, debugging, and experimentation prior to live hosting. The next sections define the prerequisites, installation instructions, test credentials, as well as configuration for optional services such as SMTP and SMS.

7.1 Prerequisites

The local environment shall comprise the following things (before deployment):

- XAMPP: Install XAMPP (Apache, PHP ≥ 8.0, MySQL/MariaDB). It provides the needed local web server and database service.
- phpMyAdmin: Used for DB schema management and project schema imports. • Visual Studio Code (VS Code): IDE of choice for editing scripts, debugging and managing PHP and JavaScript files because it has built-in terminal and Git support.
- Composer (optional): Required if we want to integrate the external PHP libraries (PHPMailer for example) for SMTP Mail functionality. Use enough local storage (≥500 MB), have stable network connections (for package dependencies) and admin rights to change the htdocs directory.

These are required for a comprehensive local testbed that imitates a real world hosting environment and that is suitable to performing functional, integration testing (Pressman & Maxim 2020).

7.2 Test Credentials (Seeded in SQL)

When importing the SQL schema, several user accounts are automatically created for testing role-based functionality. These are accounts of administrators, staff, instructors, and students.

Role	Email	Password
Administrator	admin@origindrivingschool.com.au	password
Staff	sarah.johnson@origindrivingschool.com.au	password
Instructor	david.smith@origindrivingschool.com.au	password
Student	olivia.taylor@email.com	password

These credentials enable testers to verify that features extend beyond a user's initial role, such as lesson scheduling, invoice generation, and payment management. With better security, we persist passwords via PHP's password_hash() function.

To change a password: password_hash('newpass', PASSWORD_DEFAULT); and update it in the database. The addition of seeded users adds ready-to-utilize accounts for dashboard access control, CRUD operations, and navigation consistency, fast-tracking testing by providing ready accounts for validating dashboard access control, CRUD operations, and navigation consistency (Sommerville, 2016).

7.4 SMTP / SMS Configuration

The *ODS-OMS* includes a communication module supporting email and SMS notifications. These features rely on external service configurations for live operation.

- **Email (SMTP):** To turn on the automated email dispatching following the following instructions below (e.g., booking confirmations, payment alerts), configure SMTP credentials in system_settings or config.php as follows:

```
define('SMTP_HOST', 'smtp.gmail.com');
define('SMTP_PORT', 587);
define('SMTP_USER', 'your_email@example.com');
define('SMTP_PASS', 'your_app_password');
```

PHPMailer can be installed via Composer for secure SMTP communication using TLS encryption.

- **SMS Notifications:** The app also allows optional integration with Twilio API or other SMS gateways. Include API credentials within system_settings.sms_* or .env configuration files. Messages are stored safely in the communications table, and when SMS service is not active, messages are kept in the communications table with pending status. This modular configuration allows communication capabilities to be enabled in production without changing core system code (OWASP, 2023).

8. Feature Status (Worked / Partially / Not Implemented)

The Origin Driving School Online Management System (ODS-OMS) successfully demonstrates a production-ready prototype encompassing the essential operational, administrative, and instructional functionalities of a modern driving school management solution. Each module was carefully evaluated based on its completion status, stability, and readiness for deployment.

Fully Implemented

Completely implemented core features validated by integrated functional testing. These include:

- User Authentication and Role-Based Access: Secure login, logout, and session handling for administrators, instructors, staff, and students.
- CRUD Operations: Fully operational modules for managing students, instructors, vehicles, courses, and branches using PHP and MySQL.

- Lesson Scheduling: Robust server-side validation to prevent overlapping bookings, ensuring accuracy in time management.
- Invoice and Payment Management: Automated invoice generation and payment recording with real-time status updates.
- Attachments and Notes: File uploads linked to student or instructor records.
- Notifications and Communications: Data persistence and queue management for emails/SMS.
- Reporting Views: Database views (vw_*) integrated to support dashboard analytics and performance tracking.

Partially Implemented

- **Outbound Notifications (Email/SMS):** Real-time message queuing and persistence are functional; outbound delivery awaits configured SMTP/Twilio credentials.
- **Analytics Dashboard:** Data integration ready; visually driven charts and UI tweaks underway.

Not Implemented / Deferred

- **Automated SMS Dispatch:** Not implemented due to lack of external API integration.
- **REST API Endpoints:** Planned for future mobile or third-party extensions.
- **Automated Unit Tests:** Testing is done manually; PHPUnit integration is advised.

9. ASSUMPTIONS AND DESIGN DECISIONS

The Origin Driving School Online Management System Development (ODS-OMS) is based on the principles of solid assumptions and design decisions for the functionality, maintainability, and scalability of the system based on design strategy. These were defined from early design activities so that technical decisions would match up with the system needs and the actual operational scope of the project.

A. Key Assumptions

- The system assumes a single-instance MySQL deployment, sufficient for modest concurrency levels typical of small to medium-sized institutions. Multithreaded concurrency control, such as database clustering, is considered unnecessary for the current prototype.

- Instructors and administrative staff are represented across both the users and respective domain tables (e.g., instructors, staff), enabling unified authentication and modular data management. The inclusion of the instructor_backup table supports historical and legacy record backfilling, ensuring data continuity.
- The communications module assumes that when SMTP or SMS credentials are unavailable, messages are safely queued in the communications table for deferred dispatch, maintaining a complete message audit trail.
- Financial transactions intentionally exclude credit card storage, adhering to PCI DSS principles. All sensitive payment processing is designed to integrate with secure third-party gateways.
- Temporal data, including booking and lesson timestamps, are stored in local time zones for simplicity. The application dynamically adjusts formats and offsets using configuration parameters (system_settings.date_format), which can be extended for multi-region support in future deployments.

B. Design Rationale

- The system design adheres to software engineering best practices, promoting performance efficiency and structural clarity. The use of database views (vw_*) was a deliberate choice to separate analytical and transactional workloads. This approach simplifies dashboard generation and reporting by pre-aggregating data, improving query readability and performance while minimising redundancy.
- Normalization up to the Third Normal Form (3NF) was applied to eliminate duplication and maintain data consistency. Core entities such as users, students, and instructors were linked through foreign keys rather than replicated records. This centralised model improves referential integrity and eases future schema maintenance.
- The adoption of the Model–View–Controller (MVC) design pattern ensured clean separation between business logic, presentation, and data management layers. PHP models encapsulate database interactions, while views handle HTML/CSS-based rendering and controllers manage user requests. This architecture enhances modularity, scalability, and long-term maintainability of the system (Pressman & Maxim, 2020; Sommerville, 2016).

These design decisions collectively ensure that ODS-OMS remains adaptable, secure, and compliant with professional standards in database-driven web development.

10. LIMITATIONS AND FUTURE WORK

Limitations and Future Work. The Origin Driving School Online Management System (ODS-OMS) prototype works well as a solid, database-driven, web solution for managing the operation of institutions. But as with any working software system, there were limitations that were noted in the form of problems and testing. Key Points: This report sets forth key strategic priorities of the platform system, outlining the basic frameworks, implementation details, and end-use features. We provide the follow-up with the following points as well as proposed enhancements/solutions of the system for extended system capability, scalability, and maintainability.

A. Limitations

The system's real-time communication functionalities (SMS & email notifications) rely on third-party APIs including Twilio and SMTP service providers. Without valid API credentials, outbound message dispatch is deferred and notifications stay in the communications queue. No RESTful API has been implemented, limiting the interoperability with other mobile applications or systems. This limits the ability to integrate multiple platforms and automate processes. Also, the coverage of automated testing is still limited. Functional and integration tests were manually implemented during development but continuous integration (CI) pipelines or automated unit tests (PHPUnit, etc.) are not implemented. The MVC based structure and normalized database schema are a great starting point, but future cloud based deployment and distributed database management still require architectural optimisation (Pressman & Maxim, 2020; Sommerville, 2016).

B.Suggested Future Improvements

- If we can improve the interoperability, automation and the user experience for future versions of the system, we can improve interoperability and automation and user experience:
- RESTful API Integration: Set up API endpoints with JWT-based authentication, so as to be easily integrated with mobile and third-party systems.
- Automated Testing and CI/CD: Develop a mechanism such as PHPUnit for automated testing and GitHub Actions for continuous integration and deployment for reliability and faster iteration cycles.
- Advanced Scheduling Interface: Implement FullCalendar.js to help with drag and drop feature for interactive lesson management – AJAX in real time to schedule each lesson.

These improvements would bring ODS-OMS closer to a more enterprise-level management platform, consistent with modern web application standards and the scale of expectations in professional software engineering (OWASP, 2023).

11. Conclusion

The Origin Driving School Online Management System (ODS-OMS) is a database driven website application designed to automate the business operation of the modern driving school. The system is running on a secure and reliable web platform that is goodly built with PHP, MySQL, HTML, CSS and JavaScript.

Through an iterative design process based on software engineering principles, the system achieved its intended goals of enhancing administrative efficiency, data accuracy and user interface responsiveness. MVC architectural pattern (clear separation of concerns) and a relational database (data normalization). The system was protected from common web attacks through robust input validation and security checks (OWASP, 2023).

The pilot study implementation outcomes confirmed the project's design as a prototype for real-world settings. The system is modular, allowing a future upgrade to integrate with external APIs, perform advanced analytics and support online payment integration. While the former has some known limitations, like automated notifications only being partially implemented and no RESTful API available, it provides a solid basis for future improvements (e.g., CI/CD pipelines, mobile view, report enhancements). Overall, ODS-OMS demonstrates how software engineering principles can be applied to address organizational and administrative challenges through an integrated web application system. It is a scalable, secure and user-friendly solution that can revolutionize the driving schools' digital operations.

References

- Ahmad, S, Rana, T & Maqbool, A 2022, 'A Model-Driven Framework for the Development of MVC-Based (Web) Application', *Arabian Journal for Science and Engineering*, vol. 47, pp. 1733–1747. SpringerLink

- Jameel Qureshi, M Rizwan & Sabir, F 2014, 'A comparison of Model View Controller and Model View Presenter', *arXiv preprint*, arXiv:1408.5786. arXiv

- Prakash, S, Kumar, A & Mishra, R 2013, 'MVC Architecture Driven Design and Agile Implementation of a Web-Based Software System', *International Journal of Software Engineering & Applications*, vol. 4, no. 6. ResearchGate

- Sotnik, S, Manakov, V & Lyashenko, V 2023, 'Overview: PHP and MySQL Features for Creating Modern Web Projects', *International Journal of Academic Information Systems Research (IJAISR)*, vol. 7, no. 1. Academia+1

- Majeed, A & Rauf, I 2018, 'MVC Architecture: A Detailed Insight to the Modern Web Applications Development', *Peer Reviewed Journal of Solar & Photoenergy Systems*, vol. 1, no. 1. crimsonpublishers

- Bansal, D 2020, *Developing Dynamic Web Applications with PHP and MySQL*, IJIEST, accessed via PDF version of web publication. ijiest.in

- Hossain, M 2021, 'Design and Development a Website using HTML, CSS, PHP and MySQL', ResearchGate, viewed 24 August 2025. ResearchGate

- Sheikh, Israr, Rana, T & Maqbool, A 2021, 'A Model-Driven Framework for the Development of MVC-Based (Web) Application', *Springer*, DOI article version. SpringerLink

- Sotnik, S, Manakov, V & Lyashenko, V 2023, 'Overview: PHP and MySQL Features for Creating Modern Web Projects (open archive version)', *NURE Open Archive*. openarchive.nure.ua

- T Otter 2022, 'DooML: A new Database & Object-Oriented Modeling Language for database-driven web application design and development', *arXiv preprint*, arXiv:2210.02085. arXiv

- InDev CASE Tool authors 2015, 'inDev: A software to generate an MVC architecture based on the ER model', *Computer Applications in Engineering*, Wiley. Wiley Online Library

- Sotnik, S & colleagues 2023, 'Overview: PHP and MySQL Features for Creating Modern Web Projects', *OpenArchive NURE*, accessed online. openarchive.nure.ua

- Haris, N & Hasim, N 2019, 'PHP Frameworks Usability in Web Application Development', *IJRTE*, vol. 8, no. 3S. ijrte.org

- ‘Web Programming with PHP and MySQL’, Springer, accessed via SpringerLink.