Sri Lanka Institute of Information Technology

Software Engineering Practices

(IT5030)

Continuous Assignment – 2025, Semester 1/2

DiveMaster – Diving Certificates Management System

Final Report

Group Name: Code Crafters

No	Student ID	Student name
1	MS24905336	Singh W M T R
2	MS25914078	Premarathne H M P D
3	MS24905954	Kaushalya M K B
4	MS24912570	Weerapana W M G D
5	MS25913774	Gamlath M G H C

Table of Contents

1. Introduction

1.1 Background

In an era where digital transformation has touched nearly every sector, the recreational and professional diving industry is no exception. The DiveMaster – Diving Certification Management System (DCMS) is an innovative, end-to-end digital platform designed to automate and optimize the workflows of diving centers in Sri Lanka. The island's unique geographical positioning and abundant marine biodiversity make it a hotspot for diving tourism, which continues to grow annually. Despite this rise in demand, most diving centers still rely on manual, fragmented systems for managing training, certifications, dive logs, bookings, and equipment. This results in a multitude of inefficiencies, such as miscommunication, scheduling conflicts, delayed certifications, and compromised safety standards due to lack of real-time resource tracking.

DiveMaster is introduced as a scalable and centralized system that integrates all diving-related operations into a single, user-friendly platform. The system is designed with the needs of various stakeholders in mind—ranging from students and instructors to dive center administrators and certification authorities. It ensures a reliable and secure environment where divers can learn, train, and get certified with minimal administrative overhead and maximum transparency.

1.2 Functional Requirements

Functional requirements describe what the system must do to serve its intended purpose. DiveMaster was conceptualized after conducting detailed research into existing diving certification workflows, identifying inefficiencies, and mapping user expectations into technically feasible and scalable features.

1. **User Management & Authentication**

   At the heart of any multi-user system is secure user access and identity management. DiveMaster supports a sophisticated user management module that includes account creation, login functionality, role assignment, and account recovery mechanisms. Users are categorized based on roles such as:

   - *Guest Users*
   - *Recreational / Certification-Seeking Divers*
   - *Research Divers*
   - *Instructors*
   - *Dive Center Managers*

   Each role is associated with a predefined set of permissions enforced through a Role-Based Access Control (RBAC) mechanism. Multi-Factor Authentication (MFA) is used to ensure that sensitive user accounts, especially those with administrative rights, are protected against unauthorized access.

   Users can update their profiles, track their personal training and certification history, and manage emergency contacts and booking preferences from a centralized dashboard.

2. **Dive Booking and Scheduling**

   One of the most complex and time-sensitive operations in diving centers is managing course schedules, dive reservations, and instructor availability. DiveMaster provides an intuitive, calendar-driven interface for divers to browse, reserve, and manage their sessions. This includes:

- Booking open water dives, confined water sessions, and full certification courses.
- Viewing availability of instructors, gear, and boats in real-time.
- Automatic instructor assignment based on availability and expertise.
- Conflict resolution by detecting double bookings or equipment shortages.
- Allowing instructors and divers to request modifications or cancel reservations.

Dive center managers can approve, reschedule, or reject bookings and view the entire schedule matrix at a glance, helping them make informed operational decisions.

3. Dive Booking and Scheduling

DiveMaster revolutionizes dive record-keeping by replacing paper-based logbooks with a smart digital dive log system. This allows divers to document key details such as:

- Dive location, depth, duration.
- Weather and water conditions, including visibility and temperature.
- Buddy diver details and instructor notes.
- Equipment used and any issues encountered.

A special module enables marine life tracking, which is particularly valuable for research divers. The system supports offline data collection using a Flutter-based mobile app, with offline-first architecture. After the dive, data is automatically synced with the main system when connectivity is restored.

Moreover, the system integrates AI-based tools to help divers identify fish species and underwater hand signals using uploaded images and onboard models. This adds an interactive, educational element to recreational diving and supports biodiversity research.

4. Dive Booking and Scheduling

DiveMaster supports the entire certification journey, from theoretical preparation to final credential issuance. Students can access their courses through a built-in learning management system (LMS), complete with:

- Modular lessons and video tutorials.
- Interactive quizzes, exams, and practical evaluation forms.
- Auto-grading and real-time progress tracking.

Instructors can track each student's performance and validate practical skills during training dives. Upon successful completion of all modules, the system generates a secure digital certificate with a unique QR code for instant verification.

Certification authorities can log into their portal, audit assessment history, and approve or revoke certifications when necessary.

1.3 Additional (Secondary) Functional Requirements

While the above are the core pillars of the system, several secondary requirements enhance functionality and usability:

- **Equipment Inventory & Maintenance**

  DiveMaster includes a full-featured inventory management module to track diving gear such as tanks, suits, regulators, and emergency equipment. The system assigns gear to dive sessions, monitors usage history, and alerts managers when equipment is due for servicing.

- **Reports & Analytics**

  DiveMaster generates a wide range of analytical reports for instructors, students, and managers. These include student progress, instructor workloads, dive session frequency, equipment utilization, and certification trends. Reports can be filtered by date, session type, user role, or certification level and exported in multiple formats.

Non-Functional Requirements

Non-functional requirements define the overall quality and system behavior. They are critical to ensuring that DiveMaster performs well under real-world conditions.

1. **Scalability**

   DiveMaster is designed with scalability in mind. As multiple diving centers come onboard, the system should support hundreds of concurrent users without performance degradation. Cloud-hosted database services and stateless API endpoints allow for horizontal scaling to accommodate future growth.

2. **Security**

Given the sensitive nature of the data involved—certifications, personal IDs, payment records—the system enforces strict security protocols:

- HTTPS/SSL encryption for all data transmission.
- Secure password hashing using industry-standard algorithms (e.g., bcrypt).
- Encrypted data at rest for all user and booking information.
- Logging and auditing of sensitive operations.

3. **Accessibility and User Experience**

   DiveMaster follows modern UX/UI design principles to deliver an intuitive, responsive, and accessible user experience:

- Fully responsive design (desktop, tablet, mobile).
- Offline capabilities for divers using the mobile interfaces like rugged tablets which can be used underwater.
- WCAG-compliant design to support users with visual impairments.
- Clean, distraction-free interfaces using CSS for consistent styling.

1.4 High-Level System Architecture

The system is composed of three main layers:

1. Frontend: Developed using Laravel blades for web users and Flask HTML for mobile devices, offering dynamic UI components, routing, and offline-first capabilities.
2. Backend: Built with Laravel (PHP), which serves RESTful APIs to handle business logic, validation, authentication, and database interactions.
3. Database: MySQL is used as the primary RDBMS, optimized for read/write performance and relational data integrity.

Each component is modularized to allow for continuous integration, independent updates, and streamlined maintenance.

2. Project Management

The success of any complex software solution depends not only on technical execution but also on the strength of the project management framework that supports it. For the development of the DiveMaster – Diving Certification Management System (DCMS), the project team adopted a highly structured and collaborative approach to managing the project lifecycle. This section outlines how responsibilities were distributed, the tools and methods used to track progress, and how agile practices were implemented to ensure smooth development and delivery.

2.1 Workload Distribution

The DiveMaster system was developed by a team of five members, each assigned to a specific module that aligned with their strengths and interest areas. Workload distribution was done not only to ensure individual accountability but also to mirror the real-world development practice of modular ownership. Each member was responsible for requirement gathering, design, development, testing, and documentation of their respective components.

Student ID	Student name	Assigned Component
MS24905336	Singh W M T R	Dive Logs and Marine Life Tracking Service
MS25914078	Premarathne H M P D	Scheduling and Resource Allocation Service
MS24905954	Kaushalya M K B	Assessment and Certification Service
MS24912570	Weerapana W M G D	Equipment Inventory Management
MS25913774	Gamlath M G H C	User Management and Booking System

Each member contributed toward system integration and shared responsibilities during deployment, documentation, and testing phases. Regular sync meetings ensured that no component was developed in isolation and that inter-module dependencies were communicated clearly.

2.2 Agile Scrum Methodology

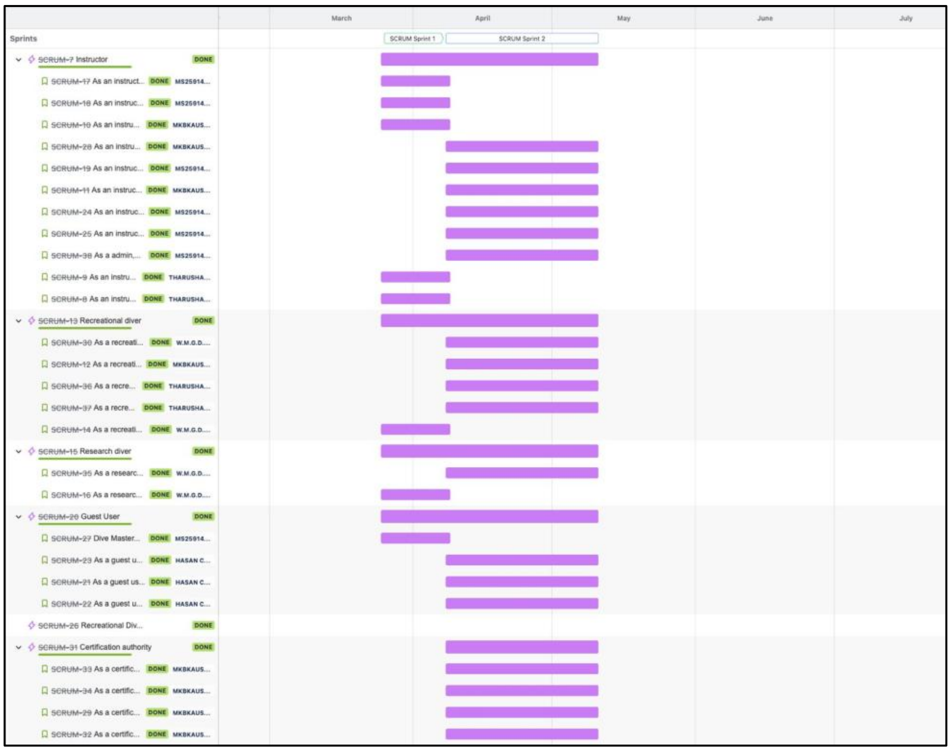
To manage time and maintain flexibility throughout the development cycle, the team followed the Agile Scrum methodology. The iterative nature of Scrum allowed for continuous feedback, faster issue resolution, and early identification of feature gaps. The project lifecycle was broken into three distinct sprints:

- **Sprint 1 – Foundation Setup & Core Frameworks**
  Focus: Initial database design, authentication system, routing setup, and UI framework initialization.
  Deliverables: Working login module, database schema, and boilerplate backend API routes.

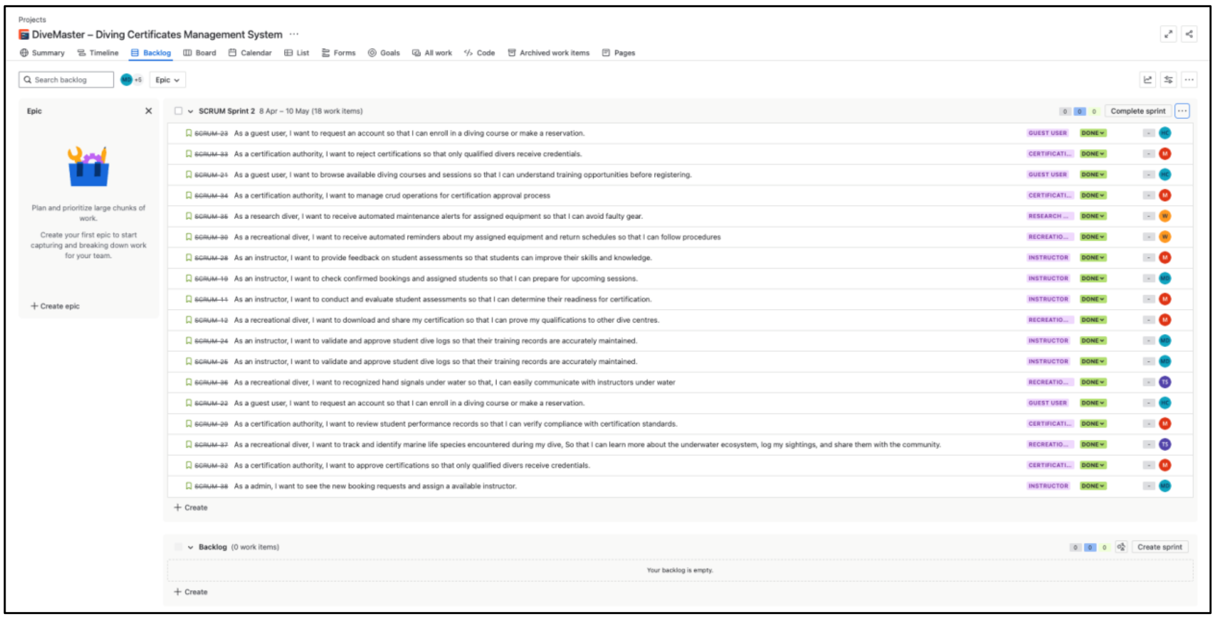
- **Sprint 2 – Module Development & Integration**
  Focus: Parallel development of all core modules (user, scheduling, equipment, logging, certification).
  Deliverables: Full CRUD operations for major components, instructor scheduling logic, equipment lifecycle management.

- **Sprint 3 – Testing, Reporting & Deployment**
  Focus: Bug fixing, final feature integration, security enhancements, and documentation.
  Deliverables: CI testing reports, final deployment, Gantt chart verification, and project wrap-up.

Sprint retrospectives were conducted at the end of each cycle to assess what went well, what didn't, and how to improve. These reflections helped maintain team motivation and identified blockers early on.



Sprints		
~~SCRUM-7~~ Instructor	DONE	
~~SCRUM-17~~ As an instruct...	DONE	MS25914...
~~SCRUM-18~~ As an instruc...	DONE	MS25914...
~~SCRUM-10~~ As an instru...	DONE	MKBKAUS...
~~SCRUM-28~~ As an instru...	DONE	MKBKAUS...
~~SCRUM-19~~ As an instruc...	DONE	MS25914...
~~SCRUM-11~~ As an instruc...	DONE	MKBKAUS...
~~SCRUM-24~~ As an instruc...	DONE	MS25914...
~~SCRUM-25~~ As an instruc...	DONE	MS25914...
~~SCRUM-38~~ As a admin,...	DONE	MS25914...
~~SCRUM-9~~ As an instru...	DONE	THARUSHA...
~~SCRUM-8~~ As an instru...	DONE	THARUSHA...
~~SCRUM-13~~ Recreational diver	DONE	
~~SCRUM-30~~ As a recreati...	DONE	W.M.G.D....
~~SCRUM-12~~ As a recreati...	DONE	MKBKAUS...
~~SCRUM-36~~ As a recre...	DONE	THARUSHA...
~~SCRUM-37~~ As a recre...	DONE	THARUSHA...
~~SCRUM-14~~ As a recreati...	DONE	W.M.G.D....
~~SCRUM-15~~ Research diver	DONE	
~~SCRUM-35~~ As a researc...	DONE	W.M.G.D....
~~SCRUM-16~~ As a researc...	DONE	W.M.G.D....
~~SCRUM-20~~ Guest User	DONE	
~~SCRUM-27~~ Dive Master...	DONE	MS25914...
~~SCRUM-23~~ As a guest u...	DONE	HASAN C...
~~SCRUM-21~~ As a guest us...	DONE	HASAN C...
~~SCRUM-22~~ As a guest u...	DONE	HASAN C...
~~SCRUM-26~~ Recreational Div...	DONE	
~~SCRUM-31~~ Certification authority	DONE	
~~SCRUM-33~~ As a certific...	DONE	MKBKAUS...
~~SCRUM-34~~ As a certific...	DONE	MKBKAUS...
~~SCRUM-29~~ As a certific...	DONE	MKBKAUS...
~~SCRUM-32~~ As a certific...	DONE	MKBKAUS...



Projects

DiveMaster – Diving Certificates Management System

SCRUM Sprint 2 8 Apr – 10 May (18 work items)

~~SCRUM-23~~	As a guest user, I want to request an account so that I can enroll in a diving course or make a reservation.	GUEST USER	DONE	HC
~~SCRUM-33~~	As a certification authority, I want to reject certifications so that only qualified divers receive credentials.	CERTIFICATI...	DONE	M
~~SCRUM-21~~	As a guest user, I want to browse available diving courses and sessions so that I can understand training opportunities before registering.	GUEST USER	DONE	HC
~~SCRUM-34~~	As a certification authority, I want to manage crud operations for certification approval process	CERTIFICATI...	DONE	M
~~SCRUM-35~~	As a research diver, I want to receive automated maintenance alerts for assigned equipment so that I can avoid faulty gear.	RESEARCH ...	DONE	W
~~SCRUM-30~~	As a recreational diver, I want to receive automated reminders about my assigned equipment and return schedules so that I can follow procedures	RECREATIO...	DONE	W
~~SCRUM-28~~	As an instructor, I want to provide feedback on student assessments so that students can improve their skills and knowledge.	INSTRUCTOR	DONE	M
~~SCRUM-19~~	As an instructor, I want to check confirmed bookings and assigned students so that I can prepare for upcoming sessions.	INSTRUCTOR	DONE	MD
~~SCRUM-11~~	As an instructor, I want to conduct and evaluate student assessments so that I can determine their readiness for certification.	INSTRUCTOR	DONE	M
~~SCRUM-12~~	As a recreational diver, I want to download and share my certification so that I can prove my qualifications to other dive centres.	RECREATIO...	DONE	M
~~SCRUM-24~~	As an instructor, I want to validate and approve student dive logs so that their training records are accurately maintained.	INSTRUCTOR	DONE	MD
~~SCRUM-25~~	As an instructor, I want to validate and approve student dive logs so that their training records are accurately maintained.	INSTRUCTOR	DONE	MD
~~SCRUM-36~~	As a recreational diver, I want to recognized hand signals under water so that, I can easily communicate with instructors under water	RECREATIO...	DONE	TS
~~SCRUM-22~~	As a guest user, I want to request an account so that I can enroll in a diving course or make a reservation.	GUEST USER	DONE	HC
~~SCRUM-29~~	As a certification authority, I want to review student performance records so that I can verify compliance with certification standards.	CERTIFICATI...	DONE	M
~~SCRUM-37~~	As a recreational diver, I want to track and identify marine life species encountered during my dive, So that I can learn more about the underwater ecosystem, log my sightings, and share them with the community.	RECREATIO...	DONE	TS
~~SCRUM-32~~	As a certification authority, I want to approve certifications so that only qualified divers receive credentials.	CERTIFICATI...	DONE	M
~~SCRUM-38~~	As a admin, I want to see the new booking requests and assign a available instructor.	INSTRUCTOR	DONE	MD

Backlog (0 work items)

Your backlog is empty.

2.3 Project Management Tools

Several modern tools and platforms were used to maintain transparency, collaboration, and quality throughout the project:

- **Jira:** Used to manage the product backlog, plan sprints, assign user stories, and track development progress. Each sprint was planned with story points, and burndown charts helped monitor task completion.
- **GitHub:** Source control was managed through private repositories. Branching strategies (feature, develop, and main branches) were used to isolate and integrate code safely. GitHub Issues and Pull Requests supported code review and collaborative debugging.
- **Google Meet / Microsoft Teams:** Weekly meetings, sprint reviews, and status updates were conducted via video conferencing platforms, ensuring regular communication among team members.
- **Gantt Chart & Task Sheets:** A comprehensive Gantt chart was created early in the project and updated after each sprint to reflect actual timelines and dependencies. This helped the team stay aligned with the academic deadline and adjust deliverables as needed.

2.4 Team Collaboration and Communication

The success of the DiveMaster project was strongly rooted in proactive communication and mutual accountability. Key practices included:

- Daily Standups (via chat or call) to report progress and blockers.
- Shared documentation (Google Docs, Notion) to centralize requirements, test cases, and design diagrams.
- Pair programming and cross-reviews for difficult modules or integration points, especially during instructor and equipment synchronization logic.

These collaborative strategies helped maintain a consistent pace and ensured that even when one member faced technical challenges, others could step in to assist, ensuring team-wide progress.

2.5 Risk Management

A basic risk register was maintained to proactively address potential issues such as:

- Module Dependency Risks: To mitigate delays, modules with high external dependency (e.g., scheduling) were developed early.

- Data Loss & Merge Conflicts: Frequent Git commits and regular backups were enforced.

- Time Constraints: A buffer was built into the Gantt plan to absorb minor delays and accommodate unexpected revisions.

By taking a risk-aware approach, the team successfully avoided major project delays and delivered the system within scope.

3. Development

The development phase of the DiveMaster system involved the application of structured software engineering practices, selection of appropriate technologies, and adherence to design principles that ensure modularity, maintainability, and future scalability. This section outlines the methodology adopted, system architecture, patterns used in design and development, and the rationale behind the selected technology stack.

3.1 Development Methodology

The project team adopted the Agile Scrum methodology to manage the development lifecycle. This iterative and incremental approach allowed the team to deliver the system in phases, ensuring that working features were demonstrated and validated at the end of each sprint.

- Agile was chosen because of its emphasis on:

- Customer feedback loops, which were simulated through advisor reviews and mock stakeholder evaluations.

- Adaptability, allowing features to be revised or added mid-cycle without breaking the system.
- Team collaboration, supported by regular stand-ups, sprint reviews, and retrospectives.

Each sprint lasted approximately two weeks, and sprint planning was done using user stories drawn from the product backlog. Tasks were broken down into manageable units, and each developer was assigned stories related to their core module. Regular commits, peer reviews, and integration testing ensured that continuous development did not disrupt stability.

3.2 System Architecture

DiveMaster was designed with a modular, layered architecture, separating concerns across the system into presentation, business logic, data access, and persistence layers. This allowed different team members to work on distinct components without interference while maintaining clear integration paths.

I. Frontend (Presentation Layer)

- Built using React.js for the web and mobile based application can be used under water.
- Provides dynamic user interfaces for each role: admin, instructor, diver.
- Connects to backend via secure RESTful APIs.

II. Backend (Business Logic Layer)

- Developed using Laravel, a PHP MVC framework.
- Responsible for authentication, authorization, business logic, and data processing.
- Exposes REST APIs for both web and mobile applications.

III. Database (Persistence Layer)

- Central system uses MySQL, optimized for relational operations.

IV. Synchronization & Integration Layer

- Offline-first sync logic handles reconciliation between local and server databases.
- Real-time scheduling conflict resolution and resource allocation.

This architecture supports horizontal scalability, is easy to deploy on a cloud-based stack, and facilitates future API integrations (e.g., integrating with international diving bodies or payment gateways).

3.3 Development Methodology

To ensure code quality, reusability, and separation of concerns, the following design patterns were implemented across the system:

- **Repository Pattern (Laravel)**
  Abstracts the data access layer, enabling decoupling between business logic and database queries. It makes the backend cleaner and more testable.
- **Observer Pattern**
  Used for triggering real-time notifications when changes occur (e.g., instructor assigned, dive log approved, schedule modified).
- **Strategy Pattern (Assessment Module)**

Allows flexible integration of multiple grading strategies for different types of assessments (e.g., MCQs, practical checklists, diving theory quizzes).

Each pattern was chosen based on its alignment with the system's needs and its ability to maintain readability and scalability as the codebase grows.

3.4 Technology Stack and Justification

The following technologies were selected based on reliability, developer familiarity, community support, and alignment with project requirements:

Category	Technology	Justification
Frontend	Laravel	Lightweight, component-based architecture; ideal for responsive dashboards.
Mobile Based Offline Solution	Flask	Cross-platform; supports offline data storage and rich UI elements and real time predictions.
Backend	Laravel (PHP)	Secure and elegant MVC structure; fast development and built-in features.
Database	MySQL	Proven relational database; ideal for structured data like certifications, logs, etc.
Version Control	Git & GitHub	Reliable collaboration and code review via PRs.
Project Management	Jira	Agile board for sprints, backlog, and task tracking.
Hosting	Shared cPanel Server	Accessible and affordable for academic deployment; SSL secured.
Testing	Selenium	Ensures stability and correctness of both backend and mobile logic.

By strategically combining these technologies, the system supports a hybrid online–offline environment and provides seamless user experiences across different devices and connection states.

4. Testing

Testing is a critical phase in the software development lifecycle, serving as the final quality gateway between implementation and delivery. For the DiveMaster – Diving Certification Management System (DCMS), the development team implemented a layered testing strategy that included unit testing, automated end-to-end testing, manual exploratory testing, and static code quality analysis. The goal was to ensure that the system was not only functionally correct but also secure, maintainable, and user-ready.

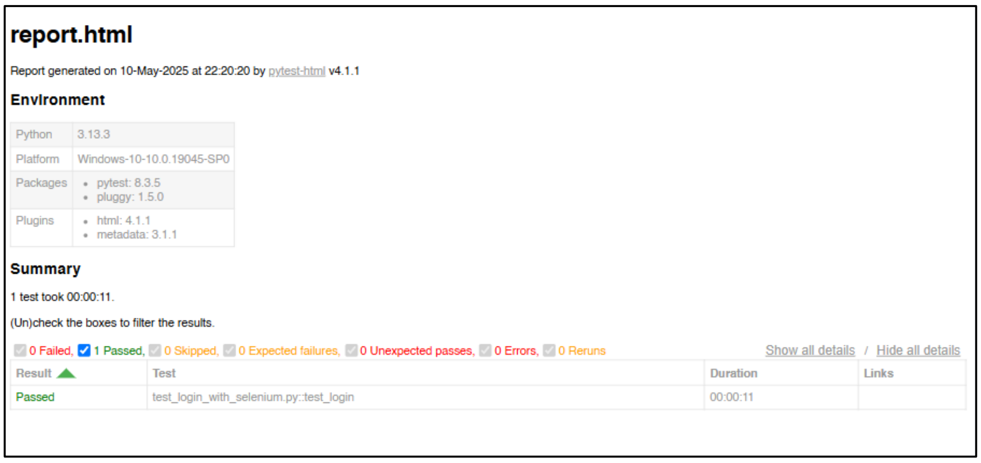
Due to the platform's role-based design and the wide range of modules it covers—from booking to dive logging, assessment, certification, and resource allocation—it was important to verify that each workflow worked both independently and in combination with others. Testing efforts were distributed across the team based on their component responsibilities, with shared participation in system-wide testing activities.

4.1 Unit Testing

Unit testing was conducted primarily on the Laravel backend, where business logic is centralized. The team used PHPUnit, Laravel's default testing framework, to validate controller methods, service logic, database interactions, and permission checks. Each test case was written to verify a single logical unit under expected and unexpected conditions.

Module	Test Objective
Authentication & RBAC	Ensure only valid users can access specific routes and functions based on roles.
Booking Engine	Verify that dive bookings are accepted or rejected based on availability.
Dive Logs	Confirm that dive entries follow required formats and validations.
Assessment System	Validate automated scoring, passing logic, and conditional certification triggers.
Certificate Issuance	Ensure that digital certificates are generated only when eligibility is confirmed.

These tests were run regularly after code merges, allowing the team to identify regressions quickly during ongoing development. Edge cases—such as attempting bookings with unavailable instructors or submitting incomplete dive logs—were prioritized to ensure robustness.



report.html

Report generated on 10-May-2025 at 22:20:20 by pytest-html v4.1.1

Environment

Python	3.13.3
Platform	Windows-10-10.0.19045-SP0
Packages	• pytest: 8.3.5 • pluggy: 1.5.0
Plugins	• html: 4.1.1 • metadata: 3.1.1

Summary

1 test took 00:00:11.

(Un)check the boxes to filter the results.

✓ 0 Failed, ✓ 1 Passed, ✓ 0 Skipped, ✓ 0 Expected failures, ✓ 0 Unexpected passes, ✓ 0 Errors, ✓ 0 Reruns Show all details / Hide all details

Result	Test	Duration	Links
Passed	test_login_with_selenium.py::test_login	00:00:11	

Figure 1; Unit testing report

4.2 Test Coverage Report

To assess how much of the system's backend logic was being validated by unit tests, the team maintained test coverage reports. This helped identify untested areas and guided further test writing.

Laravel Backend Test Coverage

Component	Coverage Level	Remarks
User Management & Auth	High (~95%)	Includes login, registration, role assignments, and password resets.

Dive Scheduling	High (~90%)	Covers instructor allocation, conflict checks, and booking logic.
Equipment Management	Moderate (~75%)	Mostly covered; dynamic alerts tested manually due to time constraints.
Assessments	High (~92%)	Full validation of quiz processing, pass/fail logic, and feedback display.
Reporting & Exports	Low (~60%)	Export logic tested manually; PDF generation partially automated.

Overall backend test coverage: ~85%

This level of coverage allowed the team to proceed with confidence, particularly for critical components like scheduling, assessments, and security.

4.3 Code Quality Assurance

Maintaining high code quality was a project priority from day one. Static code analysis, strict adherence to Laravel's MVC architecture, and consistent peer code reviews were employed to ensure clean, readable, and secure code.

Tools and Practices Used

Tool / Practice	Purpose
PHPStan (Level 5)	Static code analysis to detect type issues, dead code, and logic errors.
ESLint for Blade JS	Enforced consistent syntax and formatting in inline scripts.
DocBlocks & Comments	All functions and controllers documented for maintainability.
Pull Request Reviews	All code merged to the main branch was peer-reviewed.
Modular Structure	Each component (e.g., booking, logging) kept in isolated modules.

The team also followed PSR-12 coding standards, used dependency injection where applicable, and avoided hardcoded values in controllers or views.

4.4 Automated Functional Testing Using Selenium

To simulate full user interactions and validate the frontend's integration with backend logic, the team used Selenium WebDriver to automate browser-based testing of real-world user scenarios. These tests were designed to replicate how students, instructors, and administrators interact with the system.

Browser Testing Matrix

Browser	Status	Notes
Google Chrome	Fully Passed	Primary browser used for development/testing.
Mozilla Firefox	Fully Passed	Slight layout adjustments were validated.
Microsoft Edge	Fully Passed	No issues found.
Opera	Fully Passed	No issues found.

This form of automated functional testing helped the team validate the user interface and system interactions across multiple roles without having to manually test them after every deployment.

4.5 Code Quality Assurance

Manual testing was carried out to validate:

- Edge cases (e.g., double booking, unauthorized access, record duplication)
- PDF export features, where layout correctness was crucial
- Email triggers and notifications
- UI responsiveness and layout across devices

The team followed informal exploratory testing sessions weekly where each member used roles other than their assigned one to "stress test" the system by mimicking real-world user behavior. These sessions helped reveal logical gaps and minor usability issues not detected by automation.

5. Deployment

Deployment marks the final and essential phase in delivering the DiveMaster – Diving Certification Management System (DCMS) to end users. In this project, our aim was to ensure that the application is reliably accessible, secure, and runs efficiently under shared hosting constraints. The platform has been deployed to Hostinger’s shared Linux hosting environment, which offers full compatibility with Laravel and MySQL, and provides a cost-effective, reliable infrastructure suited for academic and small-to-medium production use cases.

5.1 Deployment / Hosting Mechanisms

Deployment Adjustments for Shared Hosting

- The Laravel public/ folder was moved to the root public_html/ directory, and index.php was updated to reference the correct internal paths (for vendor and bootstrap).
- Laravel environment configuration (.env) was placed outside the public_html directory to prevent exposure.
- Storage symlinks were recreated to allow safe file uploads via the web interface.
- Application logging was configured to use daily log rotation and reduced verbosity for production performance.

5.2 Deployment / Hosting Mechanisms

The chosen infrastructure is lightweight, cost-effective, and scalable to a moderate level of concurrent users. Below is a breakdown of the environment:

Component	Technology / Platform	Details
Hosting Provider	Hostinger (Shared Hosting)	Linux-based shared server with Laravel 10 support
Web Server	Apache with mod_rewrite enabled	Configured via .htaccess for clean URL routing
SSL Encryption	Let’s Encrypt (AutoSSL)	Enforces HTTPS across the entire application
Database	MySQL (MariaDB variant)	Used for persistent relational data storage
App Runtime	PHP 8.1	Fully supported by Hostinger for Laravel deployments
Domain	Custom domain linked via DNS	Used for public access and routing to public_html/
SSH / Terminal	Enabled via Hostinger hPanel	Used to execute Laravel commands, migrations, and optimizations

This infrastructure is sufficient for the current academic and pilot launch, with headroom to support dozens of active users and thousands of records without performance degradation.

5.3 Deployment Pipeline (Manual)

Due to the limitations of shared hosting (e.g., no CI/CD integration or Docker support), the deployment process was manual but structured and repeatable. The following pipeline was followed:

Manual Deployment Pipeline on Hostinger

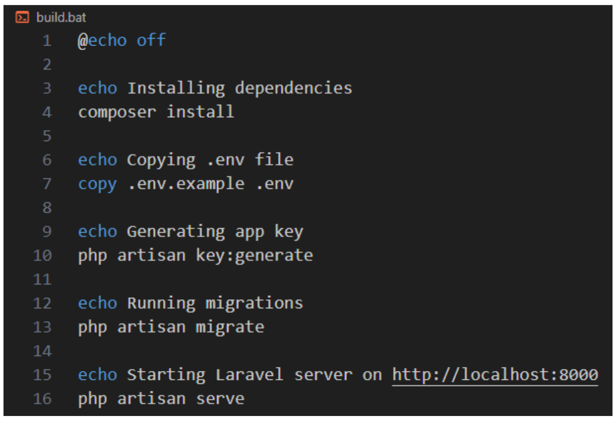
Step	Details
1. Code Packaging	Laravel project compressed locally; vendor folder excluded.
2. File Upload	ZIP file uploaded to Hostinger via File Manager or FTP.
3. Extraction & Linking	public folder content moved to public_html; paths updated in index.php.
4. Dependency Installation	composer install run via SSH.
5. Database Setup	New database and user created via hPanel, linked in .env.
6. Migrations & Seeders	php artisan migrate --seed to build tables and test data.
7. Caching Config	php artisan config:cache, route:cache, and view:cache executed.
8. Permissions Check	storage and bootstrap/cache folders set to writable.
9. SSL & Domain Binding	Free SSL certificate issued; domain linked to public folder.
10. Testing & Verification	Admin login, bookings, scheduling, and certificate flows were re-tested.

5.4 Future Improvements & Scalable Hosting Considerations

While Hostinger serves the current needs well, the following improvements can be explored in the future to scale the project:

- VPS Upgrade or Cloud Migration: Transition to platforms like DigitalOcean, AWS, or Heroku for increased control, automation, and performance.

- CI/CD Integration: Set up automated build and deploy pipelines using GitHub Actions or GitLab CI for smoother rollouts.



build.bat

```
@echo off

echo Installing dependencies
composer install

echo Copying .env file
copy .env.example .env

echo Generating app key
php artisan key:generate

echo Running migrations
php artisan migrate

echo Starting Laravel server on http://localhost:8000
php artisan serve
```

Figure 2: Auto configuration build file

- Docker Support: Containerization of the application to ensure uniform environment replication during scaling.

- Monitoring & Alerts: Integrate application monitoring tools (e.g., Laravel Telescope, Sentry) for runtime diagnostics.

6. Reflection

The development of the DiveMaster – Diving Certification Management System has been a transformative academic experience for each member of the CodeCrafters team. Beyond technical growth, the project offered valuable insight into team collaboration, client-oriented thinking, time management, and real-world software engineering practices. Each member was responsible for a distinct functional component, while also contributing collectively to system integration, testing, documentation, and deployment. Below are the individual reflections of the team members, showcasing what they learned, the challenges faced, and how this experience has shaped their future approach to software projects.

6.1 MS24905336 – Singh W M T R (Dive Logs and Marine Life Tracking)

Working on the dive log and marine life tracking component allowed me to explore how structured data collection plays a vital role in both recreational and scientific diving. My responsibilities included designing the dive log interface, managing dive-specific metadata, and implementing logic for instructor validations. I learned how to build forms that are both dynamic and user-friendly, and how to structure models and relationships to capture real-world data like dive depth, marine observations, and buddy divers.

One of the major challenges I encountered was aligning the system's logic with real-time gesture recognition. I had to collaborate closely with others to make sure my module integrated well with instructor and student workflows. This project taught me the value of modular, clean code and the importance of constant communication in a team. It strengthened my backend development skills and gave me practical experience working in a feature-driven sprint environment.

6.2 MS25914078 – Premarathne H M P D (Scheduling and Resource Allocation)

Handling the scheduling and resource allocation module gave me a deep appreciation for the complexity of real-time coordination systems. The need to dynamically assign instructors based on availability while preventing conflicts required building smart logic that could adapt to constant change. Designing an efficient resource allocation engine that worked in harmony with booking flows was one of my most technically challenging but rewarding tasks.

I also handled instructor response management and integration. One key takeaway was the importance of designing systems that handle edge cases gracefully—like last-minute cancellations, instructor reassignments, or equipment unavailability. This project helped sharpen my critical thinking, testing strategies, and Laravel development skills. I now better understand how scheduling logic and user interaction drive the overall user experience of an application.

6.3 MS24905954 – Kaushalya M K B (Assessment and Certification)

My responsibility was to design and implement the assessment and certification system—an essential part of the DiveMaster platform. The goal was to build a flow that included quiz generation, grading logic, progress validation, and automated certificate generation. One of the highlights was implementing QR-coded digital certificates that could be validated independently of the system.

A major challenge was designing assessment modules flexible enough to support different diving course structures and allowing instructors to track and evaluate students efficiently. This experience improved my understanding of secure evaluation systems, dynamic data relationships, and condition-driven workflows. It also taught me how vital accurate progress tracking and feedback are in educational tools. I feel more confident now in designing complex learning flows and integrating them into larger systems.

6.4 MS24912570 – Weerapana W M G D (Equipment Inventory Management)

Building the equipment inventory and maintenance module offered me a unique chance to understand lifecycle-based tracking systems. This module needed to manage diverse gear, track usage statistics, monitor maintenance schedules, and allow administrators to make informed decisions. I learned to build interfaces that provide quick insights into inventory status and alerts to help prevent safety risks due to overlooked maintenance.

The biggest challenge was maintaining data integrity while allowing flexibility in usage logs and allocation. This required well-structured models and careful attention to user permissions. I also had to think critically about how to design a reporting mechanism that translated raw data into meaningful information. This component pushed me to improve my Laravel backend skills, my understanding of relational data modeling, and how to balance user convenience with operational safety.

6.5 MS25913774 – Gamlath M G H C (User Management and Booking System)

My main focus was on designing the core authentication system, user profile management, and booking workflows. I worked on role-based access control (RBAC), secure login mechanisms, and building a booking system that allowed students and divers to reserve training sessions with real-time validation. This was the foundational layer for almost all system flows, so I had to coordinate with every team member to ensure alignment.

The most valuable skill I gained was implementing secure, scalable user management in Laravel. I also learned how to manage booking logic that involves multiple moving parts—users, sessions, instructors, and equipment. This project strengthened my system design thinking and made me more comfortable working across multiple modules and data flows. I now feel more equipped to lead user-centric backend architecture in future projects.

7. Conclusion

The development of the DiveMaster – Diving Certification Management System has been a holistic experience that combined technical implementation with structured project management principles. This project was not just about building a software solution; it was a comprehensive exercise in understanding and applying core software engineering methodologies, agile practices, testing techniques, and automation tools within a real-world scenario.

At its core, this module emphasized the complete software lifecycle—from the initial requirements analysis and stakeholder identification to planning, development, testing, and deployment. Through each phase, we were required to make informed decisions, adapt to emerging challenges, and ensure that the final product met both functional and non-functional requirements. The system itself was grounded in real-world use cases, representing a fully-featured solution for managing diving certifications, bookings, instructor scheduling, dive logging, assessments, and digital certificate generation.

One of the most significant academic gains was the opportunity to apply agile project management in a practical setting. By using Scrum-based sprints, Jira boards, product backlogs, and Gantt charts, we learned how to break large goals into manageable tasks, track progress incrementally, and deliver functionality iteratively. This approach made the development process not only efficient but also more aligned with professional software development workflows.

From a technical standpoint, we gained extensive experience in working with modern frameworks like Laravel, implementing role-based access control, designing relational databases, and structuring code for scalability and maintainability. We also explored the importance of modular design, integrating multiple components—such as scheduling, inventory, and assessment—into a unified and coherent system architecture.

A key focus of this project was also on software quality assurance, including automated testing using PHPUnit and Selenium, code quality checks with static analysis tools, and manual exploratory testing to cover edge cases. These practices reinforced the value of automation in reducing bugs, improving performance, and ensuring consistent behavior across the application.

Among the key learning outcomes throughout this project were:

- The importance of early and continuous communication in a team-based development process.
- How user-centric design can improve system usability and reduce friction for different user roles.

- The value of testing at every level, from units to full workflows, in delivering stable and reliable software.

- Realizing how deployment constraints (like shared hosting) influence system architecture and delivery approaches.

- The necessity of flexibility and adaptability when requirements evolve or technical challenges arise mid-development.

In conclusion, this project was more than an academic requirement—it was a transformative learning experience that equipped us with not just technical knowledge, but the mindset, tools, and confidence needed to manage and execute full-scale software projects in a real-world environment. It provided the bridge between theory and practice and prepared us for future roles as responsible, capable, and collaborative software engineers.

Appendix - A

GitHub Link:

https://github.com/HasanChamara/DiveMaster-Diving-Certificates-Management-System.git