CONTENTS Page No.

1.Abstract

2.Acknowledgement

3.Certificate from Company

4.Student declaration

5.Introduction

6.Company profile

7.Scope of the system

7.1 Existing system

7.2 Problem statement

7.3 Proposed system

8.System analysis

8.1 Review of literature

8.2 Review of research

8.3 Feasibility study

8.4 Tools Technology

8.5 Hardware and Software requirements

9. Planning

9.1 Overall description

9.1.1 Product perspective

9.1.2 Product functions

9.2 User modules

9.3 Functional and Non-functional requirements

10. System Design

10.1 System perspective

10.2 DFD diagrams

10.3 UML diagram.

10.4 Database diagram

11. Screen shots

12.Software testing

12.1 Test cases

12.1.1 Test case for Employee

12.1.2 Test case for Admin

13.Conclusion

14.Limitations

15.Future enhancement

16.Reference

17.Appendix

17.1 User manual

18. Bibliography

1. **ABSTRACT**

The **Web-Based Learning Management Solution** (LMS) is a practical and scalable platform designed to make teaching, training, and knowledge-sharing efficient, interactive, and personalized. From content creation and course delivery to learner **progress tracking and automated reminders**, the system ensures that learning activities are seamlessly managed and monitored. In today’s fast-evolving digital environment, having a reliable LMS is essential for organizations and institutions to streamline training, improve learner engagement, and enhance knowledge retention.

Traditional training methods, often dependent on scattered documents and manual tracking, frequently result in inefficiencies, missed deadlines, and difficulties in assessing performance. This solution addresses those challenges by providing a centralized platform with **role-based access**, **dynamic dashboards, real-time progress reporting, and AI-powered quiz generation** for better learning outcomes.

Built using the **MERN stack (MongoDB, Express.js, React.js, and Node.js)**, the LMS combines responsive front-end design with robust back-end processing and a scalable database structure. Key features such as **admin-controlled content management, personalized learner dashboards, automated email reminders, and Gemini API–powered quiz generation** not only improve usability but also increase accountability and ensure effective monitoring of learner progress.

1. **ACKNOWLEDGEMENT**
2. **CERTIFICAT FROM COMPANY**
3. **STUDENT DECLARATION**
4. **INTRODUCTION**
5. **COMPANY PROFILE**
6. **SCOPE OF THE SYSTEM**

The scope of the LMS covers the needs of **administrators, learners, and organizations** by offering a **flexible, scalable, and user-friendly platform** for digital learning:

* **For Administrators:**
  + Manage users, courses, and content efficiently.
  + Generate quizzes automatically using the **Gemini API**.
  + Access reporting dashboards for real-time progress and performance.
  + Send **automated and manual email reminders** to learners.
* **For Learners:**
  + Access a **personalized dashboard** with assigned courses.
  + View progress, deadlines, and completion status.
  + Explore a course catalog with **search and filter functionality**.
  + Attempt quizzes and track scores instantly.
* **For Organizations:**
  + Streamline the **training and knowledge-sharing process**.
  + Improve **accountability and decision-making** with accurate reports.
  + Reduce reliance on manual tracking, saving **time and administrative effort**.
  + Enhance **learning effectiveness and compliance monitoring**.
  1. **Existing system**

Currently, learning and training management in many organizations is handled through **manual methods**, such as shared documents, email communications, and basic record-keeping tools. While these methods provide a simple way to share content, they often cause **inconsistencies, delays, and poor learner engagement**. Learners struggle to track their progress, and administrators face challenges in monitoring completion, sending reminders, and generating accurate reports. As the number of learners and courses grows, these limitations result in **inefficient training delivery, reduced accountability, and poor decision-making**.

* 1. **Problem statement**

The absence of a centralized and automated system for learning management creates major operational challenges. Manual tracking of courses and learners increases the risk of missed deadlines, incomplete progress data, and low participation rates. Administrators spend significant time preparing reports and sending reminders, while learners often lack clarity about their assigned courses and progress. This leads to wasted time, lower training effectiveness, and difficulties in performance evaluation, especially when quick and accurate data is required for compliance and decision-making.

* 1. **Proposed system**

The proposed **Web-Based Learning Management Solution (LMS)** provides a **centralized, automated platform** for managing the complete learning lifecycle, from **content creation and course delivery to progress tracking and reporting**. It enables administrators to **upload and manage content**, create structured courses, and generate quizzes using the **Gemini API**. Learners get **personalized dashboards** showing mandatory and optional courses, deadlines, and progress status. Additional features like **real-time reporting, automated email reminders, and role-based access** make the system reliable and efficient. This solution overcomes the limitations of manual methods, reduces administrative effort, and improves overall **training effectiveness and accountability**.

1. **SYSTEM ANALYSIS**

System analysis is an essential step in developing the **Web-Based Learning Management Solution (LMS)**. It evaluates the current learning management practices, identifies existing gaps, and defines the requirements for creating a centralized, automated, and scalable platform. This stage ensures that the proposed solution is **technically feasible, cost-effective, and aligned with organizational training needs**.

* 1. **REVIEW OF LITERATURE**

Learning Management Systems (LMS) and e-learning platforms have been widely studied in the fields of **education technology and organizational training**. Research highlights that traditional learning methods relying on **manual tracking, shared files, or email-based communication** often result in **inconsistent progress tracking, low engagement, and poor accountability**.

Key findings from literature emphasize:

* **Centralized learning platforms** to improve accessibility and monitoring.
* **Role-based dashboards** to cater to admins, instructors, and learners.
* **Automated reporting and notifications** to enhance learner participation.
* **AI-powered assessments** for dynamic and scalable evaluation.

Studies consistently support **web-based solutions** with **user-friendly interfaces, scalable databases, and automation features**, which directly improve learning effectiveness and organizational training outcomes.

* 1. **REVIEW OF RESEARCH**

Recent research in e-learning systems and enterprise training shows that adopting **cloud-enabled and web-based LMS platforms** results in:

* **Higher learner engagement** through personalized dashboards.
* **Improved compliance and reporting** with real-time analytics.
* **Reduced administrative workload** via automated reminders and progress tracking.
* **Smarter assessments** with AI-based quiz generation.

Modern technology stacks, such as the **MERN stack (MongoDB, Express.js, React.js, Node.js)**, are proven to enhance **system scalability, responsiveness, and user experience**. These findings validate the **technological choices** made for this LMS project.

* 1. **FEASIBILITY STUDY**

|  |  |
| --- | --- |
| **Type of Feasibility** | **Key Findings** |
| **Technical** | The project uses the **MERN stack** (MongoDB, Express.js, React.js, Node.js), ensuring compatibility with existing IT infrastructure and supporting scalability. |
| **Operational** | A **user-friendly interface** with role-based dashboards and automated workflows ensures smooth adoption with minimal training. |
| **Economic** | The development cost is lower than the long-term expenses of manual training and tracking, making the system **cost-effective**. |

* 1. **TOOLS TECHNOLOGY**

The LMS uses a **modern web-based technology stack** to ensure performance, scalability, and maintainability.

* **Frontend:** ReactJS with Tailwind CSS for a **responsive and interactive user interface**.
* **Backend:** Express.js and Node.js to handle **business logic and APIs**.
* **Database:** MongoDB for **scalable and flexible data storage**.
* **AI Integration:** Gemini API for **automated quiz generation**.
* **Other Tools:**
  + Axios for API communication.
  + Nodemailer/SendGrid for automated email reminders.
  + Git/GitHub for version control and collaboration.
  + Docker (optional) for containerized deployment.
  1. **HARDWARE AND SOFTWARE REQUIREMENTS**

**Hardware:**

|  |  |
| --- | --- |
| **Component** | **Specification** |
| **Server** | Quad-core processor, 8 GB RAM, 250 GB SSD storage |
| **Client Systems** | Dual-core processor, 4 GB RAM, modern web browser |
| **Network** | Stable internet connection (LAN/Wi-Fi) |

**Software:**

|  |  |
| --- | --- |
| **Category** | **Specification** |
| **Operating System** | Windows / Linux / macOS |
| **Frontend** | ReactJS, Tailwind CSS |
| **Backend** | Node.js , Express.js |
| **Database** | MongoDB |
| **AI Integration** | Gemini AI |
| **Tools** | Git, VS Code, Docker (optional), Nodemailer |

1. **PLANNING**

Planning is a crucial stage in developing the **Web-Based Learning Management Solution (LMS)**. It sets the main objectives, defines system functions, and outlines how the product will work in real-world situations. The purpose is to ensure that every feature, from **course creation** to **progress tracking and reporting**, is clearly understood before implementation.

* 1. **OVERALL DESCRIPTION**

The **Learning Management Solution (LMS)** is a **centralized, web-based application** designed to simplify the complete lifecycle of digital learning. It provides organizations with a single platform to **create, deliver, and monitor courses**, while also tracking learner progress and assessments. This ensures **real-time updates, structured learning paths, and improved accountability**.

The system removes the inefficiencies of manual training and email-based communication by offering **automated workflows, user-specific dashboards, and detailed reporting**. This reduces administrative workload, enhances learner engagement, and supports **better decision-making** for organizational training programs.

* + 1. **PRODUCT PERSPECTIVE**

The proposed LMS acts as an **independent web application** that integrates smoothly with an organization’s existing IT setup. It follows a **client-server model**:

* **Frontend:** A responsive **ReactJS interface** with Tailwind CSS for learners and administrators, accessible via desktops, laptops, or mobile devices.
* **Backend:** **Node.js and Express.js** for API handling, quiz generation workflows, and business logic.
* **Database:** **MongoDB** for scalable and flexible data management.
* **AI Integration:** **Gemini API** for automated quiz generation from course content.

The product is **scalable**, supporting the addition of new courses, user roles, and groups as the organization grows. It runs on standard web browsers, requiring no special hardware on the client side.

* + 1. **PRODUCT FUNCTIONS**

The LMS provides the following key functions:

* **Content Management:** Admins can upload documents, videos, and presentations, and create structured courses with a rich-text editor.
* **Personalized Dashboards:** Learners get a customized dashboard with assigned courses, deadlines, and progress indicators.
* **Course Catalog & Search:** Users can browse and filter available courses by category, difficulty, or deadline.
* **AI-Powered Quiz Generation:** Admins can upload Word documents to automatically generate multiple-choice quizzes using the Gemini API.
* **Progress Tracking:** Real-time monitoring of course completion, quiz results, and learner performance.
* **Automated Email Reminders:** Notifications for upcoming deadlines or incomplete courses.
* **Reports & Analytics:** Detailed dashboards for admins to evaluate learner progress and training effectiveness.
  1. **USER MODULES**

The LMS is designed with **role-based modules** to ensure controlled access and clear responsibilities:

|  |  |
| --- | --- |
| **User Type** | **Key Responsibilities** |
| **Admin** | Manage users, upload content, create courses, generate quizzes, track learner progress, send reminders, and generate reports. |
| **Manager/Supervisor** | Monitor learner performance within teams, review reports, and ensure compliance with deadlines. |
| **Employee/User** | Access assigned courses, track progress, complete quizzes, and explore optional learning material |

* 1. **FUNCTIONAL AND NON-FUNCTIONAL REQUIREMENTS**

**Functional Requirements**

The system must:

* Allow admins to **add, edit, and delete courses and learning content**.
* Provide **user authentication with role-based access control**.
* Enable learners to view their **personalized dashboard** with course details, deadlines, and progress.
* Automatically generate **quizzes from uploaded documents** using the Gemini API.
* Send **email notifications and reminders** for deadlines and incomplete courses.
* Track **user progress, quiz scores, and completion status**.
* Generate **real-time reports and analytics** for admins.

**Non-Functional Requirements**

The system should:

* **Performance:** Handle multiple users accessing courses and quizzes simultaneously with low response time.
* **Scalability:** Support a growing number of users, courses, and content without major redesign.
* **Security:** Protect sensitive user and course data with encryption and role-based authentication.
* **Reliability:** Ensure consistent uptime with automatic backups for user and course data.
* **Usability:** Provide an intuitive, easy-to-use interface for both learners and admins.
* **Compatibility:** Run smoothly on major web browsers (Chrome, Firefox, Edge, Safari) and operating systems (Windows, Linux, macOS).

**10. SYSTEM DESIGN**

The System Design phase translates the functional requirements into a concrete technical blueprint for the **Web-Based Learning Management Solution (LMS)**. This phase defines the overall architecture, data flows, component interactions, and database structure to ensure the system is **scalable, secure, maintainable, and performant**.

**10.1 SYSTEMS PERSPECTIVE**

The LMS is designed as a multi-tier, web-based application following a client–server architecture with an optional AI integration layer for quiz generation:

**Client Layer (Frontend):**

* Built with **ReactJS** and Tailwind CSS for a responsive, accessible UI.
* Provides role-based views: **Admin**, **Instructor (optional)**, **Learner**, and **Manager**.
* Handles authentication flows, dashboard rendering, course playback, quiz interfaces, and file uploads (documents/videos).
* Communicates with backend over secure HTTPS REST/GraphQL APIs using Axios or fetch.

**Application Layer (Backend):**

* Implemented with **Node.js + Express.js** (or similar).
* Exposes RESTful (or GraphQL) APIs for user management, course/content management, quizzes, progress tracking, reporting, and notifications.
* Implements business logic: course lifecycle, quiz generation workflow, role-based authorization, progress calculations, scheduler for reminders, and email integration (SendGrid/Nodemailer).
* Contains an **AI Integration Module** that: accepts uploaded Word documents, sends content to the **Gemini API**, parses responses, and creates quiz questions/answers.
* Includes background workers (e.g., using Bull / Node cron / serverless functions) for heavy tasks: document parsing, quiz generation, email reminders, and report generation.

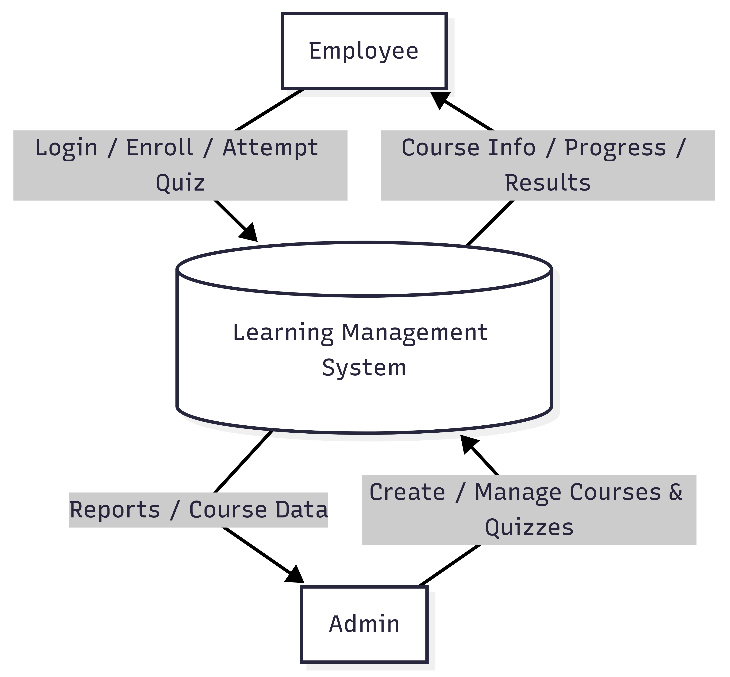
**Data Layer (Database & Storage):**

* **Primary DB:** **MongoDB** for flexible, document-oriented storage of users, courses, chapters, progress, and quizzes.
* **File Storage:** Cloud object storage (S3 / compatible) for videos, large documents, and attachments.
* **Search/Indexing (optional):** ElasticSearch / MongoDB Atlas Search for course catalog search and filters.
* **Caching (optional):** Redis for session caching, rate-limiting, and background-job locking.

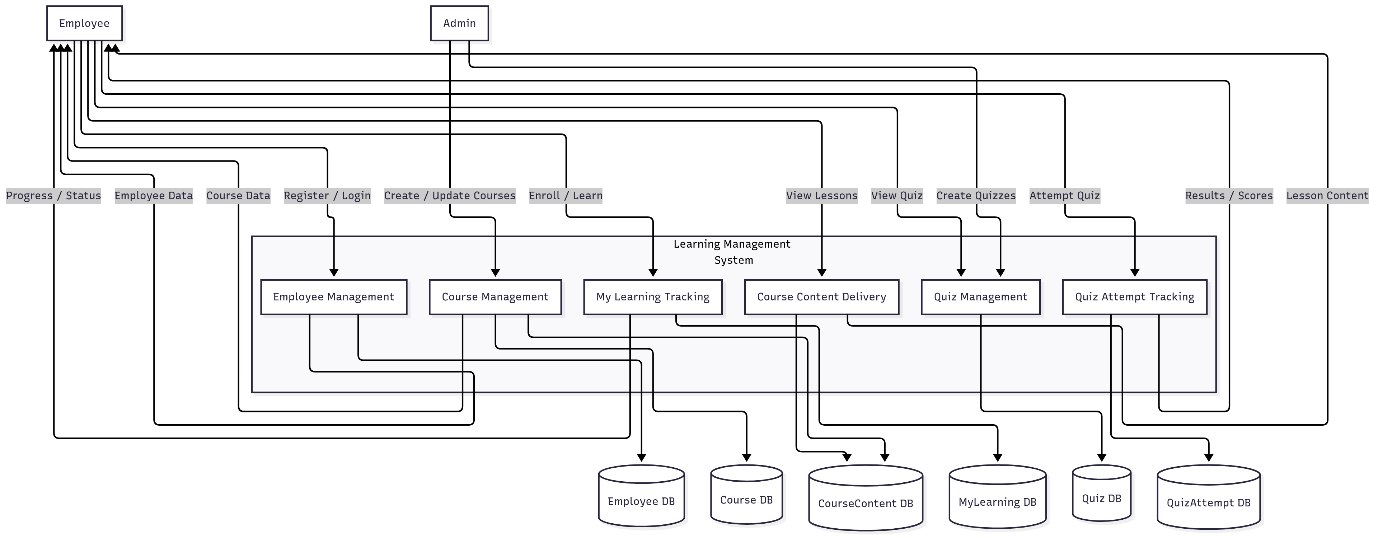
**Security & Infra Considerations:**

* HTTPS/TLS, JWT/OAuth for authentication tokens, role-based access control (RBAC), input validation, rate limiting, and secure file handling.
* Containerized deployment (Docker) and CI/CD pipeline for automated testing and deployment (Vercel/Heroku/GCP/AWS).

This architecture supports real-time updates, automated tasks, and secure integration with AI services while remaining horizontally scalable.

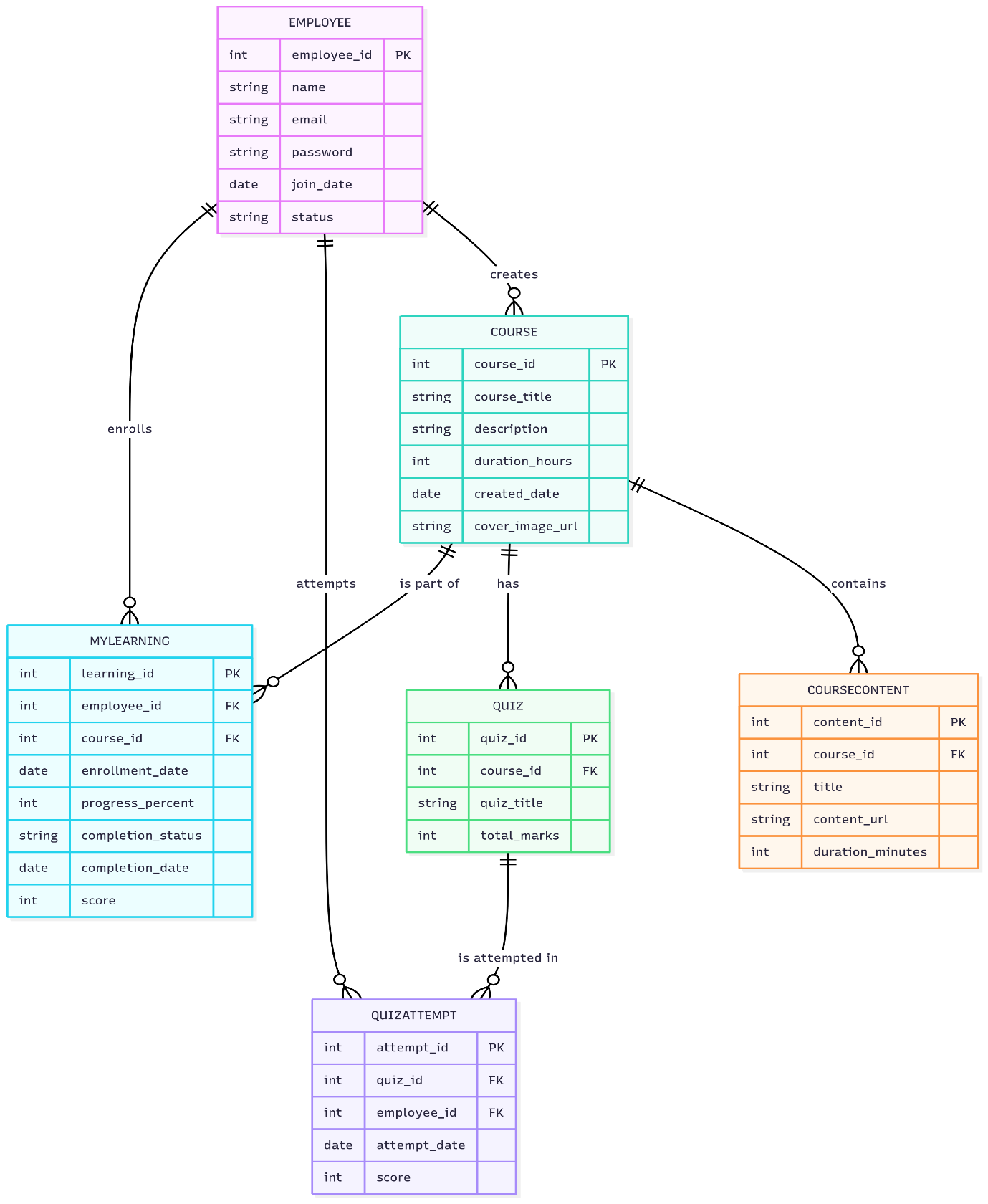
**10.2 DFD DIAGRAMS**

**Fig 1. DFD Level 0**



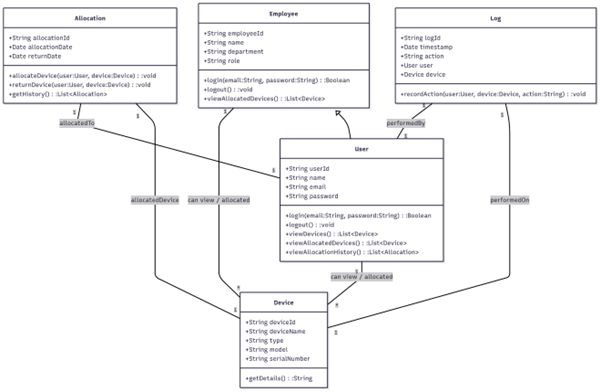
**Fig 2 . DFD Level 1**

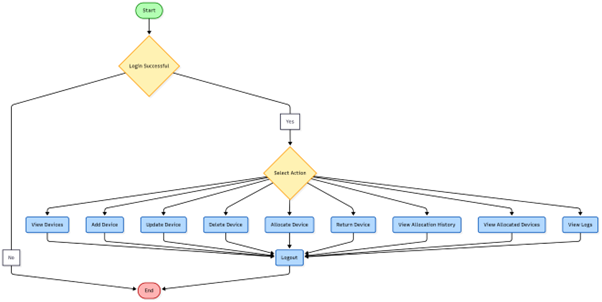
**Fig 3. DFD Level 2**



**Fig 4. ER Diagram**

**10.3 UML DIAGRAM**



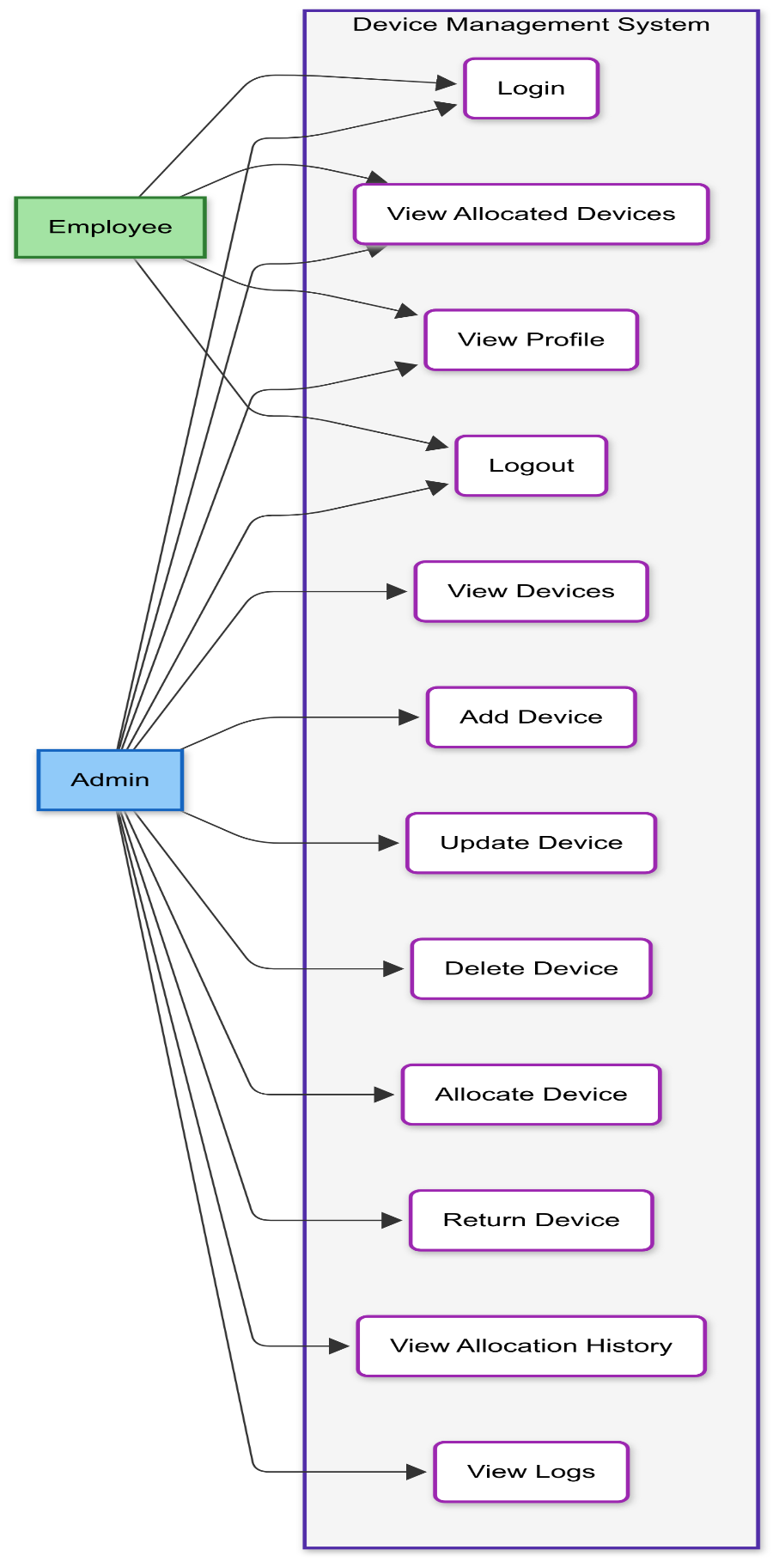
**Fig 1. Class Diagram**

**Fig 1. Activity Diagram**

A white sheet of paper with text

AI-generated content may be incorrect.

**Fig 1. Sequence Diagram**



**Fig 1. Usecase Diagram**

**A diagram of a computer program

AI-generated content may be incorrect.10.4 DATABASE DIAGRAM**

**Fig 1. Database Diagram**

**11. SCREEN SHOTS**

**Employee access**

**A screenshot of a computer

AI-generated content may be incorrect.**

Fig. 1: Employee Dashboard

This dashboard displays the user’s name and ID, and lists all assigned devices (monitor, keyboard & mouse, headset, laptop and Dock) with their make, model, serial number, and deployed status. The page is read-only for tracking assets—no editing or deleting is allowed.

A screenshot of a computer

AI-generated content may be incorrect.

Fig. 2: Employee Profile

This profile page displays the employees name, email, employee number, designation, department, supervisor, manager, and date of joining.

**User/Admin access**

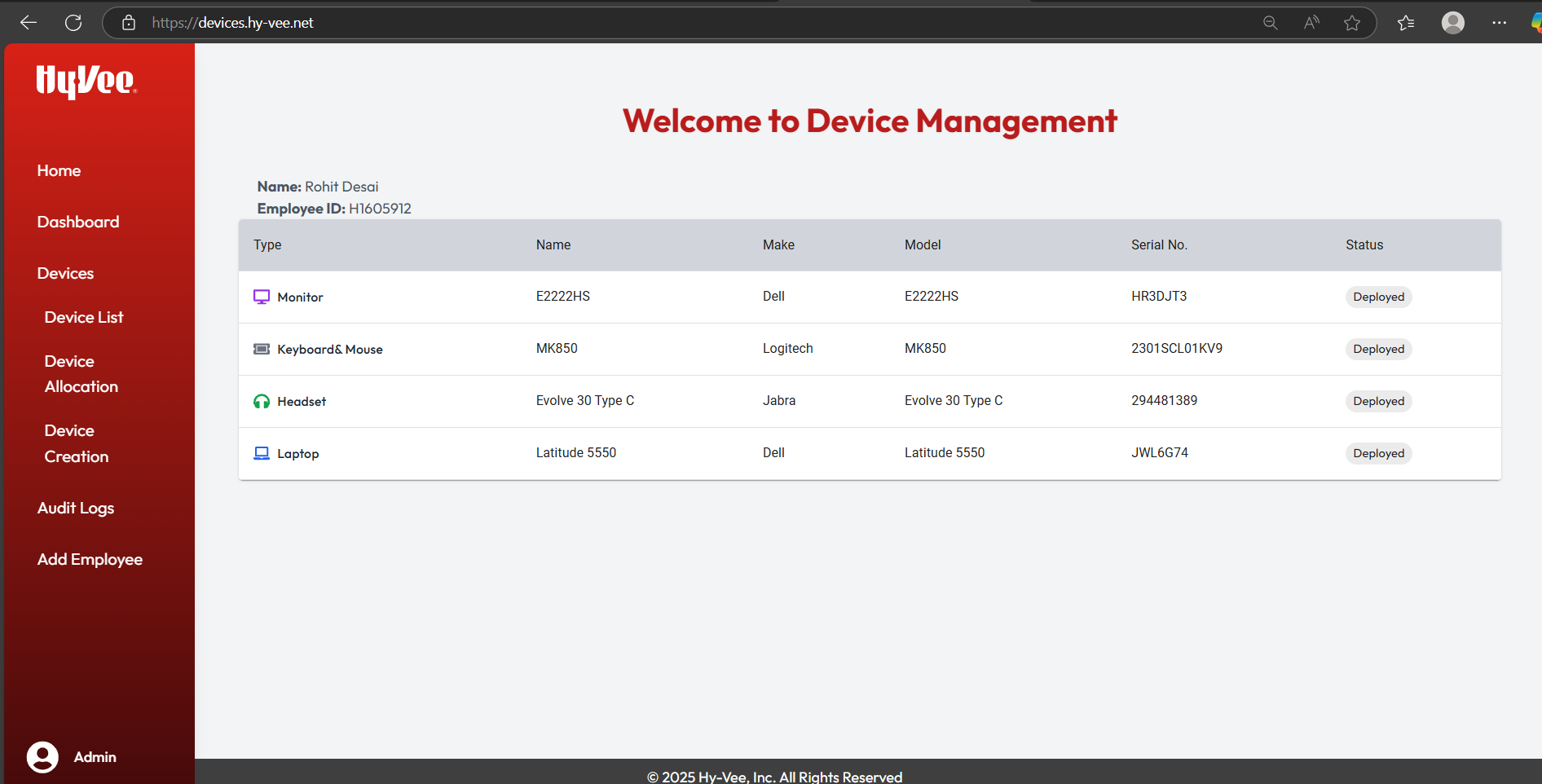


Fig. 3: Admin Dashboard

This dashboard shows the user’s name, ID, and assigned devices with full details in a read-only view. Unlike the employee view, the sidebar here has extra options like Dashboard, Device List, Device Allocation, Device Creation, Audit Logs, and Add Employee giving the user more privileges and admin access.

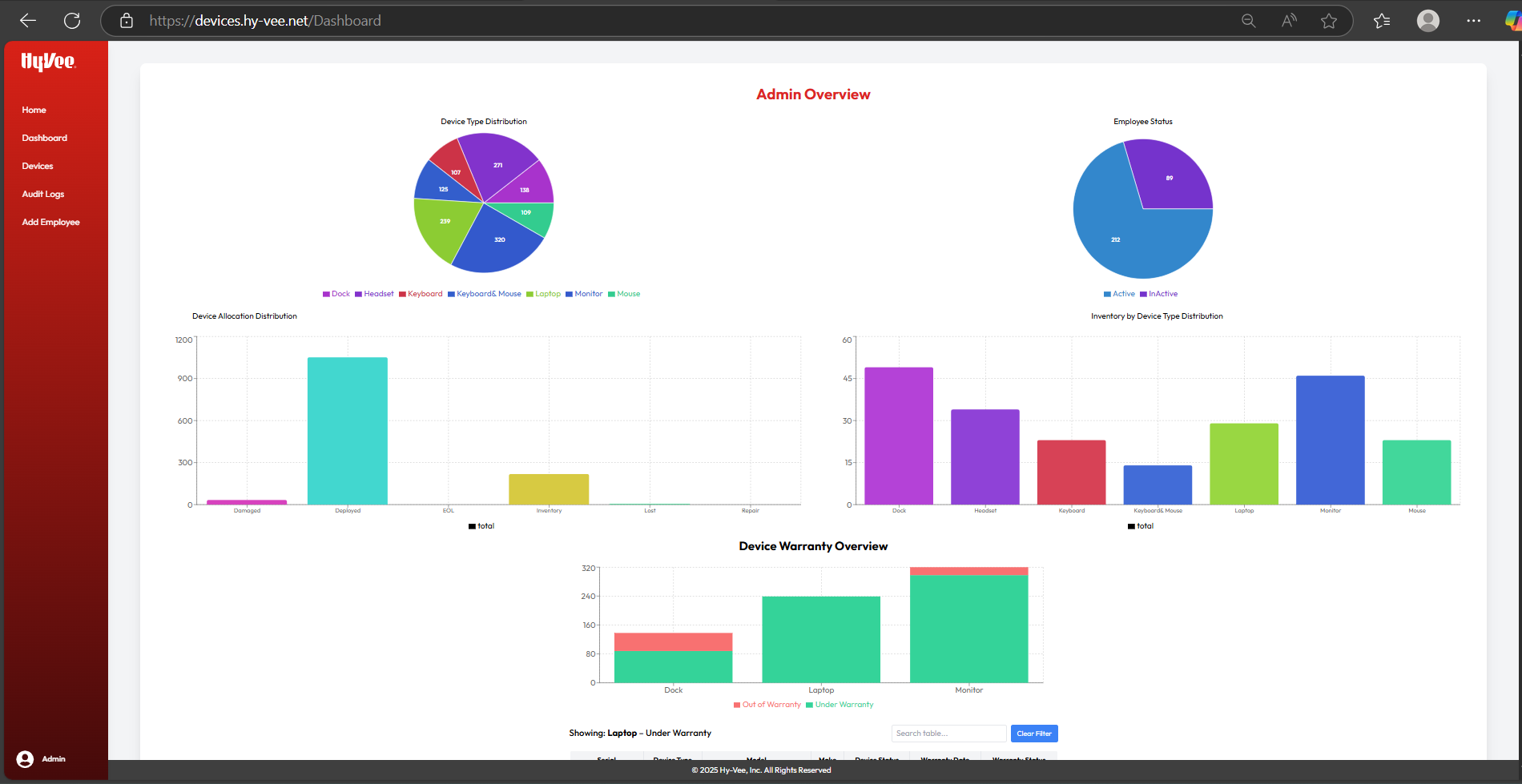


Fig. 4: Admin Analytics Dashboard

This dashboard provides an admin overview using various charts and graphs. It displays device type distribution, employee status, device allocation, inventory by device type, and device warranty, offering admins a clear snapshot of asset usage, employee allocation, and warranty status across the organization.

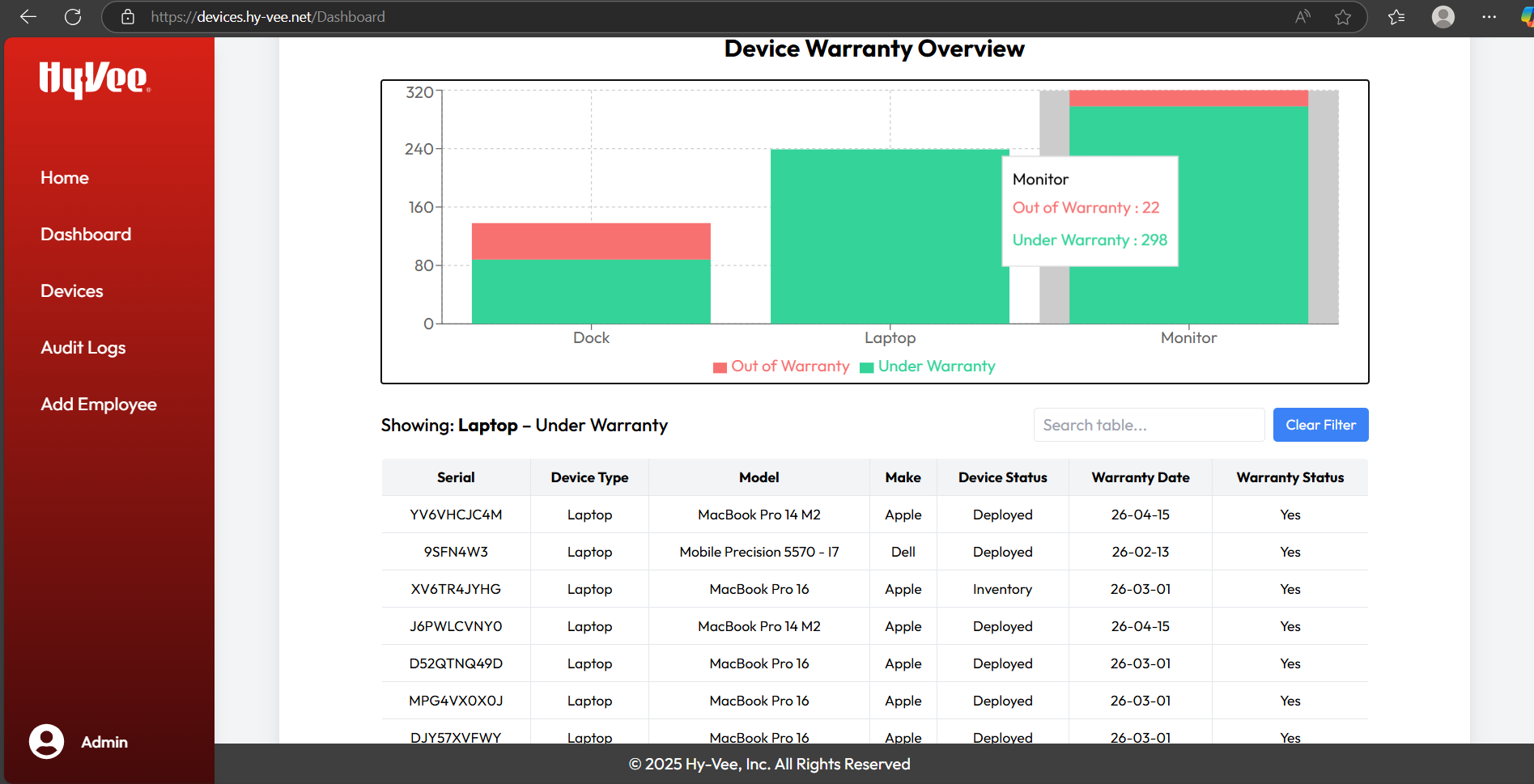


Fig. 5: admin analytics dashboard

This section displays a bar chart for device warranty status showing which devices are "under warranty" (green) or "out of warranty" (red). When you click on either the green or red section of a bar, the table below updates to show detailed information for that category. For example, clicking the green bar for laptops will list all laptops currently under warranty with.

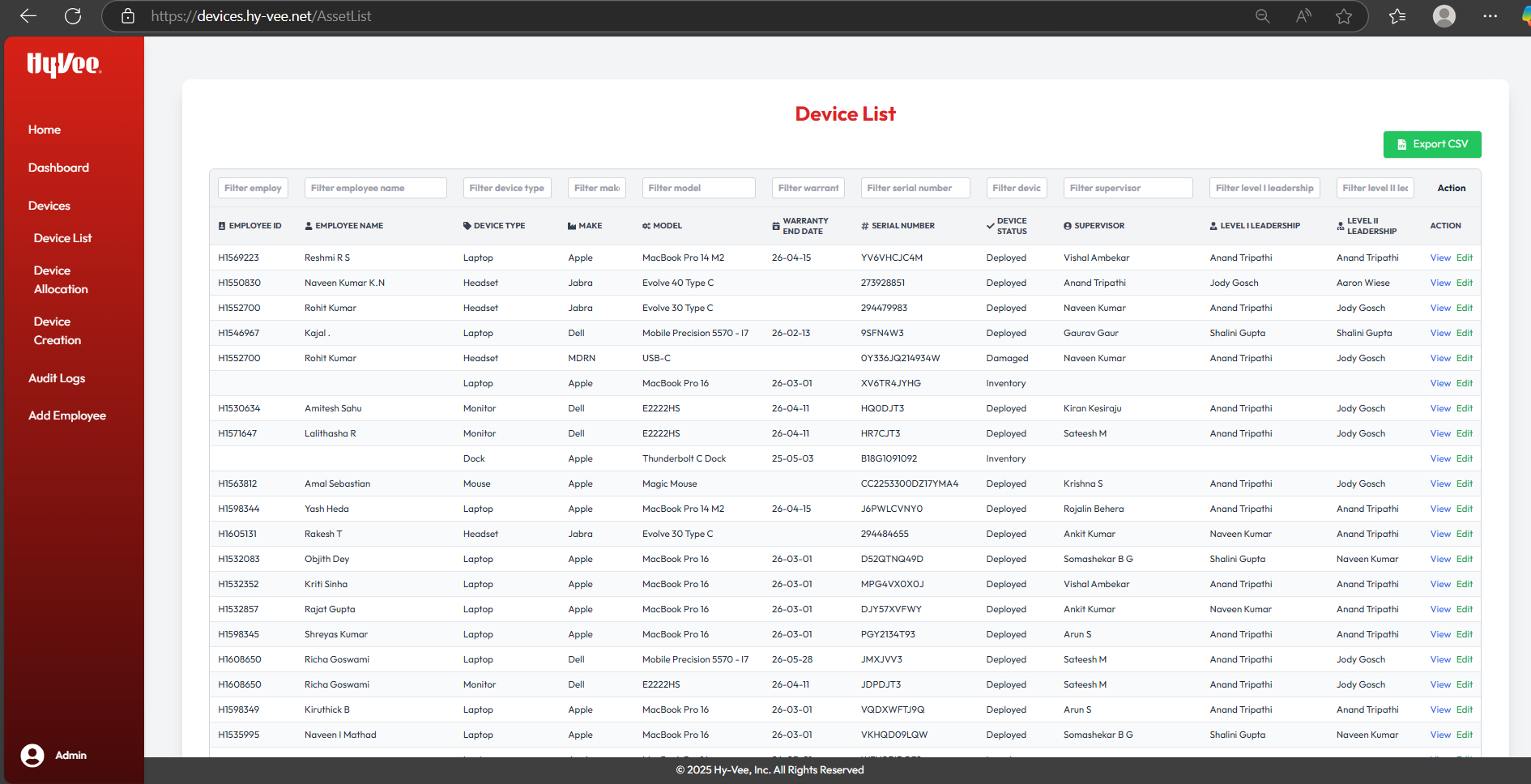


Fig. 5: Device List

This page shows a comprehensive list of all devices in the organization, including device details and leadership hierarchy. Users can filter devices by various parameters and export the data to CSV. “View” and “Edit” options allow admins to see or modify device details as needed.

A screenshot of a computer

AI-generated content may be incorrect.

Fig. 6: Device View

This page shows detailed information about a specific asset, including employee details, asset specifications, purchase and warranty info, device allocation history, and related Jira history.

A screenshot of a computer

AI-generated content may be incorrect.

Fig. 7: Device Edit

This page lets admins edit device details. After submission, updates are reflected in the system and visible in the Device List and Device View.

A screenshot of a computer

AI-generated content may be incorrect.

Fig. 8: Device Allocation Form

This page allows admins to allocate devices by selecting an employee, device type, device from inventory, device status, and entering a Jira ID and remarks. Once the Jira ID is validated, the submit button is enabled, and upon submission, the device is assigned to the selected employee.

A screenshot of a computer

AI-generated content may be incorrect.

Fig. 9: Device Creation

This page allows admins to add new devices by entering details. Devices can also be added in bulk using the CSV upload feature at the top.

A screenshot of a computer

AI-generated content may be incorrect.

Fig. 10: Audit Log Form

This page lets admins search and filter audit logs by employee, device, and date. The results table displays detailed audit records of device activity such as status changes and the admin involved helping track historical actions.

A screenshot of a computer

AI-generated content may be incorrect.

Fig. 11: Employee Form

In case of any glitches or issues with Workday employee additions, this page provides an alternative way for admins to add employees directly. It allows entering details. Upon submission, the new employee is registered in the system.

**12. Software testing**

Software testing is an important part of development. It makes sure the Device Management System works properly and meets the requirements.

Testing checks that every feature, from user authentication to device allocation and reporting, works reliably in different situations.

The main goals of testing are to:

* Find and fix errors before deployment.
* Make sure the system performs well for different user roles.
* Verify that all modules work together to provide a secure and efficient user experience.

The testing process included unit testing, integration testing, system testing, and user acceptance testing (UAT) to confirm both functional and non-functional requirements.

**12.1 Test cases**

Test cases define the specific inputs, execution conditions, and expected results or each functionality.

The following tables present sample test cases for the Employee and Admin modules of the DMS.

**12.1.1 Test case for Employee**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID** | **Test Scenario** | **Expected Result** | **Status** |
| EMP-01 | Employee Login Validation | Employee is successfully logged in and redirected to dashboard. | Pass/Fail |
| EMP-02 | Invalid Login Handling | Error message “Invalid email or password” is displayed. | Pass/Fail |
| EMP-03 | View Assigned Devices | A list of devices allocated to the employee is displayed correctly. | Pass/Fail |
| EMP-04 | Profile Viewing | Employee details (name, email, department, allocated devices) are displayed accurately. | Pass/Fail |

**12.1.2 Test case for Admin**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID** | **Test Scenario** | **Expected Result** | **Status** |
| ADM-01 | Admin Login Validation | Admin is redirected to the Admin Dashboard. | Pass/Fail |
| ADM-02 | Add New Device | Device record is created and stored in the database. | Pass/Fail |
| ADM-03 | Allocate Device to Employee | Device status changes to **Allocated** and assignment record is updated. | Pass/Fail |
| ADM-04 | Delete/Deactivate Device | Device is removed/deactivated and no longer visible to users. | Pass/Fail |
| ADM-05 | Audit Log Review | Accurate logs of all user actions are displayed with timestamps. | Pass/Fail |

**13. CONCLUSION**

The Device Management System offers a strong and effective way to manage the entire lifecycle of IT assets in an organization. By combining employee and admin functions into one application, the system makes tasks like device allocation, maintenance tracking, and audit logging simpler, replacing manual records and scattered tools. Using the MERN stack (MongoDB, Express.js, React, Node.js) ensures a responsive, scalable, and secure environment that can handle real-time updates and multiple users. Its easy-to-use interface and organized database lower administrative work, improve data accuracy, and increase transparency between employees and administrators. Overall, this project shows how modern web technologies can address real asset management problems while providing a solid base for future growth and integration with other enterprise systems.

**14. LIMITATIONS**

While the system effectively manages organizational devices, it has some limitations. The current version always needs a stable internet connection, which restricts access in places with poor or unreliable connectivity. Its database and application structure work well for small to medium-sized organizations. Large businesses may need extra scaling methods, like distributed databases or load balancing, to support more users and transactions. Notifications are currently limited to email alerts, excluding other communication options like SMS, push notifications, or in-app messaging. Furthermore, the application is built for a single-organization setup, so it does not support multi-tenant deployments where different companies can use the same infrastructure while keeping their data separate. These limitations do not disrupt regular operations, but they show areas where further development could make the system easier to use and more scalable.

**15. FUTURE ENHANCEMENT**

The Device Management System is built for scalability and flexibility. It can continuously improve to meet the changing needs of organizations. Several key upgrades can be made in future versions to boost its functionality and value:

**1. Barcode Integration**

Adding barcode integration will greatly simplify tracking and managing devices. Each device can have a unique barcode or QR code for quick access to information like device details, allocation history, and maintenance records. This upgrade will lower manual data entry, reduce errors, and speed up operations during inventory checks, device assignments, or returns. Mobile barcode scanning will also enhance on-site verification and updates for staff and administrators.

**2. Cost Report Generation**

A detailed cost reporting module will give administrators valuable insights into device-related spending. This feature will create thorough reports on buying costs, maintenance fees, and depreciation values. It will help management assess overall spending and improve budget planning. By tracking costs across different departments or time frames, the system can aid in making savings decisions, forecast future spending, and ensure better financial oversight.

**3. CR (Change Request) Creation**

Including a CR (Change Request) feature will let employees and administrators formally suggest changes related to devices, like upgrades, configuration adjustments, or reassignment requests. Each CR will contain key details such as requester information, reason for change, and approval process. This enhancement will not only improve communication but also provide a structured and traceable way to implement changes, ensuring accountability and proper documentation for every update.

**16. REFERENCE**

**Technical Documentation & Framework References**

* PostgreSQL Documentation – <https://www.postgresql.org/docs>
* Flask Documentation – https://flask.palletsprojects.com
* Python Official Documentation – https://docs.python.org
* Axios (HTTP Client) – https://axios-http.com/docs/intro
* JSON Web Tokens (JWT) – https://jwt.io/introduction
* REST API Design Guidelines – https://restfulapi.net
* HTML5 Official Specification – https://developer.mozilla.org/en-US/docs/Web/Guide/HTML/HTML5
* CSS3 Specification – https://developer.mozilla.org/en-US/docs/Web/CSS
* JavaScript Guide – https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide

**Project Development & Tools**

* Visual Studio Code – https://code.visualstudio.com/docs
* Git Documentation – <https://git-scm.com/doc>
* Docker Documentation – https://docs.docker.com *(for deployment & containerization)*
* Nginx Documentation – https://nginx.org/en/docs *(if used for hosting)*

**17. APPENDIX**

**17. 1 USER MANUAL**

**Login**

* **Employees/Admins enter valid credentials to access their respective dashboards.**

**Admin Module**

* **Add New Device: Navigate to Device Management → Add Device → Enter details → Save.**
* **Allocate Device: Select a device → Allocate → Choose employee → Submit.**
* **Deactivate/Delete Device: Go to device list → Select → Delete/Deactivate → Confirm.**
* **Audit Logs: View all activity under Audit Logs.**

**Employee Module**

* **View Devices: Navigate to My Devices to check assigned devices.**
* **Request Maintenance: Select a device → Request Maintenance → Submit details.**
* **View Profile: Go to Profile to check personal and contact details.**

**Logout**

* **Click the Logout button to securely exit the system.**

**18. BIBLIOGRAPHY**