Industrial Internship Report on

”Vaccination Portal”

Prepared by

Arnav Dewan

Executive Summary
This report provides details of the Industrial Internship provided by Upskill Campus and The IoT Academy in collaboration with Industrial Partner UniConverge Technologies Pvt. Ltd. (UCT). The internship was focused on a project/problem statement provided by UCT and had to be completed within a duration of six weeks, including both development and documentation. My project involved developing a **full-stack Vaccination Portal** to manage user registrations, appointment bookings, vaccination status tracking, and secure authentication for different user types (adults and children). This internship gave me a valuable opportunity to gain real-world exposure to industrial challenges and design practical, working solutions using modern web development tools. It was an overall enriching experience and helped strengthen both my technical and problem-solving skills.

TABLE OF CONTENTS

1 Preface

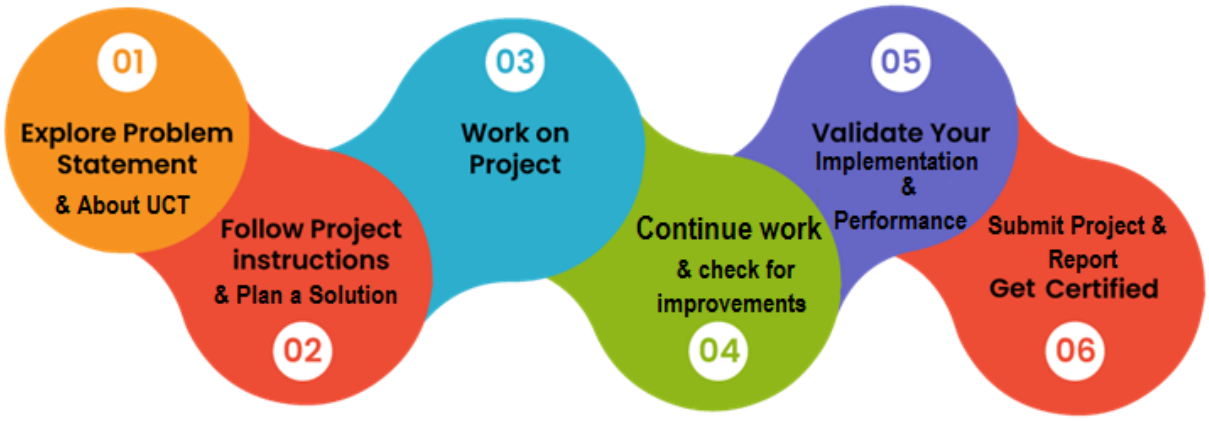
Summary of the whole 6 weeks' work.

About need of relevant Internship in career development.

Brief about Your project/problem statement.

Opportunity given by USC/UCT.

How Program was planned



Your Learnings and overall experience.

Thank to all (with names), who have helped you directly or indirectly.

Your message to your juniors and peers.

2 Introduction

2.1 About UniConverge Technologies Pvt Ltd

A company established in 2013 and working in Digital Transformation domain and providing Industrial solutions with prime focus on sustainability and RoI.

For developing its products and solutions it is leveraging various **Cutting Edge Technologies e.g. Internet of Things (IoT), Cyber Security, Cloud computing (AWS, Azure), Machine Learning, Communication Technologies (4G/5G/LoRaWAN), Java Full Stack, Python, Front end** etc.

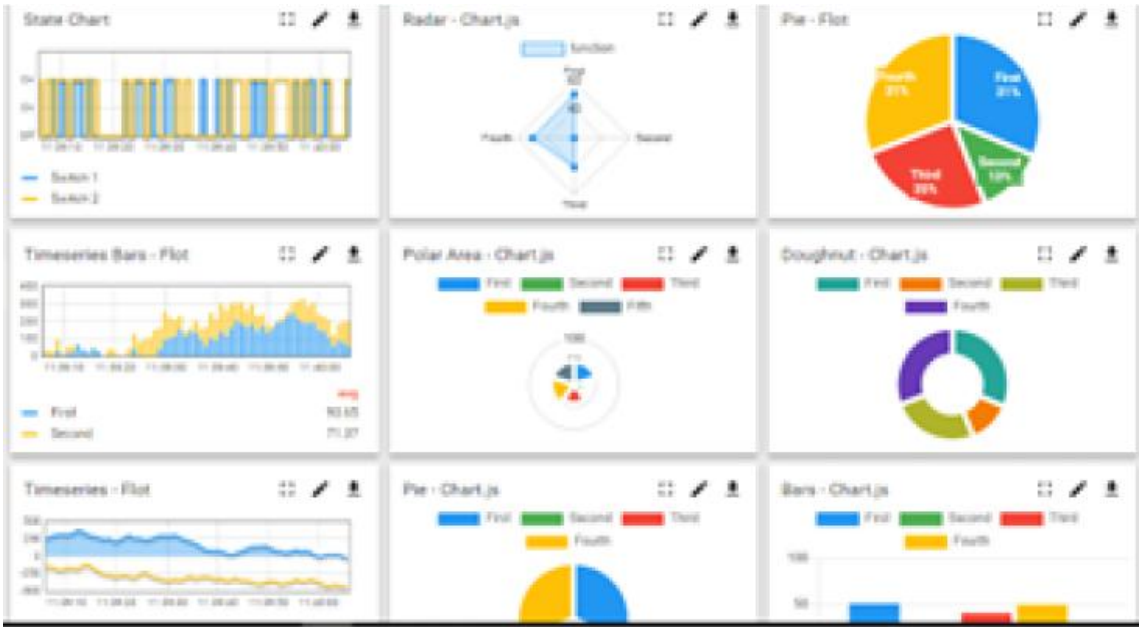
i. UCT IoT Platform (UCT Insight)

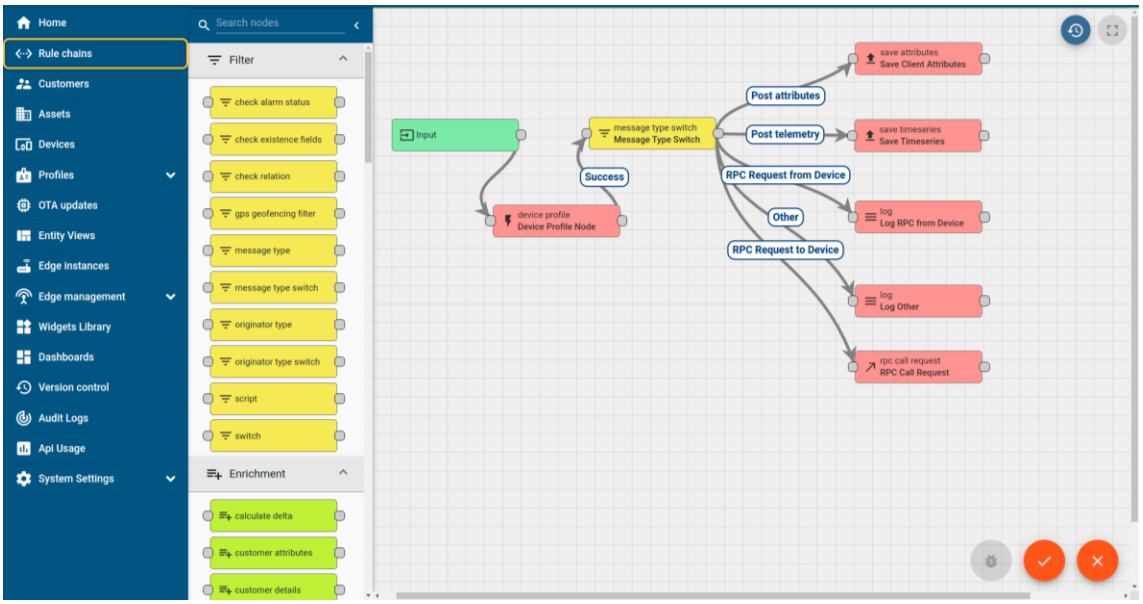
UCT Insight is an IOT platform designed for quick deployment of IOT applications on the same time providing valuable "insight" for your process/business. It has been built in Java for backend and ReactJS for Front end. It has support for MySQL and various NoSql Databases.

- It enables device connectivity via industry standard IoT protocols - MQTT, CoAP, HTTP, Modbus TCP, OPC UA
- It supports both cloud and on-premises deployments.

It has features to

- Build Your own dashboard
- Analytics and Reporting
- Alert and Notification
- Integration with third party application(Power BI, SAP, ERP)
- Rule Engine





ii. Smart Factory Platform (FACTORY WATCH)

Factory watch is a platform for smart factory needs.

It provides Users/ Factory

- with a scalable solution for their Production and asset monitoring
- OEE and predictive maintenance solution scaling up to digital twin for your assets.
- to unleased the true potential of the data that their machines are generating and helps to identify the KPIs and also improve them.
- A modular architecture that allows users to choose the service that they what to start and then can scale to more complex solutions as per their demands.

Its unique SaaS model helps users to save time, cost and money.

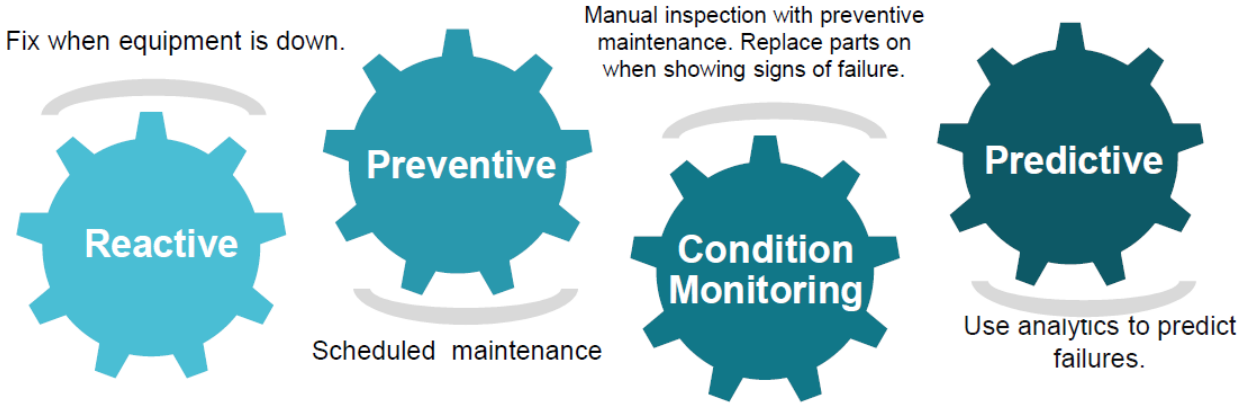
Machine	Operator	Work Order ID	Job ID	Job Performance	Job Progress		Output		Rejection	Time (mins)				Job Status	End Customer
					Start Time	End Time	Planned	Actual		Setup	Pred	Downtime	Idle		
CNC_S7_81	Operator 1	WO0405200001	4168	58%	10:30 AM		55	41	0	80	215	0	45	In Progress	i
CNC_S7_81	Operator 1	WO0405200001	4168	58%	10:30 AM		55	41	0	80	215	0	45	In Progress	i

iii. LoRaWAN based Solution

UCT is one of the early adopters of LoRAWAN teschnology and providing solution in Agritech, Smart cities, Industrial Monitoring, Smart Street Light, Smart Water/ Gas/ Electricity metering solutions etc.

iv. Predictive Maintenance

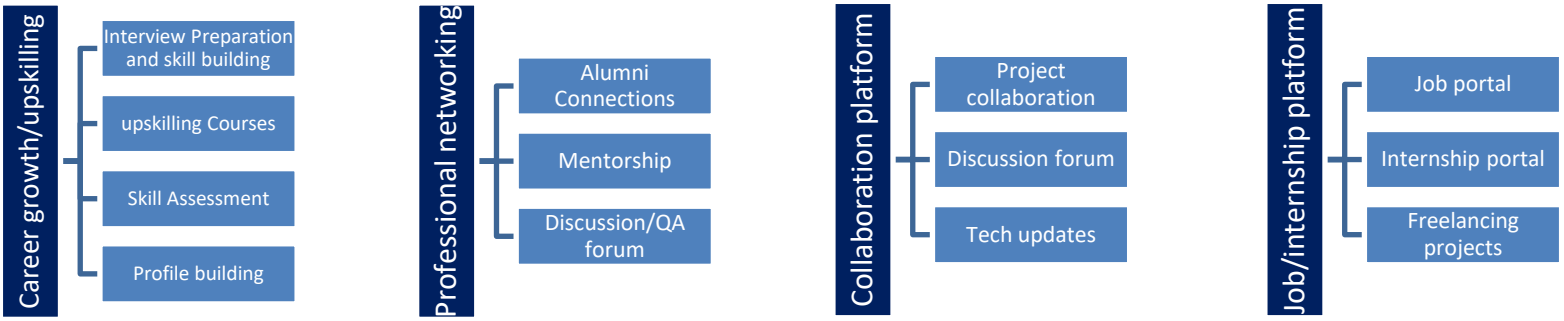
UCT is providing Industrial Machine health monitoring and Predictive maintenance solution leveraging Embedded system, Industrial IoT and Machine Learning Technologies by finding Remaining useful life time of various Machines used in production process.



2.2 About upskill Campus (USC)

upskill Campus along with The IoT Academy and in association with Uniconverge technologies has facilitated the smooth execution of the complete internship process.

USC is a career development platform that delivers **personalized executive coaching** in a more affordable, scalable and measurable way.


Career growth/upskilling
Interview Preparation and skill building
upskilling Courses
Skill Assessment
Profile building
Professional networking
Alumni Connections
Mentorship
Discussion/QA forum
Collaboration platform
Project collaboration
Discussion forum
Tech updates
Job/internship platform
Job portal
Internship portal
Freelancing projects

2.3 The IoT Academy

The IoT academy is EdTech Division of UCT that is running long executive certification programs in collaboration with EICT Academy, IITK, IITR and IITG in multiple domains.

2.4 Objectives of this Internship program

The objective for this internship program was to

- ☛ get practical experience of working in the industry.
- ☛ to solve real world problems.
- ☛ to have improved job prospects.
- ☛ to have Improved understanding of our field and its applications.
- ☛ to have Personal growth like better communication and problem solving.

2.5 Reference

[1] MDN Web Docs. (n.d.). HTML, CSS, and JavaScript Documentation. Retrieved from: https://developer.mozilla.org/

[2] Express.js. (n.d.). Fast, unopinionated, minimalist web framework for Node.js. Retrieved from: https://expressjs.com/

[3] MongoDB, Inc. (n.d.). MongoDB Documentation – The Developer Data Platform. Retrieved from: https://www.mongodb.com/docs/

2.6 Glossary

Term	Acronym	Explanation (in Project Context)
Application Programming Interface	API	Defines how the frontend (React or HTML/JS) communicates with backend services to manage users, appointments, and vaccination data.
Create, Read, Update, Delete	CRUD	Represents the four basic operations used to manage appointment records, user data, and vaccination statuses.
JSON Web Token	JWT	Used for secure user authentication, ensuring only logged-in users can book or manage appointments.
JavaScript Object Notation	JSON	A lightweight data format used to exchange user info, appointment details, and status updates between frontend and backend.
User Interface / User Experience	UI/UX	The frontend design that enables users to register, book appointments, view confirmation, and navigate easily.
Not Only SQL	NoSQL	Refers to MongoDB, which stores user profiles, appointment records, and vaccination status using a flexible document-based structure.

3 Problem Statement

In the assigned problem statement

- The objective of the assigned project was to design and develop a full-stack Vaccination Portal that streamlines the process of booking, managing, and tracking vaccination appointments through a web-based platform.

- The system aimed to allow users to register/login, book appointments for adult or child vaccination, view appointment status, and track vaccination records.

- The platform was designed to ensure secure authentication and session management, protecting user credentials and preventing unauthorized access.

- Real-time data storage and retrieval were required to reflect user bookings, appointment changes, and vaccination status updates instantly and accurately.

- The frontend was developed using HTML, CSS, JavaScript, and React.js, focusing on a clean, responsive, and user-friendly interface with features like animated taglines, dark mode, and a real-time clock.

- The backend used Node.js and Express.js to handle server-side logic, API routes, and communication between frontend and database.

- MongoDB, a NoSQL database, was implemented to store user profiles, appointment details, and vaccination status in a flexible and secure manner.

- The system followed RESTful API design principles to ensure smooth and modular communication between the frontend and backend components.

- Security features such as password hashing and JWT (JSON Web Tokens) were implemented to protect sensitive user data and ensure authorized access.

- Overall, the project involved integrating multiple technologies to build a robust, scalable, and secure vaccination management platform that simulates real-world public health systems effectively.

4 Existing and Proposed solution

- The project aimed to build a **full-stack Vaccination Portal** for managing essential vaccination-related operations through a centralized web platform.
- Users could **register, log in, book vaccination appointments** for themselves or dependents (adults/children), **update or cancel bookings**, and **track their vaccination status** securely.
- The **frontend** was developed using **React.js** (alongside HTML, CSS, and JavaScript), while the **backend** was built using **Node.js**, **Express.js**, and **MongoDB** for database management.
- Key features included **user authentication**, **secure data handling**, **appointment scheduling**, and **real-time updates** of user records and appointment status.
- The goal was to **simulate a real-world vaccination management system** with a clean interface, responsive UI, and secure backend, making the user experience seamless and medically informative.

4.1 Code submission (Github link)

https://github.com/arnavdewan/UPSKILLCAMPUS.git

4.2 **Report submission (Github link) :** first make placeholder, copy the link.

https://github.com/arnavdewan/UPSKILLCAMPUS/blob/main/VaccinationPortal_arnav_USC_UCT%20.pdf

5 Proposed Design/ Model

- The design flow began with **requirement analysis** and **system planning** based on the vaccination appointment use case. The aim was to build a system that manages registrations, bookings, and vaccination tracking efficiently.
- **Frontend development** focused on creating an intuitive and responsive user interface that allows users to register, log in, book appointments for adults or children, manage existing bookings, and track vaccination status.
- **Backend development** was responsible for implementing RESTful APIs, handling user authentication and authorization, managing session security, and connecting to the MongoDB database for storing user and appointment data.
- After frontend and backend components were developed, **integration** was performed to connect the two layers seamlessly using API calls and data exchange in JSON format.
- The integrated system underwent **thorough testing** to validate all features — including form validation, booking logic, authentication, and secure access — ensuring **functionality**, **data integrity**, and **security** across all modules.

5.1 High Level Diagram (if applicable)

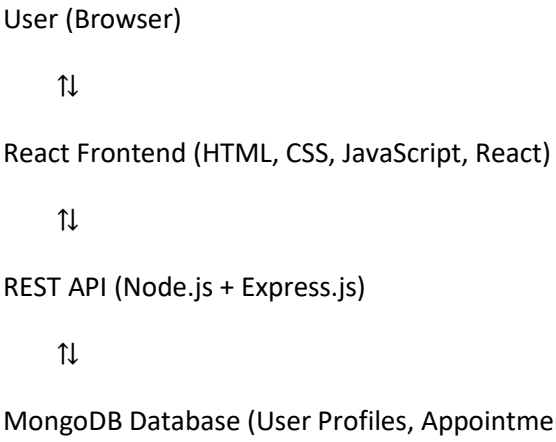


Figure 1: HIGH LEVEL DIAGRAM OF THE SYSTEM

5.2 Low Level Diagram (if applicable)

User initiates appointment booking

↓

Frontend validates input (e.g., date, user type)

↓

API call to Appointment Controller (/api/appointments)

↓

Backend checks slot availability in MongoDB

↓

If available, appointment is saved and status is updated

↓

Confirmation response sent back to frontend

↓

UI updates: show confirmation, appointment details, and booking status

5.3 Interfaces (if applicable)

Update with Block Diagrams, Data flow, protocols, FLOW Charts, State Machines, Memory Buffer Management.

6 Performance Test

1. Constraints Identified

- **Response Time:** The system must respond quickly to key actions such as login, appointment booking, and status updates to provide a smooth user experience.
- **Scalability:** The application should handle multiple users registering and booking appointments simultaneously without degrading in performance.
- **Data Consistency:** Accurate tracking of appointment slots and vaccination statuses is critical to prevent overbooking or conflicting records.
- **Security:** User credentials and health-related data must be securely stored and transmitted to ensure privacy and regulatory compliance.
- **Resource Usage:** Efficient use of backend memory, database operations, and API calls ensures the application remains lightweight, stable, and cost-effective.

2. Design Considerations to Address Constraints

- **RESTful APIs** with optimized MongoDB queries and optional pagination were used to deliver fast response times for user and appointment operations.
- **JWT-based authentication** was implemented to manage user sessions securely and to reduce repeated database lookups for validation.
- **Atomic MongoDB operations** were used to ensure consistency when booking appointments and updating user vaccination status.
- **Password hashing** (using bcrypt) and HTTPS recommendations were adopted to secure sensitive data such as login credentials and health records.
- The **backend was built with asynchronous APIs** and **connection pooling** to handle concurrent user requests and maintain responsiveness under load.

3. Test Results

- **Response times** for primary operations (login, registration, appointment booking, status check) averaged under **500 milliseconds** under moderate simulated user load.

- The system maintained **data consistency** even when multiple users performed appointment-related operations simultaneously.
- No **significant memory leaks** or **CPU spikes** were observed during multi-user simulation and stress testing.

4. Recommendations & Impact

- For **higher user volumes**, future improvements such as **load balancing** and **MongoDB indexing** (on commonly queried fields like user ID, date) are recommended for performance scaling.
- Integrating **real-time monitoring tools** like **PM2**, **New Relic**, or **Grafana** in production would assist in tracking server health, latency, and uptime.
- Introducing **caching mechanisms** (e.g., using **Redis**) for frequently accessed data such as vaccine type info or FAQ content can further optimize speed and reduce load on the database.

6.1 Test Plan/ Test Cases

To ensure the system meets both functional and non-functional requirements, the following test cases were created and executed during development and validation:

Test Case Description	Expected Outcome	Status
User Registration	User signs up with valid input	Account is created successfully
User Login	Valid credentials submitted	User is authenticated and redirected to dashboard
Book Appointment	User books an appointment with available slot	Appointment confirmed, entry saved in database
Update Appointment	User updates appointment details (e.g., time/date)	Appointment record is updated correctly
Cancel Appointment	User cancels a scheduled appointment	Appointment is removed and slot is released

Test Case Description	Expected Outcome	Status
Track Vaccination Status	User checks vaccination status	Accurate status displayed based on appointment history
Form Validation (Frontend)	User submits form with incomplete or invalid data	Error message shown; submission prevented
Session Handling (JWT)	Logged-in user accesses protected routes	Access granted; invalid token results in logout

6.2 Test Procedure

1. Environment Setup:

- **Backend Server:** Started using nodemon server.js from the backend folder
- **Frontend Application:** Launched using npm start (for the React-based frontend or by opening static HTML files in the browser)
- **Database Connection:** MongoDB connected either locally using the default URI or remotely using **MongoDB Atlas**

---

2. Testing Tools Used:

- **Manual Testing:** Conducted through the browser UI for user interactions
- **Postman:** Used to test REST API endpoints (e.g., registration, login, appointment booking)
- **Chrome Developer Tools:** Utilized for analyzing network activity, inspecting response times, and debugging frontend errors

---

3. Test Execution Procedure:

- Verified **form validation** on the frontend for:
    - Registration

  - Login
  - Appointment booking and updates

- Tested backend **API endpoints** such as:
  - POST /api/auth/register
  - POST /api/auth/login
  - POST /api/appointments/book
  - PUT /api/appointments/update
  - DELETE /api/appointments/cancel
- Simulated **multiple users** performing appointment operations (booking, updating, canceling) concurrently to test system consistency
- Monitored:
  - Backend **console logs** for API activity and error messages
  - **MongoDB** to validate data persistence and integrity
- Analyzed **response times** and network activity using Chrome Developer Tools under normal load conditions to assess performance

6.3 Performance Outcome

- All core operations including user registration, login, appointment booking, updating, and status tracking were executed within 300–500 milliseconds under standard load conditions.
- The system successfully handled simultaneous actions by multiple users (e.g., booking, updating, and cancelling appointments) without any data inconsistencies or conflicts.
- No critical issues were encountered during testing — such as UI glitches, server crashes, or memory leaks — ensuring stable runtime behaviour.
- Overall, the application demonstrated:
- A smooth user experience
- Accurate data handling through secure API calls
- Reliable performance suitable for both educational use and as a real-world prototype

7 My learnings

- Gained **hands-on experience in full-stack development**, understanding how the frontend, backend, and database work together to build a complete, real-world **Vaccination Management System**.
- Developed proficiency in **React.js**, HTML, CSS, and JavaScript for building **responsive, user-friendly interfaces** that enable users to register, book appointments, and track vaccination status.
- Strengthened **backend development skills** using **Node.js and Express.js**, learning how to design and implement **secure, scalable, and RESTful APIs** to handle authentication, bookings, and user data.
- Gained a solid understanding of **NoSQL database design** with **MongoDB**, including schema modeling and efficient queries for storing appointment records, user profiles, and vaccination statuses.
- Learned and applied **security best practices**, including **password hashing**, **JWT-based authentication**, and **secure data communication** between client and server.
- Improved **problem-solving skills** by addressing real-world challenges such as **form validation**, **data consistency**, **multi-user concurrency**, and **API integration errors**.
- Experienced the complete **software development lifecycle** — from **requirement analysis** and **system design** to **implementation**, **testing**, and **deployment readiness**.
- Enhanced **team collaboration and communication skills** through regular mentor interactions, peer discussions, and structured code reviews.
- Understood the importance of **performance testing** and **optimization techniques** to ensure the system remains responsive and reliable under real-world usage.
- This internship greatly contributed to strengthening my **technical foundation** and boosted my confidence to independently develop **scalable full-stack web applications**, preparing me for future roles in software development

8 Future work scope

The current Vaccination Portal provides a strong foundation for managing vaccination-related processes; however, there are several opportunities for enhancement and scalability to meet evolving user expectations and public health requirements.

- **Multi-Factor Authentication (MFA):** Implementing MFA would significantly improve security by adding an extra layer of protection to user accounts. Future versions could also explore **biometric login options** such as fingerprint or facial recognition for added convenience.
- **Real-Time Notifications:** Integrating **email or SMS alerts** can help users stay informed about upcoming appointments, cancellations, status updates, or important health advisories — enhancing engagement and reliability.
- **Scalability and Load Handling:** To support a growing user base, the backend infrastructure can be enhanced using **load balancing** and **horizontal scaling**, ensuring stable performance under peak usage conditions.
- **Advanced Analytics:** Introducing **admin dashboards** with analytics on user activity, appointment trends, and vaccine stock levels can provide health administrators with valuable insights for decision-making.
- **Integration with External APIs:** Connecting with **official government vaccination APIs**, hospital systems, or vaccine inventory services could help streamline appointment availability and record verification.
- **Mobile Application Development:** Building a **dedicated mobile app** (Android/iOS) would make the portal more accessible, enabling users to book and manage appointments on the go.
- **Role-Based Access Control (RBAC):** Implementing RBAC would allow for **differentiated access** for users (patients), healthcare workers, and administrators, each with custom dashboards and permissions.
- **Ongoing Monitoring & Compliance:** As the portal scales for wider institutional or public deployment, **continuous performance monitoring**, **security auditing**, and adherence to **data privacy regulations** (e.g., HIPAA, GDPR) will be essential.

Overall, these future enhancements can evolve the Vaccination Portal into a **comprehensive, secure, and scalable health management system**, suitable for institutional, governmental, or public healthcare deployment.