Industrial Internship Report on

”Library Management System”

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). This internship was focused on a project/problem statement provided by UCT. We had to finish the project including the report in 6 weeks’ time. My project was developing a full-stack Library Management System to manage book inventories, user borrowing records, and secure user access. This internship gave me a very good opportunity to get exposure to Industrial problems and design/ implement solution for that. It was an overall great experience to have this internship

TABLE OF CONTENTS

1 Preface

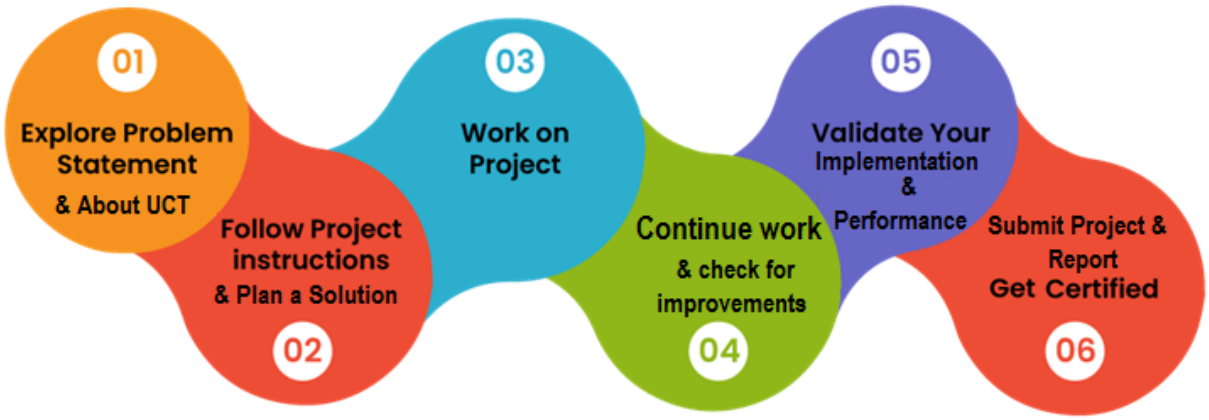
Summary of the whole 6 weeks' work.

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)

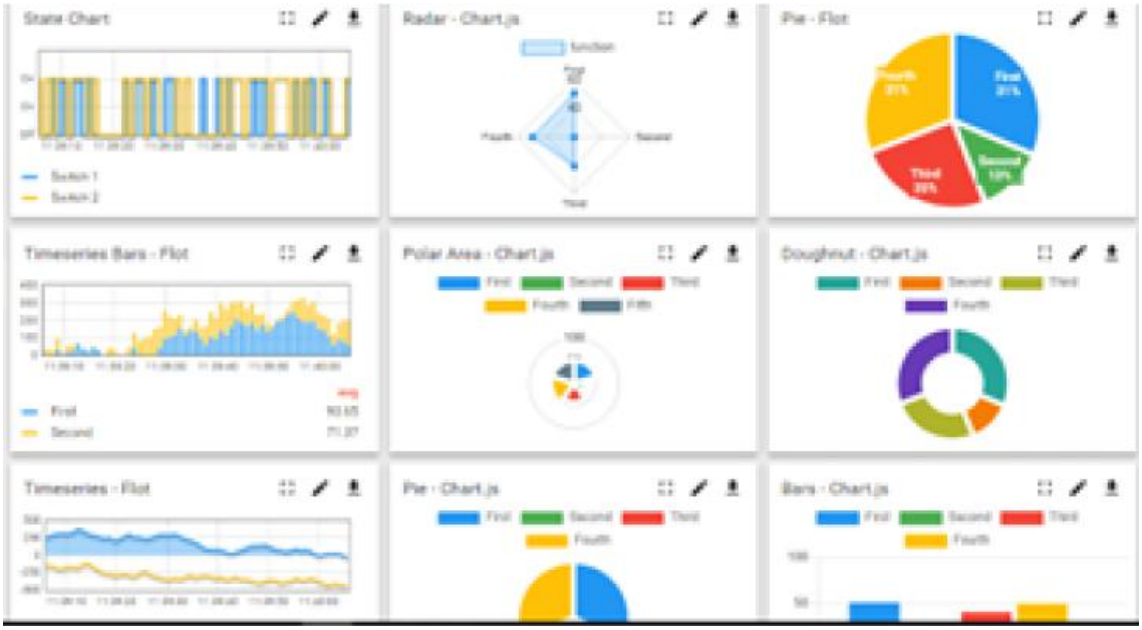
UCT Insight is an IOT platform designed for quick deployment of IOT applications on the same time providing valuable "insight" for your process/business. It has been built in Java for backend and ReactJS for Front end. It has support for MySQL and various NoSql Databases.

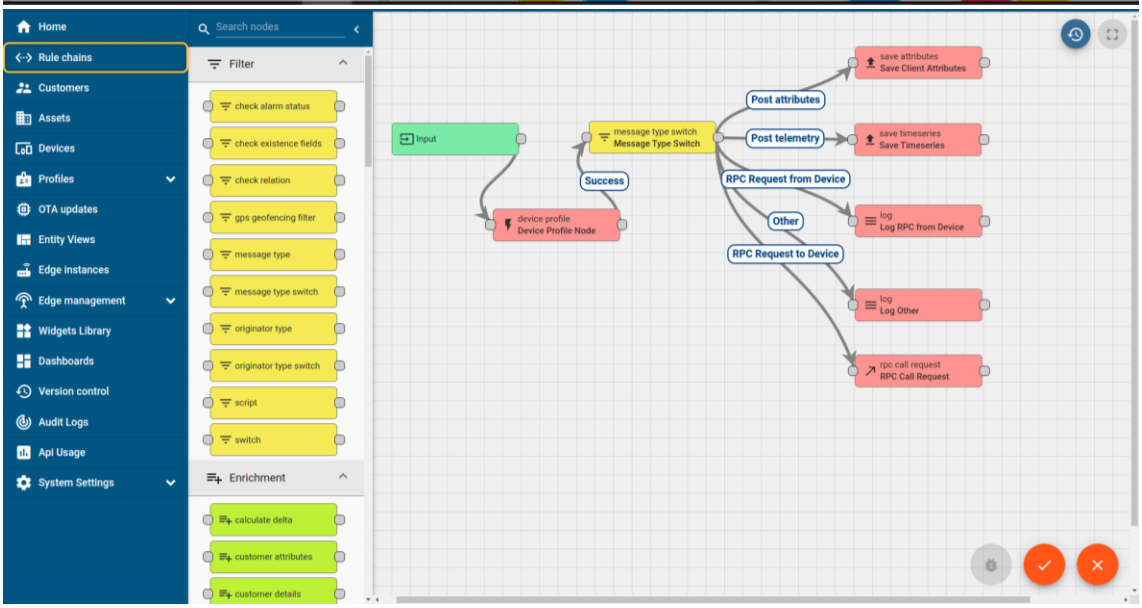
- It enables device connectivity via industry standard IoT protocols - MQTT, CoAP, HTTP, Modbus TCP, OPC UA

- 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


State Chart
Radar - Chart.js
Pie - Plot
Timeseries Bars - Plot
Polar Area - Chart.js
Doughnut - Chart.js
Timeseries - Plot
Pie - Chart.js
Bars - Chart.js


Home
Rule chains
Customers
Assets
Devices
Profiles
OTA updates
Entity Views
Edge instances
Edge management
Widgets Library
Dashboards
Version control
Audit Logs
Api Usage
System Settings
Search nodes
Filter
check alarm status
check existence fields
check relation
gps geofencing filter
message type
message type switch
originator type
originator type switch
script
switch
Enrichment
calculate delta
customer attributes
customer details
Input
message type switch
Message Type Switch
device profile
Device Profile Node
Success
Post attributes
Post telemetry
RPC Request from Device
Other
RPC Request to Device
save attributes
Save Client Attributes
save timeseries
Save Timeseries
log
Log RPC from Device
log
Log Other
rpc call request
RPC Call Request

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.

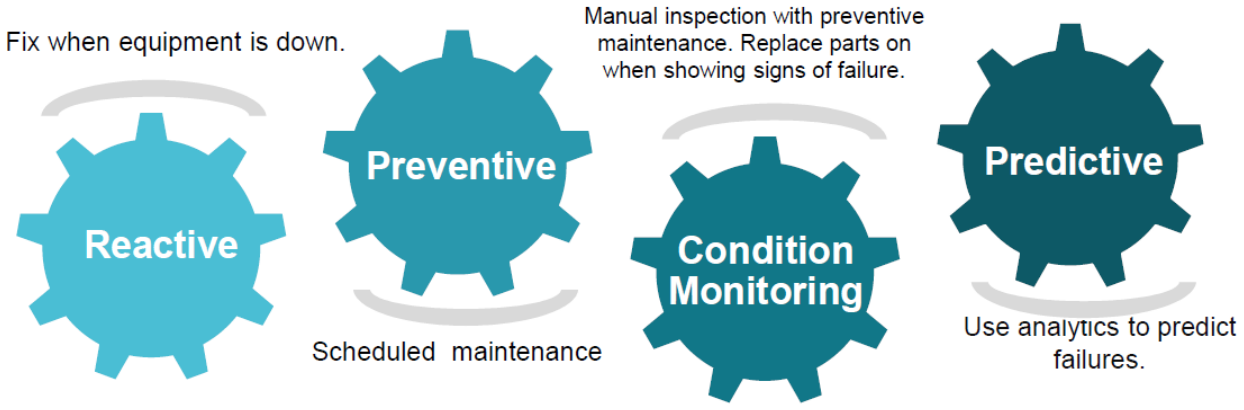
Machine	Operator	Work Order ID	Job ID	Job Performance	Job Progress		Output		Rejection	Time (mins)				Job Status	End Customer
					Start Time	End Time	Planned	Actual		Setup	Pred	Downtime	Idle		
CNC_S7_81	Operator 1	WO0405200001	4168	58%	10:30 AM		55	41	0	80	215	0	45	In Progress	i
CNC_S7_81	Operator 1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

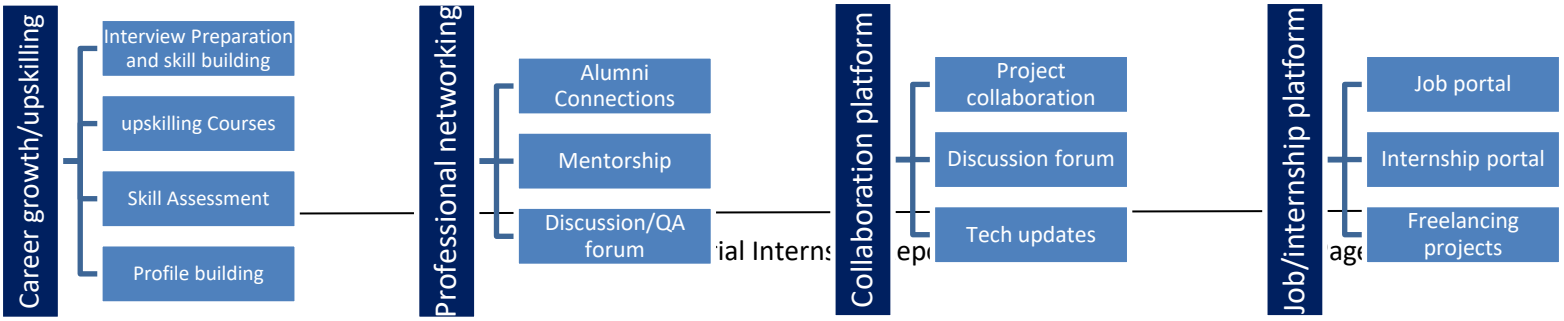
upskill Campus along with The IoT Academy and in association with Uniconverge technologies has facilitated the smooth execution of the complete internship process.

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

Seeing need of upskilling in self paced manner along-with additional support services e.g. Internship, projects, interaction with Industry experts, Career growth Services

upSkill Campus aiming to upskill 1 million learners in next 5 year

https://www.upskillcampus.com/



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

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

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

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

2.6 Glossary

Terms & Acronyms (Library Management System Context)

Term	Acronym	Explanation (in Project Context)
Application Programming Interface	API	Defines how the frontend (React) communicates with backend services to manage books and users.
Create, Read, Update, Delete	CRUD	Represents the four basic operations used to manage book records, user data, and borrowing logs.
JSON Web Token	JWT	Used for secure user authentication, ensuring only logged-in users can borrow or return books.
JavaScript Object Notation	JSON	A data format used to transfer information (e.g., book info, user login) between frontend and backend.
User Interface / User Experience	UI/UX	The frontend design that allows users to search books, view availability, and manage their account easily.
Not Only SQL	NoSQL	Refers to MongoDB, which stores book data, user profiles, and borrowing records in a flexible structure.

3 Problem Statement

In the assigned problem statement

- The objective of the assigned project was to design and develop a **full stack Library Management System** that simulates basic library operations through a web-based platform.
- The system aimed to allow users to **browse available books, borrow or return books**, and **view borrowing history**.
- The platform needed to ensure **secure authentication and session management**, protecting user credentials and preventing unauthorized access.
- **Real-time data storage and retrieval** were required to reflect book availability and user activity instantly and accurately.
- The frontend was developed using **HTML, CSS, JavaScript, and React.js**, focusing on a clean, responsive, and user-friendly interface.
- The backend used **Node.js and Express.js** to handle server-side logic and route management.
- **MongoDB**, a NoSQL database, was implemented to store **user profiles, book details, and borrowing records** efficiently and securely.
- The system followed **RESTful API design** principles for smooth communication between frontend and backend.
- Security measures like **password hashing** and **JWT (JSON Web Tokens)** were used to ensure data privacy.
- Overall, the project required integrating multiple technologies to build a **robust, scalable, and secure library management web application** that mimics real-world library functionality.

4 Existing and Proposed solution

The project aimed to build a **full stack Library Management System** for managing essential library operations.

- Users could **register, log in, browse available books, borrow or return books, and track their borrowing history** securely.
- The frontend was built using **React**, while the backend used **Node.js, Express**, and **MongoDB**.
- Key features included **user authentication, secure data handling**, and **real-time updates** of book availability.
- The goal was to simulate real-world library processes in a **user-friendly and secure web application**.

4.1 Code submission (Github link)

https://github.com/arnavdewan

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

https://github.com/arnavdewan/LibraryManager

5 Proposed Design/ Model

The design flow starts with **requirement analysis and system planning**. Frontend development focuses on building an intuitive **user interface for browsing, borrowing, and managing books**, while backend development handles **API creation, authentication logic, and database connections**.

Integration connects the frontend and backend systems, followed by thorough testing to ensure **functionality, data integrity, and security**. Finally, the complete application is **deployed**, delivering a **secure, scalable, and user-friendly library system** for practical use.

5.1 High Level Diagram (if applicable)

6 4.1 High Level Diagram (if applicable)

User (Browser) ⇄ React Frontend ⇄ REST API (Node.js + Express) ⇄ MongoDB Database

Figure 1: HIGH LEVEL DIAGRAM OF THE SYSTEM

6.1 Low Level Diagram (if applicable)

7 Low Level Diagram (if applicable)

User initiates borrow request → Frontend validates input → API call to Borrow Controller → Book availability checked and updated in MongoDB → Confirmation sent back to frontend → UI updates borrowing history and book status.

8 Performance Test

In developing the **Library Management System**, several key constraints were identified to ensure the project meets real-world industry standards beyond a basic academic solution:

- **1. Constraints Identified:**
- **Response Time:** The system must respond quickly to actions such as login, book search, borrow/return operations to maintain a smooth user experience.
- **Scalability:** The application should handle multiple users borrowing or returning books simultaneously without performance degradation.
- **Data Consistency:** Library operations require accurate book availability tracking and borrowing records to prevent conflicts or data mismatches.
- **Security:** Protecting user credentials and borrow history during transmission and storage is essential.
- **Resource Usage:** Efficient use of server memory and processing power ensures the application remains cost-effective and stable.

- **2. Design Considerations to Address Constraints:**
- Utilized **RESTful APIs** with optimized database queries and pagination for efficient data loading and faster response times.
- Implemented **JWT-based authentication** to manage sessions securely and reduce redundant database calls.
- Used **MongoDB update operations** and atomic checks to maintain book availability and borrowing record integrity.
- Applied **password hashing** (e.g., bcrypt) and recommended **HTTPS** for secure communication and data protection.

- Designed backend with **asynchronous APIs** and **connection pooling** to support concurrent operations and maintain responsiveness.

- **3. Test Results:**
- **Response times** for key operations (book search, borrow/return, login) averaged **under 500 milliseconds** under moderate load.
- The system maintained **data consistency** for borrowing records, even with multiple users interacting in parallel during test scenarios.
- No significant memory leaks or CPU spikes were observed during simulated multi-user testing.

- **4. Recommendations & Impact:**
- Under higher user loads, further improvements like **load balancing** and **MongoDB indexing** are recommended for performance scaling.
- Integration of **real-time monitoring tools** (e.g., PM2, New Relic) in production would help track latency and server health.
- Implementing **caching mechanisms** (e.g., Redis) for frequent queries like book search can improve speed and reduce database load.

8.1 Test Plan/ Test Cases

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

Test Case	Description	Expected Outcome	Status
User Registration	User signs up with valid input	Account is created successfully	✅ Passed

Test Case	Description	Expected Outcome	Status
User Login	Valid credentials submitted	User is authenticated and redirected to home	✅ Passed
Book Search	Search for books by title/author	Matching books are displayed	✅ Passed
Borrow Book	User initiates book borrow with available book	Book status updated, entry logged	✅ Passed
Return Book	User returns previously borrowed book	Book status restored to available	✅ Passed
View Borrowing History	View past borrow/return transactions	All records correctly listed	✅ Passed

8.2 Test Procedure

Environment Setup:

- Backend server started using nodemon server.js
- Frontend started with npm start in the client folder
- MongoDB connected locally using default URI or MongoDB Atlas

Testing Tools Used:

- Manual testing through UI in browser
- Postman used for API endpoint testing (CRUD operations)
- Chrome Developer Tools used to inspect response times and errors

Procedure:

- Verified form validations for registration, login, and book search

- Tested API endpoints (e.g., POST /api/borrow, GET /api/books)
- Simulated multiple users borrowing and returning books concurrently
- Monitored console and database to validate correctness and consistency
- Analyzed response time using browser dev tools under normal load

8.3 Performance Outcome

- All core operations including login, book search, borrow, and return executed **within 300–500 milliseconds** under standard load.

- System handled **simultaneous actions** by multiple users (borrow and return) without data inconsistencies.

- **No critical issues** were detected such as UI breakdowns, server crashes, or memory leaks during testing.

- The application demonstrated a **smooth user experience**, accurate data handling, and solid performance for educational and real-world prototype use.

9 My learnings

- Gained **hands-on experience in full stack development**, understanding how the frontend, backend, and database integrate to build a complete web-based **library management application**.

- Developed proficiency in **React.js** for building responsive, user-friendly interfaces to manage book listings, borrowing, and user activities.

- Strengthened backend development skills using **Node.js and Express**, learning how to create secure, scalable, and RESTful APIs for library operations.

- Worked with **MongoDB**, gaining a solid understanding of **NoSQL database design**, schema modeling, and efficient data queries for storing book records, user profiles, and borrow logs.

- Learned and applied **security best practices**, including **password hashing**, **JWT-based authentication**, and **secure data communication** between frontend and backend.

- Improved **problem-solving skills** by debugging real-world challenges such as data inconsistency, user concurrency conflicts, and API integration issues.

- Experienced the **complete software development lifecycle** — from requirement gathering, system design, and development, to testing and deployment.

- Enhanced collaboration and communication through **regular mentor feedback**, **peer discussions**, and **structured project reviews**.

- Understood the importance of **performance testing and optimization** to ensure the application performs well under realistic usage conditions.

- This internship helped **solidify my technical foundation** and gave me the confidence to independently develop scalable full stack applications — preparing me for real-world software development roles.

Future work scope

The current Library Management System provides a solid foundation, but there are multiple opportunities for enhancement and expansion to align with evolving technological trends and user expectations.

- **Implementing multi-factor authentication (MFA)** can significantly enhance the system's security, safeguarding user accounts from unauthorized access. Future updates could also include **biometric login options** like fingerprint or facial recognition for improved convenience.
- Adding **real-time notifications via email or SMS** can keep users informed about borrowing deadlines, return reminders, overdue books, and system updates—boosting user engagement and accountability.
- To accommodate a larger user base, the system can be improved with **load balancing** and **horizontal scaling** techniques to ensure high performance and availability under peak usage.
- Introducing **advanced analytics and reporting** features would help librarians and administrators gain insights into user activity, book popularity, and borrowing trends.
- **Integration with third-party APIs** such as ISBN databases or digital libraries could enhance book information and enable features like book previews or online reading links.
- Developing a **mobile application version** of the platform can increase accessibility, allowing users to search, borrow, and manage books on the go.
- Implementing **role-based access control** can distinguish permissions between students, faculty, and library staff, offering customized views and controls.
- Continuous **performance monitoring, security testing**, and **data privacy compliance** will be critical as the system scales for institutional or public deployment.

Overall, these future enhancements will transform the system into a comprehensive, secure, and user-friendly **digital library solution**, well-suited for real-world institutional and academic environments.