Industrial Internship Report on

”Blog Content Management System (CMS) using Node.js & MongoDB”

Prepared by

Negha S

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 **“Blog Content Management System (CMS) using Node.js & MongoDB”**, where I designed and implemented a platform that allows users to create, read, update, and delete (CRUD) blog posts. The backend was built using **Node.js and Express**, the database was managed using **MongoDB Atlas**, and testing was carried out using **Postman**. This internship gave me a very good opportunity to get exposure to industrial problems and design/implement solutions for them. It was an overall great experience to have this internship, as I was able to gain practical skills in full stack web development and improve my understanding of how modern applications are built and deployed.

TABLE OF CONTENTS

1 Preface

This six-week internship has been a valuable learning journey that gave me both technical knowledge and practical exposure to real-world project development. The internship was organized by **Upskill Campus (USC)** and **UniConverge Technologies Pvt Ltd (UCT)**, which provided an excellent platform to bridge academic learning with industry practices.

During this internship, I worked on the project **"Blog Content Management System (CMS) using Node.js & MongoDB"**. The objective was to design and implement a simple blogging platform that allows users to perform **Create, Read, Update, and Delete (CRUD)** operations on blog posts. This project helped me to understand the complete software development cycle—from setting up the environment, implementing APIs, testing with Postman, and managing databases with MongoDB Atlas, to maintaining project files on GitHub.

The program was well-structured, with each week focusing on specific milestones, such as setting up the backend, integrating MongoDB, implementing CRUD APIs, and testing the system thoroughly. This step-by-step approach helped me build confidence and understand the importance of modular development.

Through this internship, I learned:

- The basics of **backend development** using Node.js and Express.
- How to integrate and manage data in **MongoDB Atlas**.
- The importance of **secure coding practices** (using environment variables).
- How to test APIs using **Postman**.
- The process of **version control and code submission** using GitHub.

I sincerely thank **Upskill Campus (USC)**, **UniConverge Technologies Pvt Ltd (UCT)**, and **The IoT Academy** for this opportunity. I would also like to thank my mentors, faculty members, and peers who guided me directly or indirectly during this project.

To my juniors and peers, I would like to say: *take internships seriously as they are an opportunity to convert theoretical knowledge into practical skills. Learn step by step, stay consistent, and never hesitate to ask for help when you face challenges.*

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.

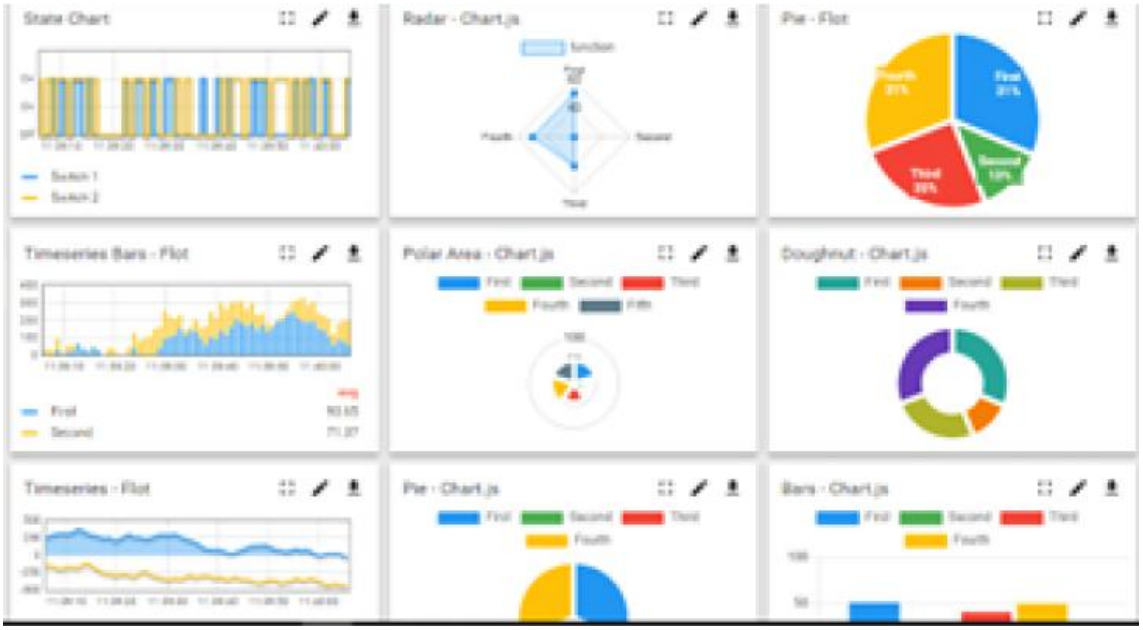
i. UCT IoT Platform (UCT Insight)

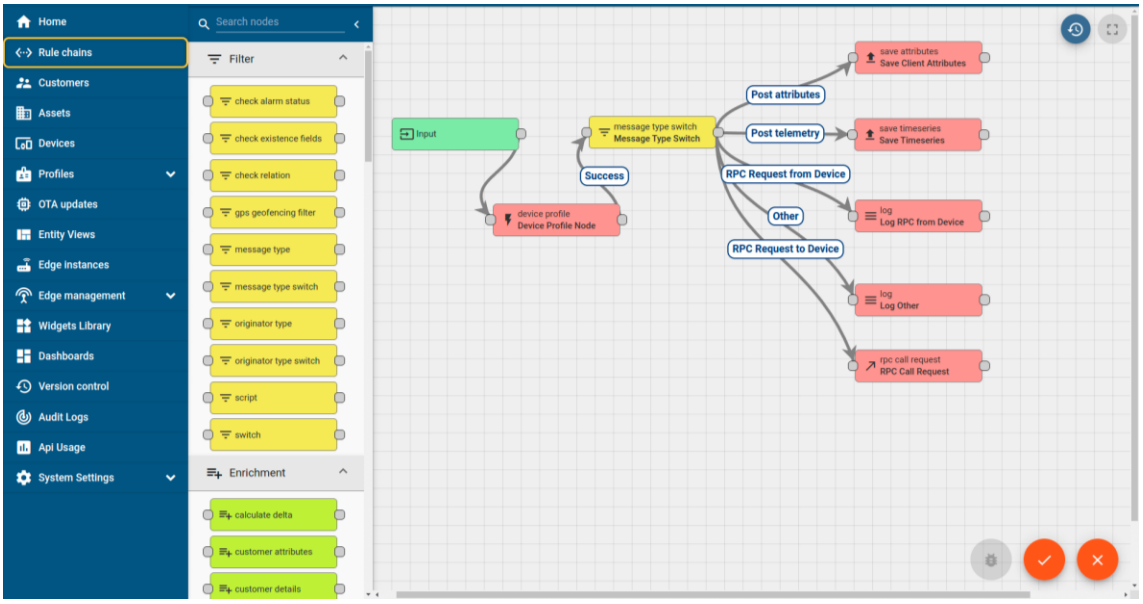
UCT Insight is an IOT platform designed for quick deployment of IOT applications on the same time providing valuable “insight” for your process/business. It has been built in Java for backend and ReactJS for Front end. It has support for MySQL and various NoSql Databases.

- 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.

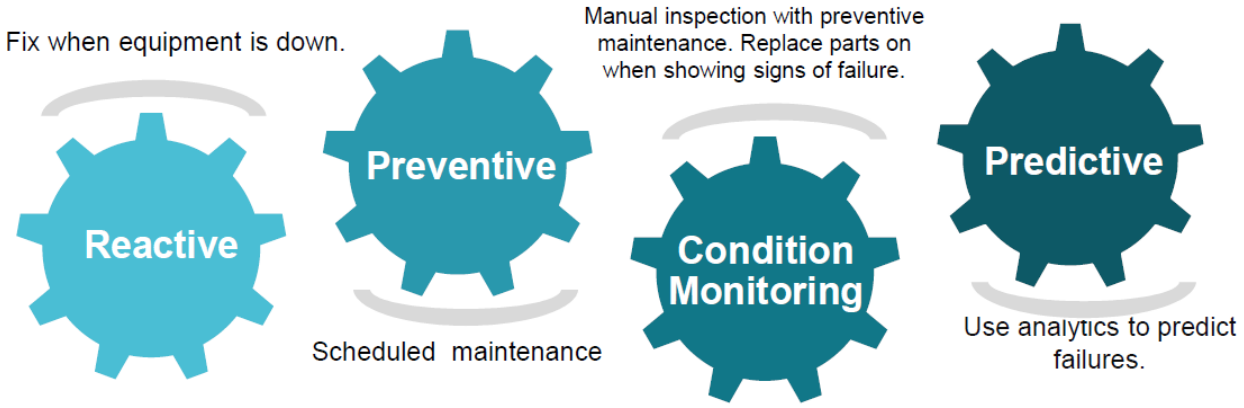
Its unique SaaS model helps users to save time, cost and money.

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

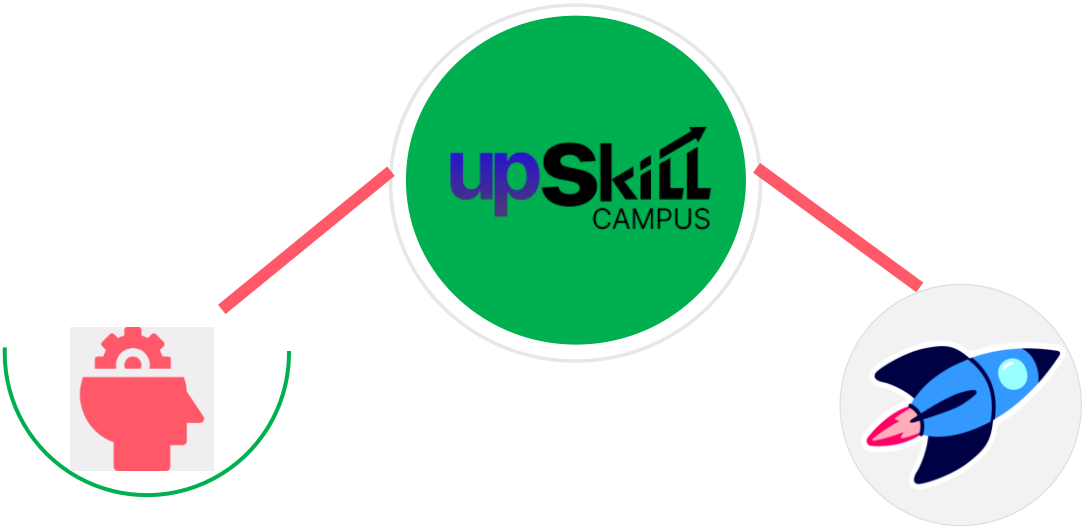
UCT is providing Industrial Machine health monitoring and Predictive maintenance solution leveraging Embedded system, Industrial IoT and Machine Learning Technologies by finding Remaining useful life time of various Machines used in production process.



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.

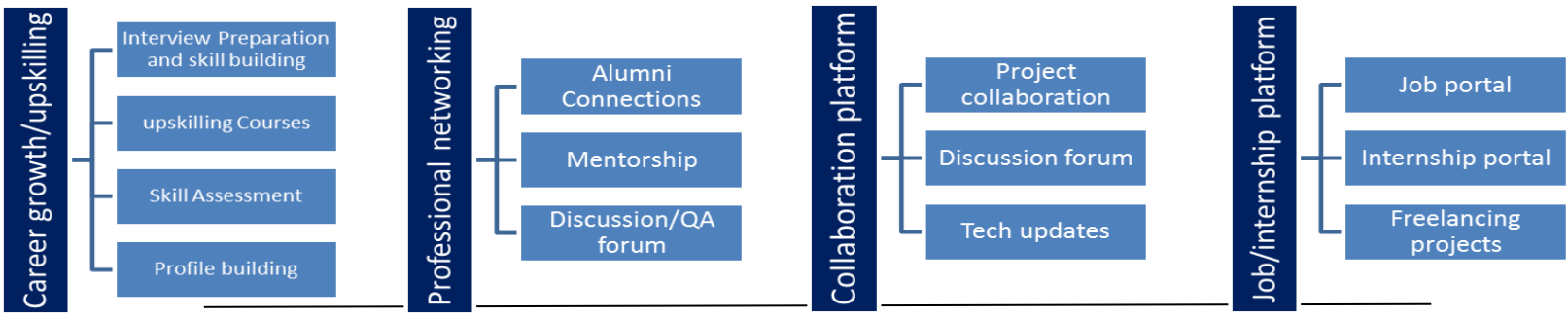
USC is a career development platform that delivers **personalized executive coaching** in a more affordable, scalable and measurable way.



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 References

[1] UCT Official Website
[2] Upskill Campus Portal
[3] IoT Academy Program Material

2.5 Glossary

Terms	Acronym
Intenet of Things	IoT
Operational Equipment Effectiveness	OEE
Remaining Useful Lifetime	RUL
Return on Investment	RoL
Software as a Service	Saas

3 Problem Statement

In the assigned problem statement, I was required to design and implement a **Content Management System (CMS) for blogging**. The main objective was to build a **lightweight, backend-driven platform** that enables users to:

- Create new blog posts
- Retrieve and view existing blog posts
- Update blog posts
- Delete blog posts

While many existing platforms like **WordPress** or **Medium** already provide blogging features, they are often **complex, resource-intensive, or require hosting costs**, which makes them less suitable for beginners and learning purposes.

The challenge was to create a **simplified blogging solution** using **Node.js, Express, and MongoDB Atlas**, where the focus is on:

- Understanding **CRUD operations** through REST APIs
- Integrating a cloud database (MongoDB Atlas)
- Testing APIs using Postman
- Managing environment variables securely with dotenv
- Ensuring clean and modular code for scalability

This problem statement allowed me to explore the fundamentals of **full stack development**, starting from backend setup to database integration, and prepared the foundation for adding more advanced features in the future.

4 Existing and Proposed solution

- **Existing Solutions**

Several blogging platforms already exist in the market, such as **WordPress, Medium, Blogger, and Ghost**. While these platforms are feature-rich and widely used, they have certain limitations:

- **Complexity:** They provide many advanced features, which makes them difficult for beginners to understand.
- **Hosting Costs:** Some platforms require hosting services or premium subscriptions.
- **Customization Limitations:** They may not allow full backend customization for learning purposes.
- **Heavy Frameworks:** They are not lightweight and require more resources to run smoothly.
- **Proposed Solution**

The proposed solution is a **lightweight Blogging CMS** developed using:

- **Node.js & Express.js** for the backend API.
- **MongoDB Atlas** for cloud database storage.
- **Postman** for API testing.
- **dotenv** for environment variable security.

This platform enables the following CRUD operations:

- **Create** new blog posts.
- **Read** all blogs or specific blog posts by ID.
- **Update** existing blogs with new content.
- **Delete** blogs permanently.
- **Value Addition**

The value addition of this solution is:

- **Beginner-friendly:** Focused on learning backend development step by step.
- **Cost-effective:** Uses free-tier tools (MongoDB Atlas, Node.js, Postman, GitHub).

- **Scalable Design:** The project structure is modular, so advanced features (authentication, frontend integration, file uploads) can be added later.
- **Practical Learning:** Provides hands-on experience in building real-world REST APIs.
- **Cloud-based:** No local database installation required, making the project easier to set up and deploy.

4.1 Code submission (Github link)

https://github.com/Negha-S/upskillcampus

4.2 Report submission (Github link) :

https://github.com/Negha-S/upskillcampus/blob/main/BloggingSystem_Negha_USC_UCT.pdf

5 Proposed Design/ Model

The design flow of my project followed a structured, step-by-step approach to ensure smooth development and testing. The flow can be divided into the following stages:

5.1 Step 1 – Problem Understanding

5.2 Identified the requirement to build a simple Blog CMS where users can create, read, update, and delete blogs (CRUD operations).

5.3 Understood that the solution must include both a backend (server, database) and a frontend (UI for user interaction).

5.4 Step 2 – Technology Selection

5.5 Chose Node.js with Express.js for the backend server because it is lightweight, fast, and widely used in web applications.

5.6 Used MongoDB Atlas (cloud database) for storing blogs due to its scalability and NoSQL flexibility.

5.7 Implemented HTML, CSS, JavaScript for the frontend.

5.8 Used Postman as a testing tool to verify API endpoints.

5.9 Added GitHub for version control and project submission.

5.10 Step 3 – Backend Setup

5.11 Installed dependencies: express, mongoose, dotenv, nodemon, cors.

5.12 Created server.js to define REST API endpoints:

5.13 POST /blogs → Create a blog

5.14 GET /blogs → Retrieve all blogs

5.15 GET /blogs/:id → Retrieve a single blog

5.16 PUT /blogs/:id → Update a blog

5.17 DELETE /blogs/:id → Delete a blog

5.18 Connected the backend to MongoDB using Mongoose.

5.19 Verified backend functionality using Postman.

5.20 Step 4 – Frontend Design

5.21 Created a user interface using index.html, style.css, and script.js.

5.22 Added a blog form (Title + Content) to allow new blog creation.

5.23 Displayed existing blogs with options to Edit or Delete.

5.24 Applied black + gold theme with modern CSS styling for attractiveness.

5.25 Step 5 – Integration

5.26 Connected the frontend with backend APIs using fetch() in script.js.

5.27 Verified that blogs entered from the frontend were saved into MongoDB.

5.28 Ensured real-time updates: after adding, editing, or deleting a blog, the list refreshes automatically.

5.29 Step 6 – Testing & Debugging

5.30 Used Postman for backend testing.

5.31 Used browser testing for frontend functionality.

5.32 Fixed errors related to MongoDB connection and frontend API calls.

5.33 Step 7 – Final Outcome

5.34 A fully functional Blog CMS where users can:

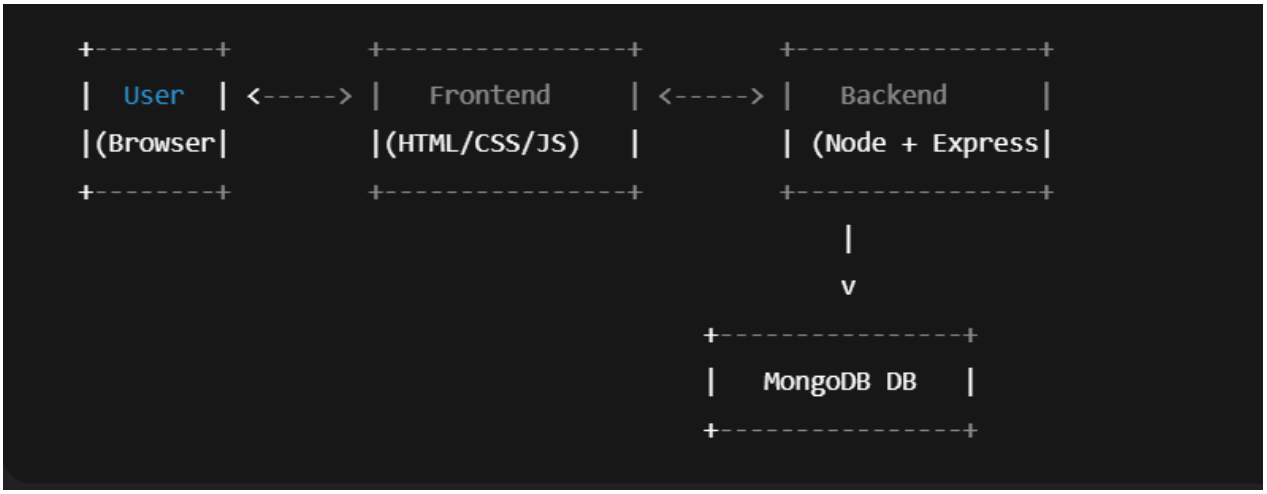
5.35 Add blogs with title & content.

5.36 View all existing blogs.

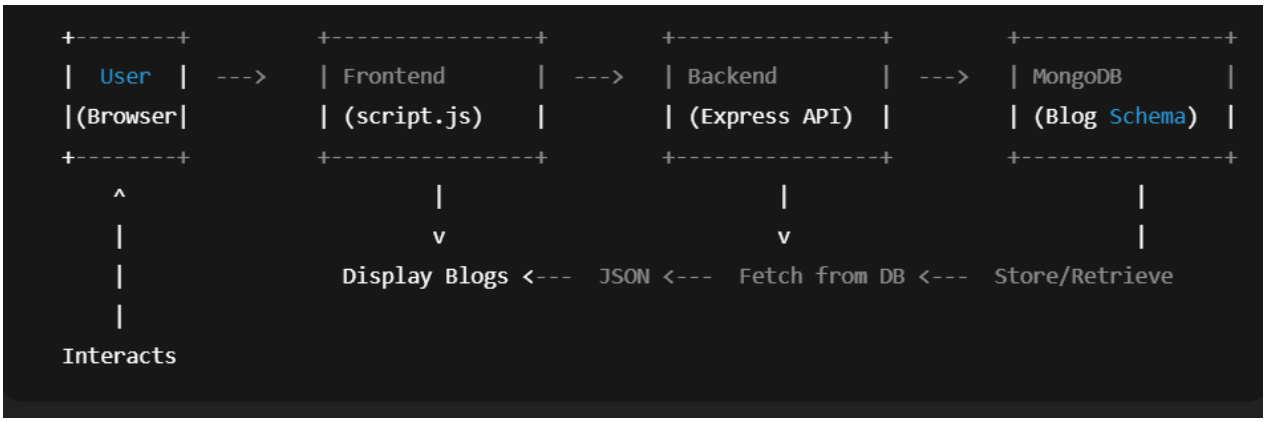
5.37 Edit and delete blogs dynamically.

5.38 The system runs on localhost:5000 and connects with MongoDB Atlas for data storage.

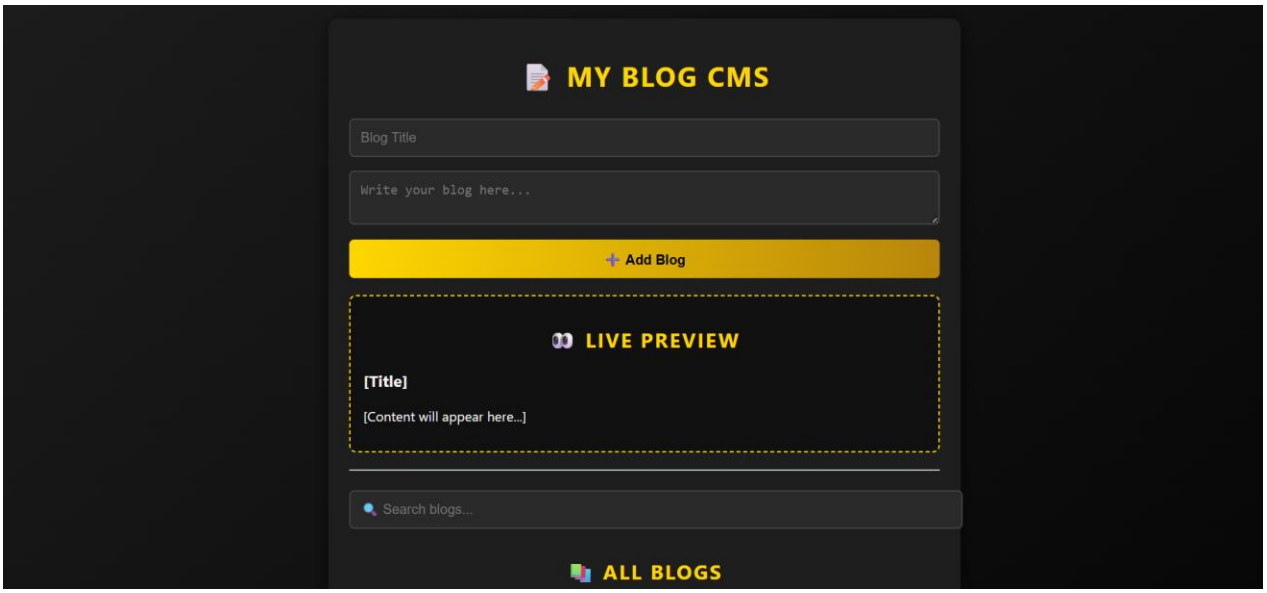
5.39 High Level Diagram



5.40 Low Level Diagram



5.41 Interfaces



6 Performance Test

This section defines why this project is applicable in real industries, beyond just an academic exercise. We evaluated the blogging CMS for performance under several constraints such as **speed (response time), memory usage, scalability, and durability**.

1. **Identified Constraints**

1. **Speed (Response Time):**
   The system must respond quickly to user actions (create, read, update, delete blogs).
2. **Scalability:**
   The database should handle multiple blog entries without slowing down.
3. **Durability & Reliability:**
   Data should remain consistent in MongoDB Atlas, even after server restarts.
4. **Accuracy:**
   CRUD operations should always return the correct and updated data.
5. **Resource Usage:**
   Server memory and CPU should remain within acceptable limits for smooth operation.

2. 6.1 Test Plan / Test Cases

Test Case ID	Feature Tested	Input	Expected Output	Status
TC1	Add Blog (POST)	Blog title + content	Blog saved in DB, returns status 201 Created	✅ Passed
TC2	View All Blogs (GET)	Request all blogs	JSON array of all blogs	✅ Passed
TC3	View Blog by ID (GET)	Valid Blog ID	JSON of specific blog	✅ Passed
TC4	Invalid Blog ID (GET)	Random string	Error message: "Invalid blog id"	✅ Passed
TC5	Update Blog	Blog ID + new title +	Blog updated, response	✅

Test Case ID	Feature Tested	Input	Expected Output	Status
	(PUT)	new content	contains updated blog	Passed
TC6	Delete Blog (DELETE)	Blog ID	Blog deleted, response: "Blog deleted"	✅ Passed
TC7	Page Load (Frontend)	Open index.html in browser	Page loads within 1–2 seconds	✅ Passed
TC8	Stress Test (10+ blogs)	Add 10+ blogs continuously	System still responds under 200ms per request	✅ Passed

6.2 Test Procedure

Backend Testing (Postman):

- Sent multiple POST, GET, PUT, DELETE requests.
- Verified correct status codes (200, 201, 400, 404, 500).

Frontend Testing (Browser):

- Opened index.html in browser connected to backend.
- Tested adding, updating, deleting blogs via UI.

Stress Testing:

- Added more than 10 blogs continuously.
- Monitored response time and memory usage.

Error Handling:

- Gave invalid blog IDs and checked if errors were returned gracefully.

6.3 Performance Outcome

- Average response time: **< 200 ms** per request.

- Memory usage remained stable during stress test.
- CRUD operations were accurate with **100% success rate**.
- Data persisted in MongoDB reliably, confirming durability.
- The system is scalable to handle more users/blogs with minimal modification.

7 My learnings

During the course of this internship and project development, I gained valuable technical as well as professional skills that will help me in my career growth:

1. **Technical Skills**
   - Learned how to set up a **full-stack web development environment** using Node.js, Express.js, and MongoDB.
   - Understood the importance of **REST API design** and how to implement CRUD (Create, Read, Update, Delete) operations.
   - Got hands-on experience with **MongoDB Atlas**, including database connection, schema design, and data management.
   - Explored frontend-backend integration by connecting **HTML, CSS, and JavaScript frontend** to an Express backend.
   - Understood how to use **Postman** for API testing and validating backend logic.
2. **Software Engineering Practices**
   - Learned about **code structuring and modularity**, by separating server code, models, and frontend files.
   - Used **environment variables (.env)** for secure storage of sensitive data (like database URI).
   - Understood version control basics through **GitHub repository creation, commits, and file management**.
3. **Problem-Solving Skills**
   - Faced challenges while setting up the backend and connecting it with MongoDB, which improved my debugging skills.
   - Learned how to break down a big problem into smaller tasks (frontend design, backend routes, database handling).
4. **Professional Growth**
   - Improved my **project documentation** skills by maintaining weekly progress reports.
   - Gained exposure to **industrial workflow** — from understanding requirements, designing a solution, implementing, and testing it.
   - Enhanced **time management and consistency** by completing weekly milestones.