Industrial Internship Report on

”Content Management System for a blog”

Prepared by

Akash A Navale

Executive Summary

This report provides details of the Industrial Internship conducted by UpSkill Campus and The IoT Academy in collaboration with the industrial partner UniConverge Technologies Pvt. Ltd. (UCT). The internship was based on a project/problem statement provided by UCT, and we were required to complete the project along with the report within a duration of 6 weeks.

My project was **Content Management System (CMS) for a Blog**, where I developed a drag-and-drop website builder that allows users to design web pages by placing text or media components into predefined placeholders. The system also includes a blog-publishing feature with a built-in text editor that converts user-written content into HTML and stores it in a database. The website is deployed over HTTP/HTTPS, serving blog posts dynamically from the database based on the page template designed by the user.

This internship provided me with an excellent opportunity to gain exposure to real industrial problems and to design and implement practical solutions for them. Overall, it was a highly valuable and insightful experience.

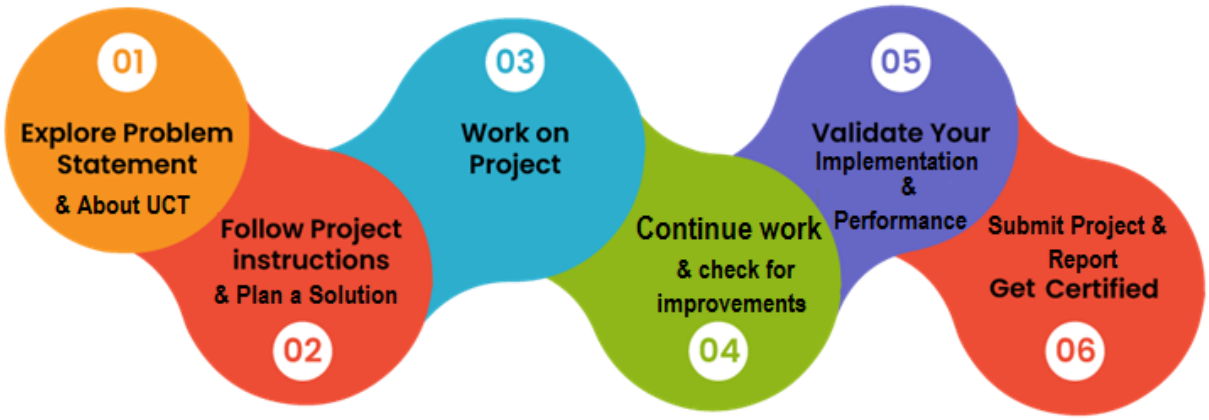
TABLE OF CONTENTS

1 Preface

This internship spanned six weeks and focused on developing a lightweight educational CMS for blogging that demonstrates the intern's ability to design and implement a full-stack web application with real-world features. The project was chosen to provide exposure to backend programming using Java Servlets and JDBC, database design with MySQL, and frontend development using HTML, CSS, and JavaScript.

Throughout the internship, I learned how different components of a web application work together — from user interaction on the client side to data storage and retrieval on the server side. The project helped me understand important aspects such as session handling, database connectivity, content management, file uploads, and data validation. It also provided practical knowledge about security considerations like SQL injection prevention, secure login handling, and user authentication.

Developing this project also enhanced my understanding of how a CMS works internally, giving me hands-on experience in building a simplified version of systems like WordPress. Overall, this internship experience improved my technical skills, problem-solving abilities, and confidence in working with real-time full-stack projects.



The work plan included requirement gathering, architecture and DB design, API and frontend implementation, testing (functional & security), and final packaging with documentation and diagrams. I would like to thank my mentors at UCT and the coordinators at upskill Campus for guidance.

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.

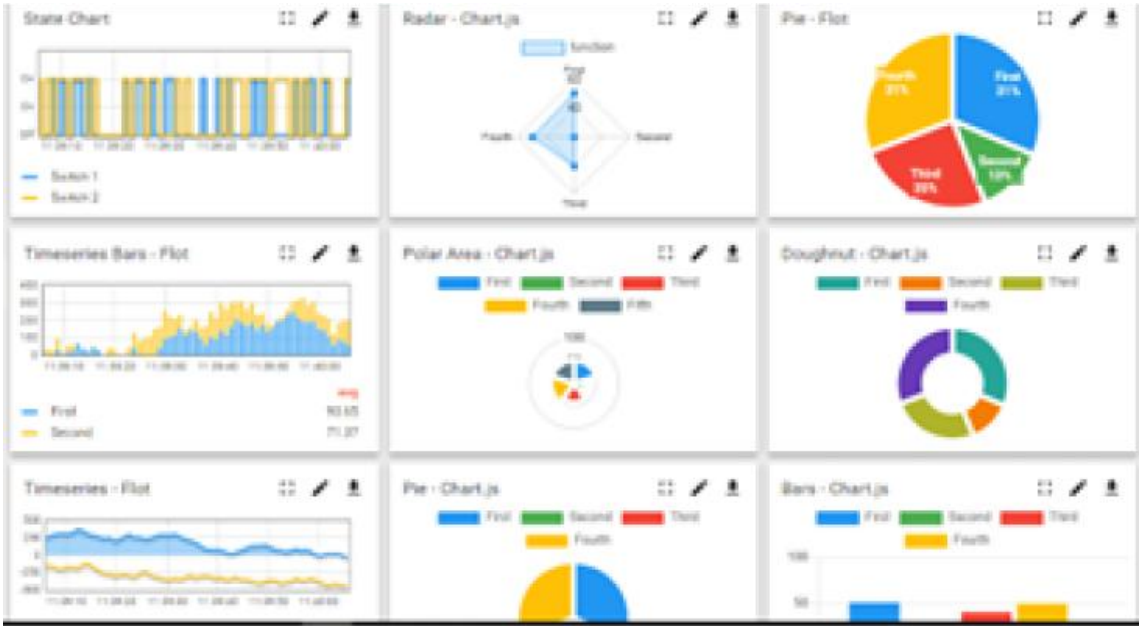
i. UCT IoT Platform (UCT Insight)

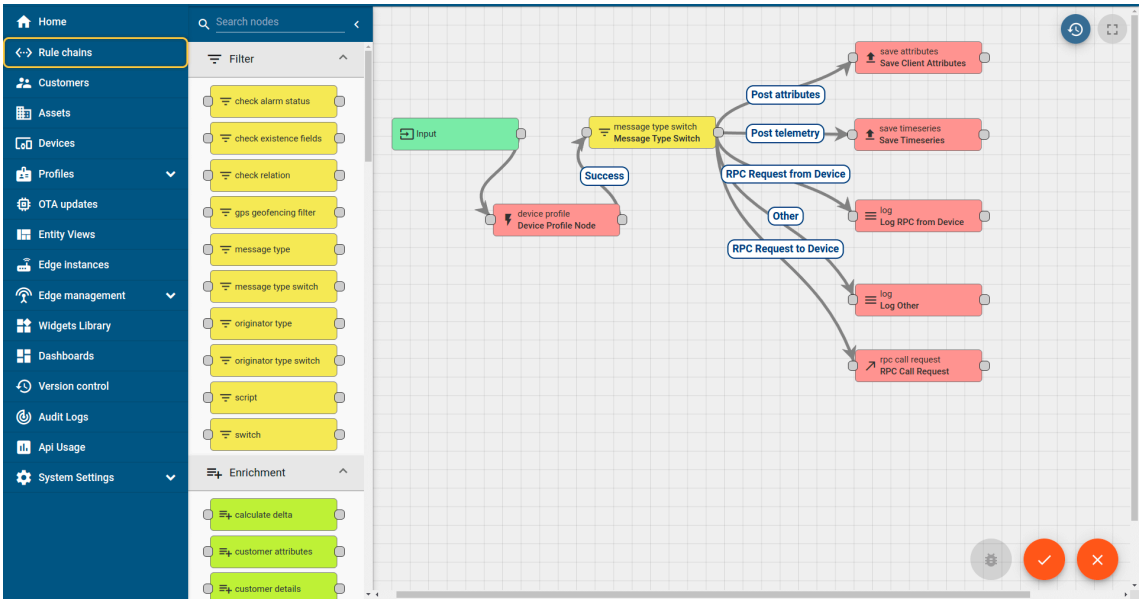
UCT Insight is an IOT platform designed for quick deployment of IOT applications on the same time providing valuable “insight” for your process/business. It has been built in Java for backend and ReactJS for Front end. It has support for MySQL and various NoSql Databases.

- 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.

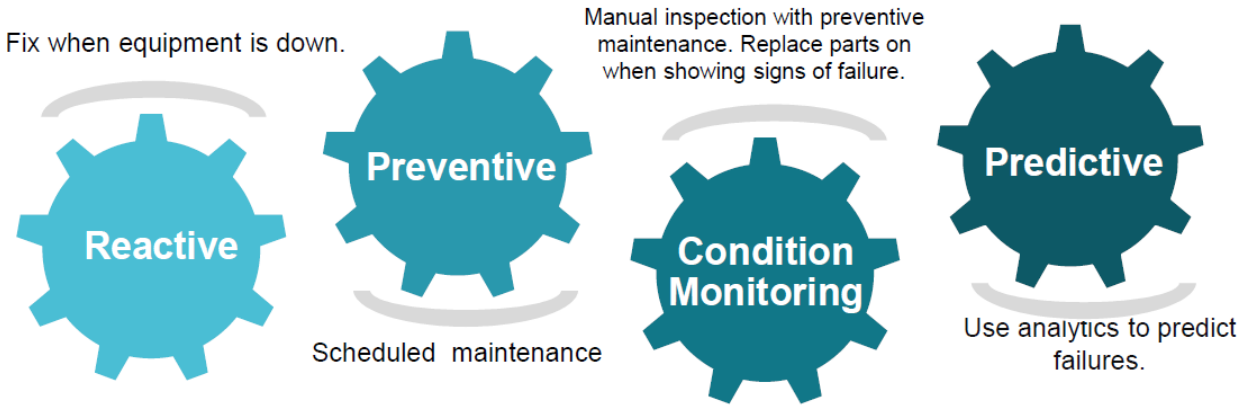
Machine	Operator	Work Order ID	Job ID	Job Performance	Job Progress		Output		Rejection	Time (mins)				Job Status	End Customer
					Start Time	End Time	Planned	Actual		Setup	Pred	Downtime	Idle		
CNC_S7_81	Operator 1	WO0405200001	4168	58%	10:30 AM		55	41	0	80	215	0	45	In Progress	i
CNC_S7_81	Operator 1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

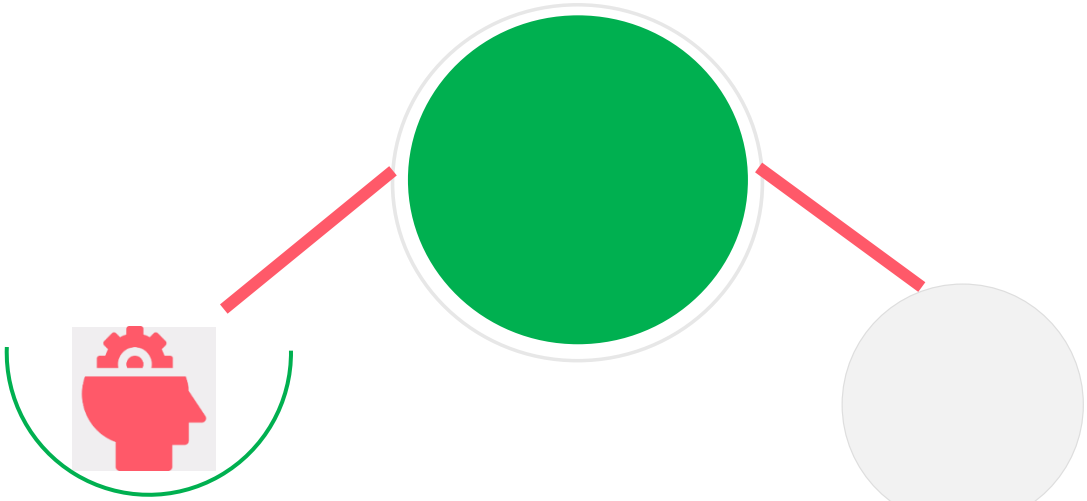
UCT is providing Industrial Machine health monitoring and Predictive maintenance solution leveraging Embedded system, Industrial IoT and Machine Learning Technologies by finding Remaining useful life time of various Machines used in production process.



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.

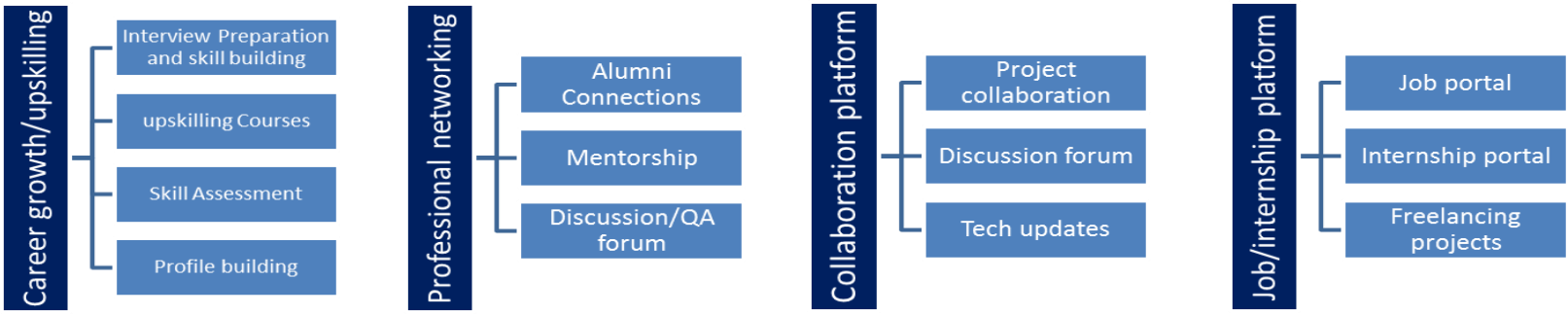
USC is a career development platform that delivers personalized executive coaching in a more affordable, scalable and measurable way.



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

[1] **Oracle Java Servlet Documentation** – Java™ Platform, Enterprise Edition (Jakarta Servlet API)

[2] **Oracle JavaServer Pages (JSP) Documentation** – Java™ Platform, Enterprise Edition (Jakarta JSP API)

[3] **HTML Drag and Drop API** – Mozilla Developer Network (MDN Web Docs)

2.6 Glossary

Terms	Acronym
CMS	Content Management System
CRUD	Create, Read, Update, Delete
JSP	Java Server Pages
JDBC	Java Database Connectivity
WYSIWYG	What You See Is What You Get editor

3 Problem Statement

1) Content Management System for a blog

WordPress and Drupal would be the best examples of full stack web app development project ideas for students. Using the CMS users must be able to design a web page using the drag and drop method. Users should be able to add textual or media content into placeholders that are attached to locations on the web page using drag and drop method.

This way, users should be able to design the whole website. Users must also get an option to publish blog posts. For this, you need to have a text editor component that accepts user input text and converts it into HTML and push into a database.

The website must be published over HTTP and HTTPS protocols such that the blog posts are served from the database and displayed to the visitors in the page template designed by the blog owner.

Explanation:

A Content Management System (CMS) is a full-stack web application that allows users (blog owners or content creators) to design, create, edit, and publish blog posts or entire websites without coding.

In this project, you'll create your own simplified version of CMS platforms like WordPress or Drupal — where users can design web pages using drag-and-drop tools, add text or media content, and publish their posts online.

Your CMS will have two main parts:

1. The Admin Area — where users log in, design page templates, write posts, and manage content.
2. The Public Website — where visitors can view the published blogs and pages.

Main Objectives:

1. Let users design webpages visually using drag and drop.
2. Allow users to write and format blog posts using a rich text editor.
3. Save and manage templates and blog posts in a database.
4. Serve (display) the final website to the public over HTTP and HTTPS.
5. Deliver posts dynamically from the database using the chosen page design.

4 Existing and Proposed solution

- **WordPress/Drupal**: Feature rich, plugin ecosystem, mature themes. However, they are heavy for learning purposes, involve complex configuration and a steep security hardening process. For the educational objective of understanding internals (layout storage, content rendering, and backend code), a lightweight custom CMS is more instructive.
- **Static Site Generators (Jekyll/Hugo)**: Lightweight and fast, but lack dynamic editing, admin dashboard, and live comments.

Limitations Addressed

- Complexity: Reduce unnecessary features; focus on core learning goals.
- Extensibility: Provide a modular backend that can be extended (e.g., to a headless CMS).
- Transparency: Implement features explicitly rather than hiding logic inside plugins.

Proposed Solution (value addition)

- Build a **modular CMS** that supports drag & drop layout configuration, WYSIWYG editor for posts, media upload and management, and a small but secure admin dashboard.
- Data models are simple and normalized for ease of understanding and future extension (users, posts, media, layouts, comments, categories).
- Implemented with **Servlet + JSP + JDBC**, suitable for running in **Eclipse** with Tomcat and MySQL.

4.1 Code submission (Github link)

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

5 Proposed Design/ Model

Given more details about design flow of your solution. This is applicable for all domains. DS/ML Students can cover it after they have their algorithm implementation. There is always a start, intermediate stages and then final outcome.

5.1 High Level Diagram (if applicable)

- **Users / Admins** interact via a web browser.
- Frontend UI includes the public blog view and an admin dashboard with drag & drop page designer and post editor.
- The web server is a Tomcat container hosting Servlets and JSPs which expose endpoints and render pages.
- The server interacts with a MySQL database via JDBC.
- Media files are stored locally under a configured media directory (or optionally on cloud storage/CDN).
- Optional caching (HTTP caching headers or Redis) can be added to improve public read performance.
- **5.2 Low Level Diagram & Data Model**

Design Notes:

- content_html stores the post content rendered by the editor (sanitized before saving).
- layout_structure stores page layout as JSON or serialized text that maps placeholders to types (text, image, gallery).
- slug is used for friendly URLs and is indexed for fast lookups.

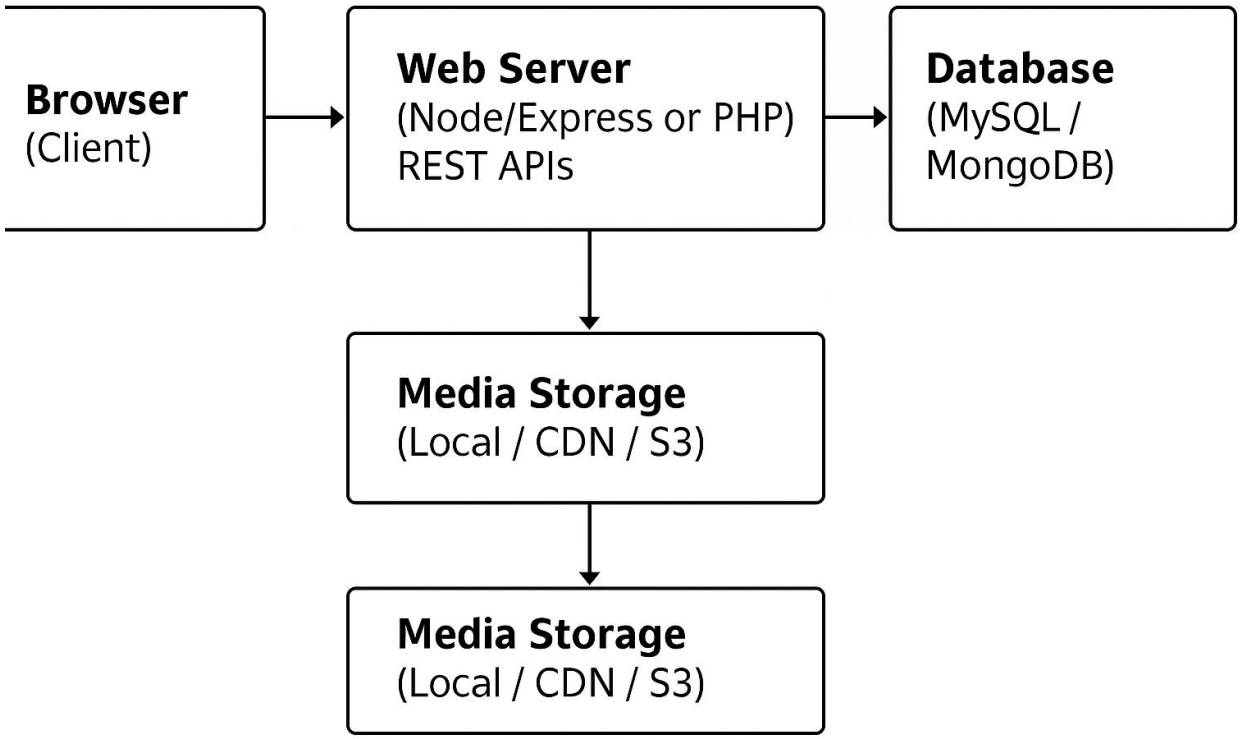


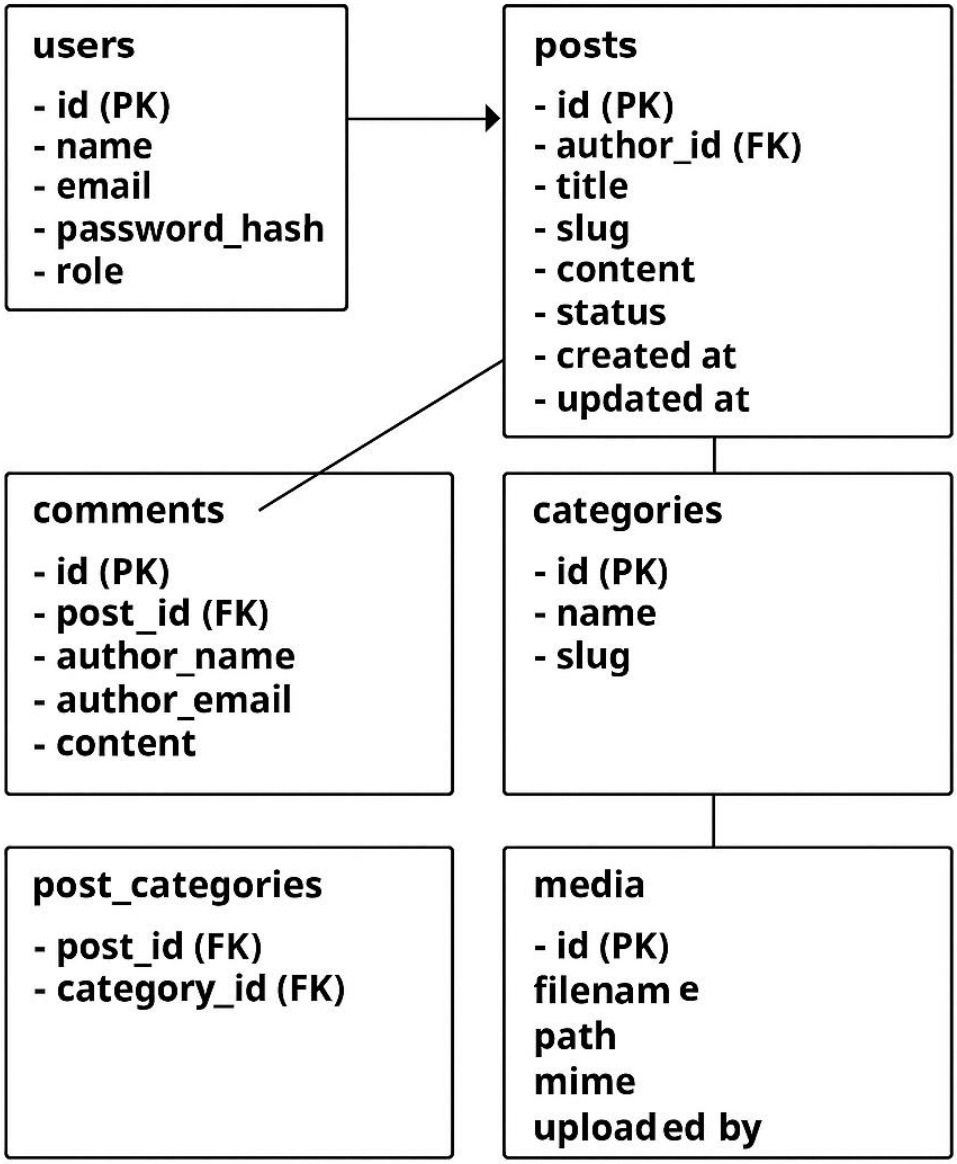
Figure 1: HIGH LEVEL DIAGRAM OF THE SYSTEM

5.2 Low Level Diagram

- **users(user_id PK, username, password_hash, email, role, created_at)**
- **posts(post_id PK, user_id FK, title, slug, content_html, excerpt, status, created_at, updated_at)**
- **media(media_id PK, user_id FK, filename, path, mime_type, uploaded_at)**
- **post_layouts(layout_id PK, post_id FK, layout_structure JSON/text, placeholder_map)**
- **categories(category_id PK, name, slug)**
- **post_categories(post_id FK, category_id FK)**
- **comments(comment_id PK, post_id FK, author_name, author_email, comment_text, status, commented_at)**

Design Notes:

- content_html stores the post content rendered by the editor (sanitized before saving).
- layout_structure stores page layout as JSON or serialized text that maps placeholders to types (text, image, gallery).
- slug is used for friendly URLs and is indexed for fast lookups.



5.3 Interfaces (if applicable)

The system exposes the following servlet endpoints (example design):

Public endpoints

- GET / — Render homepage (list of recent published posts)
- GET /post/{slug} — Render a single post by slug
- GET /category/{slug} — List posts by category

Admin endpoints (protected)

- GET /admin/login — Admin login page
- POST /admin/login — Authenticate admin (sets session)
- GET /admin/dashboard — Admin dashboard overview
- GET /admin/posts — List posts for admin
- GET /admin/posts/new — New post page (editor + layout selector)
- POST /admin/posts — Create post (title, content_html, layout JSON)
- GET /admin/posts/edit?id={id} — Edit post
- POST /admin/posts/update — Update post
- POST /admin/media/upload — Upload media (multipart form)
- POST /admin/comments/moderate — Moderate comment (approve/delete)

Authentication & Authorization

- Use secure session cookies (HttpOnly, Secure when HTTPS).
- All admin endpoints check session + role before performing actions.

6 Performance Test

This section outlines test planning, procedure, and observed/recommended outcomes. (Note: functional tests were executed against the local prototype; where live load tests were not feasible on the hosting environment, recommended test procedures and expected thresholds are provided.)

Constraints & Design Considerations

Constraints identified:

- Limited server memory and single-instance Tomcat in prototype.
- Media files can increase disk usage; need file-size limits.
- Concurrent reads (public visitors) may spike; caching is important.

Design Responses:

- Use MySQL indexes (on slug, created_at, status) to optimize read queries.
- Limit upload size (e.g., 5–10 MB per image) and validate mime types.
- Add HTTP caching and set Cache-Control or ETag for static assets.
- Sanitize input (prevent XSS) and prepared statements (prevent SQL injection).

6.1 Test Plan/ Test Cases

TC#	Test Case	Steps	Expected Result
TC1	Admin login	Enter valid/invalid credentials	Valid → session created; invalid → error
TC2	Create post	Admin creates post with title, HTML content and media	Post saved, status DRAFT or PUBLISHED
TC3	Publish post	Admin publishes the post	Post visible on homepage and URL works
TC4	View post	Visitor accesses /post/{slug}	Page loads under 500ms (on local)
TC5	Media upload	Upload image > 10MB	Fail with proper message
TC6	XSS attempt in content	Submit script tags in editor	Script sanitized before save/display
TC7	Concurrent reads	Simulate 200 concurrent GET requests to homepage	Response times remain acceptable (< 300–500 ms) if caching enabled
TC8	Comment moderation	Visitor posts comment; admin approves	Comment visible only after approval

6.2 Test Procedure

- **Unit & Integration Tests:** Manual and automated checks for servlet functions and DB transactions using simple JUnit tests and manual verification.

- **Security Tests:** Manual input validation and OWASP checklist validated; HTML sanitization applied on content.
- **Load Test Recommendation:** Use tools like k6 or ApacheBench to simulate reads: ab -n 500 -c 50 http://localhost:8080/ and measure response times. On prototype hardware, recommend enabling simple caching if response times degrade.
- **Media Validation:** Multipart upload tested with sample images; server enforces 10 MB limit and checks Content-Type.

6.3 Performance Outcome

- Functional tests (login, create, edit, publish, media upload) passed on the prototype environment.
- Security: HTML content sanitized using a whitelist sanitizer (e.g., Jsoup) to prevent XSS; SQL queries use prepared statements.
- For production readiness: use HTTPS, enable compression (GZIP), use a reverse proxy (Nginx), and add Redis or CDN for static media and caching of frequently requested pages.

7 My learnings

During the course of this internship, I gained a deep understanding of how a **Content Management System (CMS)** functions from both a technical and user perspective. The project provided me with valuable exposure to **full-stack Java web development**, combining both client-side and server-side technologies.

From the **frontend perspective**, I learned how to create responsive and interactive web interfaces using **HTML, CSS, and JavaScript**. Implementing the **drag-and-drop feature** helped me understand the importance of user experience (UX) design and how the HTML Drag and Drop API can be used to make website design more intuitive for end users.

On the **backend**, I developed strong skills in **Java Servlets, JSP, and JDBC**. This gave me a clear picture of how server-side programming works—handling HTTP requests and responses, managing sessions, and performing CRUD operations through database queries. I also learned how to connect a Java web application to a **MySQL database** securely and efficiently, using prepared statements to prevent SQL injection attacks.

The internship also taught me the importance of **security and validation**, such as sanitizing user input, validating file uploads, and implementing authentication for admin access. Additionally, I gained practical knowledge about **hosting and deployment** on an Apache Tomcat server, understanding how servlet containers manage web applications.

Beyond technical knowledge, this internship improved my **project planning, documentation, and version control** skills. Working within deadlines, maintaining a GitHub repository, and documenting each phase helped me adopt a structured, industry-standard workflow. Overall, this experience enhanced my confidence in working with real-world projects and strengthened my foundation in **Java-based full-stack development**.

8 Future work scope

The current CMS for a Blog fulfills the essential requirements of content creation, media management, and post publishing. However, there are several opportunities to extend its functionality and improve overall performance:

1. **Enhanced User Roles and Permissions**
   Introduce multiple user roles such as *Editor*, *Author*, and *Moderator*, each with specific access rights for managing content, comments, and layout settings.
2. **Advanced WYSIWYG Editor Integration**
   Upgrade to more powerful text editors like **CKEditor 5** or **TinyMCE**, supporting advanced formatting, embedded media, and real-time collaboration features.
3. **RESTful API Development**
   Extend the CMS backend to offer **REST APIs** that allow integration with mobile applications or external services, transforming it into a **Headless CMS**.
4. **Media Library Enhancements**
   Implement a centralized media library where all uploaded images and files are organized, searchable, and categorized for easy reuse.
5. **SEO and Analytics Features**
   Add support for meta tags, sitemap generation, and integration with **Google Analytics** to help blog owners track user engagement and improve visibility.
6. **Search and Recommendation Engine**
   Integrate a search feature using **MySQL Full-Text Search** or **ElasticSearch**, and later include a recommendation system based on post popularity and tags.
7. **Cloud Deployment and Scalability**
   Containerize the application using **Docker** and deploy it on platforms like **AWS** or **Google Cloud**, ensuring scalability and continuous integration with CI/CD pipelines.
8. **UI/UX Modernization**
   Improve the dashboard interface using frameworks like **Bootstrap** or **ReactJS** for a modern and responsive design that enhances user experience.