ADDIS ABABA SCIENCE AND TECHNOLOGY UNIVERSITY

College of Engineering

Department of Electrical and computer Engineering

Internship project Report on

Student Information Management System for Yekatit 12 Medical college

Name of Students ID. No:

1. Bamlak Geteye …… ETS0247/14
2. Bereket Tadiwos …….. ETS0214/13
3. Muluken Shiferaw ……ETS1129/14
4. Ruth Akalu ……...ETS1377/14
5. Samrawit Sissay ……...ETS1407/14

Supervisors:

1. Inst. Esubalew Mulat
2. Mr. Dagnachew Feleke

Submitted to Addis Ababa Science and technology University, Department of Electrical and Computer Engineering (Computer engineering focus area)

September 2025

Addis Ababa, Ethiopia

Table of Content

List of Figures

List of Abbreviations

CRVS	Civil Registration and Vital Statistics
ICCAIS	In Proceedings of the 2023 International Conference on Computer Applications and Information Science
IJRPR	International Journal of Research Publication and Reviews
AA ITDB	Addis Ababa Innovation and Technology Development Bureau
LMS	Learning Management Systems
ER	Entity-Relationship

Chapter One

Introduction of the hosting company

1.1 Brief History

The hosting company for our internship was Addis Ababa Innovation and Technology Development Bureau (ITDB). ITDB is a government organization that was originally constituted as an agency in April 2009 GC. The agency had been incorporated with the Technique and Vocational Training Agency and acquired a bureau structure, as a result of the restructuring of the city's executive bodies in September 2019 GC. Since then, the bureau has been carrying out various activities in the level of agency up to these days with a view to realize the vision to see building up of bridges to transform the city to overall ICT.

1.2 Product and Services

Throughout the 15 years of its time, the ITDB held the conducting research to improve existing technologies, developing new technology, establishing information systems, and ensuring the quality and reliability of IT infrastructure in the city. Specifically, its products and services include the following:

1. ICT Infrastructure Development: developing and designing city wide information technology infrastructures like data centers and cloud services
2. Software and Platform Development: involved in modernizing the government procedures and enhance efficiency through software solutions
3. E-Government and Digitalization: Implement electronic governance systems, improving service delivery for the public
4. Innovation Centers: foster culture of creativity and problem-solving, provide environment and resource for development and testing of up-coming technologies

5. Capacity Building: provide training programs like the Coders Initiative (also known as Ethiocoders) for the young

1.3 Customers and End users

The AA ITDB's main customers are mostly governmental bodies, residents and visitors of Addis Ababa. The bureau works to make government services more accessible and efficient towards its customers. Internally as a bureau, it also serves various city administration offices, providing them with the technological infrastructure and systems needed to facilitate their function. For instance, in partnership with Ethio-telecom, the bureau worked on a Smart Fire and Emergency Management Solution, aiming to improve emergency response time and data collection, and on the Addis Ababa Smart city project as well.

Other key partners include the Re Artificial Intelligence Institute (AI Institute), working to implement the new Civil Registration and Vital Statistics (CRVS) and e-ID system, and National ID Project Office (Fayda) in managing the registration work and integration into city services.

While the bureau is tightly involved with government offices, it also collaborates with private sector companies, academic institutions, and other organizations. Currently, private companies use the innovation centers provided by the bureau to nurture their new tech-businesses.

1.4 Organization and Workflow

The bureau is structured with directorates and sectors. Deputy Bureau Heads lead sectors, being the highest structural level after the bureau head. Directorates on the other hand are specialized operation units falling under sectors.

Under sectors:

1. Innovation and Technology Development Sector: focused on quality and safety of IT infrastructure and software system development. It has the Information Technology project Management, Software and Platform Development, and Information Technology Infrastructure Development and Design directorates under it.

2. Information Technology Operation Service Sector: involved in day-to-day management and maintenance of IT services and operations. It has a Data center and cloud, and Information Technology Operation and Maintenance directorates under it.
3. Smart City Sector: dedicated to transform the city by research and modernized by information technology, with Innovation and Technology Research and Innovation and Technology Development Bureau's capacity building and training directorates under it.

Under directorates:

1. Information System Security Directorate: prepares/improves information system security policy at city wide level and implements it, identify security issues, and conduct audit
2. Data Center and Cloud Directorate: responsible for developed infrastructures and systems for the city's data and cloud service
3. Information Technology Operation and Maintenance Directorate: manage the service and operations of IT infrastructures and their maintenance and repair
4. Information Technology Infrastructure Development and Design Directorate: responsible for implementing information technology infrastructure in accordance with international standards
5. Software and Platform Development Directorate: develops city-wide solution architectural framework, software development design, and software development feasibility studies
6. Information Technology project Management Directorate: formulate, start, plan, implement, monitor and support IT projects in the city administration institutions
7. Addis Ababa Innovation and Technology Development Bureau's capacity building and training Directorate: responsible for developing the skills of the bureau's staff and the public on IT
8. Innovation and Technology Research Directorate: responsible for advising, transferring and spreading technological results in collaboration with the community and stakeholders

Chapter two

Overall internship experience

2.1 Area of Work

During the internship period, we were assigned to the Development Section of the Addis Ababa Innovation and Development Bureau. Delivery of work from the very beginning was felt in a team environment: frontend development, backend development, UI/UX designs, database management, and system testing.

The main project we worked on was the Student Information Management System for Yekatit 12 Medical College. The system was supposed to handle student records, performance tracking, as well as support activities of the college administrative offices. The system was built using React and TypeScript for frontend development and Spring Boot for backend operations, while PostgreSQL was employed for the database, and Docker was involved for containerization, alongside Figma for UI/UX design.

Clear-cut division of labor, using professional tools, was displayed in these internships, which made the environment feel like real-world software development. It also gave each one of us the chance to develop ourselves in our respective fields while working on a common project.

2.2 Work Flow

The workflow had been minted contemporary and pleasant to the Development Section. We had a dedicated workspace, with the internet inside, that made teamwork useful.

In-person meetings were held regularly to discuss the tasks to be done, their progress, and trademark next steps to be taken.

The WhatsApp group was used for instant messages and brief topics.

Zoom meetings were held with the in-house supervisors or clients to share their vision about the expected type of work.

Besides teamwork, communication with supervisors was crucial. Any time we needed components, resources, or clarifications, we were expected to raise them on time to avoid incurring delay in our work. That way, our work was always moving at a consistent pace.

2.3 Action Plan Making

The learning point has mainly been how to translate client requirements into a concrete plan. This was especially necessary in developing the Student Information Management System. Our thought process goes:

1. Requirements gathering-What should the system achieve, consider factors of efficiency, usability, and client needs?
2. Employ components and tools: pick the right frameworks, codebases, technologies best for those functionalities (React, TypeScript, Spring Boot, Docker, PostgreSQL, Figma, for example).
3. System Design-Create wireframes and interactive prototypes with Figma and validate them with the client.
4. Implementation-Frontend and backend development working in parallel, with database integration in focus.
5. Testing & feedback-Functionality tests, demos with supervisors and clients, followed by fixes/view feature amendments with final approval.

A thorough manner like this has allowed us to give in to an operational product while at the same time learning how to apply a bit of theoretical knowledge in practice.

2.4 Tools and Platforms Used

Platforms and tools used to accomplish our tasks included:

Figma—for UI/UX design and prototyping.

React and TypeScript—for building user-friendly and type-safe interfaces.

Spring Boot—for backend logic and API development.

Docker for containerization and deployment.

PostgreSQL—managing and securing the database system.

The use of these tools has not only helped us with our project but also gave us insight into industry standards.

2.5 Challenges Faced

1. Though the majority of problems were tackled on the technical plane, some issues remained to be solved:
2. Bureaucracy: Bureaucratic processes often delayed projects from being considered officially started.
3. Waiting for client feedback: Sometimes, the flow of our work was interrupted with waiting for client feedback.
4. Learning curve: Some tools were new to certain participants, for instance, Docker.

2.6 Measures Adopted to Overcome Challenges

We put in some proactive measures to fight against these challenges:

1. Parallel works: Started working on tasks not needing any approval while these were being processed.
2. Presentations: Carried out presentations of the demos to clients and the business teams to accelerate decision-making.
3. Knowledge-sharing: In-house learning sessions were held to assist one another in adapting quickly to new tools and workflows.
4. These steps made it possible for us to maintain momentum against many external challenges.

2.7 Skills and Lessons Learned

On the whole, technical and soft skills alike were learned during this internship program:

Technical Skills

1. React and TypeScript for frontend development.
2. Spring Boot for backend development.
3. PostgreSQL for database management.
4. Docker for deployment and containerization.
5. Figma for UI/UX design.

Soft Skills

1. Teamwork and coordination.
2. Communicating with clients and supervisors.
3. Problem-solving and adaptability.
4. Time management and professional discipline.

2.8 Summary and Reflection

The internship was indeed a valuable learning opportunity, bringing theory to practice into one. We gained and improved technical skills and understood how to work within established professional workflows and organizational procedures. The development of the Student Information Management System was our challenge, and luckily, we triumphed as a team, sharpening our readiness for future employment in software development and project management.

Chapter three

SIMS: Medical College Student Services Platform

3.1 Short Summary of the Project

SIMS is a modular, containerized Student Information Management System built specifically for Yekatit 12 Memorial Medical College. It uses a microservices' backend (Java / Spring Boot) and a modern React + TypeScript frontend to provide role-based dashboards and services for administrators, faculty, staff, students, and alumni. Core features include student records, attendance tracking, course management, library services, payments, dormitory/cafeteria management, alumni/graduation workflows, and authentication/authorization. The system is designed to be deployed via Docker and uses Postgres for persistence and Eureka + API Gateway for service discovery and request routing.

3.2 Problem Statement & Justification

Problem:

Many colleges, specifically Yekatit 12 memorial medical collage, suffer from fragmented administrative systems, separate spreadsheets, manual attendance, and slow paper based workflows. These cause inefficiencies, lost records, duplicate entry, lack of transparent reporting, and slow service delivery to students and staff.

Justification:

A centralized, role aware digital system reduces administrative overhead, improves data accuracy, speeds up routine transactions (fee payments, library loans, dorm allocation), improves accountability (attendance and grading), and makes alumni management and reporting far easier. For the college, timely and accurate student records and attendance tracking are critical for accreditation, licensing, and clinical placements, making SIMS a high-value investment.

3.3 Objective of the Project

Primary objective:

Design and implement a scalable, maintainable, and secure Student Information Management System for Yekatit 12 Memorial Medical College that centralizes student related services and supports different user roles.

Specific goals:

1. Provide role-based dashboards (admin, faculty, staff, student, alumni) with tailored views and capabilities.
2. Implement core microservices for student records, courses, attendance, library, payment, dormitory, cafeteria, alumni, and graduation.
3. Offer a centralized authentication & authorization service and API gateway with service discovery.
4. Containerize services and provide reproducible deployments via Docker Compose.
5. Use PostgreSQL for durable storage and build a frontend with React + TypeScript, Vite, and Tailwind CSS that is responsive and user-friendly.
6. Ensure data privacy and role-based access control; implement audit logging and secure communication.
7. Provide extensibility points for integrations (payment gateways, national ID systems, email/SMS).

3.4 Literature Review

Introduction

Student Information Management Systems (SIMS), which refers to software solutions designed to centralize and automate the administration of student records, academic data, and institutional services. The scope of this review covers commercial enterprise-level systems, open-source platforms, and learning management systems (LMS), with a focus on how they address challenges of data management, service integration, and user experience.

This review does not explore highly specialized educational analytics systems or systems used exclusively in primary/secondary schools. Instead, the focus remains on higher education SIMS solutions that provide comprehensive administrative and academic support.

Across most of the sources reviewed, there is a general consensus that centralized systems improve efficiency, accuracy, and transparency in academic administration. However, differences emerge in system architecture, cost, scalability, and adaptability to specific institutional needs.

Main Body

Commercial Systems

Ellucian Banner, Oracle PeopleSoft, and Workday Student are leading commercial SIS solutions. They offer extensive modules covering admissions, finance, human resources, and academic records. Their methodologies emphasize integration across institutional departments and compliance with international education standards.

1. **Strengths**: Feature-rich, highly customizable, strong vendor support.
2. **Weaknesses**: High licensing and maintenance costs; complexity that may exceed the needs of smaller institutions.
3. **Relevance:** While robust, these solutions are not easily adaptable to local medical colleges due to their cost and rigid structures.

Open-source Student Information Systems

Platforms such as OpenSIS and Fedena provide cost-effective, community-driven solutions. They generally follow a monolithic architecture, combining all features into a single codebase.

1. **Strengths:** Affordable, accessible source code, flexible for customization.
2. **Weaknesses:** Limited scalability, less modular, and often lacking enterprise-grade security or support.

3. **Relevance:** These solutions demonstrate the feasibility of centralized student management but lack the scalability and modularity offered by microservices.

Learning Management Systems (LMS)

Systems like Moodle and Canvas focus on content delivery, course management, and student learning outcomes. Their methodology centers around pedagogy and student-teacher interaction rather than administrative processes.

1. **Strengths:** Excellent for e-learning and content management.
2. **Weaknesses:** Do not fully address institutional needs such as finance, dormitory, or alumni management.
3. **Relevance:** They complement SIS platforms but cannot replace them for administrative purposes.

Emerging Approaches

Recent literature and case studies highlight the adoption of microservices architecture and cloud-native deployments in education systems. Compared to monolithic systems, microservices provide better scalability, fault isolation, and independent service development. Containerization further improves deployment reproducibility and system resilience.

1. **Strengths:** Scalability, maintainability, and flexibility in service updates.
2. **Weaknesses:** Added complexity in deployment and monitoring.
3. **Relevance:** This approach aligns directly with the architecture adopted in the SIMS project.

Conclusion

The reviewed literature covers some of the solutions, from costly enterprise-grade systems to lightweight open-source platforms. A key commonality is the recognition that centralization improves efficiency and data reliability. However, existing systems show notable gaps: commercial solutions are often financially inaccessible to smaller institutions, while open-source systems struggle with scalability and flexibility.

The SIMS project bridges these gaps by leveraging a microservices architecture that is both modular and scalable, while remaining adaptable to the specific requirements of Yekatit 12 Memorial Medical College. Unlike traditional monolithic solutions, SIMS enables independent

scaling of services such as attendance, payments, or alumni management, making it a future-proof and context-aware system.

Thus, the findings of this review justify the need for SIMS as a tailored, cost-effective, and technologically modern solution for higher education institutions.

3.4 Methodology

Below is a concise description of the system architecture, technology choices, and how the pieces fit together.

High-level architecture

1. **Microservices**: Each domain area is its own Spring Boot service (e.g., studentservice, attendanceservice, libraryservice, paymentservice, dormitoryservice, cafeteriaservice, alumniservice, graduationservice, userauthenticationservice, etc.). Each has src/, pom.xml, and a Dockerfile — standard Maven + Spring Boot layout.
2. **Service Discovery**: eurekaserver provides registration and discovery of services.
3. **API Gateway**: apigateway routes external requests to internal services, applies cross-cutting concerns (auth, rate-limiting, routing).
4. **Persistence**: PostgreSQL as the relational database.
5. **Orchestration**: docker-compose.yml defines and runs the multi-container stack for local development and testing.
6. **Frontend**: /frontend/ is a React + TypeScript application using Vite as the bundler and Tailwind CSS for styling. It contains role-specific dashboards, authentication pages, and shared UI components.

Data flow

1. User submits credentials on frontend → POST to apigateway → userauthenticationservice.

2. userauthenticationservice authenticates (likely issues JWT or session token).
3. Frontend stores the token and sends it in the Authorization header for subsequent requests to apigateway.
4. apigateway validates token (or delegates to auth service) and forwards requests to target microservices (e.g., attendanceservice).
5. Microservices access Postgres and return responses to the gateway → frontend.

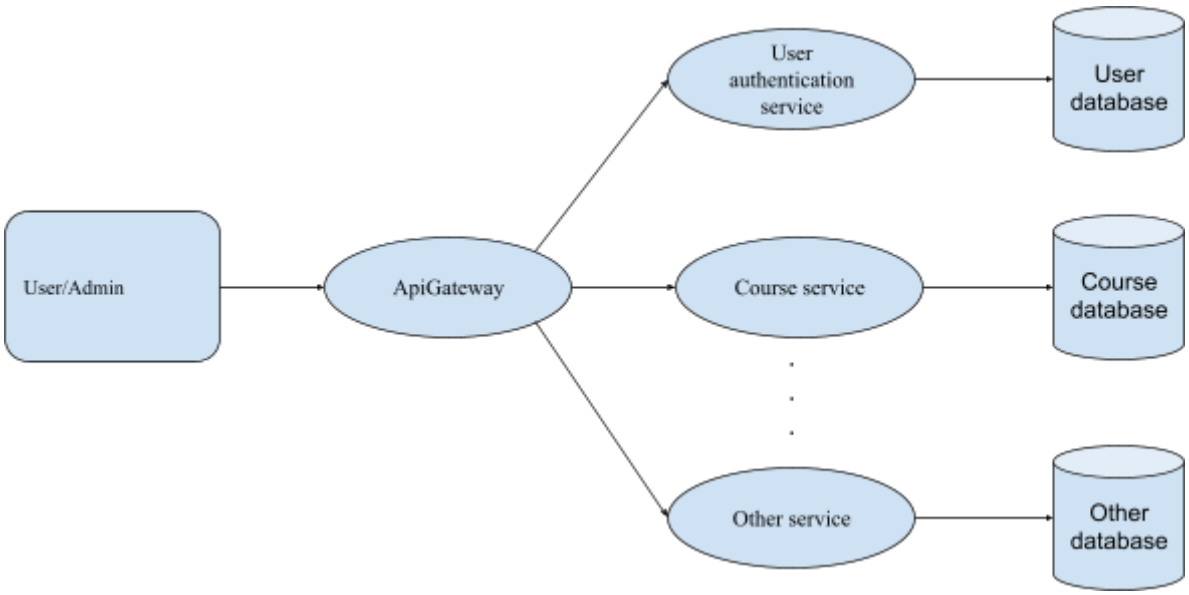

Figure 1: Data flow diagram in the system

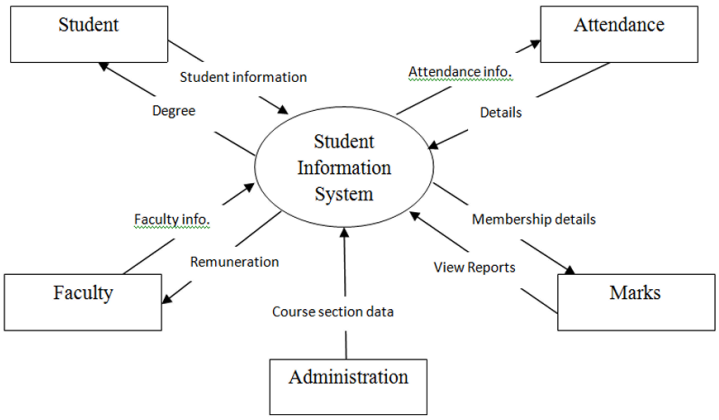

Figure 2: Data flow diagram to and from each user type

Technologies & why chosen

1. **Java + Spring Boot**: Robust ecosystem for building RESTful services, dependency injection, easy testing, and mature libraries for security (Spring Security), data access (Spring Data JPA), and application config.
2. **Maven**: Standard Java build tool, used per service.
3. **Eureka / API Gateway**: Simplifies discovery and routing in a microservice environment.
4. **PostgreSQL**: ACID, strong relational support for student records and transactions.
5. **Docker / Docker Compose**: Reproducible deployment and local dev environment.
6. **React + TypeScript**: Type safety in the frontend, modular UI components, better DX.

7. **Vite**: Fast dev server and build step.
8. **Tailwind CSS**: Utility-first styling for rapid UI development and consistent design.
9. **Role-based dashboards**: Clear separation of concerns and the UX tailored to each user class.

Implementation details & conventions

1. Each microservice includes src/main/java, src/main/resources/application.yml or application.properties, unit/integration tests under src/test.
2. Dockerfile per service enables containerization for the service and its dependencies.
3. docker-compose.yml likely defines containers for the services, Postgres, Eureka, and the API Gateway.
4. Frontend likely uses an Axios client that points to the API Gateway; components split per role into src/components/ and src/pages/.

The following are some of the database structures for the services in the system.

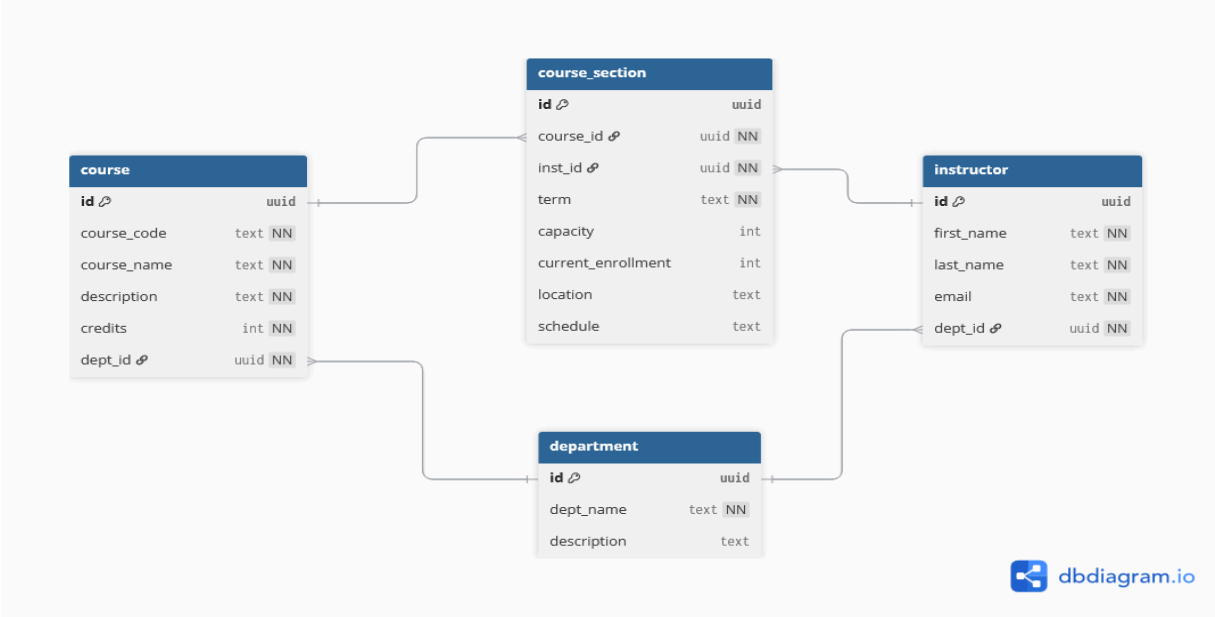

Figure 3: Designed Course Service Database Entity-Relationship (ER) diagram

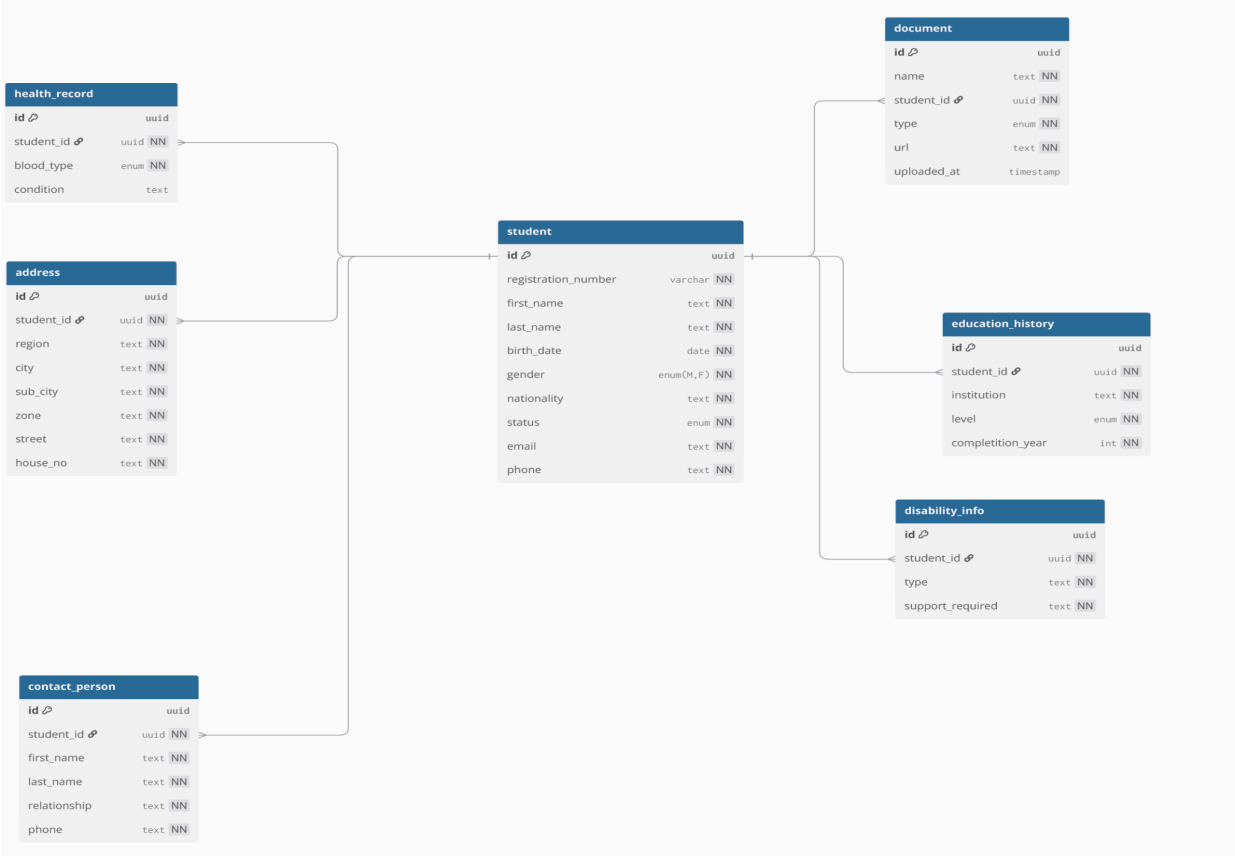

Figure 4: Designed Student Service Database ER diagram

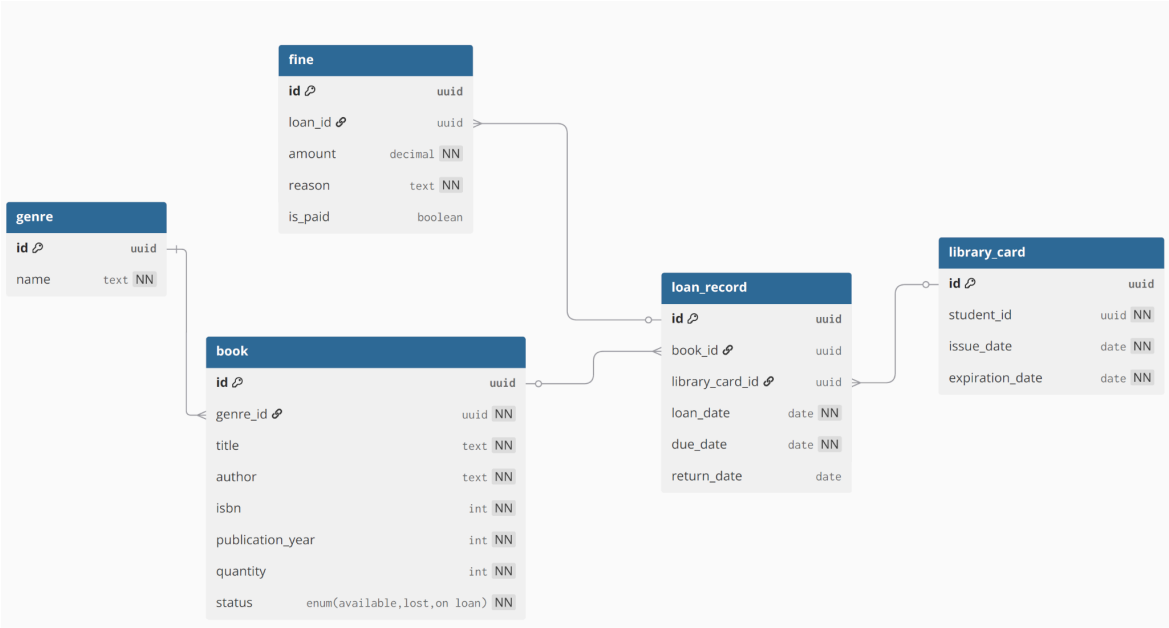

Figure 5: Designed Library Service Database ER diagram

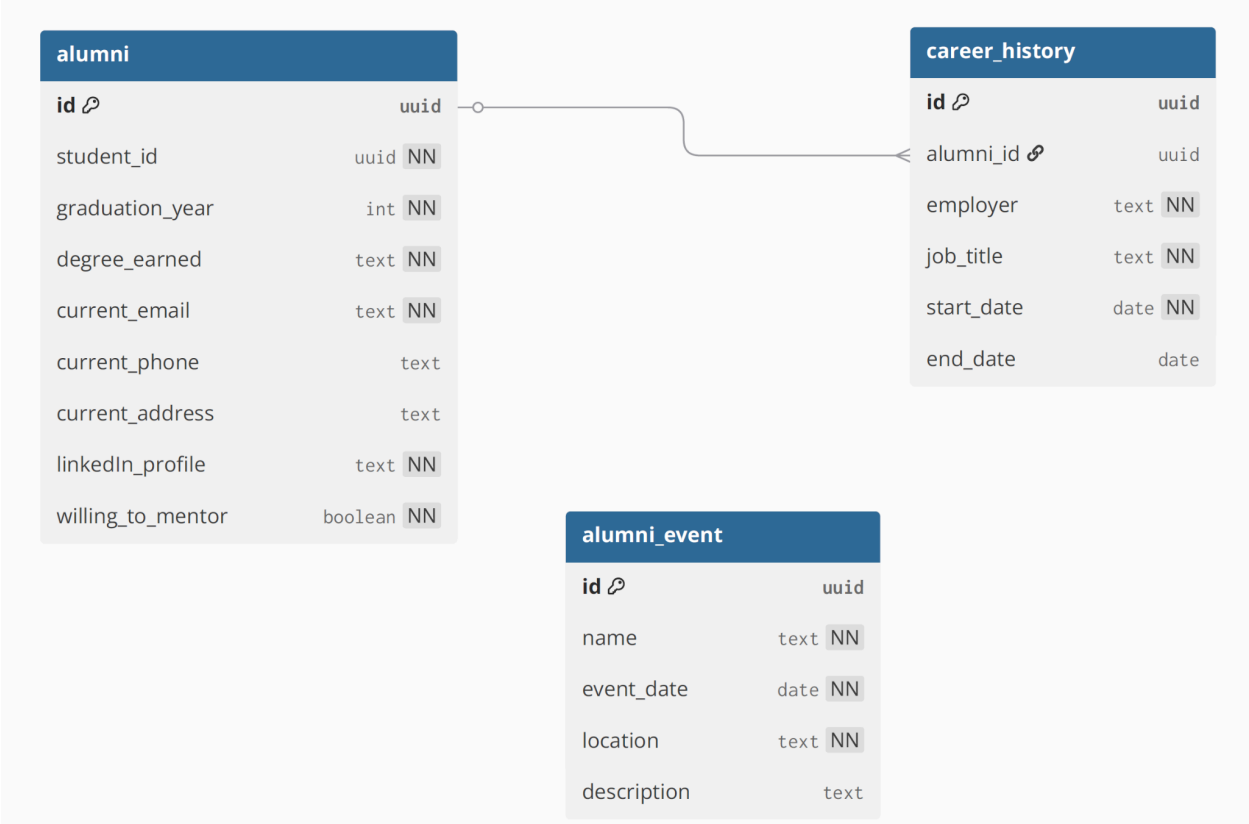

Figure 6: Designed Alumni Service Database ER diagram

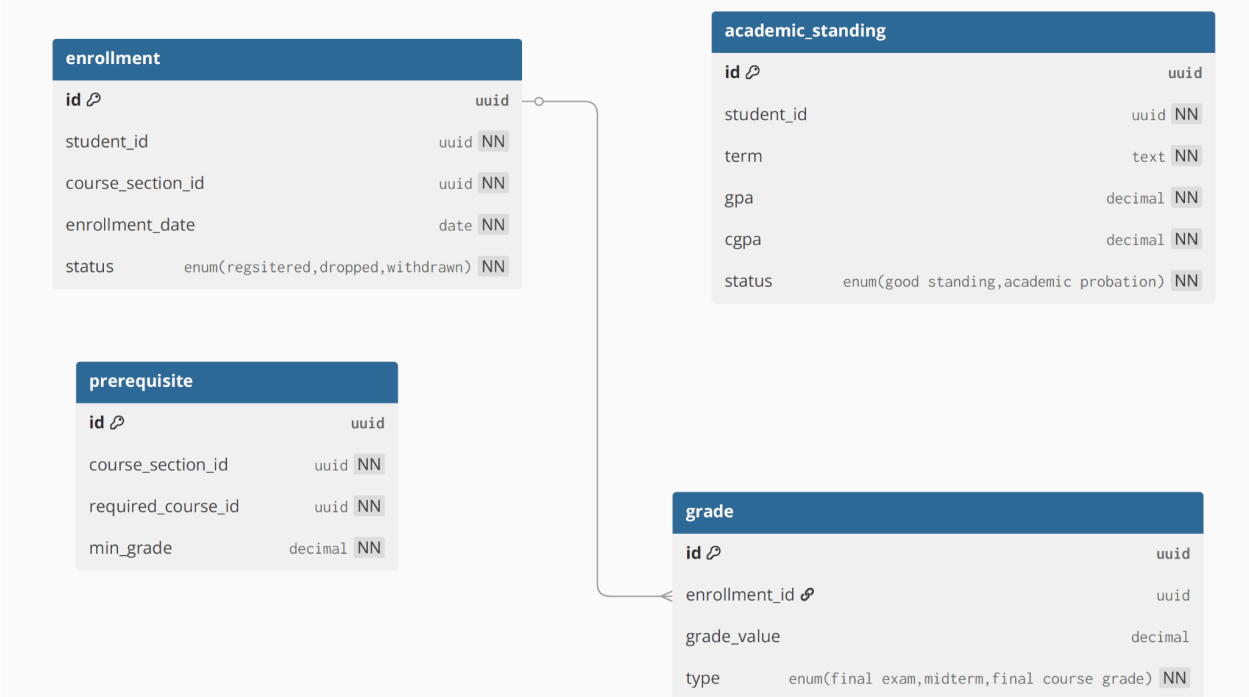

Figure 7: Designed Enrollment Service Database ER diagram

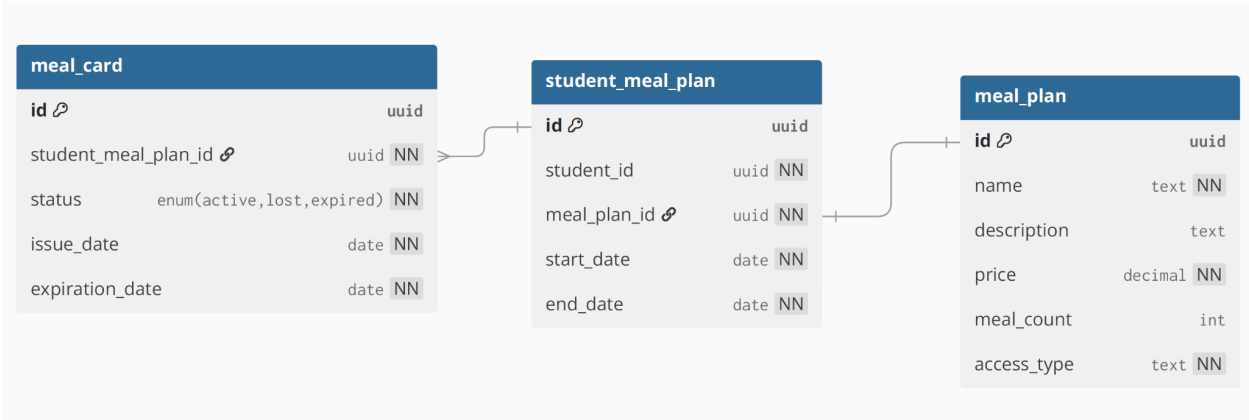

Figure 8: Designed Cafeteria Service Database ER diagram

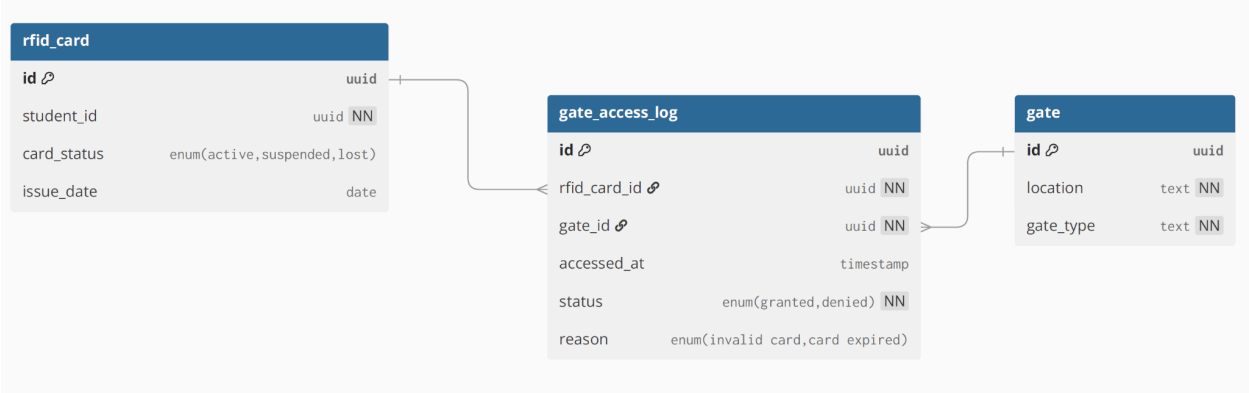

Figure 9: Designed Gate Service Database ER diagram

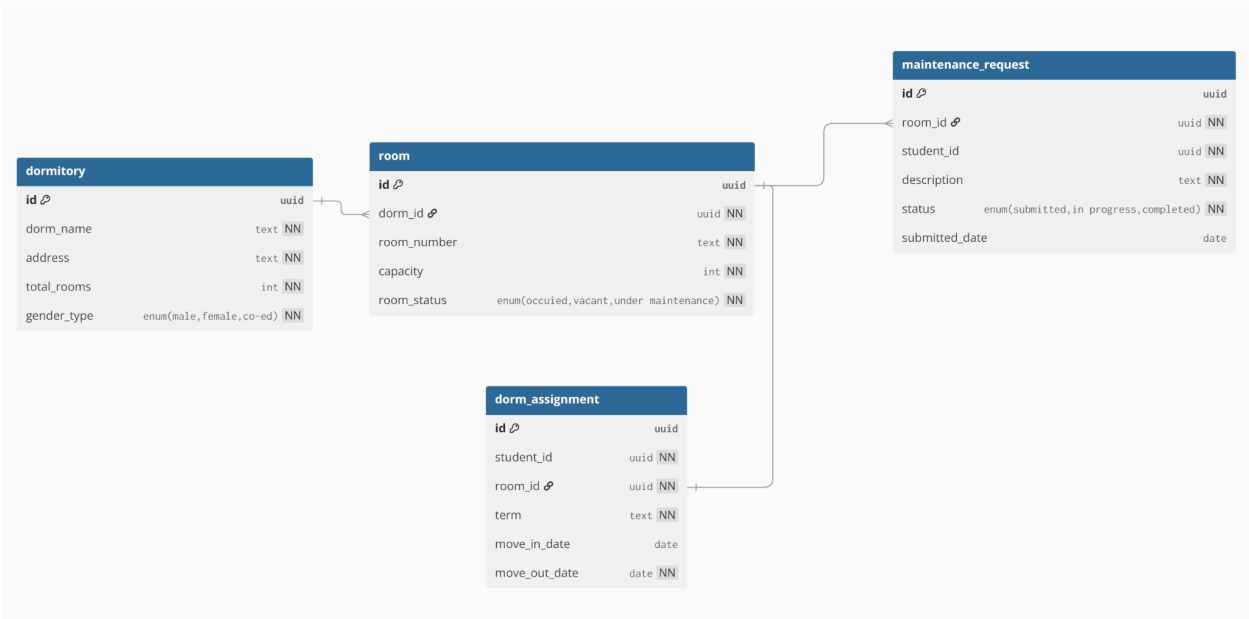

Figure 10: Designed Dormitory Service Database ER diagram

Non-functional considerations implemented

1. **Security:** central authentication service, role-based access control, HTTPS in production, password hashing, secure token issuance (JWT/OAuth2).
2. **Scalability:** microservice design allows independent scaling of high-load services (e.g., payment or gateway).
3. **Observability:** logging, centralized logs, metrics (Prometheus).

4. **CI/CD:** Docker images, tests, images pushed to registry.

3.5 Result & Discussion

Expected results

1. **Functional system** delivering integrated features: centralized student records, role-based access, attendance automation, library and cafeteria management, dormitory allocations, payment handling, alumni tracking, and graduation workflows.
2. **Reproducible deployments** for dev/testing using Docker Compose; easier path to production via container orchestration.
3. **Responsive front-end** user experience with tailored dashboards for each role.

Potential impact

1. **Administrative efficiency:** reduced manual work and improved accuracy in student records and reporting.
2. **Student experience:** faster access to services (payments, dorm requests, library loans) and clearer visibility of progress and requirements.
3. **Institutional outcomes:** better compliance readiness and traceability for accreditation, simplified reporting to regulators.

Insights & trade-offs

1. **Pros:** Modularity, independent scaling, developer autonomy, clear separation of concerns, modern UX.
2. **Cons / Challenges:** Microservices add operational complexity: distributed logging, inter-service communication failures, eventual consistency challenges, database transaction complexity across services, versioning and backward compatibility.
3. **Operational needs:** To be production-ready, the project should add monitoring, automated backup, robust security hardening, encrypted secrets, database migrations (Flyway/Liquibase), and orchestration with k8s and horizontal autoscaling.

Project Folder Structure

Backend

1. studentservice/, attendanceservice/, userauthenticationservice/, etc. → Domain-specific microservices.
2. apigateway/ → Centralized request routing.
3. eurekaserver/ → Service discovery.
4. postgres/ → Database configuration.
5. docker-compose.yml → Orchestration of services.

Frontend

1. src/components/ → Authentication, dashboards, UI elements.
2. src/pages/ → Landing page, role-based views.
3. vite.config.ts, tailwind.config.js → Build and styling.

Chapter Four

Overall internship achievements and experiences

During our internship at Addis Ababa City Innovation and Technology Development Bureau, we developed a Student Information System using Spring Boot for the backend, React for the frontend, Swagger for service documentation, eureka for service discovery, API Gateways for service management, and Figma for UI/UX design. This project provided us with several benefits in terms of knowledge, skills, and professional development.

4.1 Upgrading Theoretical Knowledge

The internship played a vital role in connecting the theories we studied in classrooms with the realities of system development. Concepts that were once abstract in textbooks became more concrete when applied to the Student Information System project. For instance, client server architecture was no longer just a diagram we memorized; we experienced firsthand how the backend and frontend communicate, exchange data, and depend on each other for performance and reliability.

Working with RESTful APIs gave us deeper insight into how data is structured, transmitted, and secured across different services. We gained a better understanding of the role of endpoints, request response handling, and the importance of API documentation, which was reinforced through the use of Swagger. Similarly, applying microservices taught us the advantage of modularizing the system into smaller, manageable services, each responsible for a specific functionality. This helped us appreciate scalability and maintainability as practical necessities rather than just theoretical ideals.

On the design side, the use of UI/UX principles through Figma revealed the importance of user centered design. We learned that designing for real users requires balancing functionality with simplicity, which deepened our theoretical grasp of human computer interaction.

Additionally, the internship broadened our theoretical knowledge of distributed systems and scalability, particularly when working with API gateways to manage multiple services. We saw how concepts like load balancing, authentication, and secure communication protocols directly contribute to system robustness.

Overall, the experience gave us a strong foundation by turning abstract theories into lived experiences. This not only reinforced what we had studied but also highlighted gaps in our theoretical understanding, motivating us to learn more beyond the classroom.

4.2 Improving Practical Skills

One of the most valuable aspects of the internship was the opportunity to strengthen our practical skills by working directly with modern tools and technologies. While theory provided us with a foundation, it was through actual implementation that we learned how to build, test, and refine a complete software system.

On the **design side**, working with **Figma** introduced us to professional UI/UX design workflows. We learned how to create wireframes, mockups, and interactive prototypes that made system development more structured and user focused. This improved our ability to translate user requirements into intuitive interfaces and helped us understand the significance of usability testing.

For frontend development, using React allowed us to gain deeper experience with component based architecture, state management, and responsive design. We became more skilled at writing reusable components, handling data flow through props and states, and integrating the frontend with backend services. This also improved our ability to debug and optimize web applications for performance and scalability.

On the backend side, working with Spring Boot was a major step forward in building enterprise level applications. We gained hands-on experience in creating RESTful APIs, managing databases, handling authentication, and ensuring security in communication. The internship also taught us how to structure backend code in a modular and maintainable way, which is a crucial skill in industry projects.

Another important area of skill development was API documentation and service management. Using Swagger for documentation and API gateways for integration allowed us to understand how services are exposed, consumed, and maintained. This practical knowledge is directly applicable to real world systems, where clarity and consistency in service communication are essential.

Altogether, these practical skills made us more confident in our ability to design, develop, and deploy software solutions that align with current industry standards.

4.3 Enhancing Industrial Problem Solving Capability

During the internship, we faced different technical and organizational challenges that required us to think critically and apply problem-solving strategies. For example, debugging backend services, fixing integration issues between the frontend and backend, and handling database errors taught us how to approach problems systematically rather than randomly trying solutions.

We also improved in translating user requirements into technical implementations. The Student Information System needed to reflect the actual workflows of the college, so we learned to analyze the problem from the user's perspective and design solutions that were both functional and efficient.

This experience strengthened our ability to deal with industry level challenges by teaching us persistence, structured troubleshooting, and the importance of testing and validation in solving problems effectively.

4.4 Improving Teamwork Skills

The internship required us to work as a group, which gave us valuable experience in team collaboration. We had to coordinate tasks, review each other's code, and make collective design decisions. This improved our ability to communicate technical ideas clearly and listen to different perspectives before reaching a conclusion.

Through pair programming and group discussions, we learned how to divide responsibilities effectively while still supporting one another when challenges arose. We also became better at using collaboration tools and version control systems to keep our work synchronized.

Overall, the teamwork experience taught us the importance of cooperation, respect for diverse ideas, and aligning individual efforts toward achieving a shared goal.

4.5 Developing Leadership Skills

Throughout the project, we had several moments where leadership was necessary. Some members took initiative in organizing meetings, dividing tasks, and guiding technical discussions. Others stepped up when unexpected issues occurred, helping the group stay focused and motivated.

These experiences allowed us to practice decision-making, task delegation, and time management in a real project setting. We learned that leadership is not only about giving directions but also about supporting teammates, ensuring progress, and setting a good example through responsibility and consistency.

By the end of the internship, we had collectively strengthened our leadership abilities, becoming more confident in handling responsibility within a team environment.

4.6 Understanding Work Ethics

The internship gave us practical exposure to professional work ethics, which are just as important as technical skills. We learned the value of punctuality by following schedules and meeting deadlines set for each phase of the project. Accountability also became clear, as every member had to deliver their assigned tasks with quality and consistency.

We also practiced integrity in coding by avoiding shortcuts that could compromise system reliability, and by maintaining proper documentation for future use. Communication with supervisors and teammates further taught us the importance of respect, transparency, and responsibility in a professional environment.

Overall, the internship showed us that strong work ethics form the foundation for building trust, maintaining productivity, and achieving long-term success in any industry.

4.7 Strengthening Entrepreneurship Skills

Working on a real project for Yekatit 12 Medical College gave us insights into how technical solutions can be transformed into valuable products and services. Developing the Student

Information System showed us how to identify a real institutional problem, propose a workable solution, and deliver it in a way that creates impact.

This process helped us understand the entrepreneurial mindset recognizing opportunities, applying innovation, and balancing user needs with technical feasibility. We also saw how teamwork, planning, and resource management contribute to successful project execution, which mirrors how startups and tech companies operate.

By the end of the internship, we had a stronger appreciation of how software development is not only a technical exercise but also a pathway to entrepreneurship, where ideas can be scaled into sustainable ventures.

Chapter Five

Conclusion & Recommendation

5.1 Conclusion

The eight-week internship at Addis Ababa City Innovation and Technology Development Bureau provided us with a transformative opportunity to bridge theoretical knowledge with practical application. Tasked with developing a comprehensive Student Information Management System for Yekatit 12 Medical College, we were able to apply modern engineering principles and technologies to solve a real institutional problem.

The project enabled us to move beyond classroom concepts by experiencing firsthand how microservice-based architectures, and frontend-backend integration operate in practice. The Spring Boot microservices, PostgreSQL for relational data handling, React with TypeScript for frontend development, and Figma for user interface design collectively demonstrated the interconnectedness of modern development tools. The architecture not only reinforced our technical competence but also underscored the importance of modularity, scalability, and maintainability in enterprise-level systems.

On a practical level, the internship significantly enhanced our skills in backend and frontend development, UI/UX design, and service documentation through Swagger. It also sharpened our ability to solve technical challenges, debug complex integrations, and deliver functional software aligned with user needs. Working within a team strengthened our collaboration, leadership, and communication abilities, while exposure to deadlines and accountability helped us internalize professional work ethics.

Equally important, the internship exposed us to industry-level challenges such as distributed service management, database consistency, and security hardening. Addressing these issues deepened our problem-solving capability and prepared us for the professional demands of the software industry. Furthermore, by developing a product with real institutional impact, we gained

an entrepreneurial perspective on how innovation and technology can transform organizational efficiency.

In summary, the internship was a holistic learning experience that reinforced theoretical foundations, improved technical and professional skills, and expanded our capacity to innovate. Beyond delivering a functional Student Information Management System, the journey itself became a milestone in our academic and professional growth. It demonstrated that innovation is not only about mastering technologies but also about integrating teamwork, responsibility, and creativity to solve real-world problems

5.2 Recommendation

During our internship at the Addis Ababa Innovation and Technology Development Bureau, we got a chance to actually design and implement a Student Management System for Yekatit 12 Medical College. We realized it was a good opportunity for us interns to put the theory we studied to practice and at the same time learn new skills in Spring Boot, React, PostgreSQL, Docker, and Figma. Suggestions from our experiences and challenges during the implementation phase:

Better Requirement Gathering and Planning

At the early stage, some functional requirements were not very well-defined, which, in the long-term, created gaps during development. Hence, it is recommended that such projects should start with a more structured need-gathering session between the interns and the end-users so that everybody sets expectation before the implementation actually starts.

Improved Access to Development Resources

While building the system, we sometimes faced delays in accessing software resources and documentation. We recommend that the Bureau provide interns with centralized access to technical resources,

Technical Supervision Closer and Feedback

The support of the supervisors was greatly appreciated. However, from time to time, technical feedback was given so late in the process that making some adjustments became very troublesome. A recommendation could be having regular technical review sessions (weekly or bi-weekly), so interns can be able to have timely feedback and align themselves with the standards of the Bureau.

Collaboration

Team collaboration was effective, but we noticed that communication between interns and staff could be improved by adopting structured project management tools

References:

1. Addis Ababa Innovation and Technology Development Bureau. (n.d.). *History and mandate*. Retrieved September 14, 2025, from https://www.aaitdb.gov.et/
2. Docker Inc. (2022). *Docker documentation*. Docker. https://docs.docker.com/
3. Docker Inc. (2022). *Compose file reference*. Docker. https://docs.docker.com/reference/compose-file/
4. Huang, H., & Li, B. (2024). Design and implementation of student management system of integrated programmable device programming system. *Scientific Reports, 14*(1), 11873. https://doi.org/10.1038/s41598-024-62844-z
5. Johnson, R., Hoeller, J., et al. (2021). *Spring Framework documentation*. Spring. https://spring.io/projects/spring-framework
6. Liu, Q., Chen, J., & Wang, L. (2023). Student information management system based on JSP. In *Proceedings of the 2023 International Conference on Computer Applications and Information Science (ICCAIS)*. ACM Digital Library. https://doi.org/10.1145/3660043.3660154
7. PostgreSQL Global Development Group. (2021). *PostgreSQL documentation*. https://www.postgresql.org/docs/
8. React Team. (2022). *React documentation*. Meta Platforms, Inc. https://react.dev
9. Zhou, H., Liu, X., & Chen, Y. (2012). Design of student information management database. *Physics Procedia, 25*, 1139–1144. https://doi.org/10.1016/j.phpro.2012.03.206
10. "Student information management system." (2022). *International Journal of Research Publication and Reviews (IJRPR), 3*(6), 197–203. https://ijrpr.com/uploads/V3ISSUE6/IJRPR5367.pdf
11. Spring Boot Team. (2023). *Spring Boot reference documentation*. Spring. https://docs.spring.io/spring-boot/reference/
12. Ethio Telecom. (2022, November 3). *Ethio Telecom and the Addis Ababa City Administration Innovation and Technology Development Bureau sign a smart fire and emergency management solution implementation agreement*. Ethio Telecom.

https://www.ethiotelecom.et/ethio-telecom-and-the-addis-ababa-city-administration-innovation-and-technology-development-bureau-sign-a-smart-fire-and-emergency-management-solution-implementation-agreement