Salale University Dormitory Management System (DMS) Documentation

By: YOHANNES TULU

Salale University Dormitory Management System (DMS) Documentation

Overview

The Dormitory Management System (DMS) is a comprehensive web application designed to streamline the management of university dormitories. It provides tools for administrators, proctors, and students to efficiently handle room assignments

1. Statement of the problem

Currently, SLU dormitory management system uses manual approach. To process the operation first the ministry of education sends all the information to the registrar bureau and gives to the student affairs (dormitory) and to the dinning office. After taking the list, they assigned students to each block and room. At that time they face different problems during operating their tasks. Working by paper based i.e. manual system is not only affecting the management members, rather it also for student during viewing of their dormitory information. Some of those problems are:-

- Data duplication and Time consuming.
- Require more human power to assign the students.
- Management inflexibility

Significance of the project

The new online dormitory management and allocation system is highly reliable, easy, fast and consistent and will play a crucial role for reliable service for students, proctors, and for the management. The significance of the system includes:

- To minimize time and efforts needed to perform tasks.
- To make tasks simple and efficient in every aspects.
- To manage the students and building information.
- Providing a well-organized and guaranteed record keeping system with minimum space and effort need.
- To enable the university to get acceptance in the outside community.
- Developing students' effective communication with the university.

1. *Requirement Analysis*

The Requirement Analysis phase was the foundation upon which the entire Dormitory Management System was built. This stage involved a deep dive into the day-to-day operations of Salale University's dormitory management, seeking to understand the challenges faced by students, proctors, and administrators.

Stakeholder Engagement

The process began with engaging stakeholders through interviews, surveys, and observation. Students expressed frustration with not knowing their room assignments until the last minute, and sometimes being assigned to rooms that did not match their preferences or needs. Proctors and dormitory managers described the overwhelming task of manually tracking room occupancy, handling student requests, and updating records. Administrators highlighted the need for accurate reporting and the ability to quickly respond to changes in student numbers or room availability.

Identifying Functional Requirements

From these discussions, several functional requirements emerged. The system needed to:

- Provide secure login for different user roles (students, proctors, admins), ensuring that only authorized users could access sensitive information or perform administrative tasks.
- Allow administrators and proctors to register new rooms and blocks, update room capacities, and manage occupancy.
- Enable the assignment of students to rooms based on criteria such as sex, batch, faculty, and room availability.
- Support the creation, editing, and deletion of user accounts, with clear role-based permissions.
- Offer students a way to view their dormitory assignments online, reducing confusion and improving transparency.
- Include a comment or feedback system, allowing users to report issues or suggest improvements.
- Generate reports on room occupancy, student lists, and other key metrics for decision-making.

Non-Functional Requirements

Beyond these core features, non-functional requirements were also identified. The system had to be:

- **Secure:** Protecting student and staff data through password hashing, session management, and input validation.

- **Usable:** Featuring a clean, intuitive interface that worked well on both desktop and mobile devices.
- **Reliable:** Ensuring data integrity and minimizing downtime, especially during peak assignment periods.
- **Scalable:** Capable of handling growth in student numbers and the addition of new dormitory blocks.
- **Maintainable:** Designed with modular code and clear documentation to facilitate future updates and troubleshooting.

Documenting Requirements

All requirements were documented in a requirements specification, serving as a contract between stakeholders and the development team. This document guided the design and implementation phases, ensuring that the final system would address the real-world needs of its users.

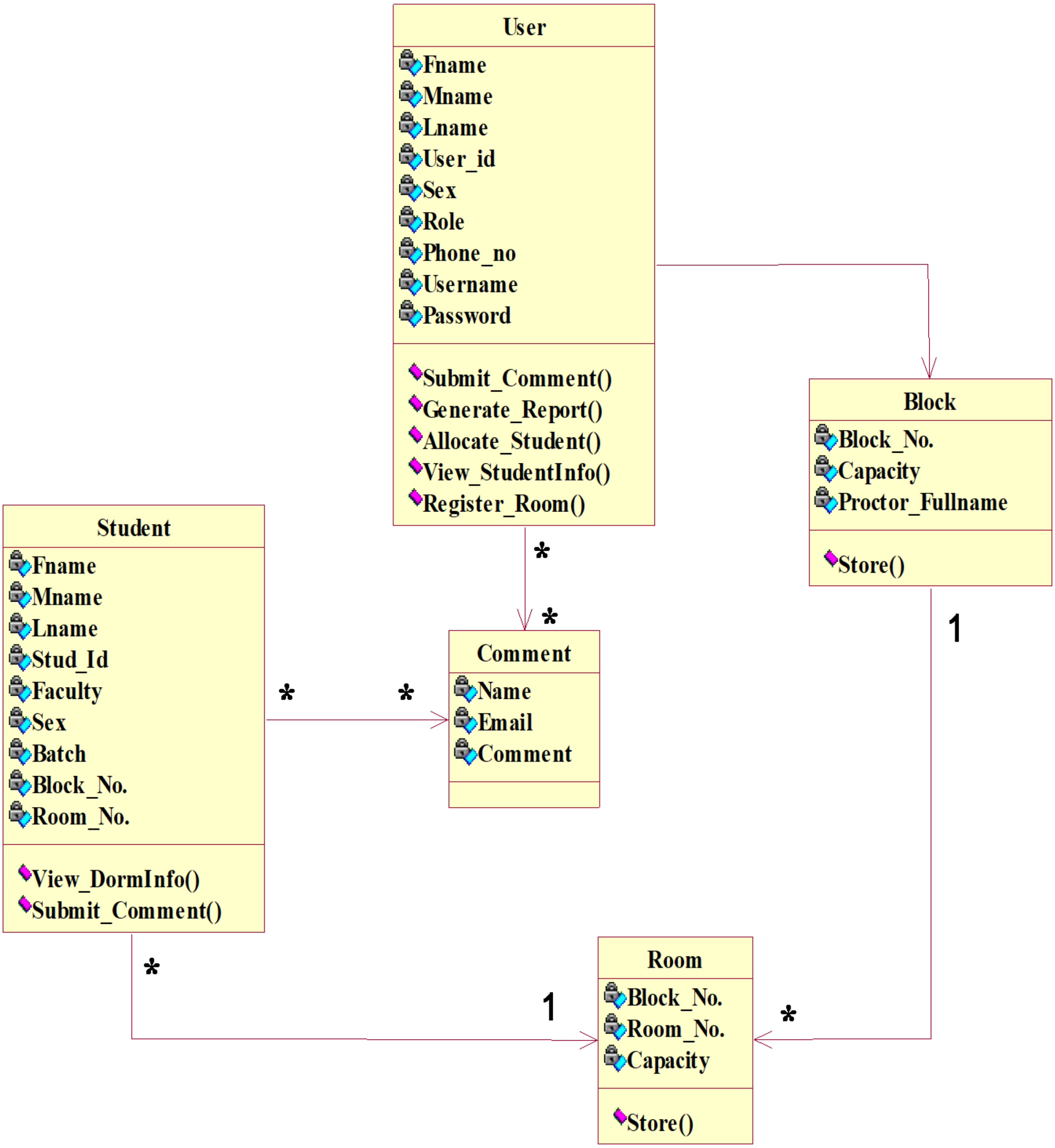

Fig . Analysis level of Class Diagram

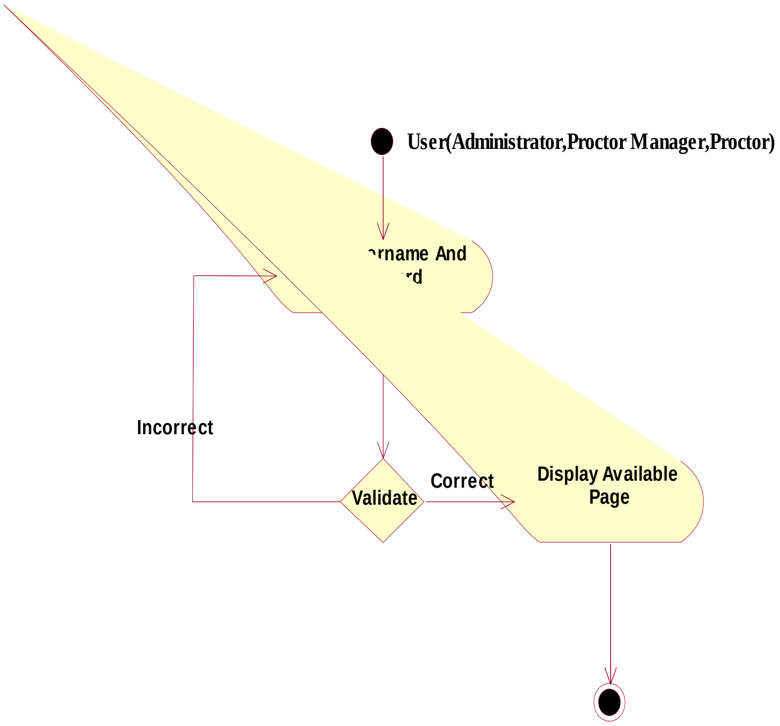

Fig 1.1 Activity diagram for Login

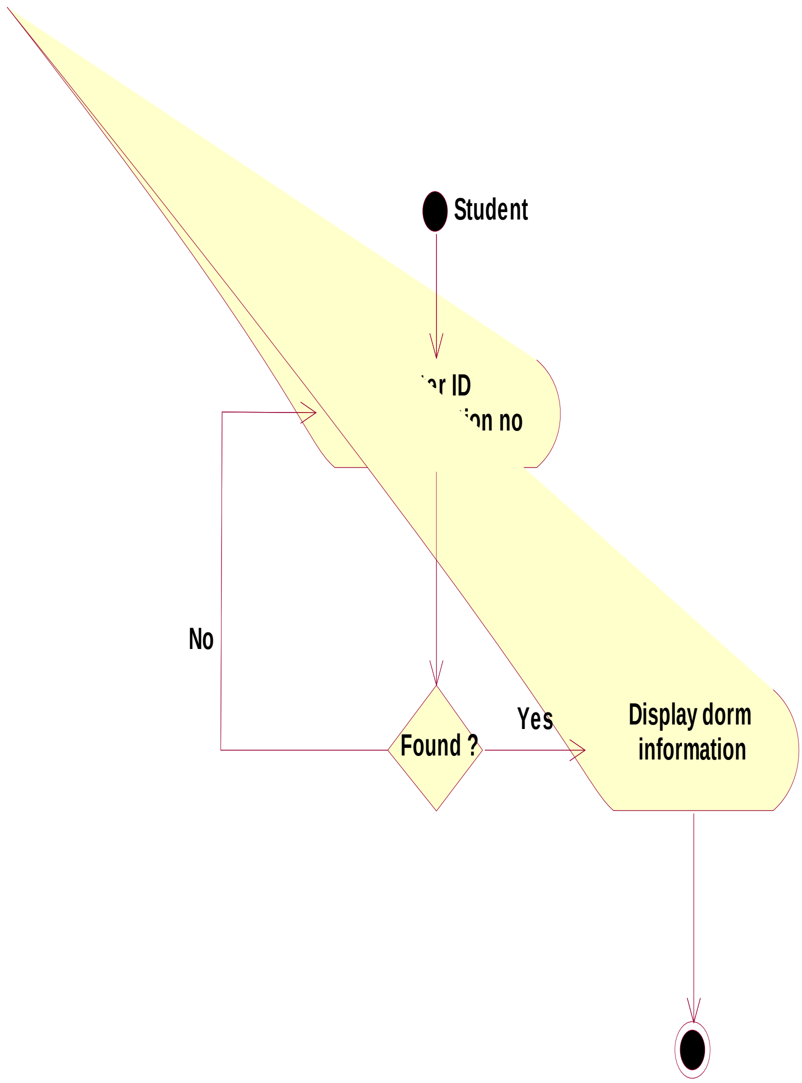

Fig 1.2 Activity diagram for View DormInfo

2. *System Design*

The System Design phase was a pivotal stage in the development of the Dormitory Management System, transforming abstract requirements into a concrete blueprint for implementation. This phase required careful consideration of both the technical architecture and the user experience, ensuring that the final product would be robust, scalable, and intuitive.

Architectural Decisions

At the heart of the DMS is a client-server architecture, chosen for its reliability and scalability. The system is web-based, allowing users to access it from any device with a browser, whether on campus or remotely. The backend is built using PHP, a language well-suited for rapid web development and widely supported in academic environments. MySQL was selected as the database engine, providing a stable and efficient means of storing and retrieving the vast array of data generated by dormitory operations.

Database Design

Designing the database was a meticulous process. I began by identifying the core entities: users, students, rooms, blocks, and assignments. Each entity was mapped to a table, with fields chosen to capture all necessary information. For example, the users table stores credentials, roles, and contact details, while the students table records personal data and dormitory assignments. Relationships between tables were established using foreign keys, ensuring data integrity and enabling complex queries—such as finding all students assigned to a particular block or room.

Entity-Relationship (ER) diagrams were created to visualize these connections, serving as a guide during both development and future maintenance. Special attention was paid to normalization, reducing redundancy and making the database easier to update and expand.

User Interface and Experience

The user interface was designed with accessibility and clarity in mind. Wireframes were sketched for each page, from the login screen to the admin dashboard and assignment forms. The color palette reflected the university's branding, creating a sense of familiarity and trust. Navigation was streamlined, with clear menus and dashboard cards guiding users to key features.

Responsive design was a priority, ensuring that the system would function seamlessly on desktops, tablets, and smartphones. CSS Flexbox and media queries were employed to adapt layouts to different screen sizes, while interactive elements were enhanced with JavaScript for validation and feedback.

Security Considerations

Security was woven into every aspect of the design. Passwords are hashed before storage, and sessions are managed to prevent unauthorized access. Input validation is performed both client-side and server-side, guarding against common vulnerabilities such as SQL injection and cross-site scripting. User roles and permissions are strictly enforced, ensuring that only authorized individuals can perform sensitive operations like assigning rooms or managing accounts.

Modularity and Maintainability

The system was designed to be modular, with separate PHP files handling distinct features. This approach simplifies maintenance and future enhancements, allowing new modules—such as reporting or notifications—to be added without disrupting existing functionality. Code comments and documentation were prioritized, making it easier for future developers to understand and extend the system.

3. Implementation

Development began with setting up the environment using XAMPP and VS Code. The code was organized into separate PHP files for each feature, with CSS and JavaScript files for styling and interactivity. The database schema was implemented according to the design, and initial data was loaded for testing.

Key features were developed iteratively. The login system was built first, followed by dashboards for each user role. The core logic for assigning students to rooms was implemented with careful attention to validation and error handling. Forms for room and block registration, as well as user account management, were created with both client-side and server-side validation.

Throughout development, the team adhered to coding standards and best practices, ensuring the code was modular, maintainable, and well-documented.

Key Objectives

- **Automate Dorm Assignment:** Simplifies the process of assigning students to available rooms based on criteria such as sex, batch, faculty, and room capacity.

- **Centralized Management:** Offers a single platform for managing all dormitory-related data, reducing paperwork and manual errors.
- **Role-Based Access:** Ensures that only authorized users (admins, proctors, managers, students) can access specific features and data.
- **User-Friendly Interface:** Provides easy-to-use forms and dashboards for all user roles.
- **Transparency:** Allows students to view their dorm assignments and related information online.
- **Scalability:** Designed to accommodate growing numbers of students, rooms, and administrative staff.

Typical Users

- **Administrators:** Oversee the entire system, manage user accounts, and handle high-level dormitory operations.
- **Proctors/Managers:** Assign students to rooms, register new rooms/blocks, and manage day-to-day dormitory activities.
- **Students:** View their dormitory placement and related information.

Main Functionalities

- Secure login and authentication for all users.
- Creation and management of user accounts.
- Registration of new dormitory blocks and rooms.
- Assignment of students to rooms based on eligibility and availability.
- Viewing and exporting dormitory assignment data.
- Responsive design for access on desktops, tablets, and mobile devices.

Benefits

- **Efficiency:** Reduces time and effort required for dormitory management.
- **Accuracy:** Minimizes assignment errors and duplicate records.
- **Accessibility:** Enables users to access dormitory information from anywhere.
- **Security:** Protects sensitive data through authentication and access control.

Folder Structure

dms/

│

```
├── about.php          # About Us page
├── admin.php          # Admin dashboard
├── assign.php         # Assign students to rooms
├── connection.php     # Database connection settings
├── cua.php            # Create User Account page
├── index.php          # Home page
├── login.php          # User login page
├── pro_manager.php    # Proctor manager dashboard
├── proctor.php        # Proctor dashboard
├── registerrooms.php  # Register rooms and blocks
├── viewdorm.php       # View dormitory assignment
├── style.css, *.css   # Stylesheets
├── aa.js, *.js        # JavaScript files
├── img/               # Images and icons
├── DATABASE/dbudms.sql  # Database schema and initial data
└── ...                # Other PHP files and assets
```

4. Testing

Testing was conducted at multiple levels. Functional tests verified that each feature worked as intended, from login to room assignment. Usability tests ensured that users could navigate the system easily and that the interface was clear and responsive. Security tests checked for vulnerabilities such as SQL injection and session hijacking.

Test cases were documented, and any bugs found were promptly fixed. The system was demoed to stakeholders for user acceptance testing, and their feedback was incorporated into the final version.

5. Deployment

Once testing was complete, the system was deployed to the university's web server. The database was imported, and configuration files were updated with production credentials. The team verified that all features worked in the live environment and provided training and documentation to users.

A go-live checklist was followed to ensure a smooth launch, including backups, permissions, and final verification.

6. Maintenance

After deployment, the system entered the maintenance phase. Regular backups were scheduled, and the team monitored for bugs and performance issues. User feedback was collected for future enhancements, such as adding notifications, advanced reporting, and integration with other university systems.

Security updates and code optimizations were performed as needed, and unused files were removed to keep the project clean.

Conclusion

The Dormitory Management System project followed a structured SDLC approach, from initial requirements gathering to deployment and maintenance. The result is a robust, secure, and user-friendly system that meets the needs of Salale University's students and staff. The project demonstrates the importance of thorough analysis, careful design, disciplined implementation, and ongoing maintenance in delivering successful software solutions.