A blue circle with white text

Description automatically generated

**COMSATS UNIVERSITY ISLAMABAD, ABBOTTABAD**

**Software Design Document of Hostel management system**

***Submitted by:***

Laiba Binte Tahir FA21-BSE-019

Arfah Ali FA21-BSE-080

***Submitted to:***

Mam Neeli Khan

Contents

[**1.** **Project Context** 3](#_Toc169381572)

[**2. Architecture Requirements** 4](#_Toc169381573)

[**2.1 Overview of Key Objectives** 4](#_Toc169381574)

[**2.2 Architecture Use Cases** 4](#_Toc169381575)

[**2.3 Stakeholder Architectural Requirements** 6](#_Toc169381576)

[**2.4 Constraints** 6](#_Toc169381577)

[**2.5 Non-functional Requirements** 6](#_Toc169381578)

[**2.6 Risks** 7](#_Toc169381579)

[**3. Solution** 8](#_Toc169381580)

[**3.1 Relevant Architectural Patterns** 8](#_Toc169381581)

[**3.2 Architecture Overview** 8](#_Toc169381582)

[**High-Level Architecture:** 8](#_Toc169381583)

[**3.3 Structural Views** 9](#_Toc169381584)

[**Diagrams:** 10](#_Toc169381585)

[**3.4 Behavioral Views** 14](#_Toc169381586)

[**Diagrams:** 14](#_Toc169381587)

[**3.5 Implementation Issues** 17](#_Toc169381588)

[**4. Architecture Analysis** 18](#_Toc169381589)

[**4.1 Scenario Analysis** 18](#_Toc169381590)

[**4.2 Risks** 18](#_Toc169381591)

[**5. Appendices** 20](#_Toc169381592)

[**5.1 DFD level 0 & 1** 20](#_Toc169381593)

[**5.2 DFD level 2** 21](#_Toc169381594)

Software Design Document (SWDD) - Hostel Management System

# **Project Context**

The goal of the Hostel Management System (HMS) project is to improve and simplify the way that educational institutions' hostel facilities are managed. Conventional hostel management techniques are frequently laborious, time-consuming, and error-prone. These difficulties stem from the manual processing of staff management, student records, room assignments, incident reports, and feedback gathering. By offering a complete, automated solution that boosts user satisfaction, increases operational effectiveness, and guarantees data security, the HMS seeks to address these problems.

The HMS will be made to meet the needs of several parties, including as employees, administrators, wardens, and students. The automation of crucial tasks including room assignment, staff and student management, incident reporting, and menu feedback will not only improve operational efficiency but also yield real-time reports and insights.

# **2. Architecture Requirements**

## **2.1 Overview of Key Objectives**

The primary objective of the Hostel Management System is to streamline the management of hostel facilities. The system aims to automate room allocation, student management, staff management, incident reporting, and feedback and menu management to enhance operational efficiency, user experience, and data security.

## **2.2 Architecture Use Cases**

* **Room allocation:** It is used for the automation in allocating room to the students, viewing the details of a particular room, creating the fee structures and cancellation of room allocations.
* **Student Management:** It shall deal with the management of student's records by adding it, viewing it, modifying a record and deleting a student record.
* **Staff Management:** This allows the admin to store the information of wardens, security personnel, and cooks.
* **Incident Reporting:** There shall have to be an incident-reporting system for the maintenance of order in the hostel.
* **Feedback and Menu Management:** This would take the feedback from the student to manage the menu of the hostel.

A diagram of a hotel management system

Description automatically generated

Figure 1 Use Case Diagram

## **2.3 Stakeholder Architectural Requirements**

* **Developers:** The detailed design specifications and guidance on coding.
* **Project Managers:** A lot of To understand the design so that stays on track with respect to the defined requirements.
* **Wardens or Administrators:** Clearly view the architecture and functionalities of the entire system. They will host/ manage the running of Hostel.
* **Students:** Able to look at personal records, Room allocation, submit their feedback, and View men.
* **Staff [Mess Staff, Security Personnel]:** Avg. Specific interfaces will do their specific Job. Update menu and even report incidence.
* **Quality Assurance Team:** System design in order to conduct proper testing and validation.

## **2.4 Constraints**

* **Data Privacy:** This system has to stick or comply fully with strict data privacy laws and regulations, such as GDPR, to handle all the valuable data of the user in a proper and secure way.
* **Budget Limitations:** It has to be completed within the availability of funds, and hence, the technologies and tools have to be cost- effective.
* Time Constraints: It should develop and implement the system within the given timeline that may be for an academic or institutional deadline.
* **Availability of Resources:** The project is limited by the availability of resources for development and testing, including personnel with the necessary expertise.
* **Legacy System Integration:** The system shall be able to communicate with other current legacy systems that may have limitations and/or constraints in data exchange and compatibility.

## **2.5 Non-functional Requirements**

* **Scalability:** Due to the hostel student and staff population rising with time, especially during the beginning of a new academic term, these systems need to accommodate an increase in both the number of users and records.
* **Reliability:** The system must be highly available; it must also be fault-tolerant with very minimal downtime, especially in critical operations such as room allocation, fee generation, and incident reports.
* **Performance:** The system shall be efficient in its processing and response time to every operation it is submitted, be it for room allocation, update of student records, or case reporting, even when it is under peak load.
* **Security:** Sensitive information like the students' personal data, data related to the staff, and their records shall be secured with the help of a sound security system that prevents any unauthorized access and data breach.
* **User Experience**: It acts as a front end which depicts most straightforward and efficient dealing, thus providing the best experience to the user; any user can use the interface very easily, so it will be easy for students to log in and update their records; even warden and staff will easily submit their experience about living in this hostel.

## **2.6 Risks**

* **Data Breaches:** The threat of a data breach is possible if security is not implemented properly and sensitive information regarding staff and students may be exposed.
* **System Downtime:** Unplanned system downtime is especially expensive in operational efficiency terms when it happens at the most critical period of a term, allocating room space, or generating fees.
* **Scalability Issues:** The software may end up slowing down or stop working when too much load is placed. It could be because of peak registration periods or even accessing of the system by too many users simultaneously.
* **Functional Bugs:** System functional bugs/ errors may affect some of the important functionalities such as room assignments, incident reporting, or user management, resulting in operational inefficiency and user dissatisfaction.
* **Compliance Failures:** It may not comply with data protection regulations, leading to consequent litigation and loss of trust among users or other stakeholders.

# **3. Solution**

## **3.1 Relevant Architectural Patterns**

The Hostel Management System adopts a Microservices Architecture to ensure modularity, scalability, and fault isolation. Each subsystem is developed and deployed independently, enabling flexibility and efficient resource utilization.

## **3.2 Architecture Overview**

### **High-Level Architecture:**

1. **Client Layer:**

* Web client: React and angular Js for a browser-based application
* Mobile application: android and Ios using React Native and Flutter

1. **API gateway:**

* Single entry point for all the client's requests, routing them to respective service. It also handles cross-cutting concerns

1. **Microservices layer**

* User management service
* Room management service
* Student management service
* Staff management service
* Incident management service
* Feedback management service

1. **Database layer**

* MySQL database for the services

1. **Authentication & authorization**

* JWT and O Auth 2.0 for centralized service handling user authentication
* User dictionary for storage of related data

1. **Inter-service communication**

* Using rest full APIS for request-response interactions in synchronous communication. A diagram of a application

  Description automatically generated

Figure 1 Architecture Diagram of HMS

* For Asynchronous messages we are using brokers and for event-driven communication Apache Kafka.

## **3.3 Structural Views**

* **Client Layer:**

Web client: React and angular Js for a browser-based application

Mobile application: android and Ios using React Native and Flutter

* **API gateway:**

Single entry point for all the client's requests, routing them to respective service. It also handles cross-cutting concerns

* **Microservices layer**

Comprises ofUser management service, Room management service, Student management service, Staff management service, Incident management service, Feedback management service

* **Database layer**

UsingMySQL database for the services mentioned above

### **Diagrams:**

* **Package Diagram:** Illustrating the decomposition of the system into packages, showing the interconnections between major subsystems and data repositories.

A diagram of a computer program

Description automatically generated with medium confidence

Figure 3 Package Diagram

* **Component Diagram:** Describing the relationship between components with well-defined interfaces. Components typically comprise multiple classes.

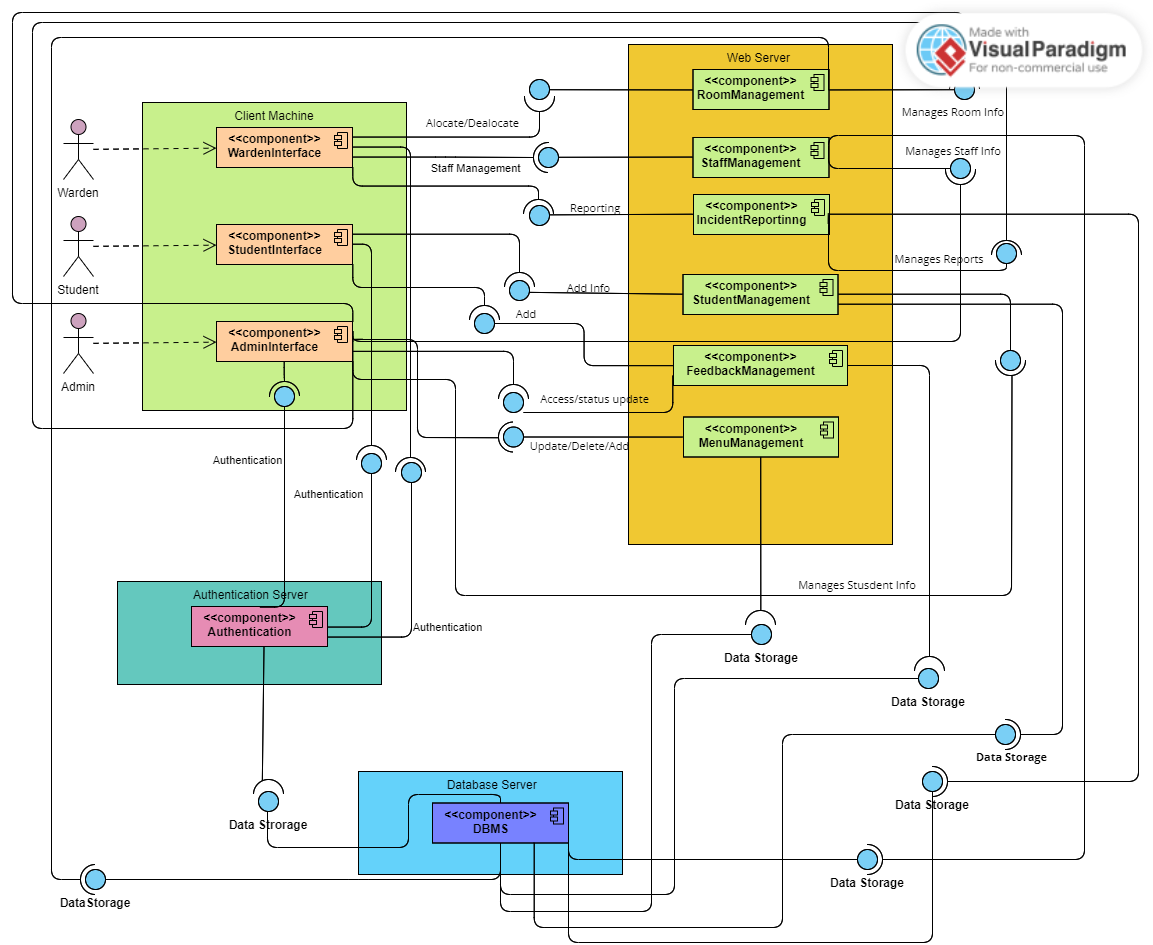


Figure 4 Component Diagram

A diagram of a company

Description automatically generated• **Composite Diagram:** The composite diagram for the Hostel Management System integrates these entities and their relationships into a cohesive structure.

Figure 5 Composite Diagram

**Deployment Diagram:** A diagram of a web server

Description automatically generated

Figure 2 Deployment Diagram

• **ER diagram:** Showing the Entities in the system and their relationships.

A diagram of a computer

Description automatically generated

Figure 6 ER Diagram

## **3.4 Behavioral Views**

* **Subsystem Interactions:** Showing how different services communicate and collaborate to provide seamless functionality to the users.

### **Diagrams:**

* **Sequence Diagram - Add Room:** Depicts the interaction between components for adding a room.

A diagram of a project

Description automatically generated

Figure 6 Sequence Diagram - Add Room

* **Sequence Diagram - Add Warden:** Depicts the interaction between components for adding a warden.

A diagram of a project

Description automatically generated

Figure 7 Sequence Diagram - add Warden

* **Activity diagrams:** Defining program logic and business processes.

A diagram of a computer program

Description automatically generated

Figure 8 Activity diagram (for student)

A diagram of a software project

Description automatically generated

Figure 9 Activity diagram for admin

## **3.5 Implementation Issues**

* **Inter-service communication:** there can be an issue in ensuring seamless communication b/w microservices to handle it use of standardized protocols and roust message brokers
* **Data consistency:** Maintaining data consistency across microservices can be handled by the implementation of distributed transactions or eventual consistency mechanisms.
* **Authentication & Authorization:** to manage the efficiency of authentication and authorization of users use centralized authentication services like O Auth 2.0
* **Monitoring and logging:** for effective implementation of monitoring and logging mechanism use comprehensive monitoring tools and logging systems.

# **4. Architecture Analysis**

## **4.1 Scenario Analysis**

* **Peak Registration period:** During the beginning of any system; the system experiences high traffic as the users logs in, reserve rooms, update profiles and interact with the system. This scenario evaluates the system's ability to handle multiple users and modules along with response time.
* **Incident Reporting: a scenario where multiple incidents are reported such as during an emergency or any major event in a hostel is reported simultaneously. This shows the reliability of the system and the efficiency of real-time data processing.**
* **Maintenance:** Regular maintenance is necessary for the smooth working of the system. This scenario shows how well the system can handle the downtime, data integrity during maintenance windows
* **Feedback submission:** students give feedback at the end of the semester of services. This shows the system's capability to handle a large volume of feedback of different services, categorize them correctly, and generate actionable insights.
* **Scalability Testing:** The scalability of the system is evaluated by gradually increasing the number of active users and data volume. This helps in understanding how to identify the bottleneck that needs to be addressed for future support and growth of the system.

## **4.2 Risks**

* **Data Breaches:** The threat of a data breach is possible if security is not implemented properly and sensitive information regarding staff and students may be exposed.
* **System Downtime:** Unplanned system downtime is especially expensive in operational efficiency terms when it happens at the most critical period of a term, allocating room space, or generating fees.
* **Scalability Issues:** The software may end up slowing down or stop working when too much load is placed. It could be because of peak registration periods or even accessing of the system by too many users simultaneously.
* **Functional Bugs:** System functional bugs/ errors may affect some of the important functionalities such as room assignments, incident reporting, or user management, resulting in operational inefficiency and user dissatisfaction.
* **Compliance Failures:** It may not comply with data protection regulations, leading to consequent litigation and loss of trust among users or other stakeholders.

# **5. Appendices**

# **5.1 DFD level 0 & 1**

A diagram of a software system

Description automatically generated

Figure 2 DFD Level 0 and 1

# **5.2 DFD level 2**

A diagram of a student

Description automatically generated

Figure 3 DFD Level 2