ABSTRACT

The **GateHost System** is a **web-based software** designed to **streamline and automate hostel operations**, focusing on **gate pass approval, student verification, real-time notifications, and efficient data management**. Developed using **Visual Studio** with **SQL Server Management Studio** as the database management system, the application ensures **secure data storage, fast processing, and seamless accessibility**. It provides a **centralized and efficient platform** for managing various hostel-related tasks while enhancing communication between **students, class tutors, Heads of Departments (HODs), and hostel wardens**.

A key feature of the system is the **Gate Pass Management Module**, which simplifies the process of **applying for and approving leave requests**. Students can submit **gate pass requests online**, which are then routed through a **hierarchical approval process**—first reviewed by the **class tutor**, then forwarded to the **HOD**, and finally **verified by the hostel warden**. Throughout this process, **real-time notifications** are sent to all relevant parties, ensuring **transparency and efficient communication**. Once approved, a **Hostel Verification Pass** is generated, containing **essential details such as student information, reason for leave, date, time, and a unique verification code (QR code or PDF format)**. This verification mechanism enhances **security at hostel exits**, prevents **unauthorized movement**, and ensures **efficient monitoring**.

It is a web-based platform designed to automate and streamline the day-to-day operations of hostel management, enhancing communication and efficiency between students, hostel staff, and academic personnel. The system integrates multiple functionalities into a single, user-friendly platform, addressing the needs of both students and administrators. A central feature of the system is the **Student Gate pass Management Module**, which simplifies the process of applying for and approving leave requests.

The **Gate pass Request Workflow** is designed to be efficient and transparent. When a student submits a gate pass request, it undergoes a hierarchical approval process. The request is first reviewed by the class tutor, then forwarded to the Head of Department (HOD), and finally to the hostel warden for verification. At each stage, real-time notifications are sent to all relevant parties, ensuring seamless communication and timely updates. This automated workflow eliminates delays and reduces the administrative burden associated with manual processes.

In addition to gate pass management, the system provides students with essential information such as **meal timings**, **hostel menus**, and **hostel IDs**, enabling them to stay informed without needing to contact the hostel administration. Students can also access and update their personal profiles, which include details such as name, class, department, and contact information.

For hostel staff and administrators, the system offers a robust **admin interface** with role-based access control. Tutors, HODs, and wardens are granted different levels of access to ensure that only authorized personnel can approve requests, manage student data, and access sensitive information. This feature enhances security and minimizes errors in the approval process.

1.INTRODUCTION

1.1 Overview of the Project

The GateHost System is a web-based application designed to streamline and automate hostel operations, particularly focusing on student gate pass approval, attendance tracking, and real-time communication between students and hostel authorities. The system is developed using Visual Studio as the primary development environment and SQL Server Management Studio as the database management system, ensuring secure data storage and efficient processing.

Traditionally, hostel management relies on manual paperwork, leading to inefficiencies such as delays in gate pass approvals, errors in student records, and difficulty in tracking student movement. This project aims to digitize and automate these processes, enhancing overall security, transparency, and operational efficiency.

Key Functionalities

- Gate Pass Management – Students can request a gate pass online, which is routed through a hierarchical approval system (Class Tutor → HOD → Warden). Upon approval, a Hostel Verification Pass is generated in QR code or PDF format for security checks.
- Attendance Tracking – Hostel staff can maintain real-time attendance records, ensuring that student presence is accurately logged.
- Room Allocation – The system automates room assignments, preventing overbooking and mismanagement.
- Notifications & Alerts – The system sends real-time notifications to students and hostel staff about approvals, announcements, and important updates.
- Student Information Management – Students can access and update their personal profiles, including hostel details, contact information, and emergency contacts.
- Role-Based Access Control – Different levels of access are assigned to Tutors, HODs, Wardens, and Administrators, ensuring data security and controlled access.
- Feedback & Complaint Handling – A feature allowing students to submit complaints regarding hostel facilities, which administrators can track and resolve efficiently.

By integrating these features, the GateHost System provides a secure, user-friendly, and efficient solution for managing hostel-related operations in educational institutions. The system reduces administrative workload, minimizes errors, and ensures a smooth experience for students, staff, and management.

1.2 Module Descriptions

The GateHost System (NASCH) consists of multiple modules, each designed to streamline specific hostel operations. These modules work together to enhance efficiency, security, and communication, reducing the administrative burden and improving the hostel experience for students and staff.

1.2.1 Gate Pass Management Module

This module facilitates a paperless and efficient system for students to apply for gate passes online. The requests follow a hierarchical approval workflow, passing through the Class Tutor → HOD → Hostel Warden for authorization. The system ensures real-time tracking and sends notifications to students and approvers via email or SMS. Once approved, the system generates a QR code or PDF-based verification pass, ensuring secure student exits and preventing unauthorized movement. This module eliminates delays and ensures a transparent leave approval process.

1.2.2 Student Attendance Management Module

The Student Attendance Management Module enables hostel authorities to automatically track student attendance, minimizing the need for manual record-keeping. It generates daily, weekly, and monthly attendance reports, helping hostel administrators monitor student presence effectively. The system also triggers automated alerts for irregular attendance patterns, ensuring student safety. This module can integrate with biometric or RFID-based tracking systems, improving accuracy and efficiency.

1.2.3 Room Allocation & Management Module

The Room Allocation & Management Module automates the assignment of hostel rooms based on availability, student preferences, and predefined rules. It prevents overbooking and conflicts, ensuring efficient room distribution. Hostel administrators can access real-time data on vacancies, current occupants, and upcoming room availability. Additionally, students can request room changes, which hostel authorities can approve or deny based on hostel policies. This module reduces manual intervention, enhances space utilization, and minimizes allocation errors.

1.2.4 Notification & Communication Module

The Notification & Communication Module ensures seamless interaction between students and hostel authorities. It automatically sends notifications via email, SMS, or in-app alerts for gate pass approvals, hostel announcements, emergency alerts, and policy updates. Students receive updates about meal timings, maintenance schedules, and hostel rules, keeping them informed at all times. This module reduces delays in communication, ensuring instant information delivery.

1.2.5 Profile Management Module

The Profile Management Module allows students to view and update their personal details, hostel information, and emergency contact numbers. Each student has a secure profile containing essential details such as name, class, department, room number, and contact information. This ensures that hostel authorities have accurate and up-to-date records, which helps in efficient administration and emergency response. The module incorporates secure authentication, preventing unauthorized access to student data.

1.2.6 Role-Based Access Control (RBAC) Module

The Role-Based Access Control (RBAC) Module enforces data security and controlled access by assigning different user roles within the system. The access levels include:

Students: Can submit gate pass requests and view personal information.

Class Tutors & HODs: Can review and approve gate pass requests.

Wardens & Administrators: Have broader control over approvals, student records, and hostel management settings.

The admin panel provides full control to authorized personnel, allowing them to configure user roles, update hostel policies, and manage database settings, ensuring data confidentiality and security.

1.2.7 Feedback & Complaint Handling Module

The Feedback & Complaint Handling Module provides students with a platform to submit complaints and feedback related to hostel facilities. Administrators can track, review, and resolve complaints efficiently, with real-time status updates provided to students. The system ensures transparency in handling student concerns and improves service quality. Additionally, students can submit anonymous feedback to freely express concerns about hostel management and facilities.

2. SYSTEM ANALYSIS

2.1 Existing System

The traditional GateHost System primarily relies on manual processes and paperwork, making daily operations cumbersome and inefficient. Students seeking gate pass approvals must physically visit multiple authorities, leading to delays, miscommunication, and frustration. Attendance tracking is handled through registers or outdated biometric systems, making it difficult to monitor student movement effectively. Room allocation is another major challenge, as manual assignment often results in overbooking, space mismanagement, and administrative errors. Additionally, communication between students and hostel authorities is slow and unstructured, as hostel notices and announcements are conveyed through physical notice boards or word-of-mouth, leading to missed updates and lack of clarity. The complaint management system is also inefficient, with students having to file complaints through paper forms or informal verbal communication, making it hard to track the resolution progress. Moreover, sensitive student data is often stored in unsecured records, making it prone to loss, tampering, or unauthorized access. These inefficiencies not only affect operational effectiveness but also compromise student satisfaction, security, and overall management transparency.

2.1.1 Disadvantages of the Existing System

The limitations of the existing manual system create several challenges for both hostel authorities and students:

Gate Pass Approval Process: Students must manually request approvals from tutors, HODs, and wardens, leading to long wait times and administrative burdens.

Inefficient Attendance Tracking: Manual registers and outdated biometric systems fail to provide real-time monitoring, making it hard to detect unauthorized movements.

Unstructured Room Allocation: Assigning hostel rooms manually often results in inefficient space utilization, overbooking, and lack of flexibility.

Delayed Notifications & Communication Gaps: Important announcements such as meal schedules, hostel rules, and emergency updates may not reach students in a timely manner.

Security Risks & Data Loss: Paper records and unprotected digital files pose risks of data breaches, document misplacement, and unauthorized access.

Ineffective Complaint Resolution: Hostel complaints often go unresolved due to the lack of a structured system to track and manage issues.

No Role-Based Access Control: All hostel personnel have unrestricted access to student records, leading to data privacy concerns.

2.2 Proposed System

The GateHost System (HMS) is a web-based, automated solution designed to eliminate inefficiencies in hostel operations and streamline daily administrative tasks through seamless digital integration. Developed using Visual Studio and SQL Server Management Studio, the system ensures secure data storage, real-time communication, and optimized workflow automation. By replacing manual, time-consuming processes with an intelligent, centralized platform, HMS enhances transparency, security, and operational efficiency within hostel management.

The system incorporates multiple automated modules that digitize and simplify essential hostel-related tasks, including Gate Pass Management, Attendance Tracking, Room Allocation, Notifications, Profile Management, and Complaint Resolution. The Gate Pass Management Module ensures a structured and secure approval process, allowing students to submit leave requests digitally, which are then processed through a hierarchical approval system involving the Class Tutor, HOD, and Hostel Warden. With real-time notifications at every stage, students receive instant updates on their requests, and upon approval, the system generates a QR code or PDF-based verification pass for authentication at hostel exits.

The Attendance Tracking Module automates student attendance monitoring using biometric authentication, RFID-based tracking, or digital logs, significantly improving accuracy and eliminating manual errors. The system generates real-time reports that help hostel authorities monitor attendance trends, detect irregular patterns, and enforce disciplinary actions if needed. Room Allocation & Management is also fully automated, preventing overcrowding and ensuring that rooms are allocated efficiently based on availability, student preferences, and predefined hostel policies. This module allows hostel staff to reassign rooms dynamically in cases of special requests, maintenance issues, or transfers, optimizing hostel space management.

To further enhance communication and operational efficiency, the Notification & Communication Module provides instant updates via SMS, email, and in-app alerts. Important announcements regarding meal schedules, hostel rules, security updates, and event notifications are sent out in real-time, ensuring students and hostel staff stay informed without delays. The Role-Based Access Control (RBAC) Module strengthens data security by implementing tiered authorization levels, ensuring that only authorized personnel can access and modify sensitive student information, thereby reducing the risks of data breaches or unauthorized alterations

Additionally, the Complaint & Feedback System offers students a structured platform to submit grievances or report issues related to hostel facilities, services, or security concerns. Administrators can review, categorize, and track complaints efficiently, with automated status updates keeping students informed about progress and resolutions, ultimately improving service quality and accountability. The system’s centralized database, powered by SQL Server Management Studio, ensures highly secure, structured, and easily accessible data storage, implementing encryption techniques, automated backups, and restricted access policies to prevent data loss or tampering.

2.2.1 Advantages of the Proposed System

The proposed GateHost System introduces a comprehensive, automated, and highly efficient approach to managing hostel operations, significantly improving administrative processes, security, and communication. One of its key advantages is the Automated Gate Pass Approval Workflow, which enables students to apply for gate passes online, eliminating the need for manual paperwork. The request seamlessly moves through an approval hierarchy, involving the Class Tutor, HOD, and Hostel Warden, ensuring a transparent and structured process. Real-time notifications are triggered at every stage, keeping all stakeholders informed, and upon approval, the system generates a QR code or PDF-based verification pass, enhancing security measures at hostel exits and reducing unauthorized movement.

Another vital enhancement is the Digital Attendance Tracking for Accuracy, which replaces traditional attendance registers with biometric authentication, RFID-based tracking, or automated logs, enabling real-time student monitoring. Hostel authorities receive daily, weekly, and monthly attendance reports, allowing them to identify irregular patterns, detect absences, and enforce disciplinary measures efficiently. The Intelligent Room Allocation & Management module further simplifies hostel operations by automatically assigning rooms based on availability, student preferences, and predefined hostel policies. This feature prevents overbooking, optimizes space utilization, and provides flexibility for special requests, maintenance, and room reallocations, significantly improving hostel living arrangements.

The system also enhances Real-Time Notifications & Instant Communication, ensuring that students and hostel staff remain well-informed. Critical updates such as meal schedules, hostel rules, announcements, emergency alerts, and gate pass approvals are instantly delivered via SMS, email, or in-app notifications, eliminating the delays associated with physical notice boards. Moreover, the implementation of Role-Based Access Control (RBAC) strengthens data security and access management. Different levels of authorization are assigned to students, tutors, HODs, wardens, and administrators, ensuring that sensitive student records are accessed and modified only by authorized personnel, thereby reducing risks related to data leaks and unauthorized alterations.

Additionally, the Streamlined Complaint & Feedback System empowers students to digitally submit grievances regarding hostel facilities, services, or security concerns. Administrators can review, categorize, and track these complaints efficiently, with the system automatically sending status updates to students, ensuring a structured resolution process that enhances accountability and service quality. Secure & Centralized Database Management is another critical advantage, as the system is powered by SQL Server Management Studio, ensuring that all student and hostel data is securely stored, systematically organized, and easily retrievable. By implementing encryption techniques, periodic backups, and restricted access controls, the system significantly reduces the risk of data loss, tampering, and unauthorized access.

The User-Friendly & Web-Based Interface is designed for seamless accessibility across devices, allowing students and hostel staff to interact with the platform through desktops, tablets, and mobile phones. Its intuitive navigation, responsive design, and visually appealing interface enhance the overall user experience, making it convenient for users to perform tasks efficiently. Finally, the system is highly scalable, supporting future expansion and additional functionalities. Features such as visitor management, lost-and-found tracking, hostel meal planning, and emergency alert systems can be easily integrated, making the Nasc Hoste la long-term, flexible, and robust solution for modern educational institutions. By addressing critical operational challenges, optimizing workflow efficiency, and enhancing security, this system sets a new standard for hostel management, offering a smart, technology-driven, and student-friendly approach

3. SYSTEM SPECIFICATION

The GateHost System(HMS) is a web-based, automated software designed to enhance the efficiency, transparency, and security of hostel operations. Developed using Visual Studio with SQL Server Management Studio (SSMS) for database management, the system ensures seamless accessibility, structured data storage, and real-time communication between students and hostel authorities. The HMS is designed to address various hostel-related tasks, including gate pass approvals, attendance tracking, room allocation, notifications, and complaint handling, making it a one-stop solution for modern hostel management.

With its scalable and modular architecture, the system can adapt to the growing needs of educational institutions, offering future integration capabilities for AI-driven analytics, IoT-based security enhancements, and visitor tracking. The system is platform-independent and can be accessed on desktops, laptops, tablets, and mobile devices, ensuring anytime, anywhere usability for students and administrators.

3.1 Software Specification

The GateHost System is built using modern software technologies to ensure high performance, security, and user-friendly operations.

Frontend Technologies:

HTML5, CSS3, JavaScript – For an interactive and responsive user interface.

Bootstrap – For a mobile-friendly and aesthetically pleasing UI.

AJAX & jQuery – For smooth page transitions and dynamic content updates.

Backend Technologies:

ASP.NET (C#) – A robust and secure framework for server-side development.

SQL Server Management Studio (SSMS) – For structured and efficient data management.

Entity Framework – For seamless data access and management.

Development Tools & Environment:

Visual Studio – The integrated development environment (IDE) used for coding, debugging, and testing.

IIS (Internet Information Services) – Web server for hosting the application.

API Integration – To enhance functionalities like automated notifications, student authentication, and external data processing.

JSON & XML – For secure and efficient data exchange between the client and the server.

3.2 Hardware Specification

For optimal performance and scalability, the GateHost System requires the following hardware configurations:

For Server Deployment:

Processor: Intel Core i5/i7 (or equivalent AMD Ryzen)

RAM: Minimum 8GB (16GB recommended for high user loads)

Storage: 250GB SSD (for faster data processing) or 1TB HDD (for larger data storage)

Operating System: Windows Server 2016 or later

Network: High-speed LAN/Wi-Fi for real-time data processing and communication

Security: Firewall protection and SSL encryption for secure connections

For Client-Side Access (Students & Administrators):

Processor: Intel Core i3/i5 or equivalent

RAM: Minimum 4GB (8GB recommended for seamless multitasking)

Storage: Minimum 100GB (SSD recommended for faster performance)

Operating System: Windows 10/11, macOS, or Linux

Browser Compatibility: Google Chrome, Mozilla Firefox, Microsoft Edge (latest versions)

Display Resolution: 1366x768 (Full HD recommended for better UI experience)

3.3 About the Software

The GateHost System is designed to digitize and streamline hostel operations, offering a centralized, automated, and paperless solution for educational institutions. The secure and structured approach eliminates manual inefficiencies, reducing paperwork, delays, and human errors.

Key Features of the Software:Web-Based System – Accessible from any device via a secure web browser.

Role-Based Authentication – Ensures different access levels for students, tutors, HODs, wardens, and administrators.

Automated Approval Workflows – Enhances the speed and accuracy of gate pass approvals, attendance tracking, and room management.

Secure Data Storage – SQL Server ensures structured and encrypted storage of student records.

Real-Time Notifications – Instant alerts via email, SMS, or in-app messages keep users informed.

Complaint & Feedback Module – Provides a streamlined way for students to report hostel issues, ensuring quick resolution.

Scalability & Future Expansion – The system is future-proof, allowing for advanced integrations like AI-based analytics, biometric authentication, IoT-based security, and mobile app extensions.

Advantages of the Software:

- Enhances operational efficiency by reducing manual workload.
- Improves security with QR-based gate passes and automated student verification.
- Minimizes approval delays with real-time tracking and notifications.
- Provides insightful reports on student attendance, complaints, and hostel activitie.
- Ensures compliance with institutional hostel policies and government regulations.

4. SYSTEM DESIGN

The System Design phase is a crucial step in the development of the GateHost System(GHS). It involves defining the system's architecture, data structure, process flow, and user interactions. The goal of system design is to create a structured and efficient solution that ensures seamless user experience, data security, and optimized performance.

4.1 Input Design

1. Student Registration Form

The student registration form is designed to collect essential details from students for system enrollment. It includes fields such as full name, date of birth, mobile number, address, email/username, and password. Students can also upload their photo for identification. Additional details like class, department, year of study, and block & room number are required to complete the registration. The form ensures data accuracy through validation checks, including mandatory fields, proper email format, strong password enforcement, and numerical validation for mobile numbers. A submit button allows students to finalize their registration.

2. Out-Pass Request Form

The out-pass request form allows students to apply for temporary leave from the hostel. The form includes fields such as student name, department, class, parent contact number, block, room number, reason for out-pass, start date and time, and end date and time. Some fields, like student name and department, are auto-filled for convenience. A calendar picker helps in selecting dates, while a time selector allows students to specify their out-pass duration. The form ensures proper validation, such as date constraints and mandatory fields, before submission.

3. Attendance Management Form

The attendance management form is designed for hostel administrators to track student presence. It includes fields such as student ID, name, department, date, and attendance status (Present/Absent). Administrators can mark attendance using checkboxes or dropdown selections. A date picker is included for selecting the attendance date. The form ensures validation by preventing duplicate entries for the same student on the same day and allows exporting records for documentation.

4. Mess Registration Form

The mess registration form is used to enroll students for hostel meal plans. It includes input fields like student name, department, room number, meal preferences (Veg/Non-Veg), and special dietary requirements. Students can opt for meal plans such as monthly, weekly, or daily subscriptions through a dropdown selection. The form includes validation to ensure students do not register multiple times within the same meal cycle and provides a confirmation message upon successful registration.

5. HoD Creation Form

The HoD (Head of Department) creation form allows administrators to register department heads in the system. It includes fields such as name, department, email, contact number, and login credentials. A password field ensures security, and email validation is applied to prevent incorrect entries. The form ensures that only one HoD is assigned per department, preventing duplicate entries. Once submitted, the HoD gains access to student verification and approval features.

6. Student Verification Form

The student verification form is designed for hostel authorities to approve registered students before granting full system access. It includes fields like student ID, name, department, and registration status. Administrators can verify details and approve or reject student applications. A search feature allows quick lookup of student records, and validation ensures only verified students gain hostel privileges such as mess access and out-pass requests.

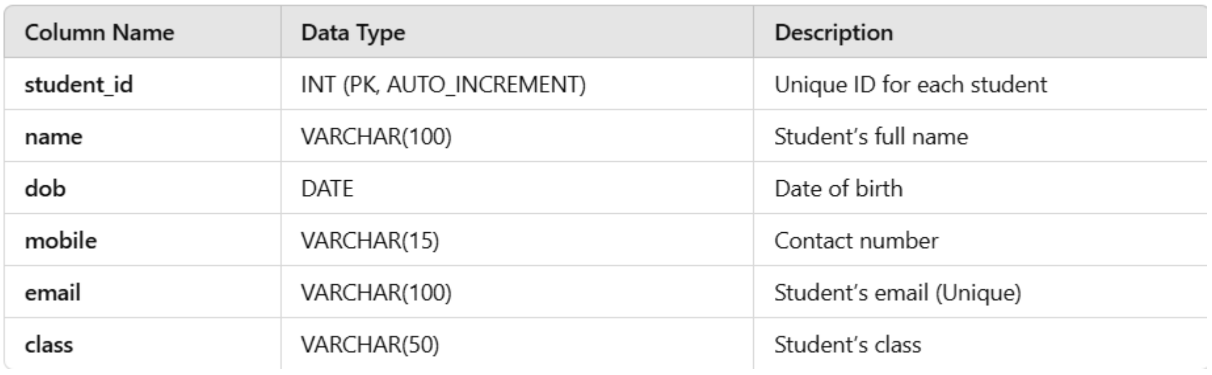
7. Department Management Form

The department management form enables administrators to manage hostel-associated departments. It includes fields for department name, department code, and head of department assignment. A dropdown selection allows assigning a previously registered HoD to a department. The form ensures validation by preventing duplicate department entries and allows updating or removing existing departments as needed.

4.2 Database Design

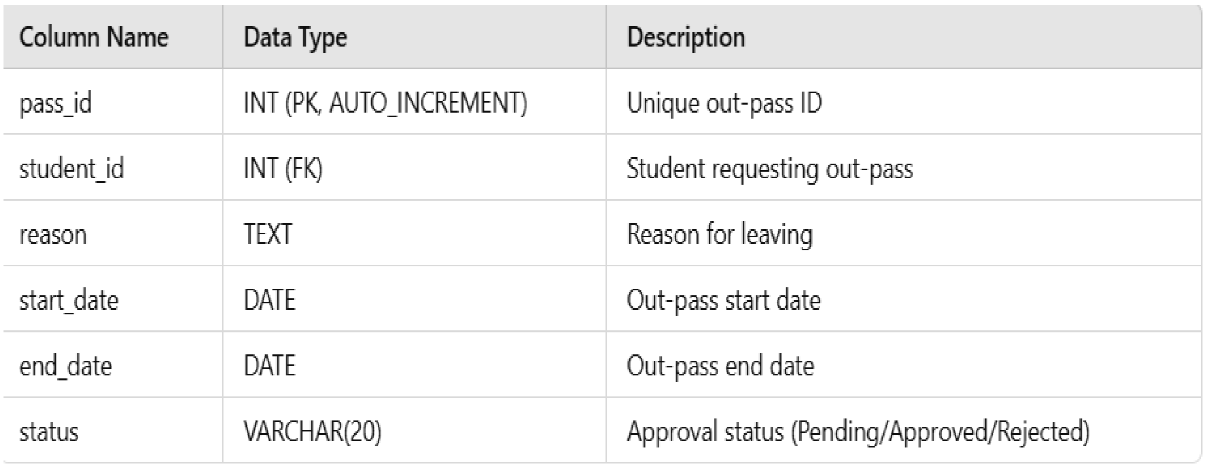
The Database Design defines how data is stored, retrieved, and managed in SQL Server Management Studio (SSMS). The database follows a structured, relational approach to ensure efficient data management and security.

Students Table



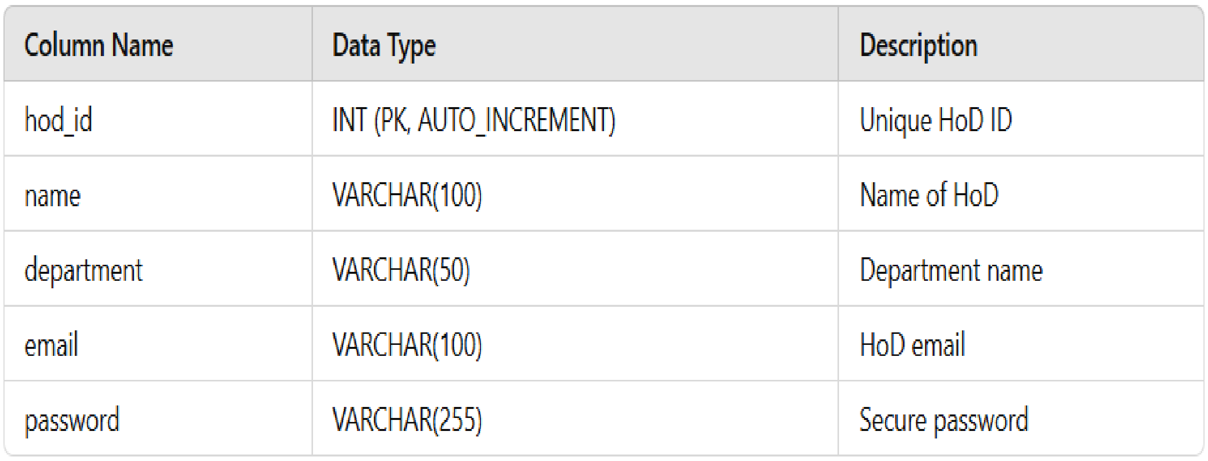
Column Name	Data Type	Description
student_id	INT (PK, AUTO_INCREMENT)	Unique ID for each student
name	VARCHAR(100)	Student’s full name
dob	DATE	Date of birth
mobile	VARCHAR(15)	Contact number
email	VARCHAR(100)	Student’s email (Unique)
class	VARCHAR(50)	Student’s class

Out-Pass Table



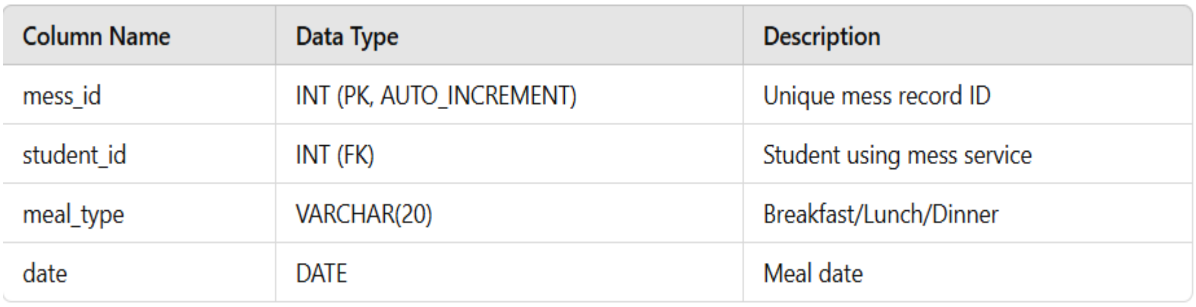
Column Name	Data Type	Description
pass_id	INT (PK, AUTO_INCREMENT)	Unique out-pass ID
student_id	INT (FK)	Student requesting out-pass
reason	TEXT	Reason for leaving
start_date	DATE	Out-pass start date
end_date	DATE	Out-pass end date
status	VARCHAR(20)	Approval status (Pending/Approved/Rejected)

HoD Creation Table



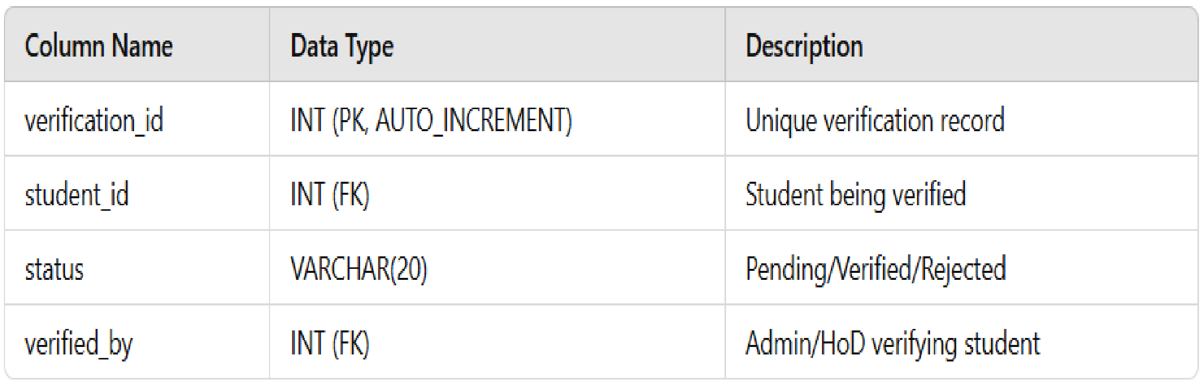
Column Name	Data Type	Description
hod_id	INT (PK, AUTO_INCREMENT)	Unique HoD ID
name	VARCHAR(100)	Name of HoD
department	VARCHAR(50)	Department name
email	VARCHAR(100)	HoD email
password	VARCHAR(255)	Secure password

Mess Table



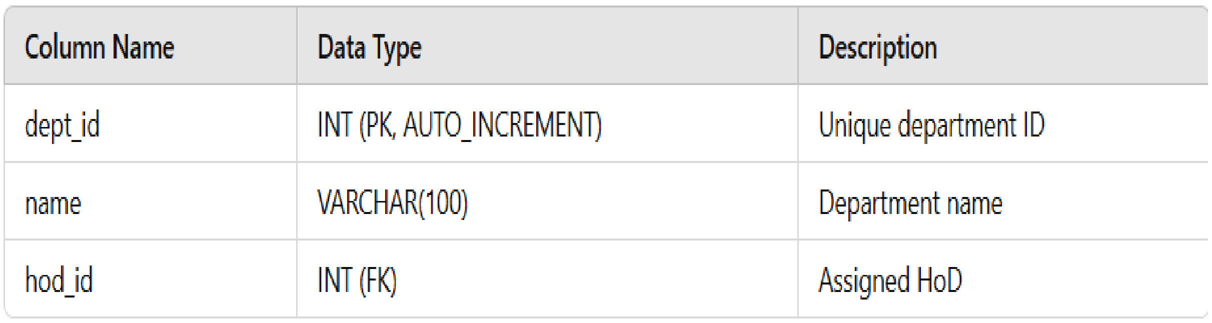
Column Name	Data Type	Description
mess_id	INT (PK, AUTO_INCREMENT)	Unique mess record ID
student_id	INT (FK)	Student using mess service
meal_type	VARCHAR(20)	Breakfast/Lunch/Dinner
date	DATE	Meal date

Student Verification Table



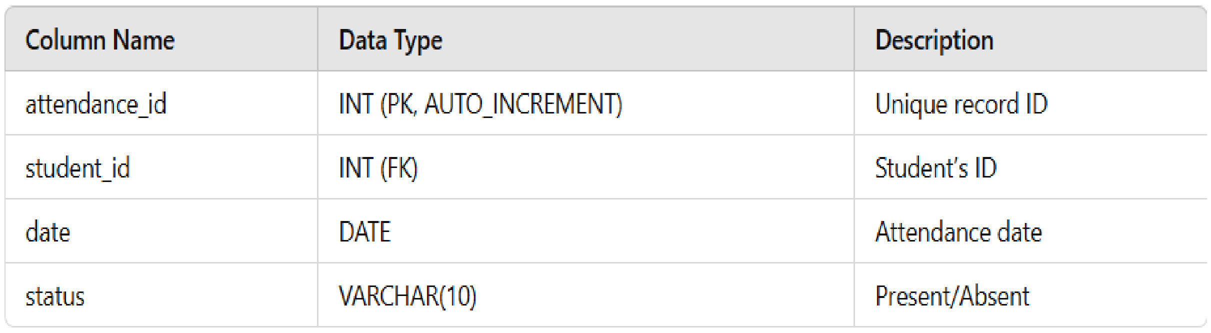
Column Name	Data Type	Description
verification_id	INT (PK, AUTO_INCREMENT)	Unique verification record
student_id	INT (FK)	Student being verified
status	VARCHAR(20)	Pending/Verified/Rejected
verified_by	INT (FK)	Admin/HoD verifying student

Departments Table



Column Name	Data Type	Description
dept_id	INT (PK, AUTO_INCREMENT)	Unique department ID
name	VARCHAR(100)	Department name
hod_id	INT (FK)	Assigned HoD

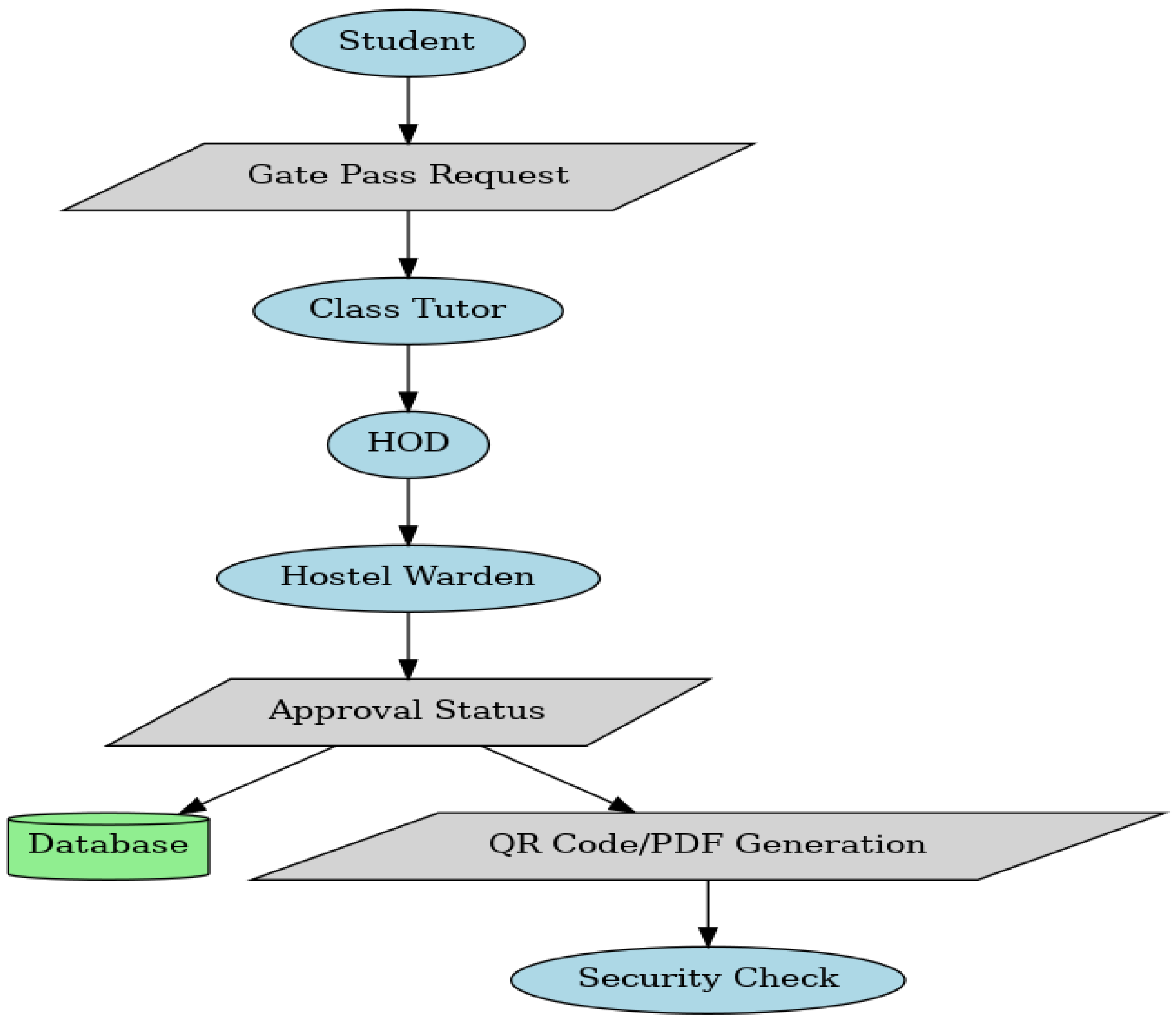
Attendance Table



Column Name	Data Type	Description
attendance_id	INT (PK, AUTO_INCREMENT)	Unique record ID
student_id	INT (FK)	Student's ID
date	DATE	Attendance date
status	VARCHAR(10)	Present/Absent

4.3 System Flow Diagram

The System Flow Diagram provides a high-level representation of the logical flow of processes in the HMS. It illustrates the sequence of operations from input to processing and final output.

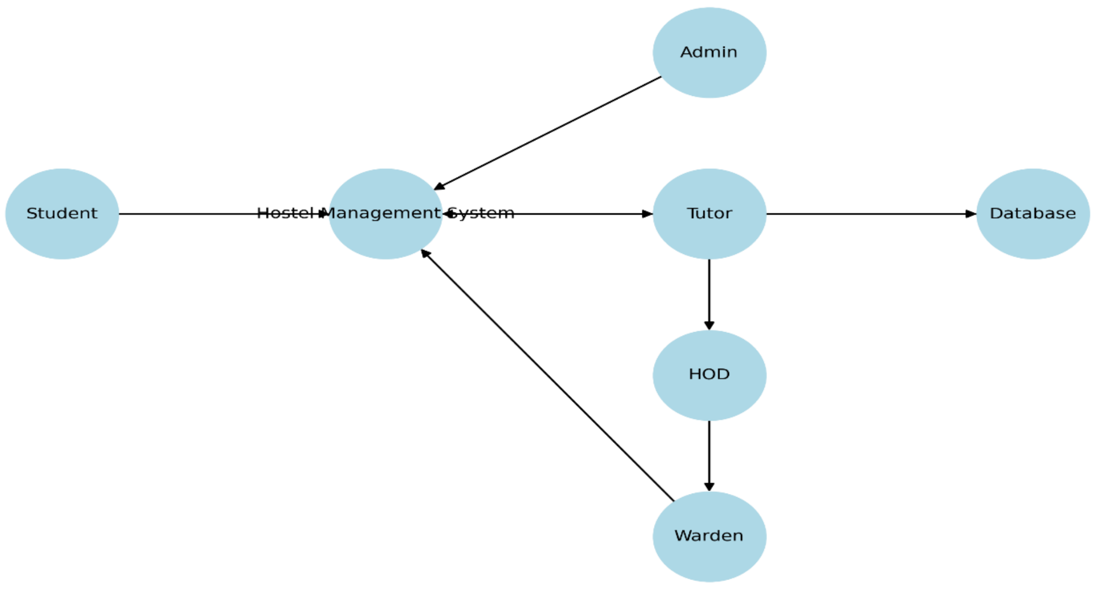


4.4 Data Flow Diagram (DFD)

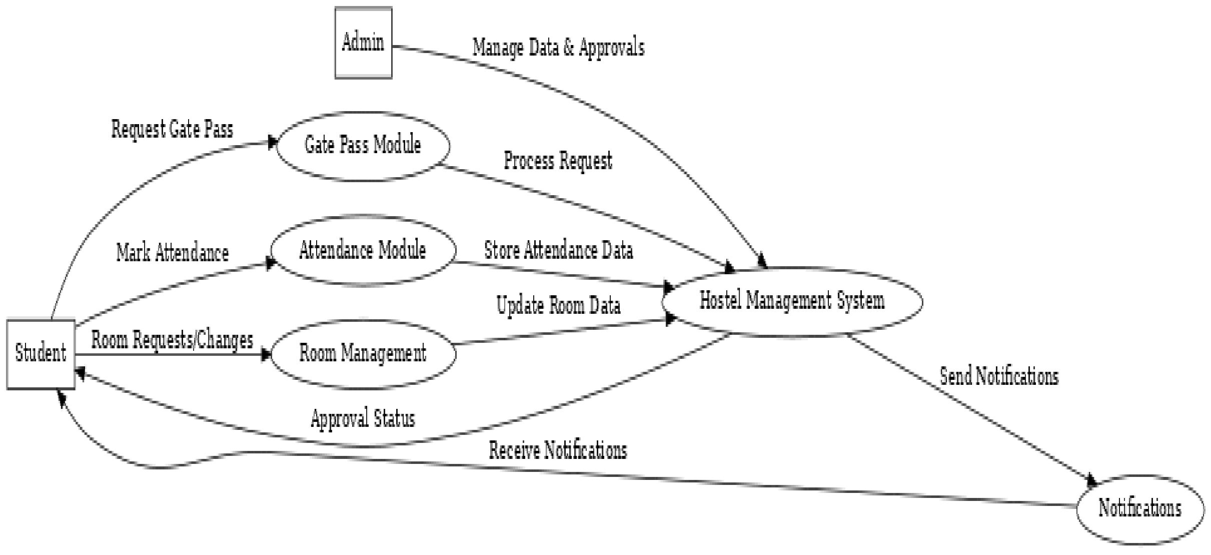
The Data Flow Diagram (DFD) visually represents how data moves within the Hostel Management System. It shows the interactions between users, processes, and data storage components.

Levels of DFD:

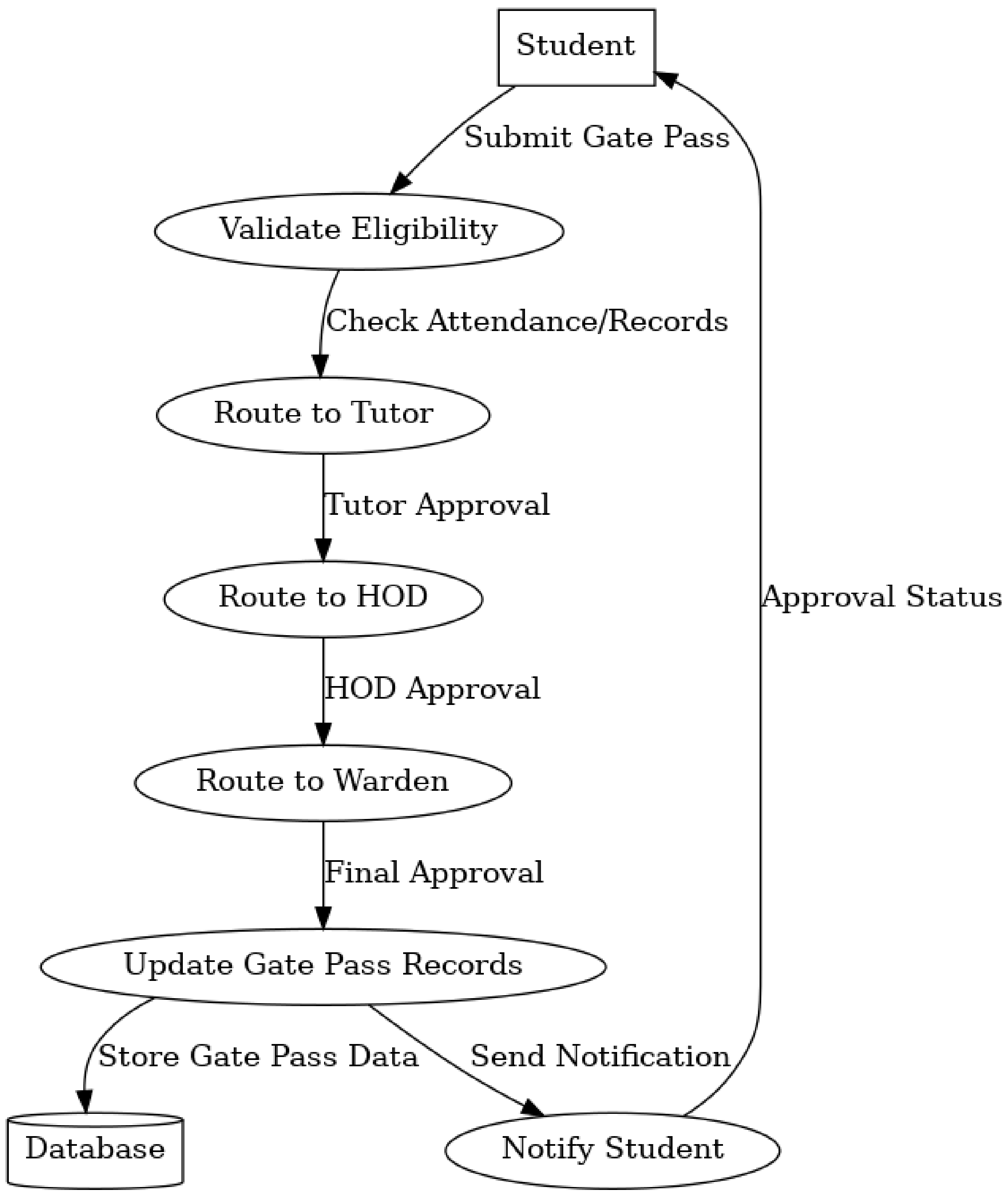
✓ DFD Level 0 (Context Diagram):



DFD Level 1 (Process Breakdown)

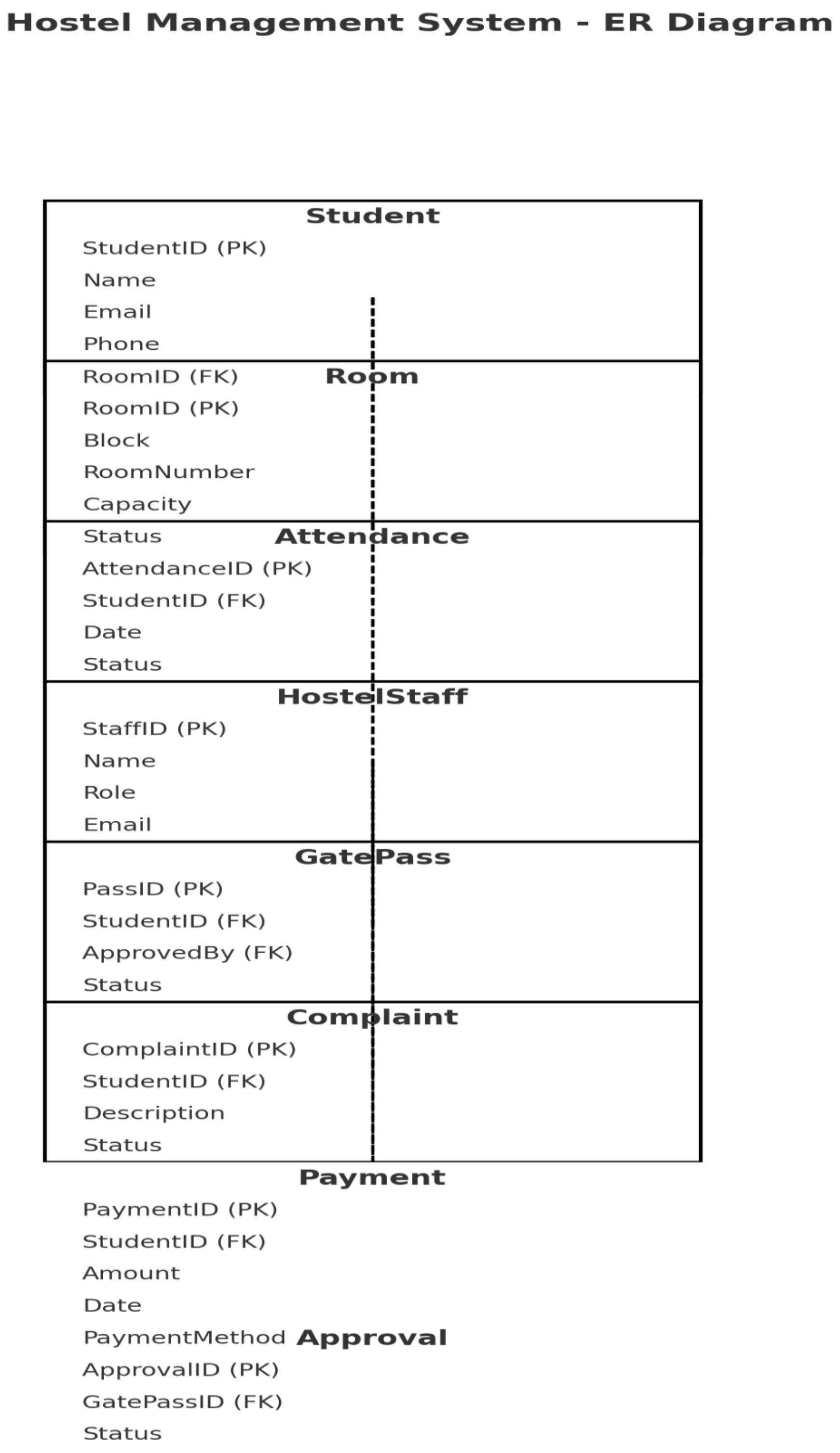


DFD Level 2 (Detailed Flow):



4.5 Entity-Relationship (ER) Diagram

The ER Diagram defines the relationships between entities (tables) in the database. It provides a graphical representation of how data is structured and interconnected.



4.6 Output Design

The Output Design focuses on how processed data is presented to users. The system generates reports, notifications, and confirmations to enhance decision-making and communication.

Key Features of Output Design in GateHost System

1. **Real-Time Notifications** – Instant alerts for gate pass approvals, complaint resolutions, and hostel announcements.
2. **Student Registration Report** – Maintains a structured record of student enrollments for easy reference.
3. **Hostel Registration Report** – Managed by wardens, this report tracks hostel allocations and student details.
4. **Gate Pass Management** – Generates gate passes, verified by both the HOD and warden before approval.
5. **Mess Pass System** – Facilitates meal tracking and ensures proper mess usage for students.
6. **Room Allocation Dashboard** – Displays real-time hostel occupancy, vacant rooms, and pending room change requests.
7. **Attendance Reports** – Provides daily, weekly, and monthly attendance summaries for hostel authorities.
8. **Complaint Tracking System** – Generates reports on complaint statuses and resolution timelines for better issue management.
9. **Admin Reports & Logs** – Includes student data analysis, hostel usage statistics, and complaint resolution insights.

5. SYSTEM TESTING AND IMPLEMENTATION

5.1 System Testing

System testing is a crucial phase in software development that ensures the **GateHost System** operates as expected before deployment. This phase helps identify and fix defects, ensuring that the system meets functional and non-functional requirements. Testing is performed in multiple stages, each addressing different aspects of the system's performance and reliability.

Unit Testing: This is the first level of testing, where individual components or modules, such as user authentication, room allocation, fee processing, and report generation, are tested in isolation. Developers use various debugging tools and test cases to ensure each module functions correctly before integrating them into the system.

Integration Testing: Once the individual modules are tested and validated, they are combined and tested as a whole. This step ensures that different modules, such as student registration, room booking, payment processing, and complaint management, interact correctly. Integration testing is essential to identify errors related to data flow between different components of the system.

Functional Testing: This testing method verifies that the system's functionalities work according to the specified requirements. Each feature, such as adding new students, assigning rooms, tracking payments, and generating hostel reports, is tested to ensure it performs as expected. Any discrepancies between expected and actual outputs are identified and resolved.

Performance Testing: The **GateHost System** must be capable of handling multiple user requests simultaneously without performance degradation. Performance testing evaluates system responsiveness under various conditions, such as high user loads, to ensure smooth operation. It includes **load testing** (checking how the system performs under expected traffic) and **stress testing** (assessing the system's behaviour under extreme conditions).

Security Testing: Given the sensitive nature of student and financial data, security testing is essential to identify vulnerabilities that could be exploited by unauthorized users. Measures such as **penetration testing** and **SQL injection testing** are performed to prevent unauthorized access, data breaches, and cyber threats. Role-based access control (RBAC) is also tested to ensure that different user roles (students, wardens, and administrators) have appropriate access levels.

User Acceptance Testing (UAT): Before final deployment, real users (such as hostel wardens, staff, and students) interact with the system to verify its usability and effectiveness. Their feedback is collected, and necessary improvements are made to enhance user experience and ensure smooth operation in a real-world environment.

5.2 System Implementation

Once testing confirms that the **GateHost System** is stable and free of critical defects, the implementation phase begins. This phase involves deploying the system in a live environment, training users, and ensuring a smooth transition from any previous system or manual processes.

Deployment: The system is deployed on a secure server, which may be cloud-based or on-premises, depending on the institution's requirements. Proper **server configuration** is ensured to support smooth operations and prevent downtime.

Data Migration: If the hostel previously maintained records manually or on another software, data migration is performed to transfer student records, room details, payment history, and other essential information to the new system. This process ensures data accuracy and integrity while preventing loss of critical information.

User Training: Training is essential for hostel staff, students, and administrators to familiarize themselves with the system. Workshops, user manuals, and tutorial videos are provided to help users navigate the system efficiently. Hands-on training sessions allow users to learn how to book rooms, update details, process payments, and generate reports.

Go-Live: After training and initial testing in a live environment, the system officially goes live. The transition is often done in phases to minimize disruptions. For instance, the system may first be rolled out to administrative staff before being made available to students.

Monitoring and Maintenance: Even after deployment, continuous monitoring is necessary to ensure the system functions optimally. Any technical issues, bugs, or security vulnerabilities that arise post-deployment are addressed through periodic software updates and patches. Regular feedback from users is gathered to improve functionality and add new features based on evolving needs.

6. CONCLUSION

The GateHost System was developed to streamline and enhance the efficiency of hostel operations, ensuring a smooth and user-friendly experience for students, wardens, and administrators. By automating key functions such as student registration, room allocation, fee management, and complaint tracking, the system significantly reduces manual effort, minimizes errors, and improves overall hostel management.

Extensive testing was conducted throughout the development process to ensure system stability, security, and performance. With a well-structured implementation plan, the system was successfully deployed, providing users with an intuitive interface and essential functionalities that simplify hostel administration. The integration of secure data handling and structured workflows ensures that sensitive student information remains protected while maintaining compliance with institutional policies.

The GateHost System has received positive feedback from students, wardens, and administrators, highlighting its effectiveness in addressing common hostel-related challenges. It has not only improved operational efficiency but also enhanced transparency and data security, creating a more accountable and structured environment.

Looking ahead, the system is designed to be scalable, with planned future enhancements such as AI-driven analytics, mobile app integration, and advanced reporting tools. These features will further optimize hostel management by enabling predictive maintenance, automated reporting, and real-time insights into hostel operations.

With continuous monitoring, timely updates, and dedicated support, the GateHost System is set to evolve as a benchmark solution in hostel administration. By leveraging technological advancements, it will continue to drive efficiency, improve the user experience, and ensure a modernized, well-managed hostel ecosystem for years to come.

7. FUTURE ENHANCEMENT

While the current system effectively addresses hostel management needs, there is room for future enhancements to further improve its functionality. Some potential upgrades include:

Mobile Application: A Dedicated App for Students and Staff

A mobile application can significantly enhance the accessibility and efficiency of the GateHost System by providing students and staff with a seamless platform to manage various hostel-related activities on their smartphones. The app can include features such as digital gate pass requests, attendance tracking, room allocation updates, complaint submissions, notifications, and direct communication with hostel staff. By enabling mobile access, students can conveniently request approvals, receive real-time alerts, and stay updated on hostel policies. Additionally, the app can support push notifications, ensuring instant updates on gate pass approvals, attendance status, and important announcements.

Biometric Authentication: Integration of Fingerprint or Facial Recognition for Enhanced Security

Biometric authentication, such as fingerprint scanning or facial recognition, can strengthen security and ensure accurate identity verification. This feature can be used for multiple purposes, including:

- **Secure Hostel Entry and Exit**: Only authorized students and staff can enter or exit the hostel premises, reducing unauthorized access and enhancing security.
- **Automated Attendance Tracking**: Students can mark their attendance using biometric authentication, eliminating manual attendance tracking and preventing proxy attendance.
- **Gate Pass Verification**: Biometric authentication can be used at exit points to verify student identity before allowing them to leave with an approved gate pass.
- **Staff & Visitor Authentication**: Hostel staff can use biometric authentication for secure login access to the system, while visitor registration can include fingerprint or facial recognition for security tracking.
- **Integration with Smart Locks and Security Systems**: Biometric authentication can be linked with hostel room locks, ensuring only assigned students have access to their rooms. Additionally, integrating with CCTV cameras can enhance security monitoring.

8. BIBLIOGRAPHY

The following sources were referenced and consulted during the development of the Hostel Management System:

1. **Books & Journals**
   - Tanenbaum, A. S. (2006). *Modern Operating Systems*. Pearson Education.
   - Elmasri, R., & Navathe, S. B. (2017). *Fundamentals of Database Systems*. Pearson.
   - Sommerville, I. (2015). *Software Engineering (10th Edition).* Pearson
   - Pressman, R. S. (2019). *Software Engineering: A Practitioner's Approach (9th Edition).* McGraw-Hill
2. **Web Resources**
   - W3Schools. (n.d.). *HTML, CSS, JavaScript, and SQL Tutorials*. Retrieved from www.w3schools.com
   - GeeksforGeeks. (n.d.). *Database Management & System Design Concepts*. Retrieved from www.geeksforgeeks.org
   - **Stack Overflow.** (n.d.). *Programming Q&A Forum.* Retrieved from www.stackoverflow.com – Helped resolve issues related to system development, debugging, and troubleshooting.
   - **MDN Web Docs.** (n.d.). *HTML, CSS, and JavaScript Documentation.* Retrieved from developer.mozilla.org – Provided guidance on best practices in web development.
3. **Research Papers & Articles**
   - Kumar, R., & Sharma, P. (2020). *Automated Hostel Management System: A Study on Efficiency & Security*. International Journal of Computer Applications.
   - Singh, A., & Verma, K. (2019). *Implementing Role-Based Access Control in Web Applications*. Journal of Information Security and Applications.
   - Patel, M., & Reddy, S. (2021). *Cloud-Based Hostel Management Systems: A Case Study on Data Security. Journal of Cloud Computing Research*
4. **Software & Tools Used**
   - Microsoft Visual Studio (2022) – Used for developing the web-based hostel management system.
   - SQL Server Management Studio – For database management and query execution.
   - Adobe XD & Figma – Used for designing UI/UX prototypes before implementation.

9.APPENDIX

9.1. Source Code

Login

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
   <title>Login – GateHost System</title>
   <link rel="stylesheet" type="text/css" href="styles.css">
</head>
<body>
   <div class="session">
      <div class="left">
      </div>
      <form class="log-in" autocomplete="off">
         <h4>NGI <span>Hostels</span></h4>
         <p>Welcome back Students, HoDs, and Wardens!</p>
         <div class="floating-label">
            <input placeholder="Email" type="email" name="email" id="email" class="aa"
autocomplete="off">
            <label for="email">Email:</label>
            <div class="icon">
               <img src="email-icon.svg" alt="Email Icon">
            </div>
         </div>
         <div class="floating-label">
            <input placeholder="Password" type="password" name="password" id="password"
class="aa" autocomplete="off">
            <label for="password">Password:</label>
            <div class="icon">
               <img src="password-icon.svg" alt="Password Icon">
            </div>
         </div>
         <br />
```

```
            <div class="radio">
                <b>User Type :</b>
                <input type="radio" name="role" value="Student"> Student
                <input type="radio" name="role" value="HoD"> HoD
                <input type="radio" name="role" value="Warden"> Warden
            </div>
            <div class="right">
                <button type="button" onclick="signUp()">Sign Up</button>
                <button type="submit" onclick="login()">Log in</button>
            </div>
        </form>
    </div>
    <script src="script.js"></script>
</body>
</html>
```

Signup

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <!----======== CSS ======== -->
    <link rel="stylesheet" href="style.css">

    <!----===== Iconscout CSS ===== -->
    <link rel="stylesheet" href="https://unicons.iconscout.com/release/v4.0.0/css/line.css">

    <title>Student Registration - NGI Hostel Management</title>
</head>
<body>
    <div class="container">
        <header>Student Registration</header>
    <form id="registrationForm">
        <div class="form first">
            <div class="details personal">
                <span class="title">Personal Details</span>
                <div class="fields">
                    <div class="input-field">
                        <label>Full Name</label>
                        <input type="text" id="fullName" class="input" placeholder="Enter your
name" required>
                    </div>

                    <div class="input-field">
                        <label>Date of Birth</label>
                        <input type="date" id="dob" class="input" required>
                    </div>

                    <div class="input-field">
                        <label>Mobile Number</label>
                        <input type="number" id="mobileNumber" class="input"
placeholder="Enter mobile number" required>
                    </div>

                    <div class="input-field">
                        <label>Address</label>
                        <input type="text" id="address" class="input" placeholder="Enter your
address" required>
                    </div>

                    <div class="input-field">
                        <label>Email / Username</label>
                        <input type="email" id="emailUsername" class="input" placeholder="Enter
your email or username" required>
                    </div>
                    <div class="input-field">
                        <label>Password</label>
                        <input type="password" id="password" class="input" placeholder="Enter
your password" required>
                    </div>

                    <div class="center">
                        <div class="input-field">
                            <label>Student Photo</label>
                            <input type="file" id="photo" class="input">
                        </div>
                    </div>
                </div>
            </div>

            <div class="details ID">
                <span class="title">Additional Details</span>
                <div class="fields">
                    <div class="input-field">
                        <label>Class</label>
                        <input type="text" id="txtclass" class="input" placeholder="Enter class"
required>
                    </div>

                    <div class="input-field">
                        <label>Department</label>
                        <input type="text" id="department" class="input" placeholder="Enter
department" required>
                    </div>

                    <div class="input-field">
                        <label>Year</label>
                        <input type="number" id="yearOfStudy" class="input" placeholder="Enter
year of study" required>
```

```
                    </div>

                    <div class="input-field">
                        <label>Block and Room Number</label>
                        <input type="text" id="txtrno" class="input" placeholder="Enter Block and
Room Number" required>
                    </div>

                    <button type="submit" class="nextBtn">Submit</button>
                </div>
            </div>
        </div>
    </form>
  </div>

  <script src="script.js"></script>
</body>
</html>
```

Hod Outpass Generation

```
<%@ Page Title="Out-Pass Application" Language="C#" MasterPageFile="~/hostel.master"
AutoEventWireup="true" CodeFile="OutPass.aspx.cs" Inherits="Student_OutPass" %>

<asp:Content ID="Content1" ContentPlaceHolderID="HEAD" runat="Server">
</asp:Content>

<asp:Content ID="Content2" ContentPlaceHolderID="BODY" runat="Server">

  <ajaxToolkit:TabContainer ID="cont" runat="server">
    <ajaxToolkit:TabPanel runat="server">
      <HeaderTemplate>Pending Requests</HeaderTemplate>
      <ContentTemplate>
        <div>
        <asp:GridView ID="gv_pending_requests" runat="server"
AutoGenerateColumns="false" EmptyDataText="No Pending Outpass Yet"
OnRowCommand="gv_pending_requests_RowCommand">
            <Columns>
                <asp:TemplateField>
                    <HeaderTemplate>S.No</HeaderTemplate>
                    <ItemTemplate>
                        <%#Container.DataItemIndex + 1 %>
                    </ItemTemplate>
                </asp:TemplateField>
                <asp:BoundField HeaderText="Student Name" DataField="StudentName"
/>
                <asp:BoundField HeaderText="Department" DataField="Department" />
                <asp:BoundField HeaderText="Class" DataField="Class" />
                <asp:BoundField HeaderText="Room Number" DataField="roomnumber"
/>
                <asp:BoundField HeaderText="Outpass Start Date"
DataField="OutpassStartDate" DataFormatString="{0:yyyy-MM-dd}" HtmlEncode="false"
/>
                <asp:BoundField HeaderText="Outpass End Date"
DataField="OutpassEndDate" DataFormatString="{0:yyyy-MM-dd}" HtmlEncode="false" />
                <asp:BoundField HeaderText="Start Time" DataField="outpassStartTime"
/>
                <asp:BoundField HeaderText="End Time" DataField="outpassEndTime" />
                <asp:BoundField HeaderText="Reason" DataField="Reason" />
                <asp:BoundField HeaderText="Parent Contact" DataField="ParentContact"
/>
                <asp:TemplateField>
                    <HeaderTemplate>Action</HeaderTemplate>
                    <ItemTemplate>
                        <asp:Label ID="transid" runat="server" Text='<%#Eval("requestid")
%>' Visible="false"></asp:Label>
```

```
                        <asp:LinkButton ID="lnkApprove" runat="server"
OnClientClick="return confirm('Are you sure you want to approve this?');"
CommandName="approve" CommandArgument='<%#Eval("requestid") %>'>
                          Approve
                        </asp:LinkButton>
                        <asp:LinkButton ID="lnkReject" runat="server"
OnClientClick="return confirm('Are you sure you want to reject this?');"
CommandName="reject" CommandArgument='<%#Eval("requestid") %>'>
                          Reject
                        </asp:LinkButton>
                      </ItemTemplate>
                    </asp:TemplateField>
                  </Columns>
                </asp:GridView>
              </div>
            </ContentTemplate>
          </ajaxToolkit:TabPanel>

          <ajaxToolkit:TabPanel runat="server">
            <HeaderTemplate>Approved Requests</HeaderTemplate>
            <ContentTemplate>
              <div>
                <asp:GridView ID="gv_outpass_approved" runat="server"
AutoGenerateColumns="false" EmptyDataText="No Outpass Requests Approved Yet">
                  <Columns>
                    <asp:TemplateField>
                      <HeaderTemplate>S.No</HeaderTemplate>
                      <ItemTemplate>
                        <%#Container.DataItemIndex + 1 %>
                      </ItemTemplate>
                    </asp:TemplateField>
                    <asp:BoundField HeaderText="Student Name" DataField="StudentName"
/>
                    <asp:BoundField HeaderText="Department" DataField="Department" />
```

```
                        <asp:BoundField HeaderText="Class" DataField="Class" />
                        <asp:BoundField HeaderText="Room Number" DataField="roomnumber" />
                        <asp:BoundField HeaderText="Outpass Start Date" DataField="OutpassStartDate" DataFormatString="{0:yyyy-MM-dd}" HtmlEncode="false" />
                        <asp:BoundField HeaderText="Outpass End Date" DataField="OutpassEndDate" DataFormatString="{0:yyyy-MM-dd}" HtmlEncode="false" />
                    </Columns>
                </asp:GridView>
            </div>
        </ContentTemplate>
    </ajaxToolkit:TabPanel>

    <ajaxToolkit:TabPanel runat="server">
        <HeaderTemplate>Rejected Requests</HeaderTemplate>
        <ContentTemplate>
            <div>
                <asp:GridView ID="gv_outpass_rejected" runat="server" AutoGenerateColumns="false" EmptyDataText="No Outpass Requests Rejected Yet">
                    <Columns>
                        <asp:TemplateField>
                            <HeaderTemplate>S.No</HeaderTemplate>
                            <ItemTemplate>
                                <%#Container.DataItemIndex + 1 %>
                            </ItemTemplate>
                        </asp:TemplateField>
                        <asp:BoundField HeaderText="Student Name" DataField="StudentName" />
                        <asp:BoundField HeaderText="Department" DataField="Department" />
                        <asp:BoundField HeaderText="Class" DataField="Class" />
                        <asp:BoundField HeaderText="Room Number" DataField="roomnumber" />
                        <asp:BoundField            HeaderText="Outpass            Start            Date"
DataField="OutpassStartDate"  DataFormatString="{0:yyyy-MM-dd}"  HtmlEncode="false"
/>
                        <asp:BoundField            HeaderText="Outpass            End            Date"
DataField="OutpassEndDate" DataFormatString="{0:yyyy-MM-dd}" HtmlEncode="false" />
                    </Columns>
                </asp:GridView>
            </div>
        </ContentTemplate>
    </ajaxToolkit:TabPanel>
  </ajaxToolkit:TabContainer>

</asp:Content>
```

Student

```
<%@   Page   Title="Mess   Card"   Language="C#"   MasterPageFile="~/hostel.master"
AutoEventWireup="true" CodeFile="MessCard.aspx.cs" Inherits="Student_MessCard" %>

<asp:Content ID="Content1" ContentPlaceHolderID="HEAD" runat="Server">
  <script src="https://cdn.tailwindcss.com"></script>
</asp:Content>

<asp:Content ID="Content2" ContentPlaceHolderID="BODY" runat="Server">
  <div class="flex justify-center items-center h-screen bg-gray-100">
    <div class="mess-card">
      <div class="mess-header">
        <h2 class="text-2xl font-semibold text-white">Mess Card</h2>
        <asp:Image ID="Image1" runat="server" CssClass="student-photo" />
      </div>

      <div class="mess-details">
        <p><span>ID:</span>
          <asp:Label ID="Label1" runat="server" class="inline-block"></asp:Label></p>
        <p><span>Name:</span>
          <asp:Label ID="Label2" runat="server" class="inline-block"></asp:Label></p>
```

```
            <p><span>Department:</span>
              <asp:Label ID="Label3" runat="server" class="inline-block"></asp:Label></p>
            <p><span>Year:</span>
              <asp:Label ID="Label4" runat="server" class="inline-block"></asp:Label></p>
            <p><span>Room No:</span>
              <asp:Label ID="Label5" runat="server" class="inline-block"></asp:Label></p>
            <p><span>Contact:</span>
              <asp:Label ID="Label6" runat="server" class="inline-block"></asp:Label></p>
            <p><span>Email:</span>
              <asp:Label ID="Label7" runat="server" class="inline-block"></asp:Label></p>
          </div>

          <div class="barcode flex items-center justify-center">
            <asp:Image ID="Image2" runat="server" />
          </div>
        </div>
    </div>
</asp:Content>
```

Student Profile

```
<%@ Page Title="Student Profile" Language="C#" MasterPageFile="~/hostel.master"
AutoEventWireup="true" CodeFile="Profile.aspx.cs" Inherits="Student_Profile" %>

<asp:Content ID="Content1" ContentPlaceHolderID="HEAD" runat="Server">
</asp:Content>

<asp:Content ID="Content2" ContentPlaceHolderID="BODY" runat="Server">
    <div>
        <h2>Student Profile</h2>

        <!-- Profile Picture -->
        <div>
            <asp:Image ID="imgProfile" runat="server" />
        </div>
```

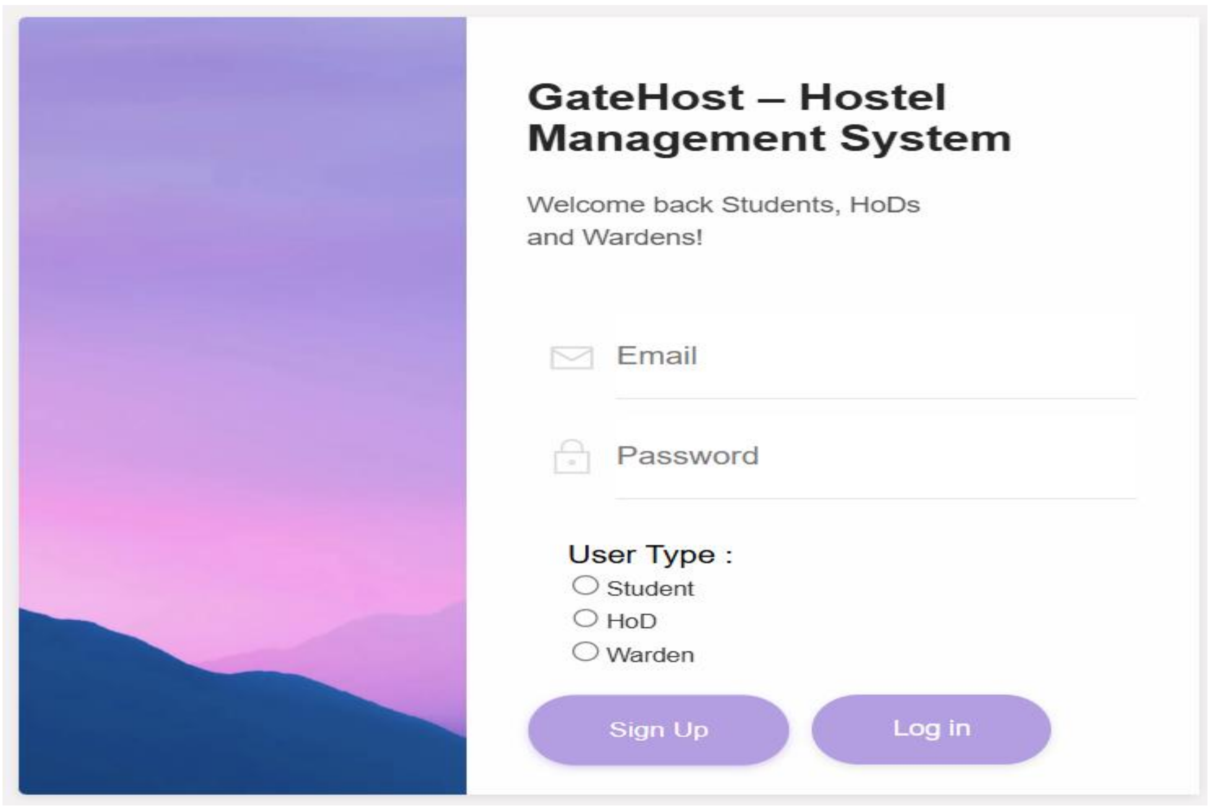
```
<!-- Two-column Layout -->
<div>
  <!-- Column 1 -->
  <div>
    <p>Student ID:</p>
    <asp:Label ID="lblStudentID" runat="server"></asp:Label>

    <p>Name:</p>
    <asp:TextBox ID="txtName" runat="server"></asp:TextBox>

    <p>DOB:</p>
    <asp:TextBox ID="txtDOB" runat="server" TextMode="Date"></asp:TextBox>

    <p>Department:</p>
    <asp:TextBox ID="txtDepartment" runat="server"></asp:TextBox>

    <p>Year:</p>
    <asp:TextBox ID="txtYear" runat="server"></asp:TextBox>

    <p>Room No:</p>
    <asp:TextBox ID="txtRoomNo" runat="server"></asp:TextBox>
  </div>

  <!-- Column 2 -->
  <div>
    <p>Username:</p>
    <asp:Label ID="lblUsername" runat="server"></asp:Label>

    <p>Contact No:</p>
    <asp:TextBox ID="txtContact" runat="server"></asp:TextBox>

    <p>Email:</p>
    <asp:TextBox ID="txtEmail" runat="server"></asp:TextBox>
```

```
            <p>Address:</p>
            <asp:TextBox ID="txtAddress" runat="server"></asp:TextBox>

            <p>Class:</p>
            <asp:TextBox ID="txtClass" runat="server"></asp:TextBox>

            <p>Password:</p>
            <asp:TextBox ID="txtPassword" runat="server"
TextMode="Password"></asp:TextBox>

            <p>Status:</p>
            <asp:Label ID="lblStatus" runat="server"></asp:Label>
          </div>
        </div>

        <!-- Update Button -->
        <div>
          <asp:Button ID="btnUpdate" runat="server" Text="Update Profile"
OnClick="btnUpdate_Click" />
        </div>
    </div>
</asp:Content>
```

Warden Attendance

```
<%@ Page Title="" Language="C#" MasterPageFile="~/hostel.master"
AutoEventWireup="true" CodeFile="Attendance.aspx.cs" Inherits="Student_Attendance" %>

<asp:Content ID="Content1" ContentPlaceHolderID="HEAD" runat="Server">
</asp:Content>

<asp:Content ID="Content2" ContentPlaceHolderID="BODY" runat="Server">
    <div>
        <h2>Warden Attendance</h2>
        <asp:Calendar                          ID="calAttendance"                              runat="server"
OnDayRender="calAttendance_DayRender"
OnSelectionChanged="calAttendance_SelectionChanged"></asp:Calendar>

        <hr>

        <!-- Student Attendance Grid -->
        <div>
            <asp:GridView    ID="gvStudents"    runat="server"    AutoGenerateColumns="False"
GridLines="None">

                <Columns>
                    <asp:BoundField DataField="RoomNo" HeaderText="Room No" />
                    <asp:BoundField DataField="StudentID" HeaderText="Student ID" />
                    <asp:BoundField DataField="Name" HeaderText="Name" />
                    <asp:TemplateField HeaderText="Attendance">
                        <ItemTemplate>
                            <asp:RadioButtonList              ID="rblAttendance"              runat="server"
RepeatDirection="Horizontal">
                                <asp:ListItem Text="Present" Value="Present"></asp:ListItem>
                                <asp:ListItem Text="Absent" Value="Absent"></asp:ListItem>
                            </asp:RadioButtonList>
                        </ItemTemplate>
                    </asp:TemplateField>
                </Columns>
            </asp:GridView>
        </div>

        <div>
            <asp:Button    ID="btnSubmit"    runat="server"    Text="Submit    Attendance"
OnClick="btnSubmit_Click" />
        </div>
    </div>
</asp:Content>
```

9.2. Screenshots



GateHost – Hostel Management System

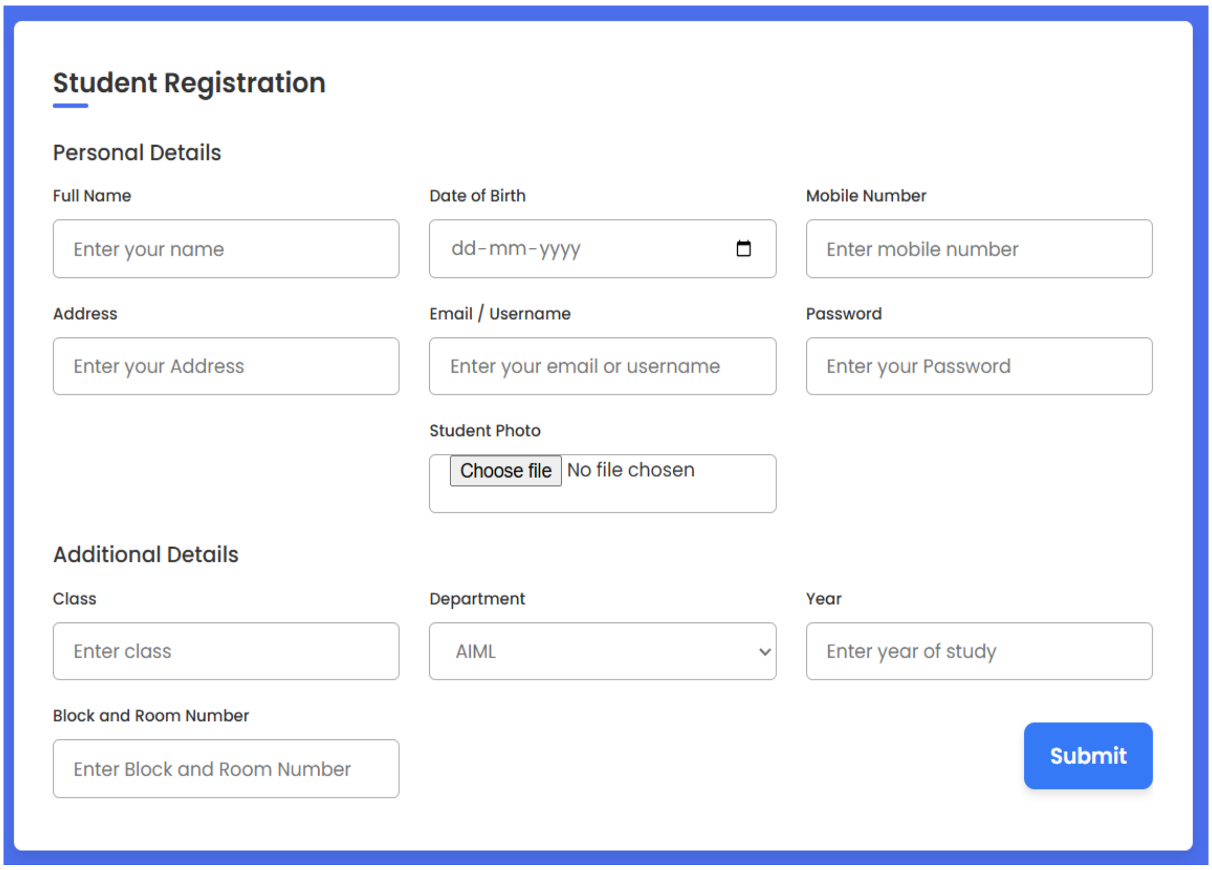
Welcome back Students, HoDs and Wardens!

Email

Password

User Type :

○ Student
○ HoD
○ Warden



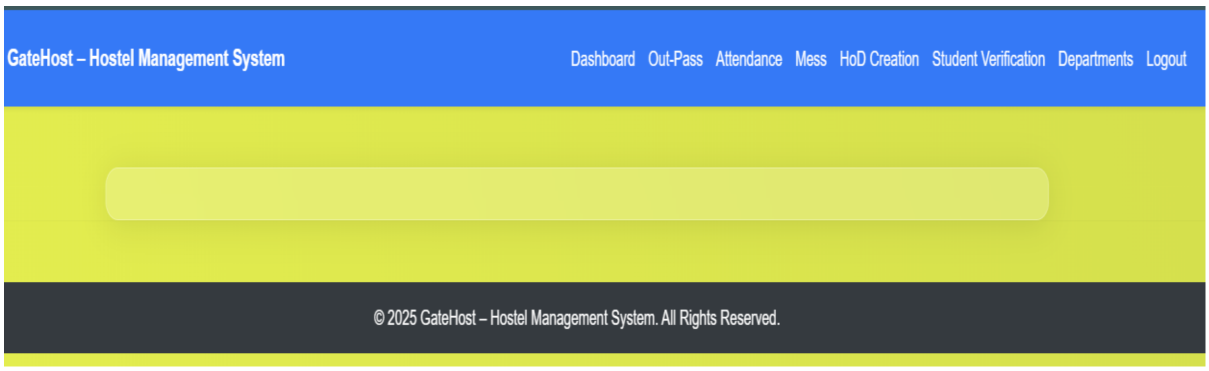
Student Registration

Personal Details

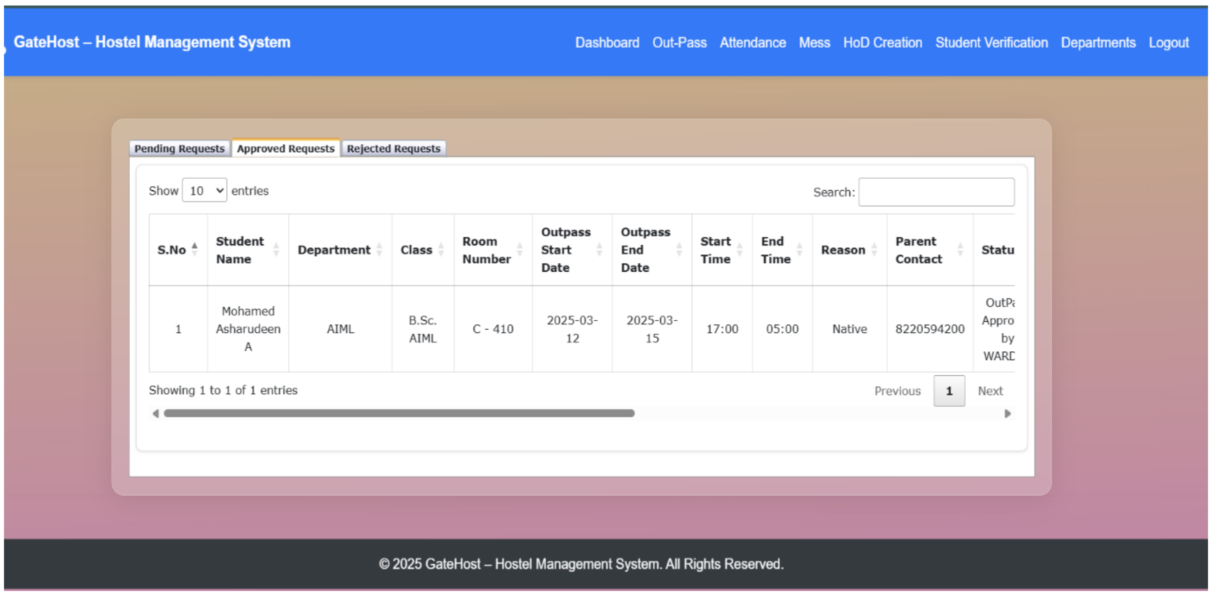
Full Name	Date of Birth	Mobile Number
Enter your name	dd-mm-yyyy	Enter mobile number
Address	Email / Username	Password
Enter your Address	Enter your email or username	Enter your Password
	Student Photo	
	Choose file No file chosen	

Additional Details

Class	Department	Year
Enter class	AIML	Enter year of study
Block and Room Number		
Enter Block and Room Number		


GateHost – Hostel Management System
Dashboard Out-Pass Attendance Mess HoD Creation Student Verification Departments Logout

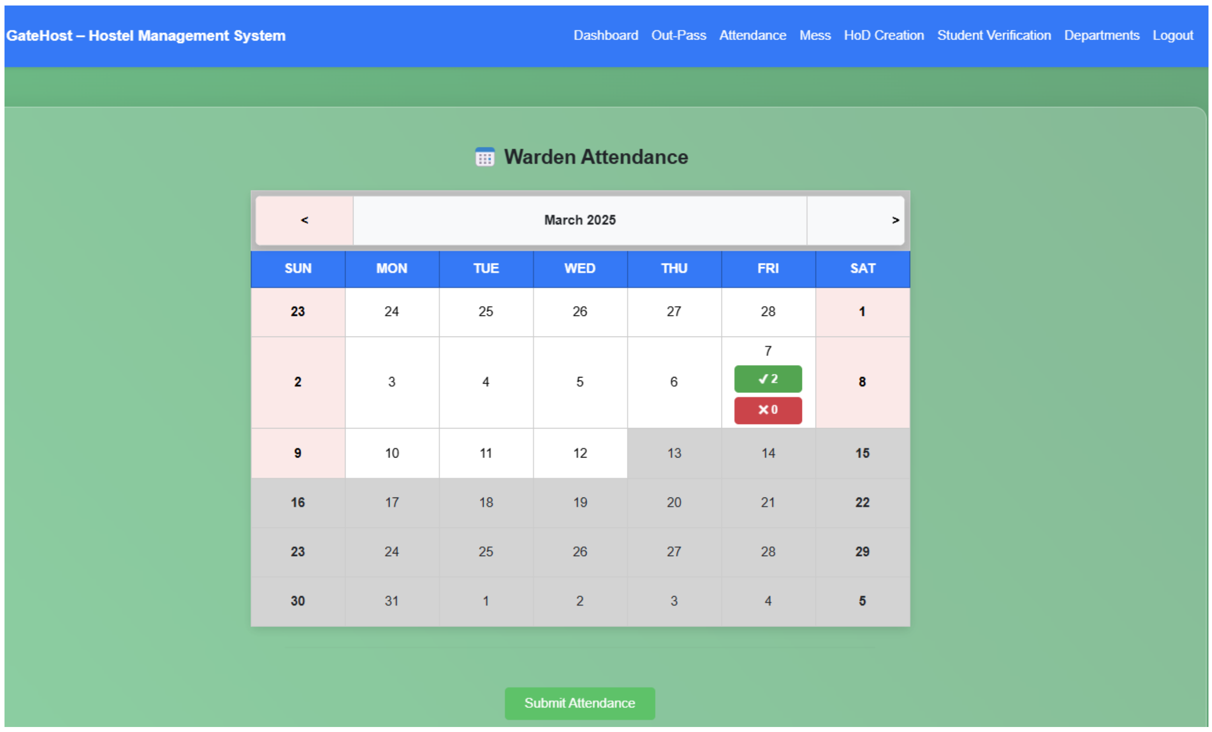
© 2025 GateHost – Hostel Management System. All Rights Reserved.


GateHost – Hostel Management System
Dashboard Out-Pass Attendance Mess HoD Creation Student Verification Departments Logout
Pending Requests
Approved Requests
Rejected Requests
Show 10 entries
Search:

S.No	Student Name	Department	Class	Room Number	Outpass Start Date	Outpass End Date	Start Time	End Time	Reason	Parent Contact	Statu
1	Mohamed Asharudeen A	AIML	B.Sc. AIML	C - 410	2025-03-12	2025-03-15	17:00	05:00	Native	8220594200	OutPa Appro by WARD

Showing 1 to 1 of 1 entries

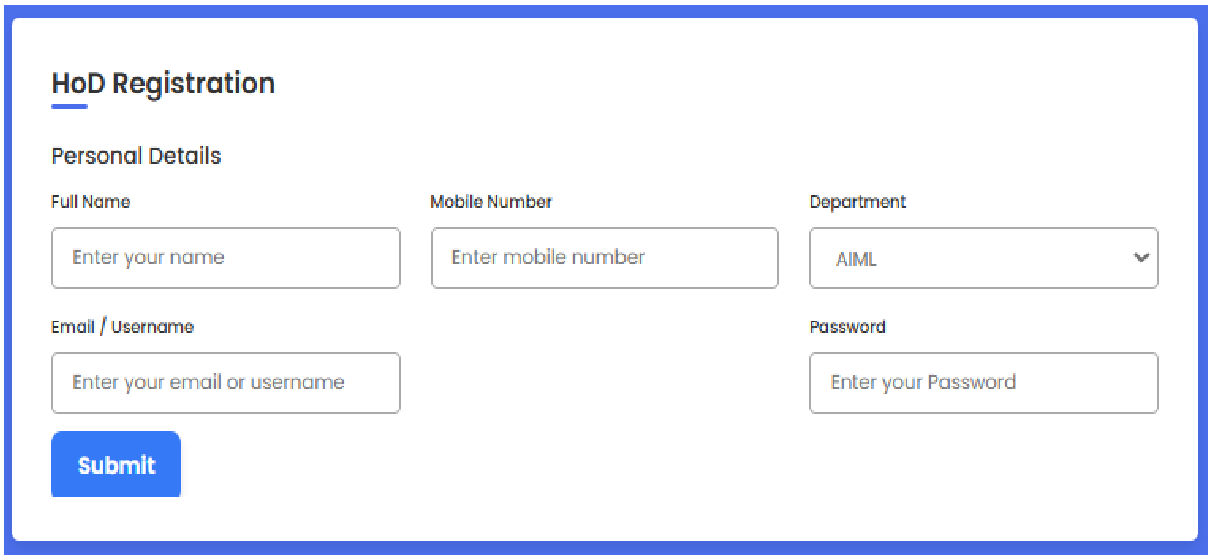
© 2025 GateHost – Hostel Management System. All Rights Reserved.


GateHost – Hostel Management System
Dashboard Out-Pass Attendance Mess HoD Creation Student Verification Departments Logout

Warden Attendance

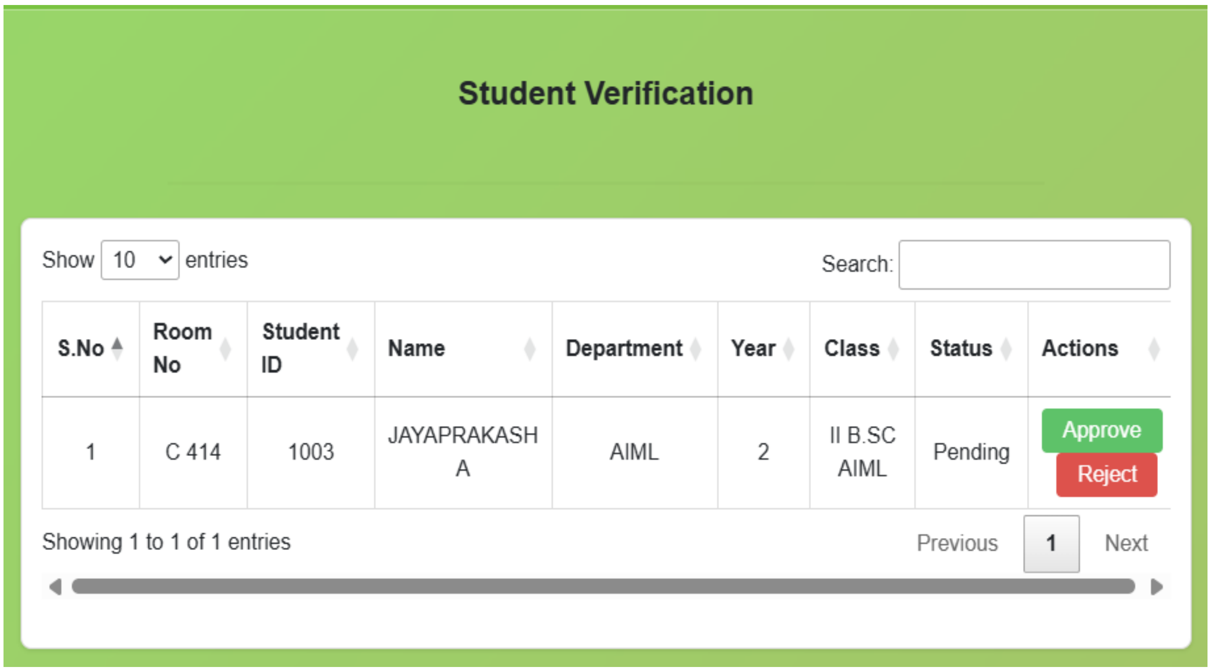
< March 2025 >

SUN	MON	TUE	WED	THU	FRI	SAT
23	24	25	26	27	28	1
2	3	4	5	6	7 ✓2 ✗0	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	31	1	2	3	4	5


HoD Registration

Personal Details

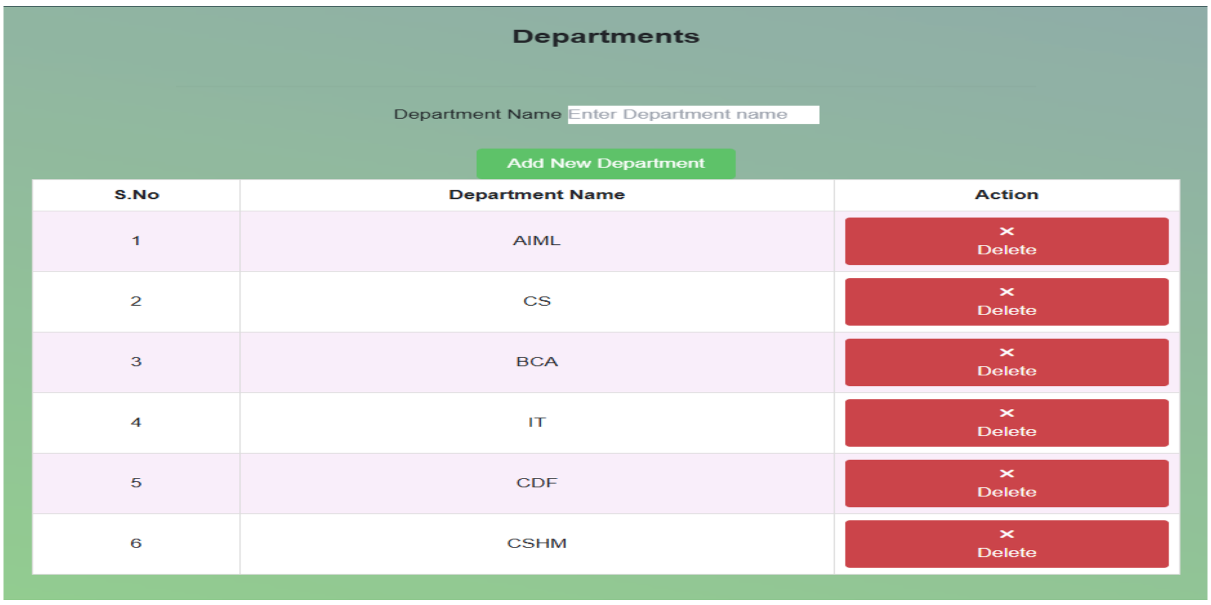
Full Name	Mobile Number	Department
Enter your name	Enter mobile number	AIML
Email / Username		Password
Enter your email or username		Enter your Password


Student Verification

Show 10 entries Search:

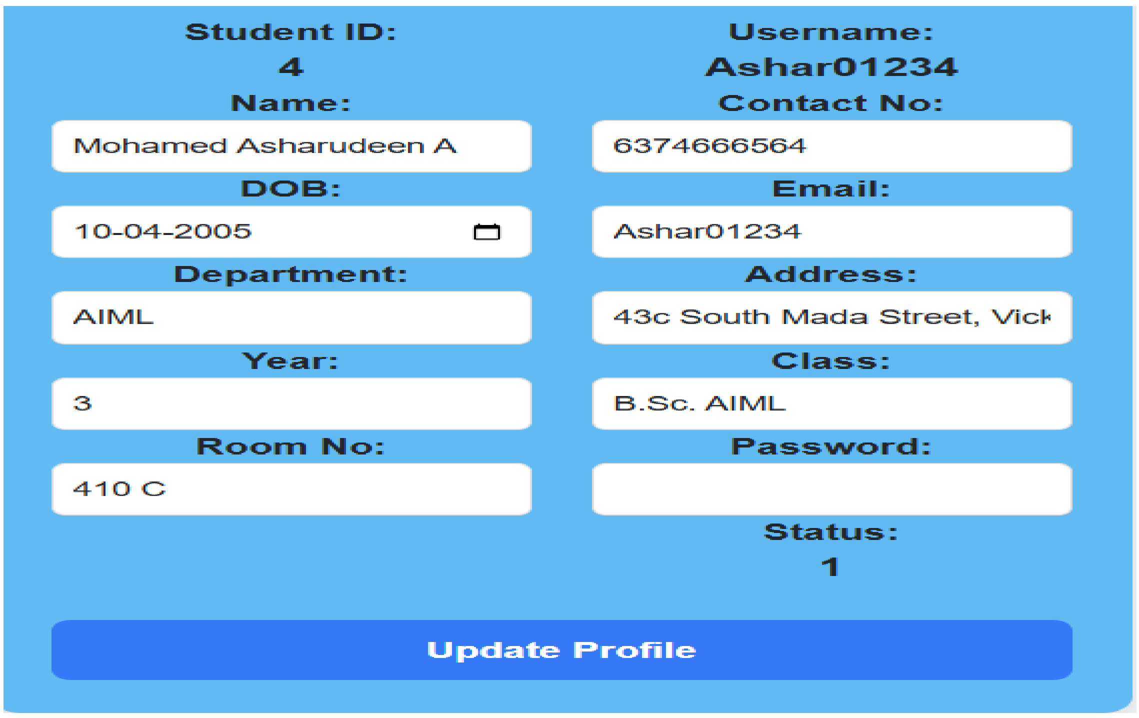
S.No	Room No	Student ID	Name	Department	Year	Class	Status	Actions
1	C 414	1003	JAYAPRAKASH A	AIML	2	II B.SC AIML	Pending	Approve Reject

Showing 1 to 1 of 1 entries

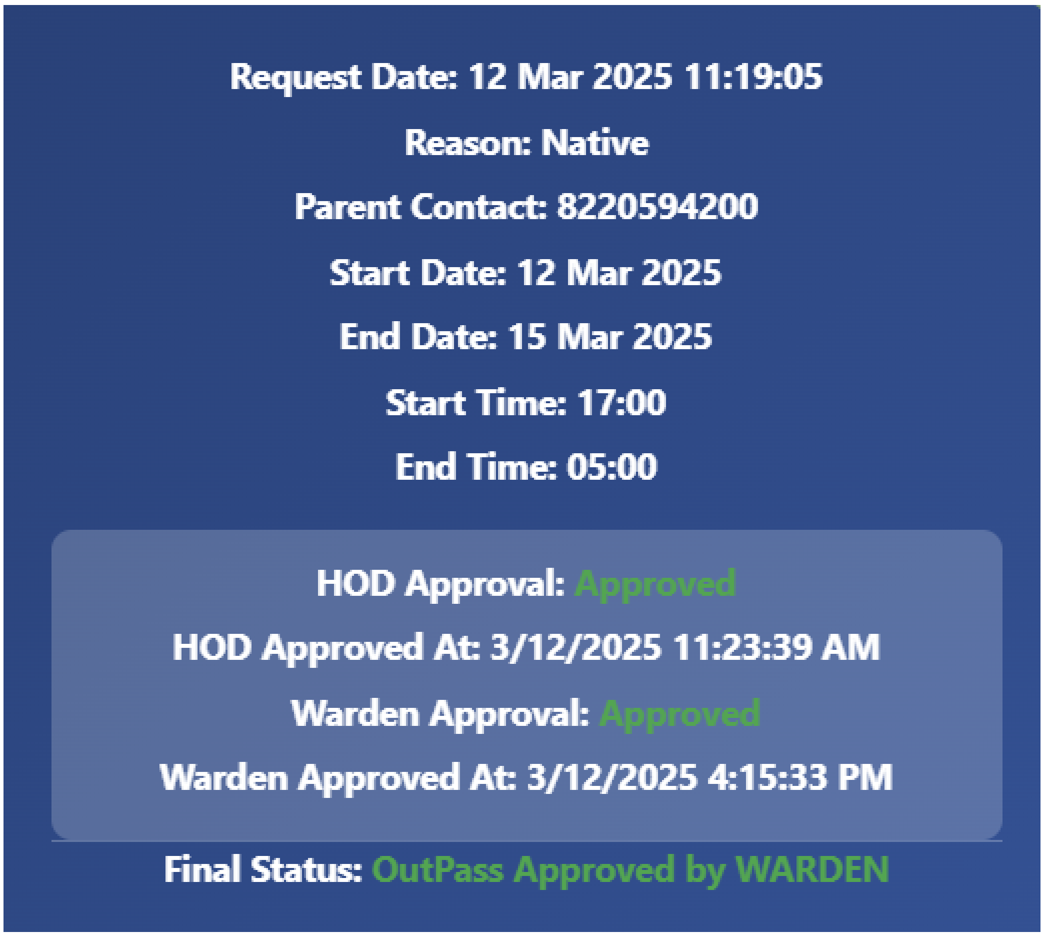

Departments

Department Name Enter Department name

S.No	Department Name	Action
1	AIML	✕ Delete
2	CS	✕ Delete
3	BCA	✕ Delete
4	IT	✕ Delete
5	CDF	✕ Delete
6	CSHM	✕ Delete



Student ID: 4	Username: Ashar01234
Name: Mohamed Asharudeen A	Contact No: 6374666564
DOB: 10-04-2005	Email: Ashar01234
Department: AIML	Address: 43c South Mada Street, Vick
Year: 3	Class: B.Sc. AIML
Room No: 410 C	Password:
	Status: 1



Request Date: 12 Mar 2025 11:19:05

Reason: Native

Parent Contact: 8220594200

Start Date: 12 Mar 2025

End Date: 15 Mar 2025

Start Time: 17:00

End Time: 05:00

HOD Approval: Approved

HOD Approved At: 3/12/2025 11:23:39 AM

Warden Approval: Approved

Warden Approved At: 3/12/2025 4:15:33 PM

Final Status: OutPass Approved by WARDEN