 <b>Marwadi University</b> Marwadi Chandarana Group	<b>Marwadi University</b> <b>Faculty of Engineering and Technology</b> <b>Department of Information and Communication Technology</b>
<b>Subject: Capstone Project</b>	<b>Aim: Implementation - Continuous progress review</b>
<b>Date: 26-9-2025</b>	<b>Enrolment No: 92310133001</b>

## 1. Abstract

The Galaxy Hostel Website is a web application designed to simplify and modernize hostel management for students, wardens, and administrators. The system replaces manual processes like attendance marking, gate pass approvals, and complaint tracking with a digital platform that is easy to use and accessible. Key features include secure login, online attendance tracking, digital gate pass requests, complaint and feedback submission, and a built-in AI assistant to provide quick guidance.

The website is built using Next.js for the front-end, with TailwindCSS for styling, and Firestore as the database to ensure real-time updates and secure storage. The system is designed to be modular and scalable, with error handling and validation to maintain reliability during daily operations. Additional tools like CSV export for admins and QR code generation for gate passes further improve usability.


Testing and validation confirm that the Galaxy Hostel Website improves efficiency, transparency, and convenience, offering students and staff a dependable platform for managing hostel activities without relying on traditional paper-based methods.

## 2. System Overview & Objectives

- ✚ Provide a simple, low-effort workflow for marking hostel attendance: students' presence is recorded digitally, allowing wardens and admins to manage records efficiently without manual registers. The system enables secure tracking, reporting, and easy export of attendance data.

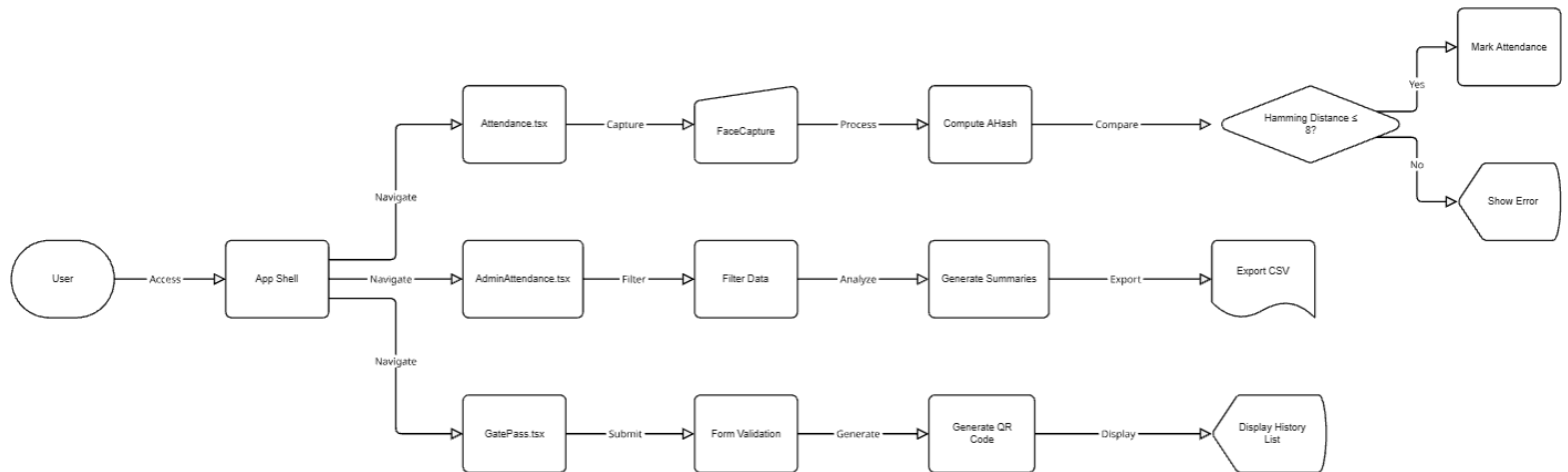
Functional Requirements Implemented:-


- Students and wardens can log in securely via the web interface.
- Attendance can be marked online, either manually or via automated recognition (if images are used in future).
- Records of students are stored and retrieved from Firestore, ensuring real-time updates.
- Admins and wardens can view, filter, and export attendance reports as Excel (.xlsx) files with Present/Absent status.
- Additional features such as complaint management, gate pass requests, feedback submission, and AI assistant support are integrated.
- Frontend is implemented using Next.js with TailwindCSS for responsive and user-friendly interfaces.
- Backend APIs handle data retrieval, validation, and secure updates, ensuring seamless integration across all modules.

 <b>Marwadi University</b> Marwadi Chandarana Group	<b>Marwadi University</b> <b>Faculty of Engineering and Technology</b> <b>Department of Information and Communication Technology</b>
<b>Subject: Capstone Project</b>	<b>Aim: Implementation - Continuous progress review</b>
<b>Date: 26-9-2025</b>	<b>Enrolment No: 92310133001</b>

### 3. High-Level Architecture

Below is a simple diagram showing component interactions (Mermaid syntax for easy rendering):



 <b>Marwadi University</b> Marwadi Chandarana Group	<b>Marwadi University</b> <b>Faculty of Engineering and Technology</b> <b>Department of Information and Communication Technology</b>
<b>Subject: Capstone Project</b>	<b>Aim: Implementation - Continuous progress review</b>
<b>Date: 26-9-2025</b>	<b>Enrolment No: 92310133001</b>

## 4. Technology Stack

### + Backend:

- Node.js / Next.js API routes – handles server-side logic, authentication, and data operations.
- Firebase Admin SDK – interacts with Firestore for storing and retrieving student, attendance, gate pass, and feedback data.
- Optional Libraries – for image handling (e.g., resizing or basic processing) if attendance via images is implemented.

### + Frontend:


- Next.js – for server-side rendering, routing, and API integration.
- TailwindCSS – for a responsive, clean, and modern user interface.
- Fetch / Axios – for API requests between frontend and backend.

### + Database / Cloud Services:

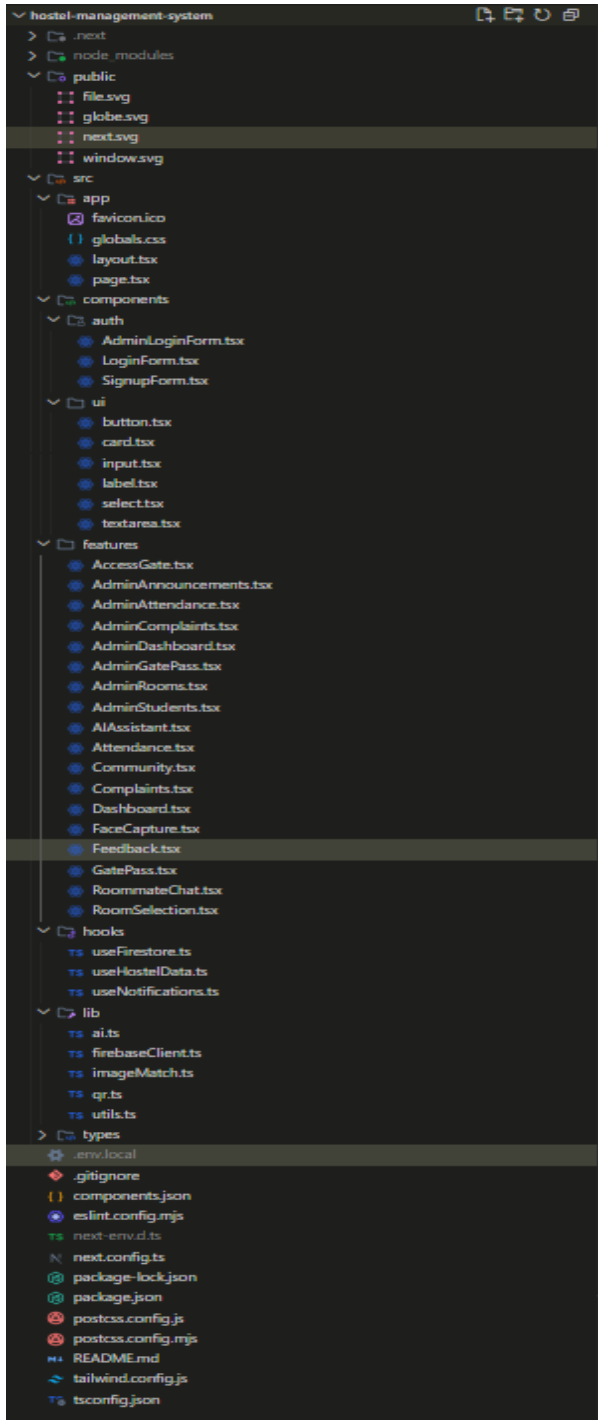
- Firestore – stores all student records, attendance logs, complaints, gate passes, and feedback data with real-time updates.
- Firebase Authentication – manages secure login for students, wardens, and admins.
- Optional – Firebase Cloud Functions can handle automated tasks like sending notifications, exporting reports, or AI assistant queries.


### + Additional Features / Tools:

- Excel Export – export attendance or gate pass records to .xlsx format.
- AI Assistant – integrates with front-end to provide instant guidance or answers for hostel-related queries.
- QR Codes – for gate pass verification and quick tracking.

 <b>Marwadi University</b> Marwadi Chandarana Group	<b>Marwadi University</b> <b>Faculty of Engineering and Technology</b> <b>Department of Information and Communication Technology</b>
<b>Subject: Capstone Project</b>	<b>Aim: Implementation - Continuous progress review</b>
<b>Date: 26-9-2025</b>	<b>Enrolment No: 92310133001</b>

## 5. Code Structure (recommended)



 <b>Marwadi University</b> Marwadi Chandarana Group	<b>Marwadi University</b> <b>Faculty of Engineering and Technology</b> <b>Department of Information and Communication Technology</b>
<b>Subject: Capstone Project</b>	<b>Aim: Implementation - Continuous progress review</b>
<b>Date: 26-9-2025</b>	<b>Enrolment No: 92310133001</b>

## 6. Key Implementation Details

### 6.1 Registration flow

#### ✚ Student Registration:

- Admin or Warden registers new students by entering their details (name, roll number, room number) and uploading a profile photo via the frontend.
- Frontend Flow:
  1. Admin uploads student image through a registration form.
  2. Form data is sent via POST request to Firebase Cloud Function.

#### ✚ Backend/Database Flow (Firebase):

1. Cloud Function receives student data and image.
2. Image is stored in Firebase Storage.
3. Student metadata is saved in Firestore, including:


### 6.2 Attendance marking (image upload)

#### ✚ Frontend Flow:

- Student submits attendance by uploading a selfie or classroom image via the attendance page.
- Data sent as multipart/form-data including image, date, class\_id, and subject.


#### ✚ Backend Flow (Firebase Cloud Function):

1. Receive the image and temporarily store it in Firebase Storage.
2. Call face recognition module to identify faces in the image by comparing with registered student embeddings stored in Firestore.
3. Build two lists:
  - Present: Matched student IDs.
  - Absent: Registered students not detected in the image.
4. Optionally, generate an Excel sheet (using a library like SheetJS) with Present and Absent students, save to Firebase Storage, or provide a downloadable link in the frontend.

 <b>Marwadi University</b> Marwadi Chandarana Group	<b>Marwadi University</b> <b>Faculty of Engineering and Technology</b> <b>Department of Information and Communication Technology</b>
<b>Subject: Capstone Project</b>	<b>Aim: Implementation - Continuous progress review</b>
<b>Date: 26-9-2025</b>	<b>Enrolment No: 92310133001</b>


## 6.3 Matching Thresholds and Edge Cases

- ✚ Face Recognition Threshold:
  - Set a configurable similarity threshold (e.g., 85%) for matching uploaded images with registered student embeddings.
- ✚ Duplicate Matches & Low Confidence:
  - If multiple students match the same face, flag for manual verification.
  - If confidence is below threshold, mark as unknown.
- ✚ Fallback Recognition:
  - If no match is found via AI embeddings, optionally use a secondary face detection model for verification (e.g., local face-api.js).
- ✚ Logging and Audit:
  - All attendance submissions are logged in Firestore with timestamp, student ID, and image URL for future auditing.

 <b>Marwadi University</b> Marwadi Chandarana Group	<b>Marwadi University</b> <b>Faculty of Engineering and Technology</b> <b>Department of Information and Communication Technology</b>
<b>Subject: Capstone Project</b>	<b>Aim: Implementation - Continuous progress review</b>
<b>Date: 26-9-2025</b>	<b>Enrolment No: 92310133001</b>

## 7. API Endpoints (example)

Endpoint	Method	Request	Response	Description
/api/register	POST	{ name, roll_no, room_no, image }	student_id	Registers a student; stores metadata in Firestore and image in Firebase Storage.
/api/attendance/upload	POST	Multipart form-data with image, class_id, date	download_link: "Excel URL"	Upload classroom image to mark attendance; generates Excel sheet with Present/Absent students.
/api/students?class_id=...	GET	Query parameter class_id	Array of student objects	Retrieves all registered students for a specific class.
/api/complaints	POST	{ student_id, complaint_text }	success: true	Logs student complaints to Firestore for admin review.
/api/complaints?status=pending	GET	Optional query parameter status	Array of complaint objects	Fetches complaints filtered by status for admin/warden dashboard.
/api/chatbot/query	POST	{ message }	response_text	Sends student query to AI chatbot; returns automated answer regarding rules, timings, etc.

 <b>Marwadi University</b> Marwadi Chandarana Group	<b>Marwadi University</b> <b>Faculty of Engineering and Technology</b> <b>Department of Information and Communication Technology</b>
<b>Subject: Capstone Project</b>	<b>Aim: Implementation - Continuous progress review</b>
<b>Date: 26-9-2025</b>	<b>Enrolment No: 92310133001</b>

## 8. Testing

### + Unit Tests

- Test utility functions such as:
  - Image preprocessing (resizing, normalization)
  - Excel generation using SheetJS
- Conduct frontend component tests with Jest + Next Testing Library for registration forms, attendance upload, and complaint submission.

### + Integration Tests

- Run `/api/attendance/upload` with real and verify:
  - Excel sheet contains correct Present and Absent entries based on registered student photos.
- Verify that images uploaded by `/api/register` are correctly stored in Firebase Storage and metadata in Firestore.

### + Manual Tests & Evidence

- Take screenshots of:
  - Registration form and successful student addition
  - Attendance upload page
  - Generated Excel sheet for attendance
  - Server logs showing recognized student IDs and attendance status

## 9. Error handling & Security

### + Validation:

- Ensure webcam is accessible and frames are not empty.
- Required fields (name, roll number, class) must be filled.
- Only valid faces are processed for attendance/registration.


### + Error Handling:

- Use try/catch for all Firebase calls.
- Return meaningful HTTP errors (400/500) and log failures.

### + Security:

- Role-based access via Firebase Authentication (student, warden, admin).
- Firestore rules restrict access based on roles.
- HTTPS enforced; no hardcoded credentials; real-time frames processed securely.



 <b>Marwadi University</b> Marwadi Chandarana Group	<b>Marwadi University</b> <b>Faculty of Engineering and Technology</b> <b>Department of Information and Communication Technology</b>
<b>Subject: Capstone Project</b>	<b>Aim: Implementation - Continuous progress review</b>
<b>Date: 26-9-2025</b>	<b>Enrolment No: 92310133001</b>

## 10. Deployment & Running Instructions

### Prerequisites

- Node 18+ (Next.js app)
- Browser camera access enabled

### Dev (local)

- npm install
- npm run dev
- Open <http://localhost:3000>

### Build (prod)

- npm run build
- npm run start
- Open <http://localhost:3000>

### Notes

- Attendance needs Face Capture and works only within time windows (morning until 17:59, night 22:00–23:59).
- Admin Attendance can filter and Export CSV.
- Gate Pass generates a QR code and shows it in your history.