

ResHelp Project Summary

Project Manager: Ms Angelina Ramsunar

Project Overview

ResHelp is a cloud-hosted maintenance management system designed to simplify the process of reporting and resolving maintenance issues in university residences. The system replaces informal or manual reporting methods with a centralized digital platform where students can report issues, maintenance staff can manage repairs, and residence managers can monitor performance.

The platform improves communication, transparency, and efficiency by allowing users to submit maintenance requests online and track their progress in real time until they are resolved. The system is accessible through a web browser and is designed to work across multiple devices.

Target Users

The system supports three primary user roles.

Students

Students use the platform to report maintenance issues in their residence rooms or buildings. They can submit complaints, upload images, track issue progress, edit or cancel requests, and rate the service once the issue is resolved.

Maintenance Staff

Maintenance staff manage and resolve reported issues. They can view new requests, accept assignments, update the repair status, and mark issues as resolved. They can also access student contact information when needed.

Residence Managers / Administrators

Managers oversee the entire maintenance process. They can monitor issue trends, track staff performance, manage users, and generate reports through the analytics dashboard.

Main System Features

Issue Reporting

Students can submit maintenance requests by providing:

- Issue category (e.g., electrical, plumbing)
- Priority level
- Location of the issue
- Title and detailed description
- Optional photo upload
- Urgent hazard flag

Issue Tracking

The system tracks maintenance requests through four status stages:

- **Pending** – Issue reported but not yet assigned

- **Assigned** – Maintenance staff accepted the task
- **In Progress** – Repairs are being performed
- **Resolved** – Issue has been fixed

Feedback System

Once a repair is completed, students can rate the service to provide feedback on maintenance quality.

Analytics and Reporting

Residence managers can access analytics dashboards showing:

- Monthly issue trends
- Issue categories
- Average resolution times
- Staff performance
- Building maintenance statistics

Reports can be exported for further analysis.

System Architecture

ResHelp uses a layered architecture consisting of frontend, backend, database, and cloud infrastructure.

Frontend

The frontend provides the user interface, including dashboards, navigation menus, and issue reporting forms.

Technologies used:

- HTML
- CSS
- JavaScript

The interface is mobile-responsive for use on smartphones and tablets.

Backend

The backend manages application logic, authentication, and communication with the database.

Technologies used:

- Node.js
- Python
- RESTful APIs

Database

System data is stored in a relational database.

Supported databases:

- PostgreSQL
- MySQL

The database stores users, maintenance requests, issue statuses, and feedback data.

Cloud Infrastructure

The system is hosted on cloud platforms such as:

- Microsoft Azure
- Amazon Web Services (AWS)

Cloud hosting provides scalability, automated backups, and high availability.

Technology Stack

The core technologies used in the project include:

Frontend

- HTML
- CSS
- JavaScript

Backend

- Node.js
- Python

Database

- PostgreSQL or MySQL

Cloud Hosting

- AWS or Microsoft Azure

Authentication

- OAuth 2.0
- JSON Web Tokens (JWT)

Monitoring

- AWS CloudWatch or Azure Monitor

System Requirements

To run the system, the following requirements are needed.

Hardware

- Dual-core CPU
- 4GB RAM
- 10GB disk space

Software

- Node.js v18 or higher
- Python 3.10 or higher

- PostgreSQL or MySQL database

Browser Support

- Google Chrome
- Mozilla Firefox
- Microsoft Edge
- Apple Safari

Installation and Setup

To run the system locally:

1. Download the Project

Clone the repository from GitHub and navigate to the project directory.

2. Install Dependencies

Install backend dependencies using:

- `npm install` for Node.js packages
- `pip install -r requirements.txt` for Python packages

3. Configure Environment Variables

Create a `.env` file containing configuration values such as:

- Database connection URL
- JWT secret key
- Cloud provider

4. Initialize the Database

Run the database initialization script to create required tables.

5. Start the Application

Launch the server and open the application in a browser at:

`http://localhost:3000`

Security Features

The system includes several security measures:

- SSL encryption for secure data transmission
- Role-based access control
- Password hashing using `bcrypt`
- Secure authentication using OAuth 2.0 or JWT
- Daily automated backups with 30-day retention

Users are encouraged to log out after sessions to maintain security.

Deployment and Maintenance

The system is developed and deployed using a structured process:

1. Project planning and scope definition
2. Development of frontend, backend, and database
3. Testing (unit, integration, and security testing)
4. Cloud deployment and system launch
5. User training and documentation
6. Ongoing technical support

Maintenance activities include monthly system updates, quarterly feature improvements, and annual security audits.

Risk Management

Potential system risks include data loss, system failure, and low user adoption.

Mitigation strategies include:

- Secure backups and recovery systems
- Cloud redundancy and load balancing
- User training and feedback collection