CompSci Journal
Double Blind Peer Reviewed, Open Access International Research Journal.
ISSN: 3067-3089| Website: https://csjour.com | Volume-1, Issue-1 | February - 2025

Original Article

E-Gate Pass System: A Smart Pass Management Solution for Educational Institutions

Udaykumar Phule[1], Shraddha Kulkarni[2], Saish Kolhapure[3], Madhav Kamble[4], Sarthak Nande[5]
[1]*Guide Department of computer Science and Engineering, SKN Sinhgad College of Engineering, Korti, Pandharpur, India*
[2,3,4,5]*Student, Department of computer Science and Engineering, SKN Sinhgad College of Engineering, Korti, Pandharpur, India*

Manuscript ID: CSJ-2025-010105

Volume 1

Issue 1

Pp. 26-36

February 2025

ISSN: 3067-3089

Submitted: 15 Jan. 2025

Revised: 20 Jan. 2025

Accepted: 15 Feb. 2025

Published: 28 Feb. 2025

Correspondence Address:
Udaykumar Phule, Guide Department of computer Science and Engineering, SKN Sinhgad College of Engineering, Korti, Pandharpur, India
Email: udaykumar.phule@sknscoe.ac.in

Quick Response Code:

Web. https://csjour.com/

DOI: 10.5281/zenodo.17709658

DOI Link: https://doi.org/10.5281/zenodo.17709658

Abstract

What is REST API?

The e-Gate Pass System is a digital platform designed to automate and streamline the student leave pass process within educational institutions. It incorporates a multi-level authorization framework that involves students, guardians, teachers, hostel rectors, and security personnel. By utilizing modern web technologies such as React Native, Node.js, and MongoDB, the system provides real-time updates and ensures secure approvals. The paper discusses the system architecture, methodology, the use of JWT for secure authentication, and Blowfish encryption for safeguarding sensitive data. Additionally, the study outlines the challenges encountered during implementation and proposes future enhancements for scalability and usability.

Keywords: *Gate Pass System, multi-level authorization, REST API, MVC, JWT, Blowfish algorithm, React Native, Node.js, MongoDB, educational institutions.*

Introduction

Traditional pass management in educational institutions often relies on manual processes, which are prone to delays, errors, and inefficiencies. These systems lack transparency and security, making it difficult to track students' movements and approvals.

The *e-Gate Pass System* addresses these issues by providing a digital platform where students can request leave passes online. The system allows for multi-level authorization and real-time updates, improving security and efficiency. It is particularly useful in managing pass requests for students staying in hostels or campus accommodations, as it provides accurate location tracking and clear communication among students, faculty, and guardians.

a) **What is MVC (Model-View-Controller)?**

The MVC architecture separates the data (Model), the user interface (View), and the business logic (Controller). In our system, this design pattern is implemented to ensure that various parts of the application remain loosely coupled, enhancing scalability and maintainability.

REST API is used for communication between the client (frontend) and server (backend). In the e-Gate Pass System, the REST API manages requests between the user interface and the backend, ensuring data is retrieved and processed efficiently.

Methodology:

The **e-Gate Pass System** follows a structured development process to ensure reliability and scalability. This includes:

a) *Architecture Design*

The system follows an MVC architecture where:

- The Model handles the data logic and database queries.
- The View represents the user interface components, designed using React Native.
- The Controller processes the incoming requests via Node.js and returns the appropriate responses.

How to cite this article:
Phule, U., Kulkarni, S., Kolhapure, S., Kamble, M., & Nande, S. (2025). E-Gate Pass System: A Smart Pass Management Solution for Educational Institutions. CompSci Journal, 1(1), 26–36. https://doi.org/10.5281/zenodo.17709658

CompSci Journal
Double Blind Peer Reviewed, Open Access International Research Journal.
ISSN: 3067-3089 Website: https://csjour.com | Volume-1, Issue-1 | February - 2025

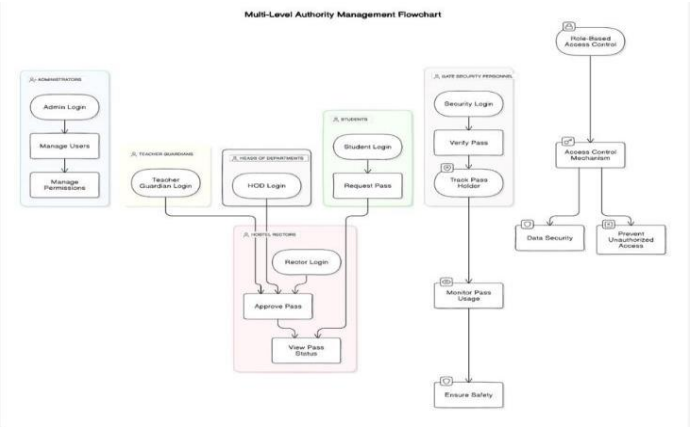


Fig 1. Flowchart for e-Gate Pass

b) **Security Implementation**

To secure user data and the approval process, the system uses:

- **JWT (JSON Web Token)**: For authentication and authorization purposes. JWT tokens are generated when users log in, ensuring that only authorized personnel can access and approve requests.
- **Blowfish Algorithm**: This encryption technique is used to encrypt sensitive information such as student records and approval data, safeguarding against unauthorized access.

c) **User Interface Design**

The user interface (UI) of the e-Gate Pass System was built with React Native for a seamless mobile experience. This ensures that both students and faculty members can interact with the system efficiently, with intuitive navigation for submitting or approving leave requests.

d) **Future Enhancements**

Integration with Biometric Systems: For added security, biometric authentication (like fingerprint or facial recognition) can be integrated into the system.

AI-based Analytics: To provide insights into patterns of leave requests, allowing educational institutions to monitor and manage student attendance more effectively.

Multilingual Support: Adding support for different languages to cater to diverse user groups.

Algorithms

A. JWT (JSON Web Token) Algorithm

JWT is used for securing the user authentication process. It works as follows:

1. *User logs in and submits credentials.*
2. *If valid, the server generates a JWT that includes user information and a signature to prevent tampering.*
3. *The token is sent to the user and stored client-side.*
4. *For future requests, the client sends the token to the server, which verifies the signature and grants access based on the encoded data.*

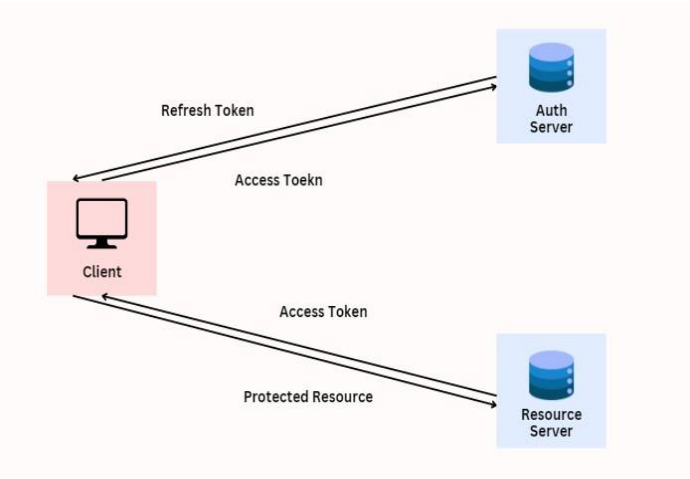


FIG 2. JWT Mechanism with Refresh Token

B. Blowfish Encryption Algorithm

CompSci Journal
Double Blind Peer Reviewed, Open Access International Research Journal.
ISSN: 3067-3089 Website: https://csjour.com | Volume-1, Issue-1 | February - 2025

Blowfish is a symmetric key encryption algorithm used to encrypt sensitive data. It works in blocks of 64 bits:

1. The plaintext is divided into blocks.
2. The key is applied to the data in multiple rounds (often 16) using complex key scheduling.
3. The encrypted ciphertext is produced, which can only be decrypted using the same secret key.

I. Scalable JWT Authentication Structure

To build a scalable JWT authentication structure, follow these components:

1. **Access Token (Short-Lived Token):**

- **Purpose**: Secure communication for short periods.
- **Expiration**: Usually 15 minutes.
- **Storage**: Stored in memory or a secure place like HttpOnly cookies to prevent XSS attacks.

2. Refresh Token (Long-Lived Token):

- **Purpose**: To issue new access tokens without re-authenticating the user.
- **Expiration**: A longer duration (e.g., 7 days or more).
- **Storage**: Stored securely, such as in HttpOnly cookies.

3. Refresh Token Strategy:

- When the access token expires, the client can use the **refresh token** to request a new access token from the server.
- The refresh token is sent to a dedicated endpoint (/token/refresh) to issue a new JWT access token.
- The new access token is issued, and the refresh token is either renewed or used repeatedly until its expiry.

4. Security Considerations:

- Use **HttpOnly cookies** for storing refresh tokens to mitigate XSS attacks.
- Implement **rate-limiting** on the refresh token endpoint to prevent abuse.
- Revoke refresh tokens when users log out or when malicious activity is detected (token invalidation).

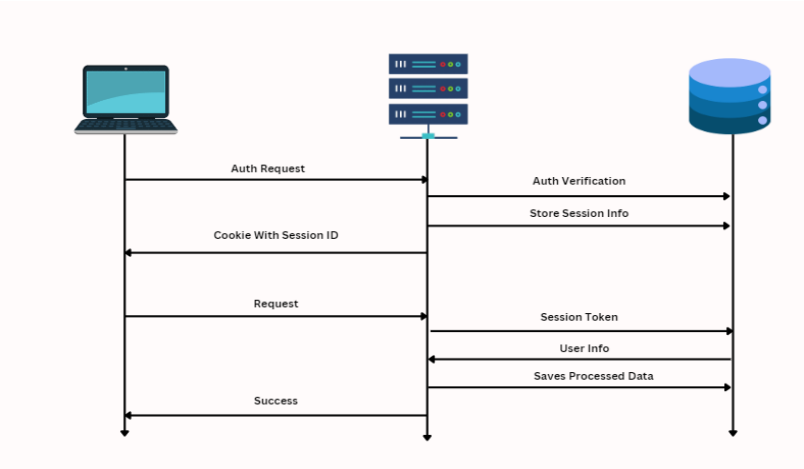


Fig 3. Scalable JWT Structure

Result And Analysis

The e-Gate Pass System was tested in a controlled environment within a campus to analyze its effectiveness and performance. The results show a significant improvement in the efficiency of leave request approvals and the security of the overall process. The real-time notifications and secure access for all parties (students, faculty, guardians) streamlined the operations, reducing the manual overhead typically involved in such procedures.

The JWT-based authentication was particularly effective in preventing unauthorized access, and Blowfish encryption provided robust protection for sensitive student information. However, there were challenges regarding user adoption and technical hiccups during peak loads, which can be addressed in future updates.

Conclusion

The e-Gate Pass System successfully automates the process of managing leave passes in educational institutions, offering a secure, scalable, and efficient alternative to traditional methods. With multi-level authorization, real-time notifications, and robust security measures in place, it significantly reduces the chances of manual errors while enhancing transparency. Future enhancements will focus on integrating AI analytics and biometric authentication to further improve the system's functionality.

Redux Architecture for State Management

CompSci Journal
Double Blind Peer Reviewed, Open Access International Research Journal.
ISSN: 3067-3089 Website: https://csjour.com | Volume-1, Issue-1 | February - 2025

Redux is a state management library widely used in JavaScript applications to handle the global state of an application in a predictable manner. It helps in maintaining a centralized store, enabling smooth data flow across components, making the application easier to debug and manage.

Key Components of Redux Architecture

1. **Store**:

- The store holds the entire state of the application.
- It is a single JavaScript object that serves as the **source of truth** for the application state.
- The store is immutable, meaning you never modify the state directly; instead, you dispatch actions that describe changes.

```
import { createStore } from 'redux';
const store = createStore(reducer);
```

2. **State**:

- The state in Redux represents the application's global data structure.
- It's maintained in the store and can be accessed by any component in the app.
- Every change to the state goes through the store, ensuring a single source of truth.

3. **Actions**:

- Actions are plain JavaScript objects that describe the changes you want to make to the state.
- They always have a type field and optionally other data fields (called payload) that are needed to update the state.
- They are dispatched from UI components or middleware to inform the store of state changes.

```
const action = {
  type: 'INCREMENT',
  payload: { value: 1 }
};
```

4. **Reducers**:

- A reducer is a pure function that takes the current state and an action, and returns the new state.
- The reducer does not modify the original state but returns a new copy of the updated state.
- The store uses reducers to decide how the state should be updated based on the action it receives.

5. **Dispatch**:

- Dispatch is the mechanism to send actions to the Redux store.
- When an action is dispatched, the store calls the appropriate reducer(s) to update the state.
- Dispatching is typically triggered by user interaction or middleware.

```
store.dispatch({ type: 'INCREMENT', payload: { value: 1 } });
```

6. **Selectors**:

- Selectors are functions used to extract specific pieces of state from the store.
- They are used to keep components decoupled from the structure of the state tree.

```
const selectCount = (state) => state.count;
```

7. **Middleware**:

- Middleware in Redux intercepts actions before they reach the reducer. It can be used for logging, asynchronous calls (e.g., API requests), or other side effects.
- Popular middleware includes redux-thunk and redux-saga for handling asynchronous actions.

```
import thunk from 'redux-thunk';
const store = createStore(reducer, applyMiddleware(thunk));
```

Redux Workflow

The flow of data in Redux follows a strict unidirectional pattern:

1. **View (UI Components)**: User interactions (e.g., button clicks) trigger action dispatches.
2. **Action**: An action is dispatched, describing the type of change requested.
3. **Reducer**: The reducer receives the current state and the action, then computes and returns the next state.
4. **Store**: The store holds the new state, replacing the old one, and notifies subscribers (typically UI components).
5. **View (UI Components)**: The updated state is reflected in the UI, and the process can repeat

System Architecture for E-Gate Pass

The "e-Gate Pass System" utilizes a client-server architecture with a mobile-first approach, ensuring that users can interact with the system efficiently through a mobile application. The architecture is designed to facilitate secure, real-time pass management, featuring multiple user roles and levels of authority.

Here's an overview of the architecture:

1. Client Layer (Frontend - Mobile App)

- **Technology**: React Native
- **Purpose**: Provides a user-friendly interface for students, teachers, guardians, and security personnel to interact with the system.
- **Functions**:

Pass Request and Approval: Students can request passes, and authorities can approve or deny them.

- Real-Time Notifications: Users receive notifications about pass approvals, rejections, and expiration.
- Location Tracking: The app requests user permission to track real-time location, ensuring compliance and security.

- **Role-Based Access**: The app customizes views and permissions based on user roles (e.g., students only request passes, while teachers and security staff can approve).

2. Backend Layer (Server)

- **Technology**: Node.js (with Express.js framework)
- **Purpose**: Processes requests from the client, manages data interactions, and enforces business logic.
- **Functions**:

- Authentication & Authorization: Secure login with role-based access control (RBAC) to restrict functions based on roles.
- API Endpoints: Exposes RESTful APIs for client interactions, such as pass creation, approval, and status updates.
- Real-Time Data Processing: Processes real- time location data to monitor pass usage and flag suspicious activity.
- Notifications: Manages notification services for sending updates on pass requests, approvals, and expirations.

3. Database Layer (Storage)

- **Technology**: MongoDB
- **Purpose**: Stores pass data, user profiles, role permissions, and activity logs.
- **Functions**:

- Stores User Profiles: Includes information such as name, role, department, and contact details.
- Pass Management: Keeps records of pass requests, statuses, approvals, and expiry times.
- Location Logs: Stores real-time location data associated with active passes for tracking.
- Role-Based Permissions: Ensures each role has specific permissions and access to relevant data.

CompSci Journal
Double Blind Peer Reviewed, Open Access International Research Journal.
ISSN: 3067-3089 Website: https://csjour.com | Volume-1, Issue-1 | February - 2025

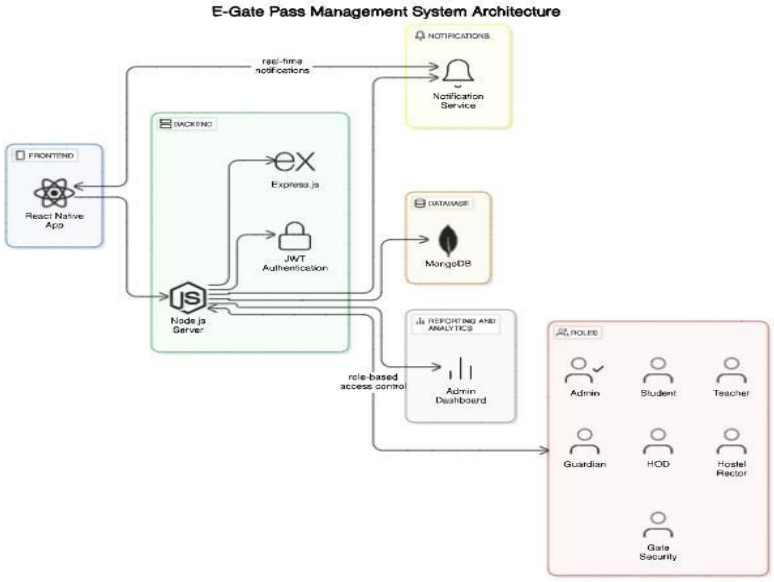


Fig 4. System Architecture For E-Gate Pass

4. Real-Time Location Tracking System

- **Purpose**: Enhances security by tracking the location of pass users in real time.
- **Components**:
  - **GPS Integration**: The mobile app integrates with the device's GPS to capture the user's location during pass usage.
  - **Location Data Processing**: The backend server periodically processes location data to ensure compliance and detect anomalies (e.g., unauthorized exit attempts).
- **Data Flow**:
  - The mobile app continuously sends GPS data to the backend while a pass is active.
  - The server verifies the data against approved exit points or allowed areas and flags any inconsistencies.

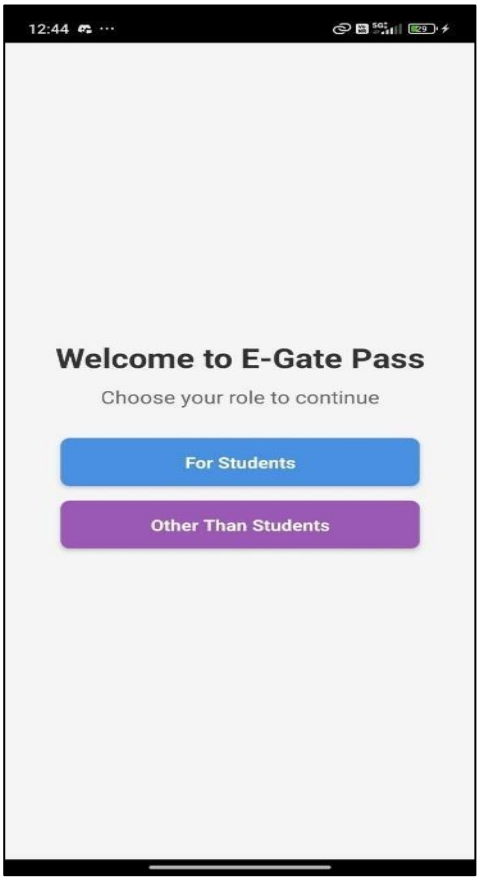
5. Role-Based Access Control (RBAC)

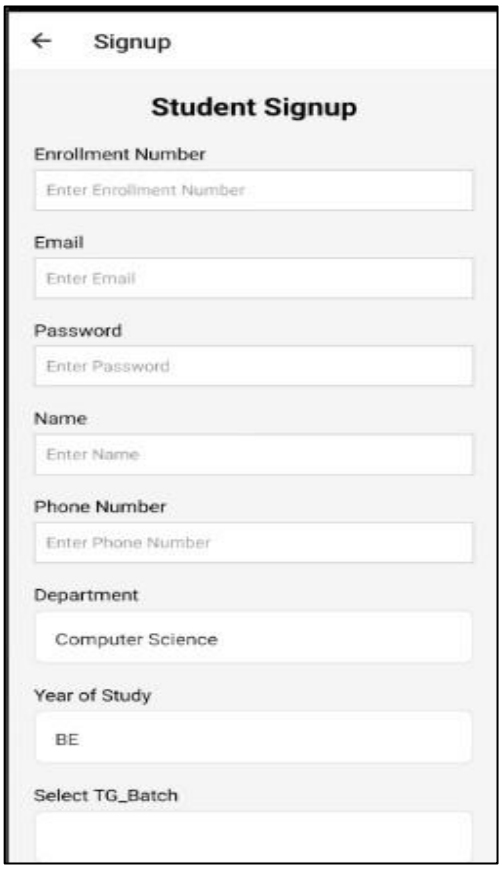
- **Purpose**: Manages permiss0ions and ensures that users only access features relevant to their roles.
- **Implementation**:
  - **Admin**: Full access to manage users, configure role permissions, and oversee pass issuance and approvals.
  - **Students**: Can only request passes and view status updates.
  - **Teachers/Guardians/HOD/Hostel Rector**: Review and approve/reject passes based on their level of authority.
  - **Gate Security**: Verifies passes at entry/exit points and checks for any restricted access.

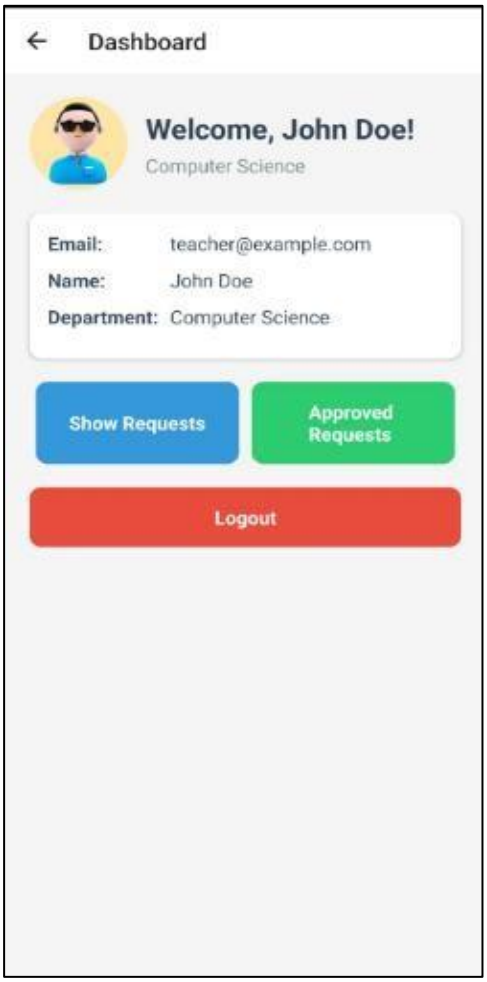
6. Flow of the System

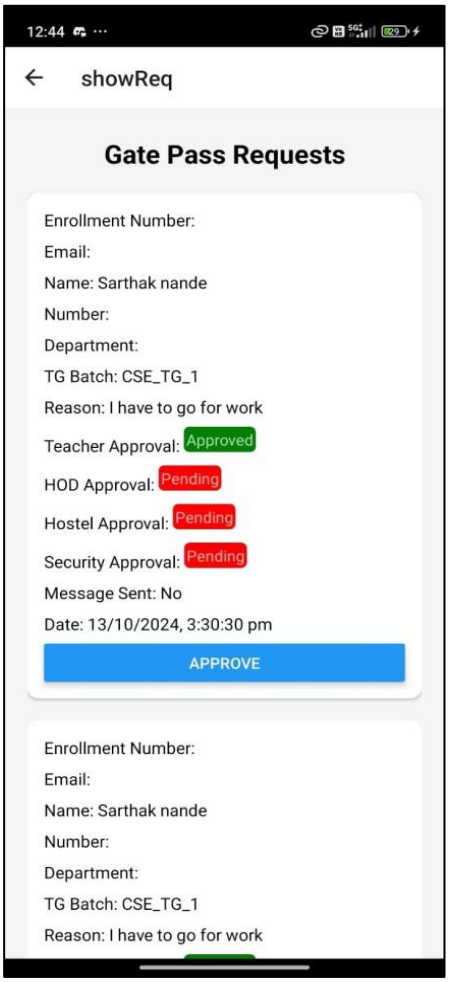
- **Pass Request**: Students submit pass requests via the mobile app.
- **Approval Workflow**: The request moves through a series of approvals based on the user's role (e.g., teachers, guardians, HODs).
- **Real-Time Location Verification**: The system tracks the student's location when the pass is active.
- **Security Verification**: Gate security verifies the pass on entry/exit, confirming user details and pass validity.

Result

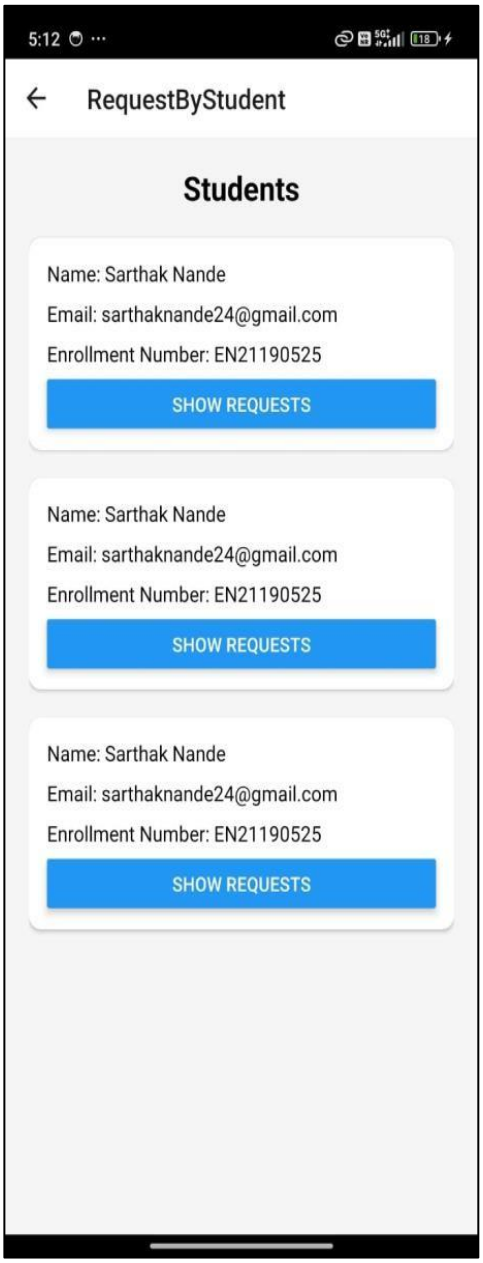

12:44
Welcome to E-Gate Pass
Choose your role to continue
For Students
Other Than Students


Signup
Student Signup
Enrollment Number
Enter Enrollment Number
Email
Enter Email
Password
Enter Password
Name
Enter Name
Phone Number
Enter Phone Number
Department
Computer Science
Year of Study
BE
Select TG_Batch

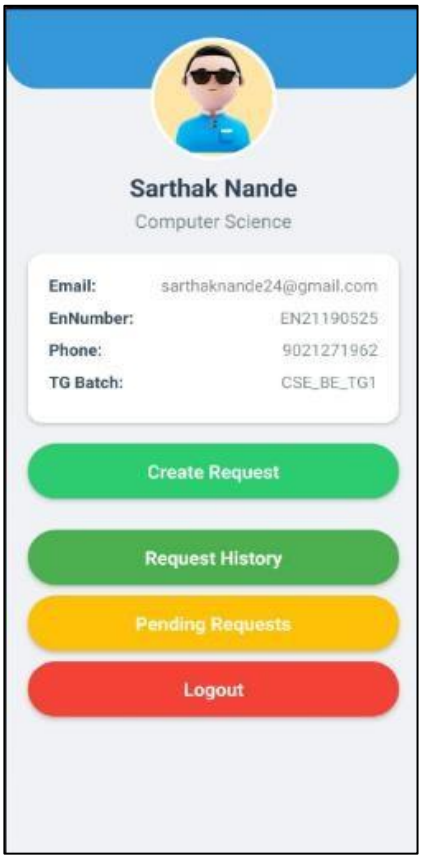

Dashboard
Welcome, John Doe!
Computer Science
Email: teacher@example.com
Name: John Doe
Department: Computer Science
Show Requests
Approved Requests
Logout

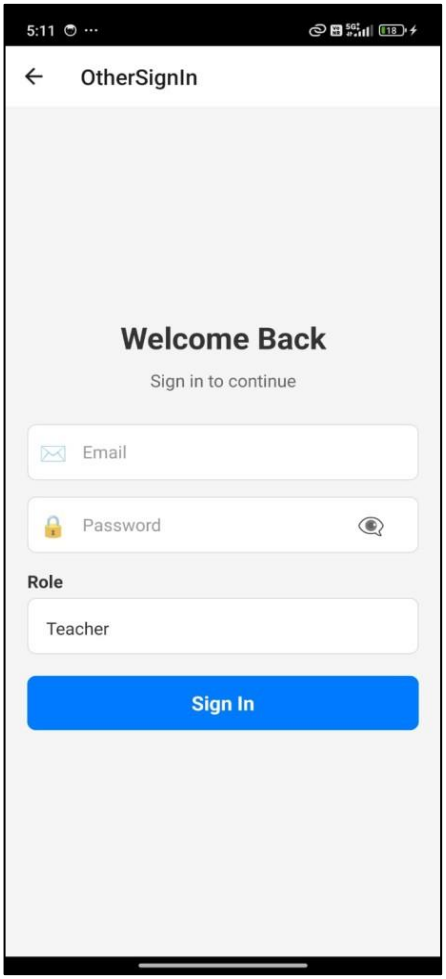

12:44
showReq
Gate Pass Requests
Enrollment Number:
Email:
Name: Sarthak nande
Number:
Department:
TG Batch: CSE_TG_1
Reason: I have to go for work
Teacher Approval: Approved
HOD Approval: Pending
Hostel Approval: Pending
Security Approval: Pending
Message Sent: No
Date: 13/10/2024, 3:30:30 pm
APPROVE
Enrollment Number:
Email:
Name: Sarthak nande
Number:
Department:
TG Batch: CSE_TG_1
Reason: I have to go for work

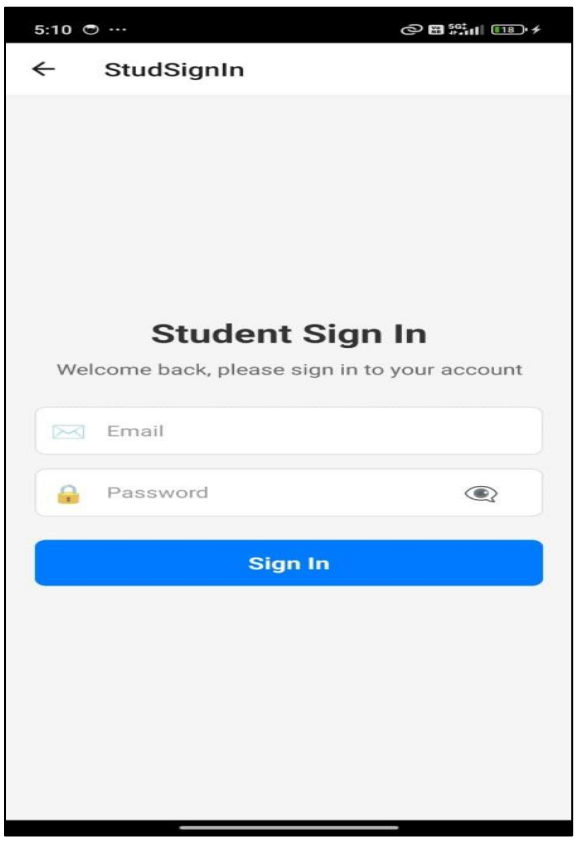
CompSci Journal
Double Blind Peer Reviewed, Open Access International Research Journal.
ISSN: 3067-3089 Website: https://csjour.com | Volume-1, Issue-1 | February - 2025



CompSci Journal
Double Blind Peer Reviewed, Open Access International Research Journal.
ISSN: 3067-3089 Website: https://csjour.com | Volume-1, Issue-1 | February - 2025







Acknowledgments
The authors thankful to Principal Dr. K. J. Karande, SKN Sinhgad College of Engineering, Pandharpur for granting permission to carry out the work.

Financial support and sponsorship
Nil.

Conflicts of interest
There are no conflicts of interest.

Reference
1. A. Noman, Z. Islam, and N. Noor, "Optical Character Recognition: A Survey and Techniques," Journal of Information & Communication Technology.
2. M. Yilmaz and Y. S. Akgul, "Mobile Turkish Scene Text-to-Speech System for the Visually Impaired," arXiv:1608.05054.
3. A. Esteva et al., "A Guide to Deep Learning in Healthcare," *Nature Medicine*, vol. 25, no. 1, 2019.
4. T. Chen and C. Guestrin, "XGBoost: A Scalable Tree Boosting System," ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2016.
5. B. Zhou et al., "Natural Language Processing for Smart Healthcare," October 2021.
6. "JWT Handbook: Learn JSON Web Tokens (JWT) in the context of building secure web applications," JWT.io Documentation.
7. B. Schneier, "The Blowfish Encryption Algorithm," Applied Cryptography, 2nd ed., John Wiley & Sons, 1996.