REVA UNIVERSITY

Bengaluru, India

SCHOOL OF C & IT

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING [ARTIFICIAL INTELIGENCE AND MACHINE LEARNING]

(Academic Year: 2024–2025)

DATABASE MANAGEMENT SYSTEM PROJECT ON

BUS PASS MANAGEMENT SYSTEM

Project Carried Out by:

Amogh M (R23EA020) / Ankit Kumar (R23EA021)

Archana C (R23EA022)

B Pavan (R23EA023) / B Sai Jitendra (R23EA024)

Abstract

The **Bus Pass Management System** is a web-based application developed to automate and simplify the process of issuing, managing, and verifying bus passes for daily, monthly, and yearly commuters. Traditional manual systems are often inefficient, time-consuming, and prone to errors. This project aims to replace such outdated methods with a digital solution that enhances transparency, reduces operational overhead, and improves the overall user experience.

Built using **Flask (Python)** for the backend and **HTML/CSS** for the frontend, the system provides a secure and responsive platform for both users and administrators. Commuters can register, log in, apply for passes, make payments, and view the status or history of their applications. On the administrative side, the system enables pass validation, route and fare management, and the generation of reports to support informed decision-making.

The backend leverages a **relational database (SQLite)** to store and manage data efficiently. The database is designed using normalization and follows core DBMS concepts such as ER modelling, CRUD operations, JOINs, and data consistency. The platform ensures user authentication and session-based access to maintain security and restrict unauthorized actions.

To improve usability, the interface is designed to be clean, intuitive, and responsive across devices, including desktops and smartphones. The modular architecture of the system allows for future scalability, such as integrating SMS/email notifications, adding more cities or transit zones, and connecting with external payment gateways or APIs for real-time validation.

Overall, this project not only demonstrates the practical implementation of web development and database management techniques but also offers a scalable, eco-friendly, and user-centric alternative to traditional paper-based bus pass systems. It improves accessibility, reduces fraud, supports sustainable public transportation management, and sets the foundation for future innovations in smart commuting solutions.

Table of Contents

Introduction

In today's fast-paced world, efficient transportation systems are vital for ensuring smooth and convenient travel for millions. Among these, buses remain a crucial mode of public transport, serving commuters, students, and workers. However, managing bus passes using traditional paper-based methods often results in inefficiencies, delays, and administrative burden.

To overcome these challenges and introduce a more streamlined approach, we present the **Bus Pass Management System**—a web-based application designed to digitize and automate the entire lifecycle of bus passes. Developed using **Flask (Python)** for the backend and **HTML/CSS** for the frontend, the system enables quick issuance, tracking, and renewal of passes through a secure and user-friendly interface.

By integrating a centralized **SQLite** database, the system allows administrators to efficiently manage user records, route details, and payment information while minimizing manual tasks. Key modules include user registration and login, pass application, payment integration, and status monitoring. Additionally, administrative tools are provided for verifying applications, approving/rejecting passes, updating route fares, and generating analytical reports.

Passengers can easily register, apply for new passes, view active or expired passes, and complete payments online—eliminating the need for physical forms, cash handling, and long queues at counters. The platform ensures secure session-based access and includes basic validation checks to enhance system integrity and user data safety.

This project showcases how modern web and database technologies can enhance the transparency, accuracy, and convenience of public transportation systems. It delivers a more connected, eco-friendly, and efficient experience for both administrators and commuters. Moreover, the system's modular design allows for future expansion—such as SMS/email alerts, QR-based verification, multi-city support, and integration with mobile apps—making it a scalable solution for smart city initiatives.

System Analysis

1.Problem Identification

In traditional bus pass management systems, most processes are handled manually—ranging from user registration and form submissions to maintaining pass records and verifying applications. These paper-based systems are prone to human error, delays, loss of data, and inefficient handling of high commuter volumes. Administrators face challenges in tracking pass renewals, validating applications, and ensuring data consistency across records. Users, on the other hand, experience inconvenience due to long queues, repeated form-filling, and lack of real-time updates.

2. Objective of the System

The primary objective of the Bus Pass Management System is to provide a digital, automated solution that simplifies the process of applying for, issuing, managing, and renewing bus passes. The system aims to reduce the administrative burden, improve data handling, and deliver a seamless experience to both users and administrators. It enables users to access all services online—from registration and application to payment and status tracking—thus ensuring efficiency, accessibility, and transparency.

3. Feasibility Study

The development and deployment of this system are technically and economically feasible. Technically, the system is built using Python Flask, SQLite, and standard web technologies like HTML and CSS, all of which are open-source and lightweight. Operationally, it is user-friendly, requiring minimal training for both commuters and administrators. Economically, the system is cost-effective to implement and maintain due to the use of free tools and frameworks. Hence, the solution is practical, scalable, and sustainable for long-term use.

4. Existing System vs Proposed System

The existing manual system is inefficient, with redundant paperwork, physical form submissions, and delayed processing. Records are difficult to retrieve and prone to loss or duplication. In contrast, the proposed digital system automates these processes. Users can register and apply online, administrators can view and process applications in real time, and all data is stored securely in a centralized database. This shift significantly improves accuracy, processing time, and user satisfaction.

5. Functional Requirements

The system must support multiple core functions including user registration, secure login, route selection, bus pass application, pass status tracking, and payment processing. For administrators, it must include pass validation, route management, viewing user data, and generating reports. These functional modules ensure smooth operation and deliver a comprehensive solution for end-to-end bus pass management.

6. Non-Functional Requirements

Beyond functionality, the system must meet non-functional expectations like performance, reliability, and security. It should offer fast response times for queries and updates, maintain data integrity even during multiple simultaneous user sessions, and ensure secure handling of sensitive information such as login credentials and payment details. Additionally, the system should be easy to maintain and scalable to support additional routes or pass types in the future.

7. User Roles

The system defines two main user roles: **Users (Commuters)** and **Administrators**. Users can create accounts, log in securely, apply for a pass, make payments, and view or renew their passes. Administrators have extended privileges, allowing them to view all pass applications, approve or reject them, manage routes and fare details, and monitor system usage. Role-based access ensures that only authorized users perform critical operations, enhancing system security and control.

System Design

1. Design Objectives

The goal of the system design is to build a robust, efficient, and scalable bus pass management solution that meets the requirements of both end users and administrators. The design focuses on modularity to allow ease of updates and maintenance, separation of concerns for cleaner architecture, and a responsive user interface to ensure accessibility across devices. The system should be simple for users to navigate, while being powerful enough to handle administrative operations like pass verification, data management, and report generation.

2. System Architecture

The Bus Pass Management System follows a **client-server architecture**. The client side (frontend) is responsible for user interaction and is built using HTML, CSS to ensure responsive design and usability. The server side (backend) is developed using the Flask web framework in Python, which handles routing, session management, and business logic. The system communicates with an SQLite database to store and retrieve data related to users, routes, passes, and transactions. This three-layer architecture (presentation, logic, and data) ensures clear separation and efficient handling of requests.

3. Modules of the System

The system is divided into key functional modules, each responsible for specific tasks:

- **User Module**: Handles registration, login, profile management, and pass application.
- **Admin Module**: Allows administrators to review applications, approve or reject passes, manage routes, and oversee payments.
- **Pass Management Module**: Manages the creation, validation, and expiry of passes. Each pass is time-bound and linked to a user and route.
- **Payment Module**: Tracks payment status associated with each pass application and stores transaction records for auditing.

This modular approach improves clarity in development, testing, and future enhancements.

4. Database Design

The backend uses a **relational database (SQLite)** which is lightweight and well-suited for small to medium-scale web applications. The key tables include:

- **users**: Stores user details like name, email, and password.
- **routes**: Stores origin, destination, and fare information.
- **bus_passes**: Stores issued pass details including user ID, route ID, issue date, valid until, and status.
- **payments**: Records transaction details such as amount, date, and associated pass ID.
- **admins**: manages and monitors bus pass applications by viewing user requests, updating their status

These tables are connected via **foreign key relationships**, ensuring data integrity. For example, each entry in bus_passes links to a valid user and route, while payments reference a specific bus pass. The schema design supports normalized data and avoids redundancy.

5. User Interface Design

The UI is designed with simplicity and usability in mind. Users can easily navigate between pages such as registration, login, pass application, and pass status. Forms include necessary validations to ensure proper data entry. Administrators access their dashboard through a separate login and can view organized tables of pending applications, approved passes, and user records. Bootstrap is used for styling, ensuring the application is responsive across desktops, tablets, and mobile devices.

6. Flow of Application

The user flow is intuitive and efficient:

1. User registers and logs into the system.
2. User views available routes and selects one to apply for a bus pass.
3. The system checks for existing valid passes and prevents duplicates.
4. A new pass is generated with calculated validity.
5. User is redirected to the payment page, and the transaction is recorded.
6. Admin later verifies applications and updates pass status.
7. Users can log in anytime to view their pass.

This logical flow ensures both users and admins can interact

Implementation

The implementation of the Bus Pass Management System involved developing a full-stack web application with a strong emphasis on security, usability, and data management. The backend was implemented using Flask, a lightweight Python web framework that allowed for rapid development of routes, session handling, and database interaction. The frontend was built using HTML, CSS, and Bootstrap to create a clean, responsive, and user-friendly interface suitable for both desktop and mobile users.

The application flow begins with a user authentication system, where users can register and log in using valid credentials. Session-based authentication ensures that only logged-in users can access protected routes, such as applying for a pass or viewing pass status. The system validates user inputs and prevents duplicate registrations, enhancing the reliability of stored user data.

Once logged in, users can access the pass application module, where they choose a route from a dynamically populated dropdown sourced from the database. The system checks for any existing active pass to prevent duplication before issuing a new one. A validity period of 30 days is automatically assigned upon application, and the user is redirected to the payment interface.

The payment module records payment details and associates them with the issued bus pass. Administrators can log in through a separate portal to verify applications, manage routes, monitor transactions, and generate reports. This dual-interface structure separates user and admin functionality to ensure clarity and security in operations.

On the backend, a SQLite database handles data storage, employing foreign keys to maintain relational integrity among tables like users, routes, bus_passes, and payments. CRUD operations were implemented through parameterized SQL queries to reduce the risk of SQL injection. Data normalization was applied to optimize storage and ensure consistency.

In terms of frontend implementation, Bootstrap was extensively used to style forms, buttons, and layouts, making the platform responsive and visually consistent. Validation messages, alerts, and interactive elements enhance user experience and guide the flow of actions clearly.

Overall, the implementation phase brought together multiple technologies in a cohesive manner to deliver a digital solution that is practical, secure, and scalable for managing public bus passes

Testing

Testing is a crucial phase in software development that ensures the system performs as expected, meets end-user requirements, and is free from major defects. The **Bus Pass Management System** underwent several levels of testing to validate its reliability, accuracy, and usability across different user roles—commuters and administrators.

1. Types of Testing Performed

a) Unit Testing

- Focused on testing individual modules and backend logic such as:
    1. User registration and login validation
    2. Pass application logic and expiry date calculation
    3. Payment processing and database insertion
    4. Admin verification and status updates
- Flask routes and core Python functions were tested using mock data.

b) Integration Testing

- Verified communication between system components, including:
    1. Interaction between Flask routes and the SQLite database
    2. Session handling between login and secure routes
    3. Data flow from frontend forms to backend processing and DB updates

c) Functional Testing

- Confirmed that each functional requirement was met:
    1. User registration and authentication
    2. Pass application and status tracking
    3. Payment linking with pass ID
    4. Admin dashboard functionalities like approving or rejecting applications

d) User Interface Testing

- Ensured frontend elements worked across browsers and screen sizes:
    1. Correct rendering of forms and tables using Bootstrap
    2. Usability of dropdowns, buttons, and links
    3. Feedback messages such as “Application submitted” or “Already applied” displayed properly

e) Security Testing

- Tested for vulnerabilities and access control:
    1. Input fields validated to prevent SQL injection
    2. Session-based access control for protecting admin pages
    3. Passwords securely stored using hashing (if applicable)
    4. Prevented re-application for active routes using backend checks

2. Test Cases and Scenarios

Test Case	Description	Expected Result	Status
TC01	User registers with valid details	Account created and redirected to login	Pass
TC02	User applies for pass with valid input	Pass entry created in database and redirected to payment	Pass
TC03	Same user re-applies for same route on same day	Warning message shown: “Already applied”	Pass
TC04	Admin logs in and views pending applications	List of pass applications displayed	Pass
TC05	Unauthenticated user accesses /apply route	Redirected to login page	Pass
TC06	Invalid route or pass ID in URL	Handled gracefully with error message	Pass

3. Bug Fixes and Iteration

During the testing phase, the following issues were encountered and resolved:

- **Incorrect redirect after login** – Fixed session logic to ensure users return to the intended page
- **Duplicate pass creation** – Added logic to check for existing active pass before insertion
- **Flash messages not displaying consistently** – Modified HTML templates to ensure proper alert rendering
- **Form validations missing for route selection** – Added required attributes and backend checks
- **Page layout issues on mobile** – Refined Bootstrap classes to improve mobile responsiveness

Each issue was resolved and re-tested to ensure proper functionality and user experience.

Result and Discussion

The implementation of the **Bus Pass Management System** successfully fulfilled its intended objectives by replacing manual procedures with a centralized, automated platform. The system improved the efficiency of issuing and managing bus passes while ensuring greater accuracy, accessibility, and user satisfaction.

Key Outcomes (Results)

- **Streamlined Application Workflow -** Users were able to register, log in, and apply for bus passes through a guided interface. The application flow was tested to ensure data consistency and proper validation at each step, reducing the chance of duplicate or invalid submissions.

- **Centralized Data Handling -** All user, route, and application information was maintained within a normalized relational database, ensuring reliable data storage and easy retrieval. The use of SQL queries enabled smooth integration between user actions and backend processing.

- **Administrator Efficiency -** The admin interface allowed for seamless management of user applications, route data, and approval workflows. Pass verification, status updates, and tracking could be done without delays, minimizing administrative overhead.

- **User Convenience and Accessibility -** The web-based nature of the system ensured users could access it at any time from any device. This eliminated the need for physical visits, long queues, and paper-based submissions, making the process more commuter-friendly.

Discussion Points

- **System Performance and Reliability -** During testing, the system handled user interactions efficiently with no major failures. Each module performed its function as intended, and no critical issues were encountered during form submission, data updates, or session handling.

- **UI and Responsiveness -** While the current design is functional and responsive, enhancements using UI frameworks like Bootstrap or Tailwind could improve visual appeal and user experience, especially on mobile devices.

- **Scalability Considerations -** The system is currently designed for smaller operational scopes (e.g., a single organization or locality). With appropriate database upgrades and optimization, it can be scaled to support multiple regions, more users, and concurrent administrative roles.

- **Security Measures -** The application ensures basic protection through session handling and input validations. However, incorporating advanced security layers such as hashed credentials, account recovery via email, and restricted access controls for sensitive operations can further strengthen the system.

- **Future Expansion Possibilities -** Future iterations could include automated renewal reminders, downloadable e-passes with QR codes for onboard scanning, and integration with payment gateways for a smoother payment process.

- **Deployment Readiness -** Although currently run on a local server, the system is capable of being deployed to a cloud platform with HTTPS support, domain configuration, and regular database backups for real-world usage.

Conclusion

The Bus Pass Management System marks a significant step toward modernizing the traditional, manual process of issuing and managing bus passes. Designed as a web-based platform using Flask (Python) for backend logic, HTML/CSS (with optional Bootstrap) for the frontend, and SQLite for the database, this system streamlines operations for both users and administrators through automation and centralized data handling.

From user registration to pass application, validation, and payment redirection, every step has been carefully integrated to reduce paperwork, human error, and administrative effort. The application supports session-based access, ensuring secure login and user-specific dashboards. The use of SQL queries and proper database normalization ensures data consistency, integrity, and efficient query performance.

For commuters, the system offers a more accessible and time-saving experience by allowing them to apply for and monitor the status of their bus passes online—eliminating the need for long queues and manual form submissions. For administrators, the system simplifies management tasks such as verifying applications, handling route information, and tracking issued passes. It also allows for future data analysis through reports and logs.

From an academic and technical standpoint, this project effectively demonstrates key database management principles such as ER modelling, CRUD operations, joins, and authentication handling. It also serves as a practical implementation of full-stack development, showcasing how front-end interfaces and back-end logic interact in real-world applications.

Furthermore, the system is built with scalability in mind. While currently suited for small to medium operations (e.g., educational campuses or small transit departments), it can be easily expanded with features like QR code-based digital passes, third-party payment gateway integration, notification systems, and advanced analytics dashboards.

In conclusion, the Bus Pass Management System is not just a solution to a common logistical problem—it is a platform that enhances transparency, improves operational efficiency, supports eco-friendly practices by reducing paper usage, and provides a foundation for future digital transformation in public transportation systems.

Future Enhancements

- **Integration of Multiple Payment Gateways** - Support for various online payment options such as credit/debit cards, mobile wallets, and UPI to provide greater flexibility to users.
- **Digital Pass with QR Code or Barcode** - Implementing digital passes embedded with QR codes or barcodes that can be scanned on buses for quick and contactless verification.
- **Automated Renewal and Expiry Notifications** - Adding email and SMS alerts to notify users about upcoming pass expirations or renewals to improve convenience.
- **Role-Based Access Control with More Granularity** - Enhancing the admin panel with different roles (e.g., super admin, route manager) to better manage permissions and responsibilities.
- **Mobile Application Development** - Creating a dedicated mobile app for Android and iOS to increase accessibility and improve user engagement.
- **Advanced Reporting and Analytics** - Incorporating dashboards with real-time statistics on pass usage, revenue, route popularity, and user demographics for data-driven decision making.
- **Integration with Public Transport Systems** - Linking the system with GPS or smart ticketing systems used by buses to provide real-time tracking and usage analytics.
- **Support for Multiple Languages** - Making the system multilingual to serve a diverse user base and improve inclusivity.
- **Enhanced Security Features** - Implementing two-factor authentication (2FA), CAPTCHA during registration, and encrypted data storage for heightened security.
- **Offline Mode for Rural Areas** - Developing an offline mode or kiosk-based system for areas with limited internet connectivity, syncing data when connectivity is restored.

Thank You