Project Report: Cloud-Based Bus Pass System

Introduction

The *Cloud-Based Bus Pass System* is a web application developed using **Python Flask framework** with **MySQL database** for storage. The system allows users to apply for a new bus pass, renew an existing one, and manage their details securely. The backend Flask application is deployed on an **AWS EC2 instance (Ubuntu server)**, while the database is hosted on **AWS RDS (MySQL)**. Both services are connected securely through a **VPC (Virtual Private Cloud)**.

The application is accessed using the **public IP address of the EC2 instance**. All user and application data is stored in the **RDS database** in AWS. This setup ensures **scalability, cloud integration, and secure data storage**.

Objectives

- To design and deploy a **cloud-based bus pass management system**.
- To enable **user registration, authentication, pass issuance, and renewal**.
- To provide an **admin dashboard** for monitoring applications and generating reports.
- To implement the system on **AWS using EC2, RDS, and VPC**.

Technologies & Tools Used

- **Programming Language:** Python
- **Framework:** Flask (Micro web framework)
- **Database:** MySQL (hosted on AWS RDS)
- **Cloud Platform:** AWS (Amazon Web Services)
    - EC2 (Ubuntu Server) → Application Hosting
    - RDS (MySQL) → Database Management
    - VPC → Networking and security
- **Deployment Tool:** Git Bash (for SSH, SCP, and deployment commands)
- **Web Server:** Flask built-in server / Gunicorn + Nginx (for production)
- **Frontend:** HTML, CSS, Bootstrap (Flask templates)

System Components

Frontend (User Side)

- HTML/CSS/Bootstrap pages rendered via Flask templates.
- Features:
    - User registration form (Name, Email, Contact Number, Address, Photo ID).
    - Login/Logout.
    - Apply for a new pass.
    - Renew existing pass.
    - View digital bus pass.

Backend (Application Layer)

- Flask application handles:
    - **Routing** (URLs for register, login, apply, renew, admin dashboard).
    - **Business logic** for pass issuance and renewals.
    - **User authentication** and session management.
    - Communication with the **MySQL database** using connectors (e.g., flask-mysqldb or mysql-connector-python).

Database Layer

- **AWS RDS (MySQL)** manages the database.
- Stores all critical data:
    - User profiles.
    - Application forms.
    - Payment/transaction info.
    - Pass renewal history.
- Benefits: Scalability, automated backup, and high availability.

Cloud Infrastructure (AWS)

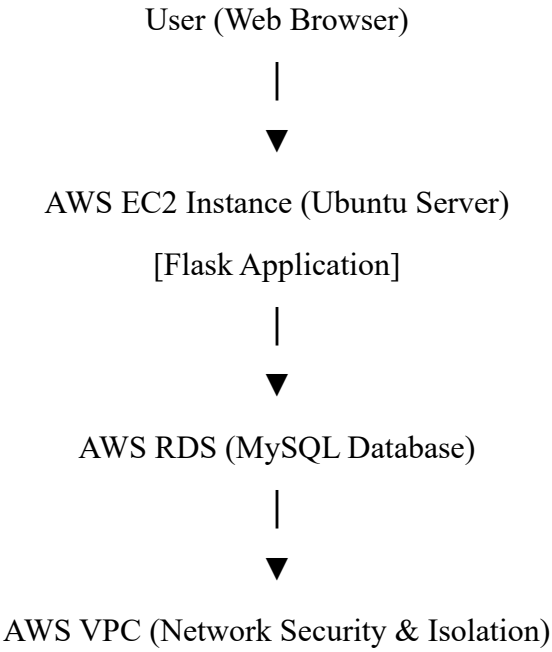
- **AWS EC2 Instance (Ubuntu):**
    - Hosts the Flask application.
    - Configured via Git Bash using SSH.
    - Runs Flask/Gunicorn + Nginx.
    - Accessible via public IP address.

- **AWS RDS (MySQL):**
    - Database service inside the VPC.
    - Connected to Flask app using endpoint, username, and password.
- **VPC (Virtual Private Cloud):**
    - Ensures secure communication between EC2 and RDS.
    - Public subnet for EC2 (accessible to users).
    - Private subnet for RDS (restricted access).

Workflow (How the System Runs)

1. **Accessing the Application**
    - User opens browser and enters **EC2 public IP** (e.g., http://13.201.xxx.xxx).
2. **User Registration/Login**
    - User registers with personal details and uploads a photo ID.
    - Flask app processes the request and stores details in **AWS RDS MySQL database**.
3. **Applying for Bus Pass**
    - User fills in application form (new/renewal).
    - Flask processes the request and updates records in the RDS database.
4. **Digital Pass Issuance**
    - On successful application, a **digital bus pass** is generated and displayed.
5. **Admin Dashboard**
    - Admin logs in to view all applications.
    - Approve/reject user requests.
    - Generate reports on pass issuance and renewals.
6. **Reports & Statistics**
    - Flask fetches data from RDS MySQL.
    - Admin views reports such as total passes issued, renewals, and active users.

Hierarchy / Architecture: -

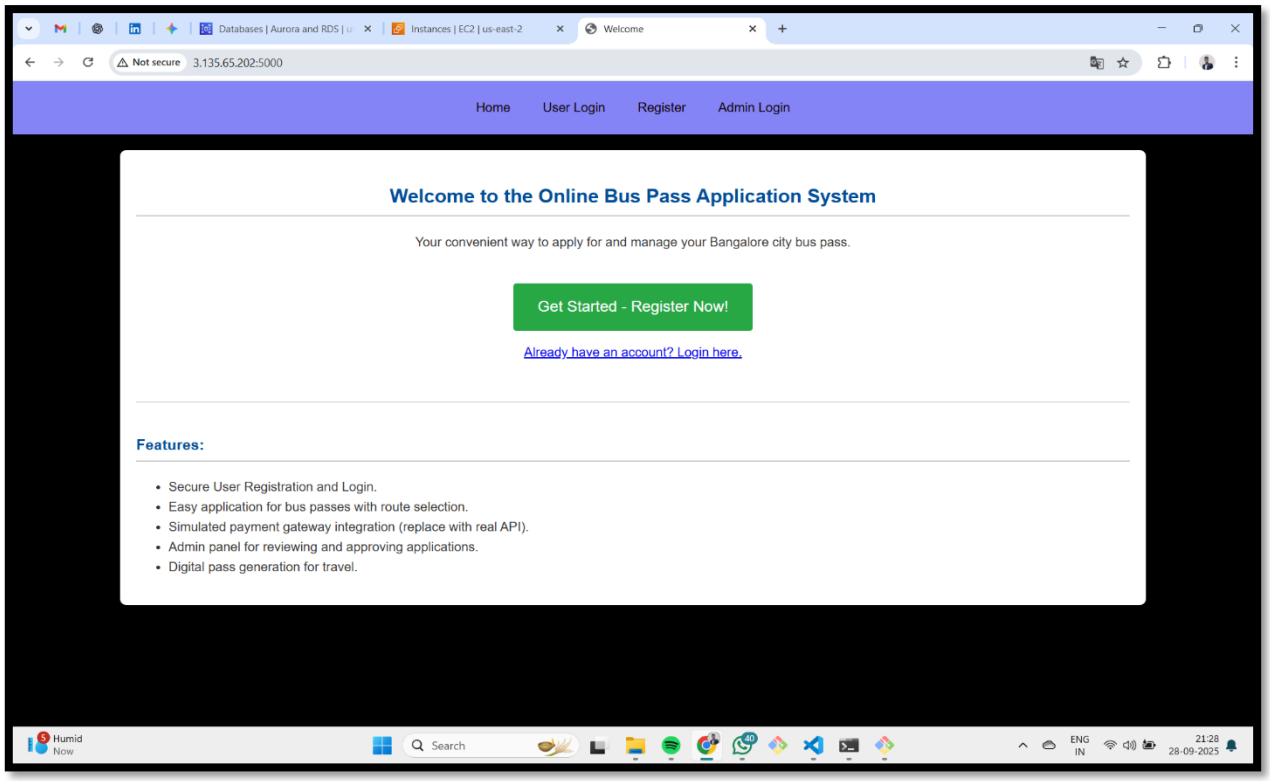


Deployment Steps (Summary)

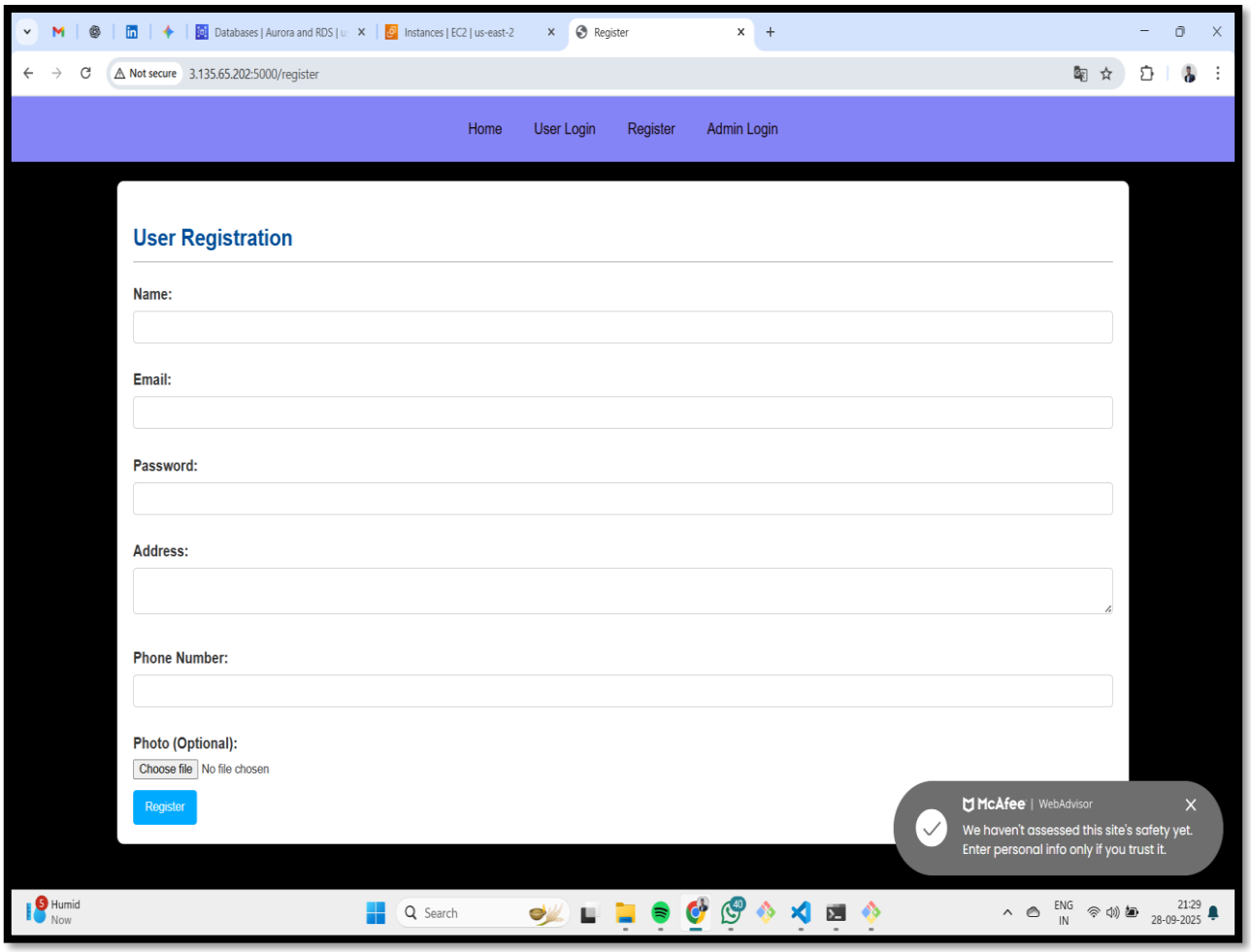
1. **Develop Flask application** locally with MySQL support.
2. **Launch EC2 instance** (Ubuntu) on AWS.
3. **Install dependencies** (Python3, pip, Flask, MySQL connector).
4. **Deploy Flask code to EC2** using Git Bash (SCP/SSH).
5. **Set up RDS (MySQL)** with database, user, and password inside a VPC.
6. **Configure Flask app** to connect to RDS using endpoint.
7. **Run migrations & initialize database**.
8. **Start Flask app** (python3 app.py) or use **Gunicorn + Nginx** for production.
9. **Access via public IP** of EC2.

Outputs (System Features)

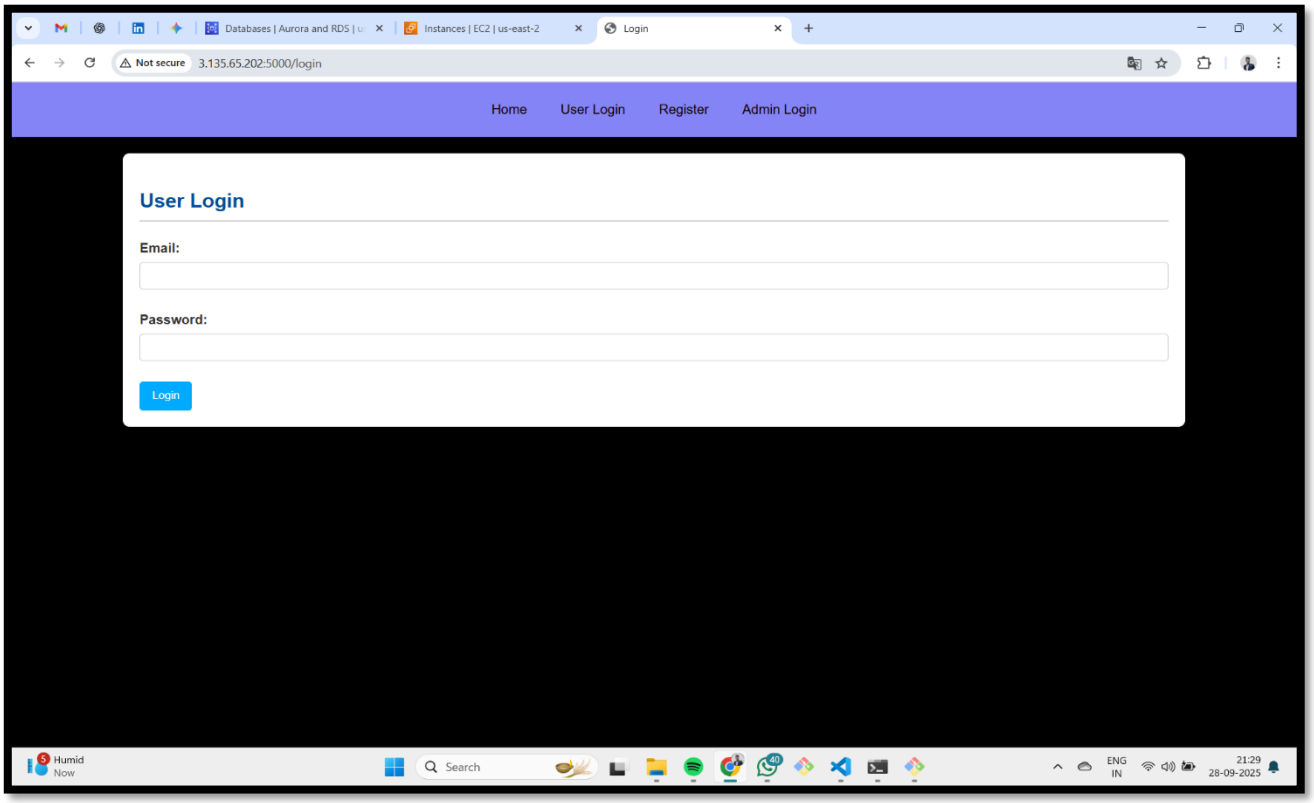
- Digital Bus Pass generated for users.
- Admin dashboard to manage applications.
- Reports on pass issuance and renewals.
- Secure cloud deployment (EC2 + RDS + VPC).
- Data stored centrally in RDS MySQL database.



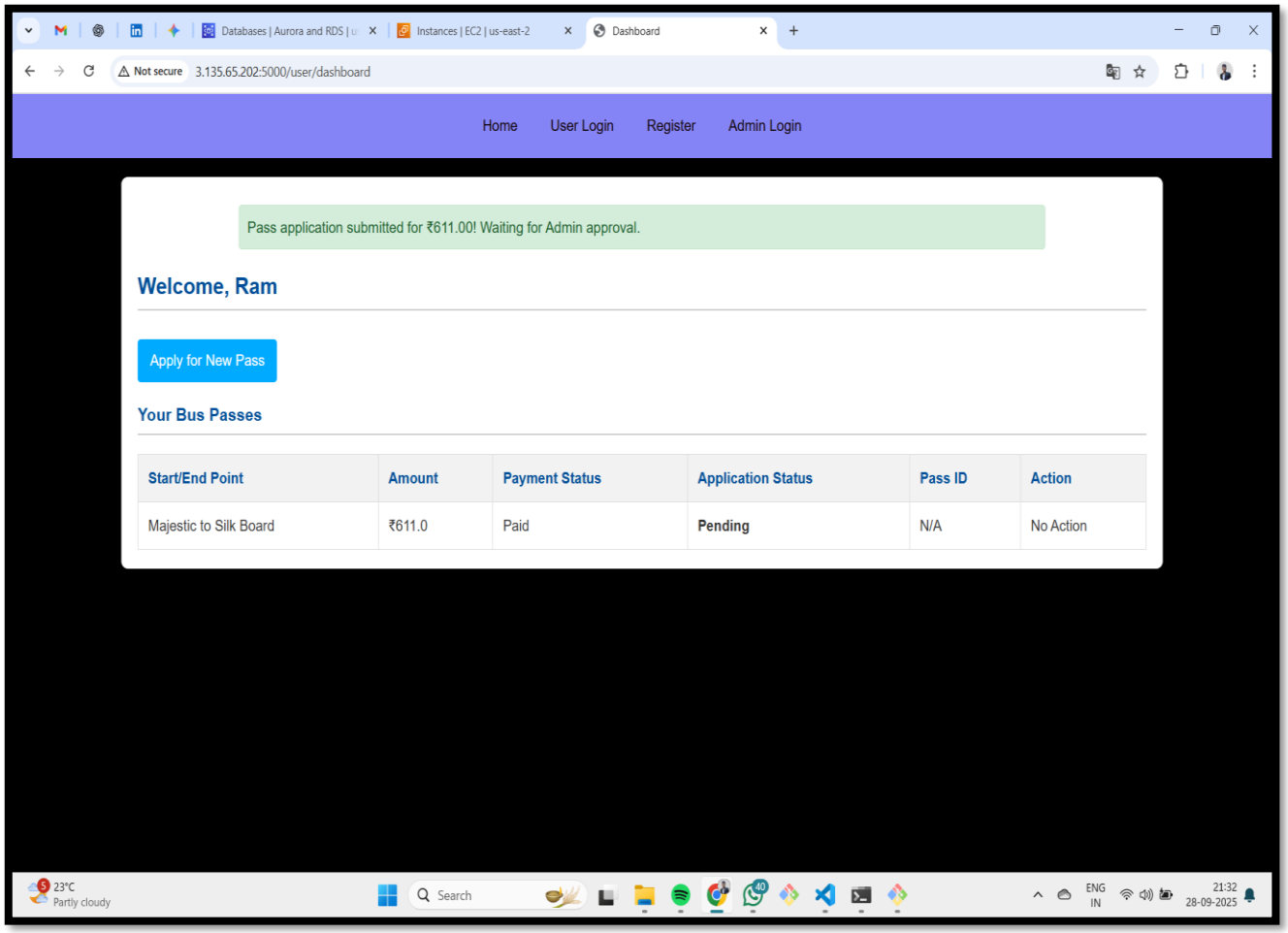
Home page



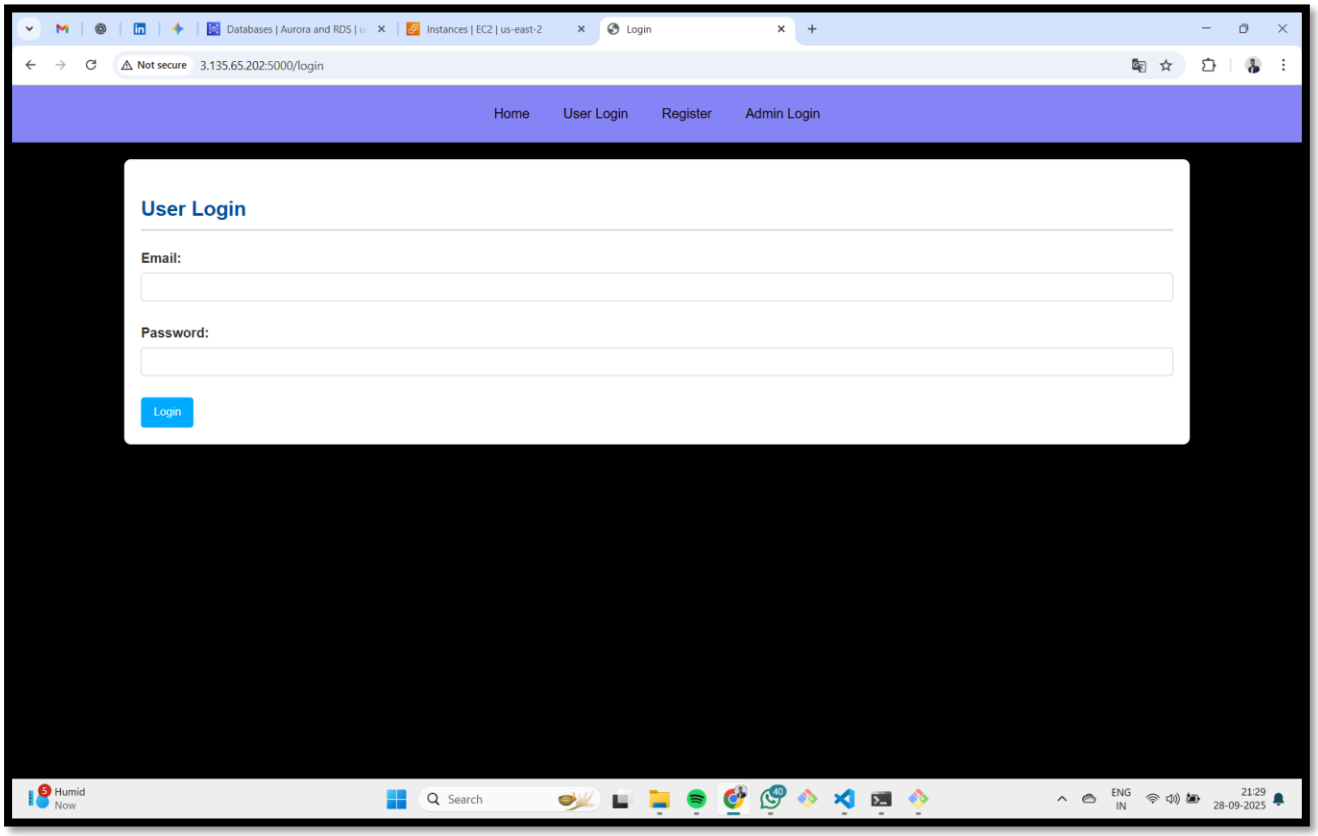
Registration Page



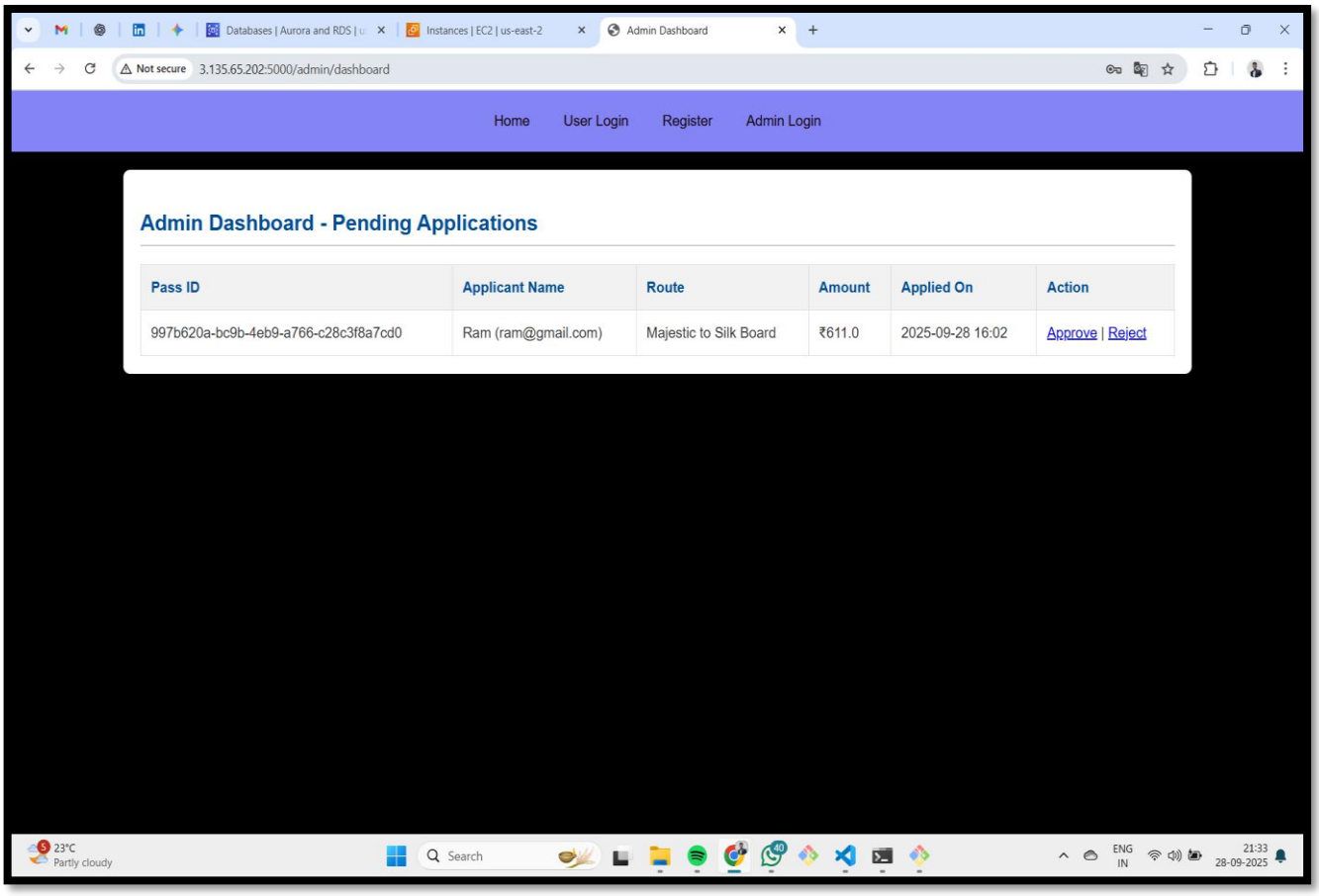
Login Page



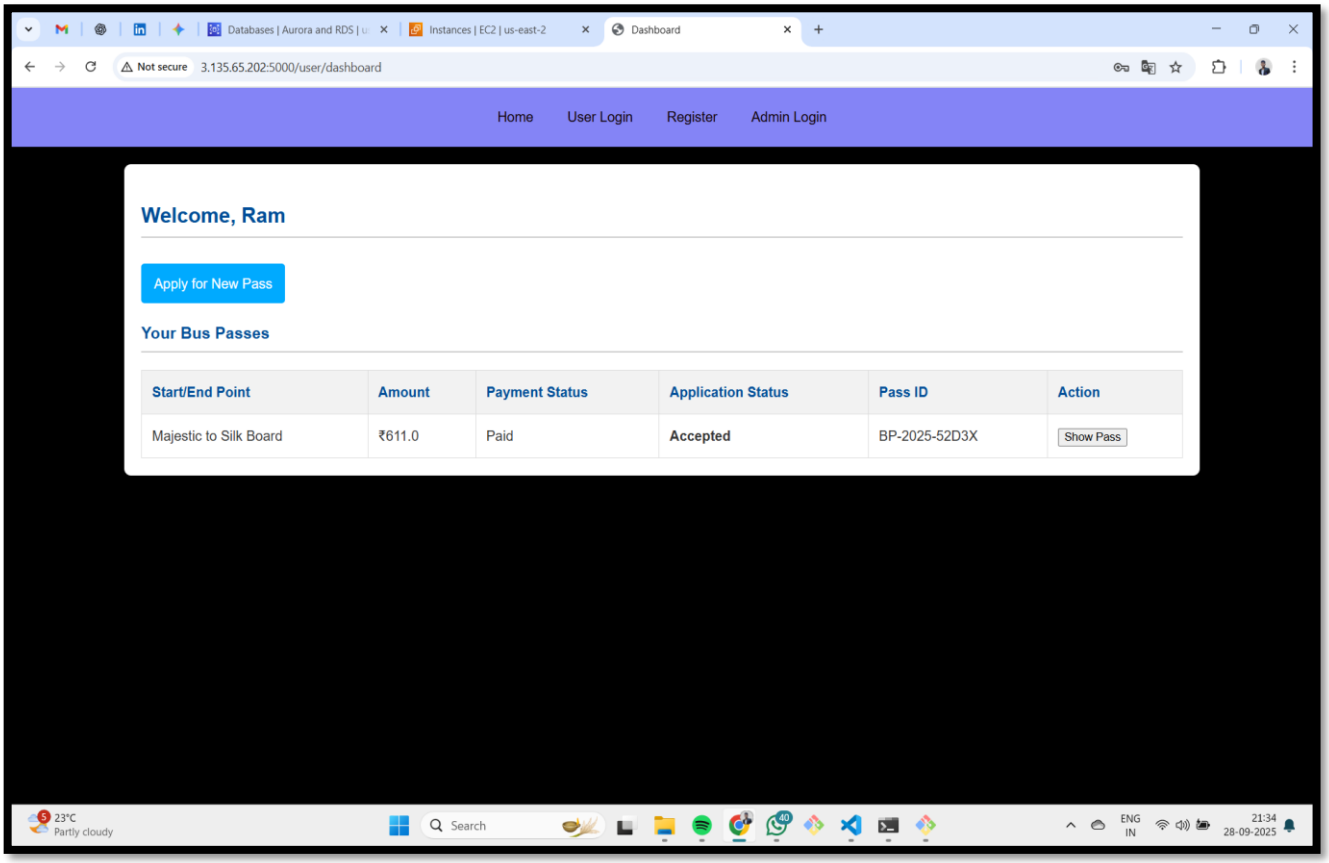
Apply for Pass



Admin Login Page



Admin Dashboard to Approve

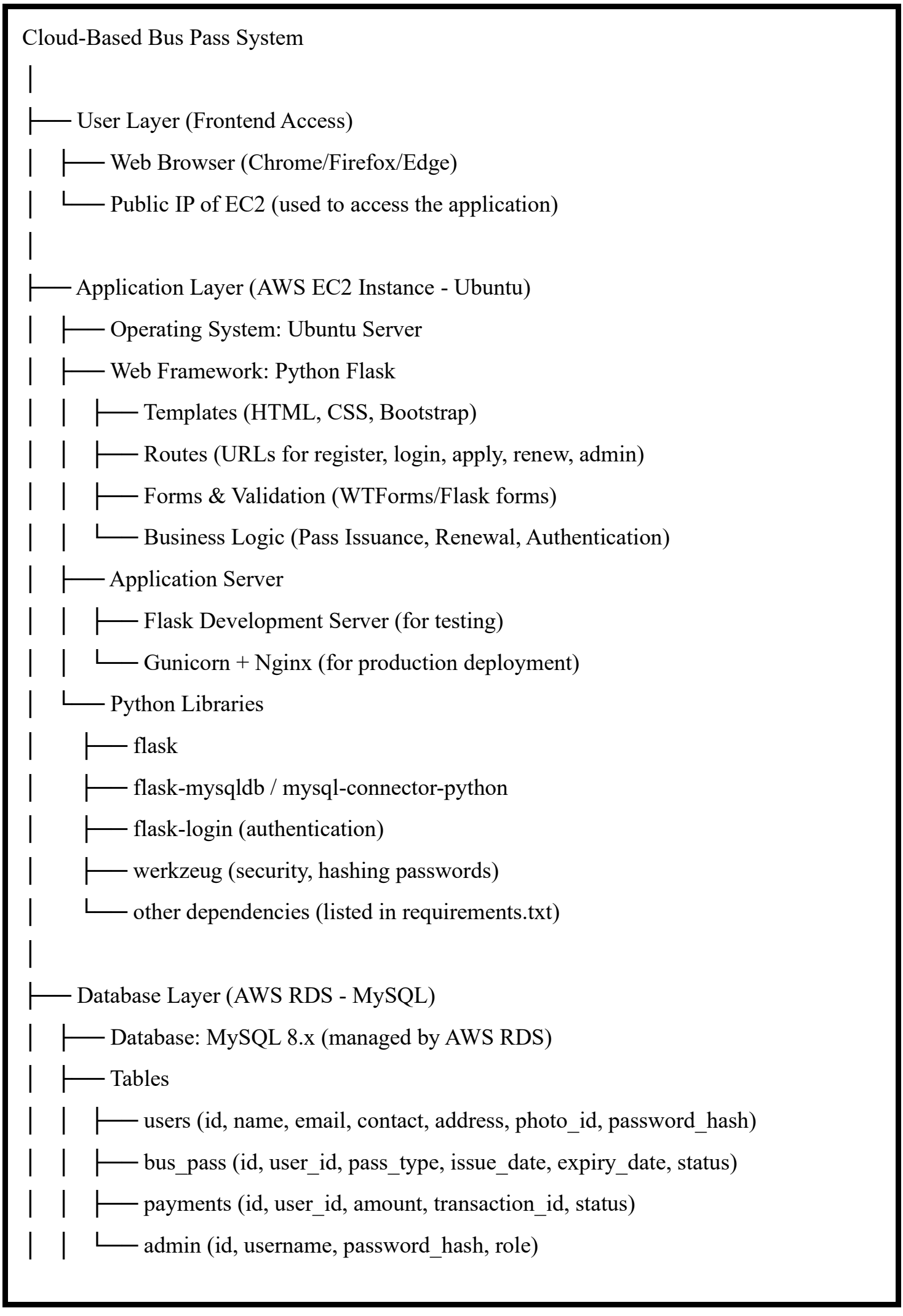


Bus Pass Generate Page

Conclusion

The *Cloud-Based Bus Pass System* demonstrates how **Flask, MySQL, and AWS services** can be integrated to build a scalable cloud application. By separating the **application (EC2)** and **database (RDS)** layers, the system achieves high availability, performance, and security.

The use of **VPC ensures safe communication**, while the **public IP of EC2 provides easy access** to end users. This project is a complete example of how modern cloud platforms can be used for **real-world applications** in transportation and digital services.


Cloud-Based Bus Pass System
User Layer (Frontend Access)
Web Browser (Chrome/Firefox/Edge)
Public IP of EC2 (used to access the application)
Application Layer (AWS EC2 Instance - Ubuntu)
Operating System: Ubuntu Server
Web Framework: Python Flask
Templates (HTML, CSS, Bootstrap)
Routes (URLs for register, login, apply, renew, admin)
Forms & Validation (WTForms/Flask forms)
Business Logic (Pass Issuance, Renewal, Authentication)
Application Server
Flask Development Server (for testing)
Gunicorn + Nginx (for production deployment)
Python Libraries
flask
flask-mysqldb / mysql-connector-python
flask-login (authentication)
werkzeug (security, hashing passwords)
other dependencies (listed in requirements.txt)
Database Layer (AWS RDS - MySQL)
Database: MySQL 8.x (managed by AWS RDS)
Tables
users (id, name, email, contact, address, photo_id, password_hash)
bus_pass (id, user_id, pass_type, issue_date, expiry_date, status)
payments (id, user_id, amount, transaction_id, status)
admin (id, username, password_hash, role)

```
│   ├── Features
│   │   ├── Automated backups
│   │   ├── Multi-AZ (optional, for high availability)
│   │   └── Secure access through VPC security groups
│
├── Networking Layer (AWS VPC)
│   ├── VPC (Virtual Private Cloud)
│   ├── Subnets
│   │   ├── Public Subnet → for EC2 instance (accessible via public IP/SSH)
│   │   └── Private Subnet → for RDS (only accessible to EC2, not public)
│   ├── Security Groups
│   │   ├── EC2 Security Group → allows HTTP (80), HTTPS (443), SSH (22)
│   │   └── RDS Security Group → allows MySQL (3306) only from EC2
│   ├── Internet Gateway → connects VPC to the internet
│   └── Route Tables → define traffic flow inside the VPC
│
└── Management & Deployment Layer
    ├── Git Bash (for SCP, SSH, deployment commands)
    ├── Code Deployment
    │   ├── Clone/upload Flask project to EC2
    │   ├── Install dependencies via pip
    │   └── Configure Flask app to connect to RDS endpoint
    ├── System Monitoring
    │   ├── AWS CloudWatch (optional logs & monitoring)
    │   └── EC2 Instance monitoring (CPU, memory, network)
    └── Admin Operations
        ├── Admin dashboard for user management
        ├── Report generation (pass issuance/renewals)
        └── Database maintenance (via RDS console or MySQL Workbench)
```