Fraudulent Transaction Detector

Part 4

Arya Goyal (ag9961)

Gitesh Chinawalkar (gc3410)

Table of Contents

Business Use Cases for Workflow-Based Application……………………………..……...3

Business Use Cases Summary…………………………………………………….……..3

Workflow Integration…………………………………………………………….…..5

Design of data-driven workflow-based database application……………………………..…8

N-Tier Architecture Adaptation………………………………………………………10

Implementation of Data-Driven Program Module and Workflow-Based Application……….8

Application Overview…………………………………………….……….………………10

Demonstration and Validation……………………………………………………………12

Walkthrough of the Application………………………………………………………13

Validation……………………………………………….…………………………….…….14

Github Link………………………………………………………………………………….15

Business Use Cases for Workflow-Based Application

The rapid expansion of the digital financial ecosystem has led to a surge in transaction volumes and complexities, making robust fraud detection mechanisms a critical requirement. This project addresses the challenges by focusing on the following objectives:

1. Real-Time Detection and Prevention:
   - Implement advanced machine learning algorithms capable of identifying fraudulent transactions as they occur.
   - Reduce financial losses by immediately flagging suspicious activities and blocking them.
2. Scalable Architectures:
   - Design a system that can handle millions of transactions per second without compromising performance or accuracy.
   - Ensure scalability through distributed computing frameworks.
3. Integration of Compliance Workflows:
   - Automate the generation of regulatory reports to meet compliance requirements (e.g., Anti-Money Laundering regulations).
   - Create a secure audit trail for investigating flagged transactions.

Business Use Cases Summary

Fraud Detection

Use Cases:

- Analyze high-frequency transactions to detect fraud patterns in real time.
- Automatically notify users and financial institutions when a transaction is identified as high-risk.

Processes:

- Perform Feature Engineering to identify fraud indicators, such as:
  - Zero balances post-transfer.
  - Transactions with unusually high amounts or frequency.
- Integrate Machine Learning Models to enable continuous monitoring and detection.

Risk Profiling

Use Cases:

- Assign dynamic fraud risk scores to user accounts based on transactional behaviors and historical patterns.

Processes:

- Apply Clustering Algorithms to segment users into risk profiles dynamically (e.g., low, medium, or high risk).

- Update risk scores in real time using feedback loops and machine learning updates.

Regulatory Reporting

Use Cases:

- Automate the creation of compliance reports containing detailed logs of all flagged transactions.
- Provide periodic updates to regulators, ensuring transparency and accountability.

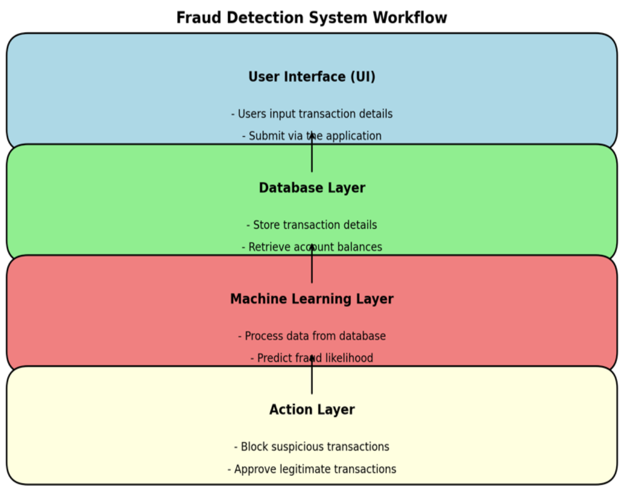
Processes:

- Maintain Auditable Logs for every flagged transaction, including time stamps, associated risk scores, and detection rationale.
- Implement secure storage and backup systems to preserve flagged transaction data, ensuring data integrity for audits.

Workflow Integration

Seamless Integration: The application workflows are designed to seamlessly integrate these business use cases, ensuring a smooth and intuitive user experience.

Data-Driven Decision Making: Each use case is backed by data-driven logic, ensuring that the application not only serves operational needs but also enhances decision-making through insights derived from data analytics and machine learning models.


Fraud Detection System Workflow
User Interface (UI)
- Users input transaction details
- Submit via the application
Database Layer
- Store transaction details
- Retrieve account balances
Machine Learning Layer
- Process data from database
- Predict fraud likelihood
Action Layer
- Block suspicious transactions
- Approve legitimate transactions

Design of Data-Driven Workflow-Based Fraud Detection Application

Our Fraud Detection System adopts a sophisticated n-tier architecture, seamlessly integrating real-time data processing, database management, and machine learning capabilities. This architecture ensures the system's scalability, security, and resilience, catering to the stringent requirements of fraud detection in financial operations.

N-Tier Architecture Adaptation

1. Presentation Layer

- Purpose: Responsible for displaying transaction details to users and providing an intuitive interface for interaction.
- Features:
  - Allows users to register and login into the application.
  - User can enter their details like address, phone number, SSN.
  - Allows users to input transaction details (e.g., sender, recipient, amount).
  - Displays fraud status notifications, account balances, and flagged transactions.
  - Allows the admin user to delete any entries or users if required.
- Technology:
  - Built with HTML, CSS, and Flask for a responsive and dynamic user experience.
  - Incorporates Flask for modern UI design and to integrate seamlessly with backend and frontend.
- Key Benefits:
  - Accessible across multiple devices.
  - Ensures a seamless and user-friendly experience.

2. Business Logic Layer

- Purpose: Acts as the intermediary between the presentation layer and backend layers, ensuring seamless execution of business rules.
- Features:
  - Validates transaction details (e.g., account verification, sufficient balance checks).
  - Orchestrates workflows, directing transaction data to the database services layer and machine learning layer.
  - Implements fraud thresholds (e.g., blocking high-value suspicious transactions).
- Technology:
  - Developed using Flask.
  - Integrates RESTful APIs for communication between layers.
- Key Benefits:
  - Provides robust application logic.
  - Handles exceptions efficiently, ensuring minimal downtime.

3. Machine Learning Layer

- Purpose: Hosts, executes, and manages machine learning models to detect fraudulent transactions.
- Features:
  - Utilizes Random Forests and Gradient Boosted Trees for fraud detection.

    - Processes structured and unstructured data to identify anomalies and predict fraud likelihood.
    - Incorporates real-time model retraining pipelines to adapt to evolving fraud patterns.
    - Gives an integer output as 1 if the transaction is fraud or 0 if the transaction is not fraud.
- Technology:
    - Powered by TensorFlow and PyTorch for advanced machine learning capabilities.
    - Deploys and saved models using Joblib library for cross-platform compatibility.
- Key Benefits:
    - Delivers high accuracy in detecting fraudulent patterns.
    - Scalable to handle large volumes of data.

4. Database Services Layer

- Purpose: The foundational layer for data storage and management.
- Features:
    - Optimized schema design for handling transactional data, user profiles, and fraud logs.
    - Provides real-time read/write operations for fraud detection workflows.
- Technology:
    - PostgreSQL for relational data storage.
    - Local file storage or on-premises databases for unstructured data, supporting machine learning pipelines.
- Key Benefits:
    - Ensures low-latency data retrieval.
    - Guarantees data integrity and compliance with financial regulations.

Key Benefits of the Architecture

1. Scalability: Handles millions of transactions daily, adapting to peak loads seamlessly.
2. Resilience: Distributed architecture ensures uninterrupted fraud detection even during infrastructure failures.
3. Accuracy: High-performance ML models achieve state-of-the-art precision in identifying fraudulent activities.
4. Security: End-to-end encryption and compliance with financial data regulations ensure robust data protection.

Implementation of Data-Driven Program Module and Workflow-Based Application

Application Structure

Our Flask-based application facilitates fraud detection for transactions, integrating machine learning predictions with database management and a user-friendly interface. The application streamlines workflows for users, including authentication, transaction initiation, and fraud alerts.

Data-Driven Modules

1. MySQL Database Integration:

    - A robust PostgreSQL database manages user accounts, transaction records, and fraud flags.
    - SQLAlchemy ORM simplifies database interactions, mapping tables to Python classes.
2. Machine Learning Integration:
    - The XG boost model predicts the likelihood of fraud based on transaction parameters. The model gives the best accuracy compared to all the other models it was tested on.
    - Features include:
        - Timestamp-based activity analysis.
        - Balance updates and verification.
        - Prediction outputs determine transaction legitimacy.
3. Fraud Detection Logic:
    - Inputs such as account balance and transaction type are preprocessed and fed into the model.
    - Predictions are handled within the transaction endpoint, with redirection to success or fraud alerts based on outcomes.

Application Features

1. Authentication and Registration

- HTML Pages:
    - Login Page (index.html): Supports secure login via Flask-Login.
    - Register Page (register.html): Allows user creation with encrypted passwords using Flask-Bcrypt.
    - Admin page(manage_users.html): Allows the admin to delete any entries or users from the database if required. This page is only visible to the admin.
- Backend:
    - app.py handles routes for login (/login), registration (/register), transaction (/transaction) integrating user validation and hashing mechanisms.

2. Transaction Processing

- HTML Pages:
    - Transaction Form (transaction.html): Allows user to make a new transaction by putting in the account number of source and destination and then the transaction amount.
    - Transaction Success Page (transaction_success.html): Confirms valid transactions.
    - Fraud Alert Page (transaction_fail.html): Displays alerts for suspicious activities.
- Backend:
    - /transaction endpoint performs the following:
        - Validates accounts and retrieves balances.
        - Prepares input for fraud detection.
        - Calls the ML model and updates the database based on predictions.

3. Dashboard and User Interface

- HTML Pages:
    - Home Page (home_page.html): Central hub for user navigation.
    - New Client Page (new_client.html): Collects and adds user details.
- Backend:
    - The dashboard and new_client routes handle respective functionalities using FlaskForms and SQLAlchemy models.

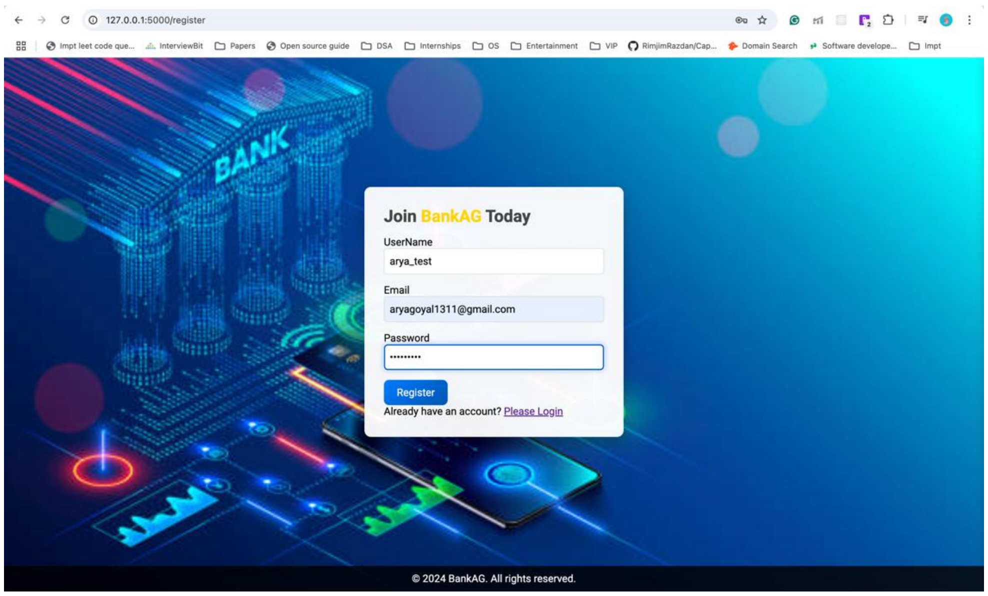
    - /manage-users lets the admin delete users from the database and shows the same on UI as well.

Workflow Optimization

1. Database Management:
    - Indexing transaction tables for quicker fraud checks.
    - Optimizing SQLAlchemy (ORM framework) queries to minimize response times.
2. Machine Learning Predictions:
    - Preloading and caching models for efficient inference.
    - Modular code ensures scalability and retraining.
3. Frontend Design:
    - Responsive design ensures usability across devices.
    - Clear messaging guides users through transaction workflows.

Demonstration and Validation

Landing Page - User registers first



127.0.0.1:5000/register

Join BankAG Today

UserName: arya_test

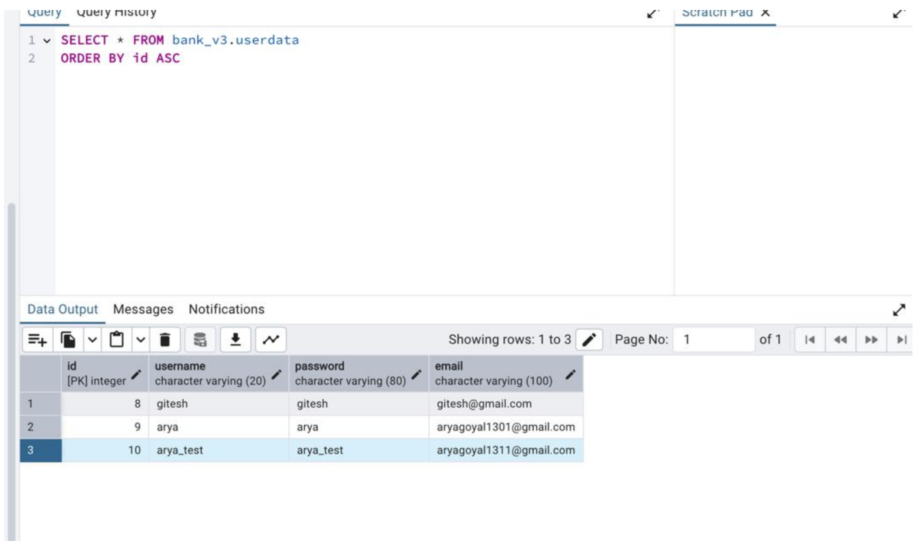
Email: aryagoyal1311@gmail.com

Password: •••••••••

Already have an account? Please Login

© 2024 BankAG. All rights reserved.

The data gets stored in the database:

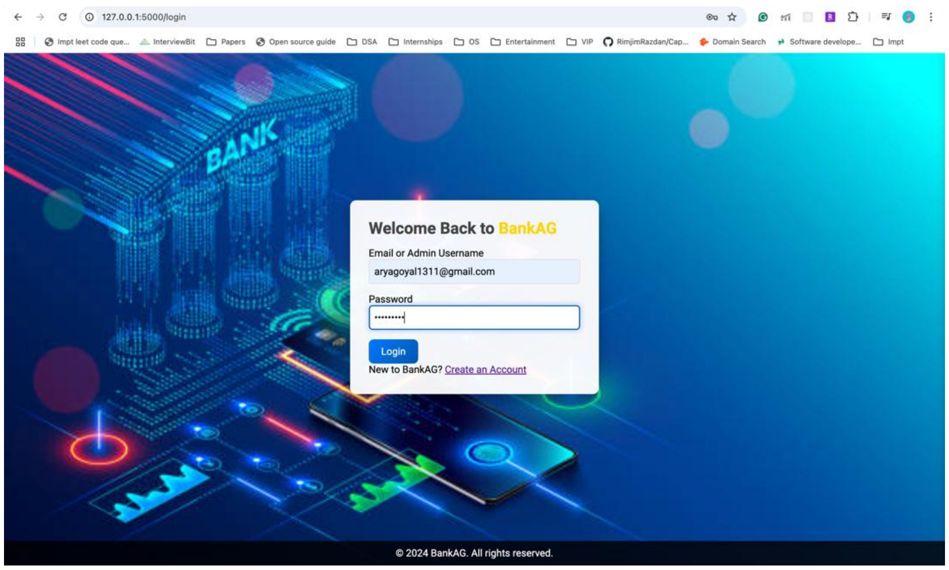


```
SELECT * FROM bank_v3.userdata
ORDER BY id ASC
```

Showing rows: 1 to 3 Page No: 1 of 1

	id [PK] integer	username character varying (20)	password character varying (80)	email character varying (100)
1	8	gitesh	gitesh	gitesh@gmail.com
2	9	arya	arya	aryagoyal1301@gmail.com
3	10	arya_test	arya_test	aryagoyal1311@gmail.com

Login - the user can login with his email id and password, these credentials are checked in the database to see if they exist or not.



Welcome Back to BankAG

Email or Admin Username: aryagoyal1311@gmail.com

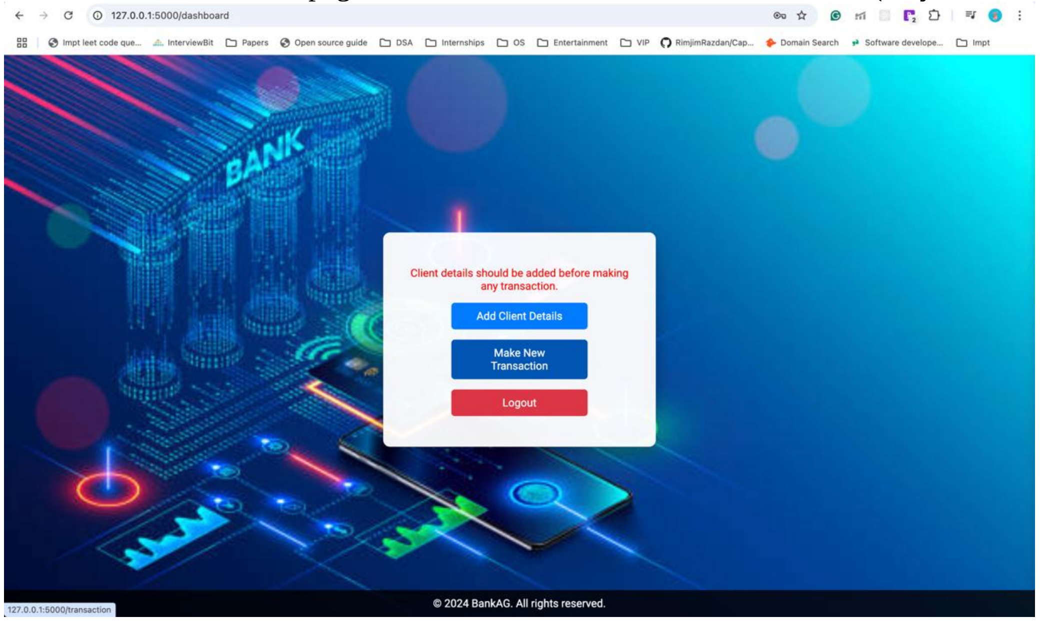
Password: ••••••••

Login

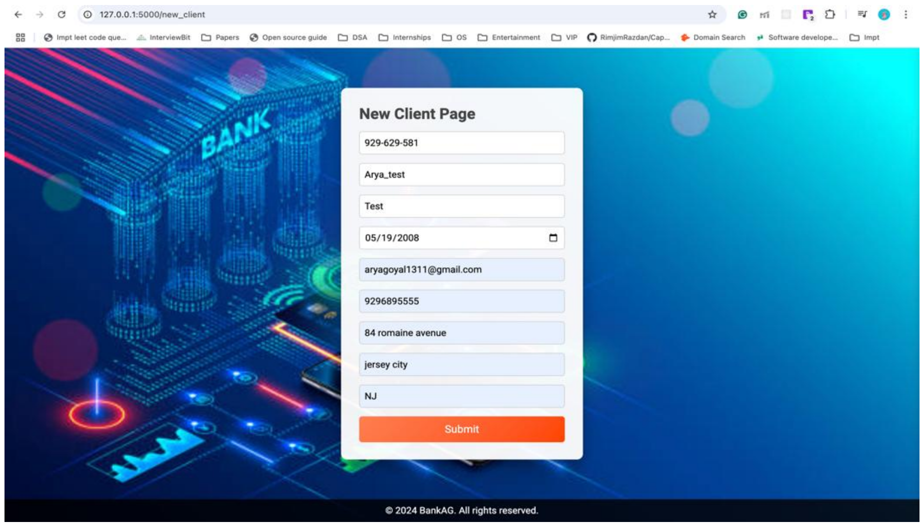
New to BankAG? Create an Account

© 2024 BankAG. All rights reserved.

The user then lands onto the home page, where he first has to add the client details (only for the first time)


Client details should be added before making any transaction.

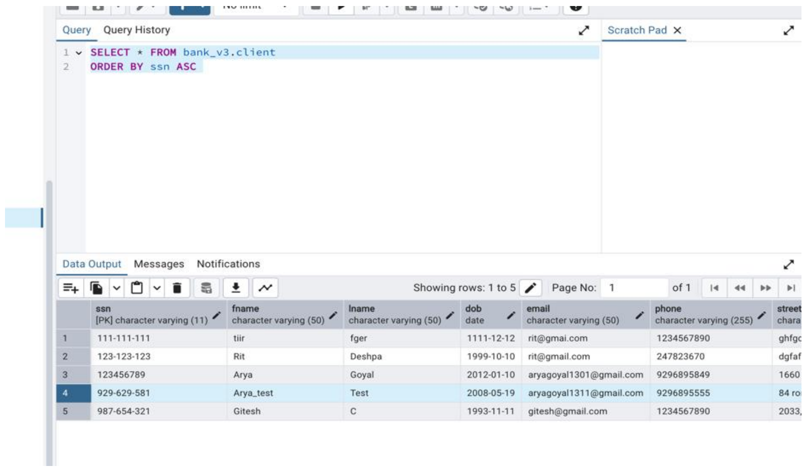
Client Details – the user then adds the client details like SSN, address, DOB, email and phone number


New Client Page

929-629-581
Arya_test
Test
05/19/2008
aryagoyal1311@gmail.com
9296895555
84 romaine avenue
jersey city
NJ

127.0.0.1:5000/new_client
Submit
© 2024 BankAG. All rights reserved.

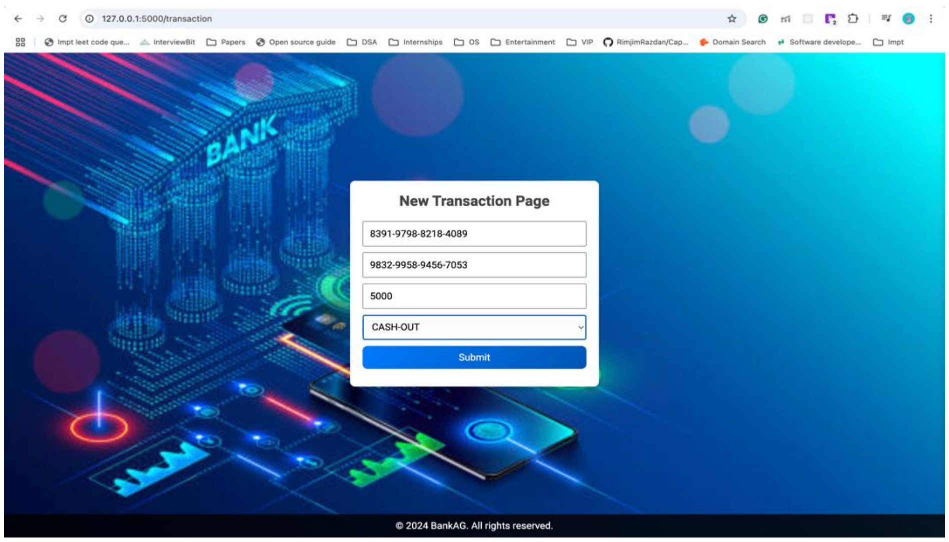
These details are then stored into the database:



```
SELECT * FROM bank_v3.client
ORDER BY ssn ASC
```

	ssn [PK] character varying (11)	fname character varying (50)	lname character varying (50)	dob date	email character varying (50)	phone character varying (255)	street chara
1	111-111-111	tiir	fger	1111-12-12	rit@gmai.com	1234567890	ghfgc
2	123-123-123	Rit	Deshpa	1999-10-10	rit@gmail.com	247823670	dgfaf
3	123456789	Arya	Goyal	2012-01-10	aryagoyal1301@gmail.com	9296895849	1660
4	929-629-581	Arya_test	Test	2008-05-19	aryagoyal1311@gmail.com	9296895555	84 ro
5	987-654-321	Gitesh	C	1993-11-11	gitesh@gmail.com	1234567890	2033,

The user can then choose to make a transaction and he goes to the transaction page:

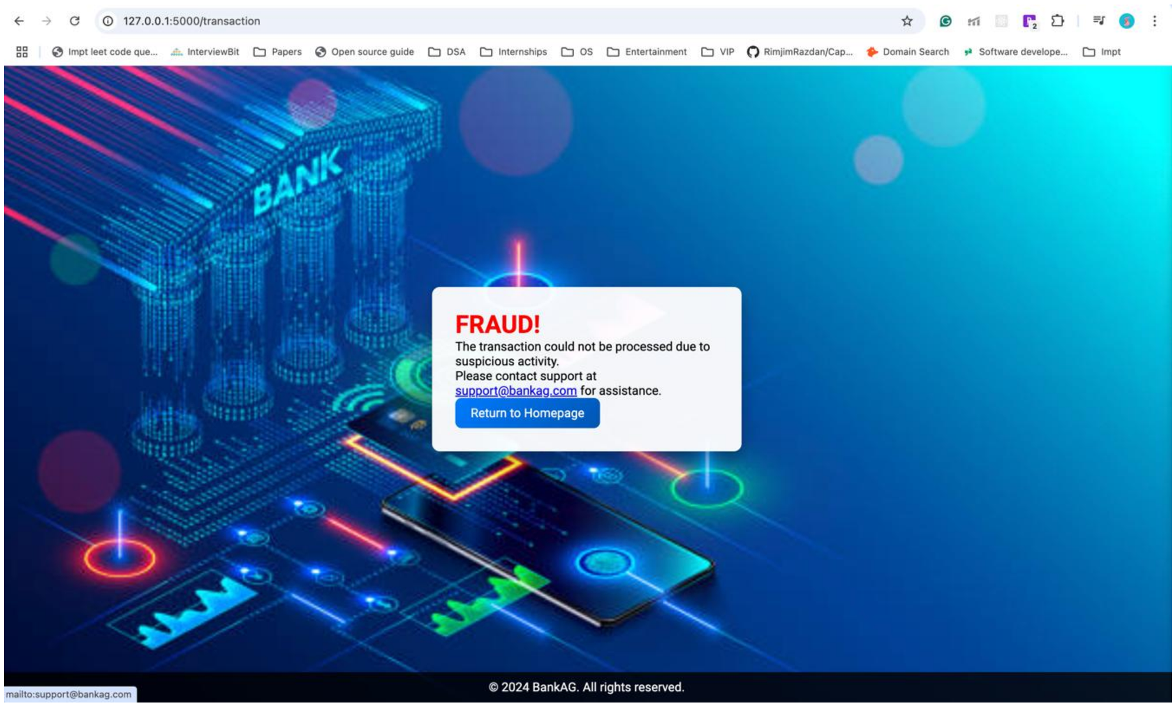


New Transaction Page

8391-9798-8218-4089
9832-9958-9456-7053
5000
CASH-OUT

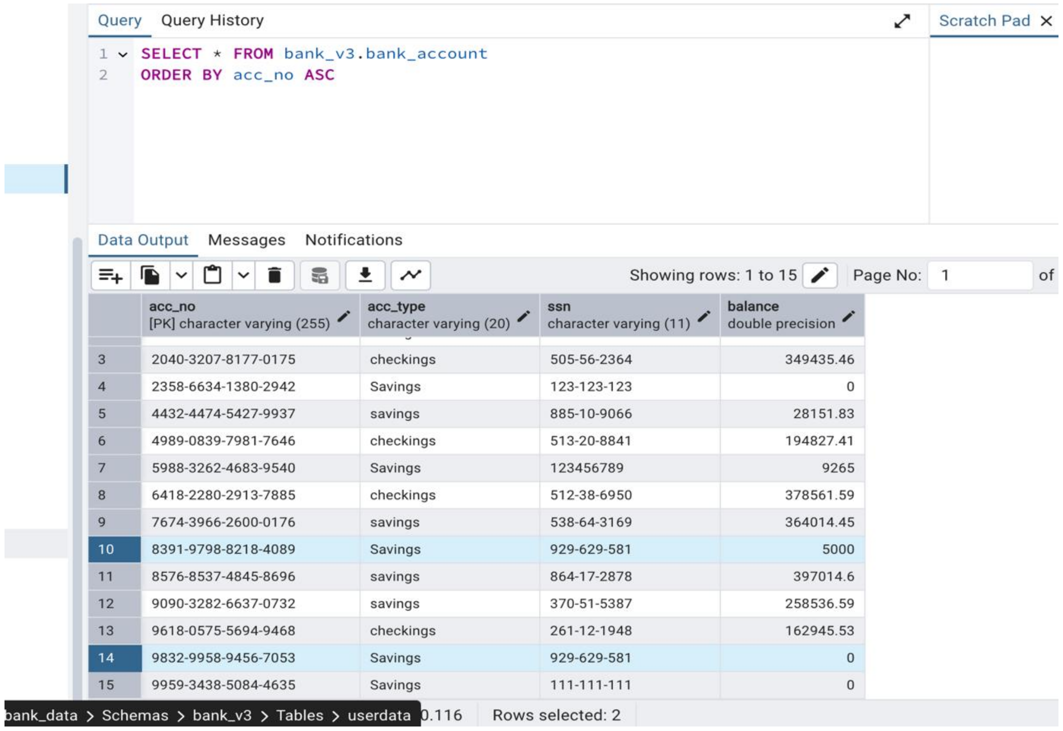
Once the submit button is clicked, the details of the balance will be fetched from the database using both the account number given and fed into the ML model to predict if the transaction is fraud or not.

If the transaction is fraud then a failure page is shown to the user and the transaction is not stored in the database.


127.0.0.1:5000/transaction

FRAUD!

The transaction could not be processed due to suspicious activity.
Please contact support at support@bankag.com for assistance.



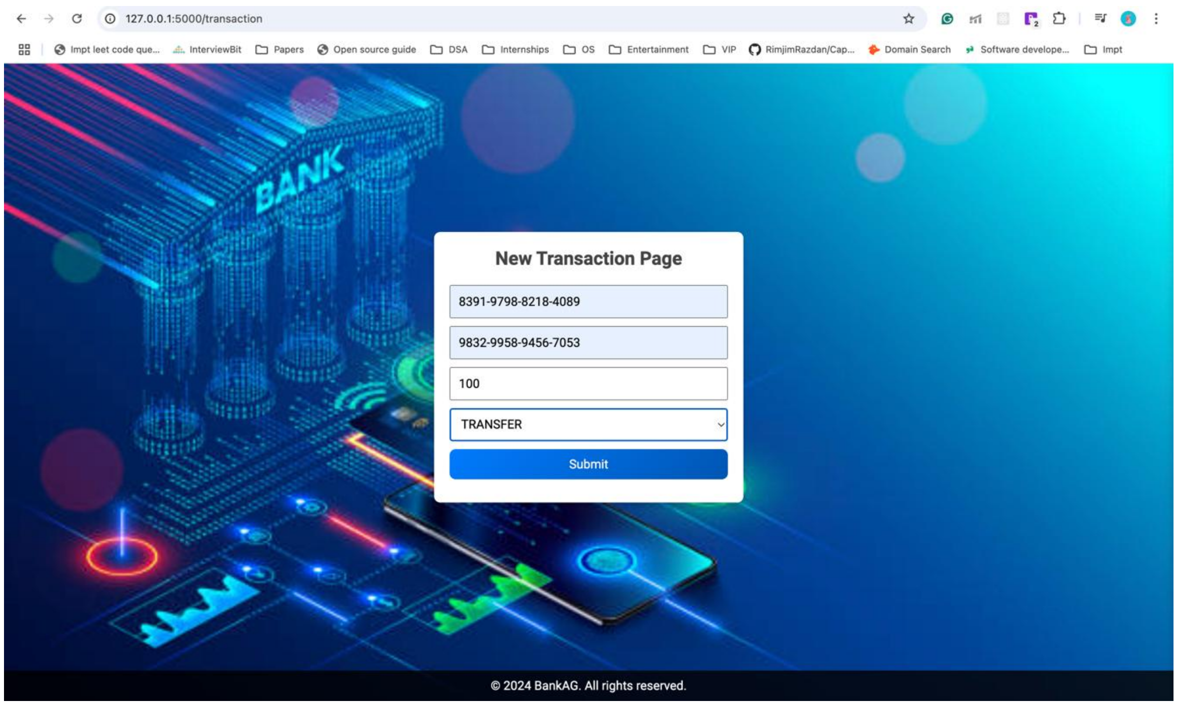
```sql
SELECT * FROM bank_v3.bank_account
ORDER BY acc_no ASC
```

Showing rows: 1 to 15 Page No: 1 of

	acc_no [PK] character varying (255)	acc_type character varying (20)	ssn character varying (11)	balance double precision
3	2040-3207-8177-0175	checkings	505-56-2364	349435.46
4	2358-6634-1380-2942	Savings	123-123-123	0
5	4432-4474-5427-9937	savings	885-10-9066	28151.83
6	4989-0839-7981-7646	checkings	513-20-8841	194827.41
7	5988-3262-4683-9540	Savings	123456789	9265
8	6418-2280-2913-7885	checkings	512-38-6950	378561.59
9	7674-3966-2600-0176	savings	538-64-3169	364014.45
10	8391-9798-8218-4089	Savings	929-629-581	5000
11	8576-8537-4845-8696	savings	864-17-2878	397014.6
12	9090-3282-6637-0732	savings	370-51-5387	258536.59
13	9618-0575-5694-9468	checkings	261-12-1948	162945.53
14	9832-9958-9456-7053	Savings	929-629-581	0
15	9959-3438-5084-4635	Savings	111-111-111	0

bank_data > Schemas > bank_v3 > Tables > userdata 0.116 Rows selected: 2

Since the transaction was of a small amount, so it gets successful and we can see a transaction successful page which is then updated in the DB


127.0.0.1:5000/transaction

New Transaction Page

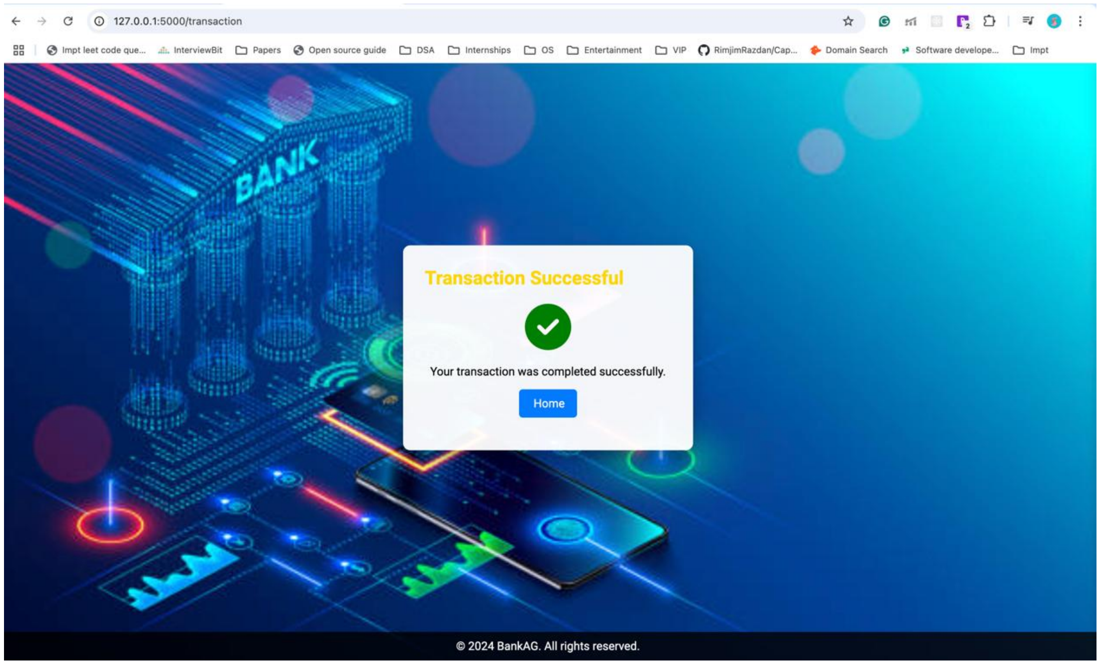
8391-9798-8218-4089

9832-9958-9456-7053

100

TRANSFER

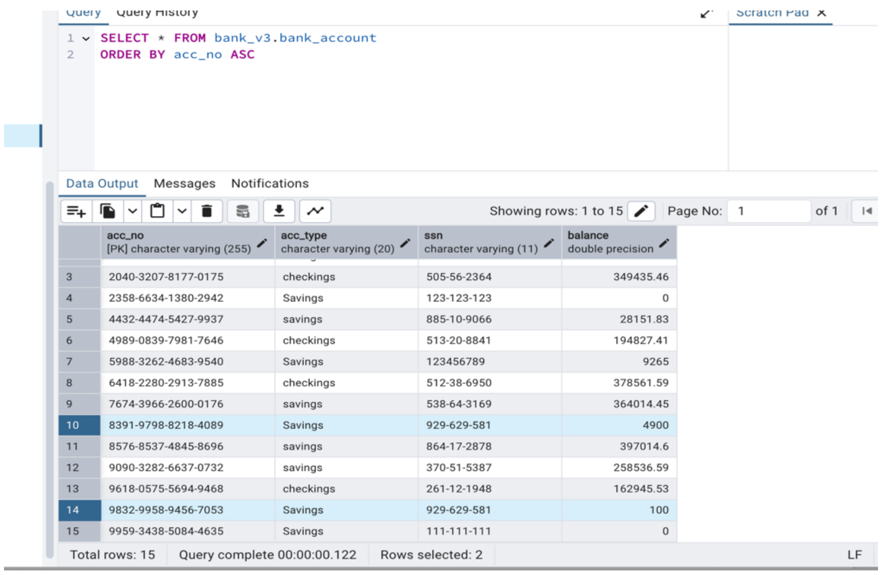
© 2024 BankAG. All rights reserved.


127.0.0.1:5000/transaction

Transaction Successful

Your transaction was completed successfully.

© 2024 BankAG. All rights reserved.



```sql
SELECT * FROM bank_v3.bank_account
ORDER BY acc_no ASC
```

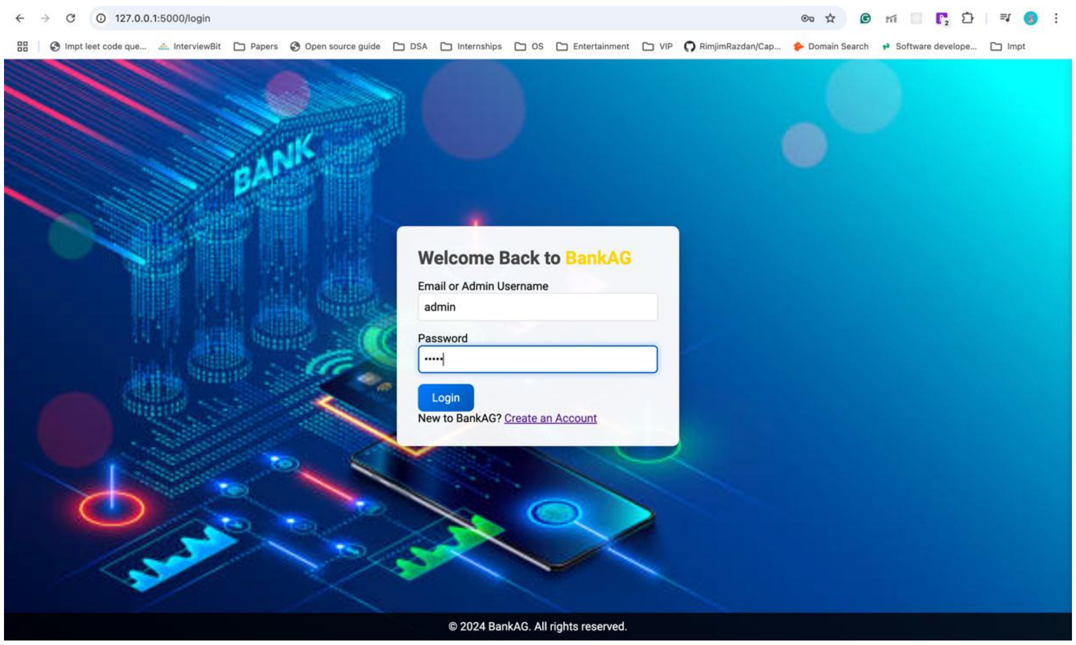
Showing rows: 1 to 15 Page No: 1 of 1

	acc_no [PK] character varying (255)	acc_type character varying (20)	ssn character varying (11)	balance double precision
3	2040-3207-8177-0175	checkings	505-56-2364	349435.46
4	2358-6634-1380-2942	Savings	123-123-123	0
5	4432-4474-5427-9937	savings	885-10-9066	28151.83
6	4989-0839-7981-7646	checkings	513-20-8841	194827.41
7	5988-3262-4683-9540	Savings	123456789	9265
8	6418-2280-2913-7885	checkings	512-38-6950	378561.59
9	7674-3966-2600-0176	savings	538-64-3169	364014.45
10	8391-9798-8218-4089	Savings	929-629-581	4900
11	8576-8537-4845-8696	savings	864-17-2878	397014.6
12	9090-3282-6637-0732	savings	370-51-5387	258536.59
13	9618-0575-5694-9468	checkings	261-12-1948	162945.53
14	9832-9958-9456-7053	Savings	929-629-581	100
15	9959-3438-5084-4635	Savings	111-111-111	0

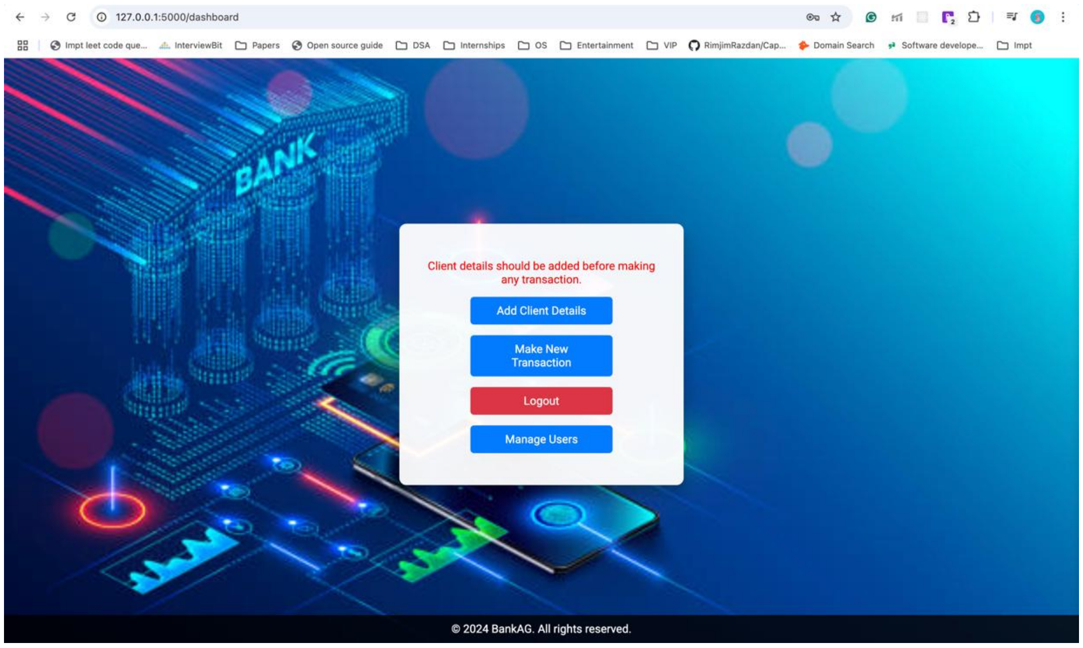
Total rows: 15 Query complete 00:00:00.122 Rows selected: 2

Admin user Details

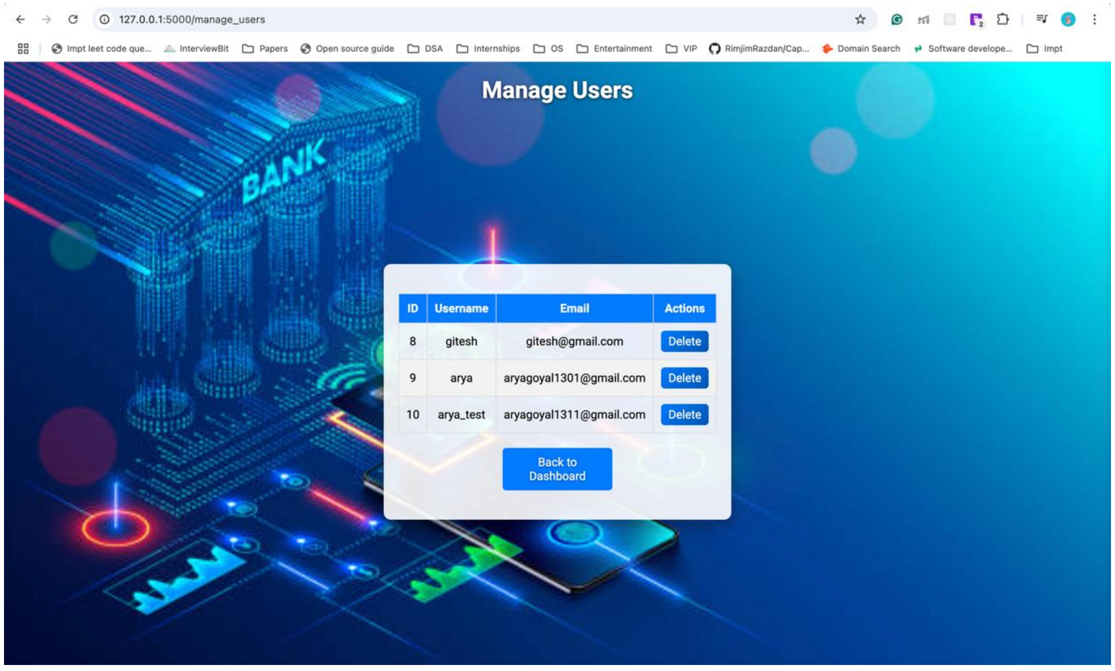
When a user login with admin/admin as username and password



Admin can see an extra button to manage the users on their dashboard.

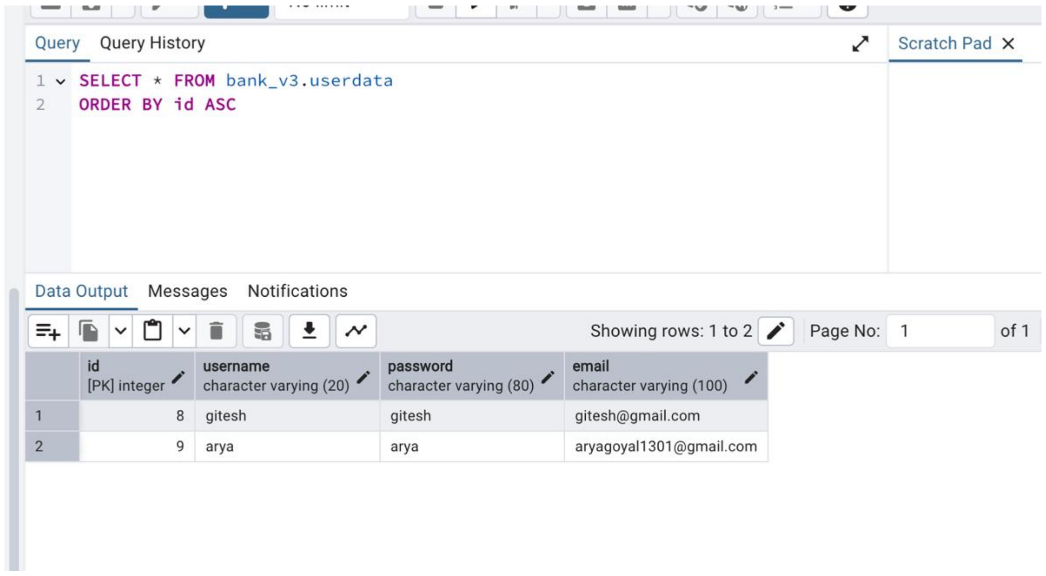

127.0.0.1:5000/dashboard
Client details should be added before making any transaction.
Add Client Details
Make New Transaction
Logout
Manage Users
© 2024 BankAG. All rights reserved.

When the user clicks on the delete button to delete a user, the entry gets removed from the database as well.



ID	Username	Email	Actions
8	gitesh	gitesh@gmail.com	Delete
9	arya	aryagoyal1301@gmail.com	Delete
10	arya_test	aryagoyal1311@gmail.com	Delete

The user arya_test gets deleted from the database since the admin deleted the user from the UI.



```sql
SELECT * FROM bank_v3.userdata
ORDER BY id ASC
```

	id [PK] integer	username character varying (20)	password character varying (80)	email character varying (100)
1	8	gitesh	gitesh	gitesh@gmail.com
2	9	arya	arya	aryagoyal1301@gmail.com

Validation

End-to-End Integration: The application demonstrates a seamless integration of data-driven insights with operational workflows, fulfilling the project's objectives.

Data Governance Compliance: Throughout the development, data governance principles have been adhered to, ensuring data quality, security, and compliance with enterprise goals.

In summary, the developed Flask application represents a comprehensive solution that successfully integrates machine learning insights into a practical, user-centric workflow. It exemplifies the power of combining data-driven programming with robust database management and intuitive user interfaces, thereby achieving the project's goals of enhancing user experience and promoting organizational excellence.

GitHub Link

The entire codebase along with supporting documentation can be found in this GitHub repository:
https://github.com/GiteshChinawalkar/Fraud-transaction-detection/