

**Math and Computer Science**

**Course title: 3784\_MCS7013\_Collaborative Research Project-1**

**Final Project Documentation**

**Student Name:**

**Mougli Pooja Rani 000784790**

**Jayanth Reddy Kuntachinnolla 000789666**

**Date: 04/05/2024**

**LTUBank Documentation**

**Table of Contents**

* Introduction
* Installation
* Usage
* Frontend Overview
* Backend Overview
* SQL Database Overview
* API Documentation
* Troubleshooting
* Machine learning Model Documentation

**1. Introduction**

Customers are crucial for any company. When customers leave, it’s not only impact revenue but also the company's reputation. LTU Bank is experiencing this problem, so they hired our team to create a solution. We're building a user-friendly web app to predict when customers might leave. This involves using a machine learning model to support churn prediction.

**Frontend:**

* Angular
* TypeScript
* HTML
* CSS
* Bootstrap 5

**Backend:**

* ASP.NET Core Web API
* C#
* ADO.NET
* SQL Server
* **Dataset Overview**
* The dataset includes the following fields:
* Customer ID: A unique identifier for each customer
* Surname: The customer's surname or last name
* Credit Score: A numerical value representing the customer's credit score
* Geography: The country where the customer resides (France, Spain or Germany)
* Gender: The customer's gender (Male or Female)
* Age: The customer's age.
* Tenure: The number of years the customer has been with the bank
* Balance: The customer's account balance
* NumOfProducts: The number of bank products the customer uses (e.g., savings account, credit card)
* HasCrCard: Whether the customer has a credit card (1 = yes, 0 = no)
* IsActiveMember: Whether the customer is an active member (1 = yes, 0 = no)
* EstimatedSalary: The estimated salary of the customer
* Exited: Whether the customer has churned (1 = yes, 0 = no)

**2. Installation**

**1. Frontend Development Environment Setup**

* Visual Studio Code (VS Code):
* Download and install VS Code from <https://code.visualstudio.com/download> .
* Follow the installation instructions provided for your operating system.

**2. Backend Development Environment Setup**

* Visual Studio: Download and install Visual Studio <https://visualstudio.microsoft.com/downloads/> .
* Choose the appropriate version based on your requirements (e.g., Visual Studio Community, Professional, Enterprise).
* Follow the installation instructions provided for your operating system.

**3. SQL Server Management Studio (SSMS) Installation**

* Download and install SQL Server Management Studio (SSMS) from <https://learn.microsoft.com/en-us/sql/ssms/download-sql-server-management-studio-ssms?view=sql-server-ver16>
* Choose the appropriate version based on your operating system (Windows).
* Follow the installation instructions provided on the download page.

**3. Usage**

**Starting the Backend Application**

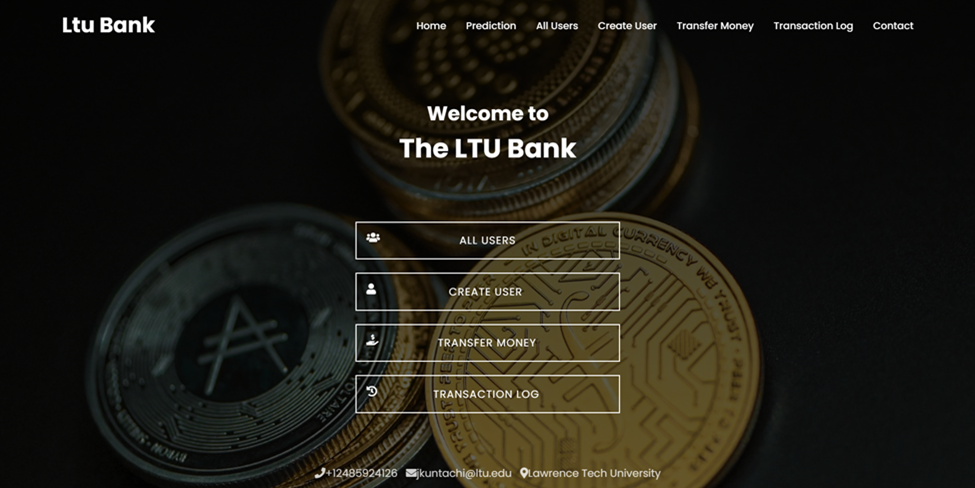
1. Open Visual Studio.
2. Load the backend solution/project.
3. Ensure that the SQL Server instance is running.
4. Click on "IIS EXPRESS" or press F5 to start the backend application.
5. The backend application will connect to the SQL Server database and be ready to handle requests from the frontend.

**Starting the Frontend Application**

1. Open Visual Studio Code.
2. Load the frontend project directory.
3. Open a terminal in VS Code.
4. Run the command **“ng s -o”** to start the Angular development server and open the browser automatically.
5. If needed, run ng b to build the Angular project before starting the server.
6. The frontend application will connect to the backend API, enabling interaction with the SQL Server database through the browser.

**4. Frontend Overview**

Our team started with the design of the web page. The front-end web page was developed with a clear layout and user-friendly interface in mind. The web page was designed with Angular, TypeScript, HTML, CSS, and Bootstrap 5, with a singular focus on clarity and intuitive navigation. Our objective remained unwavering: to deliver a seamless user experience, empowering visitors to seamlessly explore homes, Prediction, Create User, All Users, Transfer Money, and effortlessly engage with the company for assistance.

**Home Page**

**Appearance:**

• Header: Displays the LTU Bank logo and navigation menu.

• Content: Welcomes users with a message and provides quick access buttons.

• Footer: Displays contact information and a link to Lawrence Tech University.

**Functionality:**

• Navigation Menu: Allows users to navigate to different sections of the application.

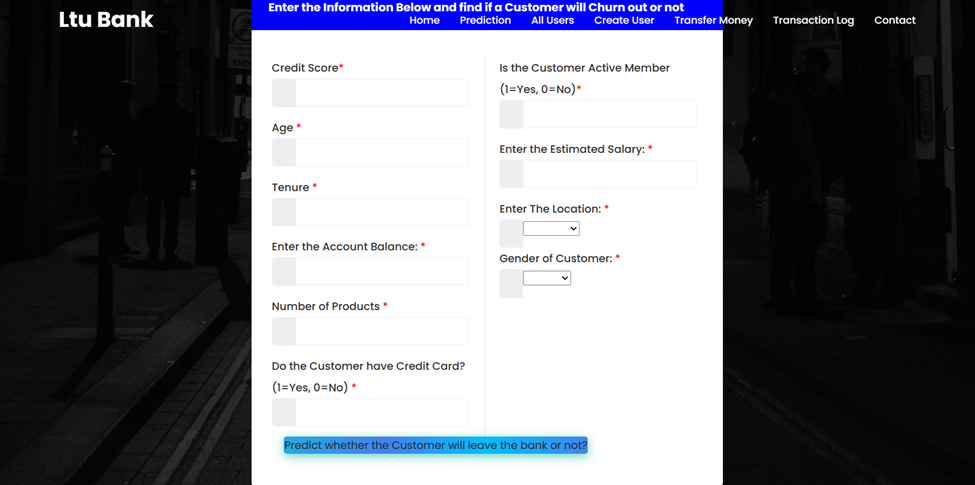
• Quick Access Buttons: Provide shortcuts to key features such as creating users and transferring money.

**Backend Interaction:**

• Endpoints:

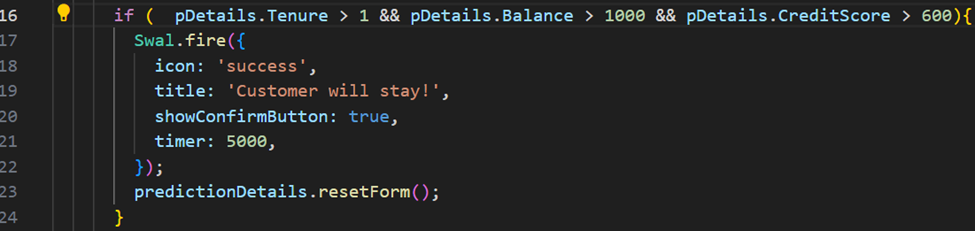
• /api/Home/Navigation: Retrieves navigation menu items.

• /api/Home/WelcomeMessage: Retrieves welcome message.

**Prediction Page**

The prediction page provides a straightforward interface to input customer data and predict churn

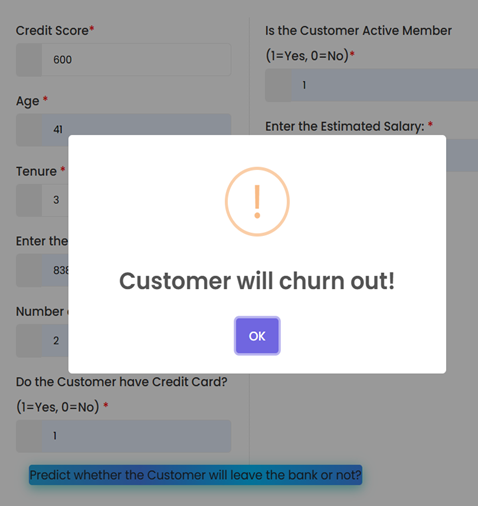
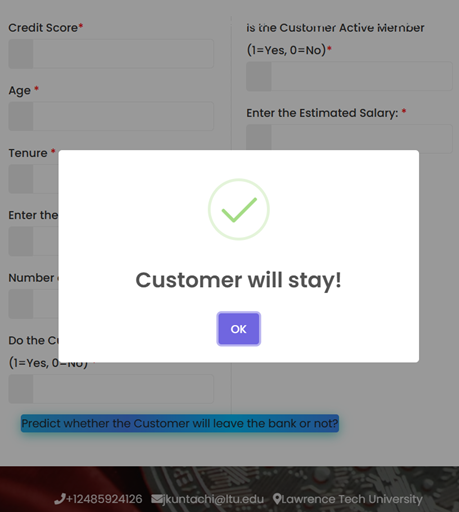
**Appearance:**

• Form: Allows users to input customer data for churn prediction. 

**Functionality:**

• Churn Prediction: Predicts whether a customer will churn based on the provided data.

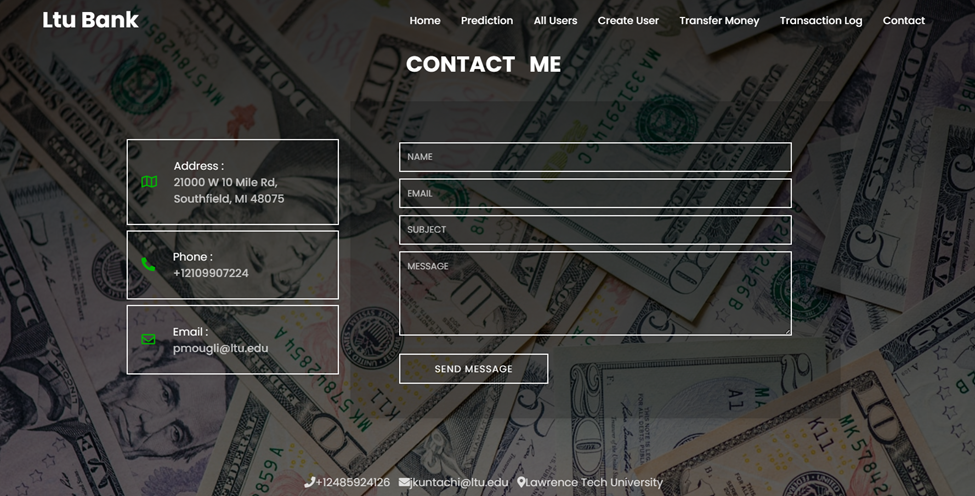
• Conditions Used: The prediction algorithm considers various customer attributes such as credit score, age, tenure, account balance, number of products, presence of a credit card, active membership status, geography, and gender.

****

**Backend Interaction:**

• Endpoint:

• /api/Prediction/PredictChurn: Handles churn prediction.

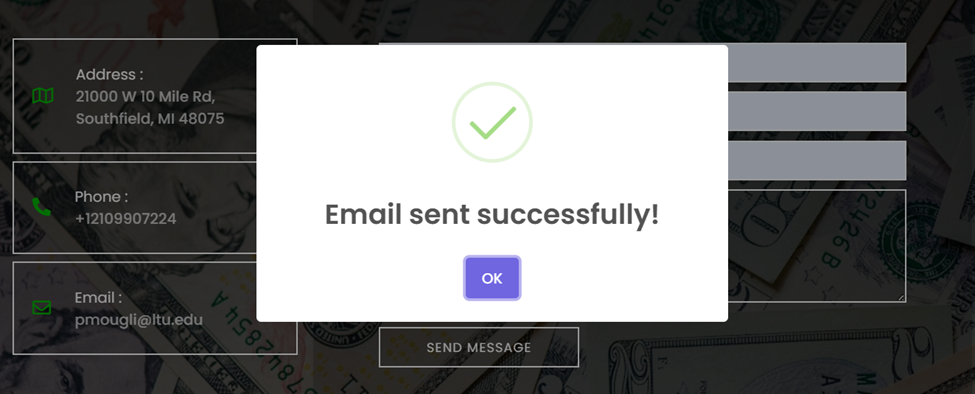
**Contact Page**

The contact page provides users with multiple channels to reach out to the service, ensuring effective communication and support.

**Appearance:**

• Contact Information: Displays contact details for LTU Bank.

• Contact Form: Allows users to send messages to LTU Bank with the following **placeholders:**

* Name
* Email
* Subject
* Message****

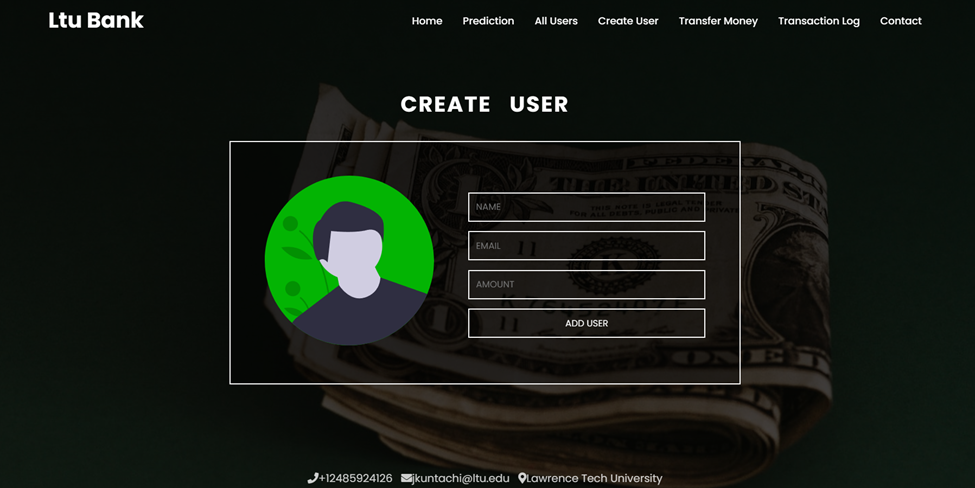
**Functionality:**

• Message Submission: Submits user messages to LTU Bank.

**Backend Interaction:**

• Endpoint:

• /api/Contact/SubmitMessage: Submits contact form data.

**Create User Page**

**Appearance:**

• Form: Allows admins to add new users with name, email, and amount fields.

**Functionality:**

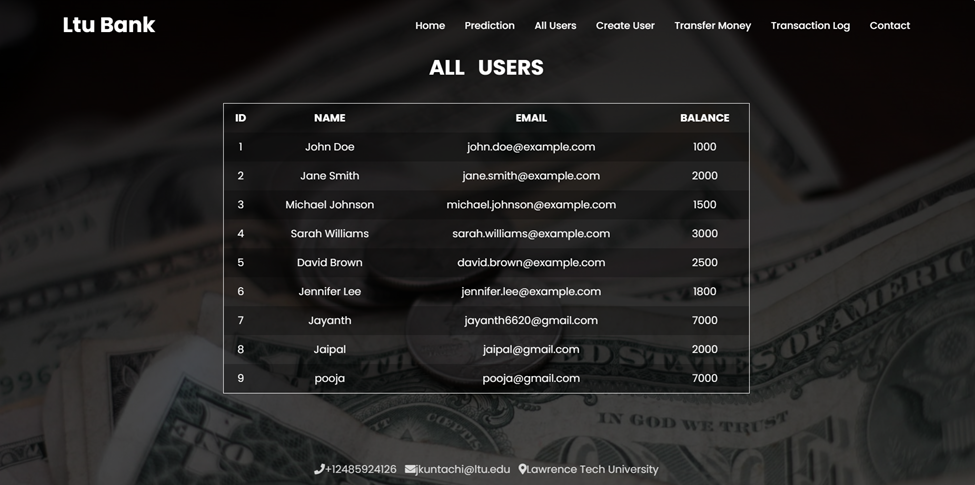
• User Creation: Adds new users to the system.

• Storage: Newly added users are stored in the tbl\_Users table in the SQL database.

**Backend Interaction:**

• Endpoint:

• /api/Users/AddUser: Adds a new user to the database.

**All Users Page**

**Appearance:**

• Table: Displays a list of all users with their details.

**Functionality:**

• User Listing: Shows a comprehensive list of all registered users.

**Backend Interaction:**

• Endpoint:

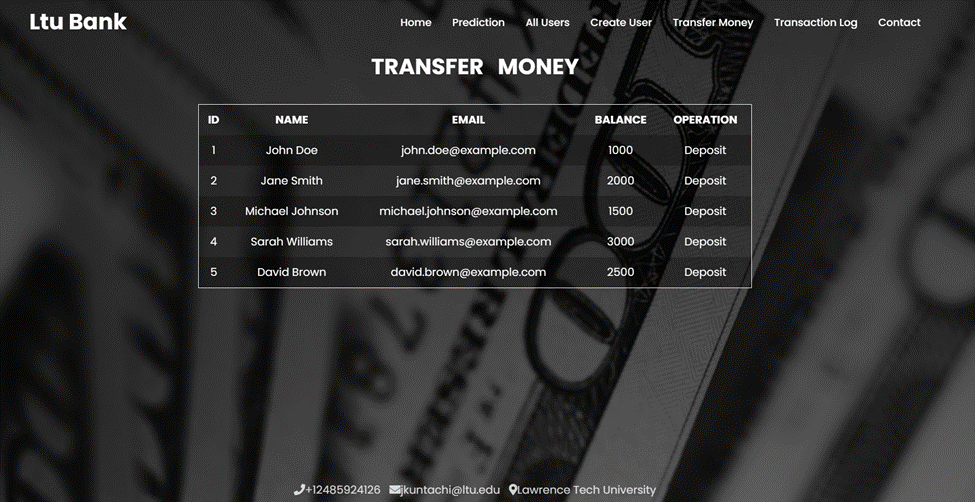
• /api/Users/GetUsersList: Retrieves a list of all users from the database.

**Transfer Money Page & Transaction Log Page**

**Appearance:**

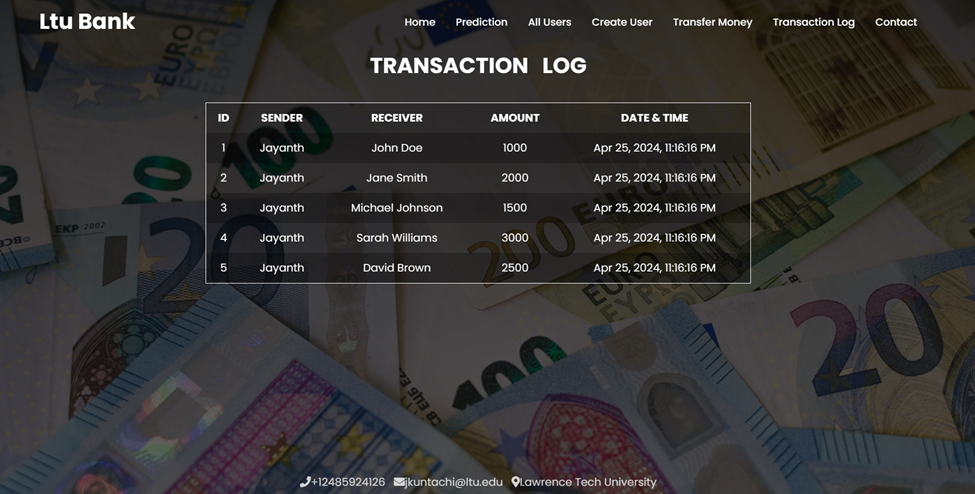
These two pages are usually interconnected, with the Transfer Money page allowing users to initiate money transfers between accounts, and the Transaction Log page displaying logs of all money transactions.

**Transfer Money Page Functionality:**

****

* Users can specify the recipient's email address and the amount to be transferred.
* Upon submission, the form data is sent to the backend API endpoint /api/Users/TransferMoney to process the transaction.
* The backend updates the respective user account balances and records the transaction details in the TransactionLogs table.

**Transaction Log Page Functionality:**

****

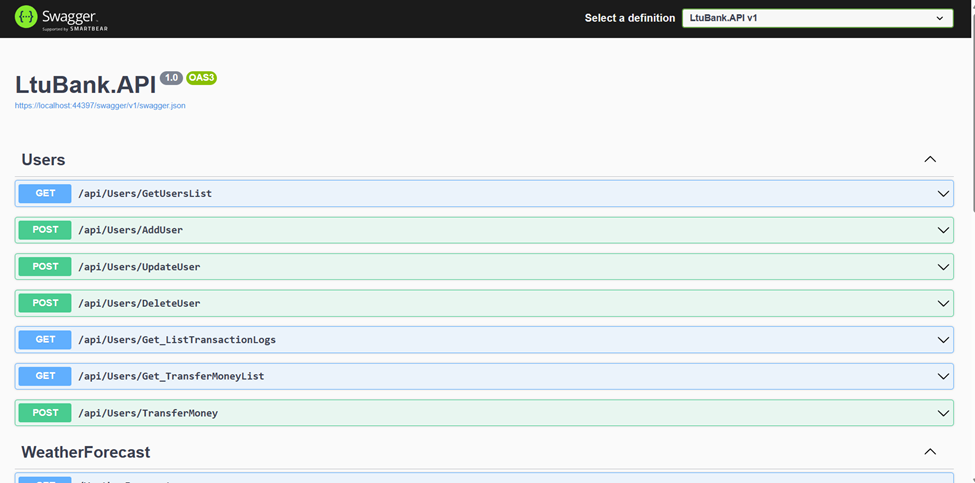
* The Transaction Log page retrieves and displays logs of all money transactions.
* Users can view details such as transaction type (debit or credit), sender, receiver, amount, and transaction date.
* Data is fetched from the backend API endpoint /api/Users/Get\_ListTransactionLogs.

**Backend Interaction:**

* Transfer Money Endpoint: /api/Users/TransferMoney
* Method: POST
* Description: Initiates a money transfer between users. Updates account balances and records transaction details.
* Transaction Log Endpoint: /api/Users/Get\_ListTransactionLogs
* Method: GET
* Description: Retrieves transaction logs from the database.

1. **Backend Overview**

The backend of the LTU Bank project is developed using ASP.NET Core Web API, which serves as the intermediary between the frontend Angular application and the SQL Server Database. It provides endpoints for various operations like fetching user data, adding users, updating users, deleting users, retrieving transaction logs, transferring money, and more.

****

**Functionality**

1. Handling API Requests: The ASP.NET Core Web API handles incoming HTTP requests from the frontend and processes them accordingly.

2. Interacting with SQL Server Database: It interacts with the SQL Server Database using ADO.NET to perform CRUD (Create, Read, Update, Delete) operations and execute stored procedures.

**API Endpoints**

• /api/Users/GetUsersList: Retrieves a list of all users from the database.

• /api/Users/AddUser: Adds a new user to the database.

• /api/Users/UpdateUser: Updates an existing user in the database.

• /api/Users/DeleteUser: Deletes a user from the database.

• /api/Users/Get\_ListTransactionLogs: Retrieves transaction logs from the database.

• /api/Users/Get\_TransferMoneyList: Retrieves money transfer records from the database.

• /api/Users/TransferMoney: Initiates a money transfer between users. Handles the transaction process and updates the database accordingly.

1. **SQL Database Overview**

The SQL Server Database stores various tables and stored procedures to manage user data, transaction logs, and money transfers.

#### Tables:

1. CustomerChurnData:
   * Stores information about customers, including their demographics, account details, and churn status.
   * Fields: RowNumber, CustomerId, Surname, CreditScore, Geography, Gender, Age, Tenure, Balance, NumOfProducts, HasCrCard, IsActiveMember, EstimatedSalary, Exited.
2. tbl\_Users:
   * Contains user details for the application, such as name, email, and account balance.
   * Fields: UserID, Name, Email, Amount, ModifiedBy, ModifiedOn, IsDeleted.
3. TransactionLogs:
   * Records transaction details, including transaction type, sender, receiver, amount, and transaction date.
   * Fields: LogID, TransactionType, Sender, Receiver, Amount, TransactionDate.
4. TransferMoney:
   * Stores information about money transfers, including the sender's and receiver's details, transferred amount, and operation type.
   * Fields: ID, Name, Email, Balance, Operation.

#### Stored Procedures:

1. CreateUser:
   * Creates a new user entry in the tbl\_Users table.
   * Parameters: Name, Email, Amount, ModifiedBy.
2. GetUserDetails:
   * Retrieves user details based on the provided UserID from the tbl\_Users table.
   * Parameters: UserID.
3. ListTransactionLogs:
   * Retrieves a list of all transaction logs from the TransactionLogs table.
4. ListTransferMoney:
   * Retrieves a list of all money transfer records from the TransferMoney table.
5. ListUsers:
   * Retrieves a list of active users from the tbl\_Users table.
6. PredictCustomerChurn (Incomplete):
   * A placeholder for predicting customer churn based on various factors.
   * Parameters: CreditScore, Age, Tenure, AccountBalance, NumberOfProducts, HasCreditCard, IsActiveMember, EstimatedSalary, Location, Gender.
7. SoftDeleteUser:
   * Marks a user as deleted by setting the IsDeleted flag to 1 in the tbl\_Users table.
   * Parameters: UserID.
8. UpdateUser:
   * Updates user information in the tbl\_Users table.

Parameters: UserID, Name, Email, Amount, ModifiedBy

**Frontend Code Integration**

In the frontend Angular application, HTTP requests are made to these API endpoints to perform various operations like fetching user data, adding users, updating users, deleting users, retrieving transaction logs, and transferring money. The frontend communicates with the backend seamlessly to provide a smooth user experience.

Overall, the backend effectively supports the frontend functionality by handling API requests, interacting with the SQL database, and executing necessary operations based on the received requests.

Models <a name="models"></a>

• User: Represents a user account.

• Transaction: Represents a transaction record.

Data Access Layer <a name="data-access-layer"></a>

• UsersRepository: Implements CRUD operations for user accounts.

• TransactionRepository: Implements CRUD operations for transactions.

1. **API Documentation**

The API documentation for the backend project is available within the project itself. It likely includes details about the available endpoints, their methods, parameters, and responses. You can refer to this documentation to understand how to interact with the backend API and make requests.

**Authentication Mechanisms**

• Specify any authentication mechanisms used (e.g., JWT tokens, OAuth2).

**Rate Limiting**

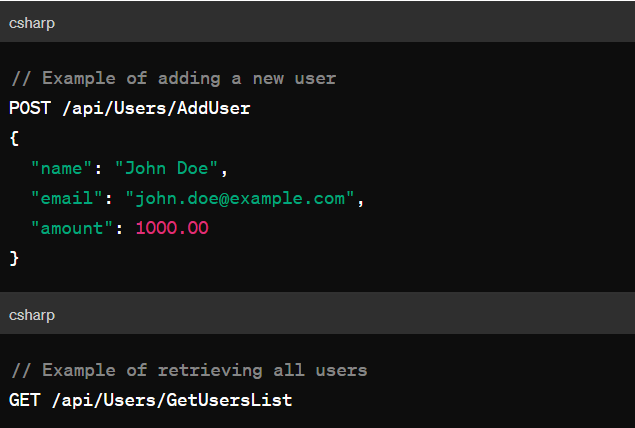
• Describe any rate-limiting policies implemented to prevent abuse of the API.

**Error Handling**

• Explain how errors are handled and returned to clients, including error response formats and status codes.

**Sample Endpoint Usage**

Below are sample code snippets demonstrating the usage of some API endpoints:



Support

• For any assistance or support related to LTU Bank, please contact us at support@ltubank.com. Our support team is dedicated to helping you with any inquiries, issues, or concerns you may have regarding the application. Don't hesitate to reach out to us, and we'll be happy to assist you promptly..

1. **Troubleshooting**

**Backend Application Not Starting**

**Issue:** The backend application fails to start, displaying an error message.

**Resolution:**

Check if all required dependencies are installed and up to date.

Ensure that the database connection string in the backend configuration is correct.

Review the logs for any specific error messages and address them accordingly.

Restart the backend application and try again.

**Frontend Not Connecting to Backend**

**Issue:** The frontend application is unable to connect to the backend API.

**Resolution:**

Verify that the backend server is running and accessible.

Check if the backend API endpoints are correctly configured and accessible.

Ensure that there are no network issues blocking the communication between frontend and backend.

Restart both frontend and backend applications and try again.

**SQL Server Connection Failure**

**Issue:** Unable to establish a connection to the SQL Server database.

**Resolution:**

Check if the SQL Server instance is running and accessible.

Verify the correctness of the SQL Server credentials (username, password).

Ensure that the SQL Server instance allows remote connections if accessing from a different machine.

Test the SQL Server connection using SQL Server Management Studio (SSMS) or other database management tools.

1. **Machine learning Model Documentation**

**Problem Statement:**

1. Despite the continuous efforts of banks to attract and retain customers, the banking industry faces a persistent challenge in the form of customer churn, leading to financial losses and reduced customer satisfaction.

2. It is advantageous for banks to know what leads a client towards the decision to leave the company.

**Project Objective:-**

1. this project aims to analyze the customer churn rate for banks because it is useful to understand why customers leave.

2. After Analyzing we need to train a Machine Learning Model that can find the key factors that significantly influence customer churn or attrition.

3. In the end choose the most reliable model that will attach a probability to the churn to make it easier for customer service to target the right customer to minimize their efforts to prevent customer churn.

**Project Overview:**

1. Churn refers to customers leaving a bank or discontinuing their banking services.

2. Banking Churn Analysis is a process of studying customer behavior in the banking industry to predict and understand customer attrition or churn.

3. Banking Churn Modeling aims to identify patterns and factors that contribute to customer churn, enabling banks to take proactive measures to retain customers and improve customer satisfaction.

**Dataset**

The dataset used in this project is sourced from Kaggle. It contains various features such as credit score, age, tenure, balance, etc., along with the target variable 'Exited' indicating whether the customer churned or not.

**Project Workflow**

**1. Data Exploration and Preprocessing:**

- Loaded the dataset using Pandas.

- Displayed top and bottom rows of the dataset.

- Checked dataset shape, information, and null values.

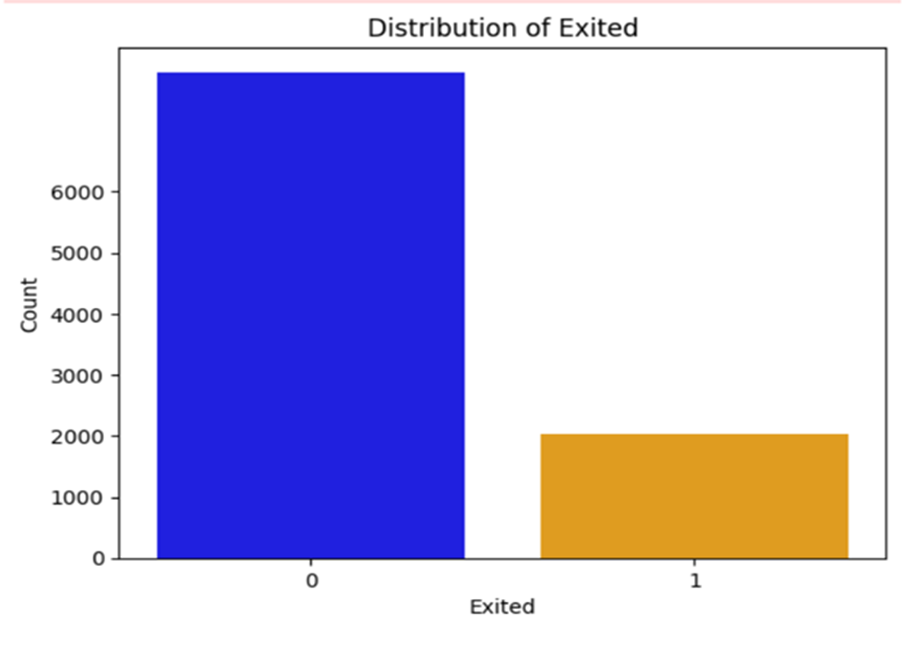
- Dropped irrelevant features like 'RowNumber', 'CustomerId', 'Surname'.

- Encoded categorical data using one-hot encoding.

**2. Handling Imbalanced Data:**

- Checked for imbalance in the target variable.

- Visualized the distribution of the 'Exited' variable using Seaborn.

- Applied Synthetic Minority Over-sampling Technique (SMOTE) to balance the dataset.

**3. Splitting Data and Feature Scaling:**

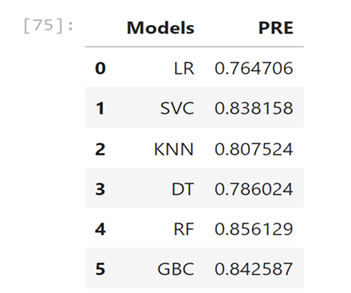
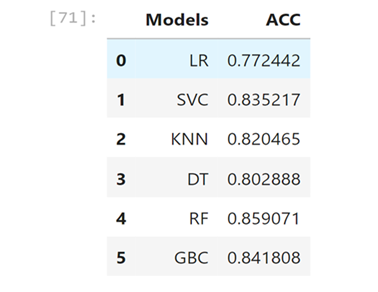
- Split the dataset into training and testing sets.

- Performed feature scaling using StandardScaler.

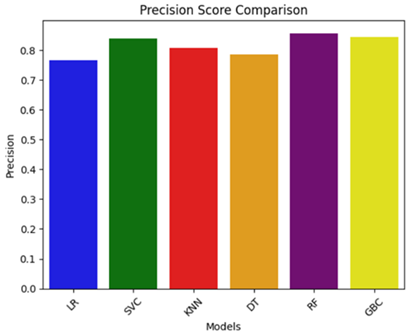
**4. Model Building and Evaluation:**

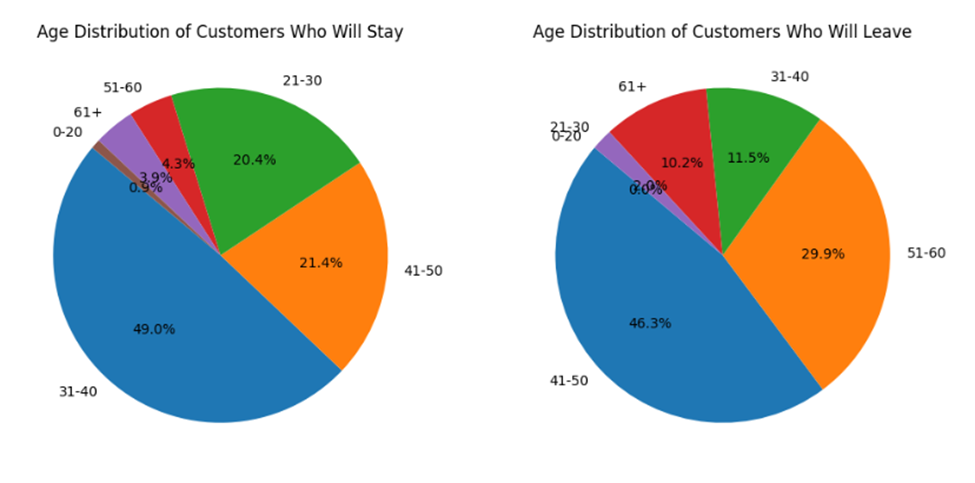
- Trained multiple machine learning models including Logistic Regression, Support Vector Classifier (SVC), K-Nearest Neighbors (KNN), Decision Tree Classifier (DT), Random Forest Classifier (RF), and Gradient Boosting Classifier (GBC).

- Evaluated models using accuracy and precision scores.



- Created a bar plot to compare model accuracies.



****

**5. Model Deployment:**

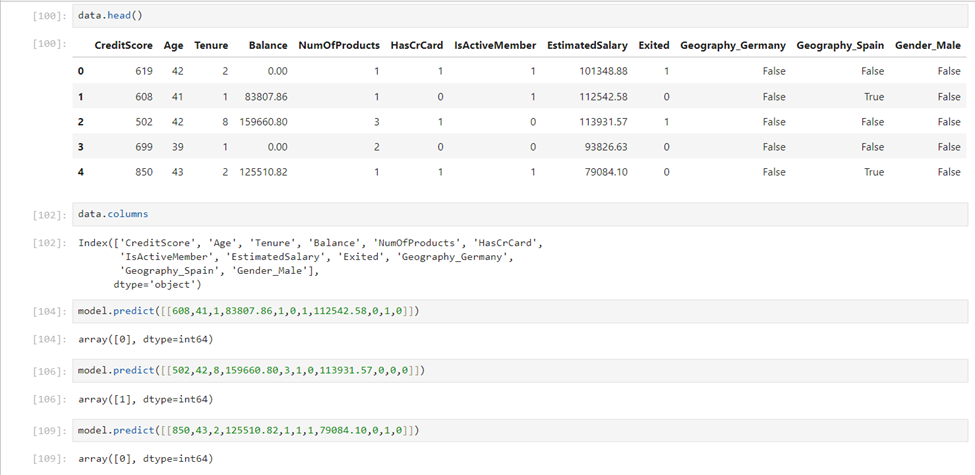
- Saved the Random Forest Classifier model using joblib for future deployment.

**Usage**

Predicted churn for a new bank customer by following steps:

1. Provided the relevant customer information such as credit score, age, tenure, balance, etc.

2. Used trained model to make predictions.



**Machin learning models Conclusion**

This project demonstrates the process of building a churn prediction model using machine learning techniques. By accurately identifying potential churners, businesses can take proactive measures to retain customers, thereby improving customer satisfaction and reducing revenue loss.