

Project Report on

**“BANK BROKERAGE SYSTEM”**

Submitted in partial fulfilment for the award of the degree

Of

**MASTER OF COMPUTER APPLICATION**

**LOVELY PROFESSIONAL UNIVERSITY**

**Submitted By: PAYEL PAUL, ROLL NO – 322100863**

**DECLARATION**

I declare that the project entitled ‘BANK BROKERAGE SYSTEM’ is my own work, conducted under the supervision of the Department of Computer Application, LOVELY PROFESSIONAL UNIVERSITY. Furthermore, I affirm that, to the best of our knowledge, the project does not contain any work that has been submitted for the award of a degree, either in this university or in any other university/deemed university, without proper citations.

**AWKNOWLEDGEMENT**

I wish to express our sincere gratitude to God for His protection, providence, guidance, and above all, for sustaining us. I would like to extend our special thanks of gratitude to our professor, who gave me the golden opportunity to undertake this wonderful project on the topic of ‘BANK BROKERAGE SYSTEM', which also aided me in conducting a lot of research and gaining valuable knowledge.

PAYEL PAUL

**CONTENTS**

* ***Abstract***
* ***Introduction***
* ***Main Objective***
* ***Requirement***
* ***Interface***
* ***Conclusion***

**ABSTRACT**

The Bank Brokerage System is a web application developed using .NET and SQL Server in Visual Studio, designed to manage customer banking activities such as deposits, withdrawals, and account management. It features comprehensive user and bank account operations through dedicated controllers, ensuring a secure and efficient platform for banking operations. The system emphasizes data integrity, scalability, and user-friendly interaction.  
The system aims to provide a secure and efficient platform for banking operations, ensuring data integrity and user-friendly interaction.

**INTRODUCTION**

The Bank Brokerage System is a sophisticated web application developed using the .NET framework, SQL Server, and Visual Studio. The .NET framework provides a versatile and robust platform for building secure, scalable, and high-performance web applications. SQL Server, a relational database management system, ensures efficient data storage, retrieval, and management. Visual Studio, an integrated development environment (IDE), offers comprehensive tools and features for coding, debugging, and deploying applications.

Leveraging these technologies, the Bank Brokerage System is designed to handle critical banking operations such as customer deposits, withdrawals, and account management. The application is structured with dedicated controllers for user and bank account management, ensuring streamlined and efficient operations. The system prioritizes data integrity, security, and a user-friendly interface, making it a reliable solution for managing various banking activities.

**MAIN OBJECTIVE**

The primary objective of the Bank Brokerage System project is to create a secure, efficient, and user-friendly web application that manages various banking operations, including customer deposits, withdrawals, and account management. The system aims to streamline banking processes, ensure data integrity and security, and provide a scalable platform that can accommodate future enhancements and additional features.

**REQUIREMENTS**

* **Functional Requirements**
* User and Account Management: Users can create, view, update, and delete their accounts and bank details, with specific constraints on account structure (e.g., 9-digit BankID, 10-16 character alphanumeric AccountNumber).
* **Non-Functional Requirements**
* Security: Secure data handling with encryption and authentication.
* Performance: Fast response times and efficient data processing.
* Usability: Intuitive, user-friendly interface.
* Scalability: Designed for future growth.
* Reliability: High availability with robust error handling.
* Compatibility: Works across modern web browsers and devices.
* Maintainability: Modular codebase with thorough documentation.
* **Software Requirements**
* OS: Windows 10 or later.
* Tools: Visual Studio, SQL Server, .NET Framework/Core, IIS, Git.
* Languages: C#, HTML, CSS, JavaScript.
* **Hardware Requirements**
* Processor: Intel i5 or equivalent.
* RAM: 8 GB (16 GB recommended).
* Storage: 500 GB SSD.
* Display:1080p resolution.
* Network: Stable internet connection.

This concise requirements section covers the essential functional and non-functional aspects, along with the necessary software and hardware specifications for the successful development and deployment of the system.

# **ARCHITECTURE**

Our Project will be based on ASP.Net MVC architecture.

MVC separates application into three components – Model, View and Controller. Each component is designed to handle different set of responsibility.

Diagram

Description automatically generated

**Model**: Model represents the shape of the data. A class in C# is used to describe a model. Model objects store data retrieved from the database.

Models that will be used inside our project

1. User Model
2. Bank Model

**View**: View in MVC is a user interface. View display model data to the user and also enables them to modify them. View in ASP.NET MVC is HTML, CSS, and some special syntax (Razor syntax) that makes it easy to communicate with the model and the controller.

**View is the User Interface.**

**Controller**: The controller handles the user request. Typically, the user uses the view and raises an HTTP request, which will be handled by the controller. The controller processes the request and returns the appropriate view as a response.

Controllers that will be included in our project.

1. **UserController -** The User Controller will handle all the Bank user related request , it will process the particular request made Through the User Interface and will return the appropriate view as a response . The UserController will have the following Action Methods : -

* AddUser
* GetUser
* UpdateUser
* DeleteUser

1. **BankController -** The BankController will handle all the Bank account related request , it will process the particular request made Through the User Interface and will return the appropriate view as a response . The BankController will have the following Action Methods : -

* AddBankAccounts
* FetchBankAccounts
* UpdateBankAccounts
* DeleteBankAccounts

**IMPLEMENTATION**

**Database Design**  
The database is designed using SQL Server, with the following key tables:

* Users: Stores user details such as UserID, FirstName, LastName, Email, etc.
* BankAccounts: Contains details like AccountID, BankID, AccountNumber, AccountType, AccountOwnerName, Status, etc.
* Transactions: Logs all transactions with fields like TransactionID, UserID, AccountID, TransactionType, Amount, Date, etc.

**Code Implementation**

* UserController.cs: Handles all user-related actions.
* BankController.cs: Manages bank account operations.
* Models: Define the data structures for users, bank accounts, and transactions.
* Views: HTML and Razor views for rendering the user interface.
* Services: Business logic for processing data and handling application workflows.

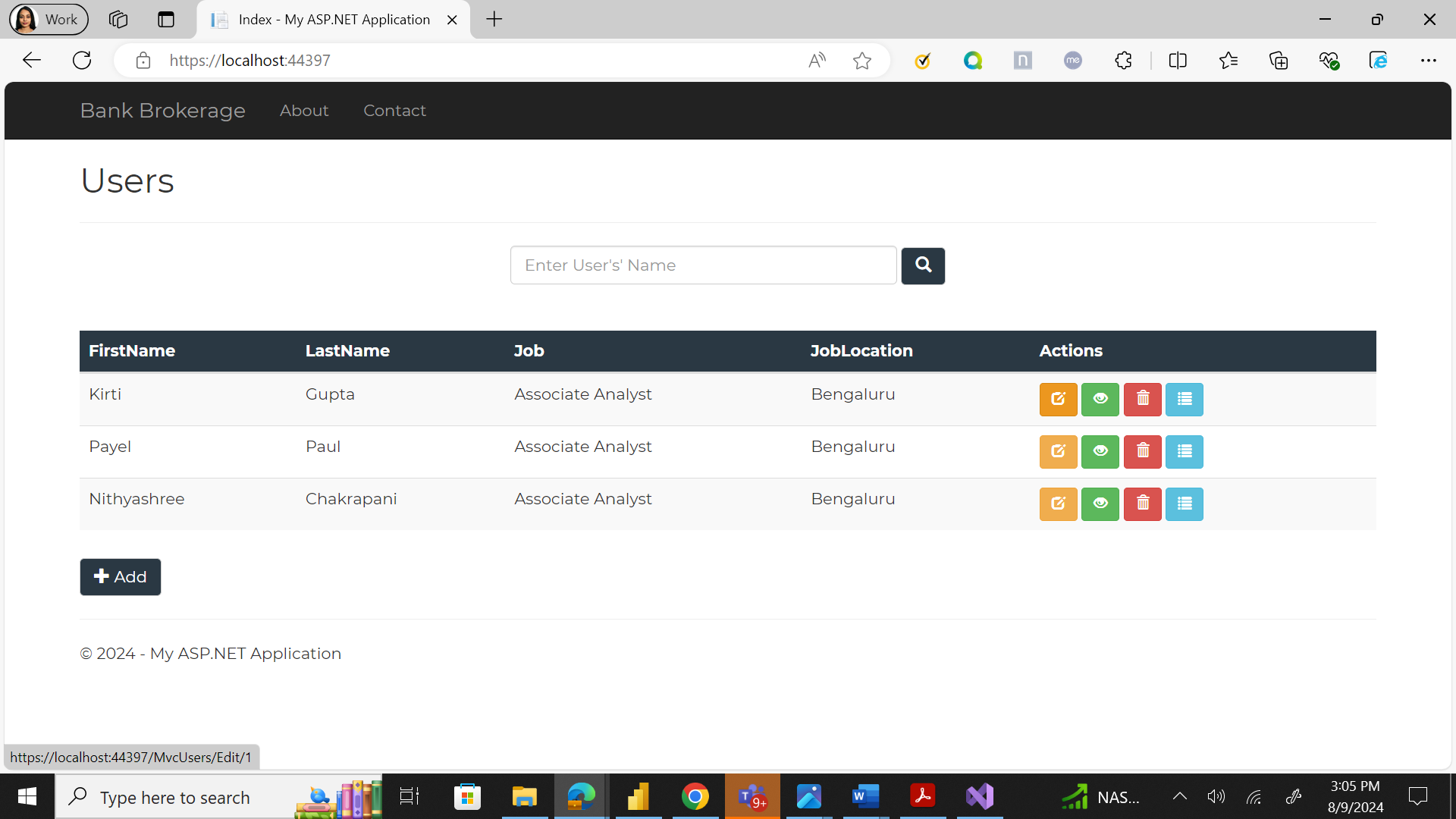
**Testing**

* **Unit Testing**  
  - Tests were conducted on individual components like controllers, models, and services to ensure correct functionality.  
  - Test cases included creating, updating, retrieving, and deleting user and bank account data.
* **Integration Testing**  
  - End-to-end tests ensured that the system's components interact as expected.  
  - Scenarios included user registration, account creation, and transaction processing.
* **User Acceptance Testing (UAT)**  
  - Final testing was conducted with end users to validate the system’s usability and functionality.  
  - Feedback was collected to make final adjustments before deployment.

**Maintenance**

The system is designed for easy maintenance, with modular code and thorough documentation. Regular updates and patches are planned to address any issues, improve performance, and add new features.

**INTERFACE**



**CONCLUSION**

The Bank Brokerage System successfully meets its objectives of providing a secure, efficient, and user-friendly platform for managing banking operations. Its scalable architecture and modular design ensure it can grow with the needs of the users, making it a reliable solution for modern banking needs.