## CHAPTER 1

**INTRODUCTION**

**1.1 OVERVIEW OF THE PROJECT**

Nowadays the use of bank applications has increased and many banking operations are done online. In this paper we proposed a multi banking system using web services (for transactions). It is developed for those customers who are having multiple accounts in various banks. Multi-banking interface is a web-based application through which customer can access his/her multiple accounts with single user id and password. The customer does not have to remember all the user id and password for each account he/she has in a banking website. The objective of this application is to allow the Customers of various Banks access their accounts and make transactions using this solution. They need not interact with various bank applications or web sites. The customer can login with single id and password. In addition to this, the customer has certain privileges such as he/she can view his/her transaction and account details and also make fund transfers from one bank account to other. Also, user can create various bank Account in a single application.

## CHAPTER 2

**SYSTEM ANALYSIS**

**2.1 EXISTING SYSTEM**

Currently we are having lot of banks in the market and any person can do transactions of any individual bank either manually or in online. But no one can do all banks transactions in a single portal or in single bank.

**Disadvantages**

* Customer need to use multi software for different account
* Waste of time for just create an bank account
* Single workers can able to handle the customer account

## 2.2 PROPOSED SYSTEM

The Multi Banking System Interface is targeted to the future banking solution for the users who is having multiple bank accounts in multiple banks. This interface integrates all existing banks and provides business solutions for both retail and corporate.

**Advantages**

* By using this portal any client who maintain accounts in various banks can directly log on to Multi Banking System Interface and make any kind of transactions.
* Can access different bank account details in an single application
* All the action has taken care by single application

## 2.3 FEASIBILITY STUDY

The feasibility of the system is analyzed in this phase and business proposal is put forth with general plan for the project and cost estimates. During the system analysis of the project, the feasibility study of proposed system is to be carried out. For feasibility analysis, some understanding of the major requirements for the system is essential. Three key considerations involved in feasibility analysis are

* + - Technical Feasibility
    - Economic Feasibility
    - Operational Feasibility

## 2.3.1 Technical Feasibility

Technical feasibility assesses the current resources (such as hardware and software) technology, which are required to accomplish user requirements in the software within the allocated time and budget. For this, whether the certain current resources and technology can be upgraded or added in the software to accomplish specified user requirements.

The technical requirements of the application are simple and basic. Python is used for the developers of the application and the framework is largely used by many, thus there will be enough support for future enhancements. The framework is stable and the support from the developers is constantly updated. The devices which have internet connectivity are enough for the application.

## 2.3.2 Economic Feasibility

Economic feasibility determines whether the required software is capable of generating financial gains for an organization. It involves the cost incurred on the software development team, estimated cost of hardware and software, cost of performing feasibility study and so on.

The cost of application development is very less and the cost of implementation is also less. It can be developed with the system with minimum requirements and can also be operated with the system with some basic requirements that are available the existing systems. For this, it is essential to consider expenses made on purchases and activities required to carry out software development.

## 2.3.3 Operational Feasibility

Operational feasibility assesses the extent to which the required software performs a series of steps to solve user requirements. This feasibility is dependent on developer and involves visualizing whether the software will operate after it has been developed and be operative.

The application is developed based on the user requirements and is developed on the priority of the user requirements such as an integrated service and reviewing platform.

## 2.4 PROBLEM ANALYSIS

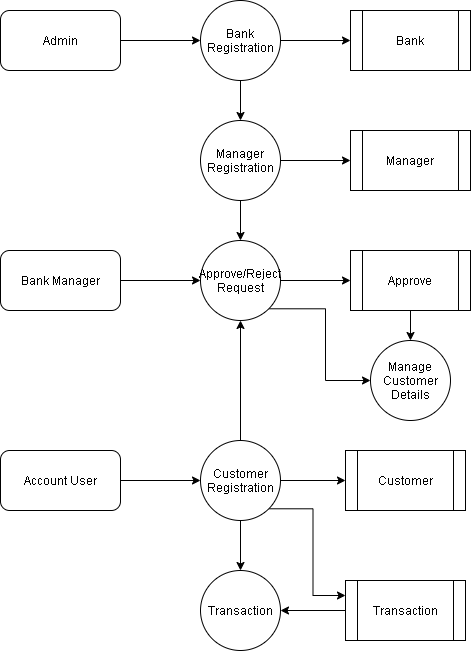
The orders from public are submitted through email, letter or form to the corresponding authority. The registered orders are forwarded to corresponding authority and report about the orders is submitted. The status of orders addressed is not recorded and maintained properly. The maps are not implemented. So, the customer does not know live location of the package. There is need for the software which receives the orders through online, forwarded to respective peoples for rectification and post the status of orders.

## 2.5 CONTEXT AND DATA FLOW DIAGRAM

A data-flow diagram (DFD) is a way of representing a flow of a data of a process or system. The DFD also provides information about the outputs and inputs of each entity and process itself. A data-flow diagram is a part of structured-analysis modeling tools.



**LEVEL 1:**



**2.6 SYSTEM CONFIGURATION**

### 2.6.1 Hardware Requirements

Processor : P 4 700 GHz

RAM Capacity : 4GB

Hard Disk : 180GB

### 2.6.2 Software Requirements

Operating System : Windows 8,10

Front End : JAVA

Back End : SQL

### SoftwareDescriptions

### Java

Java is a high-level programming language developed by Sun Microsystems. It was originally designed for developing programs for set-top boxes and handheld devices, but later became a popular choice for creating web applications.

The Java syntax is similar to C++, but is strictly an object-oriented programming language. For example, most Java programs contain classes, which are used to define objects, and methods, which are assigned to individual classes. Java is also known for being stricter than C++, meaning variables and functions must be explicitly defined. This means Java source code may produce errors or "exceptions" more easily than other languages, but it also limits other types of errors that may be caused by undefined variables or unassigned types.

Unlike Windows executables (.EXE files) or Macintosh applications (.APP files), Java programs are not run directly by the operating system. Instead, Java programs are interpreted by the Java Virtual Machine, or JVM, which runs on multiple platforms. This means all Java programs are multiplatform and can run on different platforms, including Macintosh, Windows, and Unix computers. However, the JVM must be installed for Java applications or applets to run at all. Fortunately, the JVM is included as part of the Java Runtime Environment (JRE),

### My SQL

My SQL is the popular Open-Source Relational SQL Database Management System. My SQL is being used for developing various web-based software applications. The My SQL development project has made its [source code](https://en.wikipedia.org/wiki/Source_code) available under the terms of the [General Public](https://en.wikipedia.org/wiki/GNU_General_Public_License) License. It is used to store the information.

My SQL was owned and sponsored by the single for-profit firm, the Swedish company My SQL AB, now owned by Oracle Corporation. For proprietary use, several paid editions are available, and offer additional functionality.

## CHAPTER 3

**3. SYSTEM DESIGN**

**3.1 INPUT DESIGN**

Input Design is the process of converting a user-oriented description of the input into a computer-based system. The goal of designing input is to make data entry easier and to be free from errors. The data entry screen is designed in such a way that all the data manipulates can be performed. Input Design is the first phase in the system design. Input designing is to converting the user-oriented information to the computer- oriented form. The input data items are grouped and analyzed to find out whether the proposed system can be developed from the user input. The system is developed using various processes screens formats.

The main objective of input design is to

* + - It should serve specific purpose effectively such assorting, recording, and retrieving their formation.
    - It ensures proper completion with accuracy.
    - It should be easy to fill and straight forward.
    - It should focus on user’s attention, consistency and simplicity.

### 3.2 DATABASE DESIGN

The most important consideration in designing the database is how the information will be used. The main objective of designing a database is Data Integration, Data Integrity and Data Independence.

### Data Integration

In a database, information from several files is coordinated, accessed and operated up on through it is in a single file. Logically, their formation is centralized, physically, the data may be located on different devices, connected through data communication facilities.

### Data Integrity

Data integrity means storing all data in one place only and how each application access it. This approach results in more consistent information, one update being sufficient to achieve a new record status for all applications. This leads to less data redundancy, that is data items need not be duplicated. A reduction in the direct access storage requirement.

### Data Independence

Data in dependence is the insulation of application programs from changing aspects of physical data organization. This objective seeks to allow changes in the content and organization of physical data without reprogramming of application and allow modifications to application programs without reorganizing the physical data.

### 3.2.1 TABLE DESIGN

The Table needed for each module were designed and the specification of each and every column was given based on the records and details collected during record specification of the system study.

**TABLE NAME: BANK**

|  |  |  |  |
| --- | --- | --- | --- |
| **FIELD** | **DATA TYPE** | **SIZE** | **CONSTRAINT** |
| Bank id | Int | 10 | Primary key |
| Bankname | Varchar | 20 | Not null |
| Brachname | Varchar | 20 | Not null |
| Address | Varchar | 30 | Not null |
| Landline | Int | 10 | Not null |
| Ifsccode | Varchar | 15 | Not null |

**TABLE NAME: MANAGER**

|  |  |  |  |
| --- | --- | --- | --- |
| **FIELD** | **DATA TYPE** | **SIZE** | **CONSTRAINT** |
| Manager id | Int | 10 | Primary key |
| Bank id | Int | 10 | Foreign key |
| Brach name | Varchar | 30 | Not null |
| Mobile | Int | 10 | Not null |
| Landline | Int | 10 | Not null |

**TABLE NAME: CUSTOMER**

|  |  |  |  |
| --- | --- | --- | --- |
| **FIELD** | **DATA TYPE** | **SIZE** | **CONSTRAINT** |
| Customer id | Int | 10 | Primary key |
| Firstname | Varchar | 20 | Not null |
| Lastname | Varchar | 20 | Not null |
| Mobile | Int | 10 | Not null |
| Aadhar | Int | 20 | Not null |
| License | Varchar | 20 | Not null |
| Age | Int | 2 | Not null |
| Gender | Varchar | 7 | Not null |
| Address 1 | Varchar | 30 | Not null |
| Address 2 | Varchar | 30 | Not null |
| City | Varchar | 10 | Not null |
| State | Varchar | 10 | Not null |
| Pincode | Int | 6 | Not null |

**TABLE NAME: Transaction**

|  |  |  |  |
| --- | --- | --- | --- |
| **FIELD** | **DATA TYPE** | **SIZE** | **CONSTRAINT** |
| Transaction id | Int | 10 | Primary key |
| Customer id | Int | 10 | Foreign key |
| Receiver id | Int | 10 | Foreign key |
| Ifsc | Varchar | 15 | Not null |
| Account number | Int | 10 | Not null |
| Amount | Int | 10 | Not null |
| Date time | Date | 10 | Not null |

**TABLE NAME: APPROVE**

|  |  |  |  |
| --- | --- | --- | --- |
| **FIELD** | **DATA TYPE** | **SIZE** | **CONSTRAINT** |
| Approve id | Int | 10 | Primary key |
| Bank id | Int | 10 | Foreign key |
| Customer id | Int | 10 | Foreign key |
| Approve status | Varchar | 7 | Not null |

* 1. **3.3 MODULE DESCRIPTION**

The main module in this project are listed below

* **Bank Registration**
* **Manager Registration**
* **Customer Registration**
* **Approve and Reject request**
* **Create Bank Account**
* **Transaction**

1. **Bank Registration**

Admin first login into the banking system and create linked bank details, once the admin creates the bank it will shown in the all the requested places. Admin provide all the information about the bank.

1. **Manager Registration**

Admin also have access to create manager under respective banks. this is a big process to add the manager details. Once the manager allots to the bank, he gave access to managing the customer details.

1. **Customer Registration**

Customer manually select the bank and give request to the respective bank manager; manager decide to check the customer details and give approve request. Once the customer got access the account will be created.

1. **Approve and Reject request**

This module will be handling by the bank manager, once manager get customer account request, just verified and do the action for approve or reject.

1. **Create Bank Account**

By default, it will create after the manager approving customer request. Once the account has created the customer can able to login and check the account details.

1. **Transaction**

This module will follow the customer can do the transaction and view the transaction details.Customer can transfer the amount to the another bank or same bank customers.

## CHAPTER 4

## 4 SYSTEM TESTING

Testing is an integral part of any system development lifecycle. Insufficient and untested applications may tend to crash and the result is loss of economic and manpower investment besides user's dissatisfaction and downfall of reputation. Software testing can be looked upon as one among many processes, an organization performs, and that provides the lost opportunity to correct any flaws in the developed system. Software testing includes selecting test data that have more probability of giving errors.

The first step in system testing is to develop a plan that tests all aspects of the system. Completeness, correctness, reliability and maintainability of the software are to be tested for the best quality assurance that the system meets the specification and requirements for its intended use and performance. System testing is the most useful practical process of executing a program with the implicit intention of finding errors that make the program fails. System testing is done in three phases.

* + - * Unit Testing
      * Integration Testing
      * Validation Testing

### 4.1 UNIT TESTING

Unit testing focuses verification effort on the smallest unit of software the module. Using the detailed design and the process specification testing is done to registration by the user with in the boundary of the Login module. The login form receives the username and password details and validates the value with the database. If valid, the home page is displayed.

### 4.2 INTEGRATION TESTING

IntegrationTestingistheprocessofthisactivitycanbeconsideredastesting thedesignandhencemoduleinteraction.Theprimaryobjectiveofintegrationtesting is todiscovererrorsintheinterfacesbetweenthecomponents.Loginformandregistration form are integrated and tested together. If the user is newly registered, the received details will be stored in the registration table. While logging in, the application will check for valid user name and password in the registration table and if valid the user is prompted for submittingcomplaints.

### 4.3 VALIDATION TESTING

Validation are independent procedures that are used together for checking that a product, service, or system meets [requirements](https://en.wikipedia.org/wiki/Requirement) and [specifications](https://en.wikipedia.org/wiki/Specification_(technical_standard)) and that it fulfills its in purpose the actual result from the expected result for the complaint process. Select the complaint category of the complaint by user. The input given to various forms fields are validated effectively. Each module is tested in dependently. It is tested that the complaint module fields receive the correct input for the necessary details such as complaint category, complaint id, reference name, complaint description, email for further process.

## CHAPTER 5

* 1. **CONCLUSION**

The project title is “Multibank System” is a web-based application. Every day banks need to perform many activities related to users which needs huge infrastructure with more staff members etc. But the online banking system allows the banks to perform these activities in a simpler way without involving the employees for example consider online banking

## APPENDICES APPENDIX I SAMPLE CODE

package com.example.demo.controller;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.http.ResponseEntity;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.PathVariable;

import org.springframework.web.bind.annotation.PostMapping;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RestController;

import com.example.demo.dao.ApiDao;

import com.example.demo.response.GetBankResponse;

import com.example.demo.response.GetCitizenResponse;

import com.example.demo.response.GetComplaintResponse;

import com.example.demo.response.GetCustomerBank;

import com.example.demo.service.ApiService;

@RestController

@RequestMapping(value = { "/api" })

public class ApiController {

@Autowired

ApiService service;

@Autowired

ApiDao dao;

@GetMapping("/login/{username}/{password}")

public Boolean login(@PathVariable String username,@PathVariable String password) {

return service.login(username,password);

}

@GetMapping("/manager\_login/{username}/{password}")

public Boolean manager\_login(@PathVariable String username,@PathVariable String password) {

return dao.manager\_login(username,password);

}

@PostMapping("/add\_bank/{bankname}/{address}/{ifsccode}/{landline}")

public String bank\_register(@PathVariable String bankname,

@PathVariable String address,

@PathVariable String ifsccode,

@PathVariable String landline) {

dao.bank\_register(bankname,address,ifsccode,landline);

return "Bank Saved Sucessfully";

}

@PostMapping("/add\_manager/{bankid}/{mobile}/{landline}/{username}/{password}")

public String add\_manager(@PathVariable Integer bankid,

@PathVariable String mobile,

@PathVariable String landline,

@PathVariable String username,

@PathVariable String password) {

dao.add\_manager(bankid,mobile,landline,username,password);

return "Manager Added Sucessfully";

}

@GetMapping("/get\_bank")

public ResponseEntity<List<GetBankResponse>> get\_bank() {

return ResponseEntity.ok().body(service.get\_bank());

}

@PostMapping("/add\_customer/{fname}/{lname}/{mobile}/{aadhar}/{idproof}/{age}/{gender}/{address1}/{address2}/{city}/{state}/{pincode}")

public String add\_manager(@PathVariable String fname,

@PathVariable String lname,

@PathVariable String mobile,

@PathVariable String aadhar,

@PathVariable String idproof,

@PathVariable String age,

@PathVariable String gender,

@PathVariable String address1,

@PathVariable String address2,

@PathVariable String city,

@PathVariable String state,

@PathVariable String pincode) {

dao.add\_customer(fname,lname,mobile,aadhar,idproof,age,gender,address1,address2,city,state,pincode);

return "Customer Added Sucessfully";

}

@PostMapping("/add\_bank\_account/{customer\_id}/{bank\_id}/{account type}")

public String add\_bank\_account(@PathVariable Integer customer\_id,

@PathVariable String bank\_id,

@PathVariable String account\_type) {

dao.add\_customer(customer\_id,bank\_id,account\_type);

return "Bank Request send Sucessfully";

}

@GetMapping("/get\_customer\_bank/{customer\_id}")

public ResponseEntity<List<GetCustomerBank>> get\_customer\_bank(@PathVariable Integer customer\_id) {

return ResponseEntity.ok().body(service.get\_customer\_bank(customer\_id));

}

}

package com.example.demo.controller;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.http.ResponseEntity;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.PathVariable;

import org.springframework.web.bind.annotation.PostMapping;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RestController;

import com.example.demo.dao.ApiDao;

import com.example.demo.response.GetBankResponse;

import com.example.demo.response.GetCitizenResponse;

import com.example.demo.response.GetComplaintResponse;

import com.example.demo.response.GetCustomerBank;

import com.example.demo.service.ApiService;

@RestController

@RequestMapping(value = { "/api" })

public class ApiController {

@Autowired

ApiService service;

@Autowired

ApiDao dao;

@GetMapping("/login/{username}/{password}")

public Boolean login(@PathVariable String username,@PathVariable String password) {

return service.login(username,password);

}

@GetMapping("/manager\_login/{username}/{password}")

public Boolean manager\_login(@PathVariable String username,@PathVariable String password) {

return dao.manager\_login(username,password);

}

@PostMapping("/add\_bank/{bankname}/{address}/{ifsccode}/{landline}")

public String bank\_register(@PathVariable String bankname,

@PathVariable String address,

@PathVariable String ifsccode,

@PathVariable String landline) {

dao.bank\_register(bankname,address,ifsccode,landline);

return "Bank Saved Sucessfully";

}

@PostMapping("/add\_manager/{bankid}/{mobile}/{landline}/{username}/{password}")

public String add\_manager(@PathVariable Integer bankid,

@PathVariable String mobile,

@PathVariable String landline,

@PathVariable String username,

@PathVariable String password) {

dao.add\_manager(bankid,mobile,landline,username,password);

return "Manager Added Sucessfully";

}

@GetMapping("/get\_bank")

public ResponseEntity<List<GetBankResponse>> get\_bank() {

return ResponseEntity.ok().body(service.get\_bank());

}

@PostMapping("/add\_customer/{fname}/{lname}/{mobile}/{aadhar}/{idproof}/{age}/{gender}/{address1}/{address2}/{city}/{state}/{pincode}")

public String add\_manager(@PathVariable String fname,

@PathVariable String lname,

@PathVariable String mobile,

@PathVariable String aadhar,

@PathVariable String idproof,

@PathVariable String age,

@PathVariable String gender,

@PathVariable String address1,

@PathVariable String address2,

@PathVariable String city,

@PathVariable String state,

@PathVariable String pincode) {

dao.add\_customer(fname,lname,mobile,aadhar,idproof,age,gender,address1,address2,city,state,pincode);

return "Customer Added Sucessfully";

}

@PostMapping("/add\_bank\_account/{customer\_id}/{bank\_id}/{account type}")

public String add\_bank\_account(@PathVariable Integer customer\_id,

@PathVariable String bank\_id,

@PathVariable String account\_type) {

dao.add\_customer(customer\_id,bank\_id,account\_type);

return "Bank Request send Sucessfully";

}

@GetMapping("/get\_customer\_bank/{customer\_id}")

public ResponseEntity<List<GetCustomerBank>> get\_customer\_bank(@PathVariable Integer customer\_id) {

return ResponseEntity.ok().body(service.get\_customer\_bank(customer\_id));

}

}package com.example.demo.configuration;

import java.util.Properties;

import javax.sql.DataSource;

import org.springframework.beans.factory.annotation.Value;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import org.springframework.jdbc.datasource.DriverManagerDataSource;

import org.springframework.orm.hibernate5.HibernateTransactionManager;

import org.springframework.orm.hibernate5.LocalSessionFactoryBean;

import org.springframework.transaction.annotation.EnableTransactionManagement;

@Configuration

@EnableTransactionManagement

public class HibernateConfiguration {

@Value("${db.driver}")

private String DB\_DRIVER;

@Value("${db.password}")

private String DB\_PASSWORD;

@Value("${db.url}")

private String DB\_URL;

@Value("${db.username}")

private String DB\_USERNAME;

@Value("${hibernate.dialect}")

private String HIBERNATE\_DIALECT;

@Value("${hibernate.show\_sql}")

private String HIBERNATE\_SHOW\_SQL;

// @Value("${hibernate.hbm2ddl.auto}")

private String HIBERNATE\_HBM2DDL\_AUTO;

@Value("${entitymanager.packagesToScan}")

private String ENTITYMANAGER\_PACKAGES\_TO\_SCAN;

@Bean

public LocalSessionFactoryBean sessionFactory() {

LocalSessionFactoryBean sessionFactory = new LocalSessionFactoryBean();

sessionFactory.setDataSource(dataSource());

sessionFactory.setPackagesToScan(ENTITYMANAGER\_PACKAGES\_TO\_SCAN);

Properties hibernateProperties = new Properties();

hibernateProperties.put("hibernate.dialect", HIBERNATE\_DIALECT);

hibernateProperties.put("hibernate.show\_sql", HIBERNATE\_SHOW\_SQL);

// hibernateProperties.put("hibernate.hbm2ddl.auto", HIBERNATE\_HBM2DDL\_AUTO);

sessionFactory.setHibernateProperties(hibernateProperties);

return sessionFactory;

}

@Bean

public DataSource dataSource() {

DriverManagerDataSource dataSource = new DriverManagerDataSource();

dataSource.setDriverClassName(DB\_DRIVER);

dataSource.setUrl(DB\_URL);

dataSource.setUsername(DB\_USERNAME);

dataSource.setPassword(DB\_PASSWORD);

return dataSource;

}

@Bean

public HibernateTransactionManager transactionManager() {

HibernateTransactionManager txManager = new HibernateTransactionManager();

txManager.setSessionFactory(sessionFactory().getObject());

return txManager;

}

}package com.example.demo.configuration;

import org.springframework.context.annotation.Configuration;

import org.springframework.web.servlet.config.annotation.CorsRegistry;

import org.springframework.web.servlet.config.annotation.EnableWebMvc;

import org.springframework.web.servlet.config.annotation.WebMvcConfigurerAdapter;

@Configuration

@EnableWebMvc

public class WebConfig extends WebMvcConfigurerAdapter {

@Override

public void addCorsMappings(CorsRegistry registry) {

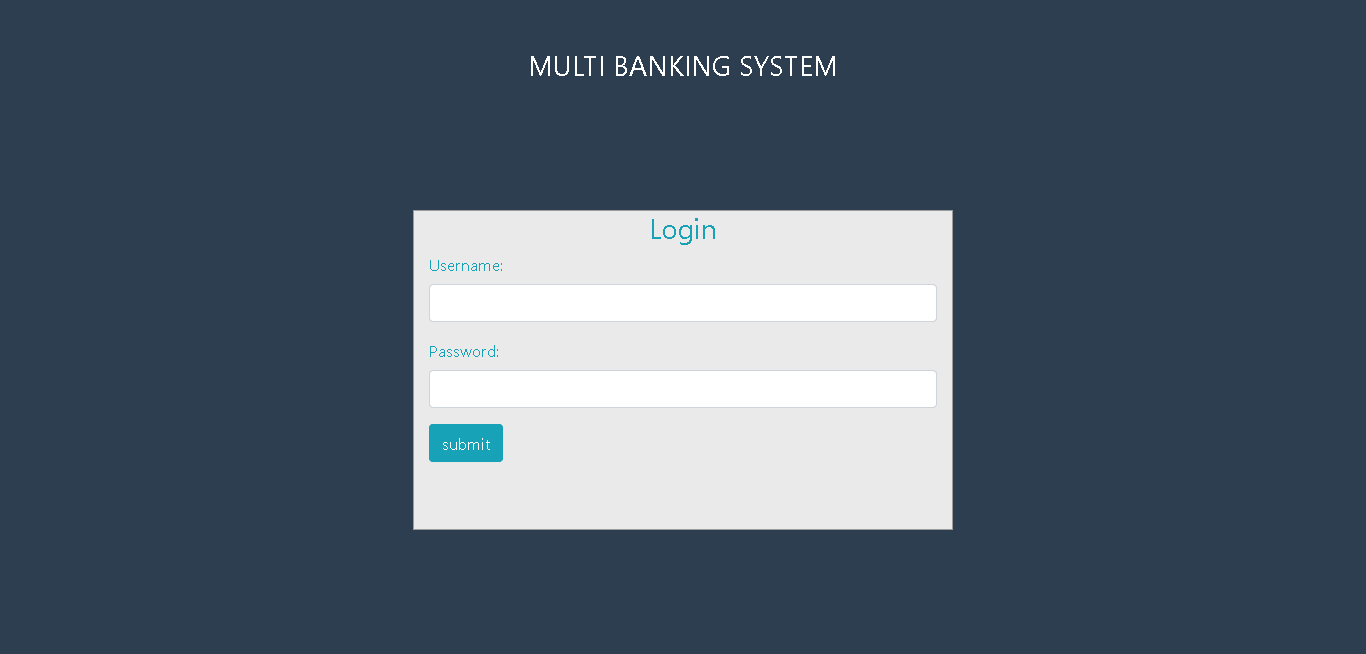
registry.addMapping("/\*\*");

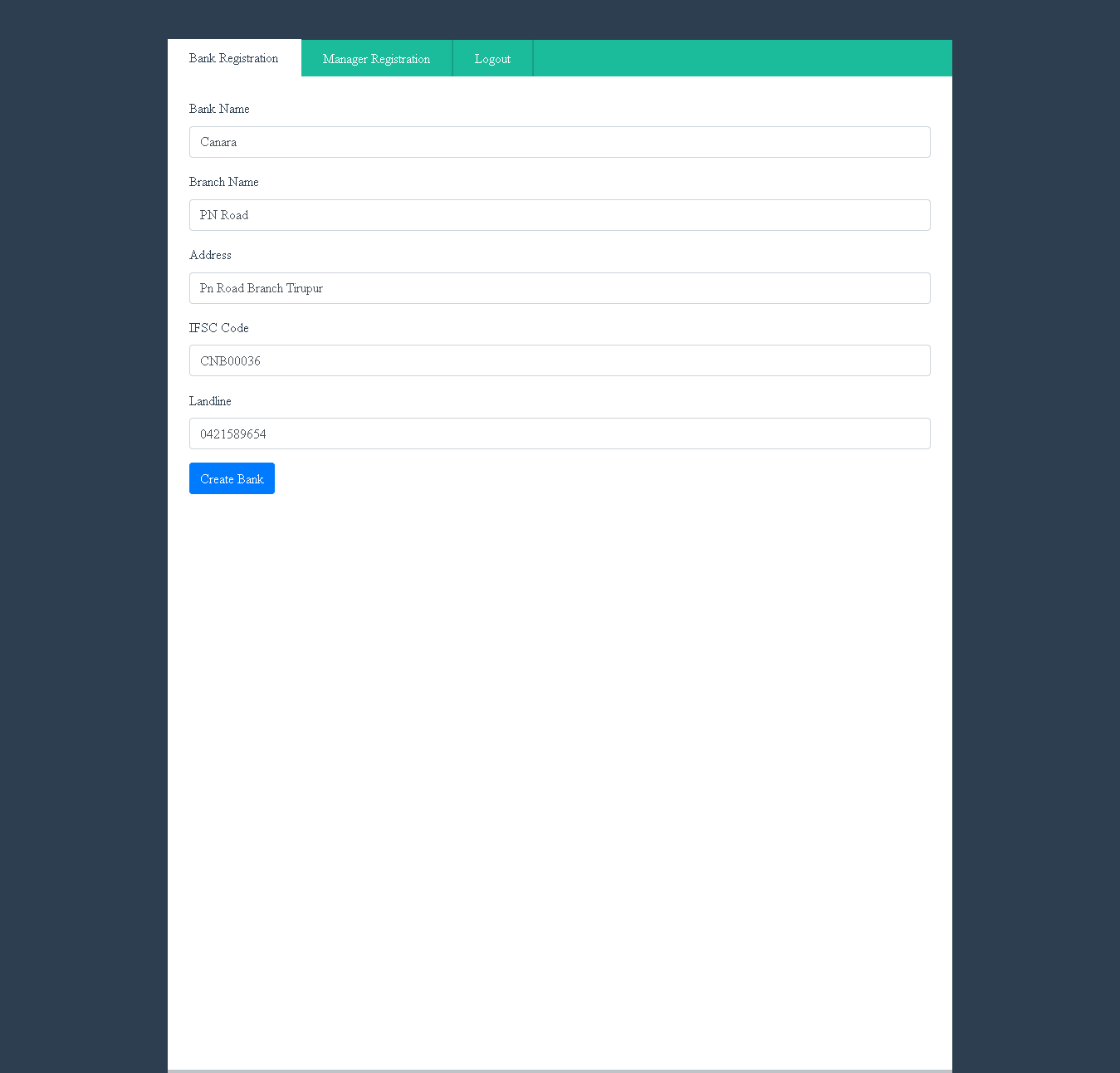
}

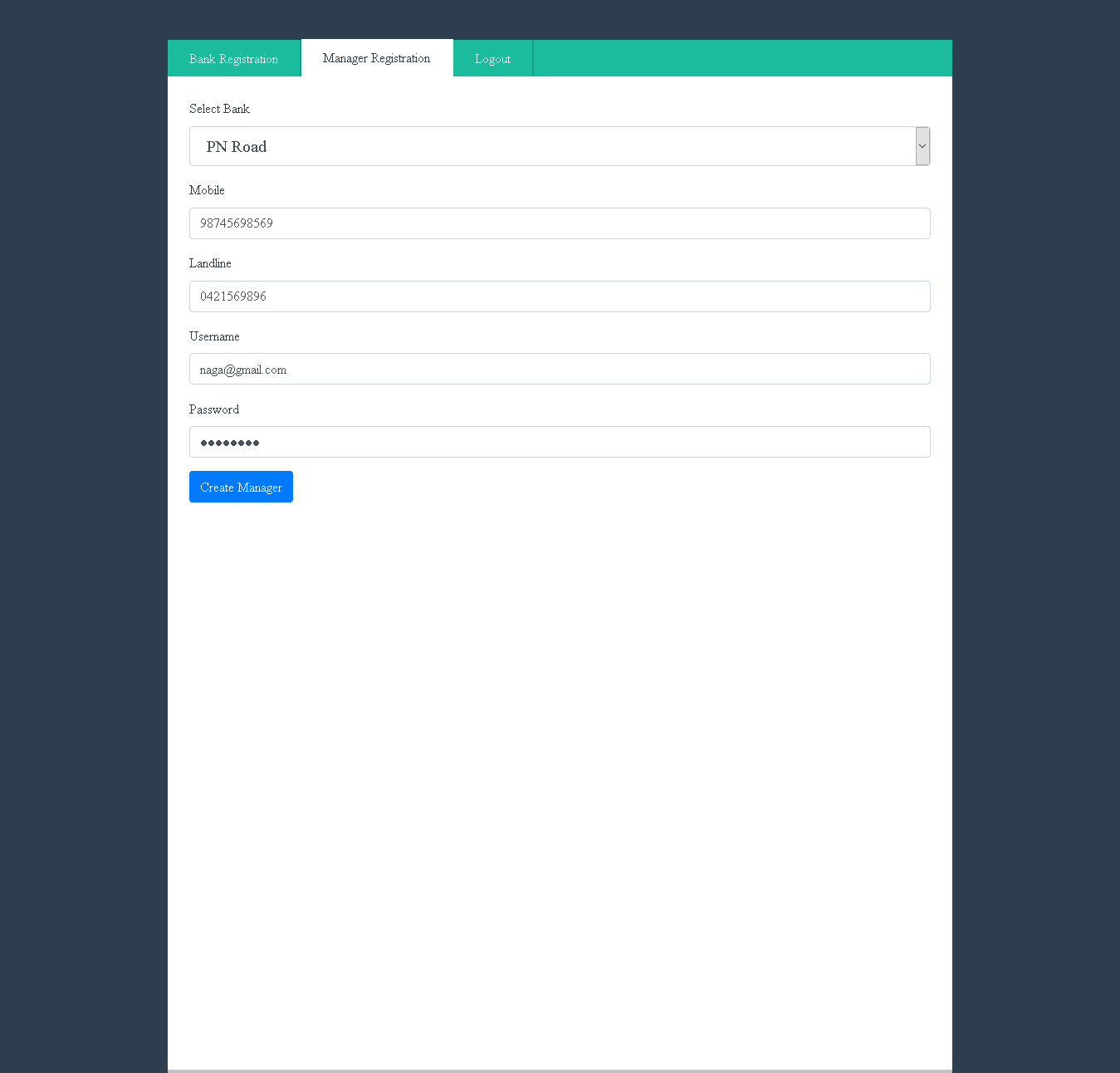
}

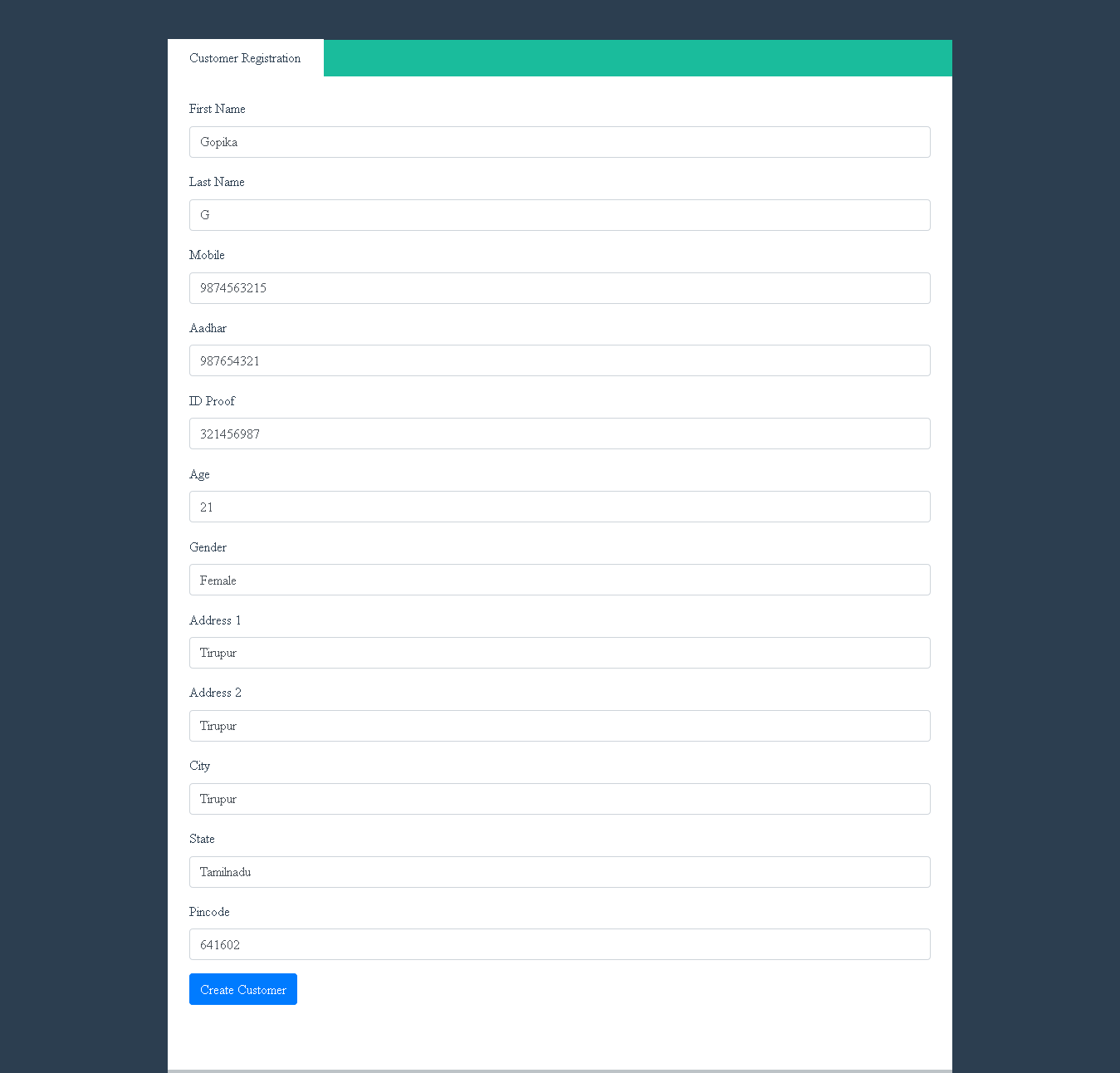
## APPENDIX II

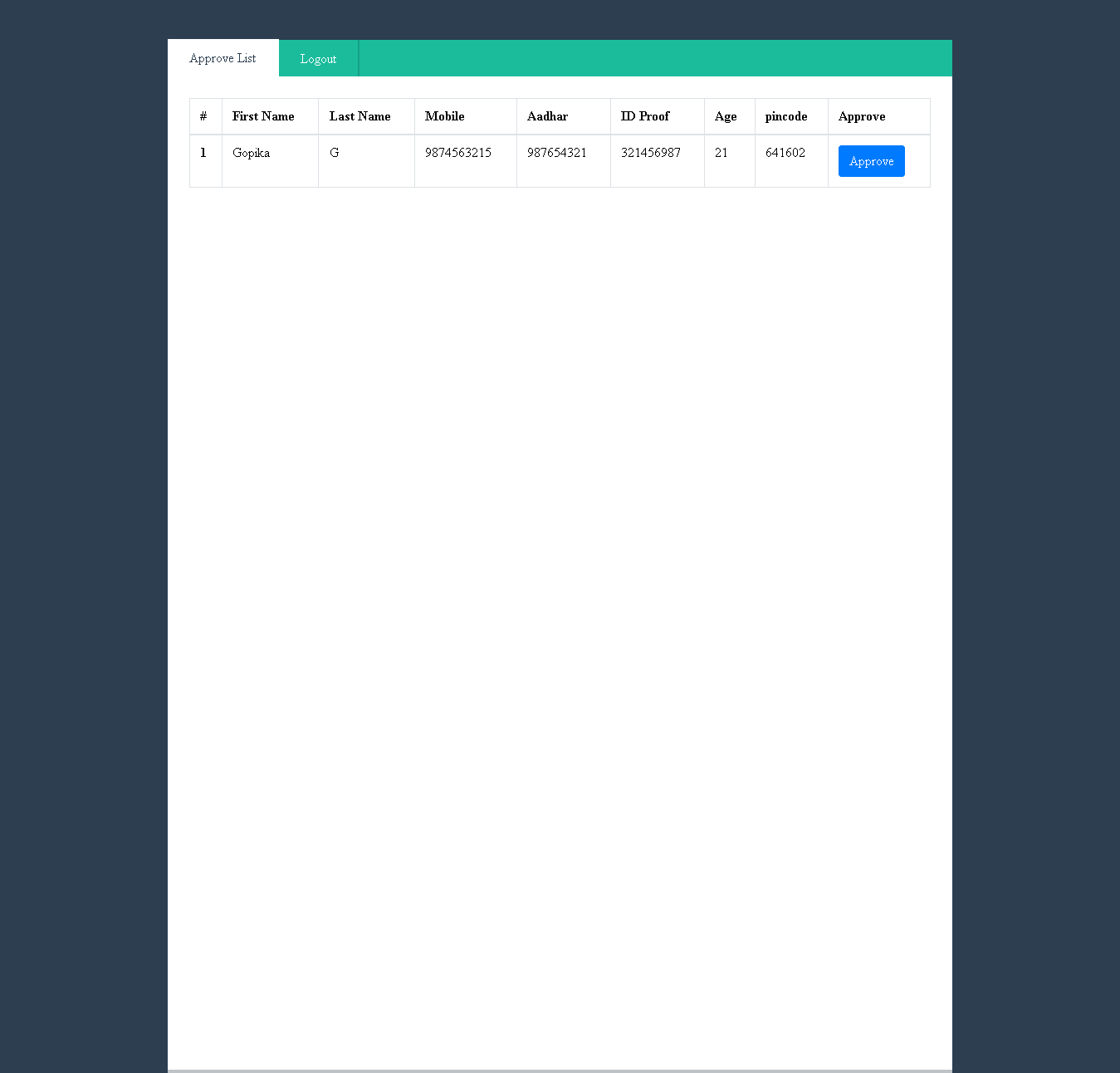
**SAMPLE SCREENSHOTS**

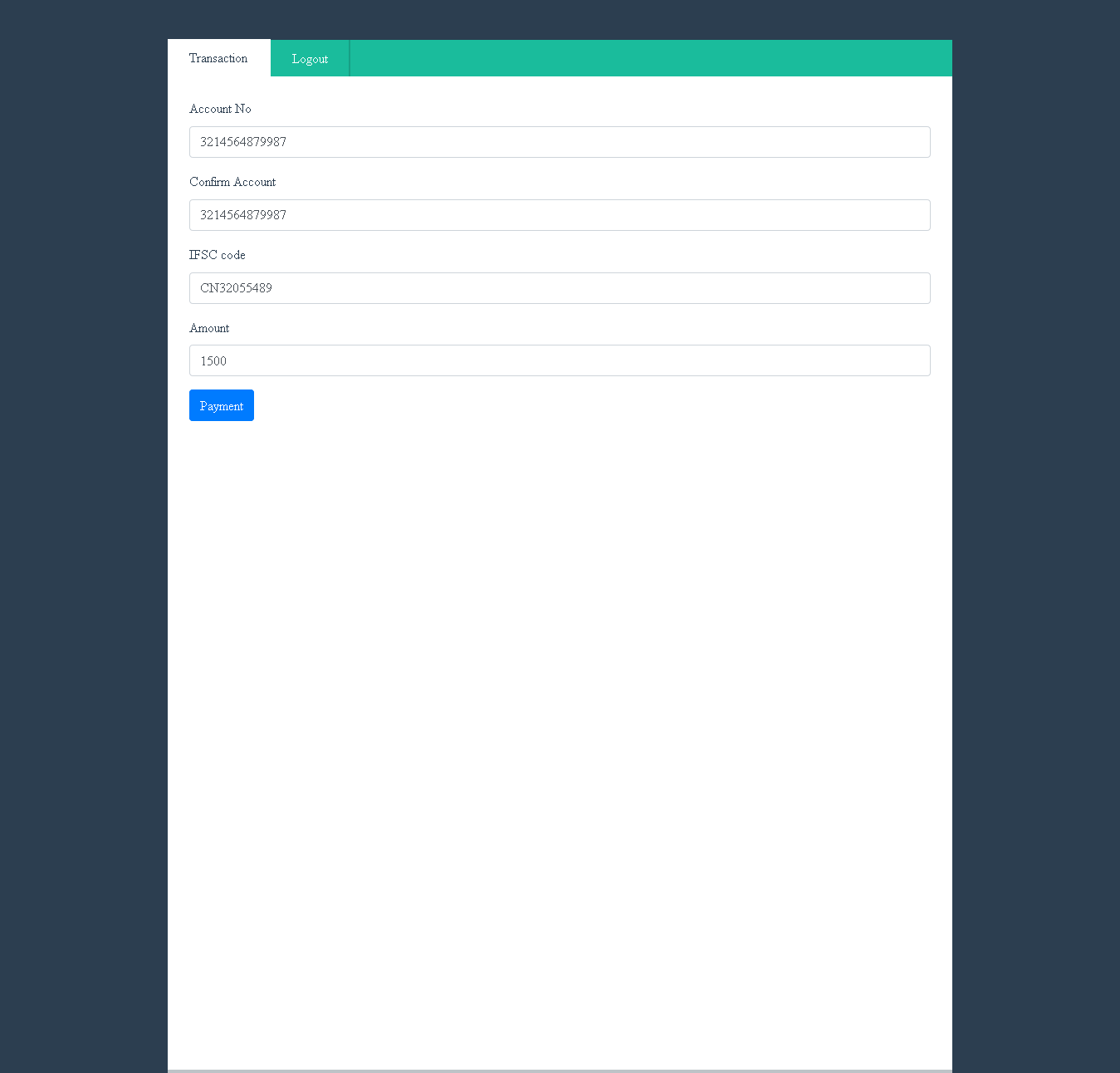












**REFERENCES**

**BOOK REFERENCES**

1. Jeff Forcier, Paul Bissex, Wesley J. Chun, (2018) “Python Web development with Django”(Developer’s Library)”, Wesley Professional.
2. Roger S. Pressman, (2017) “Software Engineering”, Tata McGraw-Hill –Europe.

## WEB REFERENCES

1. https://[www.javaprogramming.net](http://www.javaprogramming.net)
2. https://[www.w3schools.com](http://www.w3schools.com/)
3. https://[www.tutorialspoint.com](http://www.tutorialspoint.com/)