ONLINE BANKING SYSTEM

Submitted by

Project Team Members:

Avneesh.X

Arun.P

Sneha.R

Vishnu

Sahana.G.S

Nayana.R

Bharath.T

ABSTRACT

This project is aimed at developing an Online Banking for customer. The system is an online application that can be accessed throughout the organization and outside as well with proper login provided.

The project has been planned to be having the view of distributed architecture, with centralized storage of the database. The application for the storage of the data has been planned. Using the constructs of Oracle 10g and all the user interfaces have been designed using the SPRING. The database connectivity is planned using the "Database" methodology. The standards of security and data protective mechanism have been given a big choice for proper usage. The application takes care of different modules and their associated reports, which are produced as per the applicable strategies and standards that are put forwarded by the administrative staff.

The entire project has been developed keeping in view of the distributed client server computing technology, in mind. The specification has been normalized up to 3NF to eliminate all the anomalies that may arise due to the database transaction that are executed by the general users and the organizational administration. The user interfaces are browser specific to give distributed accessibility for the overall system. The internal database has been selected as Oracle 10g.The basic constructs of

table spaces, clusters and indexes have been exploited to provide higher consistency and reliability for the data storage. The Oracle 10g was a choice as it provides the constructs of high-level reliability and security. The total front end was dominated using the HTML 5. At all proper levels high care was taken to check that the system manages the data consistency with proper business rules or validations. The database connectivity was planned using the latest “Database connection” technology provided by Oracle. The authentication and authorization was crosschecked at all the relevant stages. The user level accessibility has been restricted into two zones namely.

Table of Contents

CHAPTER- 1

Introduction

1.1 Overview:

Internet Banking is all about knowing our customer need and provide them with the right service at the right time through right channel 24*7 day a week.

Being “electronic”, it not only provides its customers with faster and better facilities, it even reduces the manual overhead of accounts maintenance.

1.2 Background:

Bank is one of the top banking firms that accepts deposits from the public for the purpose of lending loans to the public. It also invests an amount in securities. Recently, the business analysts notice drop in the number of customers of the bank. They found out that online banking systems of banks like AXIS and American Express are gaining more profits by eliminating middlemen from the equation. As a result, the team decided to hire a Full Stack developer who can develop an online banking web application with a rich and user-friendly interface. You are hired as one of the Full Stack Java developers and have been asked to develop the web application. The management team has provided you with the requirements and their business model so that you can easily arrange different components of the application.

1.3 Purpose:

The Online Banking suite provides a global accounting foundation that provides the all private banks with electronic banking facilities. It allows client of private banks to carry out their day to day banking transactions.

1.4 Scope:

The Online Banking project is widely applicable with private banks. It can even be used in industries for their personal transactions (working).

1.5 Problem Definition:

The banking industry is witnessing a revolution in products, process, markets and regulations. And It's a revolution that is not about to stop or even slow down. Since the only option is to adapt and evolve, it is essential that system have the flexibility to quickly adjust the need of today's financial market. It's a tough challenge. Because today's fast-moving marketplace is also extremely competitive. Moreover, the need to retain existing customers and attract new ones often conflicts with the need to reduce costs and improve the efficiency. But whatever the challenges facing in retail banking operation, Online Banking System - OBS can help to meet and overcome them.

1.6 Functional components of the project:

Following are the functional needs of the software :

1. Customer must have a valid user ID and password to login to the system.

2. After the valid user logs in, the system shows the present balance in that particular account number.

3. Customer can perform transactions like deposit and withdrawal from his account.

4. Proper help to be provided as and when requested by the customer.

1.7 More functionality can be added to “enhance the project”:

1. By adding new modules of different accounts like saving A/C, current A/C etc. to facilitate new customers/users.

2. By the use of electronic media, “Digital Signature” on the card can be provided with the customer to make it secure and efficient.

1.8 Modules of Online Banking System:

User Accounts Module: used for managing the Account details.

Internet Banking Module: used for managing the details of Internet Banking.

User Module: used for managing the User details.

Account Module: used for managing the Transaction information.

Login Module: used for managing the login details.

Admin Module: used for managing the details of users in the system.

CHAPTER- 2
System Analysis

2.1 PRESENT SYSTEM

The developed system is an innovation in the area of private banking. In the existing system the no. of staff required for completing the work is more, while the new system requires lesser staffs generally.

The data entry process requires the data on the paper, which is then feed into the application by the operator while doing so; the data entry operator has to look into the paper again &again and thus the chances of in accuracies in the typed contents increases. Also the process includes higher transportation cost, increased handling cost, more time delays, low accuracy, more usage of resources like registers, books, papers, etc.

2.2 PROPOSED SYSTEM

"Why an Automated Private Banking System?"

- Almost 60% of today's information is still paper based.
- 30% of all office time is spent finding documents.
- The average time to manage a single document is 12 minutes,

9 minutes to re-file and 3 minutes to process.

Hence the requirement is to develop a system that minimizes all these overheads included while giving the maximum output for the organization.

The basis for the project is to develop a fully automated banking system that includes depositing of amount,

Withdrawal of amount and exporting the outcome back to the client while considering all the tools and facilities than a client may need for efficient and effective output.

2.3 Benefits of the system

- Quick, authenticated access to accounts via the desktop.
- Easily scalable to grow with changing system requirement.
- Enterprise wide access to information.
- Improved information security, restricting unauthorized access.
- Minimize Storage Space

In manual system, much storage space for data files is required so to overcome this problem, on automated well managed

database is developed for saving storage space. This s/w saves space and stores information efficiently. It ends the burden of having large manual filing storage system.

2.4 Banking System can be used extensively

- Withdrawal of amount by the client.
- Deposition of amount by the client.
- Balance enquiry.

- Cheque book request from client.

CHAPTER- 3

Software Requirement & Specification

<u>Software Required:</u>

The project is implemented using

Front end: Angular

Middle end: Spring Boot

Back end: My SQL, Hibernate

Server: Tomcat

Testing : Postman Testing Application

Angular:

Angular is a platform and framework for building single-page client applications using HTML and Typescript. Angular is written in Typescript. It implements core and optional functionality as a set of TypeScript libraries that you import into your applications.

Spring Boot:

Spring Boot is an open source, microservice-based Java web framework. The Spring Boot framework creates a fully production-ready environment that is completely configurable using its prebuilt code within its codebase.

My SQL, Hibernate:

MySQL is an open source relational database management system. For WordPress sites, that means it helps you store all your blog posts, users, plugin information, etc. It stores that information in separate "tables" and connects it with "keys", which is why it's relational. Hibernate takes care of mapping Java classes to database tables using XML files and without writing any line of code. Provides simple APIs for storing and retrieving Java objects directly to and from the database. If there is change in the database or in any table, then you need to change the XML file properties only.

Hardware Required:

- Operating System - Windows XP, Windows 7, Windows 10 or above
- Processor - Intel Dual Core or higher version RAM - Minimum 1GB.
- Hard Disk - Minimum 40GB or above

Technologies and Requriments

Tool:

SpringToolSuite4

Front End:

Angular: HTML, TypeScript, CSS, Bootstrap

Programming Language:

Spring Boot

Back End:

MySql.

CHAPTER- 4

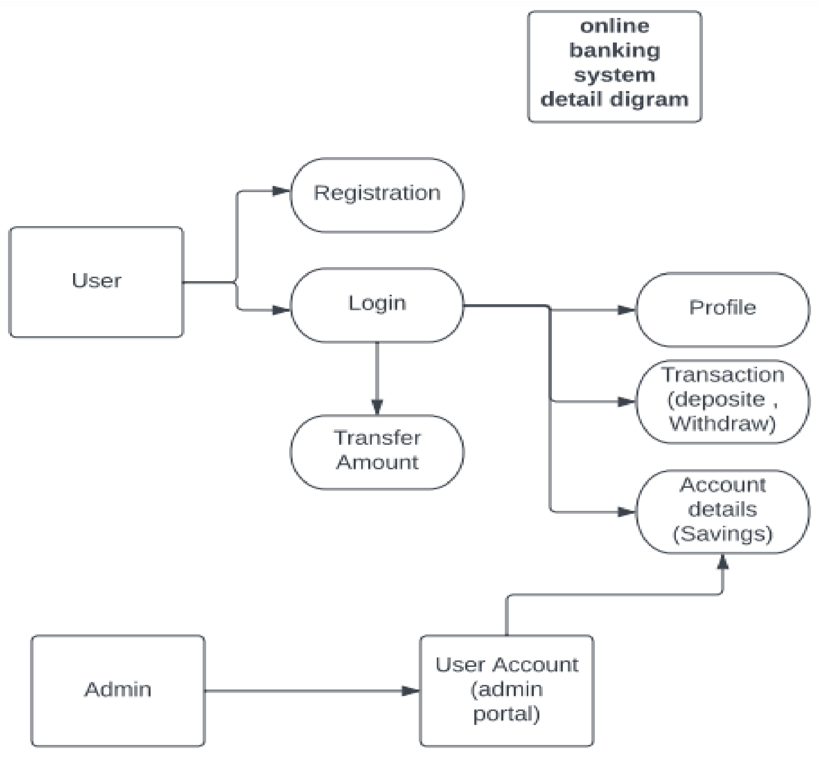
System Design

4.1 INTRODUCTION:

Software design sits at the technical kernel of the software engineering process and is applied regardless of the development paradigm and area of application. Design is the first step in the development phase for any engineered product or system. The designer's goal is to produce a model or representation of an entity that will later be built. Beginning, once system requirement have been specified and analyzed, system design is the first of the three technical activities -design, code and test that is required to build and verify software.

The importance can be stated with a single word "Quality". Design is the place where quality is fostered in software development. Design provides us with representations of software that can assess for quality. Design is the only way that we can accurately translate a customer's view into a finished software product or system. Software design serves as a foundation for all the software engineering steps that follow. Without a strong design we risk building an unstable system – one that will be difficult to test, one whose quality cannot be assessed until the last stage. During design, progressive refinement of data structure, program structure, and procedural details are developed reviewed and documented. System design can be viewed from either technical or project management perspective. From the technical point of view, design is comprised of four activities – architectural design, data structure design, interface design and procedural design.

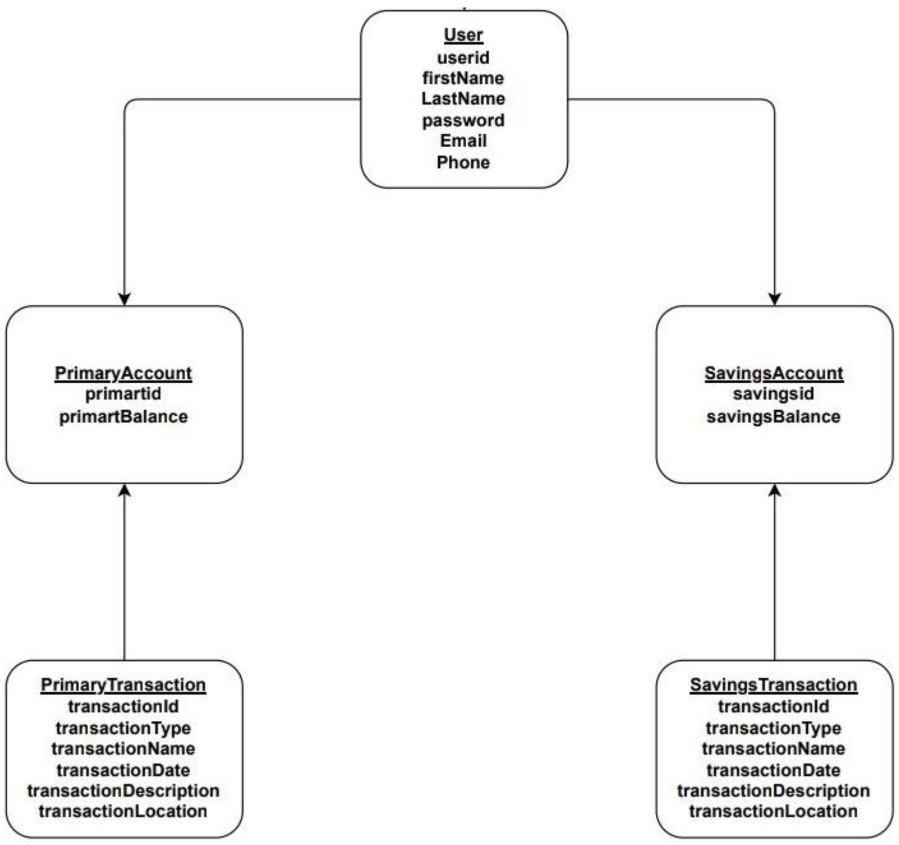
Flow of the Diagram:


online banking system detail digram
User
Registration
Login
Transfer Amount
Profile
Transaction (deposite , Withdraw)
Account details (Savings)
Admin
User Account (admin portal)

4.2 E-R DIAGRAM OF USER:

The relation upon the system is structure through a conceptual ER-Diagram, which not only specifics the existential entities but also the standard relations through which the system exists and the cardinalities that are necessary for the system state to continue.

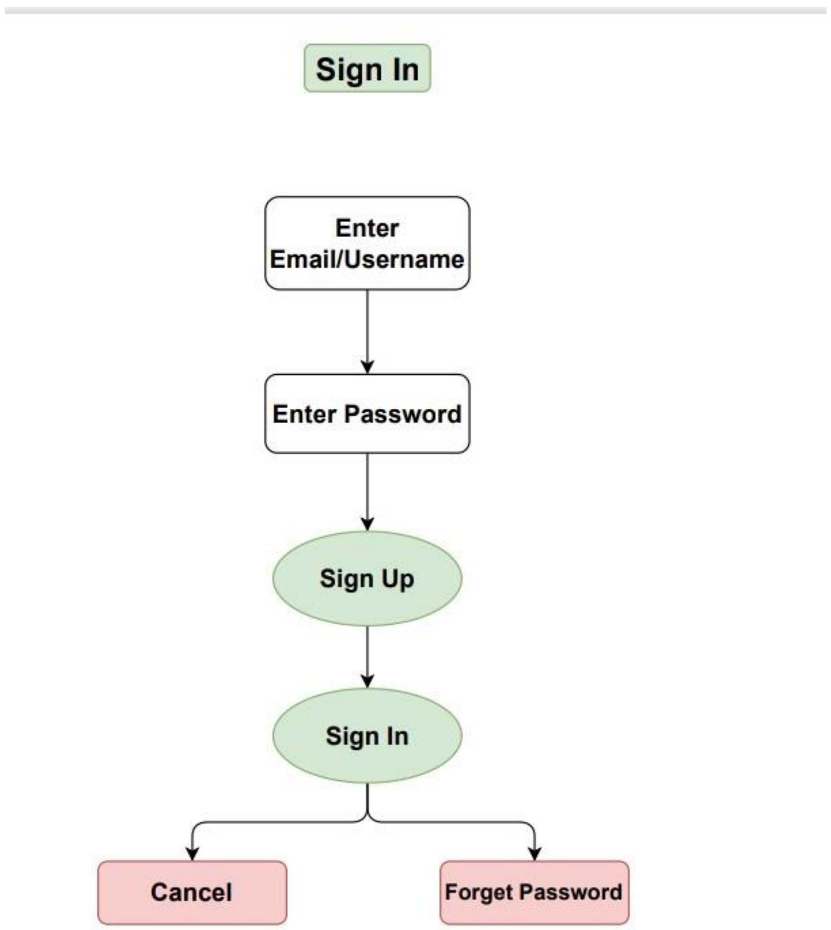
The entity Relationship Diagram (ERD) depicts the relationship between the data objects. The ERD is the notation that is used to conduct the date modeling activity the attributes of each data object noted is the ERD can be described resign a data object descriptions.



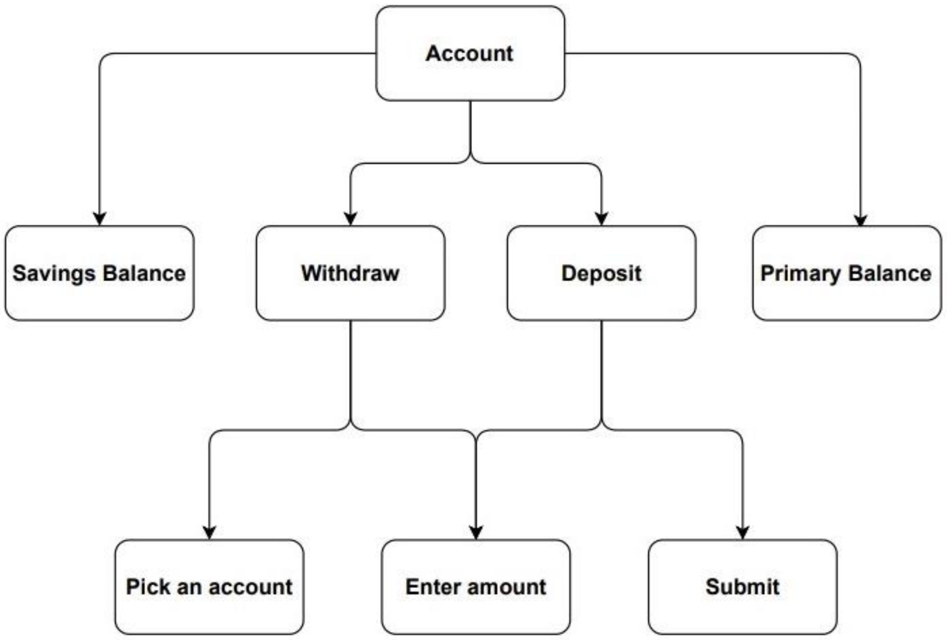
4.3 DATA FLOW DIAGRAMS:

A data flow diagram is graphical tool used to describe and analyze movement of data through a system. These are the central tool and the basis from which the other components are developed. The transformation of data from input to output, through processed, may be described logically and independently of physical components associated with the system. These are known as the logical data flow diagrams. The physical data flow diagrams show the actual implements and movement of data between people, departments and workstations. A full description of a system actually consists of a set of data flow diagrams Each component in a DFD is labeled with a descriptive nameThe development of DFD’S is done in several levels. Each process in lower level diagrams can be broken down into a more detailed DFD in the next level. The lop-level diagram is often called context diagram. It consists a single process bit, which plays vital role in studying the current system. The process in the context level diagram is exploded into other process at the first level DFD.

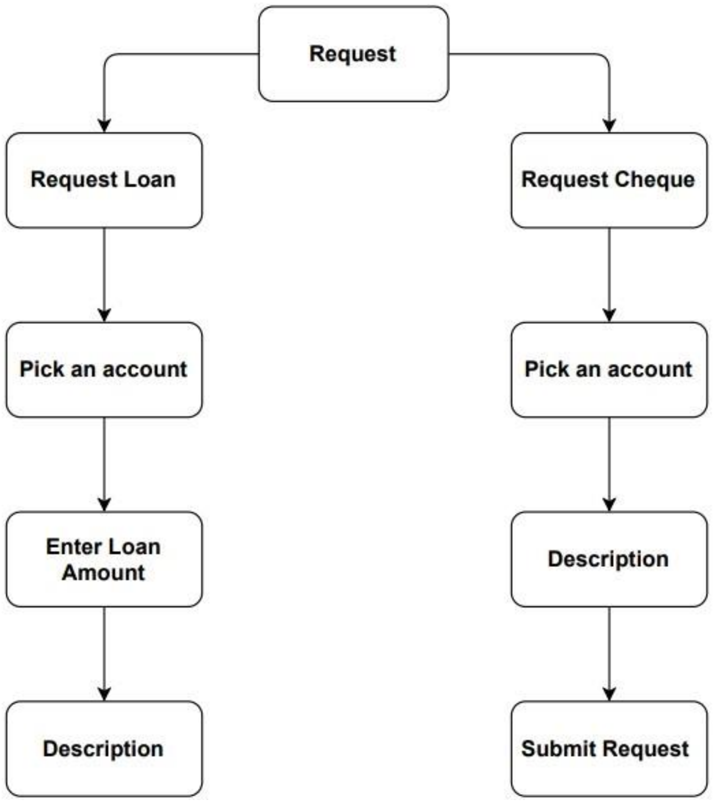
Login Module:



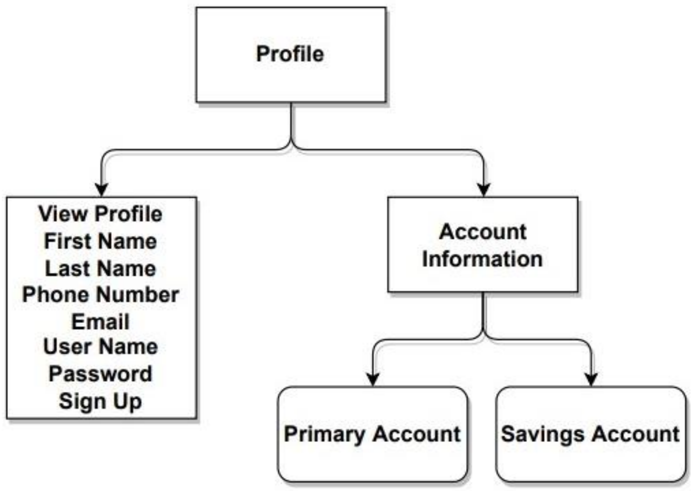
Account Module:



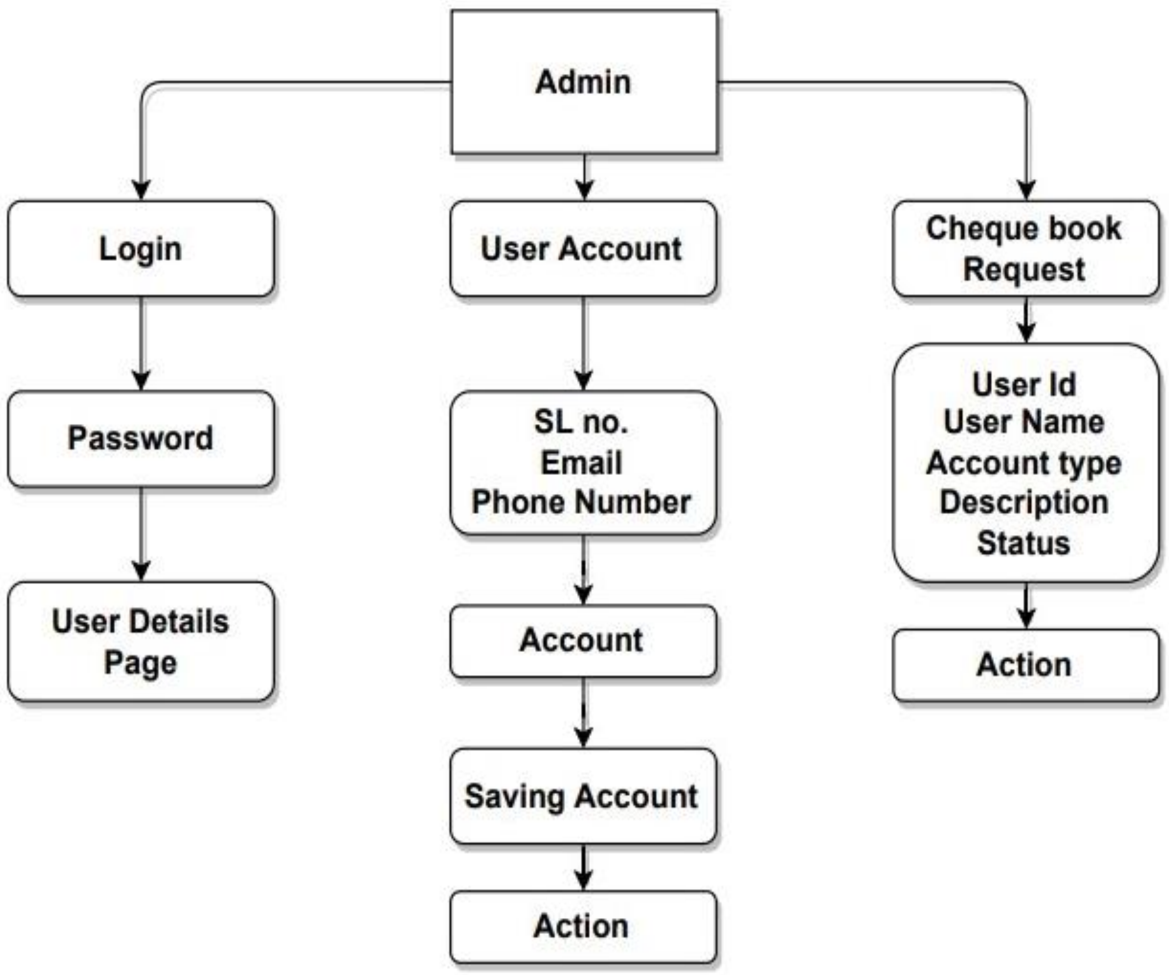
Request Module:



Profile Module:

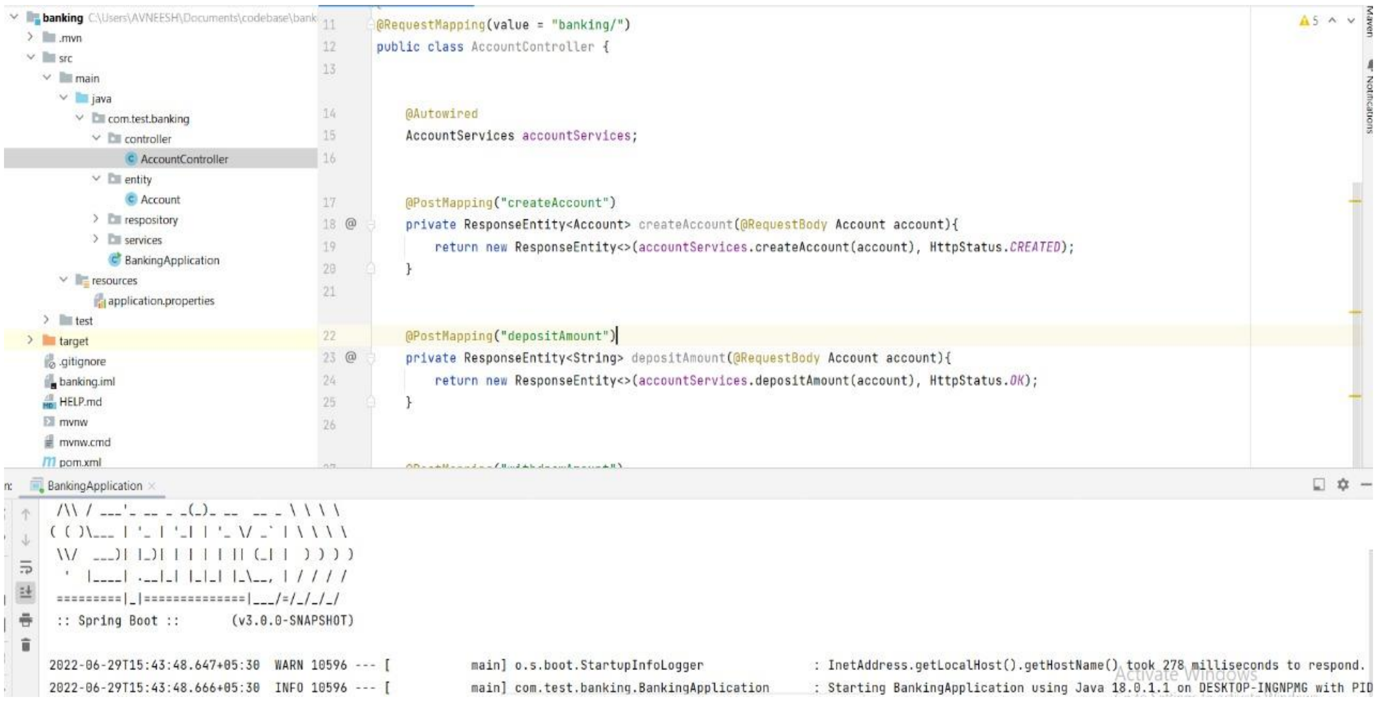


Admin Module:



CHAPTER- 5

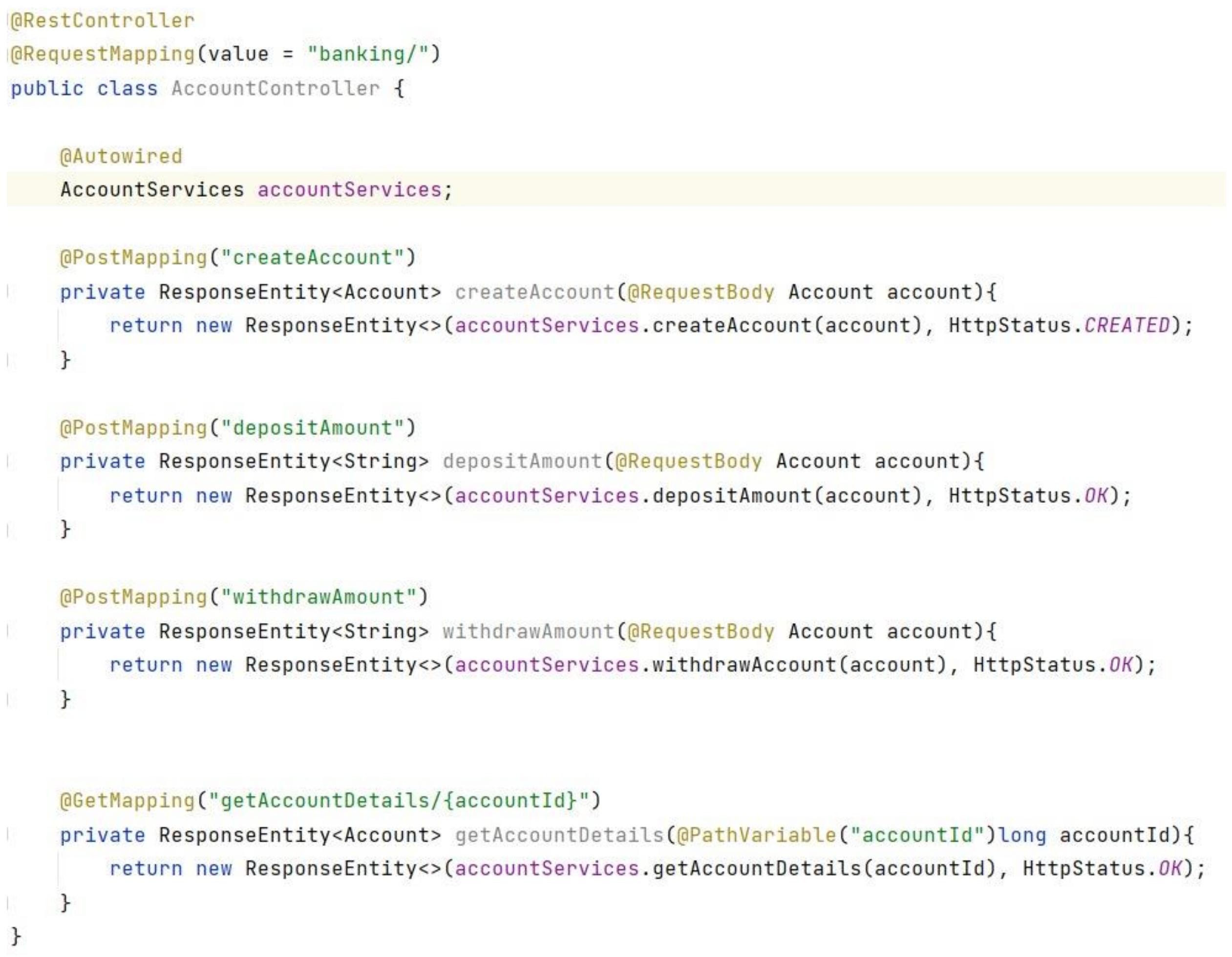
CODING



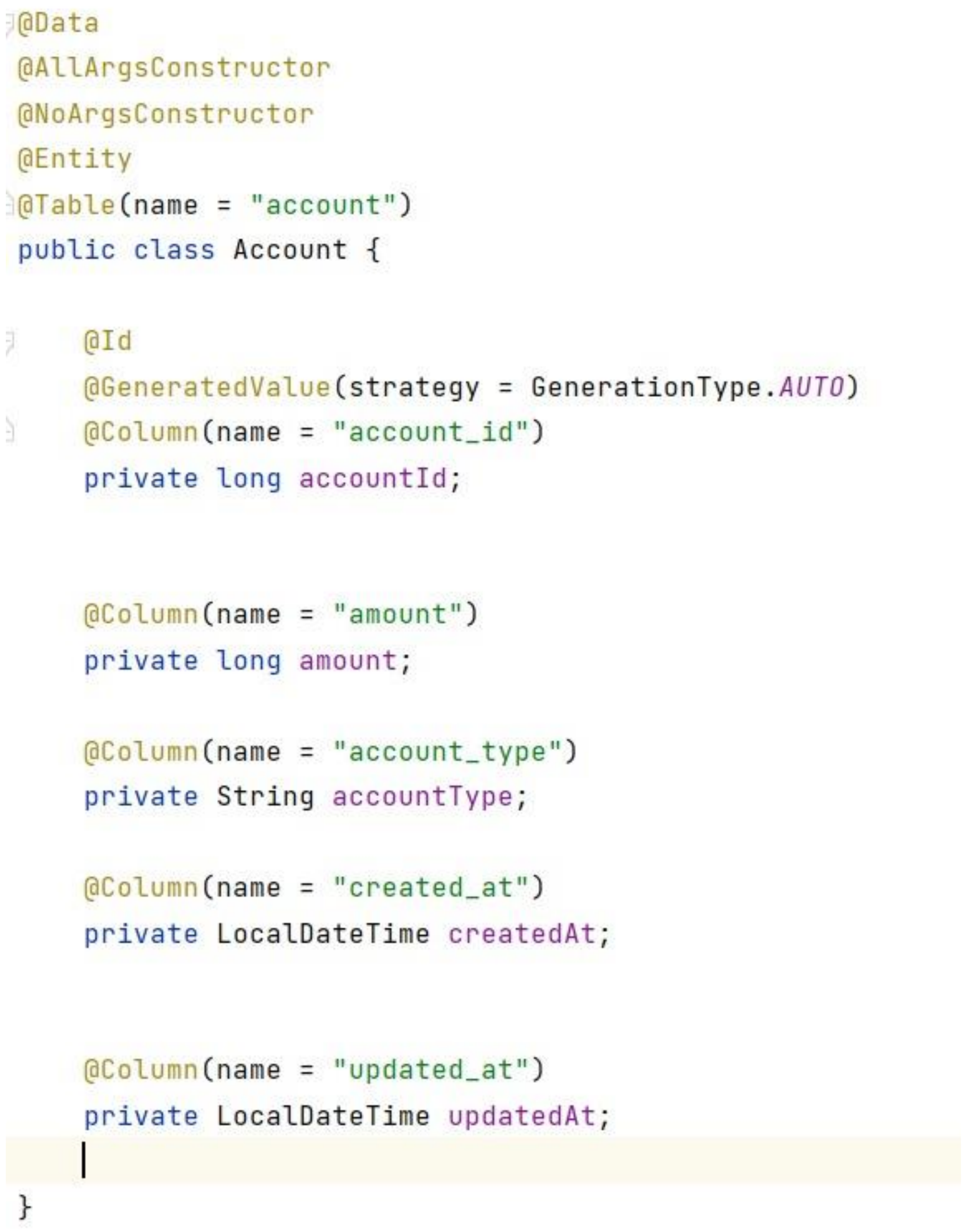
```
banking C:\Users\AVNEESH\Documents\codebase\bank
  .mvn
  src
    main
      java
        com.test.banking
          controller
            AccountController
          entity
            Account
          respository
          services
          BankingApplication
      resources
        application.properties
    test
  target
  .gitignore
  banking.iml
  HELP.md
  mvnw
  mvnw.cmd
  pom.xml
```

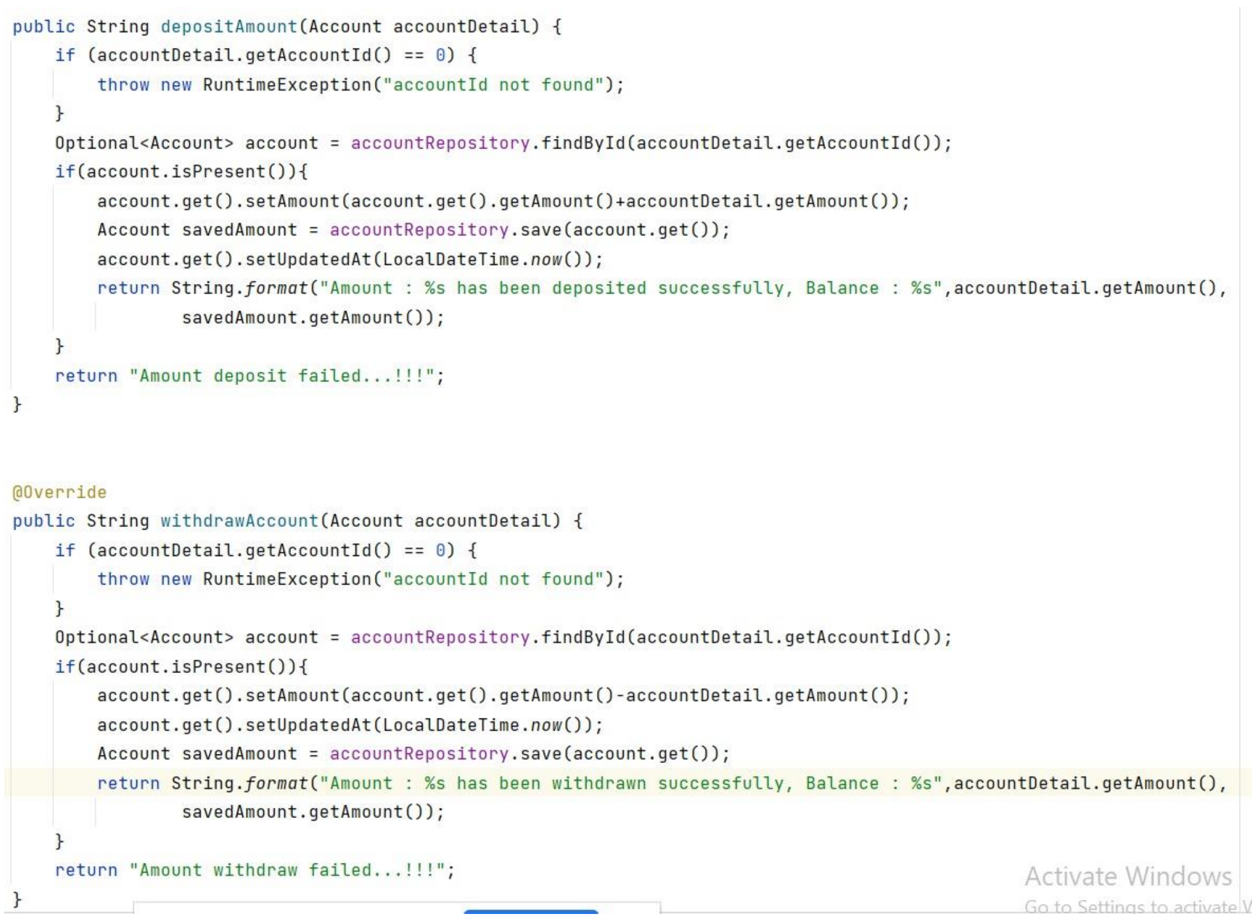
```java
@RequestMapping(value = "banking/")
public class AccountController {


    @Autowired
    AccountServices accountServices;


    @PostMapping("createAccount")
    private ResponseEntity<Account> createAccount(@RequestBody Account account){
        return new ResponseEntity<>(accountServices.createAccount(account), HttpStatus.CREATED);
    }


    @PostMapping("depositAmount")
    private ResponseEntity<String> depositAmount(@RequestBody Account account){
        return new ResponseEntity<>(accountServices.depositAmount(account), HttpStatus.OK);
    }
```

```
  /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
 ( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
  \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
   '  |____| .__|_| |_|_| |_\__, | / / / /
  =========|_|==============|___/=/_/_/_/
  :: Spring Boot ::       (v3.0.0-SNAPSHOT)

2022-06-29T15:43:48.647+05:30  WARN 10596 --- [           main] o.s.boot.StartupInfoLogger               : InetAddress.getLocalHost().getHostName() took 278 milliseconds to respond.
2022-06-29T15:43:48.666+05:30  INFO 10596 --- [           main] com.test.banking.BankingApplication      : Starting BankingApplication using Java 18.0.1.1 on DESKTOP-INGNPMG with PID
```

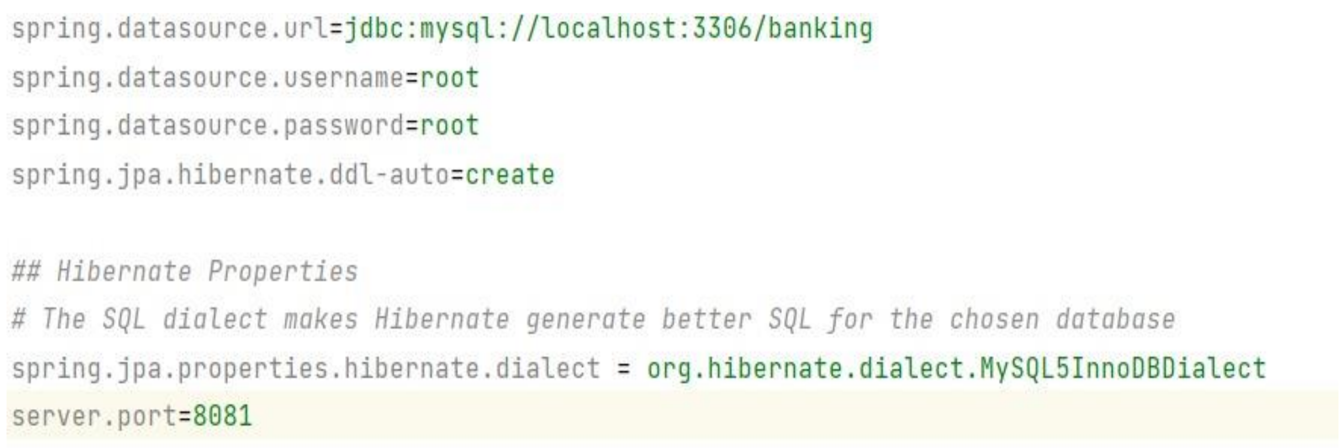


```java
@RestController
@RequestMapping(value = "banking/")
public class AccountController {

    @Autowired
    AccountServices accountServices;

    @PostMapping("createAccount")
    private ResponseEntity<Account> createAccount(@RequestBody Account account){
        return new ResponseEntity<>(accountServices.createAccount(account), HttpStatus.CREATED);
    }

    @PostMapping("depositAmount")
    private ResponseEntity<String> depositAmount(@RequestBody Account account){
        return new ResponseEntity<>(accountServices.depositAmount(account), HttpStatus.OK);
    }

    @PostMapping("withdrawAmount")
    private ResponseEntity<String> withdrawAmount(@RequestBody Account account){
        return new ResponseEntity<>(accountServices.withdrawAccount(account), HttpStatus.OK);
    }


    @GetMapping("getAccountDetails/{accountId}")
    private ResponseEntity<Account> getAccountDetails(@PathVariable("accountId")long accountId){
        return new ResponseEntity<>(accountServices.getAccountDetails(accountId), HttpStatus.OK);
    }
}
```

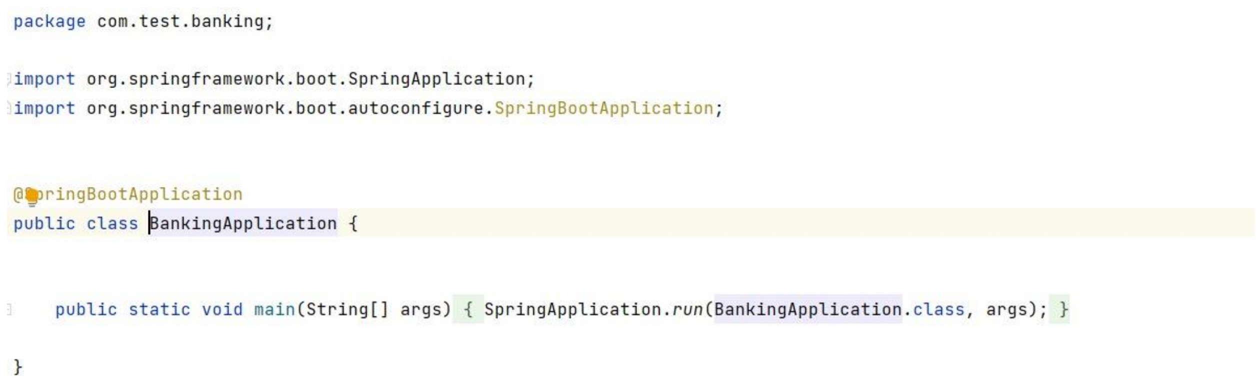

```java
@Data
@AllArgsConstructor
@NoArgsConstructor
@Entity
@Table(name = "account")
public class Account {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    @Column(name = "account_id")
    private long accountId;


    @Column(name = "amount")
    private long amount;

    @Column(name = "account_type")
    private String accountType;

    @Column(name = "created_at")
    private LocalDateTime createdAt;


    @Column(name = "updated_at")
    private LocalDateTime updatedAt;

}
```



```java
public String depositAmount(Account accountDetail) {
    if (accountDetail.getAccountId() == 0) {
        throw new RuntimeException("accountId not found");
    }
    Optional<Account> account = accountRepository.findById(accountDetail.getAccountId());
    if(account.isPresent()){
        account.get().setAmount(account.get().getAmount()+accountDetail.getAmount());
        Account savedAmount = accountRepository.save(account.get());
        account.get().setUpdatedAt(LocalDateTime.now());
        return String.format("Amount : %s has been deposited successfully, Balance : %s",accountDetail.getAmount(),
                savedAmount.getAmount());
    }
    return "Amount deposit failed...!!!";
}


@Override
public String withdrawAccount(Account accountDetail) {
    if (accountDetail.getAccountId() == 0) {
        throw new RuntimeException("accountId not found");
    }
    Optional<Account> account = accountRepository.findById(accountDetail.getAccountId());
    if(account.isPresent()){
        account.get().setAmount(account.get().getAmount()-accountDetail.getAmount());
        account.get().setUpdatedAt(LocalDateTime.now());
        Account savedAmount = accountRepository.save(account.get());
        return String.format("Amount : %s has been withdrawn successfully, Balance : %s",accountDetail.getAmount(),
                savedAmount.getAmount());
    }
    return "Amount withdraw failed...!!!";
}
```

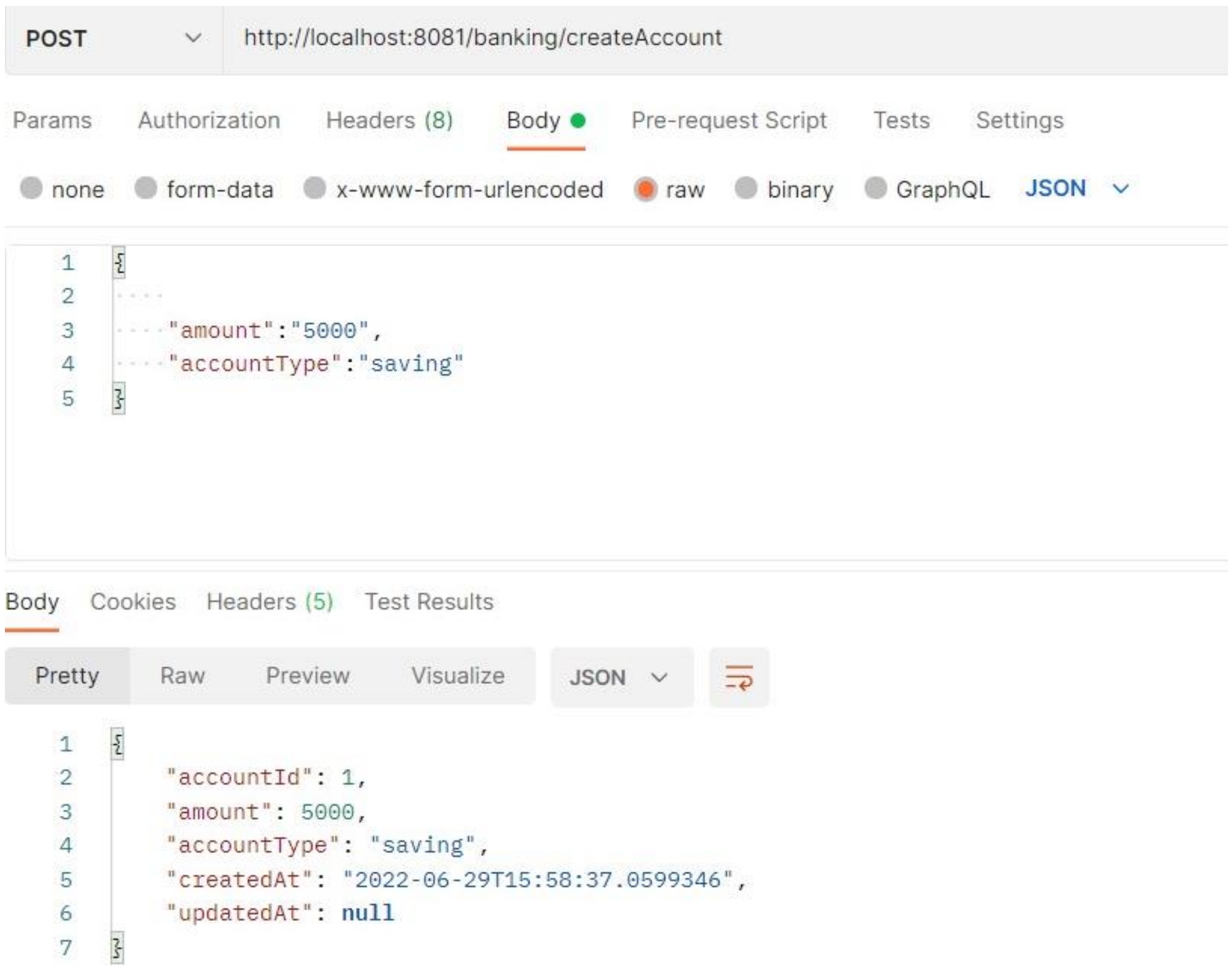


```
spring.datasource.url=jdbc:mysql://localhost:3306/banking
spring.datasource.username=root
spring.datasource.password=root
spring.jpa.hibernate.ddl-auto=create

## Hibernate Properties
# The SQL dialect makes Hibernate generate better SQL for the chosen database
spring.jpa.properties.hibernate.dialect = org.hibernate.dialect.MySQL5InnoDBDialect
server.port=8081
```



```java
package com.test.banking;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;


@SpringBootApplication
public class BankingApplication {


    public static void main(String[] args) { SpringApplication.run(BankingApplication.class, args); }

}
```

Postman Testing:

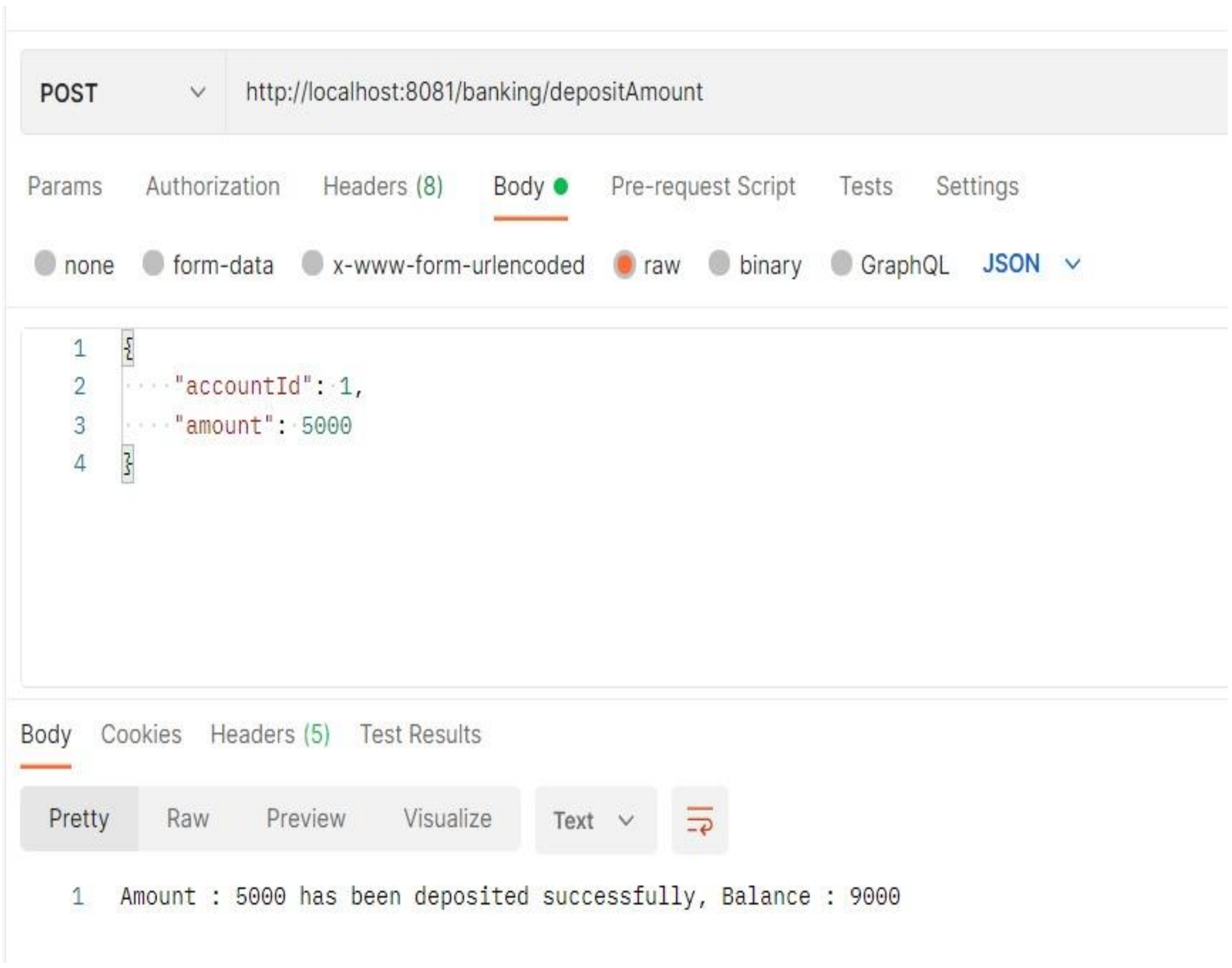


POST http://localhost:8081/banking/createAccount

```json
{

    "amount":"5000",
    "accountType":"saving"
}
```

Response Body:

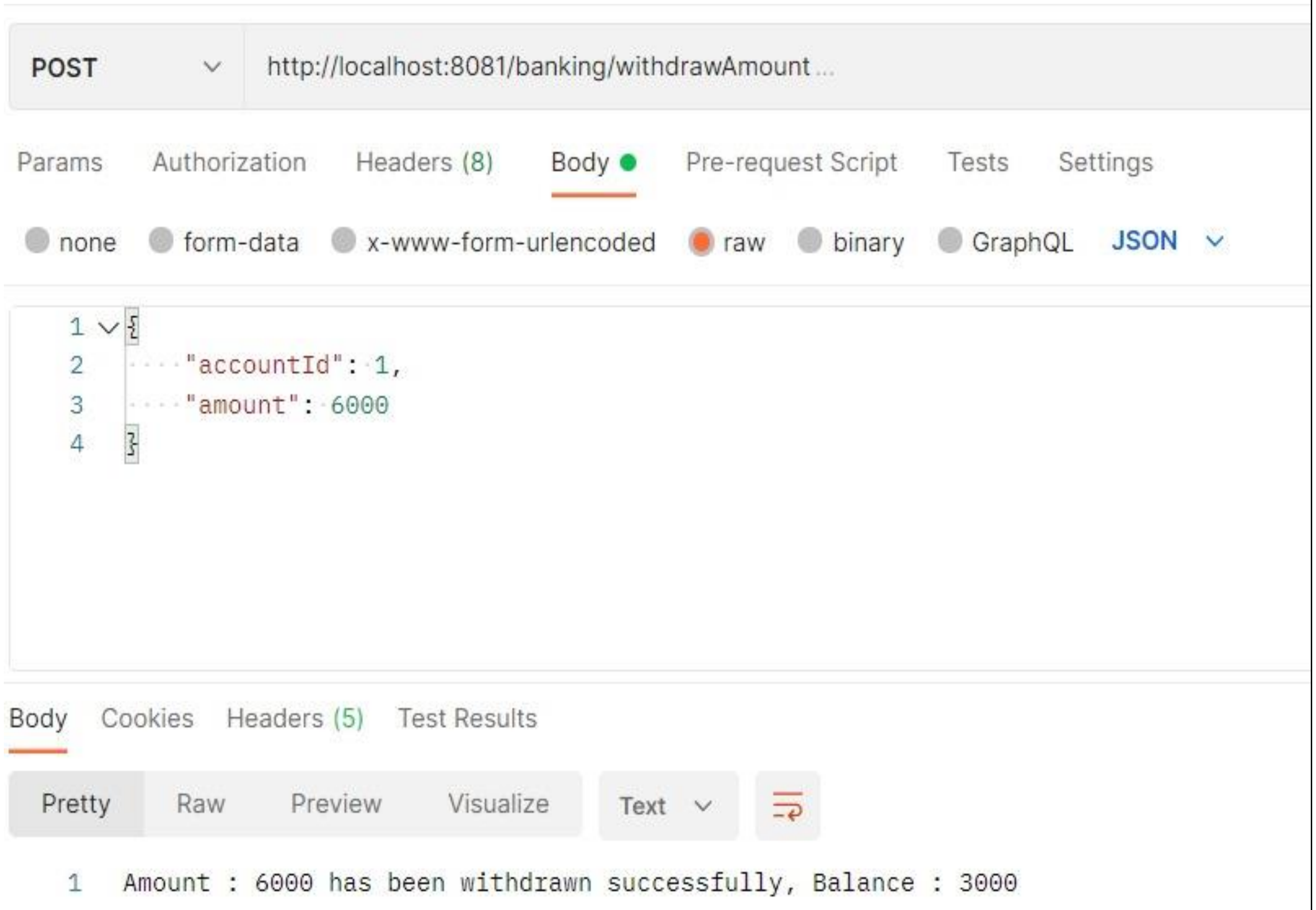
```json
{
    "accountId": 1,
    "amount": 5000,
    "accountType": "saving",
    "createdAt": "2022-06-29T15:58:37.0599346",
    "updatedAt": null
}
```



http://localhost:8081/banking/depositAmount

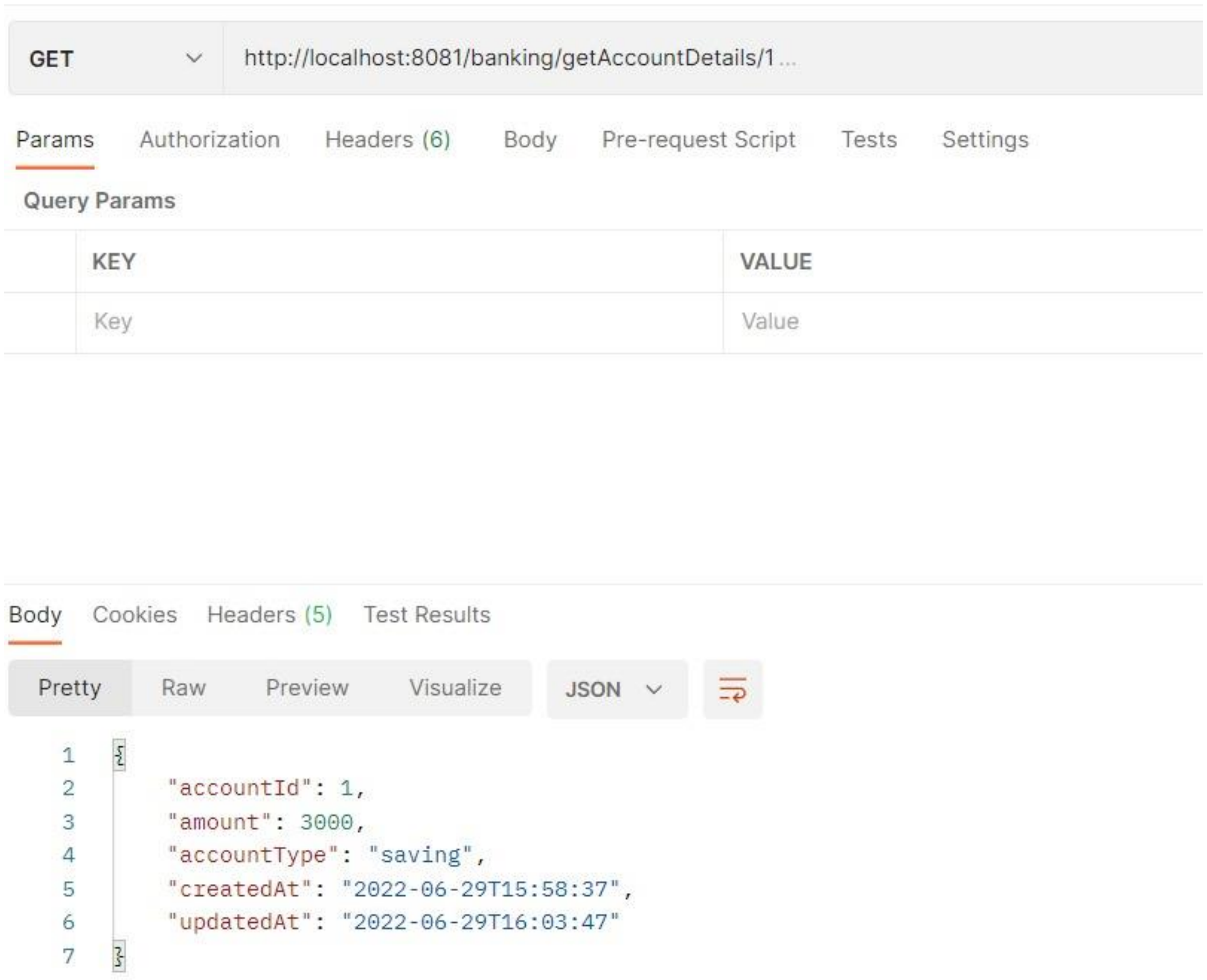
```
{
    "accountId": 1,
    "amount": 5000
}
```

```
Amount : 5000 has been deposited successfully, Balance : 9000
```


POST http://localhost:8081/banking/withdrawAmount ...

```
{
    "accountId": 1,
    "amount": 6000
}
```

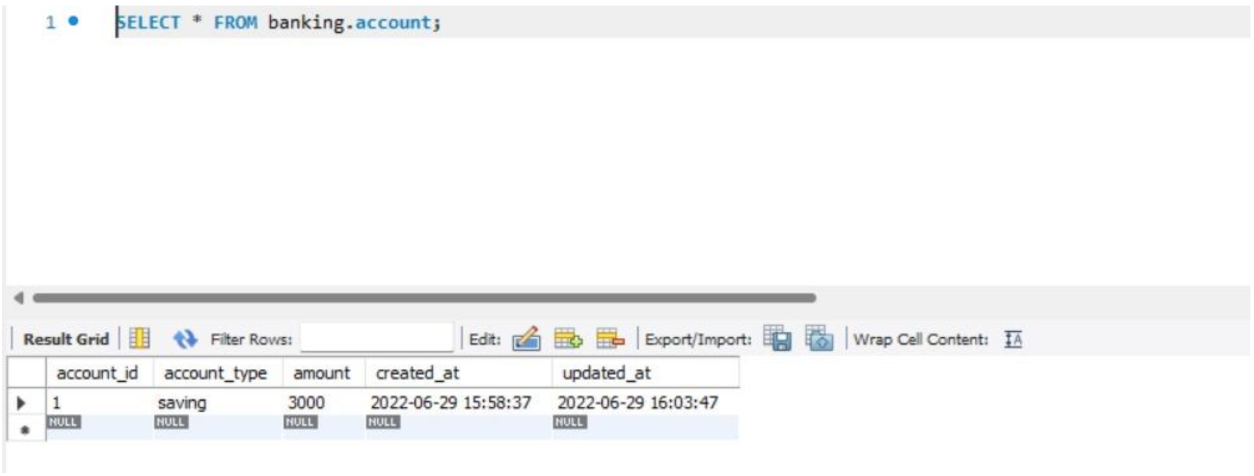
```
Amount : 6000 has been withdrawn successfully, Balance : 3000
```


GET http://localhost:8081/banking/getAccountDetails/1 ...

KEY	VALUE
Key	Value

```
{
    "accountId": 1,
    "amount": 3000,
    "accountType": "saving",
    "createdAt": "2022-06-29T15:58:37",
    "updatedAt": "2022-06-29T16:03:47"
}
```

MySql Database:


```
SELECT * FROM banking.account;
```

account_id	account_type	amount	created_at	updated_at
1	saving	3000	2022-06-29 15:58:37	2022-06-29 16:03:47
NULL	NULL	NULL	NULL	NULL

CHAPTER- 6

OUTPUT SCREENSHOTSHOTS

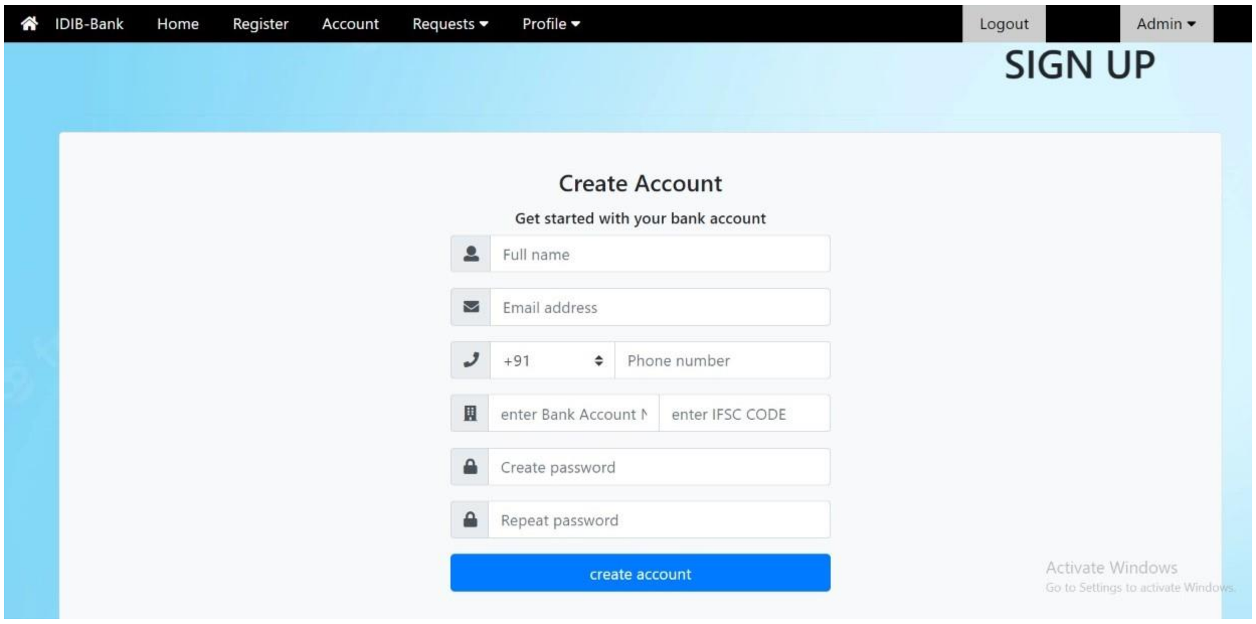


Fig 6.1: Register page

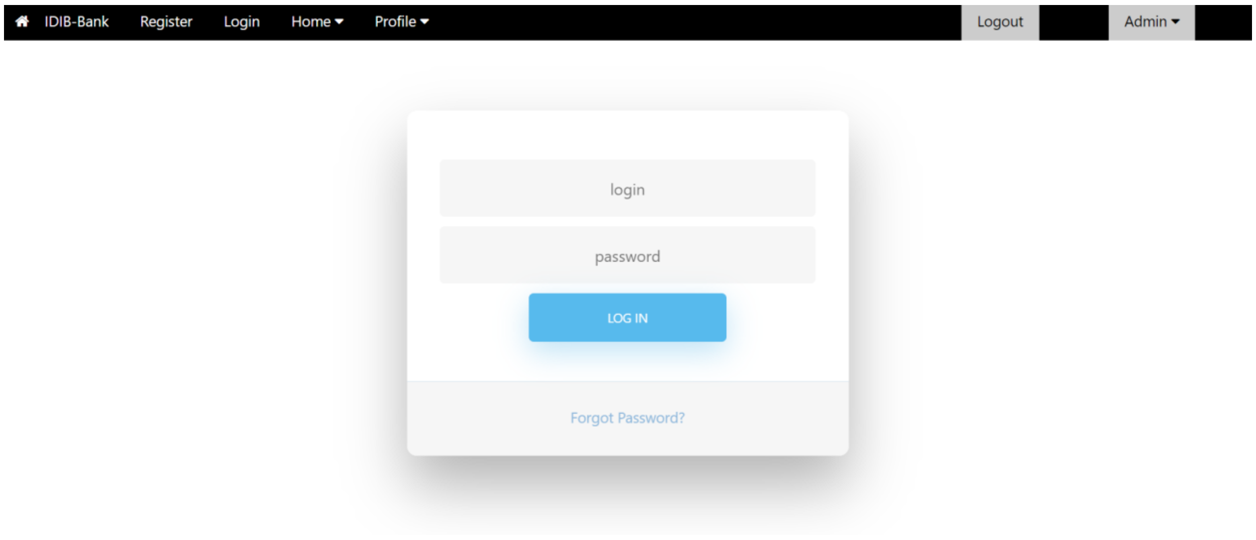


Fig 6.2: Login page

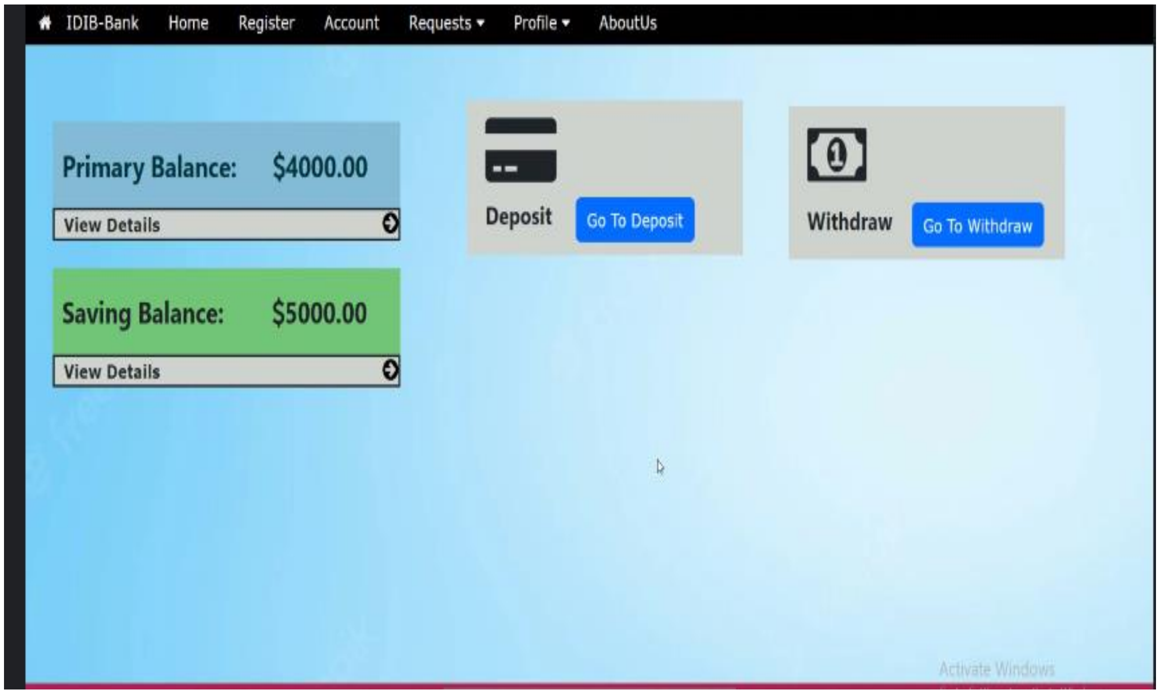
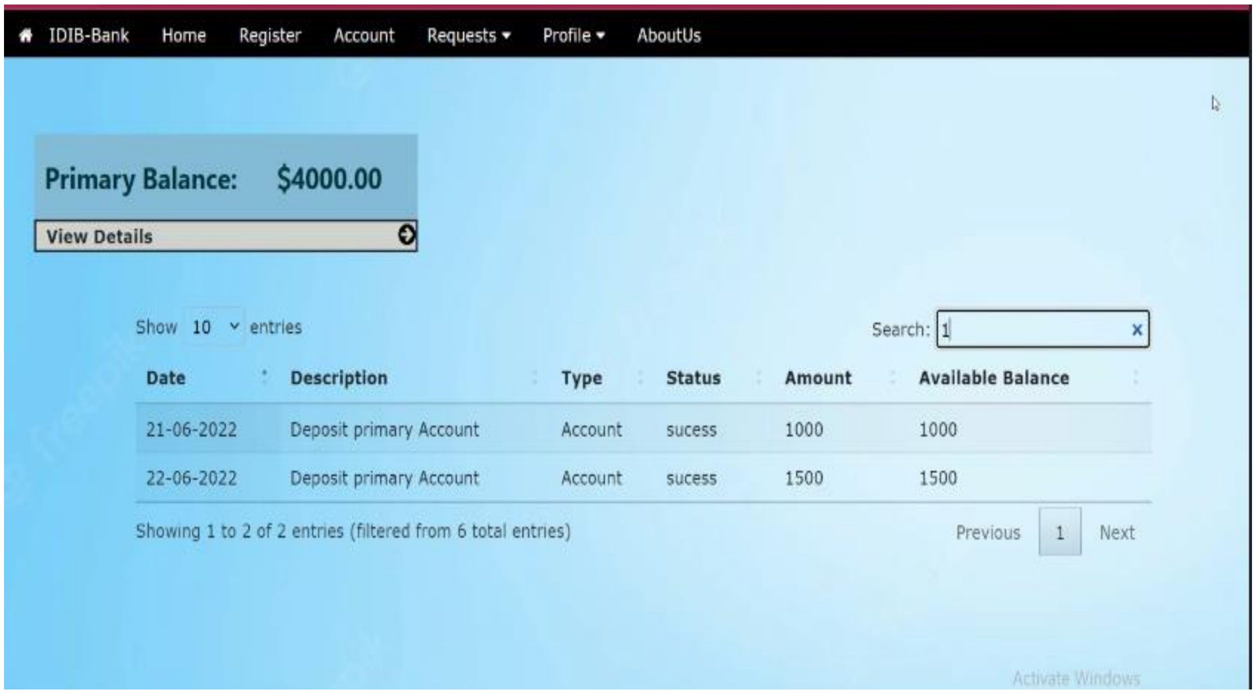


Fig 6.3: Account page



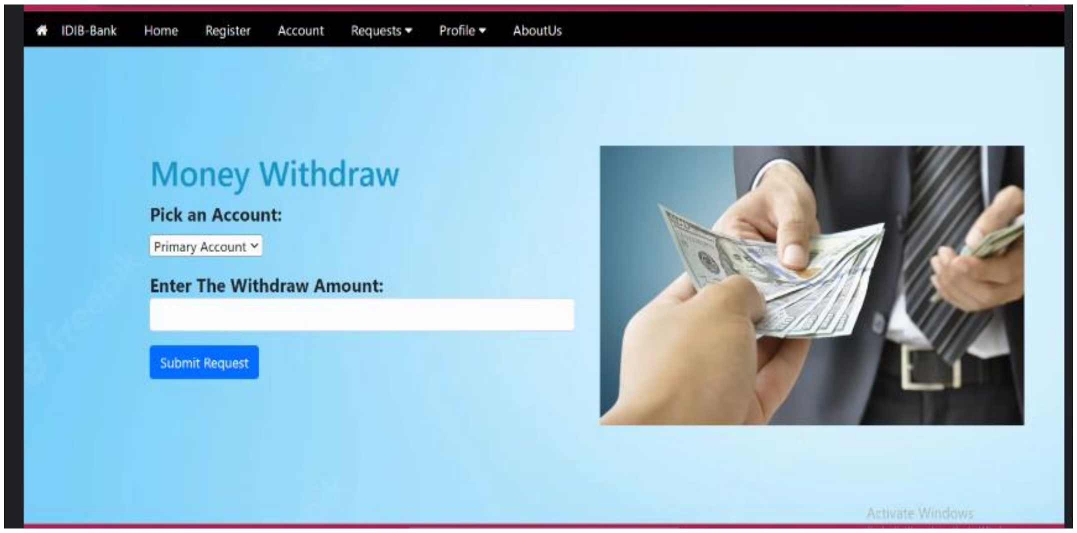
Primary Balance: $4000.00

Show 10 entries Search: 1

Date	Description	Type	Status	Amount	Available Balance
21-06-2022	Deposit primary Account	Account	sucess	1000	1000
22-06-2022	Deposit primary Account	Account	sucess	1500	1500

Showing 1 to 2 of 2 entries (filtered from 6 total entries)

Fig 6.4: Primary Account Page

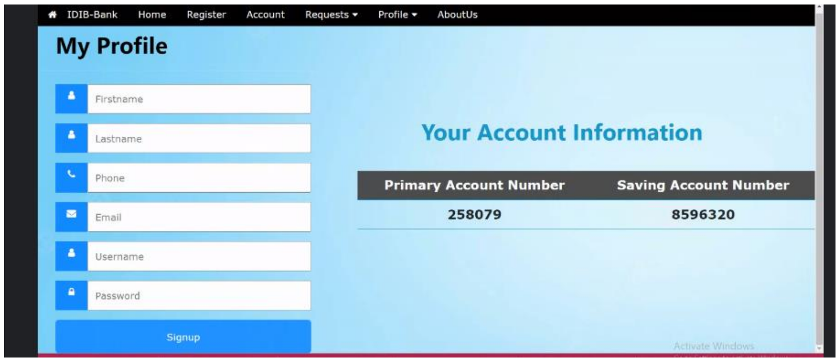


Money Withdraw

Pick an Account: Primary Account

Enter The Withdraw Amount:

Fig 6.5: Money Withdraw page



Your Account Information

Primary Account Number	Saving Account Number
258079	8596320

Fig 6.6: My Profile page

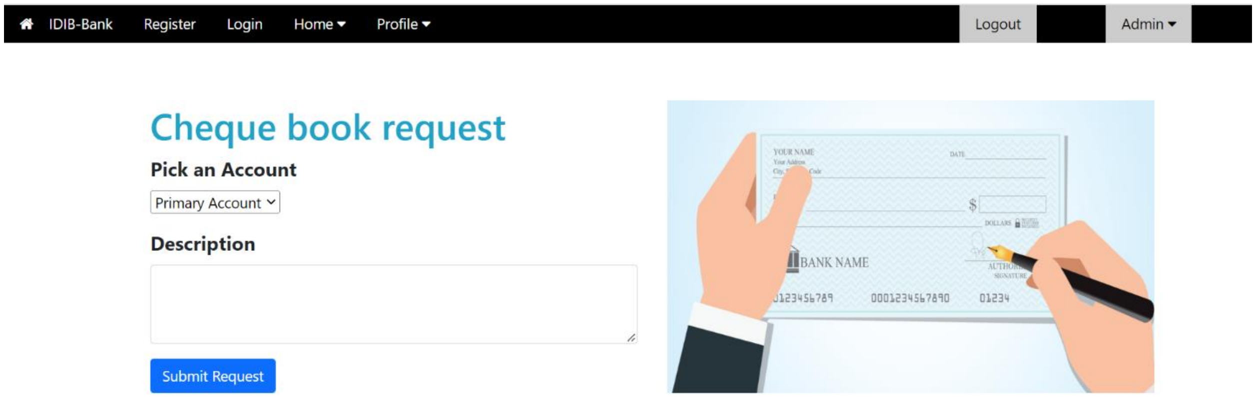
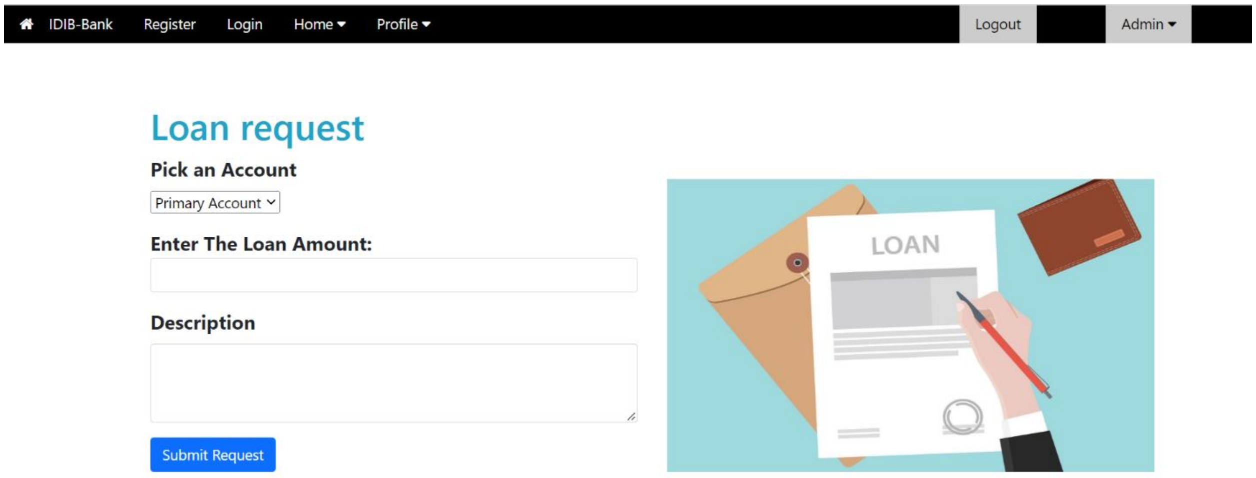


Fig 6.7: Cheque book request page



Loan request

Pick an Account

Primary Account

Enter The Loan Amount:

Description

Submit Request

Fig 6.8: Loan Request page

Admin

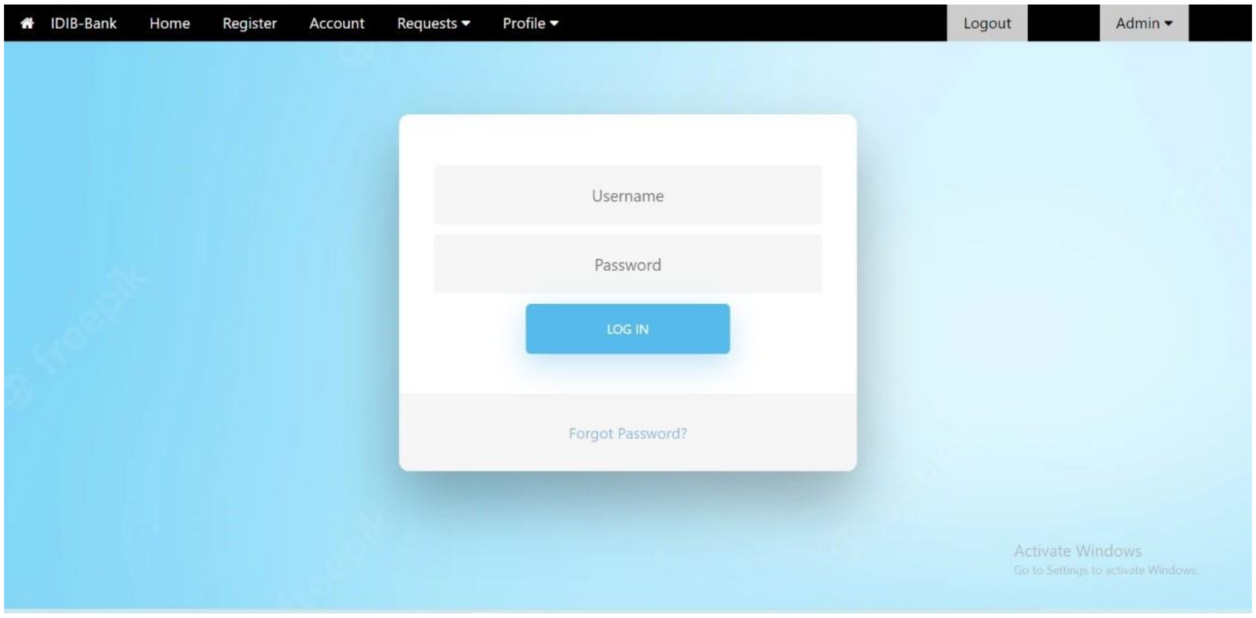
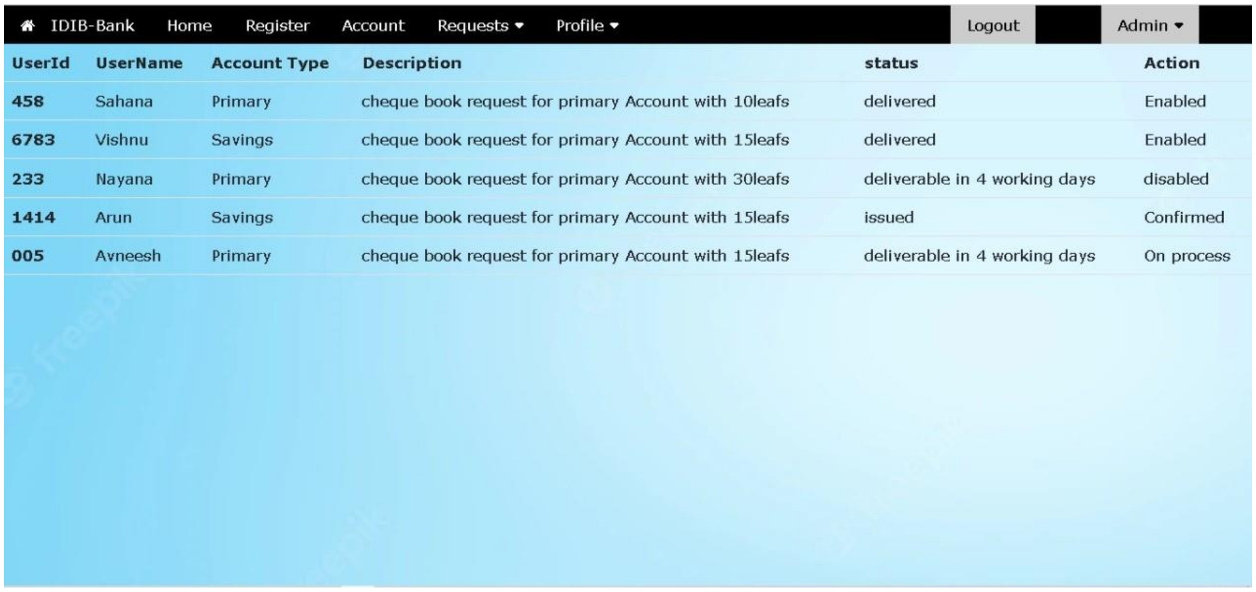
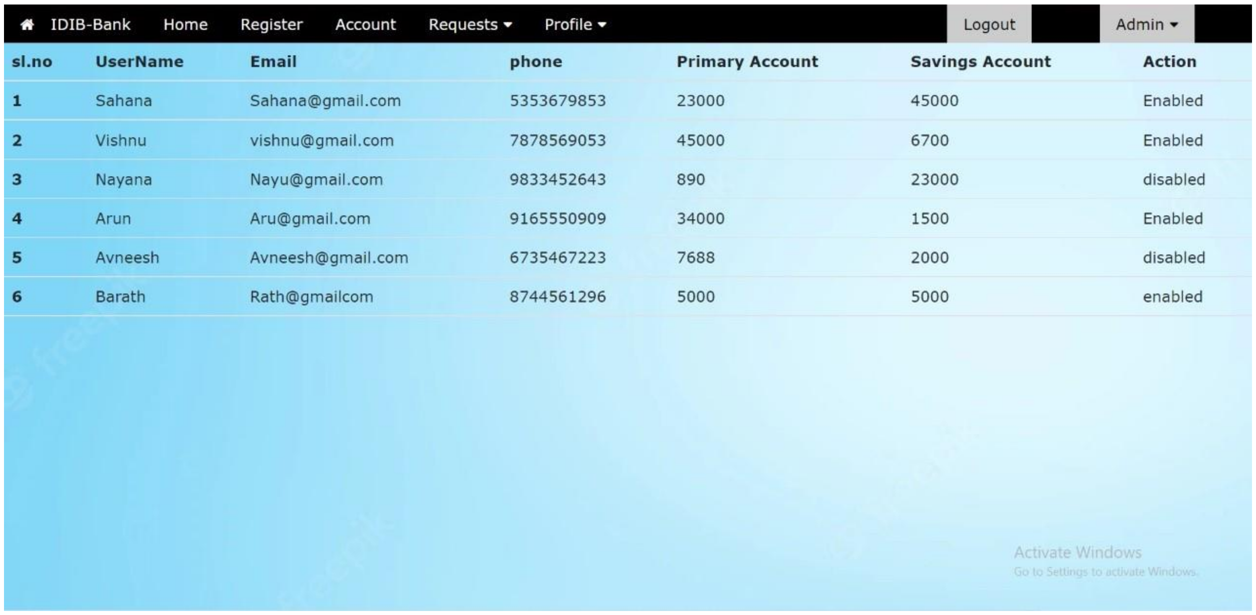


Fig 6.9: Admin login page



UserId	UserName	Account Type	Description	status	Action
458	Sahana	Primary	cheque book request for primary Account with 10leafs	delivered	Enabled
6783	Vishnu	Savings	cheque book request for primary Account with 15leafs	delivered	Enabled
233	Nayana	Primary	cheque book request for primary Account with 30leafs	deliverable in 4 working days	disabled
1414	Arun	Savings	cheque book request for primary Account with 15leafs	issued	Confirmed
005	Avneesh	Primary	cheque book request for primary Account with 15leafs	deliverable in 4 working days	On process

Fig 6.10: UserAccount Page



sl.no	UserName	Email	phone	Primary Account	Savings Account	Action
1	Sahana	Sahana@gmail.com	5353679853	23000	45000	Enabled
2	Vishnu	vishnu@gmail.com	7878569053	45000	6700	Enabled
3	Nayana	Nayu@gmail.com	9833452643	890	23000	disabled
4	Arun	Aru@gmail.com	9165550909	34000	1500	Enabled
5	Avneesh	Avneesh@gmail.com	6735467223	7688	2000	disabled
6	Barath	Rath@gmailcom	8744561296	5000	5000	enabled

Fig 6.11: Admin Chequebook request

Chapter-7
Conclusion & Scope For Further Development

CONCLUSION

This project developed, incorporated all the activities involved in the browsing centre.

It provides all necessary information to the management as well as the customer with the use of this system; the user can simply sit in front of the system and monitor all the activities without any physical movement of the file. Management can service the customers request best in time.

The system provides quickly and valuable information. These modules have been integrated for effective use of the management for future forecasting and for the current need.

SCOPE FOR FURTHER DEVELOPMENT

The system can be designed for further enhancement. This could also be developed according to the growing needs of the customer.

Thank You