PROJECT REPORT

ON

BANKING SYSTEM

IN FULFILMENT OF THE REQUIREMENT OF THIRD YEAR DBMS PROJECT

Btech Computer Science and Engineering

SUBMITTED BY

Ravi Agrawal (2020BCS132)

Indrapal Singh Shiledar (2020BCS127)

Bhagwant Singh Khalsa (2020BCS128)

UNDER GUIDANCE Of

Miss. Swapnaja Moralwar

AND

Dr. Jaishri Waghmare
(HOD of CSE Department)

Department of Computer Science & Engineering

SHRI GURU GOBIND SINGHJI INSTITUTE OF ENGINEERING AND TECHNOLOGY VISHNUPURI NANDED

- 431606 (M.S) INDIA ACADEMIC YEAR: 2021-2022

Abstract

The Bank Account Management System is an application for maintaining a person's account in a bank. In this project I tried to show the working of a banking account system and cover the basic functionality of a Bank Account Management System. To develop a project for solving financial applications of a customer in banking environment in order to nurture the needs of an end banking user by providing various ways to perform banking tasks. Also to enable the user's work space to have additional functionalities which are not provided under a conventional banking project. The Bank Account Management System undertaken as a project is based on relevant technologies. The main aim of this project is to develop software for Bank Account Management System. This project has been developed to carry out the processes easily and quickly, which is not possible with the manuals systems, which are overcome by this software. This project is developed using PHP, HTML language and MYSQL use for database connection. Creating and managing requirements is a challenge of IT, systems and product development projects or indeed for any activity where you have to manage a contractual relationship. Organization need to effectively define and manage requirements to ensure they are meeting needs of the customer, while proving compliance and staying on the schedule and within budget. The impact of a poorly expressed requirement can bring a business out of compliance or even cause injury or death. Requirements definition and management is an activity that can deliver a high, fast return on investment. The project analyzes the system requirements and then comes up with the requirements specifications. It studies other related systems and then come up with system specifications. The system is then designed in accordance with specifications to satisfy the requirements. The system design is then implemented with MYSQL, PHP and HTML. The system is designed as an interactive and content management system. The content management system deals with data entry, validation confirm and updating whiles the interactive system deals with system interaction with the administration and users. Thus, above features of this project will save transaction time and therefore increase the efficiency of the system.

INDEX

1	**INTRODUCTION**	4 - 5
1.1	Basic Information about Project	4
1.2	Flow Of Modules	5
2	**DATABASE SECTION**	6 – 8
2.1	Er Diagram	6
3	DATA FLOW DIAGRAMS	7
4	**FEATURES**	13
5	TECHNOLOGY STACK	
5	**SCREENSHOTS**	14 - 18
6	**REFERENCES**	19

1.INTRODUCTION

1.1 BASIC INFORMATION -

The "Bank Account Management System" project is a model Internet Banking Site. This site enables the customers to perform the basic banking transactions by sitting at their office or at

homes through PC or laptop. The system provides the access to the customer to create an account, deposit/withdraw the cash from his account, also to view reports of all accounts present. The customers can access the banks website for viewing their Account details and perform the transactions on account as per their requirements. With Internet Banking, the brick and mortar structure of the traditional banking gets converted into a click and portal model, thereby giving a concept of virtual banking a real shape. Thus today's banking is no longer confined to branches. E-banking facilitates banking transactions by customers round the clock globally.

The primary aim of this "Bank Account Management System" is to provide an improved design methodology, which envisages the future expansion, and modification, which is necessary for a core sector like banking. This necessitates the design to be expandable and modifiable and so a modular approach is used in developing the application software. Anybody who is an Account holder in this bank can become a member of Bank Account Management System. He has to fill a form with his personal details and Account Number.

Bank is the place where customers feel the sense of safety for their property. In the bank, customers deposit and withdraw their money. Transaction of money also is a part where customer takes shelter of the bank. Now to keep the belief and trust of customers, there is the

positive need for management of the bank, which can handle all this with comfort and ease.

Smooth and efficient management affects the satisfaction of the customers and staff members, indirectly. And of course, it encourages management committee in taking some needed decision for future enhancement of the bank.

Now a day's, managing a bank is tedious job up to certain limit. So software that reduces the work is essential. Also today's world is a genuine computer world and is getting faster and
faster day-by-day. Thus, considering above necessities, the software for bank management has became necessary which would be useful in managing the bank more efficiently.
All transactions are carried out online by transferring from accounts in the same Bank or international bank. The software is meant to overcome the drawbacks of the manual system.
The software has been developed using the most powerful and secure backend MYSQL database and the most widely accepted web oriented as well as application oriented

1.2 The flow of the Modules -

The Modules description of Bank Account Management System project. These modules will be developed in PHP source code and MYSQL database.

1. **Create New Account:** A customer who having the account in the world can create a virtual account through this module. This module receives the customer profile details and the bank account details with the proof of the ownership of the bank account.
2. **Login:** Virtual account holders can login in to the system using this module. Thus this is the secured login page for the customers in the website.
3. **Virtual Account:** After the approval of new virtual account creation, the customer assigned a unique virtual account number to make the online money transactions. This module views the details of the logged customer's virtual account.
4. **Bank Accounts:** A customer may have more than one bank account in various banks, in this case, the customer prompted to decide which bank account should reflect in the account debit or amount credit. For these operations customers can add their owned bank accounts here and it will be approved by the administrations of the system.
5. **Fund Transfer:** This is the module to make fund transfer to the virtual bank account holders or the usual bank account holders from the customer's specified bank

account.

6. **Beneficiary:** Beneficiary is a person who receives money. Here the customer can add the beneficiaries to make fund transfer in the future.

7. **Transactions:** This module displays the transactions made by the customer in the particular date with the transaction details.

8. **Administrative Control:** This module contains the administrative functions such as view all virtual account, transactions, approve bank accounts, approve virtual accounts etc.

There are other features and actions that can be performed on a back account but we are not going to look at bank accounts in their entirety only the basics, this way we avoid over complicating the exercise. The purpose of this whole exercise is to show the usefulness of object oriented programming as opposed to really wanting to create a banking system

Translating the above points into software is easy when you think of a bank account as an object:

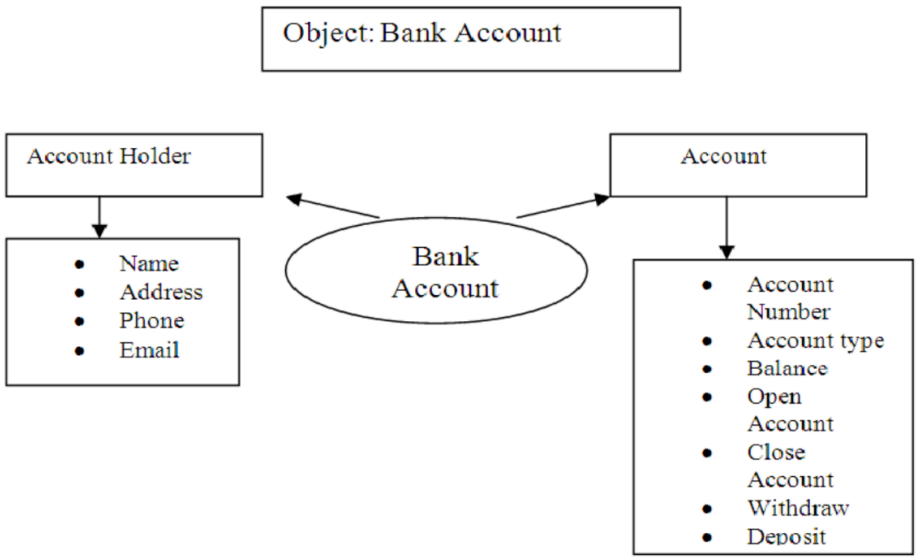


Figure-2.1: Bank Account System

Just by looking at the above picture, we can work out what methods we need our class to have:

2.DATABASE SECTION

The database, called a bank, will have two tables, one called accounts and the other called customer. Each will hold information about either the account or the customer. The two tables will be linked through a foreign key. The customer table has the following fields:

Field	Description
cusid	Creates a unique customer id for each new customer
name	Stores the customer name
address	Stores the customer address
acc_id	Links the customer to a account in the accounts table

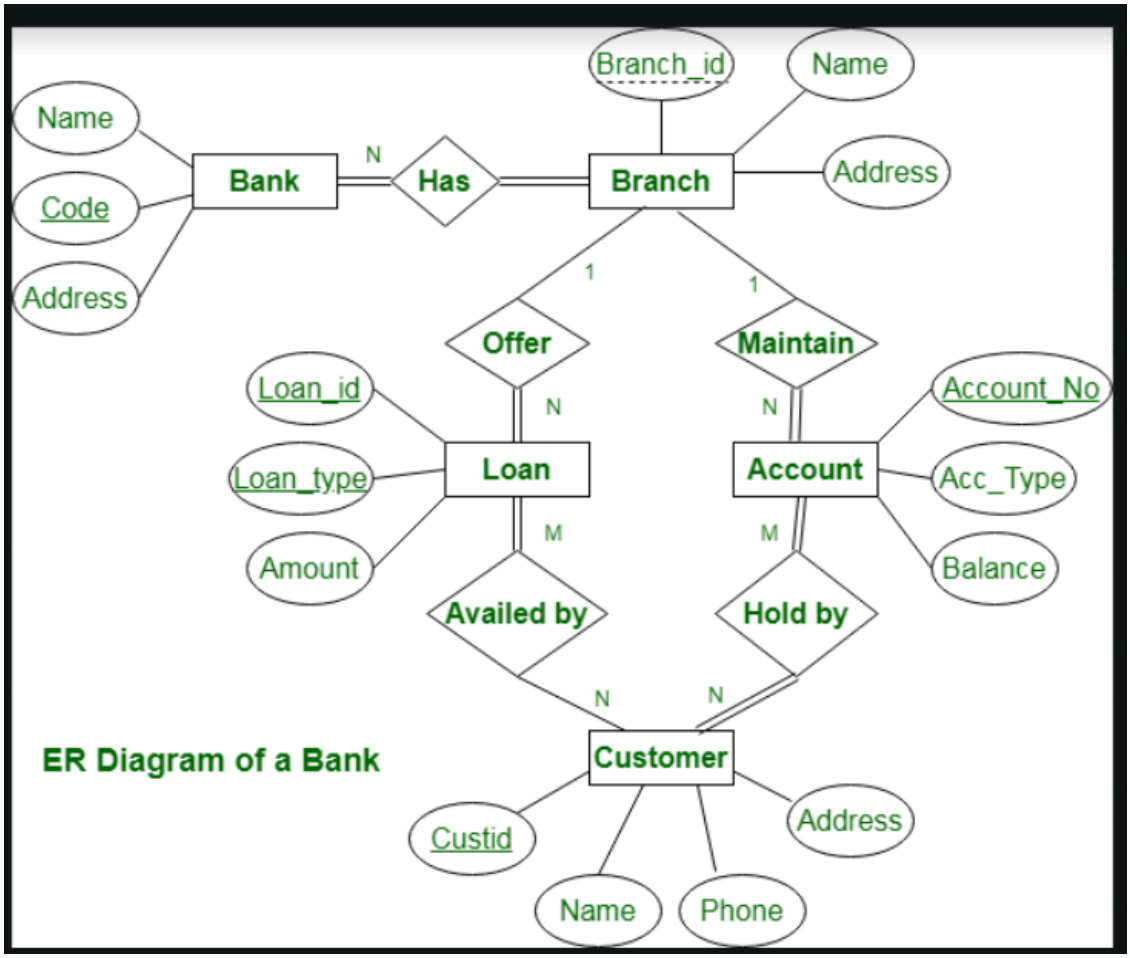
Field	Description
accid	Creates a unique account number for each new account
accno	Stores the account number
type	Stores the account type
balance	Stores the account balance
active	Shows the account status

2.1 ER DIAGRAM

ER diagram is known as Entity-Relationship diagram. It is used to analyze to structure of the Database. It shows relationships between entities and their attributes. An ER model provides a means of communication.

ER diagram of Bank has the following description :

- Bank have Customer.
- Banks are identified by a name, code, address of main office.
- Banks have branches.
- Branches are identified by a branch_no., branch_name, address.
- Customers are identified by name, cust-id, phone number, address.
- Customer can have one or more accounts.
- Accounts are identified by account_no., acc_type, balance.
- Customer can avail loans.
- Loans are identified by loan_id, loan_type and amount.
- Account and loans are related to bank's branch.



3. DATA FLOW DIAGRAM :-

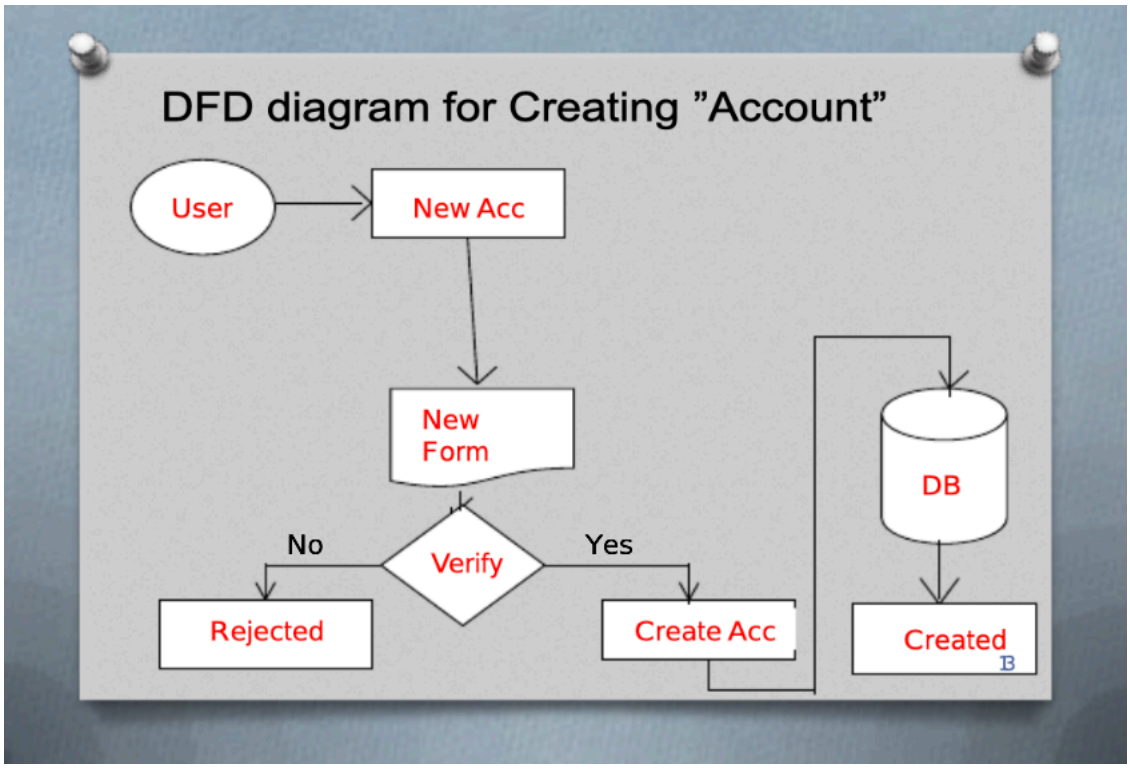
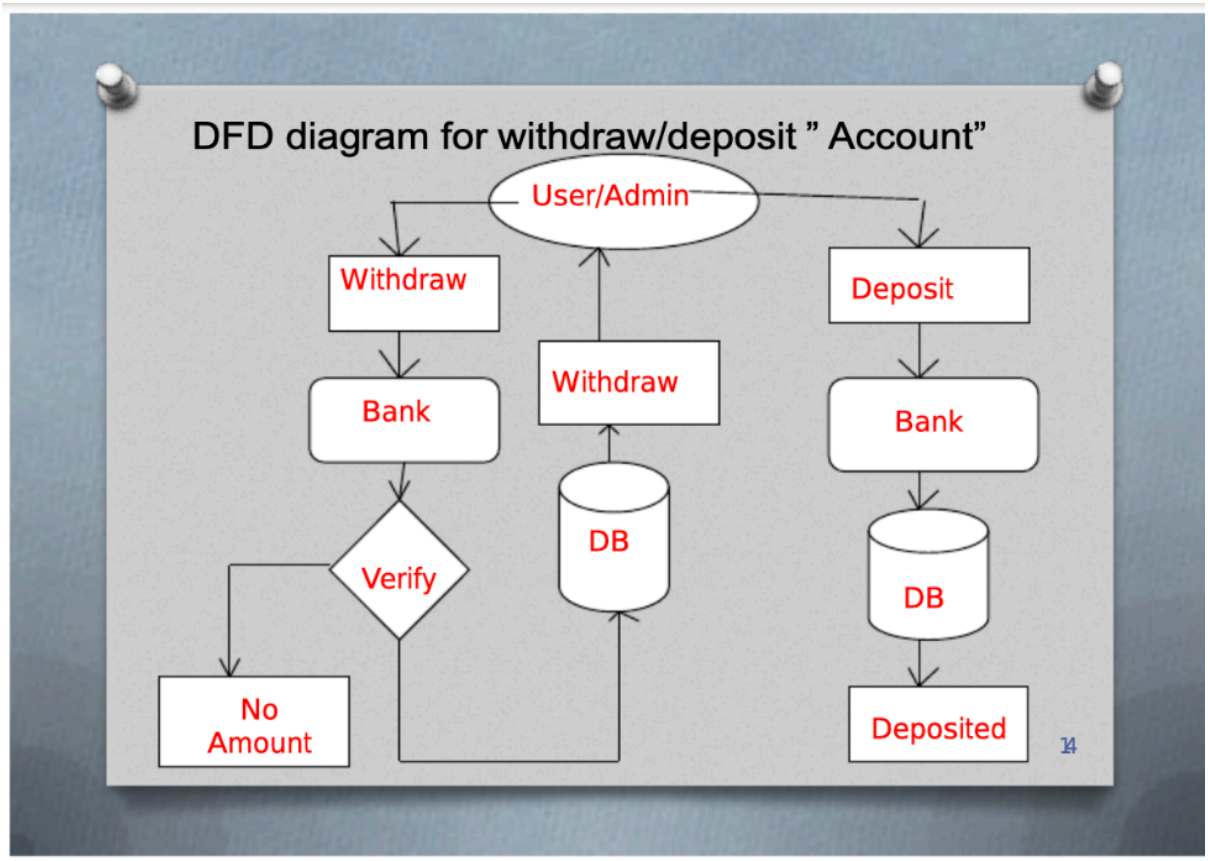


Figure-3.1: Create new account DFD



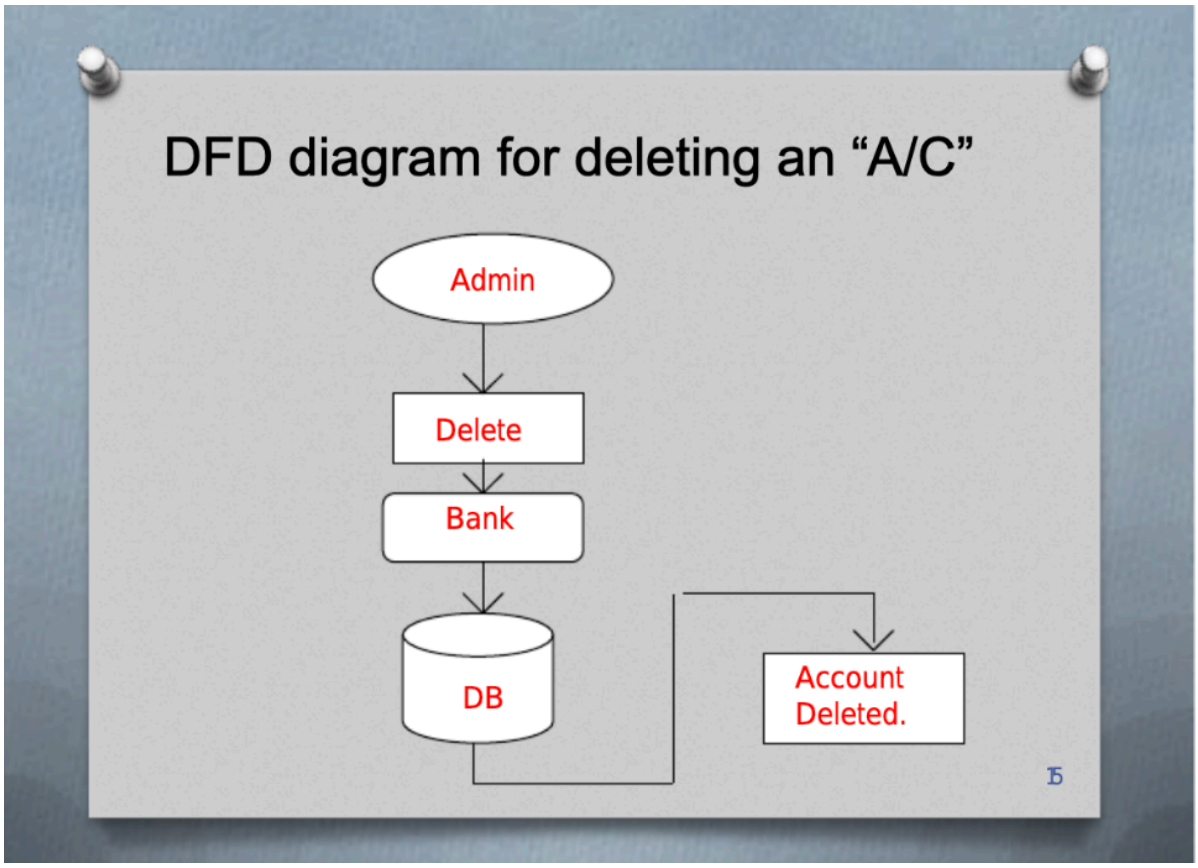


Figure-3.3: Deleting an account DFD

4.FEATURES :-

Register new bank customers. View customer information. Manage customer accounts

Functional Requirements :-

- Sign in with login and password.
- Update personal details.
- Change password.
- View balance.
- View personal history of transactions.
- Transfer money.
- Withdraw.
- Submit Cash.

Functions of banking systems:

- Accepting of deposits and withdrawals.
- Lending of funds.
- Offering loans.
- Offering checking accounts,
- Remittance of funds.
- Offering credit and debit cards.
- Bill payment services and safe deposits.

Technology Stack

The Project can be made using various technology stacks but the MySQL was compulsory to be used. As it is a website, we used technologies related to the web.

The Technologies are :

- HTML5
- CSS3
- Bootstrap
- PHP
- MySQL
- JavaScript
- JQuerry
- Ajax

Applications Used are :

- Xampp
- Vs-code

Technologies Explained :

HTML5:

Html 5 is the latest version of Hypertext Markup Language which is the code behind the scenes that describes the structure and architecture of the website. We have added many elements of PHP code to the HTML the document type is not specified as index.html its index.php. But HTML is used in it.

CSS3 :

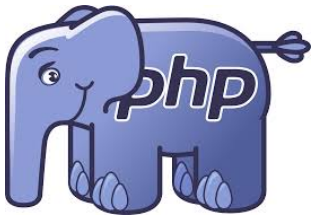
CSS3 (Cascading Style Sheets) has been used in with HTML to create formatted content that looks nice on the webpage. The navbar, login page, and the user section styling are done using CSS 3 individually.

Bootstrap 5:

Bootstrap 5 is a free open-source front-end development framework used for creating websites and web apps. Designed to enable responsive development it has a collection of template code for everything that is used in web dev. Here we have used dropdowns, Modals, and JS transitions to make the website more live and playful.

PHP :

Hypertext Pre-processor (PHP) is a general-purpose scripting the language that is especially suited to web development. Fast, flexible, and Powerful tool for browser-side scripting. The Php is used in the HTML template to instantly feed in the information according to the user by interacting with our MySQL database.



JavaScript:

JavaScript, often abbreviated as JS, is a programming language that is one of the core technologies of the World Wide Web, alongside HTML and CSS.

Jquery:

jQuery is a JavaScript library designed to simplify HTML DOM tree traversal and manipulation, as well as event handling, CSS animation, and Ajax. It is free, open-source software using the permissive MIT License.

AJAX :

AJAX stands for **A**synchronous **Ja**vaScript and **X**ML. AJAX is a new technique for creating better, faster, and more interactive web applications with the help of HTML, CSS, and Bootstrap. Here we communicate with our Apache Tomcat server by xampp using AJAX to send the request, get the request, and simultaneously maintain the user session.

MySQL :

MySQL, the most popular Open-Source SQL database management system, is developed, distributed, and supported by Oracle Corporation. Database in MySQL is relational mean that the data is stored in form of interrelated tables Also, the SQL queries that it used to feed in or fetch out the data are executed using PHP on the browser side.

Applications Used :

Xampp :

Xampp is the free and open-source cross-platform web server

package developed by Apache. It consists of an Apache HTTP server, MariaDB as an SQL database, and interpreters and scripts written in PHP. It is a small tool that makes a website run on a local server while interacting with the SQL database via MariaDB.

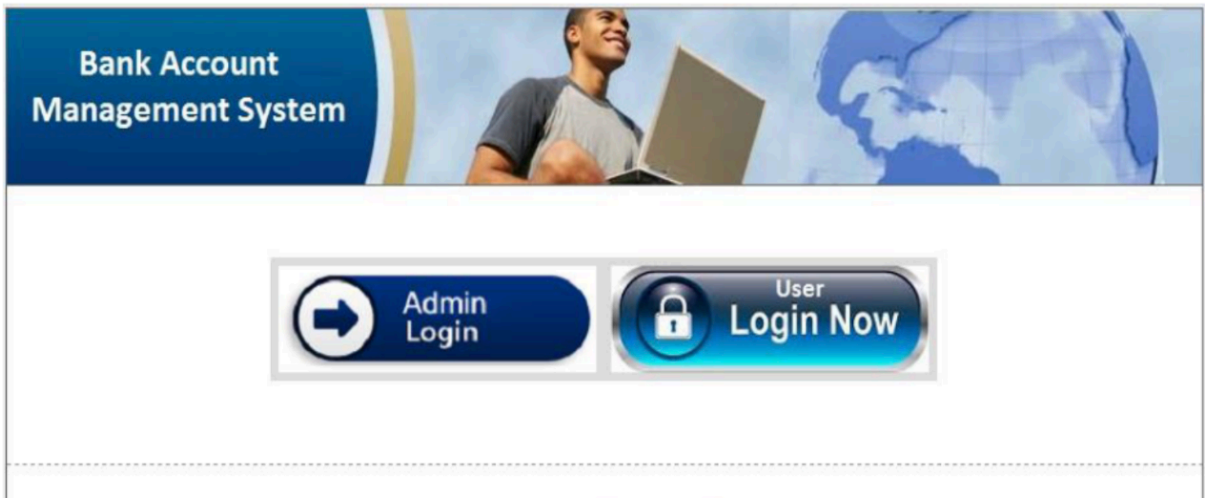
- Apache HTTP server
- MariaDB MySQL Database

VS-code :

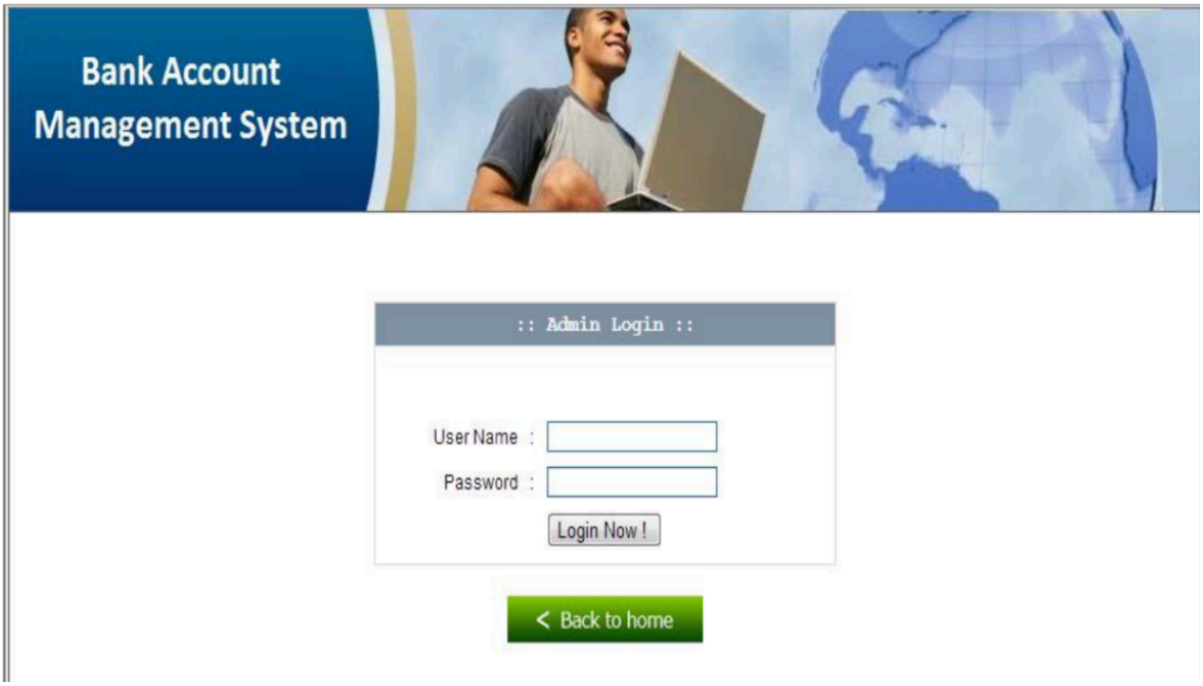
Visual Studio Code, also commonly referred to as VS Code is an open-source editor made by Microsoft with Electron Framework for all the machines out here. It is an IDE Interactive Development An environment that helps along the way to develop various applications effectively. It is free and very popular and developers love to use vs-code

Screenshots :

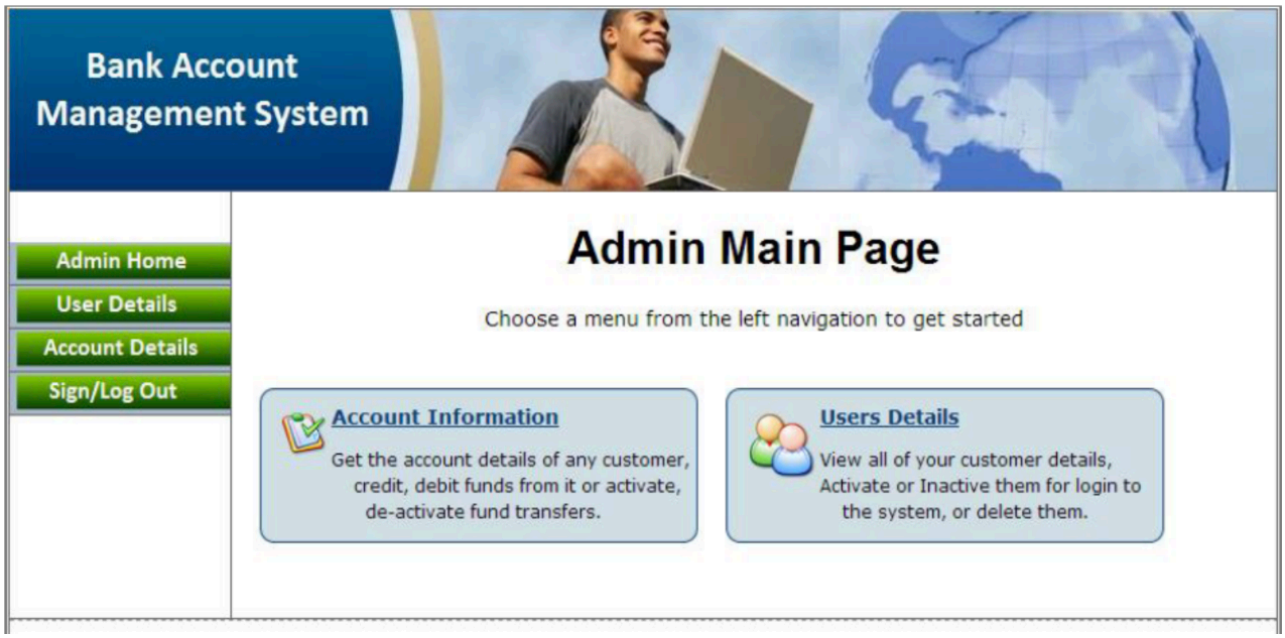
1. Login and Signup Page :-



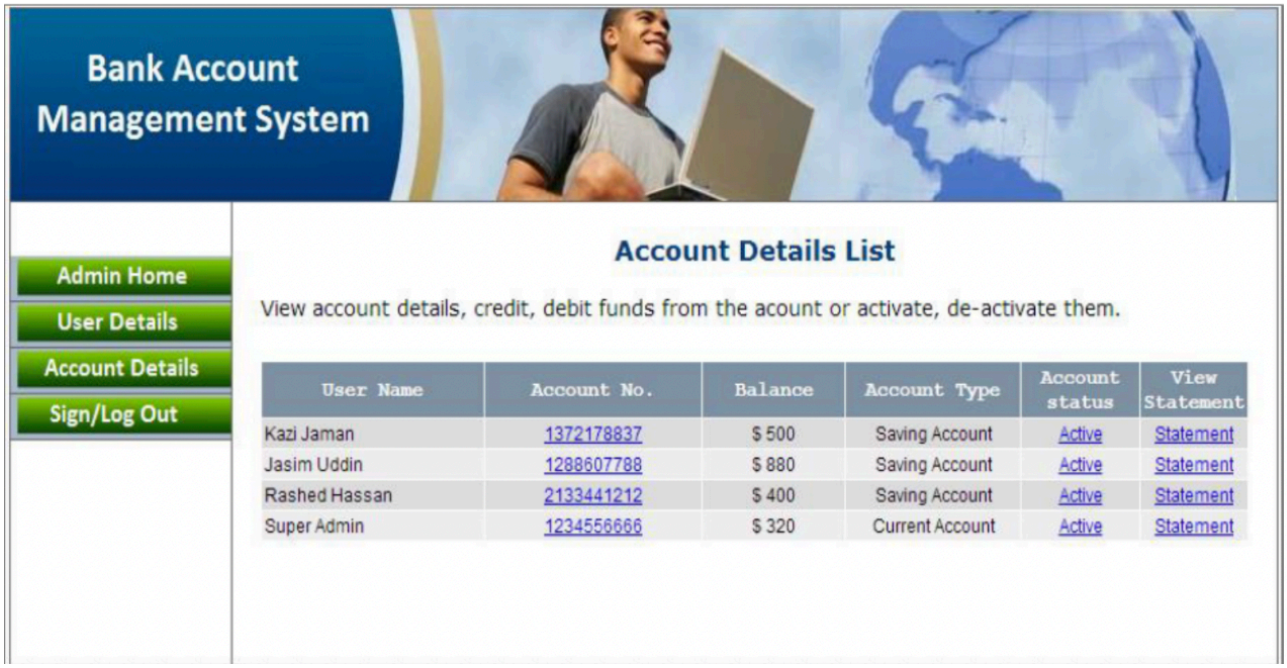
2.Admin Login Section:-



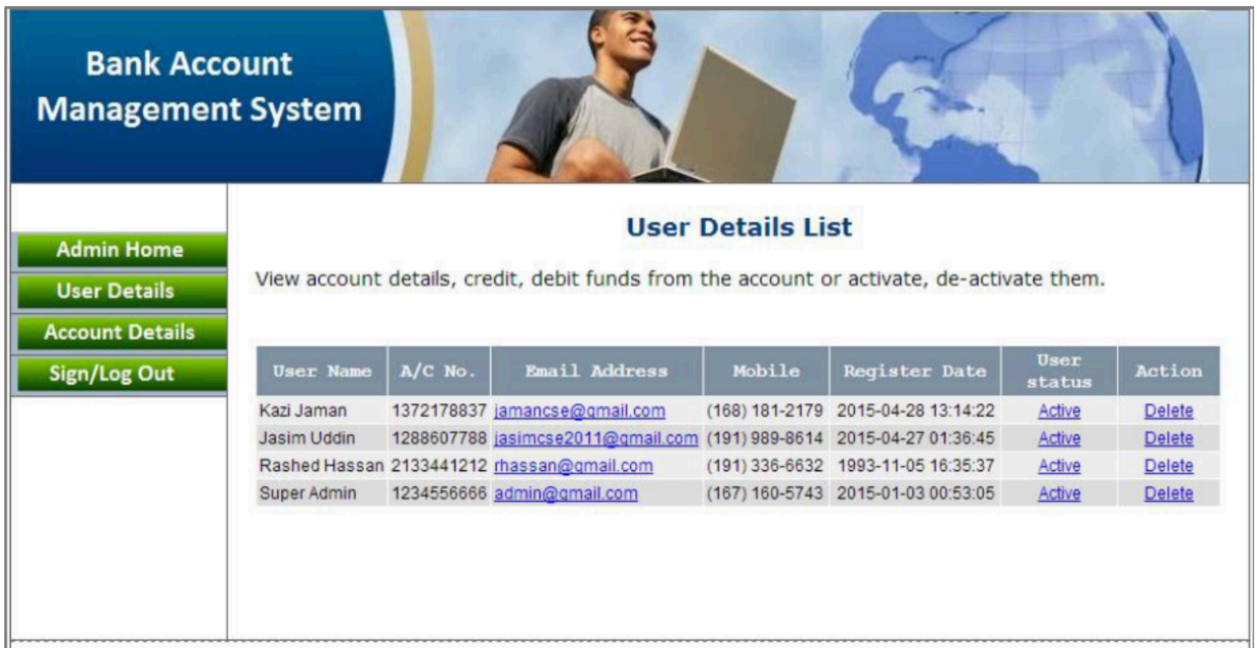
3.Admin Home page:-



4.Account Information:-

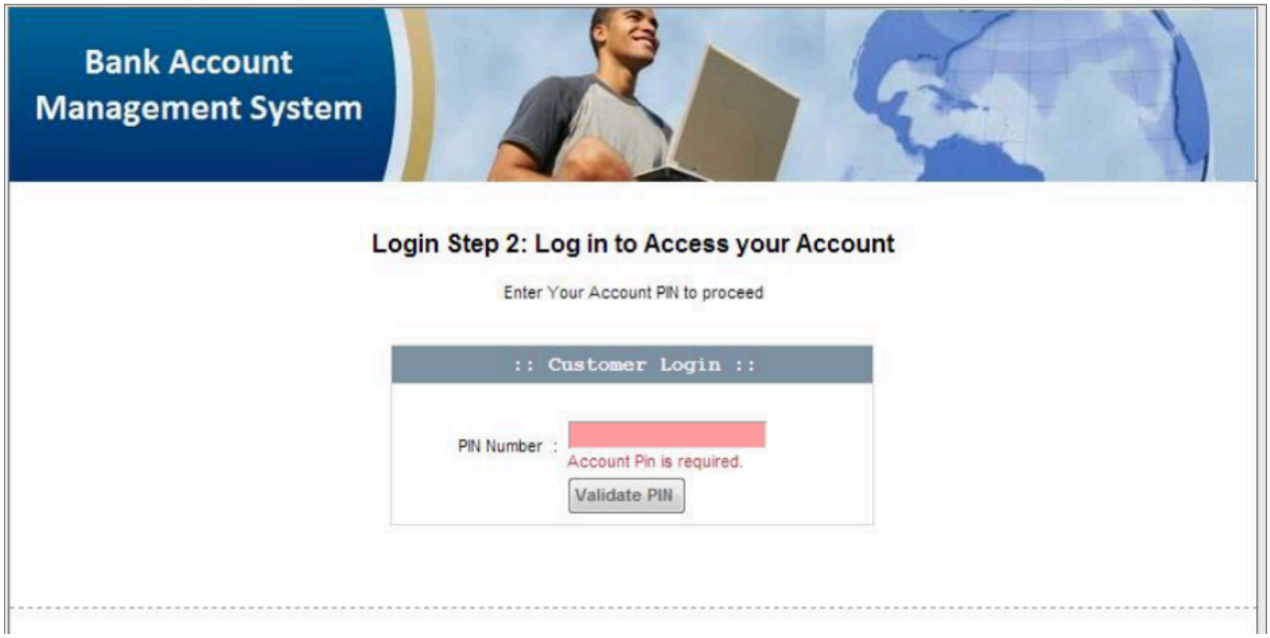


5.User Details List:-

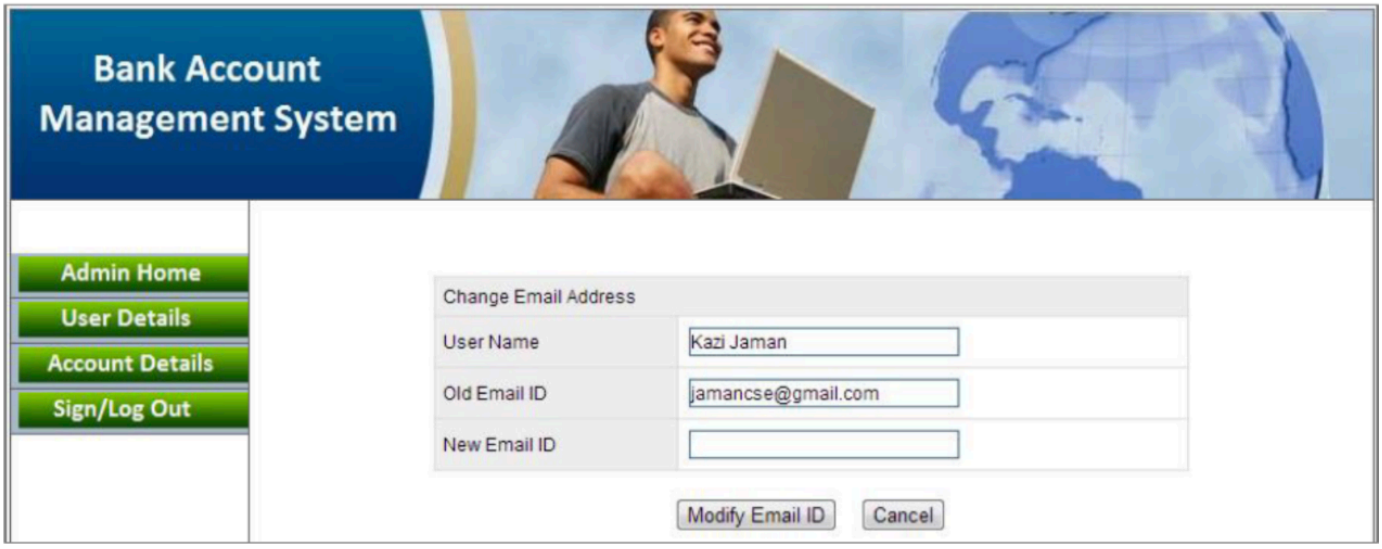


6.User Login Section:-

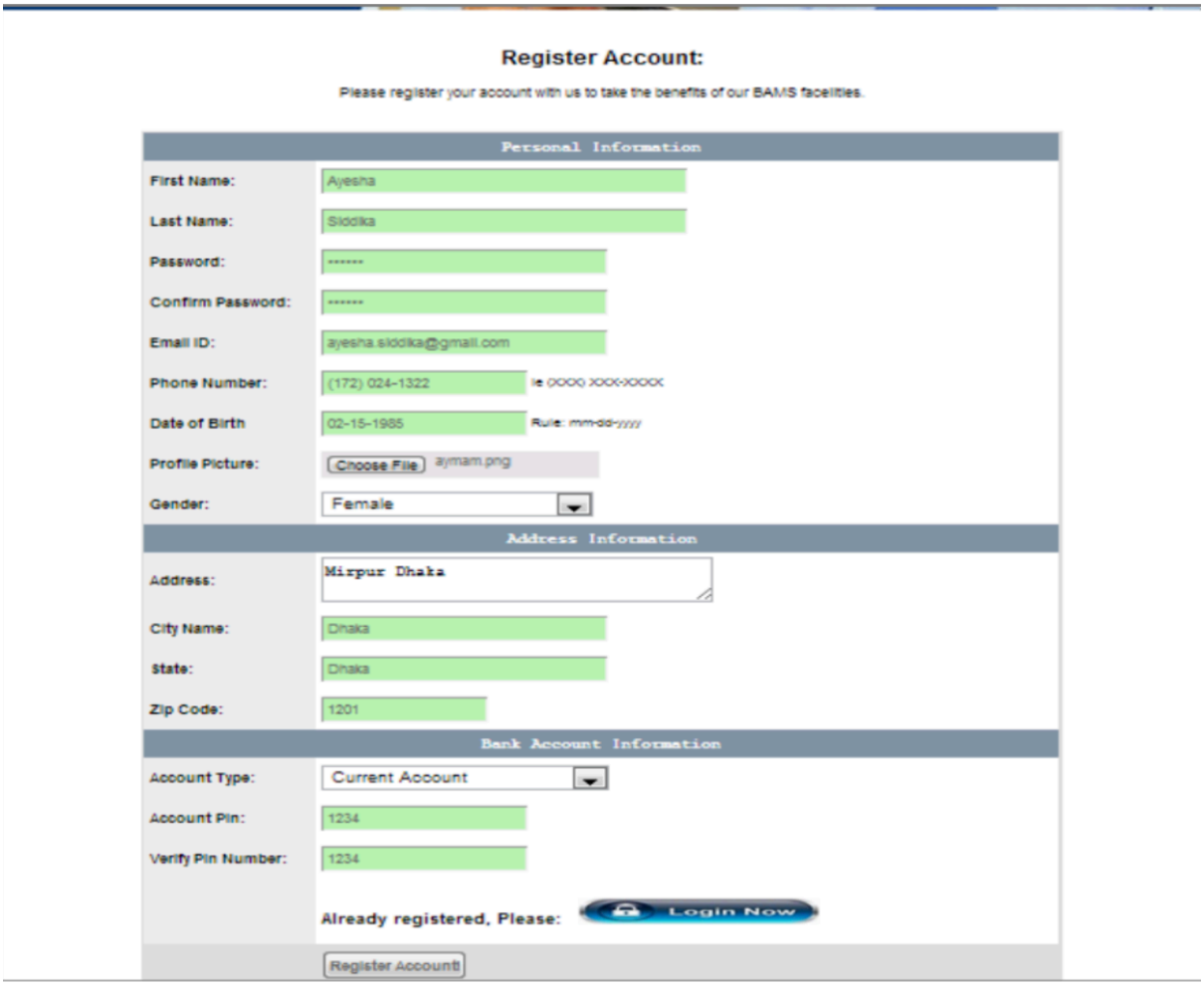
7.Provide PIN :-

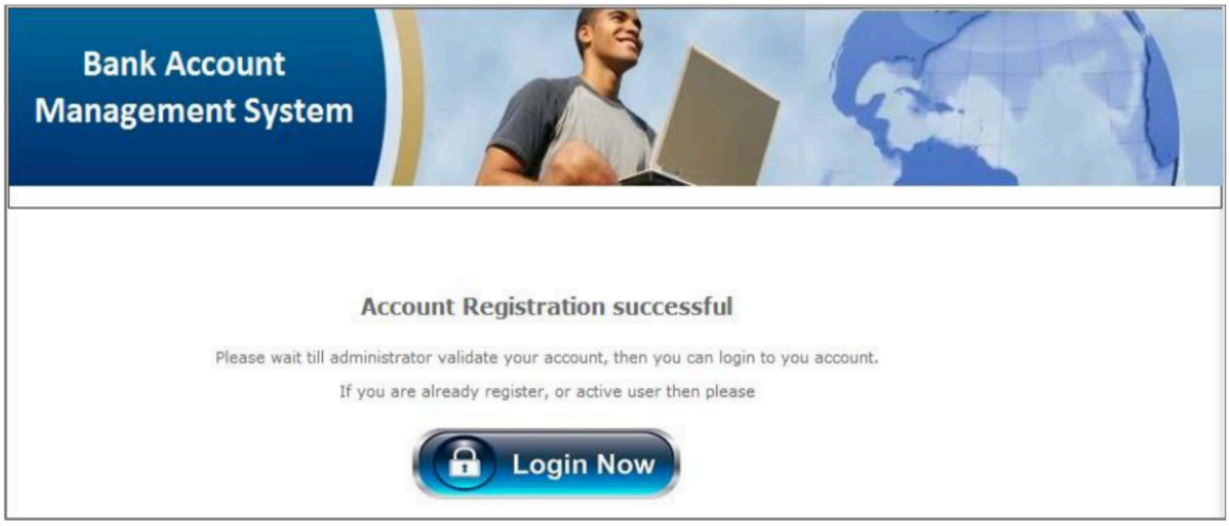


8. Modify User E-mail :-



Open New Account/Register Form :-





jupyter BANKING SYSTEM Last Checkpoint: 09/17/2022 (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Not Trusted | Python 3 (ipykernel)

Run Code

In [ ]:
```
import sqlite3
from flask import Flask,request

app = Flask(__name__)

@app.route('/')
def welcome():
    return "WELCOME TO ABC BANK"

@app.route('/register', methods = ['POST'])
def register():
    data = request.get_json()
    connection = sqlite3.connect('abc.db')
    cursor = connection.cursor()
    cursor.execute('CREATE TABLE IF NOT EXISTS client (id text , password text,balance text)')
    cursor.execute("INSERT INTO client VALUES ('{}','{}','{}')".format(data['id'],data['password'],data['balance']))
    connection.commit()
    return 'Account created successfully...'
```

jupyter BANKING SYSTEM Last Checkpoint: 09/17/2022 (autosaved)

Logout

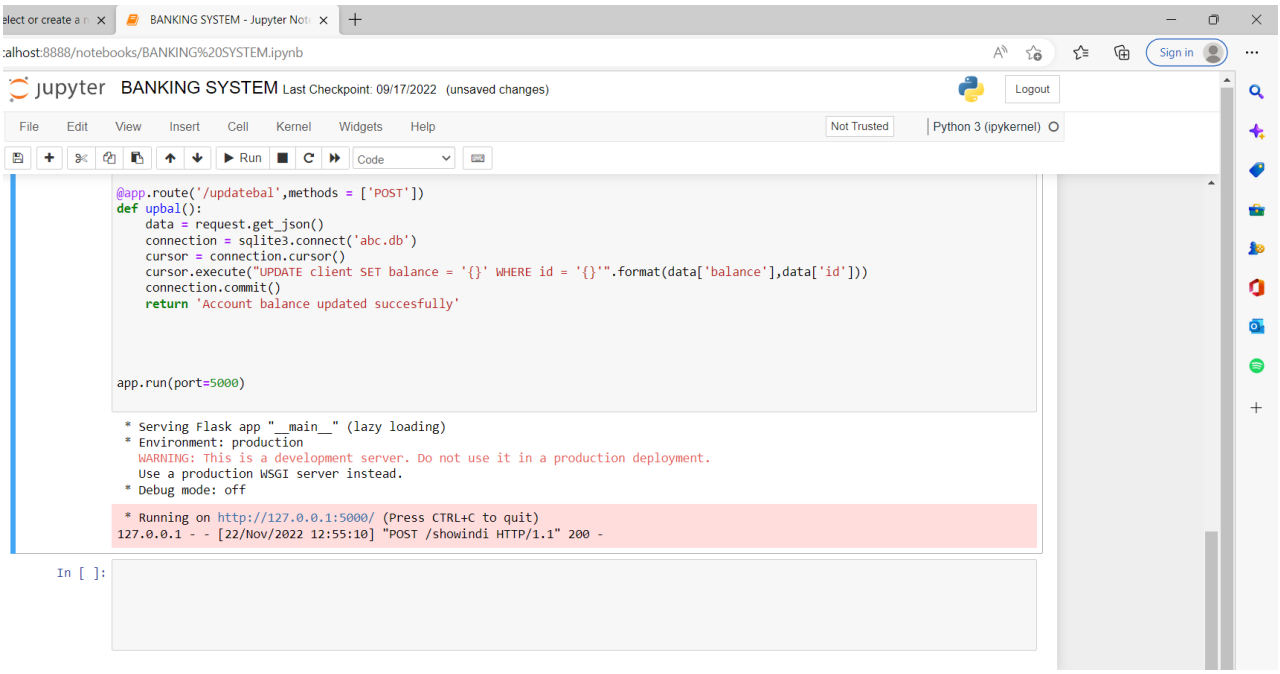
File Edit View Insert Cell Kernel Widgets Help

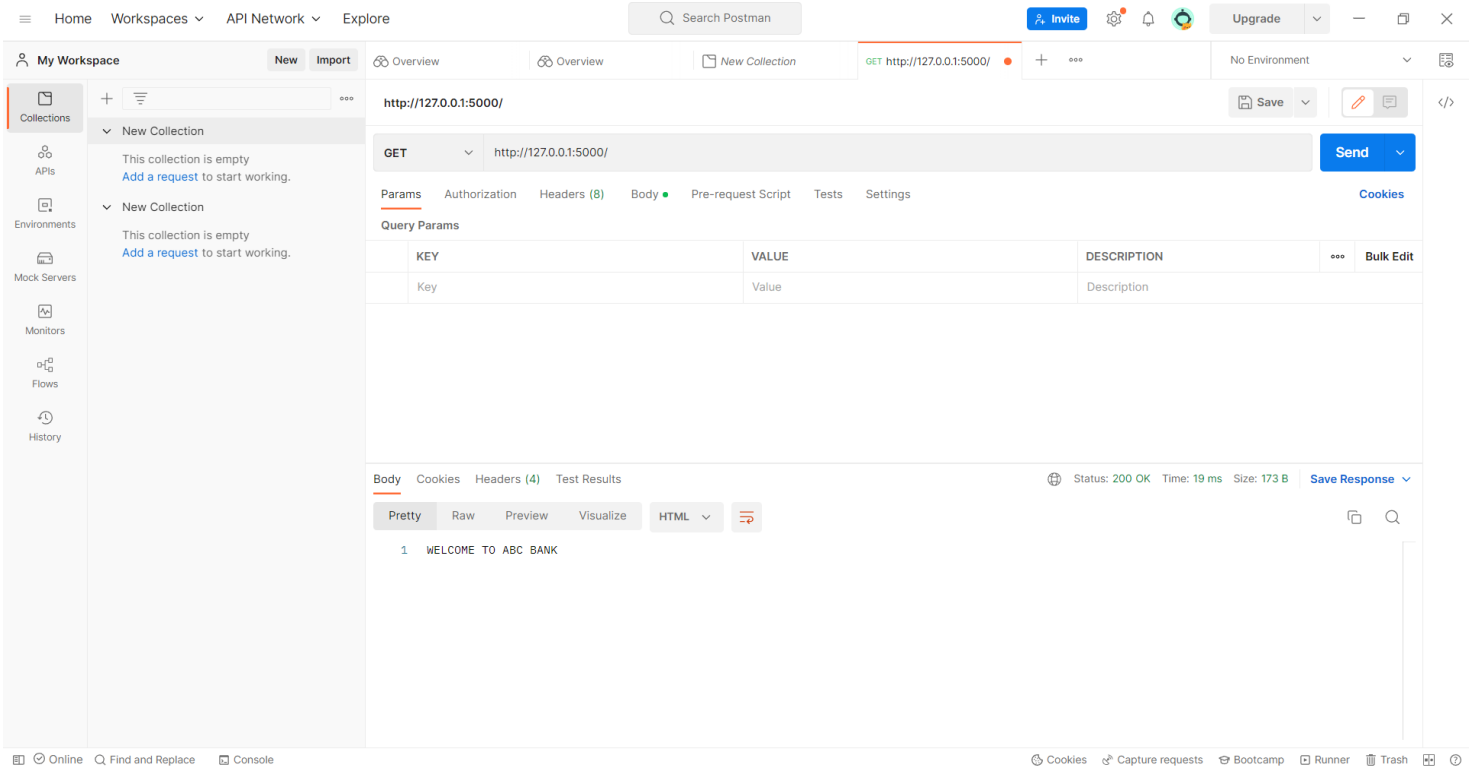
Not Trusted | Python 3 (ipykernel)

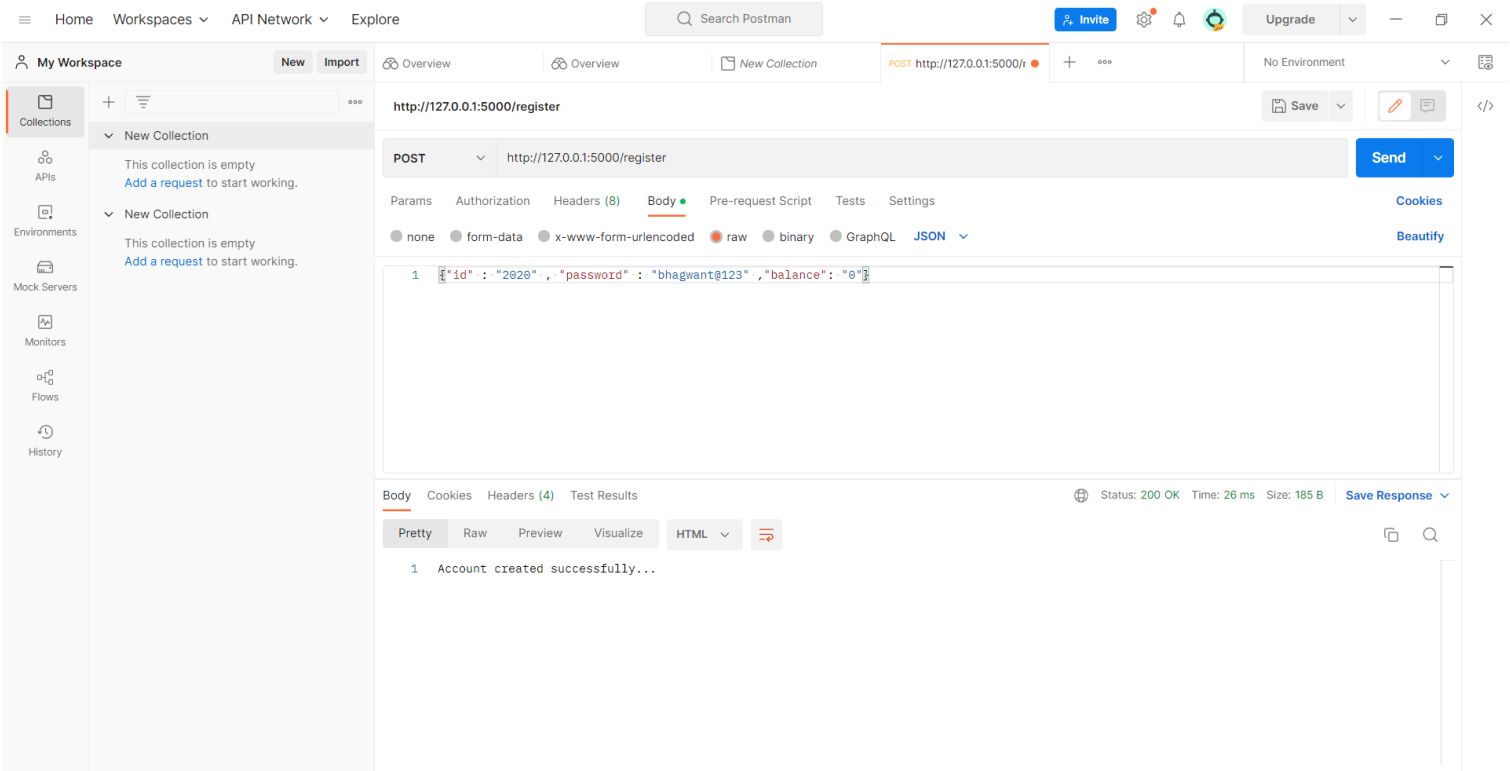
Run Code

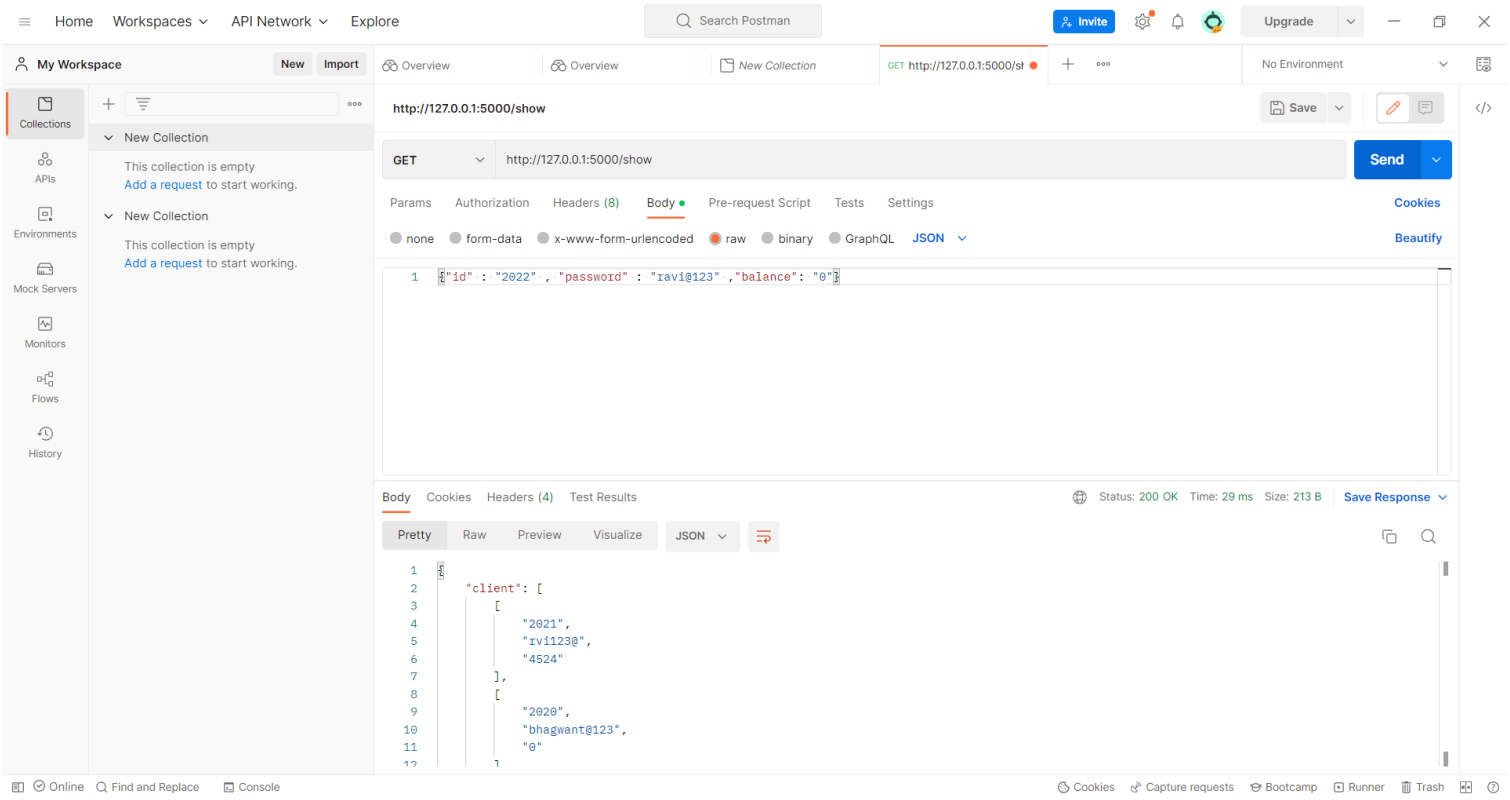
```
@app.route('/show')
def show():
        connection = sqlite3.connect('abc.db')
        cursor = connection.cursor()
        return {'client': list(cursor.execute("SELECT * FROM client"))}


@app.route('/showindi',methods = ['POST'])
def showindi():
    data = request.get_json()
    connection = sqlite3.connect('abc.db')
    cursor = connection.cursor()
    return "'client': {} , {} ".format(data['id'],data['password'])
```

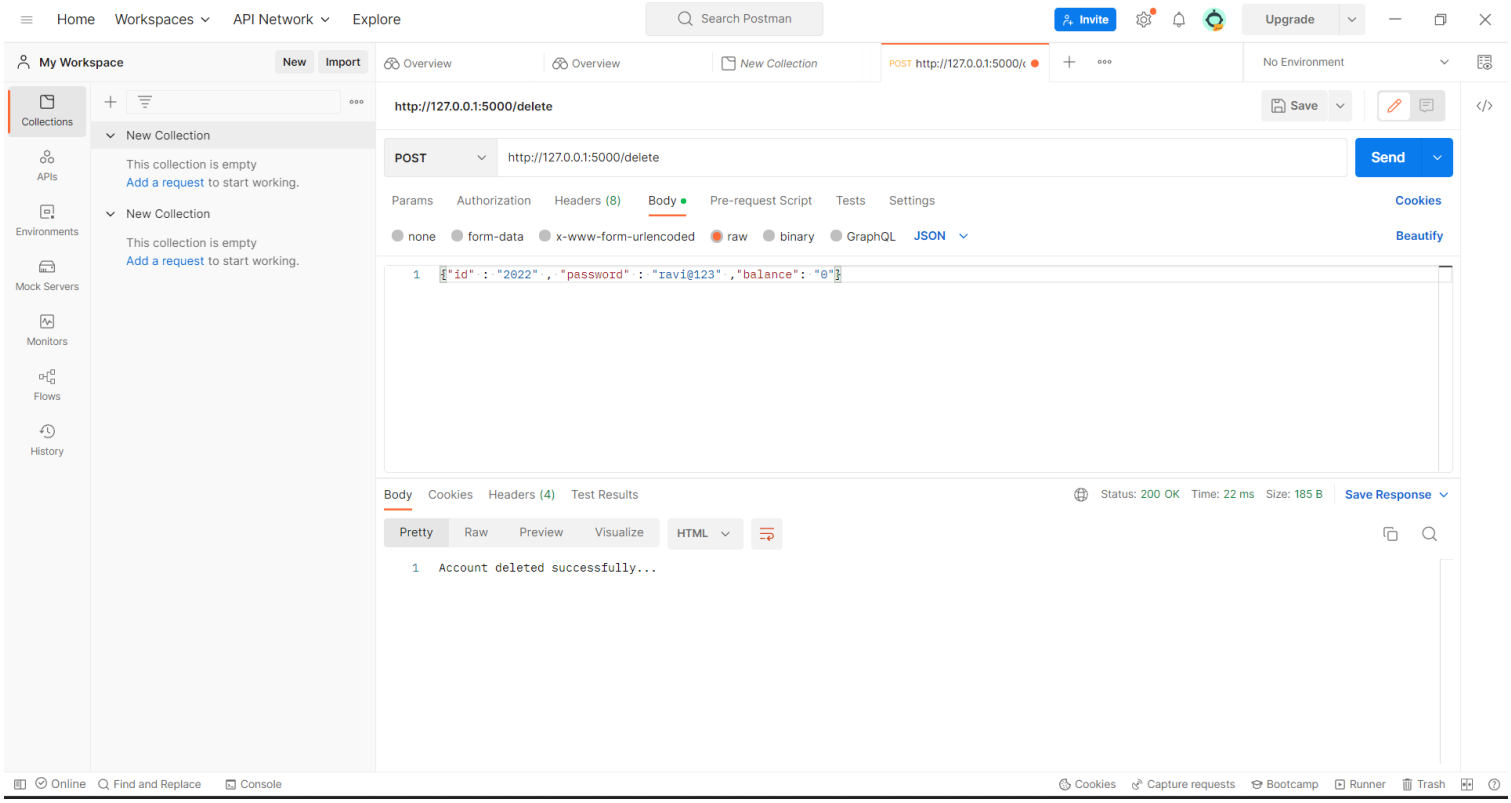

jupyter BANKING SYSTEM Last Checkpoint: 09/17/2022 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code
@app.route('/update', methods = ['POST'])
def update():
data = request.get_json()
connection = sqlite3.connect('abc.db')
cursor = connection.cursor()
cursor.execute("UPDATE client SET password = '{}' WHERE id = '{}'".format(data['password'],data['id']))
connection.commit()
return 'Account details changed successfully...'
@app.route('/delete', methods = ['POST'])
def delete():
data = request.get_json()
connection = sqlite3.connect('abc.db')
cursor = connection.cursor()
cursor.execute("DELETE FROM client WHERE id = '{}'".format(data['id']))
connection.commit()
return 'Account deleted successfully...'


BANKING SYSTEM - Jupyter Not
alhost:8888/notebooks/BANKING%20SYSTEM.ipynb
Sign in
jupyter BANKING SYSTEM Last Checkpoint: 09/17/2022 (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code
@app.route('/updatebal',methods = ['POST'])
def upbal():
data = request.get_json()
connection = sqlite3.connect('abc.db')
cursor = connection.cursor()
cursor.execute("UPDATE client SET balance = '{}' WHERE id = '{}'".format(data['balance'],data['id']))
connection.commit()
return 'Account balance updated succesfully'
app.run(port=5000)
* Serving Flask app "__main__" (lazy loading)
* Environment: production
WARNING: This is a development server. Do not use it in a production deployment.
Use a production WSGI server instead.
* Debug mode: off
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
127.0.0.1 - - [22/Nov/2022 12:55:10] "POST /showindi HTTP/1.1" 200 -
In [ ]:


Home
Workspaces
API Network
Explore
Search Postman
Invite
Upgrade
My Workspace
New
Import
Overview
Overview
New Collection
GET http://127.0.0.1:5000/
No Environment
Collections
APIs
Environments
Mock Servers
Monitors
Flows
History
New Collection
This collection is empty
Add a request to start working.
New Collection
This collection is empty
Add a request to start working.
http://127.0.0.1:5000/
Save
GET
http://127.0.0.1:5000/
Send
Params
Authorization
Headers (8)
Body
Pre-request Script
Tests
Settings
Cookies
Query Params
KEY
VALUE
DESCRIPTION
Bulk Edit
Key
Value
Description
Body
Cookies
Headers (4)
Test Results
Status: 200 OK
Time: 19 ms
Size: 173 B
Save Response
Pretty
Raw
Preview
Visualize
HTML
1 WELCOME TO ABC BANK
Online
Find and Replace
Console
Cookies
Capture requests
Bootcamp
Runner
Trash


Home
Workspaces
API Network
Explore
Search Postman
Invite
Upgrade
My Workspace
New
Import
Overview
Overview
New Collection
POST http://127.0.0.1:5000/r
No Environment
Collections
APIs
Environments
Mock Servers
Monitors
Flows
History
New Collection
This collection is empty
Add a request to start working.
New Collection
This collection is empty
Add a request to start working.
http://127.0.0.1:5000/register
Save
POST
http://127.0.0.1:5000/register
Send
Params
Authorization
Headers (8)
Body
Pre-request Script
Tests
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
Beautify
1 {"id" : "2020" , "password" : "bhagwant@123" ,"balance": "0"}
Body
Cookies
Headers (4)
Test Results
Status: 200 OK
Time: 26 ms
Size: 185 B
Save Response
Pretty
Raw
Preview
Visualize
HTML
1 Account created successfully...
Online
Find and Replace
Console
Cookies
Capture requests
Bootcamp
Runner
Trash


Home
Workspaces
API Network
Explore
Search Postman
Invite
Upgrade
My Workspace
New
Import
Overview
Overview
New Collection
GET http://127.0.0.1:5000/sl
No Environment
Collections
APIs
Environments
Mock Servers
Monitors
Flows
History
New Collection
This collection is empty
Add a request to start working.
New Collection
This collection is empty
Add a request to start working.
http://127.0.0.1:5000/show
Save
GET
http://127.0.0.1:5000/show
Send
Params
Authorization
Headers (8)
Body
Pre-request Script
Tests
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
Beautify
1 {"id" : "2022" , "password" : "ravi@123" ,"balance": "0"}
Body
Cookies
Headers (4)
Test Results
Status: 200 OK
Time: 29 ms
Size: 213 B
Save Response
Pretty
Raw
Preview
Visualize
JSON
1 {
2 "client": [
3 [
4 "2021",
5 "rvi123@",
6 "4524"
7],
8 [
9 "2020",
10 "bhagwant@123",
11 "0"
12]
Online
Find and Replace
Console
Cookies
Capture requests
Bootcamp
Runner
Trash


Home
Workspaces
API Network
Explore
Search Postman
Invite
Upgrade
My Workspace
New
Import
Overview
Overview
New Collection
POST http://127.0.0.1:5000/
No Environment
Collections
APIs
Environments
Mock Servers
Monitors
Flows
History
New Collection
This collection is empty
Add a request to start working.
New Collection
This collection is empty
Add a request to start working.
http://127.0.0.1:5000/delete
Save
POST
http://127.0.0.1:5000/delete
Send
Params
Authorization
Headers (8)
Body
Pre-request Script
Tests
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
Beautify
1 {"id" : "2022" , "password" : "ravi@123" ,"balance": "0"}
Body
Cookies
Headers (4)
Test Results
Status: 200 OK
Time: 22 ms
Size: 185 B
Save Response
Pretty
Raw
Preview
Visualize
HTML
1 Account deleted successfully...
Online
Find and Replace
Console
Cookies
Capture requests
Bootcamp
Runner
Trash