

WEB-TECHNOLOGIES PROJECT REPORT

on

E-BANKING SYSTEM

Bachelor of Technology

In

Computer Science and Engineering



Under the esteemed guidance of

Mrs. Venkata Ratnam

Asst. Professor, Dept. of CSE

**DEPARTMENT OF COMPUTER SCIENCE AND
ENGINEERING**

**ANIL NEERUKONDA INSTITUTE OF
TECHNOLOGY AND SCIENCES**

SANGIVALASA, VISAKHAPATNAM

Bonafide Certificate

This is to certify that this project report "E-Banking System" is the Bonafide work of D. Chaitanya(A22126510144), P. Meghana(A22126510167), P. Sai Deepak (A22126510168), N. Lokesh(A22126510166), M. Nikitha(A22126510161). This project is carried out and is submitted in the partial fulfillment of the requirements for the award of BACHELOR OF TECHNOLOGY in Computer Science and Engineering, under Anil Neerukonda Institute of Technology and Sciences during the academic year 2024-2025.

Head of the Department
Prof. G. Srinivas
(Professor & HOD)
Department of CSE
ANITS

Project Guide
Mrs. K.Venkataratnam
(Asst.Professor)
Department of CSE
ANITS

Project Reviewer
G. Pranitha
(Asst. Professor)
Department of CSE
ANITS

PROJECT TITLE: ONLINE BANKING SYSTEM

Name of the Student	Roll No
D. Chaitanya	A22126510144
P. Meghana	A22126510167
P. Sai Deepak	A22126510168
N. Lokesh	A22126510166
M. Nikitha	A22126510161

Declaration

This is to certify that the project work entitled "E-Banking System" is a Bonafide work carried out by as a part of BTech 3rd year 2nd semester of Computer Science and Engineering of Anil Neerukonda Institute of Technology and Sciences, Visakhapatnam during the academic year 2024-2025.

We are D. Chaitanya, P. Meghana, P. Sai Deepak, N. Lokesh, M. Nikitha of 3rd year B.Tech, Department of Computer Science and Engineering from ANITS, Visakhapatnam, hereby declare that the project work entitled "E-Banking System" is carried out by us and is submitted in the fulfillment of the requirement for the award of Bachelor of Technology in Computer Science and Engineering, under Anil Neerukonda Institute of Technology and Sciences during the academic year 2024-2025 has not been submitted to any other university for the award of any kind of degree.

Name	Reg.No
D. Chaitanya	A22126510144
P. Meghana	A22126510167
P. Sai Deepak	A22126510168
N. Lokesh	A22126510166
M. Nikitha	A22126510161

Acknowledgement

The satisfaction that accompanies the successful completion of any task would be incomplete without mentioning the people who made it possible, and whose constant guidance and encouragement always upheld the morale. We take a great pleasure in presenting a project, which is the result of a studied blend of both research and knowledge.

We first take the privilege to thank our Head of the department Prof M. G. SRINIVAS, for permitting us in laying the first stone of success. We feel grateful to thank MRS. K. VENKATA RATNAM Madam, who is our project guide and who shared her valuable knowledge with us and made us understand the real essence of the topic and created interest in us to put our continuous efforts in the project.

Name	Reg.No
D. Chaitanya	A22126510144
P. Meghana	A22126510167
P. Sai Deepak	A22126510168
N. Lokesh	A22126510166
M. Nikitha	A22126510161

Contents

1	Project Abstract	5
2	Objectives	6
3	Software & Modules Used	7
4	Structure of the Project	9
4.1	Overview	9
4.2	Project File Structure	9
5	Code Listings	11
5.1	Entry Point: main.ts	11
5.2	Database Connection: database.ts	13
5.3	Database Init-Schema: init-db.ts	13
5.4	Auth Middleware: auth.ts	15
5.5	Auth Controller: authController.ts	16
5.6	Account Controller: accountController.ts	18
5.7	Dashboard Controller: dashboardController.ts	21
5.8	Dashboard Routes: dashboard.ts	22
5.9	Auth Routes: auth.ts	23
5.10	Transaction Controller: transactionController.ts	23
5.11	Account Routes: accounts.ts	27
5.12	Transaction Routes: transactions.ts	27
5.13	User Model: user.ts	28
5.14	Transaction Model: transaction.ts	31
5.15	Account Model: account.ts	33
5.16	Dashboard View: dashboard.ejs	36
5.17	Login View: login.ejs	41
5.18	Register View: register.ejs	43
5.19	Transfers View: transfers.ejs	47
6	Data Flow Diagram	51
7	Outputs	52
8	References	54

Chapter 1

Project Abstract

The proliferation of online banking has revolutionized the way financial transactions are conducted, offering unparalleled convenience and accessibility. However, with this convenience comes the challenge of ensuring robust security measures to protect sensitive financial data and prevent unauthorized access.

This project aims to address these challenges by proposing innovative solutions to enhance both security and user experience in online banking systems.

The project will begin with a comprehensive analysis of existing security protocols and user interface designs in online banking systems. This analysis will identify potential vulnerabilities and areas for improvement. Subsequently, the project will focus on the development and implementation of advanced security measures such as multi-factor authentication, encryption techniques, and real-time fraud detection algorithms.

Additionally, user-centric design principles will be employed to create intuitive and user-friendly interfaces that enhance the overall banking experience.

The effectiveness of the proposed solutions will be evaluated through rigorous testing procedures, including simulated cyber-attacks and user feedback surveys. Furthermore, the project will explore the integration of emerging technologies such as biometrics and blockchain to further enhance security and streamline transactions.

By the conclusion of this project, we anticipate significant advancements in the security and usability of online banking systems. These advancements will not only bolster consumer trust and confidence but also contribute to the continued growth and evolution of digital banking services in an increasingly interconnected world.

Chapter 2

Objectives

Here are a few objectives for your E-Banking System project:

1. To develop a secure and user-friendly online banking platform that allows users to perform various banking transactions, such as account creation, fund transfers, and transaction history viewing.
2. To implement robust security measures to protect user data and prevent unauthorized access to accounts.
3. To provide an intuitive user interface that enhances the overall user experience and simplifies the process of managing finances online.
4. To enable real-time monitoring of transactions and account activities for both users and administrators.
5. To facilitate easy integration with existing banking systems and databases for seamless data management.
6. To ensure compatibility with various devices and platforms, including web and mobile applications, to provide users with flexibility and convenience in accessing their accounts.
7. To conduct thorough testing and validation of the system to ensure its reliability, performance, and security.
8. To gather user feedback and make necessary improvements to enhance the system's functionality and usability.
9. To explore the potential for future enhancements, such as the integration of emerging technologies like artificial intelligence and machine learning for fraud detection and personalized banking experiences.

Chapter 3

Software & Modules Used

1. **Deno:** Deno is a runtime for JavaScript and TypeScript, which is used to run the server-side code. It is a secure runtime that allows you to run JavaScript and TypeScript code outside of a web browser. It also serves a built-in package manager and supports ES modules.
2. **Express.js:** Express.js is a web application framework for Node.js, designed for building web applications and APIs. It provides a robust set of features for web and mobile applications, including routing, middleware support, and template rendering.
3. **MySQL:** MySQL is an open-source relational database management system. It is widely used for web applications and is known for its reliability and performance. MySQL allows you to store and retrieve data efficiently, and it supports SQL (Structured Query Language) for querying and managing databases. In this project, MySQL is used to store user information, transaction details, and other relevant data. This project uses the MariaDB version of MySQL, which is an open-source fork of MySQL that is fully compatible with it.
4. **EJS (Embedded JavaScript):** EJS is a simple templating language that lets you generate HTML markup with plain JavaScript. It is used to render dynamic content on the server side and send it to the client. EJS allows you to include JavaScript code within your HTML, making it easy to create dynamic web pages.
5. **Git:** Git is a distributed version control system that allows you to track changes in your codebase. It is widely used for source code management in software development. Git allows multiple developers to work on the same project simultaneously without conflicts. It provides features like branching, merging, and version history, making it easier to collaborate on projects.
6. **GitHub:** GitHub is a web-based platform that uses Git for version control. It provides a user-friendly interface for managing Git repositories, making it easier to collaborate with other developers. GitHub also offers features like issue tracking, pull requests, and project management tools, making it a popular choice for open-source projects and team collaboration.
7. **Code Editor (Visual Studio Code):** Visual Studio Code (VS Code) is a lightweight but powerful source code editor that runs on your desktop. It is available for Win-

dows, macOS, and Linux. VS Code includes support for debugging, syntax highlighting, intelligent code completion, snippets, code refactoring, and embedded Git. It also has a rich ecosystem of extensions for additional features and functionality.

- Utilize Visual Studio Code as the code editor for project development and editing.

Installation Steps:

1. **Install Deno:** Follow the official Deno installation guide to set up Deno on your system. You can find the installation instructions at <https://deno.land/#installation>.
2. **Install Git:** Download and install Git from the official website: <https://git-scm.com/downloads>. Follow the instructions for your operating system.
3. **Project Setup:** Clone the project repository from GitHub using the following command:

```
1 git clone https://github.com/Atan-D-RP4/wt_project
```

Navigate to the project directory:

```
1 cd wt_project/express
```

Now, simply run the following command to start the server:

```
1 deno run dev
```

This will install all the required dependencies and start the server.

Chapter 4

Structure of the Project

4.1 Overview

The project structure is organized into several directories and files, each serving a specific purpose. Below is a brief overview of the project structure:

- **main.ts:** The main entry point of the application, where the server is initialized and configured.
- **init-db.ts:** A script to initialize the database and create necessary tables.
- **database.ts:** Contains the database connection logic and configuration settings.
- **public/:** Contains static files such as CSS and JavaScript files for the frontend.
- **models/:** Contains TypeScript files defining the data models for the application, such as user, account, and transaction models.
- **controllers/:** Contains TypeScript files that handle the business logic and interact with the models.
- **middlewares/:** Contains middleware functions for authentication and logging.
- **routes/:** Contains route definitions for handling HTTP requests.
- **views/:** Contains HTML templates for rendering dynamic content using EJS.
- **deno.json:** Configuration file for Deno, specifying dependencies and runtime options.
- **deno.lock:** Lock file for Deno, ensuring consistent dependency versions.

4.2 Project File Structure

Figure 4.1: Project File Structure

```
project-root
├── main.ts
├── database.ts
├── init-db.ts
├── public
│   ├── css
│   └── js
├── models
│   ├── transaction.ts
│   ├── account.ts
│   └── user.ts
├── controllers
│   ├── transactionController.ts
│   ├── accountController.ts
│   ├── userController.ts
│   └── databaseController.ts
├── middlewares
│   ├── authMiddleware.ts
│   └── loggerMiddleware.ts
├── routes
│   ├── transactions.ts
│   ├── accounts.ts
│   └── auth.ts
├── views
│   ├── dashboard.html
│   ├── login.html
│   ├── register.html
│   └── transfer.html
├── deno.json
└── deno.lock
```

Chapter 5

Code Listings

5.1 Entry Point: main.ts

```
1 // File: main.ts
2 // @ts-types="npm:@types/express"
3 import express from "express";
4 import session from "express-session";
5 import cors from "cors";
6 // Routes
7 import accountRoutes from "../routes/accounts.ts";
8 import authRoutes from "../routes/auth.ts";
9 import dashboardRoutes from "../routes/dashboard.ts";
10 import transactionRoutes from "../routes/transactions.ts";
11 // Auth
12 import { authMiddleware } from "../middleware/auth.ts";
13
14 // Learn more at https://docs.deno.com/runtime/manual/examples/module\_metadata#concepts
15 if (import.meta.main) {
16   const app = express();
17   const PORT = 3000;
18
19   // Middleware
20   app.use(express.json());
21   app.use(express.urlencoded({ extended: true }));
22   app.use(session({
23     secret: "yourSecretKey", // Replace with a strong secret key
24     resave: false,
25     saveUninitialized: true,
26     cookie: {
27       secure: false,
28       httpOnly: true,
29       maxAge: 1000 * 60 * 60 * 4, // 4 hours
30     }, // Set to true in production with HTTPS
31   }));
32   app.use(cors({
```

```
33     origin: "http://localhost:3000", // specific origin or '*'
      for all
34     methods: "GET,POST,PUT,DELETE",
35     credentials: true,
36   }));
37
38   // Static files and view engine
39   app.use(express.static("public"));
40   app.set("view engine", "ejs");
41
42   // Route setup
43   app.use("/auth", authRoutes);
44   app.use("/api/accounts", accountRoutes);
45   app.use("/api/transactions", transactionRoutes);
46   app.use("/api/dashboard", dashboardRoutes);
47
48   // Index page
49   app.get("/", (_req: express.Request, res: express.Response) =>
50     {
51       res.redirect("/dashboard");
52     });
53
54   // Login page
55   app.get("/login", (_req: express.Request, res: express.Response) => {
56     res.render("login");
57   });
58
59   app.get("/register", (_req: express.Request, res: express.
60     Response) => {
61     res.render("register");
62   });
63
64   // Protected routes
65   app.use(authMiddleware);
66   app.get("/dashboard", (req: express.Request, res: express.
67     Response) => {
68     res.render("dashboard", { user: req.user?.username });
69   });
70
71   app.get("/transfer", (_req: express.Request, res: express.
72     Response) => {
73     res.render("transfer");
74   });
75 }
```

5.2 Database Connection: database.ts

```
1
2 // File: database.ts
3 import { Client } from "https://deno.land/x/mysql@v2.12.1/mod.ts";
4
5 class Database {
6   private static instance: Database;
7   private client: Client;
8
9   private constructor() {
10    this.client = new Client();
11  }
12
13   public static async getInstance(): Promise<Database> {
14     if (!Database.instance) {
15       Database.instance = new Database();
16       await Database.instance.connect();
17     }
18     return Database.instance;
19   }
20
21   private async connect() {
22     await this.client.connect({
23       hostname: "127.0.0.1",
24       username: "root",
25       password: "password",
26       db: "project",
27       poolSize: 3,
28     });
29   }
30
31   public getClient(): Client {
32     return this.client;
33   }
34 }
35
36 export default await Database.getInstance();
```

5.3 Database Init-Schema: init-db.ts

```
1
2 // File: init-db.ts
3 import db from "../database.ts";
4
5 const client = db.getClient();
6 await client.execute('CREATE DATABASE IF NOT EXISTS project');
```

```
7 await client.execute('USE project');
8
9 await client.execute(
10   'CREATE TABLE IF NOT EXISTS users (
11     id VARCHAR(36) PRIMARY KEY,
12     fullName VARCHAR(100),
13     email VARCHAR(100),
14     phone VARCHAR(20),
15     address VARCHAR(100),
16     city VARCHAR(50),
17     state VARCHAR(50),
18     zipCode VARCHAR(10),
19     username VARCHAR(50),
20     password VARCHAR(100),
21     createdAt DATETIME,
22     accountType ENUM('checking', 'savings', 'both')
23   )',
24 );
25
26 await client.execute(
27   'CREATE TABLE IF NOT EXISTS accounts (
28     id VARCHAR(36) PRIMARY KEY,
29     userId VARCHAR(36),
30     type ENUM('checking', 'savings', 'both'),
31     balance DECIMAL(10, 2),
32     createdAt DATETIME,
33     FOREIGN KEY (userId) REFERENCES users(id)
34   )',
35 );
36
37 await client.execute(
38   'CREATE TABLE IF NOT EXISTS transactions (
39     id VARCHAR(36) PRIMARY KEY,
40     fromId VARCHAR(36),
41     toId VARCHAR(36),
42     type ENUM('deposit', 'withdrawal', 'transfer'),
43     amount DECIMAL(10, 2),
44     description TEXT,
45     balance DECIMAL(10, 2),
46     createdAt DATETIME,
47     FOREIGN KEY (fromId) REFERENCES accounts(id),
48     FOREIGN KEY (toId) REFERENCES accounts(id)
49   )',
50 );
51
52 console.log("Database initialized");
53 client.close();
```


5.4 Auth Middleware: auth.ts

```
1 // File: middleware/auth.ts
2 // @ts-types="npm:@types/express-session"
3 // @ts-types="npm:@types/express"
4 import { NextFunction, Request, Response } from "express";
5
6 // In your auth.ts middleware file
7 declare global {
8   namespace Express {
9     interface Request {
10       user?: {
11         id: string;
12         username: string;
13         email: string;
14       };
15     }
16   }
17 }
18
19 // Extend express-session
20 declare module "express-session" {
21   interface SessionData {
22     user?: {
23       id: string;
24       username: string;
25       email: string;
26     };
27   }
28 }
29
30 export const authMiddleware = (
31   req: Request,
32   res: Response,
33   next: NextFunction,
34 ) => {
35
36   // Check if user is logged in
37   if (!req.session?.user) {
38     // Send a message to the user and then redirect to login page
39     console.log("Redirect");
40     return res.status(300).redirect("/login");
41   }
42
43   // Pass the user object to the next middleware
44   req.user = req.session.user; // No need for casting now
45
46   console.log("Passed Authentication");
47   res.setHeader("X-Content-Type-Options", "nosniff");
48   res.setHeader("X-XSS-Protection", "1; mode=block");
```

```
49   res.setHeader("X-Frame-Options", "DENY");
50   next();
51 };
```

5.5 Auth Controller: authController.ts

```
1
2 // File: controllers/authController.ts
3 // @ts-types="npm:@types/bcryptjs"
4 // @ts-types="npm:@types/express"
5 import { Request, Response } from "express";
6
7 import { UserModel } from "../models/user.ts";
8 import { AccountModel } from "../models/account.ts";
9 import * as bcrypt from "bcryptjs";
10
11 export const authController = {
12   register: async (req: Request, res: Response) => {
13     console.log("Registering user...");
14     try {
15       const {
16         fullName,
17         email,
18         phone,
19         address,
20         city,
21         state,
22         zipCode,
23         username,
24         password,
25         accountType,
26       } = req.body;
27
28       // Validate input
29       if (!fullName || !email || !password || !username) {
30         return res.status(400).json({ error: "Missing required fields" });
31       }
32
33       // Check if user already exists
34       const existingUser = await UserModel.findByEmail(email);
35       if (existingUser) {
36         return res.status(400).json({ error: "User already exists" });
37       }
38       const salt = bcrypt.genSaltSync(10);
39       const hashedPassword = bcrypt.hashSync(password, salt);
40
41       // Create user
```

```
42     const user = await UserModel.create({
43         fullName,
44         email,
45         phone,
46         address,
47         city,
48         state,
49         zipCode,
50         username,
51         password: hashedPassword,
52         accountType,
53     });
54
55     // Create accounts based on accountType
56     await AccountModel.create({
57         userId: user.id,
58         type: accountType,
59         balance: 0,
60     });
61
62     // Return success response
63     res.status(201).json({
64         message: "Account created successfully",
65         user: { id: user.id, username: user.username, email: user
66             .email },
67     });
68     } catch (error) {
69         console.error("Registration error:", error);
70         res.status(500).json({ error: "Server error during
71             registration" });
72     }
73 },
74
75 login: async (req: Request, res: Response) => {
76     try {
77         console.log("Logging in user...");
78         const { username, password } = req.body;
79
80         // Find user
81         let user = await UserModel.findByUsername(username);
82         if (!user) {
83             user = await UserModel.findByEmail(username);
84             if (!user) {
85                 console.log("User not found");
86                 return res.status(401).json({ error: "Invalid
87                     credentials" });
88             }
89         }
90
91         // Verify password (should use proper comparison in real
92         implementation)
```

```
89     const isMatch = await bcrypt.compare(password, user.  
90         password);  
91     if (isMatch === false) {  
92         return res.status(401).json({ error: "Invalid credentials  
93             " });  
94     }  
95  
96     req.session.user = {  
97         id: user.id,  
98         username: user.username,  
99         email: user.email,  
100     };  
101  
102     console.log("Session ID:", req.session.id, "\nUser ID:",  
103         user.id);  
104  
105     res.status(200).json({  
106         message: "Login successful",  
107         user: { id: user.id, username: user.username, email: user  
108             .email },  
109     });  
110 } catch (error) {  
111     console.error("Login error:", error);  
112     res.status(500).json({ error: "Server error during login"  
113     });  
114 }  
115 },  
116  
117 logout: (req: Request, res: Response) => {  
118     // Destroy the session  
119     console.log(req.session);  
120     req.session.destroy((err: Error) => {  
121         if (err) {  
122             console.error("Logout error:", err);  
123             return res.status(500).json({ error: "Server error during  
124                 logout" });  
125         }  
126         // Clear the session cookie (using the default name "  
127             connect.sid")  
128         res.clearCookie("connect.sid");  
129         res.status(200).json({ message: "Logged out successfully"  
130             });  
131     });  
132     console.log("Session destroyed");  
133 },  
134 },  
135 };
```

5.6 Account Controller: accountController.ts

```
1
2 // File: controllers/accountController.ts
3 // @ts-types="npm:@types/node"
4 import { Request, Response } from "npm:express@^4.21.2";
5 import { AccountModel } from "../models/account.ts";
6
7 export const accountController = {
8   createAccount: async (req: Request, res: Response) => {
9     try {
10       const userId = (req as any).user.id; // From auth
11       middleware
12       const { type } = req.body;
13
14       console.log("Creating an account");
15       if (!type) {
16         console.log("Type are required");
17         return res.status(400).json({ error: "Name and type are
18           required" });
19       }
20
21       const account = await AccountModel.create({
22         userId,
23         type,
24         balance: 1000.0,
25       });
26       console.log("Account created:", account);
27
28       res.status(201).json({ account });
29     } catch (error) {
30       console.error("Create account error:", error);
31       res.status(500).json({
32         error: "Server error creating account",
33         details: (error as Error).message,
34       });
35     }
36   },
37   getAllAccounts: async (req: Request, res: Response) => {
38     try {
39       const userId = (req as any).user.id; // From auth
40       middleware
41       const accounts = await AccountModel.findByUserId(userId);
42
43       if (!accounts || accounts.length === 0) {
44         return res.status(200).json({ accounts: [] });
45       }
46
47       res.status(200).json({ accounts });
48     } catch (error) {
49       console.error("Get accounts error:", error);
50     }
51   }
52 }
```

```
48     res.status(500).json({
49       error: "Server error retrieving accounts",
50       details: (error as Error).message,
51     });
52   }
53 },
54
55 getAccount: async (req: Request, res: Response) => {
56   try {
57     const { accountId } = req.params;
58     const userId = (req as any).user.id; // From auth
59     middleware
60
61     if (!accountId) {
62       return res.status(400).json({ error: "Account ID is
63         required" });
64     }
65
66     const account = await AccountModel.findById(accountId);
67
68     // Check if account exists and belongs to user
69     if (!account || account.userId !== userId) {
70       return res.status(404).json({ error: "Account not found"
71         });
72     }
73
74     res.status(200).json({ account });
75   } catch (error) {
76     console.error("Get account error:", error);
77     res.status(500).json({
78       error: "Server error retrieving account",
79       details: (error as Error).message,
80     });
81   }
82 },
83
84 getTransactions: async (req: Request, res: Response) => {
85   try {
86     const { accountId } = req.params;
87     const userId = (req as any).user.id; // From auth
88     middleware
89
90     if (!accountId) {
91       return res.status(400).json({ error: "Account ID is
92         required" });
93     }
94
95     const account = await AccountModel.findById(accountId);
96
97     // Check if account exists and belongs to user
98     if (!account || account.userId !== userId) {
```

```
94     return res.status(404).json({ error: "Account not found"
95         });
96     }
97     const transactions = await AccountModel.findTransactions(
98         accountId);
99     res.status(200).json({ transactions });
100 } catch (error) {
101     console.error("Get account transactions error:", error);
102     res.status(500).json({
103         error: "Server error retrieving account transactions",
104         details: (error as Error).message,
105     });
106 }
107 },
108 };
```

5.7 Dashboard Controller: dashboardController.ts

```
1
2 // File: controllers/dashboardController.ts
3
4 import { Request, Response } from "express";
5 import { AccountModel } from "../models/account.ts";
6 import { TransactionModel } from "../models/transaction.ts";
7
8 export const dashboardController = {
9     getDashboardData: async (req: Request, res: Response) => {
10         console.log("Getting dashboard data...");
11         try {
12             // deno-lint-ignore no-explicit-any
13             const userId = (req as any).user.id; // from auth
14             // middleware
15
16             // Get all accounts for the user
17             const accounts = await AccountModel.findByUserId(userId);
18
19             // For each account, get recent transactions (limit to 5
20             // for demonstration)
21             const accountsWithTransactions = await Promise.all(
22                 accounts.map(async (account) => {
23                     const transactions = await TransactionModel.
24                         findById(
25                             account.id,
```

```
26         transactions: transactions.slice(0, 5), // recent 5
           transactions
27     };
28     }),
29 );
30
31     // Calculate a simple financial summary (this can be made
       more detailed)
32     let totalBalance = 0;
33     accounts.forEach((account) => {
34         totalBalance += account.balance;
35     });
36
37     res.status(200).json({
38         accounts: accountsWithTransactions,
39         summary: {
40             totalBalance,
41             monthlySpending: 1245.62, // Dummy values; compute as
               needed
42             monthlyIncome: 3850.0,
43             savingsGoal: 10000.0,
44             savingsProgress: Math.round((totalBalance / 10000.0) *
               100),
45             creditScore: 760,
46         },
47     });
48 } catch (error) {
49     console.error("Dashboard data error:", error);
50     res.status(500).json({ error: "Server error retrieving
       dashboard data" });
51 }
52 },
53 };
```

5.8 Dashboard Routes: dashboard.ts

```
1
2 // File: routes/dashboard.ts
3 import express from "express";
4 import { dashboardController } from "../controllers/
   dashboardController.ts";
5 import { authMiddleware } from "../middleware/auth.ts";
6
7 const router = express.Router();
8
9 router.use((req, res, next) => {
10     console.log("Authenticating dashboard");
11     authMiddleware(req, res, next);
12 });
```



```
13
14 // Dashboard data endpoint (dynamic data for UI)
15 router.get("/", async (req, res) => {
16   await dashboardController.getDashboardData(req, res);
17   if (res.statusCode === 500) {
18     res.redirect("/login");
19   }
20 });
21
22 export default router;
```

5.9 Auth Routes: auth.ts

```
1
2 // File: routes/auth.ts
3 // @ts-types="npm:@types/express"
4
5 import express from "express";
6 import { authController } from "../controllers/authController.ts";
7
8 const router = express.Router();
9
10 // Registration endpoint
11 router.post("/register", async (req: express.Request, res:
12   express.Response) => {
13   await authController.register(req, res);
14 });
15
16 // Login endpoint
17 router.post("/login", async (req: express.Request, res: express.
18   Response) => {
19   await authController.login(req, res);
20 });
21
22 // Logout endpoint
23 router.post("/logout", authController.logout);
24
25 export default router;
```

5.10 Transaction Controller: transactionController.ts

```
1
2 // File: controllers/transactionController.ts
3 // @ts-tpes="npm:@types/node"
4 import { Request, Response } from "npm:express@^4.21.2";
```

```
5 import { AccountModel } from "../models/account.ts";
6 import { TransactionModel } from "../models/transaction.ts";
7 import db from "../database.ts";
8
9 export const transactionController = {
10   getTransactions: async (req: Request, res: Response) => {
11     try {
12       const { accountId } = req.params;
13       // deno-lint-ignore no-explicit-any
14       const userId = (req as any).user.id; // From auth
15       middleware
16
17       if (!accountId) {
18         return res.status(400).json({ error: "Account ID is
19           required" });
20       }
21
22       const account = await AccountModel.findById(accountId);
23
24       // Check if account exists and belongs to user
25       if (!account || account.userId !== userId) {
26         return res.status(404).json({ error: "Account not found"
27           });
28       }
29
30       const transactions = await TransactionModel.findByAccountId
31         (accountId);
32
33       res.status(200).json({ transactions });
34     } catch (error) {
35       console.error("Get transactions error:", error);
36       res.status(500).json({
37         error: "Server error retrieving transactions",
38         details: (error as Error).message,
39       });
40     }
41   },
42
43   getTransactionHistory: async (req: Request, res: Response) => {
44     const client = db.getClient();
45     try {
46       const userId = (req as any).user.id;
47       const transactions = await client.execute(
48         `SELECT * FROM transactions
49         WHERE accountId IN (SELECT id FROM accounts WHERE userId =
50           ?)
51         AND type = 'transfer'`,
52         [userId],
53       );
54       res.json({ transactions });
55     } catch (error) {
```

```
51     // Handle error
52     console.error("Transaction history error:", error);
53   }
54 },
55
56 createTransaction: async (req: Request, res: Response) => {
57   const client = db.getClient();
58   try {
59     const { fromAccountId, toAccountId, amount, description } =
60       req.body;
61     // deno-lint-ignore no-explicit-any
62     const userId = (req as any).user.id; // From auth
63     middleware
64     const amountNum = Number(amount);
65
66     // Validate input
67     if (!fromAccountId || !toAccountId || !amount || amount <=
68       0) {
69       return res.status(400).json({ error: "Invalid transfer
70         details" });
71     }
72
73     if (fromAccountId === toAccountId) {
74       return res.status(400).json({ error: "Same Sender and
75         Receiver accounts" });
76     }
77
78     // Get sender account and check ownership
79     const senderAccount = await AccountModel.findById(
80       fromAccountId);
81     if (!senderAccount || senderAccount.userId !== userId) {
82       return res.status(404).json({ error: "Sender account not
83         found" });
84     }
85
86     // Get receiver account
87     const receiverAccount = await AccountModel.findById(
88       toAccountId);
89     if (!receiverAccount) {
90       return res.status(404).json({ error: "Receiver account
91         not found" });
92     }
93
94     // Check for sufficient funds
95     if (senderAccount.balance < amount) {
96       return res.status(400).json({
97         error: "Insufficient funds for transfer",
98       });
99     }
100   }
101 }
```

```
93     console.log("Creating transaction...");
94     console.log("From Account ID:", fromAccountId);
95     console.log("To Account ID:", toAccountId);
96     console.log("Amount:", amountNum);
97     console.log("Description:", description);
98     console.log("User ID:", userId);
99
100    client.execute("START TRANSACTION");
101    const newReceiverBalance = Number(receiverAccount.balance)
      + amountNum;
102    const newSenderBalance = Number(senderAccount.balance) -
      amountNum;
103    console.log(newSenderBalance);
104    console.log(newReceiverBalance);
105
106    // Update sender balance
107    await AccountModel.updateBalance(fromAccountId,
      newSenderBalance);
108
109    // Update receiver balance
110    await AccountModel.updateBalance(toAccountId,
      newReceiverBalance);
111
112    // Create a new transaction
113    const Transaction = await TransactionModel.create({
114      fromId: fromAccountId,
115      toId: toAccountId,
116      type: "transfer",
117      amount,
118      description,
119      balance: newSenderBalance,
120    });
121    client.execute("COMMIT");
122
123    res.status(201).json({
124      message: "Transfer completed successfully",
125      Transaction,
126      newSenderBalance,
127      newReceiverBalance,
128    });
129  } catch (error) {
130    client.execute("ROLLBACK");
131    console.error("Transfer error:", error);
132    res.status(500).json({ error: "Server error processing
      transfer" });
133  }
134 },
135 };
```

5.11 Account Routes: accounts.ts

```
1
2 // File: routes/accounts.ts
3 import express from "express";
4 import { authMiddleware } from "../middleware/auth.ts";
5 import { accountController } from "../controllers/
    accountController.ts";
6
7 const router = express.Router();
8
9 // Apply auth middleware to all account routes
10 router.use((req, res, next) => {
11     authMiddleware(req, res, next);
12 });
13
14 router.get("/", async (req, res) => {
15     await accountController.getAllAccounts(req, res);
16 });
17
18 router.get("/create", async (req, res) => {
19     await accountController.createAccount(req, res);
20 });
21
22 // Get account details
23 router.get("/:accountId", async (req, res) => {
24     await accountController.getAccount(req, res);
25 });
26
27 // Get account transactions
28 router.get("/:accountId/transactions", async (req, res) => {
29     await accountController.getTransactions(req, res);
30 });
31
32 export default router;
```

5.12 Transaction Routes: transactions.ts

```
1 // File: routes/transaction.ts
2 import express from "express";
3 import { authMiddleware } from "../middleware/auth.ts";
4 import { transactionController } from "../controllers/
    transactionController.ts";
5
6 const router = express.Router();
7
8 // Apply auth middleware to all transaction routes
9 router.use((req, res, next) => {
```

```
10 console.log("Authenticating new transaction");
11 authMiddleware(req, res, next);
12 });
13
14 router.post("/", async (req, res) => {
15   await transactionController.createTransaction(req, res);
16 });
17
18 router.post("/history", async (req, res) => {
19   await transactionController.getTransactionHistory(req, res);
20 });
21
22 router.post("/:accountId", async (req, res) => {
23   await transactionController.getTransactions(req, res);
24 });
25
26 export default router;
```

5.13 User Model: user.ts

```
1
2 // File: models/user.ts
3 // This is a simple in-memory model for demonstration
4 // In a real application, you'd use a database
5
6 import { AccountModel, AccountType } from "../account.ts";
7 import db from "../database.ts";
8
9 interface User {
10   id: string;
11   fullName: string;
12   email: string;
13   phone: string;
14   address: string;
15   city: string;
16   state: string;
17   zipCode: string;
18   username: string;
19   password: string;
20   createdAt: Date;
21   accountType: AccountType;
22 }
23
24 const client = db.getClient();
25
26 export const UserModel = {
27   create: async (userData: Omit<User, "id" | "createdAt">):
28     Promise<User> => {
29     // Check if user already exists
```

```
29     const users = await client.execute(  
30         "SELECT * FROM users WHERE email = ? or username = ?",  
31         [  
32             userData.email,  
33             userData.username,  
34         ],  
35     );  
36  
37     if (users.rows == undefined) {  
38         console.log("SQL Query Error");  
39         throw new Error("SQL Query Error");  
40     }  
41     if (users.rows.length > 0) {  
42         console.log("User already exists");  
43         return users.rows[0] as User;  
44     }  
45  
46     let id = 0;  
47     const count = await client.query(  
48         "SELECT COUNT(*) FROM users",  
49     );  
50     if (count.rows !== undefined) {  
51         id = count.rows[0]["COUNT(*)"] + 1;  
52     }  
53     const newUser: User = {  
54         ...userData,  
55         accountType: userData.accountType as AccountType,  
56         id: String(id),  
57         createdAt: new Date(),  
58     };  
59  
60     await client.execute(  
61         "INSERT INTO users (id, fullName, email, phone, address,  
62             city, state, zipCode, username, password, createdAt,  
63             accountType) \  
64         VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)",  
65         [  
66             newUser.id,  
67             newUser.fullName,  
68             newUser.email,  
69             newUser.phone,  
70             newUser.address,  
71             newUser.city,  
72             newUser.state,  
73             newUser.zipCode,  
74             newUser.username,  
75             newUser.password,  
76             newUser.createdAt,  
77             newUser.accountType,  
78         ],  
79     );
```

```
78
79 // Add an account for the user
80 await AccountModel.create({
81   userId: newUser.id,
82   type: newUser.accountType,
83   balance: 1000.0,
84 });
85
86 return newUser;
87 },
88
89 findByEmail: async (email: string): Promise<User | undefined>
90   => {
91     const users = await client.execute("SELECT * FROM users WHERE
92       email = ?", [
93       email,
94     ]);
95     if (users.rows == undefined) {
96       return undefined;
97     }
98     return users.rows.length > 0 ? users.rows[0] : undefined;
99   },
100
101 findByUsername: async (username: string): Promise<User |
102   undefined> => {
103     const users = await client.execute(
104       "SELECT * FROM users WHERE username = ?",
105       [
106       username,
107     ],
108   );
109   if (users.rows == undefined) {
110     return undefined;
111   }
112   return users.rows[0] as User;
113 },
114
115 findById: async (id: string): Promise<User | undefined> => {
116   const users = await client.execute("SELECT * FROM users WHERE
117     id = ?", [
118     id,
119   ]);
120   if (users.rows == undefined) {
121     return undefined;
122   }
123   return users.rows[0] as User;
124 },
125 };
```


5.14 Transaction Model: transaction.ts

```

1
2 // File: models/transaction.ts
3
4 import db from "../database.ts";
5
6 export enum TransactionType {
7   Deposit = "deposit",
8   Withdrawal = "withdrawal",
9   Transfer = "transfer",
10 }
11
12 export interface Transaction {
13   id: string;
14   fromId: string;
15   toId: string;
16   type: "deposit" | "withdrawal" | "transfer";
17   amount: number;
18   description: string;
19   balance: number;
20   createdAt: Date;
21 }
22
23 // In-memory storage
24 const client = db.getClient();
25
26 export const TransactionModel = {
27   create: async (
28     transactionData: Omit<Transaction, "id" | "createdAt">,
29   ): Promise<Transaction> => {
30     let id = 0;
31     const count = await client.execute(
32       "SELECT COUNT(*) FROM transactions",
33     );
34     if (count.rows !== undefined) {
35       id = count.rows[0]["COUNT(*)"] + 1;
36     }
37
38     const newTransaction: Transaction = {
39       ...transactionData,
40       createdAt: new Date(),
41       id: String(id),
42     };
43
44     await client.execute(
45       "INSERT INTO transactions (id, toId, fromId, type, amount,
46         description, balance, createdAt) VALUES (?, ?, ?, ?, ?,
47         ?, ?, ?)",
48     );
49   },
50 };

```

```
47     newTransaction.id,  
48     newTransaction.fromId,  
49     newTransaction.toId,  
50     newTransaction.type,  
51     newTransaction.amount,  
52     newTransaction.description,  
53     newTransaction.balance,  
54     newTransaction.createdAt,  
55   ],  
56   );  
57   return newTransaction;  
58 },  
59  
60 findById: async (id: string): Promise<Transaction | undefined>  
61   => {  
62     const transactions = await client.execute(  
63       "SELECT * FROM transactions WHERE id = ?",  
64       [id],  
65     );  
66     if (transactions.rows == undefined) {  
67       throw new Error("Account not found");  
68     }  
69     return transactions.rows.length > 0 ? transactions.rows[0] :  
70       undefined;  
71   },  
72  
73 findById: async (userId: string): Promise<Transaction[]> =>  
74   {  
75     const transactions = await client.execute(  
76       "SELECT * FROM transactions WHERE userId = ?",  
77       [userId],  
78     );  
79     if (transactions.rows == undefined) {  
80       throw new Error("Account not found");  
81     }  
82     return transactions.rows as Transaction[];  
83   },  
84  
85 findById: async (accountId: string): Promise<Transaction  
86   []> => {  
87     const transactions = await client.execute(  
88       "SELECT * FROM transactions WHERE fromId = ?",  
89       [accountId],  
90     );  
91     if (transactions.rows == undefined) {  
92       throw new Error("Account not found");  
93     }  
94   }
```

```
94     return transactions.rows as Transaction[];
95   },
96 };
```

5.15 Account Model: account.ts

```
1
2 // File: models/account.ts
3
4 import db from "../database.ts";
5 import { Transaction } from "../transaction.ts";
6
7 export enum AccountType {
8   Checking = "checking",
9   Savings = "savings",
10  Both = "both",
11 }
12
13 export interface Account {
14   id: string;
15   userId: string;
16   type: AccountType;
17   balance: number;
18   createdAt: Date;
19 }
20
21 const client = db.getClient();
22
23 export const AccountModel = {
24   create: async (
25     accountData: Omit<Account, "id" | "createdAt">,
26   ): Promise<Account | undefined> => {
27     try {
28       let id = 0;
29       const count = await client.execute(
30         "SELECT COUNT(*) FROM accounts",
31       );
32       if (count.rows !== undefined) {
33         id = count.rows[0]["COUNT(*)"] + 1;
34       }
35
36       const newAccount: Account = {
37         ...accountData,
38         id: String(id),
39         createdAt: new Date(),
40       };
41
42       // Check if user already exists
43       const accounts = await client.execute(
```

```
44     "SELECT * FROM accounts WHERE id = ?",
45     [newAccount.id],
46   );
47
48   if (accounts.rows == undefined) {
49     console.log("SQL Query Error");
50     return newAccount;
51   }
52   if (accounts.rows.length > 0) {
53     console.log("Account already exists");
54     return accounts.rows[0] as Account;
55   }
56
57   await client.execute(
58     "INSERT INTO accounts (id, userId, type, balance,
59       createdAt) VALUES (?, ?, ?, ?, ?)",
60     [
61       newAccount.id,
62       newAccount.userId,
63       newAccount.type,
64       newAccount.balance,
65       newAccount.createdAt,
66     ],
67   );
68   console.log("Account created");
69
70   return newAccount;
71 } catch (error) {
72   console.error("Account Creation Error: ", (error as Error).
73     message);
74   throw new Error("Server error creating account");
75 }
76 },
77
78 listAll: async (): Promise<Account[]> => {
79   const accounts = await client.execute("SELECT * FROM accounts
80     ");
81
82   if (accounts.rows == undefined) {
83     return [];
84   }
85
86   return accounts.rows as Account[];
87 },
88
89 findById: async (id: string): Promise<Account | undefined> => {
90   const accounts = await client.execute(
91     "SELECT * FROM accounts WHERE id = ?",
```

```
92     if (accounts.rows == undefined) {
93         throw new Error("Account not found");
94     }
95
96     return accounts.rows.length > 0 ? accounts.rows[0] :
97         undefined;
98 },
99 findByUserId: async (userId: string): Promise<Account[]> => {
100     const accounts = await client.execute(
101         "SELECT * FROM accounts WHERE userId = ?",
102         [userId],
103     );
104
105     if (accounts.rows == undefined) {
106         return [];
107     }
108
109     return accounts.rows as Account[];
110 },
111
112 findTransactions: async (id: string): Promise<Transaction[]> =>
113     {
114         const transactions = await client.execute(
115             "SELECT * FROM transactions WHERE fromId = ?",
116             [id],
117         );
118
119         if (transactions.rows == undefined) {
120             return [];
121         }
122
123         return transactions.rows as Transaction[];
124 },
125
126 updateBalance: async (
127     id: string,
128     newBalance: number,
129 ): Promise<Account | undefined> => {
130     console.log(
131         "UPDATE accounts SET balance = " + newBalance + " WHERE id
132         = " + id,
133     );
134     await client.execute(
135         "UPDATE accounts SET balance = ? WHERE id = ?",
136         [newBalance, id],
137     );
138     return AccountModel.findById(id);
139 },
140 };
```

5.16 Dashboard View: dashboard.ejs

```

1
2 <!--// File: views/dashboard.ejs-->
3 <!DOCTYPE html>
4 <html lang="en">
5
6 <head>
7     <meta charset="UTF-8">
8     <meta name="viewport" content="width=device-width, initial-
9         scale=1.0">
10    <title>E(xpress)-Bank Dashboard</title>
11    <!-- Bootstrap CSS from CDN -->
12    <link href="https://cdnjs.cloudflare.com/ajax/libs/bootstrap
13        /5.3.0/css/bootstrap.min.css" rel="stylesheet">
14    <!-- Font Awesome for icons -->
15    <link href="https://cdnjs.cloudflare.com/ajax/libs/font-
16        awesome/6.4.0/css/all.min.css" rel="stylesheet">
17    <!-- Custom CSS -->
18    <link rel="stylesheet" href="/css/dashboard.css" id='
19        dashboard.css'>
20    <link rel="stylesheet" href="/css/styles.css" id='dashboard.
21        css'>
22 </head>
23
24 <body>
25     <!-- Sidebar -->
26     <div class="sidebar d-none d-lg-block">
27         <div class="bank-logo">E(xpress)-Bank</div>
28         <ul class="nav flex-column">
29             <li class="nav-item">
30                 <a class="nav-link active" href="#"><i class="fas
31                     fa-home"></i> Dashboard</a>
32             </li>
33             <li class="nav-item">
34                 <a class="nav-link" href="/transfer"><i class="
35                     fas fa-exchange-alt"></i> Transfers</a>
36             </li>
37             <li class="nav-item">
38                 <a class="nav-link" href="#"><i class="fas fa-cog
39                     "></i> Settings</a>
40             </li>
41             <li class="nav-item mt-5">
42                 <a class="nav-link text-danger" id="logoutButton"
43                     > <i class="fas fa-sign-out-alt" onclick="
44                         logout()"></i> Logout</a>
45             </li>

```

```

36     </ul>
37 </div>
38
39 <!-- Main Content -->
40 <div class="main-content">
41     <!-- Header with user greeting -->
42     <div class="d-flex justify-content-between align-items-
43         center mb-4">
44         <div class="user-greeting">
45             Welcome back, <span class="user-name" id="
46                 userName"><%- user %></span>!
47         </div>
48         <a href="/details" class="profile-button" id="
49             profileInitials">U</a>
50     </div>
51
52     <!-- Accounts Overview Section -->
53     <h5 class="mb-4">Your Accounts</h5>
54     <div class="row" id="accountsContainer"> </div>
55
56     <!-- Quick Stats Section -->
57     <!-- <h5 class="mb-4 mt-4">Financial Summary</h5> -->
58     <!-- <div class="row"> -->
59     <!-- Monthly Spending -->
60     <!-- <div class="col-md-6 col-lg-3 mb-4"> -->
61     <!-- <div class="stats-card"> -->
62     <!-- <div class="stat-label">Monthly Spending
63     </div> -->
64     <!-- <div class="stat-value">$1,245.62</div>
65     -->
66     <!-- <div class="text-muted small"> -->
67     <!-- <i class="fas fa-arrow-down text-
68         success"></i> 12% from last month -->
69     <!-- </div> -->
70     <!-- </div> -->
71     <!-- Income -->
72     <!-- <div class="col-md-6 col-lg-3 mb-4"> -->
73     <!-- <div class="stats-card"> -->
74     <!-- <div class="stat-label">Monthly Income</
75         div> -->
76     <!-- <div class="stat-value">$3,850.00</div>
77     -->
78     <!-- <div class="text-muted small"> -->
79     <!-- <i class="fas fa-arrow-up text-
80         success"></i> 5% from last month -->
81     <!-- </div> -->
82     <!-- </div> -->
83     <!-- </div> -->
84     <!-- </div> -->

```

```

78         <!-- Savings Goal -->
79         <!--         <div class="col-md-6 col-lg-3 mb-4"> -->
80         <!--             <div class="stats-card"> -->
81         <!--                 <div class="stat-label">Savings Goal</
            div> -->
82         <!--                     <div class="stat-value">$10,000.00</div>
            -->
83         <!--                         <div class="text-muted small">78%
            complete</div> -->
84         <!--                             <div class="progress savings-progress">
            -->
85         <!--                                 <div class="progress-bar bg-success"
            role="progressbar" style="width: 78%" aria-valuenow="78"
            -->
86         <!--                                     aria-valuemin="0" aria-valuemax=
            "100"></div> -->
87         <!--                                         </div> -->
88         <!--                                             </div> -->
89         <!--                                                 </div> -->
90         <!------->
91         <!-- Credit Score -->
92         <!--         <div class="col-md-6 col-lg-3 mb-4"> -->
93         <!--             <div class="stats-card"> -->
94         <!--                 <div class="stat-label">Credit Score</
            div> -->
95         <!--                     <div class="stat-value">760</div> -->
96         <!--                         <div class="text-muted small"> -->
97         <!--                             <i class="fas fa-arrow-up text-
            success"></i> Excellent -->
98         <!--                                     </div> -->
99         <!--                                         </div> -->
100        <!--                                             </div> -->
101        <!--                                                 </div> -->
102        <!------->
103        <!-- Recent Transactions Section -->
104        <!--         <h5 class="mb-4">Recent Transactions</h5> -->
105        <!--             <div class="transaction-list"> -->
106        <!--                 <div class="list-group list-group-flush" id="
            transactionsList"> -->
107        <!--                     <!-- Transaction items will be populated
            dynamically -->
108        <!--                         <div class="transaction-item deposit"> -->
109        <!--                             <div class="d-flex justify-content-
            between align-items-center"> -->
110        <!--                                 <div> -->
111        <!--                                     <div class="fw-bold">Deposit:
            Payroll</div> -->
112        <!--                                         <div class="text-muted small">
            Feb 25, 2025</div> -->
113        <!--                                             </div> -->

```



```

114         <!--                <div class="transaction-amount
deposit">+$1,850.00</div> -->
115         <!--                </div> -->
116         <!--                </div> -->
117         <!--                <div class="transaction-item withdrawal"> --
>
118         <!--                <div class="d-flex justify-content-
between align-items-center"> -->
119         <!--                <div> -->
120         <!--                <div class="fw-bold">Walmart</
div> -->
121         <!--                <div class="text-muted small">
Feb 23, 2025</div> -->
122         <!--                </div> -->
123         <!--                <div class="transaction-amount
withdrawal">-$87.45</div> -->
124         <!--                </div> -->
125         <!--                </div> -->
126         <!--                <div class="transaction-item withdrawal"> --
>
127         <!--                <div class="d-flex justify-content-
between align-items-center"> -->
128         <!--                <div> -->
129         <!--                <div class="fw-bold">Amazon
Prime</div> -->
130         <!--                <div class="text-muted small">
Feb 21, 2025</div> -->
131         <!--                </div> -->
132         <!--                <div class="transaction-amount
withdrawal">-$14.99</div> -->
133         <!--                </div> -->
134         <!--                </div> -->
135         <!--                <div class="transaction-item withdrawal"> --
>
136         <!--                <div class="d-flex justify-content-
between align-items-center"> -->
137         <!--                <div> -->
138         <!--                <div class="fw-bold">Uber Ride</
div> -->
139         <!--                <div class="text-muted small">
Feb 20, 2025</div> -->
140         <!--                </div> -->
141         <!--                <div class="transaction-amount
withdrawal">-$24.50</div> -->
142         <!--                </div> -->
143         <!--                </div> -->
144         <!--                <div class="transaction-item deposit"> -->
145         <!--                <div class="d-flex justify-content-
between align-items-center"> -->
146         <!--                <div> -->

```

```

147      <!--                                <div class="fw-bold">Transfer
      from Savings</div> -->
148      <!--                                <div class="text-muted small">
      Feb 18, 2025</div> -->
149      <!--                                </div> -->
150      <!--                                <div class="transaction-amount
      deposit">+$500.00</div> -->
151      <!--                                </div> -->
152      <!--                                </div> -->
153      <!--                                </div> -->
154      <!--                                <div class="p-3 text-center"> -->
155      <!--                                <button class="btn btn-link text-decoration-
      none">View All Transactions</button> -->
156      <!--                                </div> -->
157      <!--                                </div> -->
158      <!-- </div> -->
159
160      <!-- Quick Action Button -->
161      <div class="quick-actions">
162          <div class="dropdown">
163              <button class="quick-action-btn" type="button" id="
      quickActionsDropdown" data-bs-toggle="dropdown"
164                  aria-expanded="false">
165                  <i class="fas fa-plus"></i>
166              </button>
167              <ul class="dropdown-menu dropdown-menu-end" aria-
      labelledby="quickActionsDropdown">
168                  <li><a class="dropdown-item" href="#"><i class="
      fas fa-exchange-alt me-2"></i> New Transfer</a>
169                  </li>
170                  <li><a class="dropdown-item" href="#"><i class="
      fas fa-credit-card me-2"></i> Pay Bill</a></li>
171                  <li><a class="dropdown-item" href="#"><i class="
      fas fa-mobile-alt me-2"></i> Mobile Deposit</a>
172                  </li>
173                  <li><a class="dropdown-item" href="#"><i class="
      fas fa-user-friends me-2"></i> Send to Friend<
174                  /a></li>
175              </ul>
176          </div>
177      </div>
178
179      <!-- Bootstrap JS -->
180      <script src="https://cdnjs.cloudflare.com/ajax/libs/bootstrap
      /5.3.0/js/bootstrap.bundle.min.js"></script>
181
182      <!-- Dashboard Script -->
183      <script src="/js/dashboard.cjs"></script>
184      <script>
185          // Logout functionality

```

```

183     async function logout() {
184         console.log("Logging out...");
185         try {
186             const response = await fetch("/auth/logout", {
187                 method: "POST",
188                 headers: {
189                     "Content-Type": "application/json",
190                 },
191             });
192             console.log("Logout response:", response);
193
194             if (response.ok) {
195                 // Redirect to the login page after
196                 // successful logout
197                 globalThis.window.location.href = "/login";
198             } else {
199                 console.error("Logout failed");
200             }
201         } catch (error) {
202             console.error("Error during logout:", error);
203             globalThis.window.location.href = "/login";
204         }
205     }
206 </script>
207 </body>
208
209 </html>

```

5.17 Login View: login.ejs

```

1
2 <!--// File: views/login.ejs-->
3 <!DOCTYPE html>
4 <html lang="en">
5
6 <head>
7     <meta charset="UTF-8">
8     <meta name="viewport" content="width=device-width, initial-
9         scale=1.0">
10    <title>E(xpress)-Bank Login</title>
11    <!-- Bootstrap CSS from CDN -->
12    <link href="https://cdnjs.cloudflare.com/ajax/libs/bootstrap
13        /5.3.0/css/bootstrap.min.css" rel="stylesheet">
14    <link href="/css/styles.css" rel="stylesheet">
15 </head>
16
17 <body class="bg-light">
18     <div class="container py-5">

```

```

17     <div class="card shadow form-container">
18         <div class="bank-header">
19             <div class="bank-logo">E(xpress)-Bank</div>
20             <h2 class="mt-2">Log In</h2>
21             <p class="text-muted">Access your secure banking
                account</p>
22         </div>
23
24         <form id="loginForm">
25             <div class="mb-3">
26                 <label for="username" class="form-label">
                    Username</label>
27                 <input type="text" class="form-control" id="
                    username" required>
28             </div>
29
30             <div class="mb-3">
31                 <label for="password" class="form-label">
                    Password</label>
32                 <input type="password" class="form-control"
                    id="password" required>
33             </div>
34
35             <div class="mb-3 form-check">
36                 <input type="checkbox" class="form-check-
                    input" id="rememberMe">
37                 <label class="form-check-label" for="
                    rememberMe">Remember me</label>
38             </div>
39
40             <button type="submit" class="btn btn-primary w
                -100">Log In</button>
41
42             <div class="text-center mt-3">
43                 <!-- <p><a href="#" class="text-decoration-
                    none">Forgot password?</a></p> -->
44                 <p>Don't have an account? <a href="/register"
                    class="text-decoration-none">Sign up</a><
                    /p>
45             </div>
46         </form>
47     </div>
48 </div>
49
50 <!-- Bootstrap JS -->
51 <script src="https://cdnjs.cloudflare.com/ajax/libs/bootstrap
    /5.3.0/js/bootstrap.bundle.min.js"></script>
52
53 <!-- Login Script -->
54 <script lang="javascript">

```

```

55     document.getElementById('loginForm').addEventListener('
        submit', async function (e) {
56         e.preventDefault();
57
58         const username = document.getElementById('username').
            value;
59         const password = document.getElementById('password').
            value;
60
61         try {
62             const response = await fetch('/auth/login', {
63                 method: 'POST',
64                 headers: {
65                     'Content-Type': 'application/json'
66                 },
67                 body: JSON.stringify({username, password})
68             });
69
70             const data = await response.json();
71
72             if (response.ok) {
73                 // Session cookie is set automatically, so
                    simply redirect to dashboard
74                 window.location.href = '/dashboard';
75             } else {
76                 alert(data.error || 'Login failed');
77             }
78         } catch (error) {
79             console.error('Login error:', error);
80             alert('An error occurred during login');
81         }
82     });
83 </script>
84
85 </body>
86
87 </html>

```

5.18 Register View: register.ejs

```

1
2 <!--// File: views/register.ejs-->
3 <!DOCTYPE html>
4 <html lang="en">
5
6 <head>
7     <meta charset="UTF-8">
8     <meta name="viewport" content="width=device-width, initial-
        scale=1.0">

```

```

9      <title>E(xpress)-Bank</title>
10     <!-- Bootstrap CSS from CDN -->
11     <link href="https://cdnjs.cloudflare.com/ajax/libs/bootstrap
      /5.3.0/css/bootstrap.min.css" rel="stylesheet">
12     <link href="/css/styles.css" rel="stylesheet">
13 </head>
14
15 <body class="bg-light">
16     <div class="container py-5">
17         <div class="card shadow form-container">
18             <div class="bank-header text-center">
19                 <div class="bank-logo">E(xpress)-Bank</div>
20                 <h2 class="mt-2">Create an Account</h2>
21                 <p class="text-muted">Join our secure banking
      platform</p>
22             </div>
23             <form id="registrationForm">
24                 <!-- Personal Information -->
25                 <fieldset class="mb-4">
26                     <legend>Personal Information</legend>
27                     <div class="mb-3">
28                         <label for="fullName" class="form-label">
      Full Name</label>
29                         <input type="text" class="form-control"
      id="fullName" required>
30                     </div>
31                     <div class="row mb-3">
32                         <div class="col">
33                             <label for="email" class="form-label">
      >Email</label>
34                             <input type="email" class="form-
      control" id="email" required>
35                         </div>
36                         <div class="col">
37                             <label for="phone" class="form-label">
      >Phone Number</label>
38                             <input type="tel" class="form-control
      " id="phone" required>
39                         </div>
40                     </div>
41                 </fieldset>
42                 <!-- Address Information -->
43                 <fieldset class="mb-4">
44                     <legend>Address Information</legend>
45                     <div class="mb-3">
46                         <label for="address" class="form-label">
      Address</label>
47                         <input type="text" class="form-control"
      id="address" required>
48                     </div>
49                     <div class="row mb-3">

```

```

50         <div class="col">
51             <label for="city" class="form-label">
52                 City</label>
53             <input type="text" class="form-
54                 control" id="city" required>
55             </div>
56         <div class="col">
57             <label for="state" class="form-label">
58                 State</label>
59             <input type="text" class="form-
60                 control" id="state" required>
61             </div>
62         </div>
63     </fieldset>
64     <!-- Account Security -->
65     <fieldset class="mb-4">
66         <legend>Account Security</legend>
67         <div class="mb-3">
68             <label for="username" class="form-label">
69                 Username</label>
70             <input type="text" class="form-control"
71                 id="username" required>
72             </div>
73         <div class="mb-3">
74             <label for="password" class="form-label">
75                 Password</label>
76             <input type="password" class="form-
77                 control" id="password" required>
78             <div class="password-requirements mt-1">
79                 Password must be at least 8
80                 characters and include uppercase,
81                 lowercase, number, and special
82                 character.
83             </div>
84         </div>
85         <div class="mb-3">
86             <label for="confirmPassword" class="form-
87                 label">Confirm Password</label>
88             <input type="password" class="form-
89                 control" id="confirmPassword" required
90             >
91         </div>
92     </fieldset>
93     <!-- Account Type Selection -->
94     <fieldset class="mb-4">

```

```

85         <legend>Account Type</legend>
86         <div class="form-check">
87             <input class="form-check-input" type="
                radio" name="accountType" id="
                checkingAccount" value="checking"
                checked>
88             <label class="form-check-label" for="
                checkingAccount">Checking Account</
                label>
89         </div>
90         <div class="form-check">
91             <input class="form-check-input" type="
                radio" name="accountType" id="
                savingsAccount" value="savings">
92             <label class="form-check-label" for="
                savingsAccount">Savings Account</label
                >
93         </div>
94         <div class="form-check">
95             <input class="form-check-input" type="
                radio" name="accountType" id="
                bothAccounts" value="both">
96             <label class="form-check-label" for="
                bothAccounts">Both Checking & Savings<
                /label>
97         </div>
98     </fieldset>
99     <!-- Terms and Conditions -->
100     <div class="mb-4">
101         <div class="form-check">
102             <input class="form-check-input" type="
                checkbox" id="termsAgreement" required
                >
103             <label class="form-check-label" for="
                termsAgreement">
104                 I agree to the <a href="#" data-bs-
                    toggle="modal" data-bs-target="#
                    termsModal">Terms and Conditions</
                    a>
105             </label>
106         </div>
107     </div>
108     <!-- Submit Button -->
109     <button type="submit" class="btn btn-primary w
        -100">Create Account</button>
110     <div class="text-center mt-3">
111         <p>Already have an account? <a href="/login">
            Log in</a></p>
112     </div>
113 </form>
114 </div>

```



```

115     </div>
116     <!-- Bootstrap JS from CDN -->
117     <script src="https://cdnjs.cloudflare.com/ajax/libs/bootstrap
118         /5.3.0/js/bootstrap.bundle.min.js"></script>
119     <!-- Custom JS -->
120     <script src="/js/registration.cjs"></script>
121 </body>
122 </html>

```

5.19 Transfers View: transfers.ejs

```

1
2 <!--// File: views/transfer.ejs-->
3 <!DOCTYPE html>
4 <html lang="en">
5 <head>
6     <meta charset="UTF-8">
7     <meta name="viewport" content="width=device-width, initial-
8         scale=1.0">
9     <title>Transfer Funds - E-Bank</title>
10    <!-- Bootstrap CSS from CDN -->
11    <link href="https://cdnjs.cloudflare.com/ajax/libs/bootstrap
12        /5.3.0/css/bootstrap.min.css" rel="stylesheet">
13    <!-- Font Awesome for icons -->
14    <link href="https://cdnjs.cloudflare.com/ajax/libs/font-
15        awesome/6.4.0/css/all.min.css" rel="stylesheet">
16    <link rel="stylesheet" href="/css/dashboard.css">
17    <link rel="stylesheet" href="/css/styles.css">
18 </head>
19 <body>
20    <!-- Sidebar -->
21    <div class="sidebar d-none d-lg-block">
22        <div class="bank-logo">E(xpress)-Bank</div>
23        <ul class="nav flex-column">
24            <li class="nav-item">
25                <a class="nav-link" href="/dashboard"><i class="
26                    fas fa-home"></i> Dashboard</a>
27            </li>
28            <li class="nav-item">
29                <a class="nav-link active" href="/transfer"><i
30                    class="fas fa-exchange-alt"></i> Transfers</a>
31            </li>
32            <li class="nav-item">
33                <a class="nav-link" href="#"><i class="fas fa-
34                    credit-card"></i> Cards</a>
35            </li>
36            <li class="nav-item">

```

```

31         <a class="nav-link" href="#"><i class="fas fa-
           chart-pie"></i> Investments</a>
32     </li>
33     <li class="nav-item">
34         <a class="nav-link" href="#"><i class="fas fa-cog
           "></i> Settings</a>
35     </li>
36     <li class="nav-item mt-5">
37         <a class="nav-link text-danger" id="logoutButton"
           > <i class="fas fa-sign-out-alt"></i> Logout</
           a>
38     </li>
39 </ul>
40 </div>
41
42 <!-- Main Content -->
43 <div class="main-content">
44     <!-- Header with title -->
45     <div class="d-flex justify-content-between align-items-
           center mb-4">
46         <h4>Transfer Funds</h4>
47     </div>
48
49     <div class="row">
50         <div class="col-md-8">
51             <div class="card shadow-sm mb-4">
52                 <div class="card-body">
53                     <h5 class="card-title mb-3">New Transfer<
                       /h5>
54                     <form id="transfer-form">
55                         <div class="mb-3">
56                             <label for="fromAccount" class="
                               form-label">From Account</
                               label>
57                             <select class="form-select" id="
                               fromAccount" name="fromAccount
                               " required>
58                                 <option value="">Select
                                   account</option>
59                             </select>
60                         </div>
61
62                         <div class="mb-3">
63                             <label for="toAccount" class="
                               form-label">To Account</label>
64                             <input type="text" class="form-
                               select" id="toAccount" name="
                               fromAccount" required>
65                         </div>
66
67                     <div class="mb-3">

```

```

68         <label for="amount" class="form-
        label">Amount</label>
69         <div class="input-group">
70             <span class="input-group-text"
              >$</span>
71             <input type="number" class="
              form-control" id="amount"
              name="amount" min="0.01"
              step="0.01" required>
72         </div>
73     </div>
74
75     <div class="mb-3">
76         <label for="note" class="form-
        label">Note (optional)</label>
77         <input type="text" class="form-
        control" id="note" name="note"
        placeholder="What's this
        transfer for?">
78     </div>
79
80     <button type="submit" class="btn btn-
    primary">Transfer Funds</button>
81 </form>
82 </div>
83 </div>
84 </div>
85
86 <div class="col-md-4">
87     <div class="card shadow-sm">
88         <div class="card-body">
89             <h5 class="card-title mb-3">Transfer Tips
            </h5>
90             <ul class="small text-muted">
91                 <li>Transfers between your accounts
                    are instant</li>
92                 <li>You can transfer up to $10,000
                    per day</li>
93                 <li>Adding a note helps you track
                    transfers later</li>
94                 <li>Transfers to external accounts
                    may take 1-3 business days</li>
95             </ul>
96         </div>
97     </div>
98 </div>
99 </div>
100
101 <div class="row mt-4">
102     <div class="col-12">
103         <div class="card shadow-sm">

```

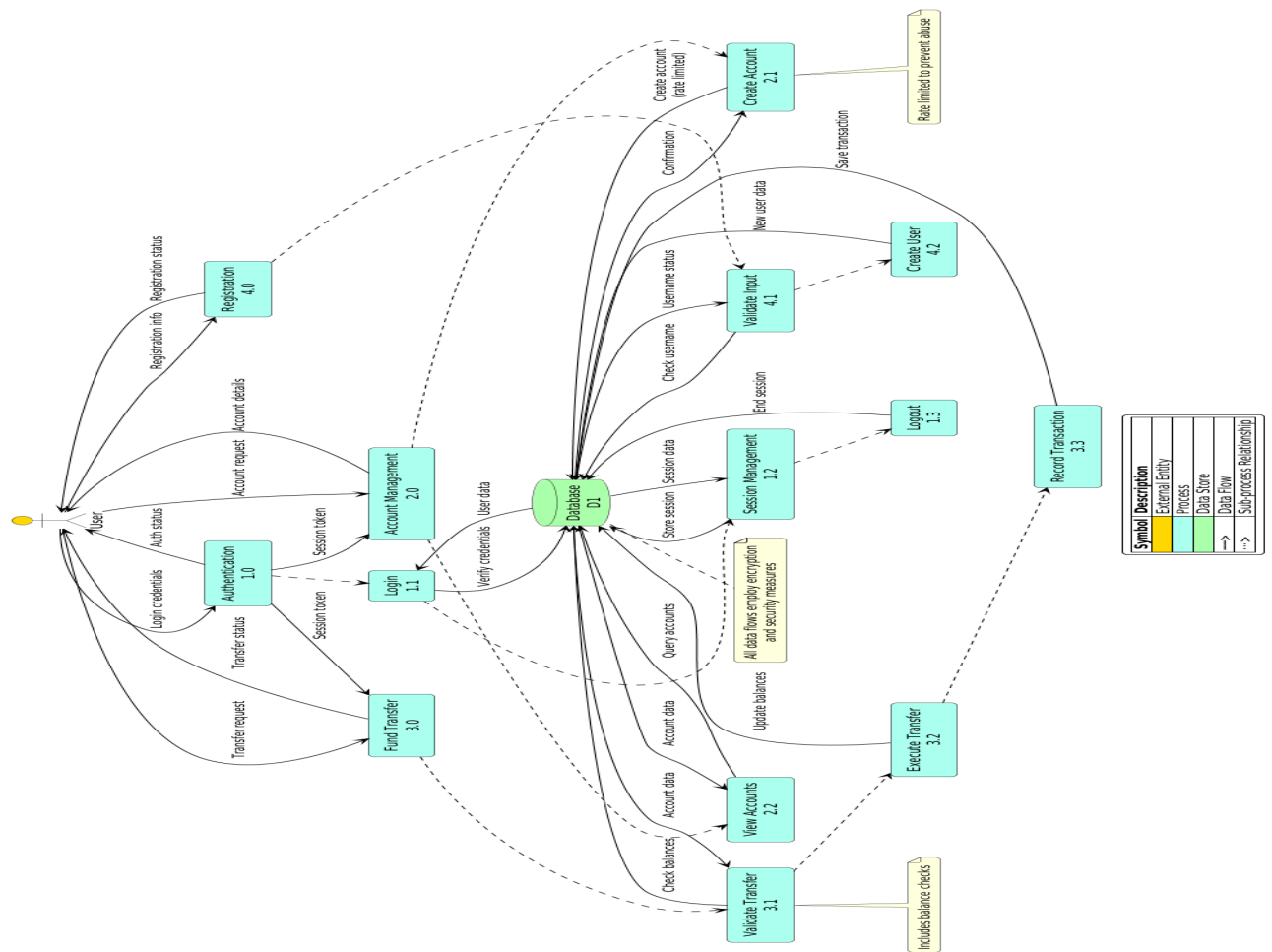
```

104         <div class="card-body">
105             <h5 class="card-title mb-3">Recent
                Transfers</h5>
106             <div id="transfer-history" class="
                transaction-list">
107                 <!-- Transfer history will be loaded
                    here -->
108                 <p class="text-center text-muted">
                    Loading transfer history...</p>
109             </div>
110         </div>
111     </div>
112 </div>
113 </div>
114 </div>
115
116 <!-- Bootstrap JS -->
117 <script src="https://cdnjs.cloudflare.com/ajax/libs/bootstrap
    /5.3.0/js/bootstrap.bundle.min.js"></script>
118 <!-- Transfer Script -->
119 <script src="/js/transfers.cjs"></script>
120
121 <script>
122     // Logout functionality
123     document.getElementById('logoutButton').addEventListener
        ('click', async function() {
124         try {
125             const response = await fetch("/auth/logout", {
126                 method: "POST",
127                 headers: {
128                     "Content-Type": "application/json",
129                 },
130             });
131
132             if (response.ok) {
133                 window.location.href = "/login";
134             } else {
135                 console.error("Logout failed");
136             }
137         } catch (error) {
138             console.error("Error during logout:", error);
139             window.location.href = "/login";
140         }
141     });
142 </script>
143 </body>
144 </html>

```

Chapter 6

Data Flow Diagram



Chapter 7

Outputs

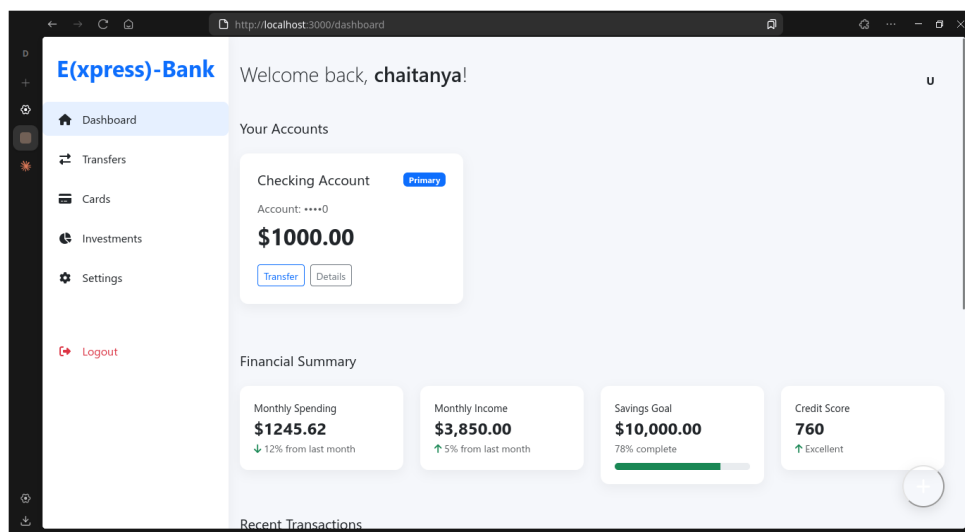


Figure 7.1: Dashboard Interface

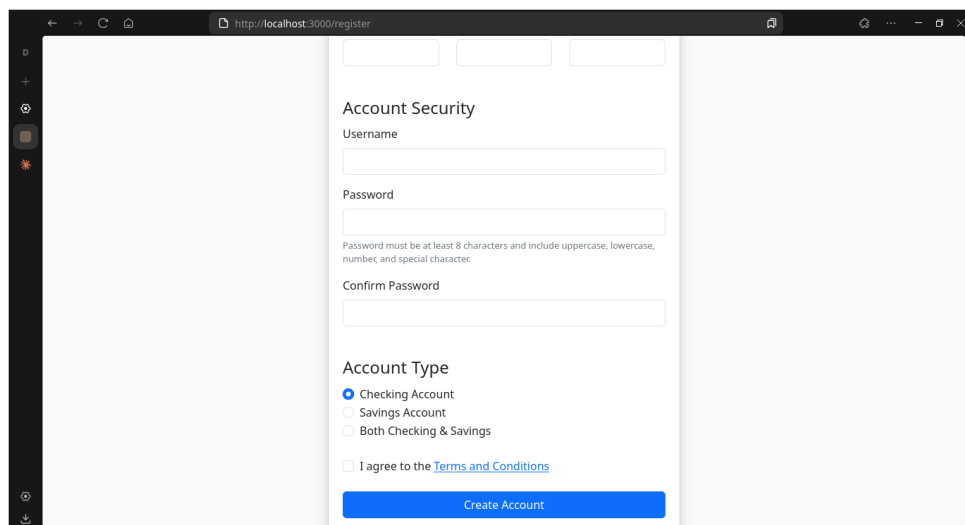


Figure 7.2: User Sign Up Interface

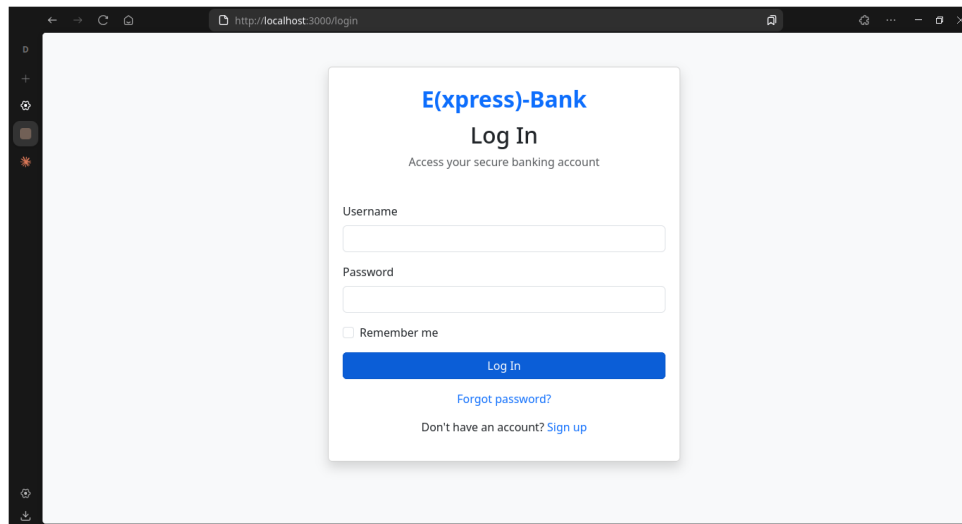


Figure 7.3: User Account Login Interface

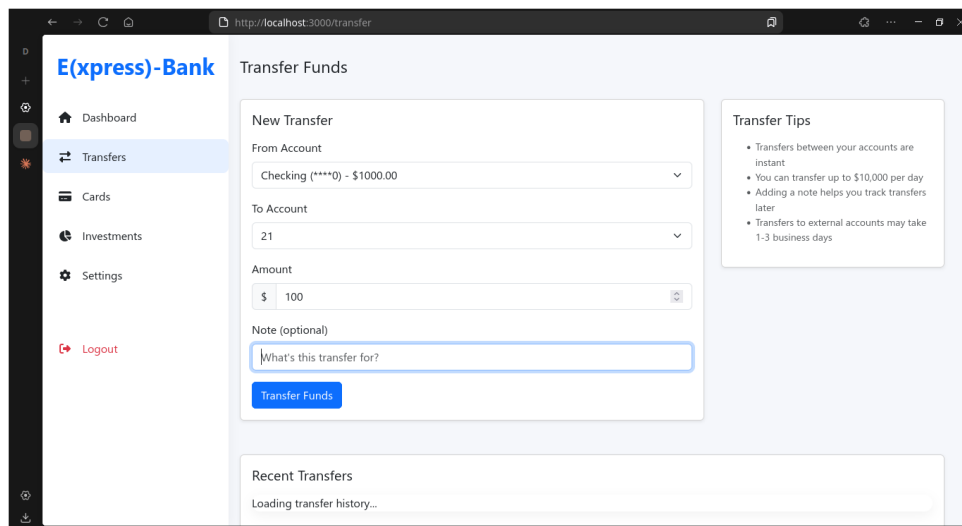


Figure 7.4: Transaction Interface

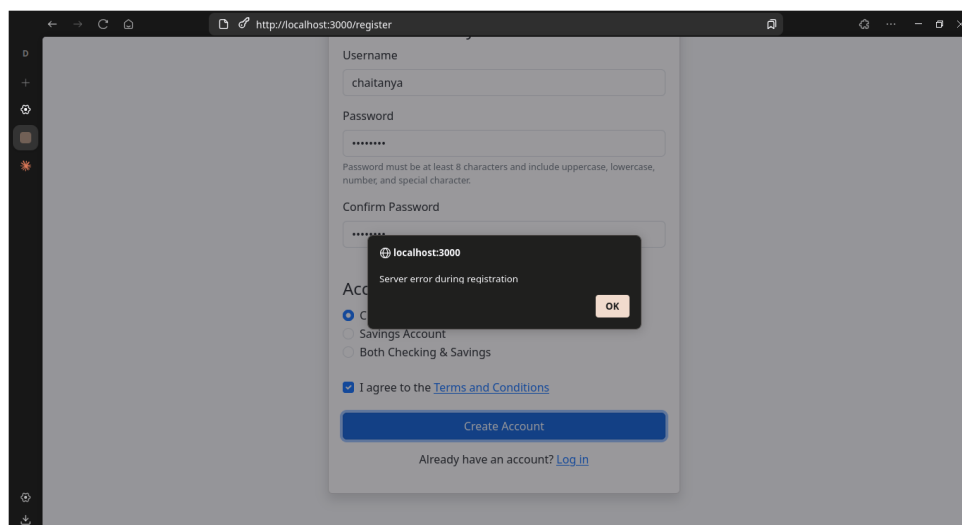


Figure 7.5: Error Messages

Chapter 8

References

1. <https://stackoverflow.com/questions/5132923/how-to-add-a-primary-key-to-a-mysql-table>
2. <https://stackoverflow.com/questions/9070764/insert-auto-increment-primary-key-to-existing-table>
3. <https://stackoverflow.com/questions/5813771/in-ejs-template-engine-how-do-i-include-a-footer>
4. <https://stackoverflow.com/questions/22212440/using-math-class-to-convert-cosine-into-degrees>
5. <https://www.geeksforgeeks.org/how-to-connect-node-js-application-to-mysql/>
6. <https://stackoverflow.com/questions/13396119/how-to-add-a-value-to-existing-value-using-sql-query>
7. <https://stackoverflow.blog/css/>

Bibliography

- [1] Full Project Source Code: https://github.com/Atan-D-RP4/wt_project
- [2] Git: <https://git-scm.com/>
- [3] GitHub: <https://github.com/>
- [4] Deno: <https://deno.com/>
- [5] Node.js: <https://nodejs.org/en/>
- [6] Bootstrap: <https://getbootstrap.com/>
- [7] Express.js: <https://expressjs.com/>
- [8] EJS: <https://ejs.co/>
- [9] MySQL: <https://www.mysql.com/>
- [10] Visual Studio Code. <https://code.visualstudio.com/>