Project Report

Save-Up: Money Management

Abhishek Nayak (E22CSEU1499)
Neha Nayak (E22CSEU1518)
Utkarsh Singh (E22CSEU0540)

A report submitted in part fulfilment of the degree of

B.Tech in Computer Science

Supervisor: Dr. Naween Kumar

School of Computer Science Engineering and Technology
Bennett University

April 27, 2025

1. Introduction

1.1 Overview

Save-Up is a money management tracker that allows users to track their expenses and savings. It has a dashboard that shows updates, lots of tools for managing transactions, goals, billing reminders, and bank statement upload. It is built using modern web technologies and hosted on AWS cloud. The platform is made simple so that users can use it easily and connect with cloud services smoothly.

1.2 Motivation

Managing money is difficult because of too many tools and doing things manually. This project tries to fix it by making one place where everything is managed. Users can track their money, set savings goals, get reminders for bills, and download bank statements. The aim is to make finance management easy and give live updates, email alerts and a clean simple interface.

2. Key Features

The Money Tracking Website have these main features:

- Dashboard Overview: A dashboard showing live income, expenses, and savings. There are charts and summaries to help users see their money clearly.
- Transactions Management: Users can add new transactions, sort them by type, date or name. It is easy to find old expenses and income.
- Savings Goals Tracking: Users can create savings goals (like for car, phone etc.) and check how close they are to reaching it. They can also edit goals later.
- Bills Management and Reminders: Users can add bills (monthly or one-time) and will get email reminders when due date is coming or missed.
- Bank Statement Upload and Generation: Users can upload their bank statements. The platform will read and save them. Also users can create PDF reports for any selected date.
- Email Notifications: Email is sent for important things like overdue bills or when bank statement is uploaded, using AWS email services.

3. System Architecture

3.1 High-Level Architecture Diagram

The website is using microservices style, and the main parts are:

- Frontend (React, Next.js): What user sees and interacts.
- Backend (Next.js API Routes, tRPC): Does the work behind the scenes and sends/receives data from frontend.
- Cloud (AWS EC2, RDS, S3, SES): Takes care of hosting, database, file storage and sending emails.

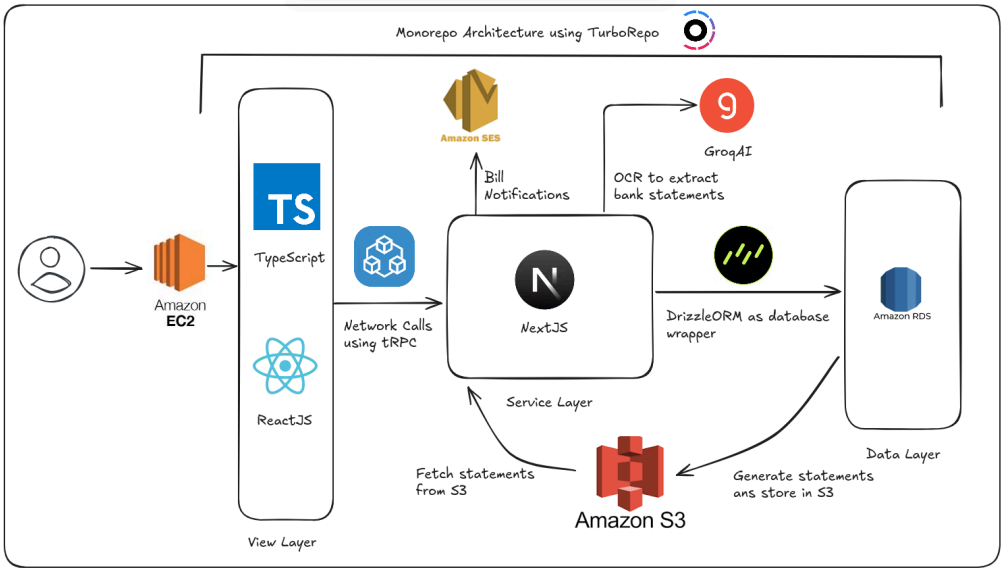

Figure 1: Project Architecture

3.2 Backend and Frontend Interaction

Frontend (React, Next.js) talks to backend through tRPC for safe sending and getting data.
Backend has Next.js API routes for handling tasks like transactions and bills.
NextAuth is used for login and keeping users signed in securely.

3.3 Database Schema

The data is saved in Amazon RDS (PostgreSQL). Important tables are:

- User: Has user info and login data.
- Transactions: Keeps record of income and expenses.
- Savings Goals: Stores savings goals made by users.
- Bills: Saves the bills data.
- Bank Statements: Saves uploaded bank statements in S3 storage.

The database is made to work fast, store data safe and connect different tables properly.

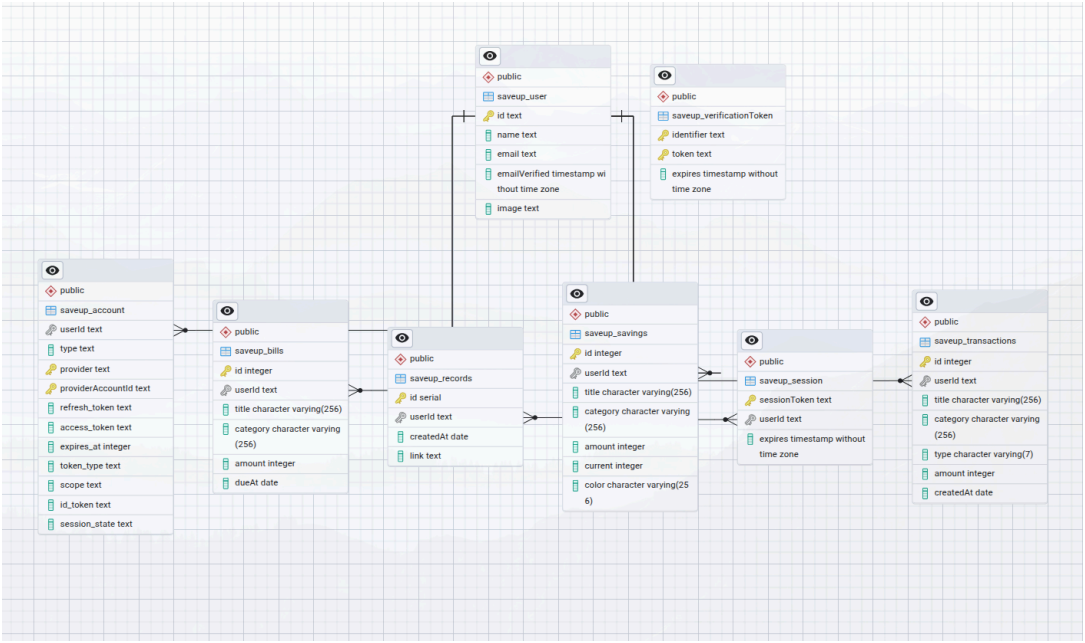

Figure 2: Database ERD Diagram

4. Technology Stack

4.1 Frontend Technologies

- **React**: It is JavaScript library for making user interface. It use components so things stay organized and easier to manage.
- **Next.js**: It is framework on top of React. It helps do server-side rendering (SSR) and static generation (SSG) which make website faster.
- **Tailwind CSS**: CSS framework with many ready classes. It make styling fast and less headache.
- **TypeScript**: Like JavaScript but with types. It help catch mistakes early and make code better and safer.

4.2 Backend & API Structure (tRPC, Next.js, GroqAI)

- **Next.js API Routes**: It handle backend logic. It give APIs so frontend can send data like transactions, bills, etc.
- **tRPC**: Help frontend and backend talk together without writing extra code. It is type-safe.
- **GroqAI:** Handles extracting transactions from bank statements using OCR.

4.3 Authentication (NextAuth)

- **NextAuth**: It handle login system. Support login with Google, GitHub and others. It manage user session with JWT token for better security.

4.4 ORM & Database (DrizzleORM, RDS)

- **DrizzleORM**: ORM tool for talking to database. It safe and type-checked so less bugs.
- **Amazon RDS (PostgreSQL)**: It is main database where user info, transactions, bills and statements are stored safely.

5. Cloud Infrastructure (AWS)

5.1 EC2 and Load Balancer for Deployment

AWS EC2 instances used for hosting the Money Tracking Website. It give scalable server power to run app properly. Elastic Load Balancer (ELB) put in front, to send traffic between multiple EC2 instances, so website stay available even if one server has issue.

5.2 RDS for User Data

Amazon RDS (PostgreSQL) is used for store all user-related data like transactions, savings goals, bills and more. RDS makes sure database is safe, backup happens automatic and it can grow when more user come.

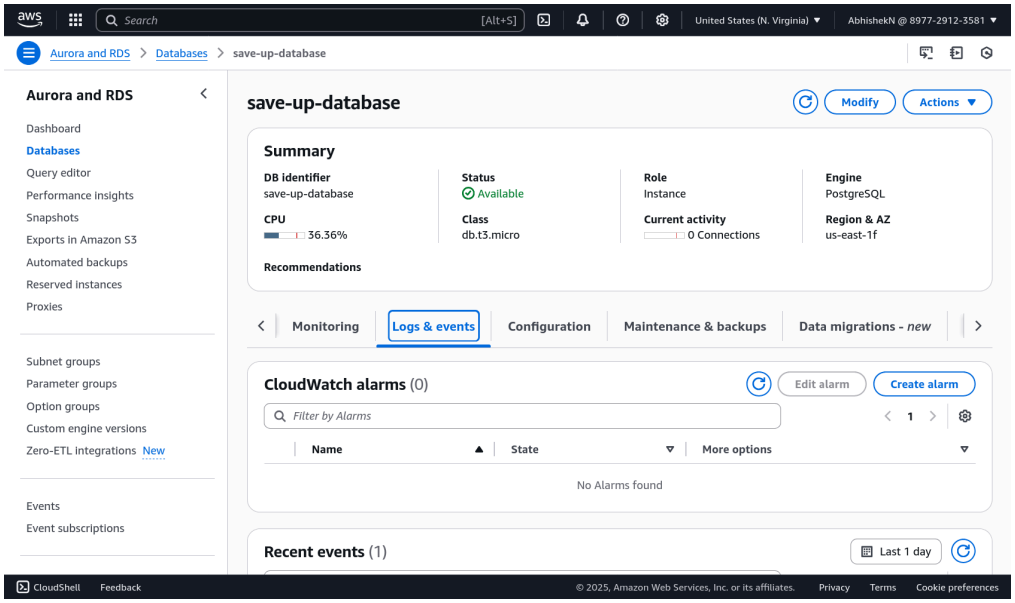

Figure 3: RDS deployment

5.3 S3 for File Storage (Bank Statements)

Amazon S3 used for keep uploaded bank statement files. S3 make sure files are safe, not lost, and user can access whenever needed.

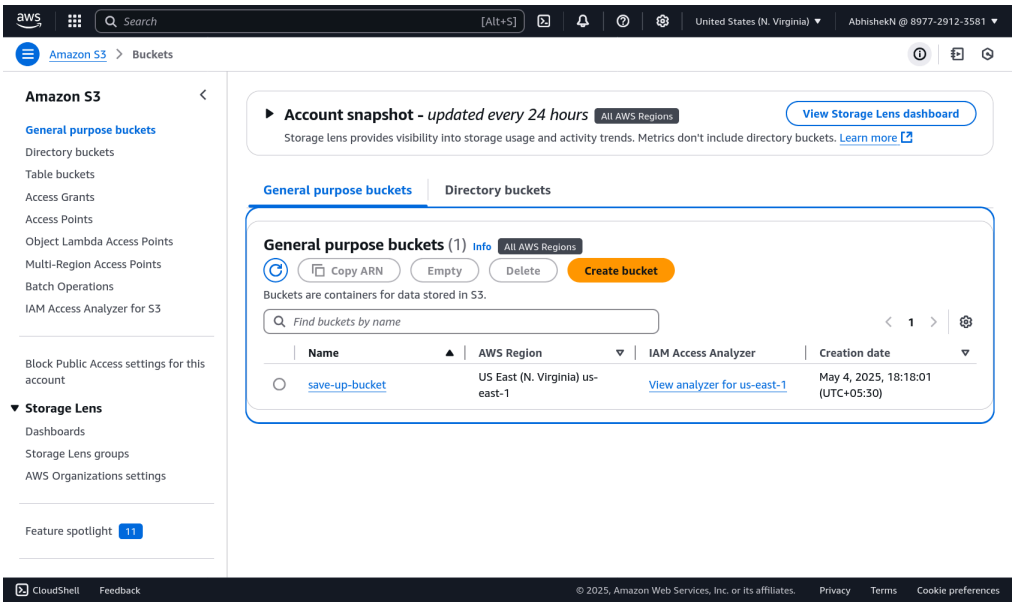

Figure 4: S3 Bucket

5.4 SES & SNS for Email Notifications

Amazon SES used to send emails, like overdue bill reminders or successful uploads. Amazon SNS helps send important messages faster, like alerts for users when something happen on their account.

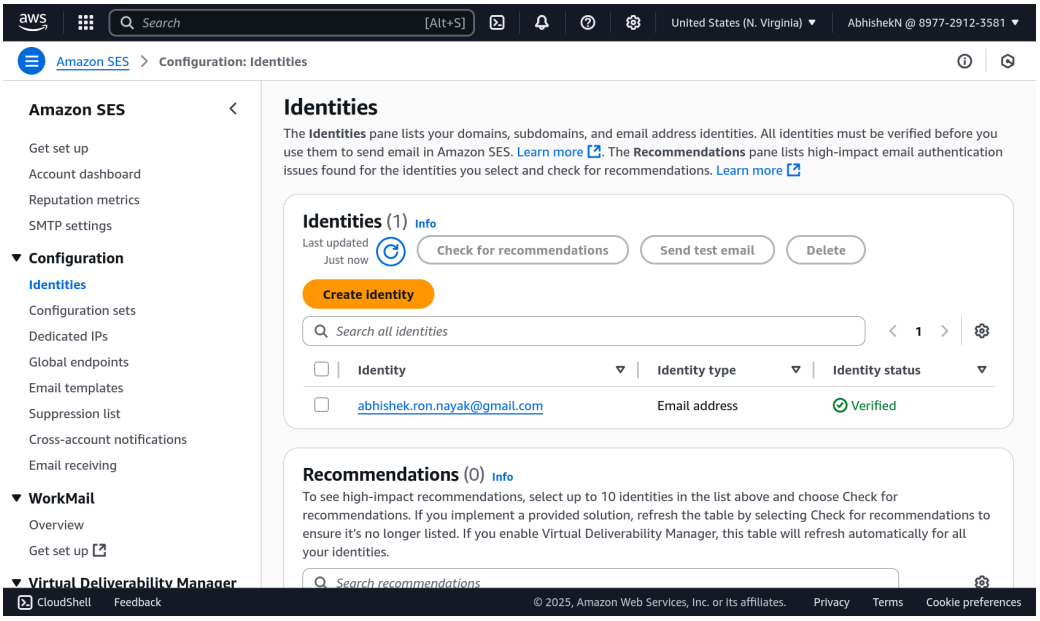

Figure 5: SES (Simple Email Service)

6. Core Functionalities

6.1 Dashboard Analytics

- Dashboard show graphs and charts of income, expense and savings.
- It help users understand where money coming from and going to.

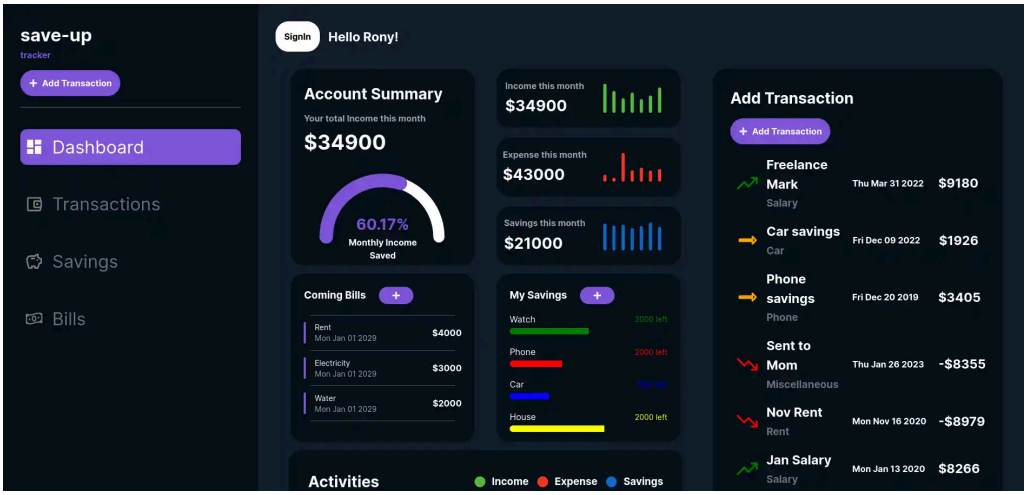

Figure 6: Dashboard

6.2 Transaction Management

- **Adding and Searching Transactions**: User can add incomes or expenses. They can also search old transactions by description or amount.
- **Filtering by Type, Time, and Text**: User can filter transactions by income/expense, by date range, or by keyword to find fast.

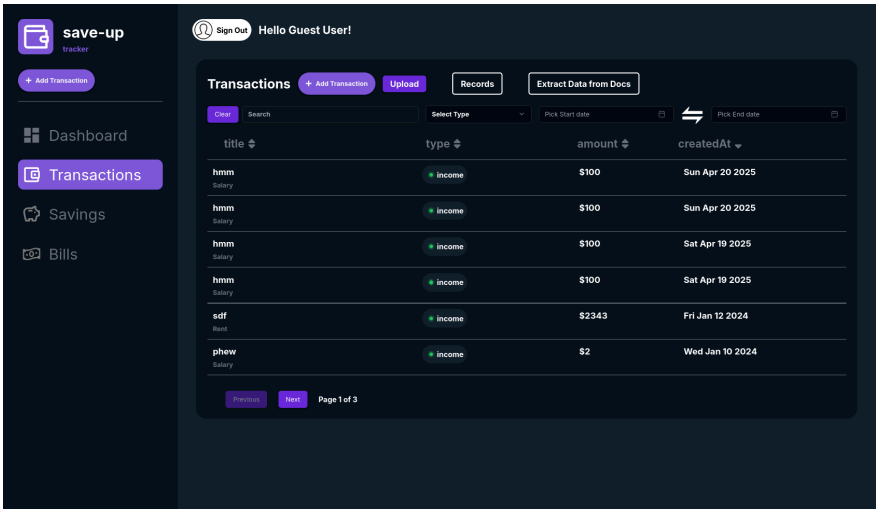

Figure 7: Transactions Page

6.3 Savings Page

- User can create savings goals, like for buy phone or trip.
- They can update progress and see how much left to save.

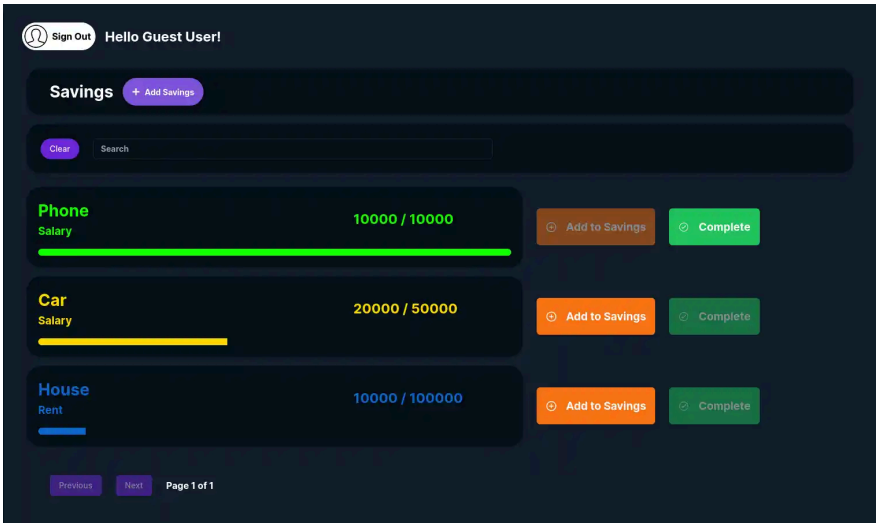

Figure 8: Savings Page

6.4 Bills Page

- User add bills, like electricity, rent, water bills etc.
- System send email if bill is near due date or already overdue.

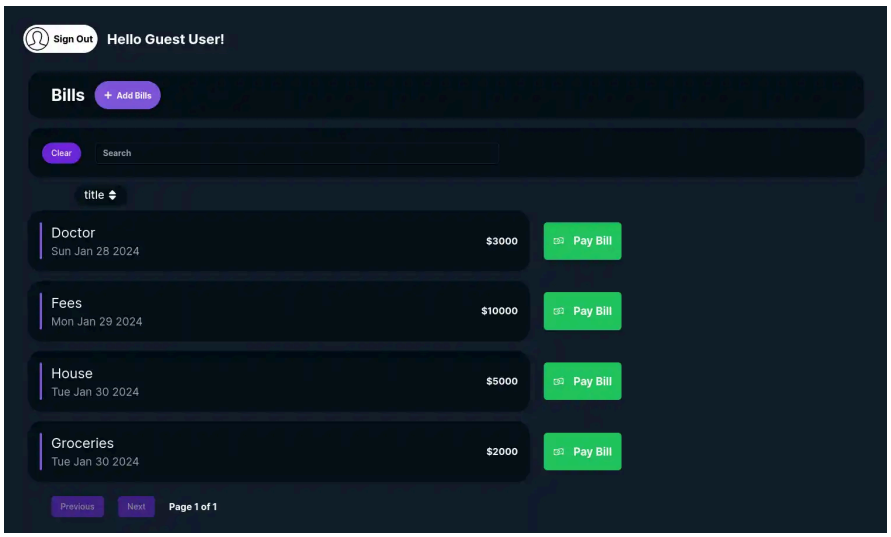

Figure 9: Bills Page

6.5 Bank Statement Handling

- User upload their bank statement file. It gets saved and processed automatically using GroqAI.
- User can also download a clean PDF bank statement for any time period.

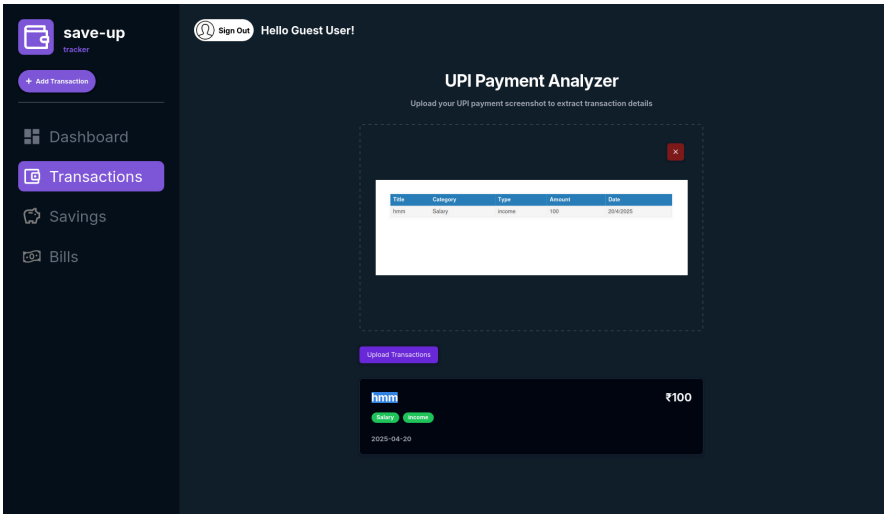

Figure 10: Transaction Extractor from Bank Statement

7. Security and Authentication

7.1 OAuth via NextAuth

- NextAuth used for secure login.
- User can sign in using Discord, GitHub etc., so no need to store password in platform.

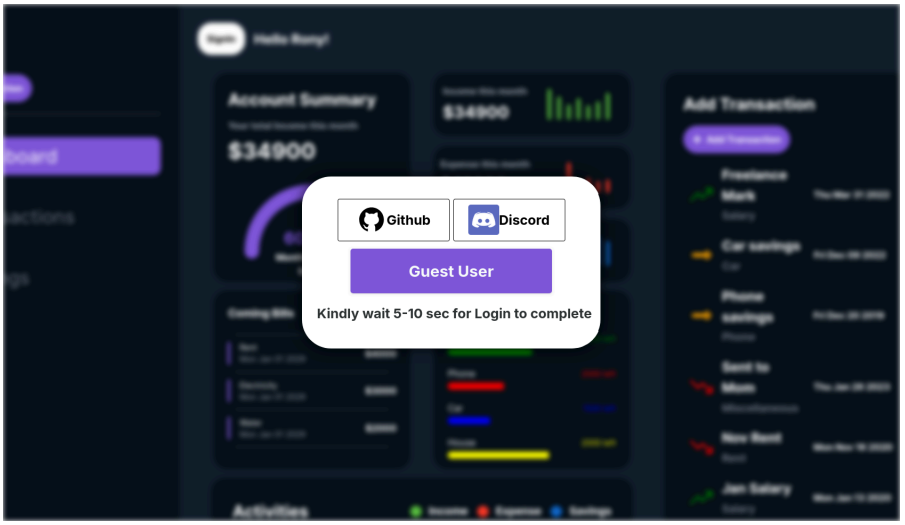

Figure 11: Login Page

7.2 Secure Data Storage

- All data is encrypted while sending (HTTPS) and also while stored (RDS, S3).
- Database managed securely with backups, access control and encryption.

7.3 Access Control and User Sessions

- Only user can see their own data, nobody else.
- Sessions are managed by JWT, so user stay logged in safely without risking data.

8. Challenges Faced and Solutions

8.1 Technical or Architectural Issues

- **Challenge**: Connecting all different tech like React, Next.js, tRPC and AWS together without problem was hard.
- **Solution**: Used tRPC for easy and type-safe APIs, and Next.js API Routes to handle server work neatly.

8.2 AWS Integration Challenges

- **Challenge**: Setting up EC2, RDS, S3, SES correctly took time, especially making them work nicely together.
- **Solution**: Used Load Balancer to handle scaling traffic. RDS backups setup automatic. Used IAM roles for security.

8.3 Performance or Scalability Considerations

- **Challenge**: Making sure app stay fast even when lot of users upload transactions and bank statements.
- **Solution**: Optimized database queries, added indexes on important columns, and used RDS scaling for bigger load.

9. Future Enhancements

9.1 Planned Upgrades or New Features

- **Mobile Support**: Plan to build iOS and Android app so user can manage money from phone easily.
- **Advanced Analytics**: Add more detailed reports like spending by category, month-by-month tracking.
- **Budget Planning**: Feature for users to set monthly/yearly budgets, and get warning if they overspend.

10. Conclusion

10.1 Final Thoughts

Money Tracking Website gives full solution for personal finance.
With simple dashboard, goal tracking, reminders and cloud support, users can easily control their money life.

10.2 Summary of Learnings and Outcomes

- Learned how to use AWS services properly like EC2, RDS, S3, SES.
- Build type-safe API using tRPC and Next.js.
- Secure authentication with NextAuth.
- Project completed main goals, and will keep growing with more features in future.