A Major Project Report

on

MyHeaven

For

Bachelor of Technology

in

Computer Science and Engineering

(2025-2026)

Submitted to

RAJIV GANDHI PROUDYOGIKI VISHWAVIDYALAYA
BHOPAL (M.P.)

Submitted By

Akansha dwivedi (0302CS221011)

Gargi Singh (0302CS221043)

Approved By
Dr. K.P. Tripathi
(H.O.D. CSE Dept.)

Guided By
Dr. Uday Singh Kushwaha
(Asst. Prof. CSE Dept.)

Project Coordinator
Dr. Uday Singh Kushwaha
(Asst. Prof. CSE Dept.)

TRIGGERING
TRANSFORMATIONS

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
VINDHYA INSTITUTEOF TECHNOLOGY & SCIENCE
SATNA (M.P)

Rajiv Gandhi Proudyogiki Vishwavidyalaya Bhopal (M.P)
Vindhya Institute of Technology & Science Satna (M.P)

CERTIFICATE

This is to certify that the project entitled as "**MyHeaven"** which has been completed & submitted by **Akansha Dwivedi, Gargi Singh** in partial fulfillment of the requirement for the award of the degree of Bachelor of Technology in Computer Science and Engineering for the session 2025-2026 is a bonafide work by them and has been completed under my guidance and supervision. It has not been submitted elsewhere for any other degree.

H.O.D
Dr. K.P. Tripathi

Guided By
Dr. Uday Singh Kushwaha

Principal
Dr. P.K. Shukla

Rajiv Gandhi Proudyogiki Vishwavidyalaya Bhopal (M.P)
Vindhya Institute of Technology & Science Satna (M.P)

CERTIFICATE

This is to certify that the project entitled as "**MyHevaen"** which has been completed & submitted by **Akansha Dwivedi, Gargi Singh** in partial fulfillment of the requirement for the award of the degree of Bachelor of Technology in Computer Science and Engineering for the session2025-2026 is a bonafide work by them and has been completed under my guidance and supervision. It has not been submitted elsewhere for any other degree.

(External Examiner) **(Internal Examiner)**

ACKNOWLEDGEMENT

A project like this one involves many people and would be incomplete without the mention of all those people whose guidance and encouragement helped in the successful completion of this project.

Our heartily thanks to all faculty members of **Department of Computer Science and Engineering, Vindhya Institute of Technology & Science, Satna** for their effort towards our project.

We would like to thanks our guide **and** Project coordinator **Dr. Uday Singh Kushwaha** who has been a great source of inspiration for us and without whose humble guidance of project was never to shape.

We are also indebted to our H.O.D. **Dr. K.P. Tripathi** for his encouragement, guidance and support.

We are grateful to **Dr. P.K. Shukla** , Principal of **Vindhya Institute of Technology & Science (VITS)**, for his constant support and for providing us with the necessary resources and a conducive environment to complete this project.

We are also thankful to many people whose timely help but paucity of space is restricting us from mentioning their name. And finally, we also thank to all my colleagues who were constant support during the whole project.

DECLARATION

We hereby declare that the work which is being presented in the project report entitled "**MyHeaven"** partial fulfillmentof the requirement of the degree of **Bachelor of Technology in Computer Science and Engineering** branch is an authentic record of our work carried out under the guidance of **Dr. Uday Singh Kushwaha .** The work has been carried out at **Vindhya Institute of Technology & Science, Satna.**

Signature

Akansha Dwivedi (0302CS221011)
Gargi Singh (0302CS221043)

TABLE OF CONTENT

CHAPTER NO.	TITLE	PAGE NO.
	ABSTRACT	II
	LIST OF ABBREVATIONS	III
	LIST OF FIGURES	IV
1	INTRODUCTION	1
2	LITERATURE SURVEY	2
3	EXISTING & PROPOSED	3-8
	3.1 EXSISTING SYSTEM	3
	3.2 PROPOSED SYSTEM	3
	3.3 FLOW CHART	4
	3.4 DATA FLOW DIAGRAM	5
	3.5 E-R DIAGRAM	6
	3.6 WORKING PRINCIPLE	7-8
4	COMPONENT/TOOLS/TECHNOLOGY DETAILS	9-11
	4.1 FRONT END	9
	4.2 BACK-END	10
	4.3 DATABASE	10
	4.4 ADDITIONAL TOOLS	11
5	IMPLEMENTATION	12-27
6	OUTPUT IMAGES	28-30
7	FUTURE SCOPE AND CONCLUSION	33-34
	7.1. CONCLUSION	31
	7.2 FUTURE SCOPE	32
8	REFERENCES	33

ABSTRACT

The MyHeaven project is a comprehensive full-stack personal finance management system designed to simplify and automate the process of tracking personal income and expenses. It empowers users to maintain a complete record of their daily financial activities, categorize their transactions, and visualize their overall financial health through an interactive dashboard. The system aims to promote better financial planning, transparency, and awareness among users by providing them with a seamless digital platform.

The backend of the application is developed using Spring Boot, a powerful Java-based framework that supports the creation of production-ready and scalable applications. It follows a layered architecture, consisting of Controller, Service, Repository, and Entity layers to ensure modularity, separation of concerns, and easier maintenance. The backend exposes RESTful APIs that handle user authentication, category management, income and expense tracking, data filtering, and dashboard analytics. The Spring Security framework is integrated for robust user authentication and authorization, employing JWT (JSON Web Token)-based stateless authentication to ensure secure data transmission between client and server. Passwords are encrypted using the BCrypt hashing algorithm, adding an additional layer of security.

The backend interacts with a MySQL database, which serves as the centralized data repository for storing user profiles, categories, income records, expense details, and related metadata. Using Spring Data JPA (Hibernate), the system efficiently performs CRUD (Create, Read, Update, Delete) operations while maintaining data integrity and consistency. The database schema is normalized to minimize redundancy and support optimized querying and retrieval of transactional data.

The frontend of the application is developed using React.js, a modern JavaScript library used for building fast and interactive user interfaces. Tailwind CSS has been used to design a responsive and visually appealing user experience, providing a clean and minimalistic layout. The frontend consumes the REST APIs provided by the backend and presents real-time updates of user data. It allows users to register, log in, manage their income and expenses, apply filters for date or category-based analysis, and view graphical summaries of their financial data on the dashboard.

In addition to the core functionality, the project supports CORS configuration, enabling smooth communication between the React frontend (running on a different port) and the Spring Boot backend. The application follows industry-standard development practices, including dependency injection, DTO (Data Transfer Object) mapping, validation handling, and exception management.

Overall, MyHeaven offers a secure, scalable, and user-friendly platform for managing personal finances. It demonstrates the effective integration of modern web technologies such as React.js, Tailwind CSS, Spring Boot, and MySQL, making it a robust example of a full-stack web application designed with a focus on performance, usability, and security.

LIST OF ABBREVIATION

Abbreviation	Full Form	Meaning / Use in Project
HTML	HyperText Markup Language	Used to create webpage structure
CSS	Cascading Style Sheets	Used for website styling
JS	JavaScript	Used to add interactivity to websites
React JS	React JavaScript Library	Used to build dynamic frontend user interfaces
API	Application Programming Interface	Communication layer between frontend and backend
REST	Representational State Transfer	Architecture style for designing APIs
CRUD	Create, Read, Update, Delete	Basic operations for working with databases
SQL	Structured Query Language	Used to write database queries
DB	Database	System for storing data
RDBMS	Relational Database Management System	Table-based structured database system
JPA	Java Persistence API	ORM tool for mapping Java objects to database
JWT	JSON Web Token	Token-based user authentication
JSON	JavaScript Object Notation	Lightweight format for data exchange
IDE	Integrated Development Environment	Development tool (e.g., VS Code, IntelliJ)
DTO	Data Transfer Object	Structure for transferring data between layers
ORM	Object Relational Mapping	Mapping between objects and database tables
MVC	Model View Controller	Application architectural pattern
HTTP	HyperText Transfer Protocol	Communication between client and server

LIST OF FIGURES

Figure No.	Title	Page No.
Fig. 3.4	Flow Chart of MyHeaven Platform	4
Fig. 3.5	Data-Flow Diagram	5
Fig. 3.6	E-R Diagram	6

CHAPTER 1 : INTRODUCTION

The **Personal Finance Management System (MyHeaven)** is a full-stack web application designed to help users effectively manage their income and expenses in a simple, secure, and user-friendly manner. In today's fast-paced world, individuals often struggle to track their daily financial activities, leading to poor budgeting and unplanned spending. This system provides a structured and automated solution to record, categorize, and analyze all financial transactions efficiently.

The project is built using **Spring Boot** on the backend, **React.js** with **Tailwind CSS** on the frontend, and **MySQL** as the database. The integration of these modern technologies ensures high performance, scalability, and a seamless user experience.

On the **frontend**, React.js is used to build a dynamic, component-based user interface that enhances interactivity and responsiveness. Tailwind CSS provides a clean and modern design with customizable styles, ensuring the application is visually appealing across devices. Users can easily navigate through the dashboard to add, view, update, or delete income and expense records. The frontend communicates with the backend through RESTful APIs to ensure real-time synchronization of data.

The **backend**, developed using **Spring Boot**, serves as the core of the application. It handles business logic, data validation, and interaction with the database. It includes modular components such as **controllers**, **services**, **repositories**, and **entities** that work together following best practices of the **MVC (Model-View-Controller)** architecture. The backend ensures secure authentication and authorization for users, accurate data management, and optimized API responses.

The **MySQL database** is used to store all financial records, user details, and categorized transaction data. Its relational structure allows for efficient data retrieval and ensures consistency. Each user has a personalized dataset to maintain privacy and accuracy in financial tracking.

Overall, this system bridges the gap between users and financial awareness by providing an intuitive platform for **tracking, managing, and analyzing personal finances**. It encourages users to understand their spending habits, save more effectively, and make informed financial decisions. The modular and scalable design also allows future expansion, such as adding data visualization, reports, and AI-based financial recommendations.

CHAPTER 2 : LITERATURE SURVEY

The concept of **Personal Finance Management Systems (PFMS)** has become increasingly relevant in today's digital era, as people rely more on technology to manage their income, savings, and expenses effectively. Traditional methods such as maintaining spreadsheets or manual records often lead to errors, inefficiency, and data loss. Hence, web-based applications like **Mint**, **YNAB (You Need A Budget)**, and **PocketGuard** have emerged to automate and simplify financial tracking. However, these existing tools often come with limitations such as premium restrictions, complex navigation, limited customization options, and concerns regarding data privacy.

Recent research in financial technology emphasizes the need for systems that provide **user-friendly interfaces, strong data security, and real-time data visualization**. Studies also indicate that personalized insights and categorized financial summaries help users achieve better control over their spending habits. The **MyHeaven** project addresses these gaps by providing a lightweight, secure, and customizable solution tailored for individual users.

MyHeaven utilizes **Spring Boot** as its backend framework, offering robust performance, scalability, and simplified API development. It implements **JWT (JSON Web Token)** authentication to enhance security and ensure that user data remains private. The **React.js** frontend delivers a dynamic, interactive experience, while **Tailwind CSS** ensures an elegant, responsive design that works seamlessly across devices. The system's **MySQL database** provides structured data storage and fast query execution, ensuring reliability and efficiency in handling financial records.

Compared to existing systems, **MyHeaven** prioritizes simplicity and accessibility without compromising functionality. It enables users to register, log in, and manage personal finance categories effortlessly. The integration of advanced frameworks like Spring Boot and React ensures smooth communication between frontend and backend while maintaining security and performance. The inclusion of Tailwind CSS gives the application a modern aesthetic, improving user engagement.

In conclusion, the **MyHeaven** project stands as an innovative step toward modern financial management. It combines advanced web technologies with practical usability, creating a platform that empowers users to track, analyze, and control their finances with ease. Through this system, individuals can experience a perfect blend of technology, design, and functionality for everyday financial planning.

CHAPTER 3 : EXISTING & PROPOSED SYSTEM

3.1 Existing System

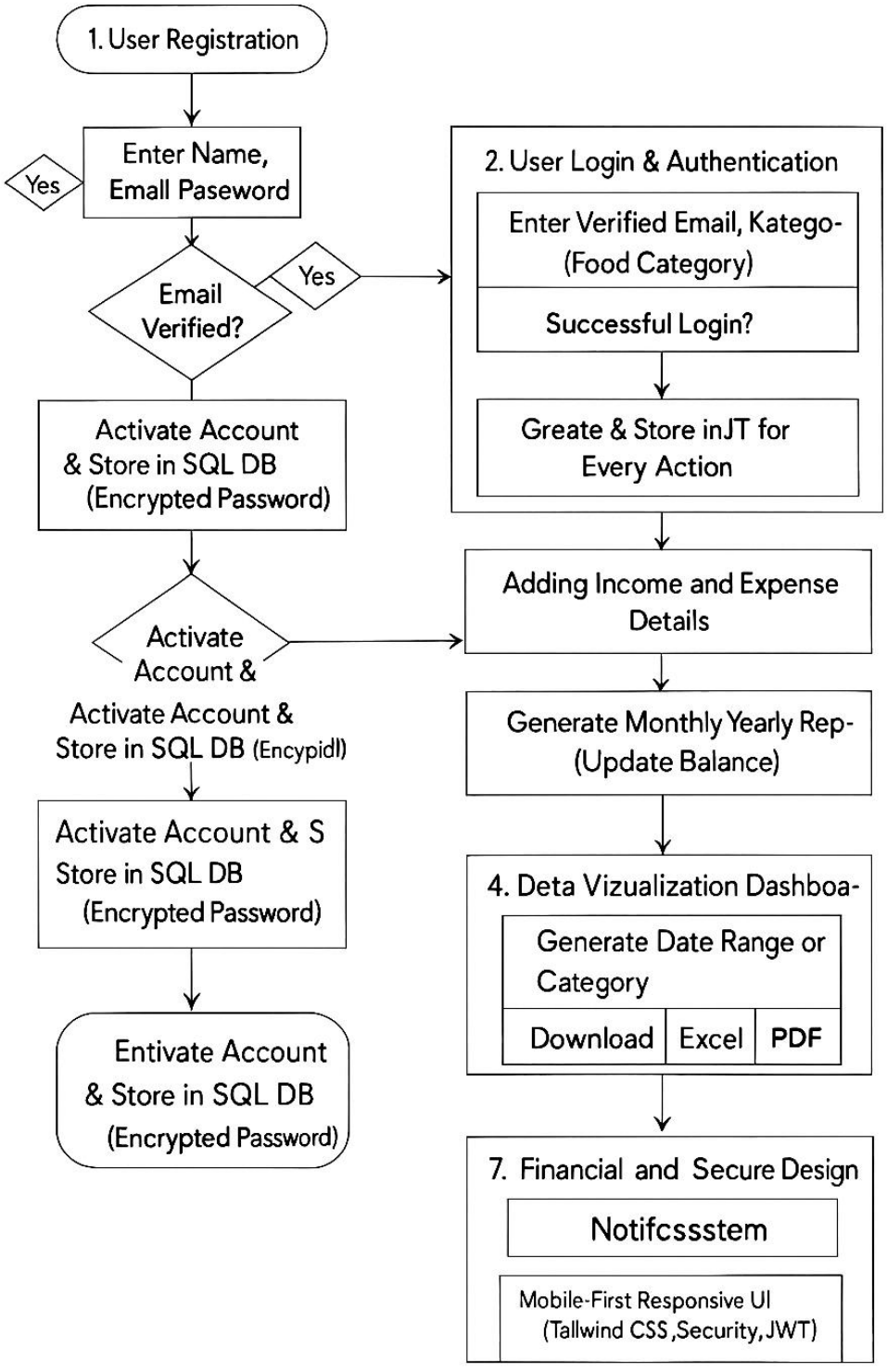
1. Most people manage income and expenses manually using spreadsheets, paper diaries, or basic note apps.
2. These methods are:
   - Time-consuming and prone to human errors.
   - Lacking in data security and automated analysis.
   - Difficult to maintain over time and across multiple devices.
3. Existing financial management apps like Mint, YNAB, and Expense Manager offer automation but have limitations such as:
   - Premium subscriptions required for full functionality.
   - Data privacy concerns due to third-party data storage.
   - Limited customization and category management options.

Overall, current systems fail to provide a secure, customizable, and user-friendly platform for personal finance tracking**.**

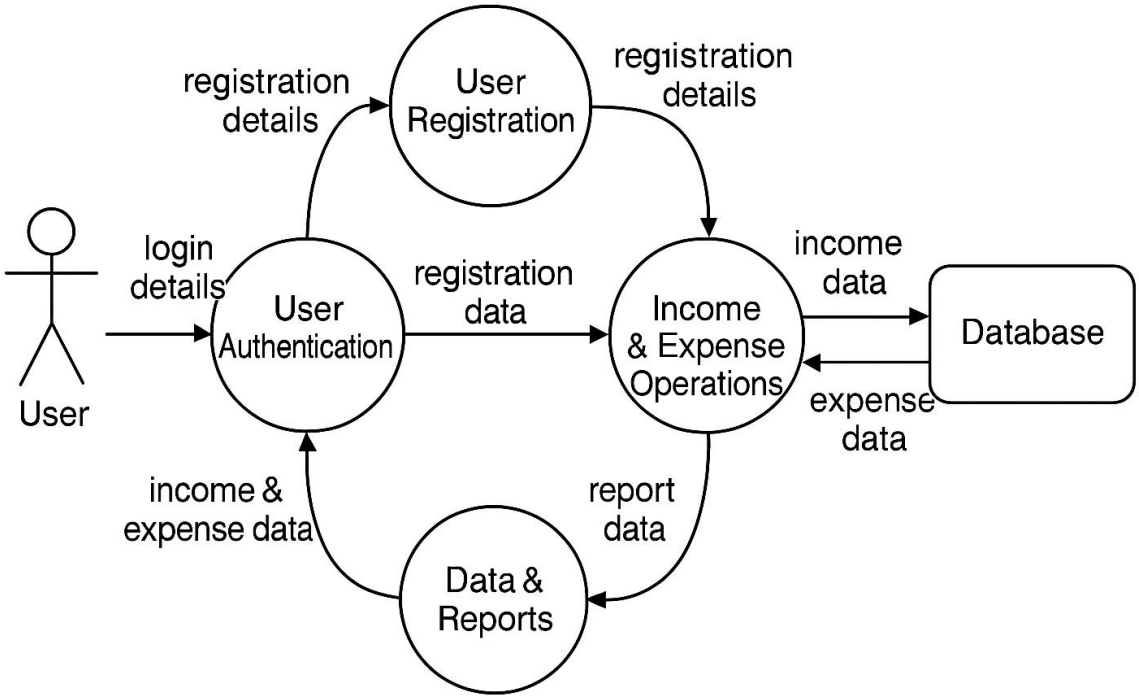
3.2 Proposed System – MyHeaven

1. The **MyHeaven** project introduces a **secure and modern web-based application** for income and expense management.
2. **Key Technologies Used:**
   - **Backend:** Spring Boot
   - **Frontend:** React.js with Tailwind CSS
   - **Database:** MySQL
3. **Core Features:**
   - Secure **JWT-based authentication** for user login and data access.
   - **Add, view, and delete** income and expense records.
   - Real-time data management with **automatic validation**.
   - **Responsive UI** with a clean, user-friendly dashboard.
4. **System Advantages:**
   - Ensures **data privacy and security** with token-based access.
   - Provides **easy customization** and scalability for future features.
   - Offers **better accessibility** and a smooth user experience compared to manual or premium systems.
5. **Future Enhancements:**
   - Report generation and graphical data visualization.
   - AI-based expense prediction and smart budgeting.

3.3 Flow Chart

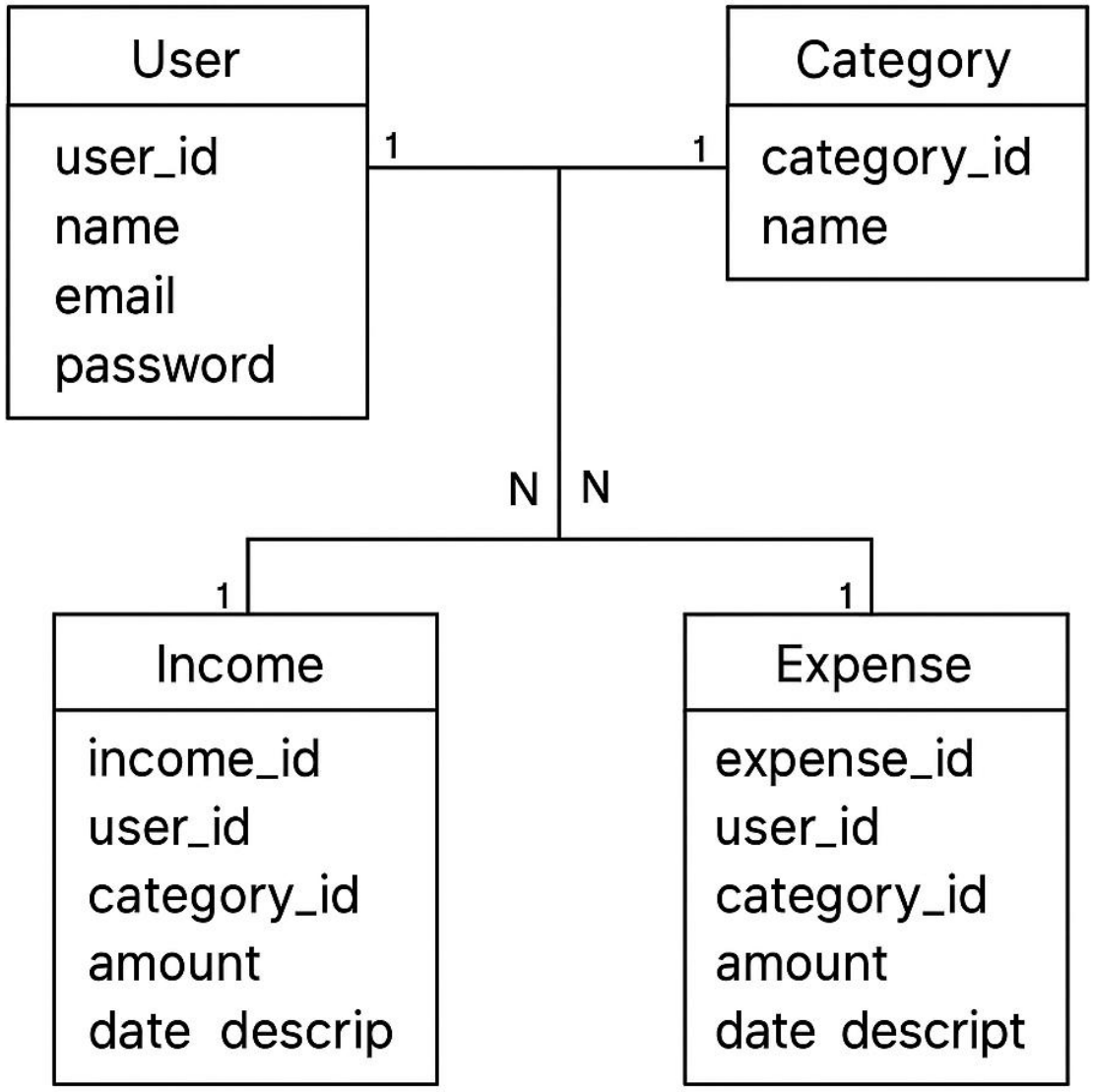


3.4 Data-Flow-Diagram



Personal expense manager system

3.5 E-R Diagram



3.6 Working Principle

1. User Registration (Sign Up)

The working of the system begins with user registration.

- When a new user wants to use the system, they first go to the **Sign-Up page**.
- The user enters details like **Name, Email, and Password**.
- Once the form is submitted, the backend validates all the fields and checks if the email already exists in the database.
- After validation, an **email verification link** is sent to the user's registered email address for authentication.
- Only after successful verification, the user's account is activated and stored securely in the **SQL database**.
- Passwords are encrypted before saving to ensure data security.

2. User Login & Authentication

- After registration, the user can log in using their verified email and password.
- On successful login, the backend generates a **JWT (JSON Web Token)** which is used for session management and authentication.
- This token is sent to the frontend and stored temporarily, allowing the user to stay logged in during their session.
- Each time a user performs any action (like adding income or expenses), this token is verified by the backend to ensure the request is from an authenticated user.
- This process ensures **secure access** and **data privacy** for every user.

3. Category Management

- Once logged in, users can create and manage their own **categories** such as *Food, Travel, Rent, Shopping, Salary, Investment,* etc.
- Categories help in grouping transactions and simplifying financial tracking.
- The backend stores each category with a unique ID linked to the specific user account, ensuring personalized data storage.
- The system allows adding, editing, or deleting categories, and all actions are reflected in real time through API communication between the frontend and backend.

4. Adding Income and Expense Details

- After setting up categories, the user can start recording their financial data.
- The **Add Transaction** section allows users to input details like **amount, date, category, payment mode, and description**.
- Transactions are divided into two main types:
    - **Income:** money received from sources like salary, freelancing, or investments.
    - **Expense:** money spent on items or services such as food, bills, shopping, or entertainment.

- When a transaction is added, the frontend sends the details to the backend API, which validates and stores the record in the SQL database.
- Users can later **edit or delete** these transactions if needed, and the system automatically updates the balance and reports.

5. Data Visualization and Dashboard

- The **dashboard** is one of the core parts of the system. It provides a graphical view of the user's financial summary.
- Using **dynamic charts and graphs**, the system displays comparisons of total income, total expenses, and savings for selected time periods.
- The frontend utilizes **chart libraries** integrated with React.js to present interactive visuals.
- Users can apply filters like **date range** or **category** to view specific insights.
- This helps users identify spending patterns, manage budgets, and make better financial decisions.

6. Financial Reports and Data Export

- The system provides an option to **generate detailed financial reports** summarizing the user's monthly or yearly income and expenses.
- Users can **download** their data in **Excel format** directly from the dashboard.
- Additionally, the system has an **email feature** that automatically sends the report to the user's registered email address.
- These reports improve financial transparency and allow users to maintain offline records of their finances.

7. Notification System

- The system also includes a **notification module** that alerts users about important financial updates, category limits, or monthly summaries.
- Notifications can appear within the dashboard or be sent via email.
- This ensures users stay updated about their spending and can take timely actions to manage their budgets.

8. Responsive and Secure Design

- The entire application is built with a **mobile-first responsive design** using **Tailwind CSS**, making it accessible on all devices.
- The backend ensures **data integrity**, **role-based access**, and **secure APIs** using Spring Security and JWT tokens.
- The combination of React's fast rendering and Spring Boot's strong backend framework provides a smooth and secure user experience.

CHAPTER 4 : COMPONENTS / TOOLS / TECHNOLOGY DETAILS

The MyHeaven – Personal Finance Management System web application is developed using a combination of modern frontend and backend technologies to ensure efficiency, scalability, and a smooth user experience.
The system consists of three main components — Frontend, Backend, and Database — each playing a crucial role in providing an interactive and secure environment for managing personal finances.

4.1 Front-End

The frontend is responsible for the **User Interface (UI)** that allows users to interact with the system — including registration, login, adding transactions, viewing dashboards, and generating reports. It is **fully responsive**, ensuring a consistent experience across mobile, tablet, and desktop devices.

Technology	Purpose
React.js	Builds dynamic and component-based UI for faster rendering and better performance.
Tailwind CSS	Provides utility-first styling for responsive layouts and modern design aesthetics.
Chart.js / Recharts	Used to display interactive graphs and charts for visual financial analytics.
Axios	Handles HTTP requests and connects the frontend with backend APIs.
JavaScript (ES6)	Adds interactivity and logic for user actions, validations, and API responses.
HTML5	Defines the structure and layout of UI components like forms, buttons, and tables.
CSS3	Handles visual design elements including colors, spacing, animations, and layout styling.

Features Developed in Front-End:

- User-friendly dashboard with real-time data visualization
- Add/Edit/Delete transactions
- Responsive navigation menu
- Graphs and charts for income vs. expense trends
- Excel report download and email features

4.2 Back-End

The backend handles all the core business logic, authentication, and database operations. It ensures data integrity, user security, and seamless communication between the frontend and database**.**

Technology	Purpose
Spring Boot (Java)	Core backend framework used to build RESTful APIs and handle business logic.
Spring Security + JWT	Manages user authentication, authorization, and secure API access.
Hibernate / JPA	Simplifies database operations such as saving, updating, and fetching data.
Maven	Used for project management and dependency handling.
Java Mail API	Sends financial summary reports and notifications via email.
Lombok	Reduces boilerplate code for models and entities (getters/setters, constructors).

Features Developed in Back- End:

- User registration and secure login with JWT
- Category and transaction management APIs
- Automated report generation and email delivery
- Data filtering, sorting, and analytics computation
- Notification and alert system

4.3 Database

The database stores all financial records securely, ensuring organized and efficient data retrieval.

Technology	Purpose
MySQL / SQL Database	Stores user, category, and transaction details in relational tables.
JPA / Hibernate ORM	Maps Java entities to database tables for easy CRUD operations.
SQL Queries	Used for data retrieval, filtering, and generating financial summaries.

Features Developed in Database:

- User data storage with encrypted credentials
- Categorized income and expense records
- Real-time transaction updates
- Data backup and integrity maintenance

4.4 Additional Tools

Tool / Platform	Purpose / Usage
IntelliJ IDEA	Used as the primary IDE for developing and testing the backend (Spring Boot APIs). It provides powerful debugging, code completion, and build management features**.**
SQL Workbench	Utilized for managing the MySQL database — creating tables, writing SQL queries, and verifying stored data.
Postman	Used for testing and validating RESTful APIs between frontend and backend to ensure correct data transfer.

CHAPTER 5 : IMPLEMENTATION

The implementation phase of the MyHeaven – Personal Finance Management System focuses on the practical realization of all planned features and modules.
This phase involves developing backend APIs, configuring authentication, integrating frontend interfaces, connecting the database, and finally deploying the complete application.
Each step was implemented systematically to ensure proper functionality, scalability, and security of the system**.**

5.1 Project Introduction and Demo

The project implementation begins with an overview and setup of the MyHeaven system — defining project objectives, structure, and development environment.
The demo illustrates the system workflow, including user registration, login, income–expense management, and dashboard visualization.

5.2 Backend Implementation

a. Register API with Email Configuration

- Implemented the user registration API in Spring Boot.
- Configured SMTP email service to send verification links to users after signup.
- On successful verification, user details are securely stored in the database.
- Passwords are encrypted before saving to ensure account security.

b. Login API with JWT Authentication

- Developed the login endpoint that validates user credentials.
- Integrated JWT (JSON Web Token) for secure user authentication and session management.
- Tokens are used for all further user requests to verify identity and maintain access control.

c. Category API

- Designed APIs for adding, editing, deleting, and viewing categories (e.g., Food, Rent, Salary, Bills).
- Each category is linked to the authenticated user, ensuring data isolation and personalization**.**

d. Income and Expense API

- Implemented backend endpoints for managing income and expense transactions.
- Added validation for category selection, transaction amount, and date.
- The API automatically updates the total balance and stores transaction details in the SQL database.

e. Dashboard API

- Developed APIs to fetch aggregate financial data for income, expenses, and savings.
- The data is processed and sent to the frontend for visualization using charts and graphs.
- Provides real-time insights into financial status and monthly trends**.**

f. Filter API

- Added filtering features allowing users to search transactions based on category, date range, or amount.
- This helps in easy tracking of financial records and analysis of specific periods.

g. Notifications API

- Implemented notification functionality that alerts users about monthly summaries, expense limits, and updates.
- Notifications are generated automatically and sent via email using Java Mail API.

h. Initial Backend Deployment

- Deployed the backend to a local server for testing purposes.
- Verified all REST APIs using Postman to ensure correct data flow and response handling.
- Successfully connected the backend with the SQL database through JPA/Hibernate.

5.3 Frontend Implementation

a. Setting up React Project

- Initialized the React.js project structure and configured required dependencies (React Router, Axios, Tailwind CSS, Chart libraries).
- Established API connections with the backend using Axios.

b. Frontend Signup Module

- Developed the registration page where users can enter details and receive email verification.
- Integrated form validation and connected it to the backend Register API.

c. Frontend Login Module

- Built the login component for user authentication using the backend JWT API.
- On successful login, JWT tokens are stored securely for user session management.

d. Upload Profile Picture

- Added feature to upload and update user profile pictures for personalization.
- Integrated file upload functionality through REST API.

e. Menubar and Sidebar Setup

- Designed and implemented the navigation system including menubar and sidebar for smooth user navigation between modules like Dashboard, Income, Expense, and Categories.

f. Category Module

- Developed the UI for managing categories — add, edit, and delete options.
- Connected it with the Category API to ensure real-time updates.

g. Income Module

- Created forms and tables to record income details.
- Integrated with Income API for data submission and retrieval.

h. Download Excel and Email API Integration

- Backend: Developed functionality to export financial data to Excel and send it via email.
- Frontend: Integrated these features allowing users to download or receive reports directly

from the dashboard.

i. Expense Module

- Implemented the UI for adding, editing, and deleting expense records.
- Connected to backend APIs for data synchronization and automatic updates in total balance.

j. Filter Module

- Developed frontend filters for viewing transactions by category, amount, or date.
- Ensures efficient financial tracking and analytical comparisons.

k. Dashboard Module

- Created an interactive dashboard using Chart.js/Recharts to visualize income vs. expense trends.
- Displays dynamic graphs, statistics, and total savings in real time.
- Enhances user understanding of their financial behavior through visual analytics**.**

5.4 Final Testing and Integration

After completing both backend and frontend implementations, the entire system was tested thoroughly**.**

- Postman was used for API testing.
- SQL Workbench was used to verify database transactions.
- The frontend and backend were fully integrated and deployed successfully for demo purposes**.**

CODING

ProfileEntity.java

```
package com.gargi.myheaven.entity;

import jakarta.persistence.*;
import lombok.AllArgsConstructor;
import lombok.Builder;
import lombok.Data;
import lombok.NoArgsConstructor;
import org.hibernate.annotations.CreationTimestamp;
import org.hibernate.annotations.UpdateTimestamp;

import java.time.LocalDateTime;

@Entity
@Table(name="tbl_profiles")
@Data
@AllArgsConstructor
@NoArgsConstructor
@Builder
public class ProfileEntity {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private String fullName;
    @Column(unique = true)
    private String email;
    private String password;
    private String profileImageUrl;
    @Column(updatable = false)
    @CreationTimestamp
    private LocalDateTime createdAt;
    @UpdateTimestamp
    private LocalDateTime updatedAt;
    private Boolean isActive;
    private String activationToken;

    public void prePersist(){
        if(this.isActive == null){
            isActive = false;
        }
    }


}
```

ProfileService.java

```
package com.gargi.myheaven.service;

import com.gargi.myheaven.dto.AuthDto;
import com.gargi.myheaven.dto.ProfileDto;
import com.gargi.myheaven.entity.ProfileEntity;
import com.gargi.myheaven.repository.ProfileRepository;
import com.gargi.myheaven.util.JwtUtil;
import lombok.RequiredArgsConstructor;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.dao.DataIntegrityViolationException;
import org.springframework.security.authentication.AuthenticationManager;
import org.springframework.security.authentication.UsernamePasswordAuthenticationToken;
import org.springframework.security.core.Authentication;
import org.springframework.security.core.context.SecurityContextHolder;
import org.springframework.security.core.userdetails.UsernameNotFoundException;
import org.springframework.security.crypto.password.PasswordEncoder;
import org.springframework.stereotype.Service;

import java.util.Map;
import java.util.Optional;
import java.util.UUID;

@Service
@RequiredArgsConstructor
public class ProfileService {
    private final ProfileRepository profileRepository;
    private final EmailService emailService;
    private final PasswordEncoder passwordEncoder;
    private final AuthenticationManager authenticationManager;
    private final JwtUtil jwtUtil;

    @Value("${app.activation.url}")
    private String activationURl;

    public ProfileDto registerProfile(ProfileDto profileDto) {
        try {
            // Check if email already exists
            Optional<ProfileEntity> existing =
```

```
profileRepository.findByEmail(profileDto.getEmail());
        if (existing.isPresent()) {
            throw new RuntimeException("Email already registered!");
        }

        // Save new profile
        ProfileEntity newProfile = toEntity(profileDto);
        newProfile.setActivationToken(UUID.randomUUID().toString());
       //newProfile.setPassword(passwordEncoder.encode(newProfile.getPassword()));
        newProfile = profileRepository.save(newProfile);

        // Send activation email
        String activationLink =activationURl+ "/api/v1.0/activate?token=" +
newProfile.getActivationToken();
        String subject = "Activate your MyHeaven account";
        String body = "Click the link to activate your MyHeaven account: " + activationLink;

        try {
            emailService.sendEmail(newProfile.getEmail(), subject, body);
        } catch (Exception e) {
            System.err.println(" Failed to send email: " + e.getMessage());
        }

        return toDto(newProfile);

    } catch (DataIntegrityViolationException e) {
        throw new RuntimeException("Duplicate email not allowed!");
    } catch (Exception e) {

        throw new RuntimeException("Registration failed: " + e.getMessage());
    }
  }

  public ProfileEntity toEntity(ProfileDto profileDto) {
    return ProfileEntity.builder()
            .id(profileDto.getId())
            .fullName(profileDto.getFullName())
            .email(profileDto.getEmail())
            .password(passwordEncoder.encode(profileDto.getPassword()))
            .profileImageUrl(profileDto.getProfileImageUrl())
            .createdAt(profileDto.getCreatedAt())
```

```
                .updatedAt(profileDto.getUpdatedAt())
                .build();
    }

    public ProfileDto toDto(ProfileEntity profileEntity) {
        return ProfileDto.builder()
                .id(profileEntity.getId())
                .fullName(profileEntity.getFullName())
                .email(profileEntity.getEmail())
                .profileImageUrl(profileEntity.getProfileImageUrl())
                .createdAt(profileEntity.getCreatedAt())
                .updatedAt(profileEntity.getUpdatedAt())
                .build();
    }
    public boolean activateProfile(String activationToken) {
        return profileRepository.findByActivationToken(activationToken)
                .map(profile->{
                    profile.setIsActive(true);
                    profile.setActivationToken(null); //clear the activation token
                    profileRepository.save(profile);
                    return true;
                })
                .orElse(false);
    }

    public boolean isAccountActive(String email){
        return profileRepository.findByEmail(email)
                .map(ProfileEntity::getIsActive)
                .orElse(false);
    }

    public ProfileEntity getCurrentProfile(){
        Authentication authentication= SecurityContextHolder.getContext().getAuthentication();
        return profileRepository.findByEmail(authentication.getName())
                .orElseThrow(() -> new UsernameNotFoundException("profile not found via email :"
+authentication.getName()));
    }

    public ProfileDto getPublicProfile(String email){
        ProfileEntity currentUser=null;
        if(email==null){
```

```
            currentUser= getCurrentProfile();
        }else{
            currentUser=profileRepository.findByEmail(email)
                    .orElseThrow(() -> new UsernameNotFoundException("profile not found via email
:" +email));
        }
        return ProfileDto.builder()
                .id(currentUser.getId())
                .fullName(currentUser.getFullName())
                .email(currentUser.getEmail())
                .profileImageUrl(currentUser.getProfileImageUrl())
                .createdAt(currentUser.getCreatedAt())
                .updatedAt(currentUser.getUpdatedAt())
                .build();
    }

    public Map<String, Object> authenticateAndGenerateToken(AuthDto authDto) {
        try{
            authenticationManager.authenticate(new
UsernamePasswordAuthenticationToken(authDto.getEmail(), authDto.getPassword()));
            //generate jwt token
            String token= jwtUtil.generateToken(authDto.getEmail());
            return Map.of("token",token,
                    "user" ,getPublicProfile(authDto.getEmail()));

        }catch (Exception e){
            throw new RuntimeException("Invalid email or password!");
        }
    }
}
```

ProfileController.java

```
package com.gargi.myheaven.controller;

import com.gargi.myheaven.dto.AuthDto;
import com.gargi.myheaven.dto.ProfileDto;
import com.gargi.myheaven.repository.ProfileRepository;
import com.gargi.myheaven.service.ProfileService;
import lombok.RequiredArgsConstructor;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
```

```
import org.springframework.web.bind.annotation.*;

import java.util.Collections;
import java.util.Map;

@RestController
@RequiredArgsConstructor

public class ProfileController {
   private final ProfileService profileService;

   @PostMapping("/register")
   public ResponseEntity<ProfileDto> registerProfile(@RequestBody ProfileDto profileDto) {
      ProfileDto registeredProfile = profileService.registerProfile(profileDto);
      return ResponseEntity.status(HttpStatus.CREATED).body(registeredProfile);
   }

   @GetMapping("/activate")
   public ResponseEntity<String> activateProfile(@RequestParam String token) {
      boolean isActivated = profileService.activateProfile(token);
      if (isActivated) {
         System.out.println("Activated profile");
         return ResponseEntity.ok("Activated successfully");
      }
      else {
         return ResponseEntity.status(HttpStatus.NOT_FOUND).body("Invalid activation
Token");
      }
   }

   @PostMapping("/login")
   public ResponseEntity<Map<String , Object>> login(@RequestBody AuthDto authDto) {
      try{
         if(!profileService.isAccountActive(authDto.getEmail())) {
            return ResponseEntity.status(HttpStatus.FORBIDDEN).body(Map.of(
                  "message","Account is not active. Please try again."
            ));
         }
         Map<String,Object> response=profileService.authenticateAndGenerateToken(authDto);
         return ResponseEntity.ok(response);

      } catch (Exception e) {
         return ResponseEntity.status(HttpStatus.BAD_REQUEST).body(Map.of(
               "message",e.getMessage()
         ));
      }
   }
```

```
    @GetMapping("/profile")
    public ResponseEntity<ProfileDto> getPublicProfile(){
        ProfileDto profileDto= profileService.getPublicProfile(null);
        return ResponseEntity.ok(profileDto);
    }



}
```

ProfileRepository.java

```
package com.gargi.myheaven.repository;

import com.gargi.myheaven.entity.ProfileEntity;
import org.springframework.data.jpa.repository.JpaRepository;

import java.util.Optional;

public interface ProfileRepository extends JpaRepository<ProfileEntity,Long> {

    //select * from tbl_profiles where email=?
    Optional<ProfileEntity> findByEmail(String email);

    //select * from tbl_profiles where activation_token=?
    Optional<ProfileEntity> findByActivationToken(String activationToken);
}
```

ProfileDto.java

```
package com.gargi.myheaven.dto;

import jakarta.persistence.Column;
import lombok.AllArgsConstructor;
import lombok.Builder;
import lombok.Data;
import lombok.NoArgsConstructor;
import org.hibernate.annotations.CreationTimestamp;
import org.hibernate.annotations.UpdateTimestamp;
```

```
import java.time.LocalDateTime;

@Data
@AllArgsConstructor
@NoArgsConstructor
@Builder
public class ProfileDto {
   private Long id;
   private String fullName;
   private String email;
   private String password;
   private String profileImageUrl;
   private LocalDateTime createdAt;
   private LocalDateTime updatedAt;
}
```

Main Method

```
package com.gargi.myheaven;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.scheduling.annotation.EnableScheduling;

@EnableScheduling
@SpringBootApplication
public class MyheavenApplication {

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

}
```

Signup.jsx

```
import {useState} from "react";

import {assets} from "../assets/assets.js";
import Input from "../components/Input.jsx";
```

```
import { Link, useNavigate } from "react-router-dom";
import axiosConfig from "../util/axiosConfig.jsx";
import {API_ENDPOINTS} from "../util/apiEndpoints.js";
import toast from "react-hot-toast";
import {validateEmail} from "../util/validation.jsx";
import {LoaderCircle} from "lucide-react";
import ProfilePhotoSelector from "../components/ProfilePhotoSelector.jsx";
import uploadProfileImage from "../util/uploadProfileImage.js";



const Signup=()=>{

    const [fullName, setFullName] = useState("");
    const [email, setEmail] = useState("");
    const [password, setPassword] = useState("");
    const [error, setError] = useState(null);
    const [isLoading, setIsLoading] = useState(false);
    const [profilePhoto, setProfilePhoto] = useState(null);


    const navigate =useNavigate();


    const handleSubmit = async (e) => {
        e.preventDefault();
        let profileImageUrl = "";
        setIsLoading(true);

        //basic validation
        if (!fullName.trim()) {
            setError("Please enter your fullname");
            setIsLoading(false);
            return;
        }

        if (!validateEmail(email)) {
            setError("Please enter valid email address");
            setIsLoading(false);
            return;
        }

        if (!password.trim()) {
            setError("Please enter your password");
            setIsLoading(false);
            return;
        }
```

```
    setError("");
    //signup api call
    try {
      //upload image if present
      if(profilePhoto){
        const imageUrl=await uploadProfileImage(profilePhoto);
        profileImageUrl = imageUrl || "";

      }

      const response = await axiosConfig.post(API_ENDPOINTS.REGISTER, {
        fullName,
        email,
        password,
        profileImageUrl

      })
      if (response.status === 201) {
        toast.success("Profile created successfully.");
        navigate("/login");
      }
    } catch(err) {
      console.error('Something went wrong', err);
      setError(err.message);
    } finally {
      setIsLoading(false);
    }
  }

  return (

    <div className="w-full h-screen relative flex items-center justify-center">
      {/* Background Image */}
      <img
        src={assets.login_bg}
        alt="Background"
        className="absolute inset-0 w-full h-full object-cover "
        style={{ filter: "blur(2px)" }}/>
      <div className="relative z-10 w-full max-w-lg px-6 " >

        <div className="bg-opacity-95 backdrop-blur-sm rounded-lg shadow-2xl p-8 max-h-
[90vh] overflow-y-auto" style={{ backgroundColor: "#E3C9F8" }} >
          <h3 className="text-2xl font-semibold text-black text-center mb-2">
            Create An Account
          </h3>
          <p className="text-sm text-700 text-center mb-8 text-black" >
```

```
            Start tracking your spendings by joining with us.
          </p>
          <form onSubmit={handleSubmit} className="space-y-4">
            <div className="flex justify-center mb-6">
              <ProfilePhotoSelector image={profilePhoto} setImage={setProfilePhoto}/>
            </div>
            <div className="grid grid-cols-2 md:grid-cols-2 gap-4">
              <Input
                value={fullName}
                onChange={(e) => setFullName(e.target.value)}
                label="Full Name"
                placeholder="Enter full name"
                type="text"
              />

              <Input
                value={email}
                onChange={(e) => setEmail(e.target.value)}
                label="Email address"
                placeholder="Enter your email address"
                type="text"
              />

              <div className="col-span-2">
                <Input
                  value={password}
                  onChange={(e) => setPassword(e.target.value)}
                  label="Password"
                  placeholder=" enter your secure password"
                  type="password"
                />
              </div>
            </div>
            {error && (
              <p className="text-red-800 text-sm text-center bg-red-50 p-2 rounded">
                {error}
              </p>
            )}
            <button disabled={isLoading} className={`btn-primary w-full py-3 text-lg font-
medium flex items-center justify-center gap-2 ${isLoading ? 'opacity-60 cursor-not-allowed':
"}`} type="submit">
              {isLoading ? (
                <>
                  <LoaderCircle className="animate-spin w-5 h-5" />
                  Signing Up...
                </>
              ): (
```

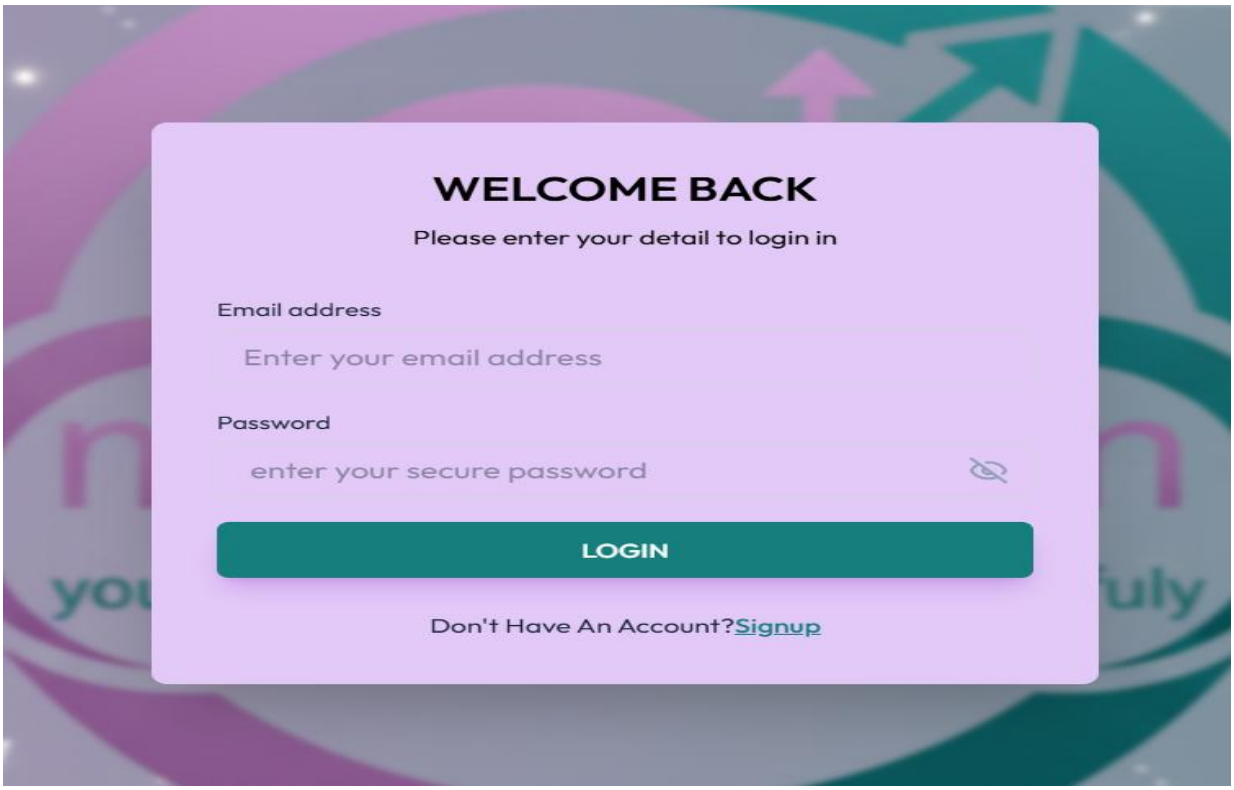
```
                        "SIGN UP"
                      )}
                    </button>
                    <p className="text-sm text-slate-800 text-center mt-6">
                      Already have an account?
                      <Link to="/login" className="font-medium text-primary underline hover:text-
primary-dark transition-colors" style={{ color: "#167F7D" }}>Login</Link>
                    </p>
                  </form>
                </div>
        </div>
        </div>

    )
}

export default Signup;
```

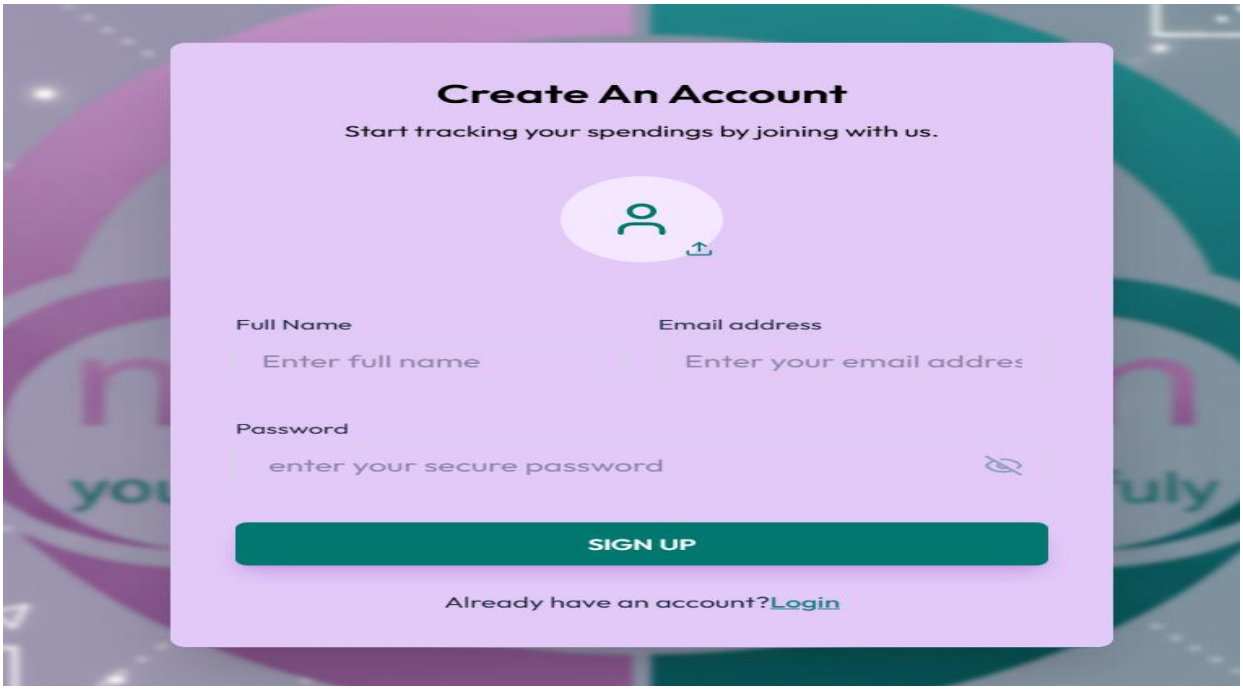
Assets.js

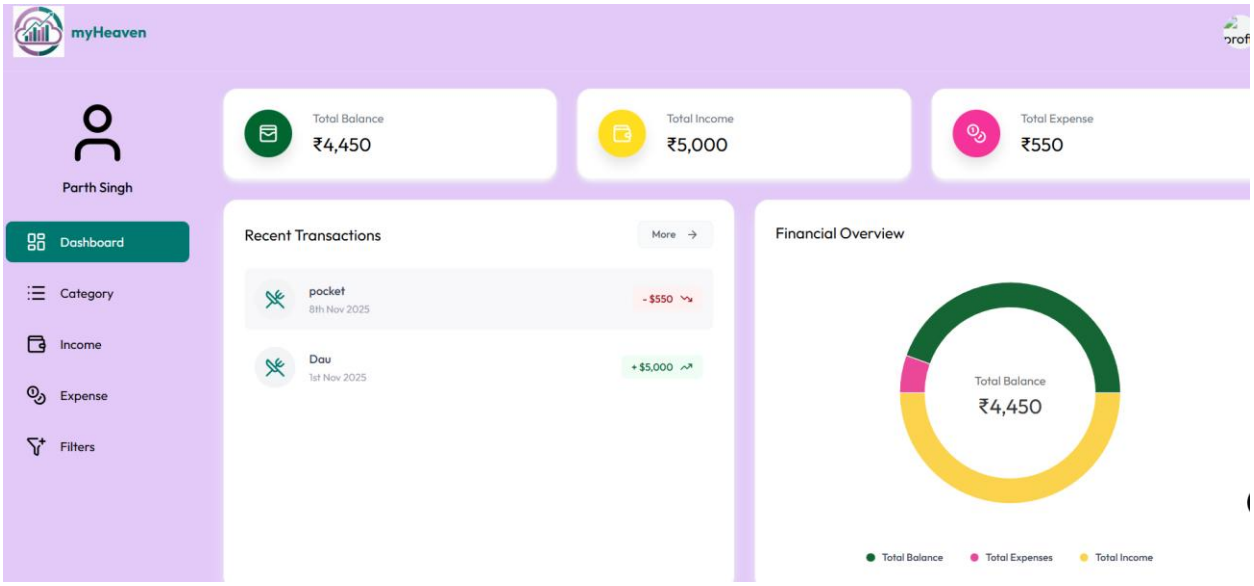
```
import logo_myheaven from "./final1.png";
import login from "./login.png";
import bk from "./bk.jpg";
import {Coins, FunnelPlus, LayoutDashboard, List, Wallet} from "lucide-react"; // agar use
karna ho

export const assets = {
    logo: logo_myheaven,
    login_bg: login,
};
export const SIDE_BAR_DATA = [
    {
        id: "01",
        label: "Dashboard",
        icon: LayoutDashboard,
        path: "/dashboard",
    },
    {
        id: "02",
        label: "Category",
        icon: List,
        path: "/category",
    },
    {
        id: "03",
```

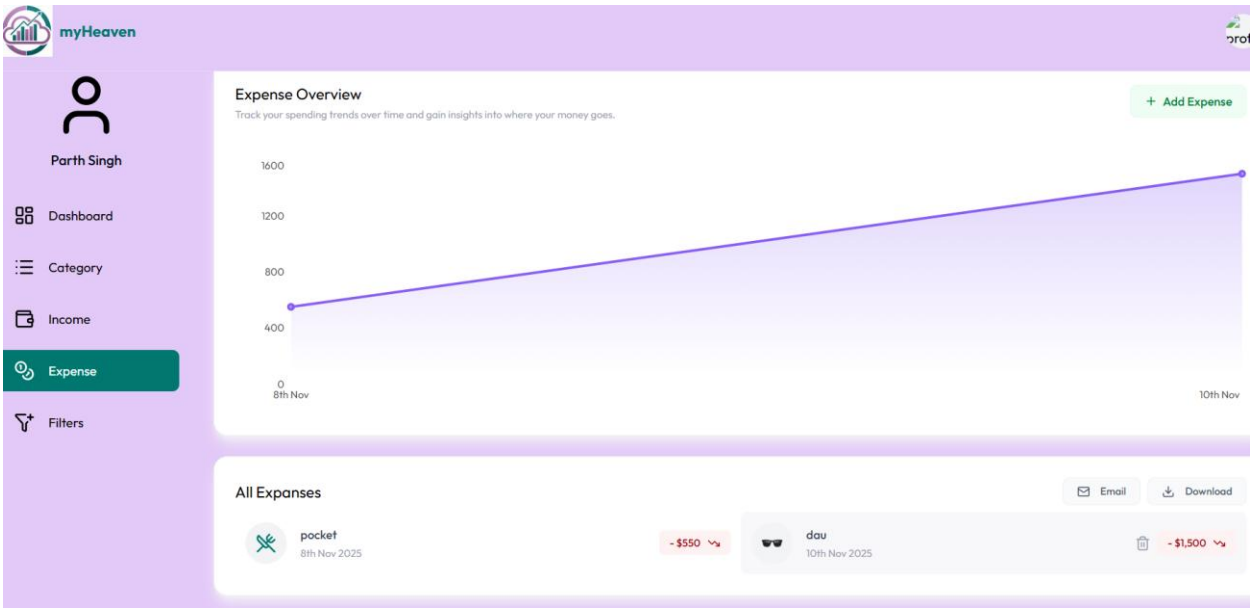
```
    label: "Income",
    icon: Wallet,
    path: "/income",
  },
  {
    id: "04",
    label: "Expense",
    icon: Coins,
    path: "/expense",
  },
  {
    id: "05",
    label: "Filters",
    icon: FunnelPlus,
    path: "/filter",
  },
];
```

CHAPTER 6 : OUTPUT IMAGES

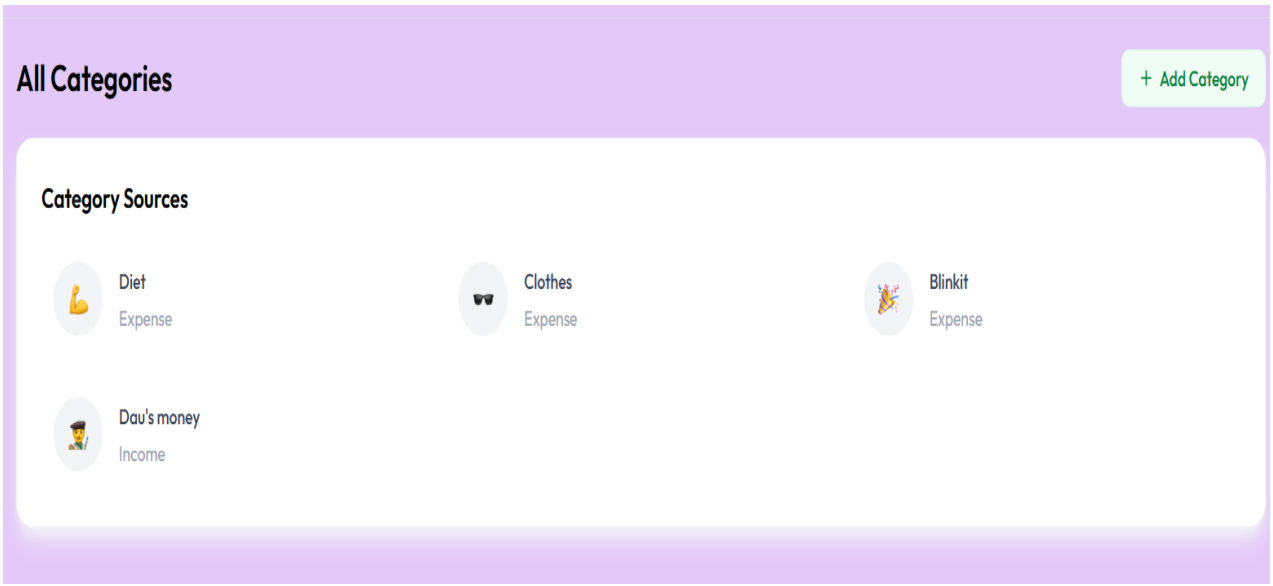





myHeaven
Parth Singh
Dashboard
Category
Income
Expense
Filters
Total Balance
₹4,450
Total Income
₹5,000
Total Expense
₹550
Recent Transactions
More
pocket
8th Nov 2025
- $550
Dau
1st Nov 2025
+ $5,000
Financial Overview
Total Balance
₹4,450
Total Balance
Total Expenses
Total Income


myHeaven
Parth Singh
Dashboard
Category
Income
Expense
Filters
Expense Overview
Track your spending trends over time and gain insights into where your money goes.
+ Add Expense
1600
1200
800
400
0
8th Nov
10th Nov
All Expanses
Email
Download
pocket
8th Nov 2025
- $550
dau
10th Nov 2025
- $1,500


Filter Transactions
Select the filters
Type
Income
Start Date
dd-mm-yyyy
End Date
dd-mm-yyyy
Sort Field
Date
Sort Order
Ascending
Search
Search...
Transactions
Select the filters and click apply to filter the transactions


All Categories
+ Add Category
Category Sources
Diet
Expense
Clothes
Expense
Blinkit
Expense
Dau's money
Income

CHAPTER 7 : FUTURE SCOPE AND CONCLUSION

7.1 CONCLUSION

The **MyHeaven – Personal Finance Management System** has been designed as a comprehensive platform to manage personal finances, but there is significant potential for further enhancement and expansion. The following points outline the future improvements that can make the system more efficient and user-friendly:

1. **Mobile Application Development:**
    - A dedicated Android and iOS mobile app can be developed to make the system more accessible and convenient for users on the go.
2. **AI-Based Expense Prediction:**
    - Implementing **machine learning models** can help predict future expenses, detect spending patterns, and provide personalized financial advice.
3. **Budget Planning and Goal Tracking:**
    - The addition of goal-based budgeting features will allow users to set financial targets, such as saving for travel, education, or investments, and monitor their progress.
4. **Voice and Chatbot Integration:**
    - Integration of a **chatbot** or **voice assistant** can help users quickly access their financial data or add transactions using natural language commands.
5. **Multi-User and Family Accounts:**
    - Enabling shared or family-based accounts can allow multiple users to track collective expenses and contributions within a single dashboard.
6. **Bank Integration & Auto Sync:**
    - Future versions can include APIs for **bank account synchronization**, allowing automatic transaction imports and real-time balance updates.
7. **Advanced Analytics and Insights:**
    - Enhanced visualization tools and deep analytics can provide predictive insights, spending trends, and customized saving recommendations.
8. **Cloud Deployment:**
    - Hosting the system on cloud platforms such as AWS or Azure will ensure scalability, faster access, and better data security.

7.2 FUTURE SCOPE

The **MyHeaven – Personal Finance Management System** successfully addresses the challenges individuals face in managing their daily income and expenses.
It provides users with a **secure, responsive, and interactive platform** to track their financial

activities efficiently. With features like **user authentication, dynamic dashboards, category management, real-time analytics, Excel report generation, and notifications**, the system ensures transparency and helps users maintain financial discipline.
The combination of **React.js (frontend)**, **Spring Boot (backend)**, and **SQL (database)** makes the system robust and scalable.

Through this project, the practical implementation of full-stack development concepts — such as API integration, authentication using JWT, responsive UI design, and data visualization — has been successfully demonstrated.

In conclusion, **MyHeaven** serves as a valuable tool for personal financial management and lays a strong foundation for future enhancements that could transform it into a complete smart finance assistant.

CHAPTER 8 : REFERENCE

- **React.js Documentation** – Official React documentation for building dynamic and component-based user interfaces.
  https://react.dev/
- **Tailwind CSS Documentation** – Utility-first CSS framework used for responsive and modern UI design.
  https://tailwindcss.com/
- **Spring Boot Reference Guide** – Official guide for developing RESTful backend APIs and microservices.
  https://spring.io/projects/spring-boot
- **Spring Security and JWT Authentication** – For implementing secure login, signup, and token-based authorization.
  https://spring.io/guides/topicals/spring-security-architecture
- **MySQL Documentation** – Official database reference for relational schema design, queries, and data management.
  https://dev.mysql.com/doc/
- **Chart.js / Recharts** – Libraries used for creating dynamic and interactive charts for financial data visualization.
  https://www.chartjs.org/
  https://recharts.org/
- **Axios Library** – For handling HTTP requests and communication between frontend and backend.
  https://axios-http.com/
- **Postman API Platform** – Used for testing, debugging, and validating backend APIs.
  https://www.postman.com/
- **IntelliJ IDEA** – Integrated development environment used for backend development and project management.
  https://www.jetbrains.com/idea/
- **SQL Workbench** – Tool for database management, SQL query execution, and schema testing.
  http://www.sql-workbench.eu/
- **Bravo Studio (Bravo.io)** – Used for creating and previewing responsive frontend UI layouts.
  https://www.bravostudio.app/
- **Java Mail API** – Used for sending email notifications and financial reports from the backend.
  https://javaee.github.io/javamail/