

AI FINANCE PLATFORM

Thesis Submitted in
partial fulfillment for the award of

Bachelor of Computer Application

Submitted by

1. 2204030102153: Parate Dhvanit
2. 2204030102835: Zapadiya Rahul
3. 2204030102644: Solanki Meet

Department of Computer Application

Under the supervision of

Mrs. Vaishali Patel

Assistant professor, Computer Application



**Silver Oak University
Gota, Ahmedabad-382481,
Gujarat, India**

April-2025

CERTIFICATE

It is certified that the project titled **AI Finance Platform** has been carried in group are bonafide students of BCA VI semester, Department of Computer Application, Silver Oak University, Ahmedabad, Gujarat. The project is a record of the work accomplished during the sixth semester of BCA as a partial fulfillment of the requirement for the award of degree in Bachelor of Computer Applications (BCA), under my guidance.

Name of students	Enrollment of student
-------------------------	------------------------------

Parate Dhvanit:	2204030102153
-----------------	---------------

Zapadiya Rahul :	2204030102835
------------------	---------------

Solanki Meet :	2204030102644
----------------	---------------

Mrs. Vaishali Patel, Assistant Professor,

Department of Computer Application,

Silver Oak University, Ahmedabad, Gujarat.

DECLARATION

We hereby declare that project entitled **Ai Finance Platform** in partial fulfillment for the degree of **Bachelor of Computer Application** to **Silver Oak University**, Ahmedabad, is bonafide record of the project work carried out at **SILVER OAK COLLEGE OF COMPUTER APPLICATION** under the supervision of **Mrs. Vaishali Patel** and that no part of any of these reports has been directly copied from any student's report or taken from any other sources, without providing due reference.

Name of students	Sign of student
-------------------------	------------------------

Parate Dhvanit	
----------------	--

Rahul zapdiya	
---------------	--

Solanki Meet	
--------------	--

ACKNOWLEDGEMENT

We would like to express our profound sense of deepest gratitude to our guide **Mrs. Vaishali Patel**, Silver Oak College of Computer Application (SOCCA), Ahmedabad for valuable guidance, sympathy and co-operation for providing necessary facilities and sources during the entire period of this project.

We wish to convey our sincere gratitude to all the faculties of Department of Computer Application who have enlightened us during our studies. The facilities and cooperation received from the technical staff of department is thankfully acknowledged.

The main credit for the successful completion of this project can be given to Principal of our college Dr. Hemal Patel who supported us throughout this enterprise.

We express our thanks to all those who helped us in one way or other.

Last, but not least, we would like to thank the authors of various research articles and books that were referred to.

Name of students	Enrollment of student
Parate Dhvanit	2204030102153
Zapadiya Rahul	2204030102835
Solanki Meet	2204030102644

ABSTRACT

This project is a fully responsive, web-based AI Finance platform designed to help users manage their financial transactions and budgets efficiently. Using the Gemini API for receipt scanning, the platform enables users to automatically add transactions by simply uploading images of receipts. Users can create, edit, and manage transactions, set monthly budgets, and receive budget alerts when their spending reaches 80% of the allocated budget. The platform also generates comprehensive monthly financial reports.

Built using a modern tech stack, the frontend is developed with Next.js and React, styled using Tailwind CSS and Shadcn UI components for a clean and mobile-friendly user interface. User authentication is handled securely through Clerk, while Supabase serves as the database for storing user accounts, transaction details, and budget information. Arject provides additional protection for sensitive financial data. The integration of the Gemini API for receipt scanning demonstrates practical application of AI to solve real-world financial management problems. Overall, this project aims to provide a user-friendly, secure, and intelligent financial management experience that simplifies tracking expenses and maintaining budgets.

FIGURES/DIAGRAMS:

	Pg. No
Figure 1: Flow Chart.....	13
Figure 2: ER Diagram	14
Figure 3: Screenshots.....	36

TABLE OF CONTENTS

Certificate from the Guide	i
Declaration from the Student	ii
Acknowledgement	iii
Abstract.....	iv
List of Figures	v
Table of Contents	v

1 Introduction

1.1 Project Profile	1
1.2 Overview	1
1.3 Motivation	2
1.4 Goal	2
1.5 Organization of the Report/Thesis	3

2 Initial System Study	4
2.1 Chapter Introduction	4
2.2 Drawbacks of the existing system	5
2.3 Problem definition	5
2.4 The proposed system	6
2.5 Scope of the system	6
2.6 Scope of this project	7
2.7 System development approach	7
3 Feasibility Analysis	9
3.1 Technical Feasibility	9
3.2 Economic Feasibility	9
3.3 Operational Feasibility	10
4 System Analysis	13
4.1 Introduction	13
4.2 Data flow diagram	13
4.3 Entity relationship diagram	14
4.4 Physical and behavioral aspects of the system	15
4.4.1 Sub section here	15
5 Software Requirements Specifications	16
5.1 General Description	16
5.1.1 Product Perspective	16
5.1.2 Product Functions	16
5.1.3 User Characteristics	17

5.1.4 General Constraints	17
5.1.5 Assumptions and Dependencies	17
5.1.6 Functional Requirements	18
5.1.7 Functional Requirement	18
5.2 External Interface Requirements	18
5.2.1 User Interfaces	19
5.2.2 Hardware Interfaces	19
5.2.3 Software Interfaces	19
5.3 Performance Requirements	19
5.4 Design Constraints	19
5.4.1 Standard Compliance	19
5.4.2 Hardware Constraints	20
5.4.3 Other Requirements	20
5.4.4 Scope of this project	20
6 System design	21
6.1 Introduction	21
6.2 System Architecture	21
6.3 Module Design	21
6.4 Database Design	22
6.5 Input Output Design	25
6.6 Algorithm Design	25
6.7 Electronic Data Communication Design	25

6.8 System Maintenance	26
6.9 Other Alternatives Considered	26
7 System Implementation	27
7.1 Hardware Components	27
7.2 Software Environment	27
7.3 System Development Platform	28
7.4 Project Accomplishment Status	29
7.5 Guidelines for Continuation	29
7.6 Hardware Components	30
7.7 Software Environment	30
8 System Testing	31
8.1 Test Plan	31
8.2 Test Cases	33
9 Conclusion & Future Direction of Work	42
9.1 Conclusion	42
9.2 Future Direction of work	42
Bibliography/Reference	43



1. Introduction

1.1 Project Profile

The project titled "AI Finance Platform" is a comprehensive web-based application developed to provide users with an intelligent financial management system. It leverages the power of Google's Gemini API to automate transaction entry through receipt scanning, while offering robust features for transaction management, budget tracking, and financial reporting. The objective is to deliver a seamless, secure, and scalable platform that addresses the evolving demands of modern users in personal finance management.

This system is not merely a static expense tracker; it is a dynamic application that allows users to manage their finances intelligently. Users can scan receipts to automatically extract transaction details, manually add transactions, edit existing entries, and set monthly budget limits. The system provides timely alerts when spending approaches budget thresholds and generates comprehensive monthly financial reports.

Furthermore, the platform incorporates secure authentication via Clerk, reliable data storage through Supabase, and enhanced protection with Arject. The modern and intuitive interface, built using Next.js, React, and Tailwind CSS with Shadcn UI components, ensures a consistent and pleasant user experience across all devices.

1.2 Overview

The rapid digital transformation in personal finance management, coupled with the increasing demand for automated tools, has given rise to intelligent platforms that simplify financial tracking. This project aims to create a sophisticated yet user-friendly system that leverages AI to reduce manual data entry while providing robust financial management capabilities.

The developed web application serves as a digital bridge between property owners seeking to monetize unused spaces and travelers in need of temporary accommodation. Its modular architecture ensures separation of concerns, allowing for a clean division between frontend, backend, and database management.

Key features of the system include:

- User registration and secure login functionality through Clerk authentication
- AI-powered receipt scanning for automatic transaction entry using Gemini API
- Manual transaction creation, editing and categorization.
- Monthly budget setting and management with threshold alerts.
- Comprehensive financial reporting and insights
- Responsive design for consistent usability across all devices

The application leverages modern web development principles to provide a secure, efficient, and engaging user experience. The integration of AI for receipt scanning represents a practical application of emerging technologies to solve real-world problems in personal finance management.

1.3 Motivation

The Motivation behind developing this platform lies in the challenges individuals face when managing personal finance. Traditional methods of expense tracking often involve tedious manual data entry, making it difficult to maintain consistent financial records. Existing applications may lack intelligent features or offer poor user experiences, leading to abandoned financial management efforts

By undertaking this project, the aim is to design and implement a robust and intelligent solution that addresses these pain points:

- Tedious manual entry of transaction details.
- Difficulty in maintaining consistent financial records.
- Lack of timely budget alerts to prevent overspending.
- Limited reporting capabilities for financial insights.
- Poor user experiences leading to abandoned financial tracking.

Furthermore, this project provides an opportunity to apply theoretical knowledge acquired during the academic program to a real-world problem domain, while exploring cutting-edge technologies like AI-powered document scanning and modern web development frameworks.

Ultimately, the motivation is to deliver a solution that not only demonstrates technical proficiency but also provides genuine utility to users seeking better financial management tools.

1.4 Goal

The overarching goal of this project is to create a scalable, secure, and user-centric web application that empowers users to manage their finances efficiently. The project aims to demonstrate how AI can be integrated into practical applications to reduce manual effort while providing valuable functionality.

Specific goals include:

- Implementing AI-powered receipt scanning to automate transaction entry.
- Creating an intuitive interface for managing financial transactions and budgets.
- Developing a reliable budget alert system to help users manage spending.
- Building comprehensive financial reporting capabilities.
- Ensuring robust security for sensitive financial data.
- Delivering a responsive design that works seamlessly across all devices.

In the long term, this project aims to serve as a foundation for exploring additional AI-powered financial management tools and potentially expanding into more sophisticated features like spending pattern analysis and financial forecasting.

1.5 Organization of the Report

This report is organized into nine chapters, each focusing on different aspects of the project development process. The introduction sets the stage by outlining the project's objectives, motivation, and goals. Subsequent chapters dive into the detailed analysis, design, implementation, and testing of the AI Finance Platform.

The initial system study examines the limitations of existing financial management tools and defines the problem being addressed. The feasibility analysis evaluates the technical, economic, and operational viability of the proposed solution. System analysis and software requirements specifications outline the detailed functional and non-functional requirements.

The design chapter explains the architectural decisions, database structure, and module organization, while implementation details the actual development process and technologies used. Testing describes the verification procedures employed to ensure system quality, and the conclusion summarizes achievements and suggests future enhancements.

Throughout the report, diagrams, code snippets, and screenshots are included to provide visual context and clarity. This structured approach ensures comprehensive documentation of the entire project lifecycle from conception to completion.

2. Initial System Study

2.1 Chapter Introduction

This chapter establishes the foundation for developing the AI Finance Platform by examining the current landscape of financial management applications. It outlines the limitations of existing systems and highlights the need for an intelligent, automated solution. By analyzing user pain points such as manual data entry and inadequate budget tracking, this chapter justifies the development of a more efficient and user-friendly financial management tool.

The introduction sets the context for understanding how the AI Finance Platform addresses gaps in current offerings through features like AI-powered receipt scanning, intelligent transaction management, and budget alerts. It also provides an overview of the project's organizational background and the methodology employed in developing this solution.

2.2 Drawbacks of the Existing System

Existing financial management applications, while functional, often suffer from limitations that affect user adoption and long-term usage. The Primary challenges Include:

- **Manual Data Entry Burden**

Most applications require users to manually input transaction details, a time-consuming process that often leads to inconsistent record-keeping.

- **Limited Receipt Processing**

Few platforms offer receipt scanning, and those that do often have accuracy issues or limited extraction capabilities.

- **Ineffective Budget Management**

Many applications offer basic budget setting but lack proactive alerts to help users avoid overspending.

- **Complex User Interfaces**

Many financial applications have steep learning curves with cluttered interfaces that discourage regular use.

- **Limited Cross-Platform Functionality**

Many solutions are not fully responsive or offer inconsistent experiences across different devices.

These limitations highlight the need for an intelligent, user-friendly financial management platform that reduces manual effort while providing comprehensive functionality.

2.3 Problem Definition

The project aims to address the gap in personal financial management tools by developing a system that combines AI-powered automation with comprehensive transaction and budget management. The core problem being solved is the lack of an integrated platform that minimizes manual data entry while providing robust financial tracking, proactive budget alerts, and insightful reporting.

Key Issues Identified:

- Time-consuming manual entry of transaction details.
- Lack of intelligent receipt processing capabilities.
- Absence of proactive budget monitoring and alerts.
- Limited reporting for financial insights.
- Inconsistent user experiences across devices.
- Concerns about security of financial data.

The solution focuses on building a user-centric platform where AI reduces the friction of financial tracking while providing tools that help users make better financial decisions.

2.4 The Proposed System

The proposed AI Finance Platform addresses all identified issues by providing a comprehensive, intelligent financial management solution. Its key features include:

- **AI-Powered Receipt Scanning** : Integration with Google's Gemini API to automatically extract transaction details from receipt images, eliminating manual data entry.
- **Comprehensive Transaction management**: Tools to create, edit, categorize, and track transactions from multiple sources.
- **Intelligent Budget Management** : Features for setting monthly budgets with proactive alerts when spending reaches 80% of allocated limits.
- **Financial Reporting**: Detailed monthly reports providing insights into spending patterns and financial health.
- **Secure Authentication**: Implementation of Clerk for robust user authentication and account management.
- **Database Security**: Utilization of Supabase for secure data storage with additional protection through Arject.
- **Responsive Interface**: Modern UI built with Next.js, React, and Tailwind CSS with Shaden UI components, ensuring consistency across all devices.

2.5 Scope of the System

The AI Finance Platform is designed to serve individuals seeking comprehensive financial management tools. It supports the creation of user accounts, transaction management through both AI-powered receipt scanning and manual entry, budget setting and monitoring, and financial reporting. The system includes proactive notification features to alert users when spending approaches budget thresholds.

The system aims to provide a balance between automation and user control, allowing individuals to leverage AI for efficiency while maintaining oversight of their financial data. This approach ensures that the platform remains useful for users with varying levels of financial complexity and management preferences.

2.6 Scope of This Project

This project focuses on designing and implementing the essential components of the AI Finance Platform, serving as a robust foundation for future expansion. The primary scope includes the development of both front-end and back-end components, the integration of the Gemini API for receipt scanning, implementation of secure user authentication, and the creation of core financial management features.

While the current project establishes a comprehensive financial management platform, certain advanced features are reserved for future development. These include integration with banking APIs for automatic transaction import, advanced financial analytics, and mobile application development. This phased approach ensures the delivery of a high-quality core product while establishing a foundation for continued enhancement.

2.7 System Development Approach

The development of the AI Finance Platform follows an Agile methodology, emphasizing iterative progress, continuous feedback, and adaptive planning. This approach allows the project to evolve in response to emerging challenges and changing requirements, ensuring that the final product aligns with user needs and technical feasibility.

The development process is broken down into sprints, each focusing on specific features or components. This structure enables regular evaluation of progress and allows for adjustments to the implementation strategy as needed. The iterative nature of Agile development is particularly valuable for a project that integrates emerging technologies like AI-powered document scanning.

Technology Stack Used:

- Frontend: Next.js, React.js, Tailwind CSS, Shadcn UI
- Authentication: Clerk
- Database: Supabase
- Security Enhancement: Arject
- AI Integration: Google Gemini API
- Hosting: Vercel, Render
- Version Control: Git & GitHub

3. Feasibility Analysis

3.1 Technical Feasibility

The AI Finance Platform has been evaluated for technical viability across multiple dimensions, with favorable results indicating that the project is technically feasible.

Key Points:

- **Technology Stack Readiness:** The chosen technologies—Next.js, React, Tailwind CSS, Supabase, and Clerk—are mature, well-documented frameworks with strong community support. These technologies are proven in production environments and provide the necessary capabilities for building a secure, responsive financial management platform.
- **AI Integration Capabilities:** Google's Gemini API provides robust document analysis capabilities suitable for receipt scanning. Initial testing confirms that the API can successfully extract transaction details from receipt images with acceptable accuracy. The API's documentation and support resources ensure that integration challenges can be effectively addressed.
- **Development Skills:** The development team possesses the necessary expertise in JavaScript/TypeScript, React component architecture, and API integration. Any knowledge gaps can be bridged through available documentation and community resources.
- **Infrastructure Requirement:** The application will be deployed on Vercel, which provides reliable hosting for Next.js applications. Supabase offers a managed database service that eliminates the need for complex database administration.
- **Security Implementation:** The combination of Clerk for authentication, Supabase for data storage, and Arject for additional protection provides a comprehensive security approach suitable for handling sensitive financial information.

3.2 Economic Feasibility

The economic assessment of the AI Finance Platform indicates favorable cost-benefit outcomes, particularly in the context of an academic project with potential for future commercialization.

Development Cost:

- **Software Tools:** Most development tools and frameworks are open-source or offer free tiers (Next.js, React, Tailwind CSS).
- **API Usage:** Gemini API offers a free tier sufficient for development and initial usage.
- **Hosting Platforms:** Vercel and Supabase provide generous free tiers adequate for project development and demonstration.
- **Development Labor:** As an academic project, development time is considered an educational investment rather than a direct cost..

Potential Benefits:

- **Educational Value:** Hands-on experience with modern web technologies and AI integration.
- **Portfolio Enhancement:** A fully functional project demonstrating technical capabilities.
- **Future Commercialization:** Potential for evolving into a commercial product with subscription-based revenue.
- **User-Value:** Significant time savings through automation of financial tracking tasks.

3.3 Operational Feasibility

The operational feasibility assessment focuses on how effectively the AI Finance Platform can be used, maintained, and supported after deployment.

Key Aspects:

- **Intuitive Interface:** The use of Shadcn UI components and Tailwind CSS ensures a clean, modern interface that reduces the learning curve.
 - **Maintenance Requirements:** The modular architecture and clear code organization facilitate ongoing maintenance and updates.
 - **Support Needs:** The intuitive interface minimizes the need for extensive user support.
 - **Documentation:** Comprehensive user guides and technical documentation will be developed.
 - **Issue Resolution:** A structured approach for identifying and addressing technical issues will be established.
-

3.4 Legal Feasibility

The legal feasibility analysis examines regulatory compliance and legal considerations relevant to the AI Finance Platform.

Regulatory Considerations:

- **User Privacy:** The platform implements basic GDPR principles, allowing users to access, edit, and delete their personal data.
- **Secure Data Storage:** Supabase provides encryption and security features that help maintain data confidentiality.
- **Gemini API Terms:** The platform's use of the Gemini API will comply with Google's terms of service and usage policies.
- **Non-Banking Status:** The application will clearly communicate that it is not a banking or financial institution and does not provide financial advice.

3.5 Schedule Feasibility

This section assesses whether the project can be completed within the given timeframe and academic calendar.

Timeline:

- **Weeks 1–2:** Requirement analysis, technology selection, and project setup
- **Weeks 3–4:** Frontend design and backend setup
- **Weeks 5–6:** Authentication implementation and basic transaction management.
- **Week 7–8:** Gemini API integration for receipt scanning
- **Week 9–10:** Budget management and alert system development
- **Week 11–12:** Financial reporting and dashboard implementation
- **Week 13–14:** Testing, debugging, and optimization.
- **Week 15:** Documentation and final deployment.



3.6 Resource Feasibility

Resource feasibility evaluates whether the team has access to adequate personnel, hardware, and software resources.

Available Resources:

- Human Resources: 5 student developers with supervision from a faculty guide
- Software Resources: Open-source frameworks and development tools
- Hardware Resources: Personal computers with stable internet connectivity
- Conclusion: All required resources are available, and no additional investment is necessary.

4. System Analysis

4.1 Introduction

The AI Finance Platform is designed to provide users with an intelligent solution for managing personal finances. This chapter analyzes the system's data flow, entity relationships, and behavioral characteristics to provide a comprehensive understanding of how the application works.

The system facilitates several key workflows:

- User Registration and authentication.
- Receipt scanning and transaction extraction.
- Manual transaction management.
- Financial reporting and analysis.

4.2 Data Flow Diagram

The Data Flow Diagram (DFD) illustrates how data moves within the system between users, subsystems, and data storage.

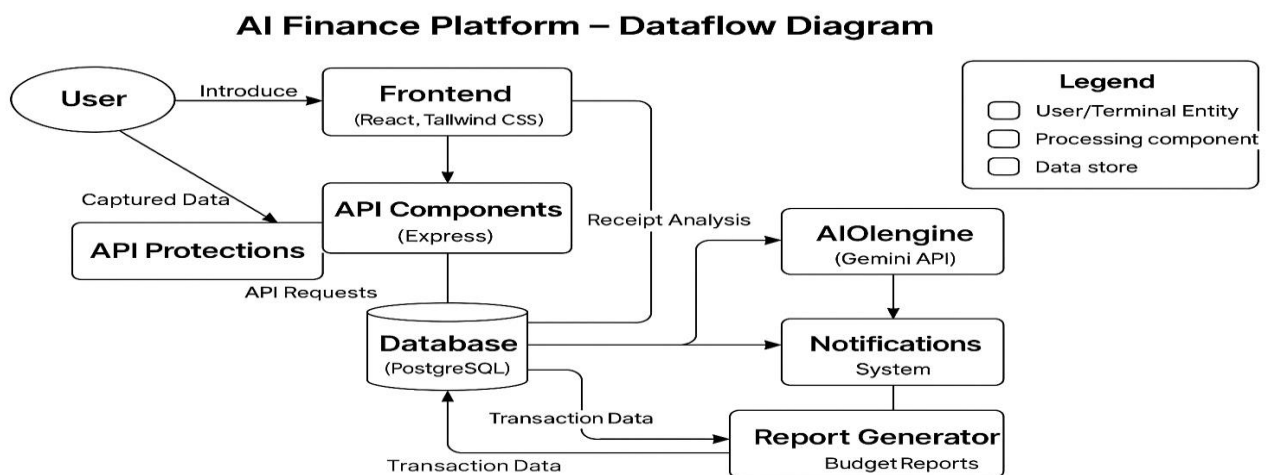


Figure 1: Dataflow Daigram

4.3 Entity Relationship Diagram

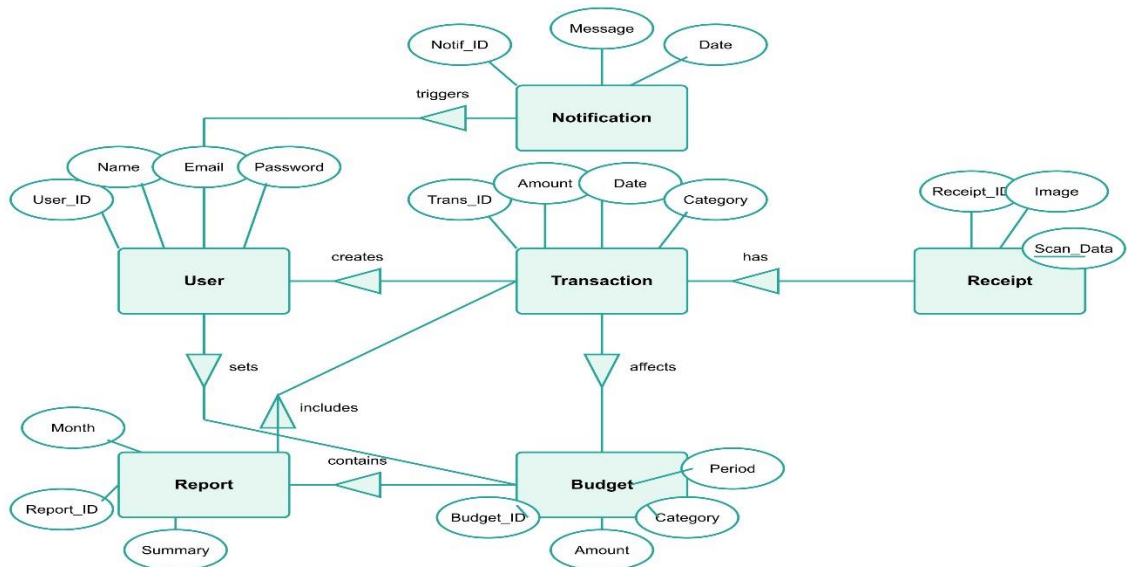
An Entity Relationship Diagram is used to define the relationships between data entities within the system.

Entities and Attributes:

- User: user_id, email, name, created_at
- Transaction: transaction_id, user_id, amount, date, description, category, payment_method, receipt_image_url
- Budget: budget_id, user_id, category, amount, period, start_date
- Alert: alert_id, user_id, budget_id, threshold_percentage, triggered_at, is_read
- Reports: report_id, user_id,

Relationships:

- One User can have many Transactions.
- One User can set One Budgets.
- One Budget can generate many Alerts.
- One User can have many Reports.



4.4 Physical and Behavioural Aspects of the System

Physical Aspects

The physical design of the AI Finance Platform encompasses its deployment architecture and technology stack:

- Frontend: Built with React and Tailwind CSS, hosted on Vercel.
- Backend: Next.js API routes for server-side processing , Integration with external services(Gemini API, Clerk, Supabase).
- Database: Supabase (PostgreSQL) for structured data storage, Secure access controls and data encryption.
- External Services: Clerk for authentication and user management or Arject for additional security measures.

Behavioural Aspects

The system behavior is defined by user interaction patterns and application responses:

- New users register through Clerk's secure authentication flow.
- Users upload receipt images through the application interface.
- The system processes images using Gemini API.
- Confirmed transactions are added to the database.
- Users can manually add, edit, or delete transactions.
- Transactions are categorized for budget tracking and reporting.
- Users set category-specific or overall budget limits.
- The system tracks spending against budget thresholds.
- The system generates periodic financial reports.
- Users can view spending patterns by category and time period.

5. Software Requirements Specifications

5.1 General Description

The AI Finance Platform is a comprehensive web-based application designed to help users manage their finances through intelligent transaction tracking, budget management, and financial reporting. The system leverages AI technology to reduce manual data entry and provides tools for monitoring spending and maintaining financial discipline.

5.1.1 Product Perspective

The product is developed as a full-stack web application using modern technologies and cloud services:

Component	Technology Used
Frontend	Next.js, React.js + Tailwind CSS, ShadcnUI
AI Intergration	Google Gemini API
Security	Arject
Database	Supabase(PostgreSQL)
Deployment	Vercel
Authentication	Cleark

The system operates independently and is designed to be easily scalable and maintainable.

5.1.2 Product Functions

Functionality	Description
User Registration/Login	Secure account creation and authentication via Clerk
Transaction management	Creation, editing, and categorization of financial transactions
Receipt Scanning	AI-powered extraction of transaction details from receipt images.



Functionality	Description
Budget Setting	Defining spending limits for various categories
Budget Alerts	Notifications when spending reaches 80% of budget limits
Financial Reporting	Monthly summaries and analysis of spending patterns
Data Export	Exporting transaction data for external use

5.1.3 User Characteristics

The platform is designed for individuals who want to manage their finances efficiently:

- General Users: Individuals seeking to track expenses and maintain budgets.
- Receipt Heavy User: People who make frequent purchases and need to track receipts.
- Budget-Conscious Users: Individuals trying to maintain financial discipline.
- Financial Planners: Users who want detailed insights into their spending patterns.

5.1.4 General Constraints

- The application must be responsive and work across various devices (desktop, tablet, mobile)
- Receipt scanning accuracy depends on image quality and the capabilities of the Gemini API
- Budget alerts are only generated when the application is accessed, not in real-time
- User data must be securely stored and processed in compliance with privacy standards.
- The system relies on cloud services that may have their own availability constraints.

5.1.5 Assumptions and Dependencies

- Users have internet access and modern web browsers
- The Gemini API remains available and maintains its current functionality.
- Clerk and Supabase services continue to operate reliably.
- Users grant necessary permissions for storing financial data.
- Receipt scanning works best with clear, well-lit images of standard format receipts.

5.1.6 Functional Requirements

Here are the main functional requirements:

- User Management
 - Users must be able to register and create accounts
 - Users must be able to securely log in and out.
 - User authentication must be maintained across sessions.
- Receipt Scanning
 - Users must be able to upload receipt images from their devices.
 - The system must process images using the Gemini API.
 - The system must extract transaction details (date, amount, vendor, items).
- Transaction Management
 - Users must be able to manually add transactions.
 - Users must be able to edit or delete existing transactions.
 - Users must be able to search and filter transaction history.
- Budget Management
 - Users must be able to set monthly budgets by category
 - The system must track spending against budget limits
 - The system must generate alerts when spending reaches 80% of budget
- Financial Reporting
 - The system must generate monthly financial summaries.
 - Users must be able to view spending by category.
 - Users must be able to compare spending across different periods.
 - Users must be able to export reports in common formats.

5.2 External Interface Requirements

5.2.1 User Interfaces

- Web-based responsive UI designed using modern frameworks like React and Tailwind CSS
- Dashboards for Users
- Receipt Scanner Camera/Upload interface for capturing receipt images.

5.2.2 Hardware Interfaces

- Client Devices: Compatible with desktop computers, laptops, tablets, and smartphones.
- Camera: Access to device camera for receipt scanning (optional, as users can upload images)
- Network: Internet connection for accessing cloud services and APIs.

5.2.3 Software Interfaces

- Frontend Framework: React.js with Tailwind CSS for modern, responsive user interfaces.
- Backend Technology: Next.js (for both Client Side and Server Side).
- Database: Supabase for data storage.
- APIs: Google Gemini API
- Authentication: Clerk for secure user authentication.

5.3 Performance Requirements

- Load Time: Pages should load within 2 seconds under normal conditions.
- Database Queries: All database queries should complete within 200 milliseconds.
- Scalability: The architecture should support horizontal scaling to accommodate growth in user base and transaction volume.

5.4 Design Constraints

- Mobile Responsiveness: The website will be usable on various screen sizes, including smartphones and tablets.
- Accessibility: Compliance with WCAG 2.1 for accessibility to users with disabilities.
- User Experience: Intuitive navigation and a minimalistic design to enhance user engagement.

5.4.1 Standard Compliance

- Web Standards: Adhere to HTML5 and CSS3 standards for web development.

5.4.2 Hardware Constratints

- Processor: Intel i5 or higher (or equivalent AMD)
- RAM: Minimum 8 GB (16 GB recommended)
- Storage: SSD with at least 50 GB free space
- Display: Minimum 13” screen with 1366x768 resolution
- Network: Stable internet connection for API testing and database access

For Hosting Server (Production Environment):

- Processor: 2-core CPU or higher
- RAM: Minimum 4 GB (8 GB recommended)
- Storage: At least 50 GB SSD
- Network: High-speed internet with minimum 100 Mbps bandwidth
- OS: Linux-based server (Ubuntu preferred)

For End Users:

- Device: Any smartphone, tablet, or PC
- Browser: Latest version of Chrome, Firefox, Safari, or Edge
- Internet: Minimum 1 Mbps connection for smooth page loading and image rendering

5.4.3 Other Requirements

- Security: Implementation of HTTPS, data encryption, and secure authentication via Clerk.
- Responsive Design: The application must be fully functional across various device sizes.
- Data Backup: Regular automated backups of the Supabase database.
- Error Logging: Comprehensive error logging and monitoring system.

5.4.4 Scope of this Project

- Core Features User authentication, transaction management, receipt scanning, budget management, alerts system, and monthly reports.
- Future Features: Advanced financial analytics, investment tracking, tax preparation assistance, and multi-currency support.
- Exclusions: Direct bank integration and cryptocurrency management are outside the current project scope.

6. System Design

6.1 Introduction

The AI Finance platform is designed to provide users with a comprehensive solution for managing personal finances, tracking expenses, and maintaining budgets. This chapter outlines the system architecture, component design, database structure, and communication patterns that form the foundation of the platform. The design leverages modern web technologies and artificial intelligence to create an intuitive, responsive, and secure financial management tool.

6.2 System Architecture

The application follows a client-server architecture with a modern front-end framework communicating with back-end service :

- Front-End: Built with Next.js and React, providing a component-based UI with server-side rendering capabilities. Styled using Tailwind CSS and shadcn/ui components for a consistent, responsive interface.
- Back-End: Next.js API routes handle server-side operations, data processing, and third-party service integration. Authentication is managed through Clerk, providing secure user identity management.
- Database: Supabase provides a PostgreSQL database with real-time capabilities, handling structured data for users, transactions, budgets, and financial reports.
- Integrations: Connects with the Gemini API for AI receipt scanning and processing, and implements services for email notifications and report generation.
- Security: Arject protection for enhanced security, along with Clerk authentication and data encryption.

6.3 Module Design

User Management Module:

- Handles user registration, authentication, and profile management through Clerk.
- Manages user preferences, notification settings, and account information.
- Implements role-based access control for multi-user scenarios.

Transaction Management Module:

- Provides CRUD operations for financial transactions.
- Includes manual transaction entry with categorization.
- Integrates with Gemini API for automated receipt scanning and data extraction.

- Supports transaction editing, categorization, and tagging.

Reporting Module:

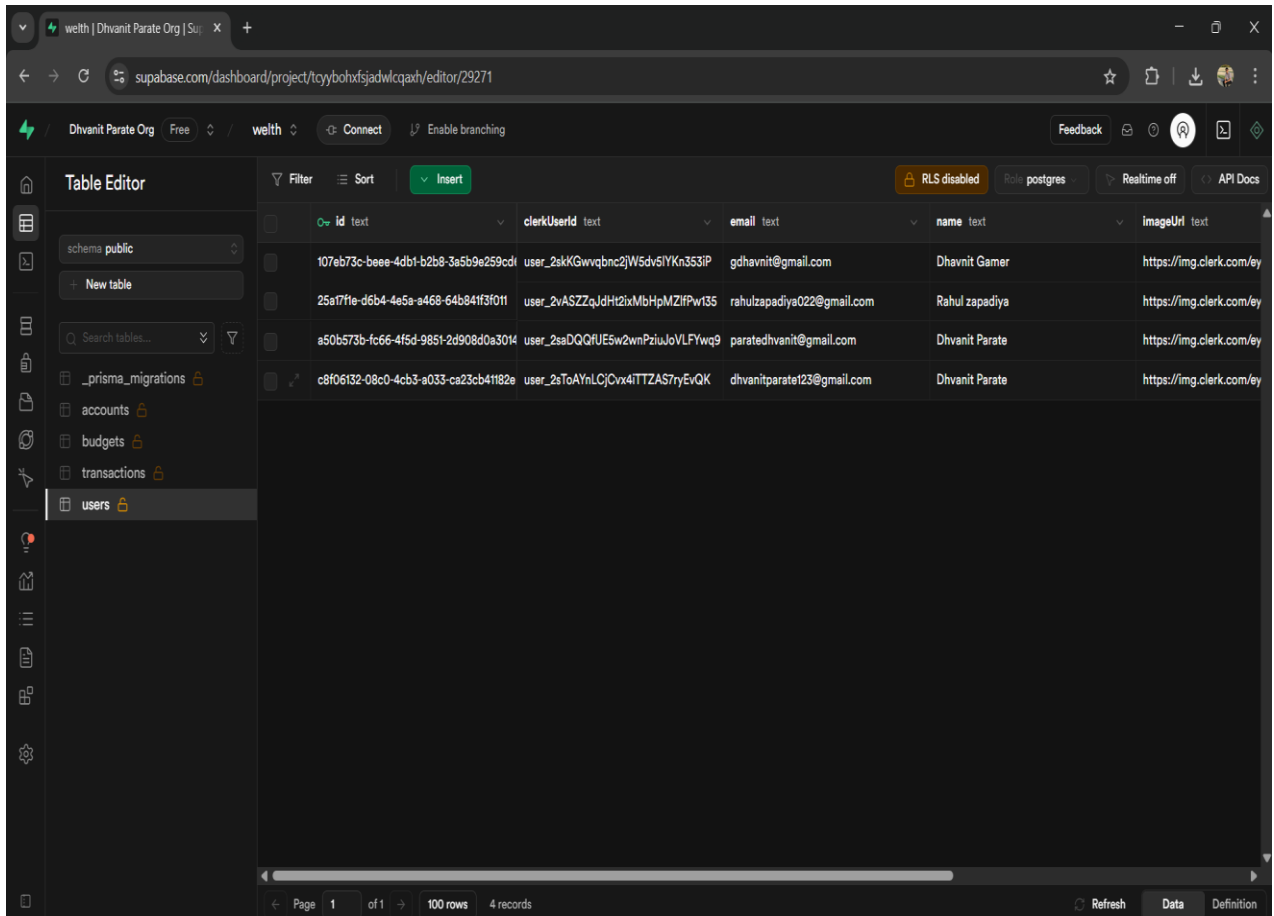
- Generates monthly financial summaries and reports.
- Produces visualizations of spending patterns and trends.
- Supports export functionality for reports in various formats.
- Schedules and delivers automated reports via email.

Notification System:

- Monitors budget thresholds and triggers alerts when spending reaches 80% of allocation.
- Sends email notifications for important financial events.
- Provides in-app notifications for budget alerts and report availability.

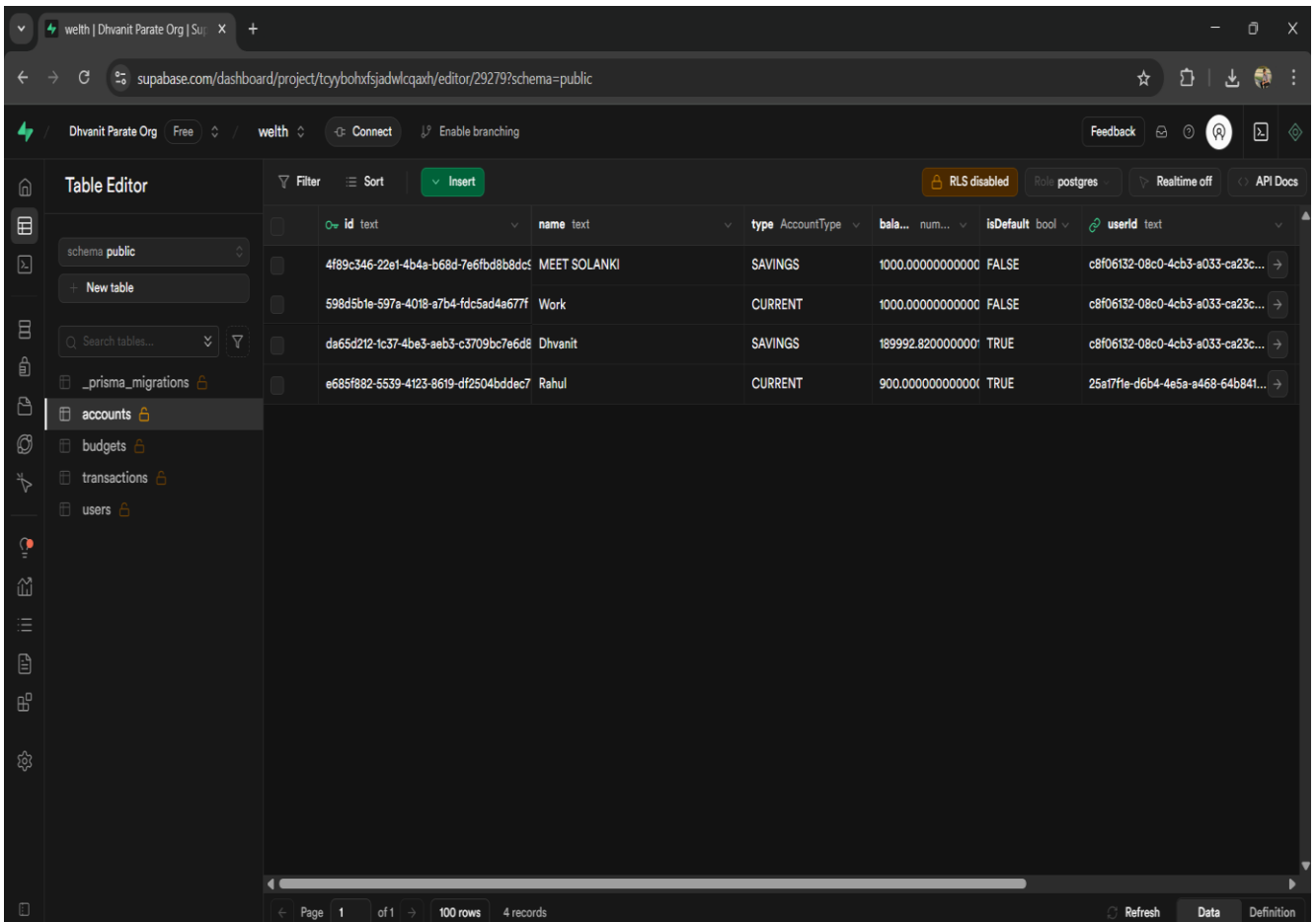
6.4 Database Design

6.4.1 User



id	clerkUserId	email	name	imageUrl
107eb73c-b0ee-4db1-b2b8-3a5b9e259cd1	user_2akKGwvqbnc2jW5dv5iYKn353iP	gdhavit@gmail.com	Dhavit Gamer	https://img.clerk.com/ey
25a17f1e-d6b4-4e5a-a468-64b84f13f011	user_2vASZZqJdHt2ixMbHpMZIfPwI35	rahulzapadiya022@gmail.com	Rahul zapadiya	https://img.clerk.com/ey
a50b573b-fc66-4f5d-9851-2d908d0a3014	user_2saDQqfUE5w2wnPziJoVLFYwq9	paratedhavit@gmail.com	Dhavit Parate	https://img.clerk.com/ey
c8f06132-08c0-4cb3-a033-ca23cb41182e	user_2sToAYnLCjCvx4ITZAS7ryEvQK	dhavitparate123@gmail.com	Dhavit Parate	https://img.clerk.com/ey

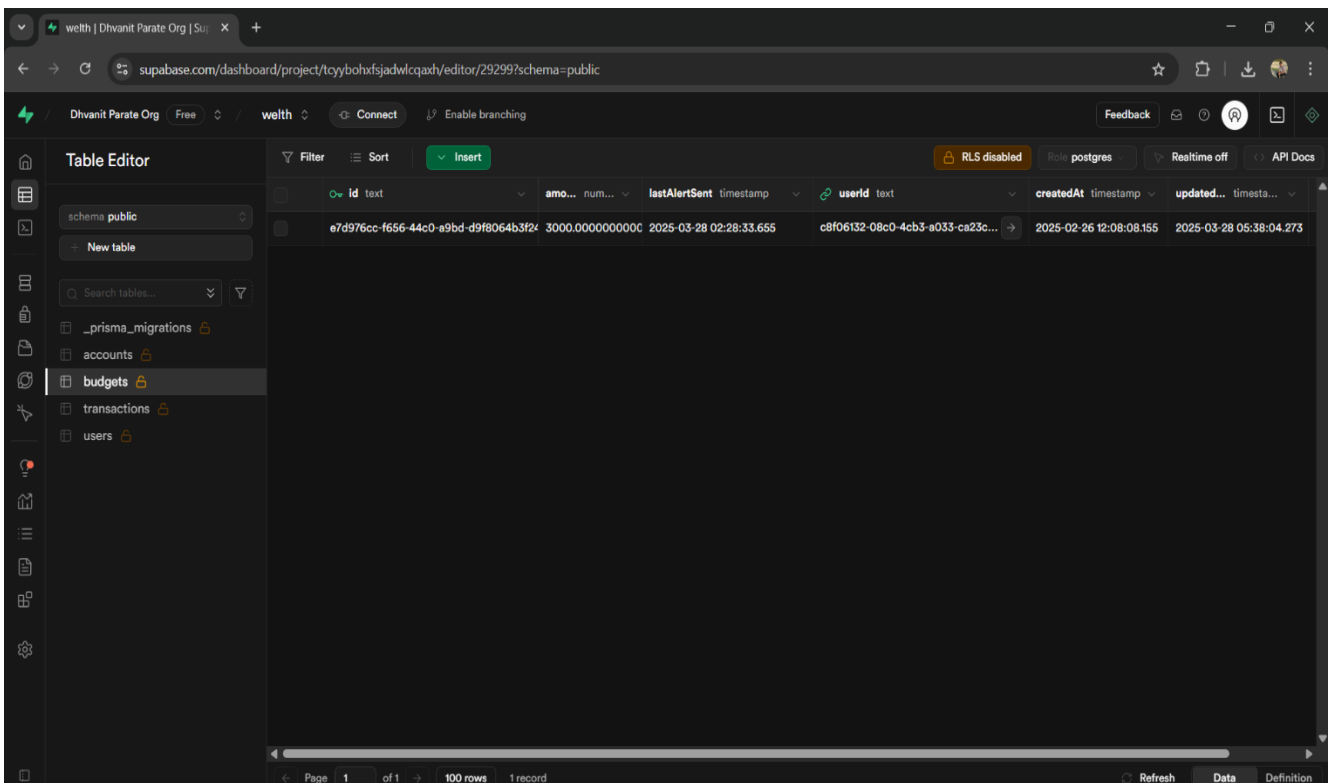
6.4.2 Accounts



The screenshot shows the Supabase Table Editor interface for the 'accounts' table in the 'public' schema. The table has the following columns: id (text), name (text), type (AccountType), balance (numeric), isDefault (boolean), and userId (text). There are 4 records in the table.

id	name	type	balance	isDefault	userId
4f89c346-22e1-4b4a-b68d-7e6fbd8b8dcf	MEET SOLANKI	SAVINGS	1000.000000000000	FALSE	c8f06132-08c0-4cb3-a033-ca23c...
598d5b1e-597a-4018-a7b4-fdc5ad4a677f	Work	CURRENT	1000.000000000000	FALSE	c8f06132-08c0-4cb3-a033-ca23c...
da65d212-1c37-4be3-aeb3-c3709bc7e6d8	Dhvanit	SAVINGS	189992.8200000000	TRUE	c8f06132-08c0-4cb3-a033-ca23c...
e685f882-5539-4123-8619-df2504bddec7	Rahul	CURRENT	900.000000000000	TRUE	25a77f1e-d6b4-4e5a-a468-64b841...

6.4.3 Budget



The screenshot shows the Supabase Table Editor interface for the 'budgets' table in the 'public' schema. The table has the following columns: id (text), amount (numeric), lastAlertSent (timestamp), userId (text), createdAt (timestamp), and updatedAt (timestamp). There is 1 record in the table.

id	amount	lastAlertSent	userId	createdAt	updatedAt
e7d976cc-f656-44c0-a9bd-d9f8064b3f2f	3000.000000000000	2025-03-28 02:28:33.655	c8f06132-08c0-4cb3-a033-ca23c...	2025-02-26 12:08:08.155	2025-03-28 05:38:04.273

6.4.4 Transaction

The screenshot shows the Supabase Table Editor interface for the 'transactions' table in the 'public' schema. The table contains 214 records. The columns are: id (text), type (TransactionType), amount (num...), description (text), date (timestamp), and category (text). The records are sorted by 1 rule. The table editor includes a sidebar with a list of tables: schema public, _prisma_migrations, accounts, budgets, transactions (selected), and users. The transactions table is currently selected, and its data is displayed in the main area. The data includes various transactions such as 'Paid for Gas Cylinder', 'Paid for books', 'Other income', 'Movies', 'Paid for Dinner', 'Paid for food', 'Paid for Petrol', '2.77L of Petrol', 'Paid for grocery', 'Paid for Dinner', 'Netflix (Recurring)', 'Received Salary (Recurring)', '2.77L of Petrol', 'Movies', 'Shopping for Home', and '2.77L of Petrol'. The table editor also includes a 'Filter' button, a 'Sorted by 1 rule' dropdown, and an 'Insert' button. The bottom of the interface shows a pagination bar with 'Page 1 of 3', '100 rows', and '214 records'. There are also buttons for 'Refresh', 'Data', and 'Definition'.

id	type	amount	description	date	category
937be5a9-130e-4c41-e531-d188c12cd23e	EXPENSE	880.000000000000	Paid for Gas Cylinder	2025-04-14 06:05:40.30	utilities
1308fb61-2481-484b-af20-5f301965d212	EXPENSE	330.000000000000	Paid for books	2025-04-06 13:32:29.97	education
d1bf233b-e858-41aa-9b5d-8d3f7b53d41e	INCOME	2399.990000000000	Other income	2025-04-04 13:11:19.21	investments
9c114e9-e673-4fac-ad4f-04b759ebe8c5	EXPENSE	270.000000000000	Movies	2025-04-03 03:37:04.21	entertainment
7953f50e-c580-473d-ac37-24996ef47ee5	EXPENSE	500.000000000000	Paid for Dinner	2025-04-02 10:06:47.58	groceries
8e6472c-fcf3-40ea-9ea2-40e7d7dff596	EXPENSE	100.000000000000	Paid for food	2025-04-02 09:38:28.97	food
4852906d-e317-4b79-9dd5-030366fbb1	EXPENSE	249.980000000000	Paid for Petrol	2025-03-28 05:38:19.58	transportation
8900971c-d918-45c3-af42-d650d0d204d	EXPENSE	254.970000000000	2.77L of Petrol	2025-03-21 18:30:00	transportation
29798b60-909b-463e-a932-53694f83fcb	EXPENSE	390.000000000000	Paid for grocery	2025-03-15 00:00:00	groceries
901359d4-6901-47c2-b35c-cf7f3afaf68	EXPENSE	994.000000000000	Paid for Dinner	2025-03-15 00:00:00	food
47c0d4c7-7207-4c4e-88b0-98d17f4d5552	EXPENSE	10.000000000000	Netflix (Recurring)	2025-03-12 13:48:30.305	entertainment
7e8528cc-858a-48fa-84cc-5b5c6363784	INCOME	6999.970000000000	Received Salary (Recurring)	2025-03-12 13:48:28.658	salary
217fdd6d-ed19-4eba-ac54-4fda252726cb	EXPENSE	269.990000000000	2.77L of Petrol	2025-03-08 00:00:00	travel
e2452b64-3462-4e8d-b661-624f9c5c5471	EXPENSE	10.000000000000	Movies	2025-03-07 18:30:00	entertainment
a0e7129c-bd54-4e03-9770-c9f8bb49b62	EXPENSE	107.600000000000	Shopping for Home	2025-03-03 18:30:00	groceries
a2760719-8db5-4797-8087-d4055cd57c61	EXPENSE	262.000000000000	2.77L of Petrol	2025-03-01 18:30:00	travel

6.5 Input Output Design

- Input Design:
- Transaction Form: Clean, intuitive form for manual transaction entry with fields for amount, description, category, date, and payment method.
- Receipt Upload: Simple drag-and-drop or file selection interface for uploading receipt images.
- Search Filters: Advanced search with multiple filters for transaction history.
- Output Design:
- Dashboard: Visual representation of financial status, recent transactions, and budget progress.
- Budget Progress: Visual indicators showing budget utilization with color-coded alerts.
- Alerts: Non-intrusive notifications for budget thresholds and important updates.

6.6 Algorithm Design

The system employs optimized algorithms to handle critical operations:

Receipt Scanning Algorithm:

- User Uploads receipt image
- Image is preprocessed(resizing, enhancement).
- Gemini API processes the image to extract text and structured data.
- Extracted data is validated and matched against predefined categories.
- Transaction is created with extracted information.
- User verifies and confirms or edits the details

Booking Algorithm:

- When a new transaction is added: a. Retrieve all budget categories affected by the transaction b. For each category, calculate current spending as percentage of budget c. If spending exceeds 80% threshold and no alert has been sent:
 - i. Create alert record
 - ii. Generate notification
 - iii. Send email alert if user has enabled email notifications.

6.7 Electronic Data Communication Design

The system employs RESTful APIs for efficient communication between the front-end and back-end. Key aspects include:

API Endpoints:

- /api/auth/: Authentication endpoints managed by Clerk.
- /api/transactions: CRUD operations for transactions.
- /api/budget: CRUD operations for budgets.
- /api/reports: Report generation and retrieval.
- /api/receipt: Receipt upload and processing with Gemini API.
- /api/alert: Alert creation and management.

Security:

- HTTPS for all communications.
- CORS policies to restrict unauthorized access.
- Rate limiting to prevent abuse.
- Encrypted data storage in Supabase.

6.8 System Maintenance

To ensure long-term reliability and performance, the system incorporates the following maintenance strategies:

- Regular Updates: Dependencies (e.g., React, Next.js, Supabase) are routinely updated to address security vulnerabilities and leverage performance improvements.
- Monitoring: Performance monitoring of API endpoints and database queries, OR Error logging and tracking.
- Backup and Recovery: Daily automated backups of the Supabase database, with recovery procedures in place for rapid restoration.

6.9 Other Alternatives Considered

During the design phase, alternative technologies were evaluated to ensure the optimal tech stack:

- Front-End Frameworks: Vue.js and Angular were evaluated, but Next.js with React was selected for its server-side rendering capabilities and robust ecosystem.
- Database Systems: MongoDB was considered but Supabase was chosen for its PostgreSQL foundation, real-time capabilities, and built-in authentication services.
- Authentication Services: Firebase Authentication and Auth0 were considered before selecting Clerk for its comprehensive features and seamless integration with Next.js.

7. System Implementation

7.1 Hardware Components

The system is designed to operate efficiently across a range of hardware configurations, ensuring accessibility for both server-side and client-side operations. The hardware requirements are as follows:

- **Server-Side Requirements:**
 - **Processor:** Minimum 2.8 GHz multi-core processor to handle concurrent API requests and database operations.
 - **RAM:** 8 GB or higher to support efficient processing of user interactions and data queries.
 - **Storage:** 500 GB SSD for storing the application, database, and media files, with provisions for scalability to accommodate growing data.
 - **Network:** High-speed internet connection with at least 100 Mbps bandwidth to ensure low-latency API responses.
- **Client-Side Requirements:**
 - **Processor:** Minimum 1.33 GHz processor, sufficient for rendering the web-based user interface.
 - **RAM:** 2 GB to support modern web browsers and client-side JavaScript execution.
 - **Storage:** 20 GB of free disk space for browser cache and temporary files.
 - **Devices:** Compatible with desktops, laptops, tablets, and smartphones with modern web browsers (e.g., Chrome, Firefox, Safari).

7.2 Software Environment

The software environment is carefully selected to ensure compatibility, performance, and ease of development. The following components form the core of the software stack:

- **Development Tool:**
 - Visual Studio Code with ESLint, Prettier, and TypeScript plugins.
 - Git for version control.
 - Vercel preview deployments for testing.

- **Web Server:**
 - Next.js with serves as the web server, handling HTTP requests and routing API calls.
 - Deployed on Vercel or Heroku for seamless scalability and continuous deployment.
- **Front-End Tools:**
 - React.js: Provides a dynamic, component-based front-end for rendering interactive user interfaces.
 - Tailwind CSS: Ensures responsive and visually consistent styling across devices.
 - JavaScript (ES6+): Powers client-side logic and asynchronous operations.
- **Back-End Tools:**
 - Supabase: A cloud-hosted PostgreSQL database for scalable data storage and management.
 - TypeScript: Enhances code maintainability and type safety for back-end development.
 - Next.js: Supports server-side rendering and static site generation for improved performance and SEO.

7.3 System Development Platform

The development platform is structured to facilitate rapid iteration, testing, and deployment, leveraging industry-standard frameworks and tools:

- **Front-End Development:**
 - React.js: Enables modular component development, reducing redundancy and improving reusability.
 - Tailwind CSS: Streamlines styling with utility-first classes, ensuring a responsive and modern design.
 - Vite: Used as the build tool for faster development and optimized production builds.
 - **Back-End Development:**
 - Next.js: Enhances back-end capabilities with server-side rendering and API routes.
 - TypeScript: Ensures robust code quality through static typing and early error detection.
 - Authentication: Clerk for authentication and user management.
 - API: Integration with Gemini API for AI-powered receipt processing.
 - **Database Management:**
 - Supabase PostgreSQL database with row-level security.
 - Database migrations for version control.
 - Foreign key constraints for data integrity.
-



- Version Control and Collaboration:
- Git and GitHub: Used for version control, enabling collaborative development and code reviews.
- Continuous Integration/Deployment (CI/CD): Integrated via Vercel or GitHub Actions to automate testing and deployment pipelines.

7.4 Project Accomplishment Status

The AI Finance platform has been successfully implemented with the following key components:

Completed Features:

- User authentication with Clerk
- Transaction management (create, read, update, delete)
- Receipt scanning with Gemini API
- Budget creation and tracking
- Monthly financial reports
- Responsive user interface
- Email notifications

7.5 Guidelines for Continuation

To ensure the continued success and evolution of the AI Finance platform, the following guidelines are recommended:

- **Code Management:**
 - Maintain Clean Code practices and documentation.
 - Follow the established branching strategy (feature branches, pull requests).
 - Keep dependencies updated regularly
- **Feature Roadmap:**
 - Prioritize features based on user feedback and business value.
 - Consider implementing advanced analytics in the next phase.
 - Explore integration with financial institutions for automatic transaction import
 - Develop mobile applications for iOS and Android.
- **Performance Optimization:**
 - Regularly audit and optimize database queries.
 - Implement caching where appropriate
 - Monitor API response times and optimize slow endpoints

- Consider implementing edge functions for global performance.
- **Security Maintenance:**
- Conduct regular security audits.
- Keep all dependencies updated to patch vulnerabilities.
- Implement additional layers of data encryption.
- Regular penetration testing.

7.6 Hardware Components

Server Side Requirement	
Processor	2.8 GHz
RAM	2 GB
Hard disk	4 GB(Free Space)

7.7 Software Environment

Server Side Requirement	
Operating System	Window Server 2003 or Any Compatible Server OS
Web Server	NextJs vercel
Front-End Tools	React , JavaScript, Tailwind CSS
Back-End Tool	Supabase, NextJs, typescript

8. System Testing

8.1 Test Plan

The testing strategy for the AI Finance platform encompasses multiple levels of testing to ensure functionality, performance, security, and usability meet the defined requirements:

- Unit Testing: Individual components and functions are tested in isolation.
- Integration Testing: Interactions between components are verified.
- Security Testing: End-to-end functionality is validated.
- Performance Testing: System behavior under various load conditions is assessed.
- Security Testing: Vulnerability assessment and penetration testing.
- Compatibility Testing: Ensures consistent functionality across supported browsers (e.g., Chrome, Firefox, Safari) and operating systems.

Testing Environment

- Hardware:
The server environment used for testing is equipped with a 2.8 GHz multi-core processor, 8 GB of RAM, and a 500 GB SSD. These specifications simulate a standard production environment and are sufficient to test the performance and scalability of the backend under typical usage conditions. On the client side, testing is conducted using multiple devices including desktops, laptops, and smartphones running modern browsers such as Google Chrome, Mozilla Firefox, Microsoft Edge, and Safari to validate cross-browser and cross-device compatibility.
- Software:
The platform is developed using a modern JavaScript stack. The backend is built with Next.js (for both Client and Server Side), while Supabase is used as the cloud-hosted PostgreSQL database. The frontend is created using React.js, with Tailwind CSS used for building responsive and visually appealing user interfaces. The application is deployed and tested on cloud hosting platforms like Vercel to ensure seamless performance during real-time access and to test deployment-related issues.

Testing Phases

The testing process is divided into multiple structured phases to ensure that the application meets both functional and non-functional requirements. Each phase focuses on a specific level of system validation and involves different types of test cases.

- Phase 1: Development Testing

Unit tests during feature development.
Component integration tests
Code quality checks via ESLint and TypeScript

- Phase 2 : Integration Testing

API endpoint validation.
Database operation verification.
Third-party service integration (Gemini API, email service)

- Phase 3 : System Testing

End-to-end user flows
Cross-browser compatibility
Responsive design validation

- Phase 4 : Non-Functional Testing

Performance under load.
Security vulnerability assessment
Accessibility compliance

- Phase 5 : User Acceptance Testing

Testing with actual users
Feedback collection and implementation
Final validation before deployment

8.2 Test cases

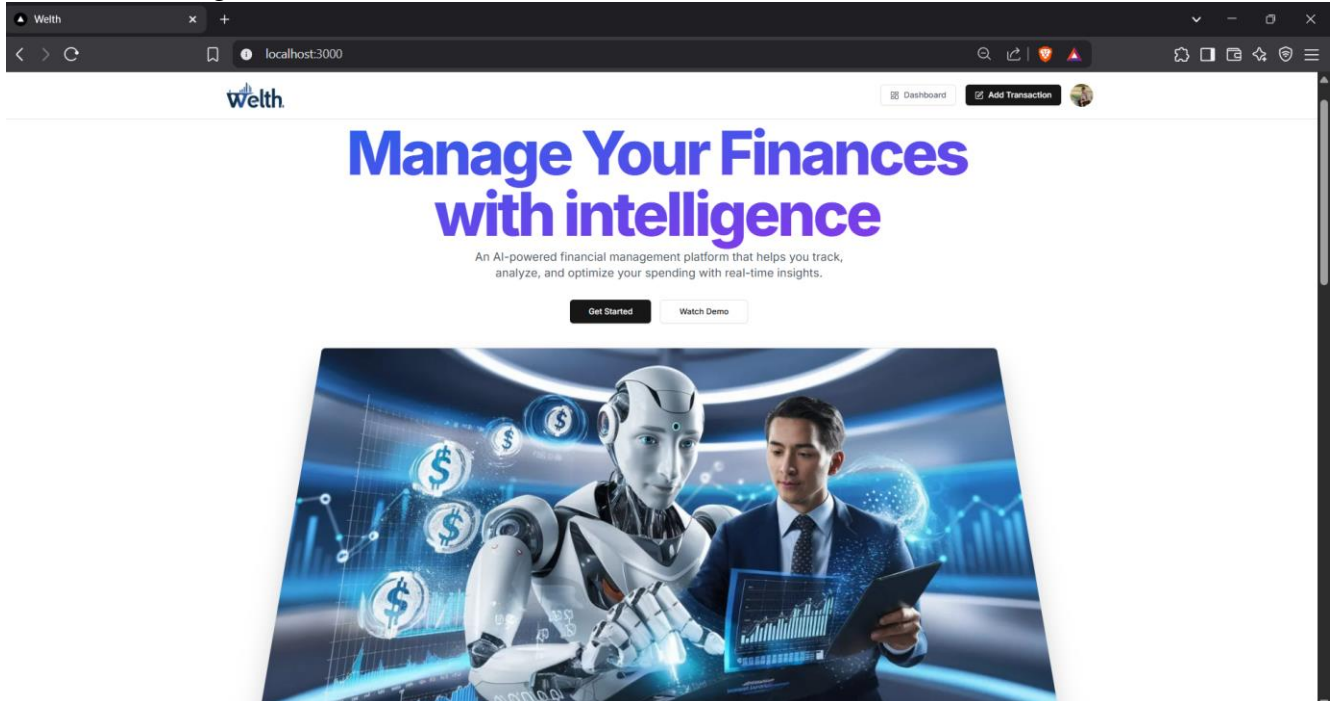
No.	Test Condition	Description	Expected Result	Actual Output	Status
1	User Registration	Register new user with valid credentials	User account created successfully	Registration successful	Pass
2	User Registration	Register with existing email	System shows validation error	"Email already exists"	Pass
3	User Login	Login with valid credentials	User authenticated and redirected to dashboard	Login successful	Pass
4	User Login	Login with invalid credentials	System shows error message	"Invalid email or password"	Pass
5	Transaction Creation	Add transaction with valid details	Transaction saved to database	Transaction created successfully	Pass

No .	Test Condition	Description	Expected Result	Actual Output	Status
6	Transaction Creation	Add transaction without amount	System shows validation error	"Amount is required"	Pass
7	Receipt Scanning	Upload valid receipt image	Receipt data extracted accurately	Receipt processed successfully	Pass
8	Receipt Scanning	Upload invalid file format	System shows error message	"Invalid file format"	Pass
9	Budget Creation	Create budget with valid details	Budget saved to database	Budget created successfully	Pass
10	Budget Alert	Add transaction that exceeds 80% of budget	Alert triggered and notification sent	Alert generated and email sent	Pass
11	Monthly Report	Generate monthly report	Report created with accurate data	Report generated successfully	Pass
12	Responsive Design	Access application on mobile device	UI adapts to screen size	Responsive layout rendered	Pass

No	Test Condition	Description	Expected Result	Actual Output	Status
13	Data Filtering	Filter transactions by date range	Only transactions within range displayed	Filtered results shown	Pass
14	Data Editing	Update existing transaction	Transaction updated in database	Transaction updated successfully	Pass
15	Performance	Load dashboard with 1000+ transactions	Page loads within 3 seconds	Page loaded in 2.5 seconds	Pass

8.3 Screenshots

1. Home Page



2. Authentication

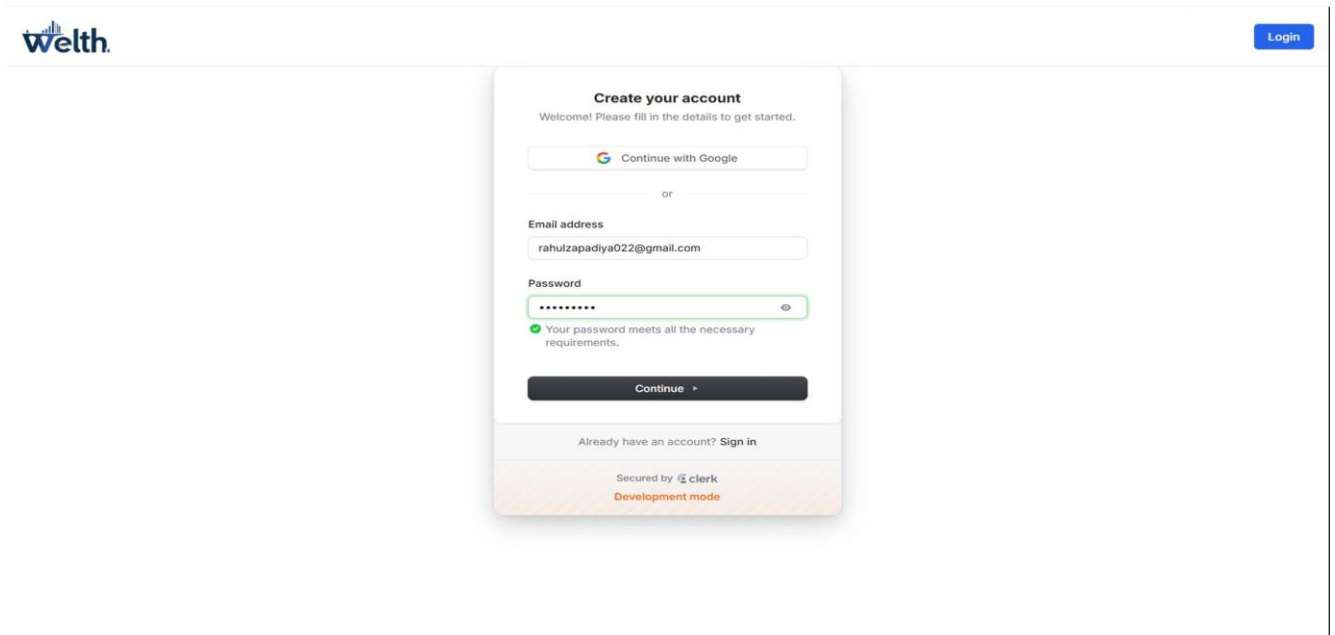
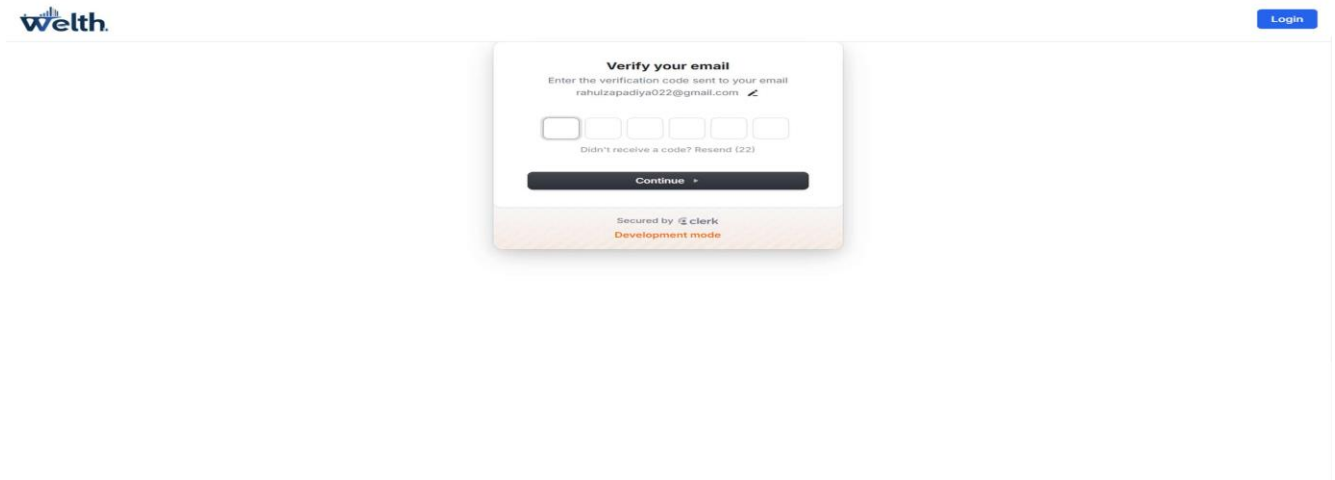
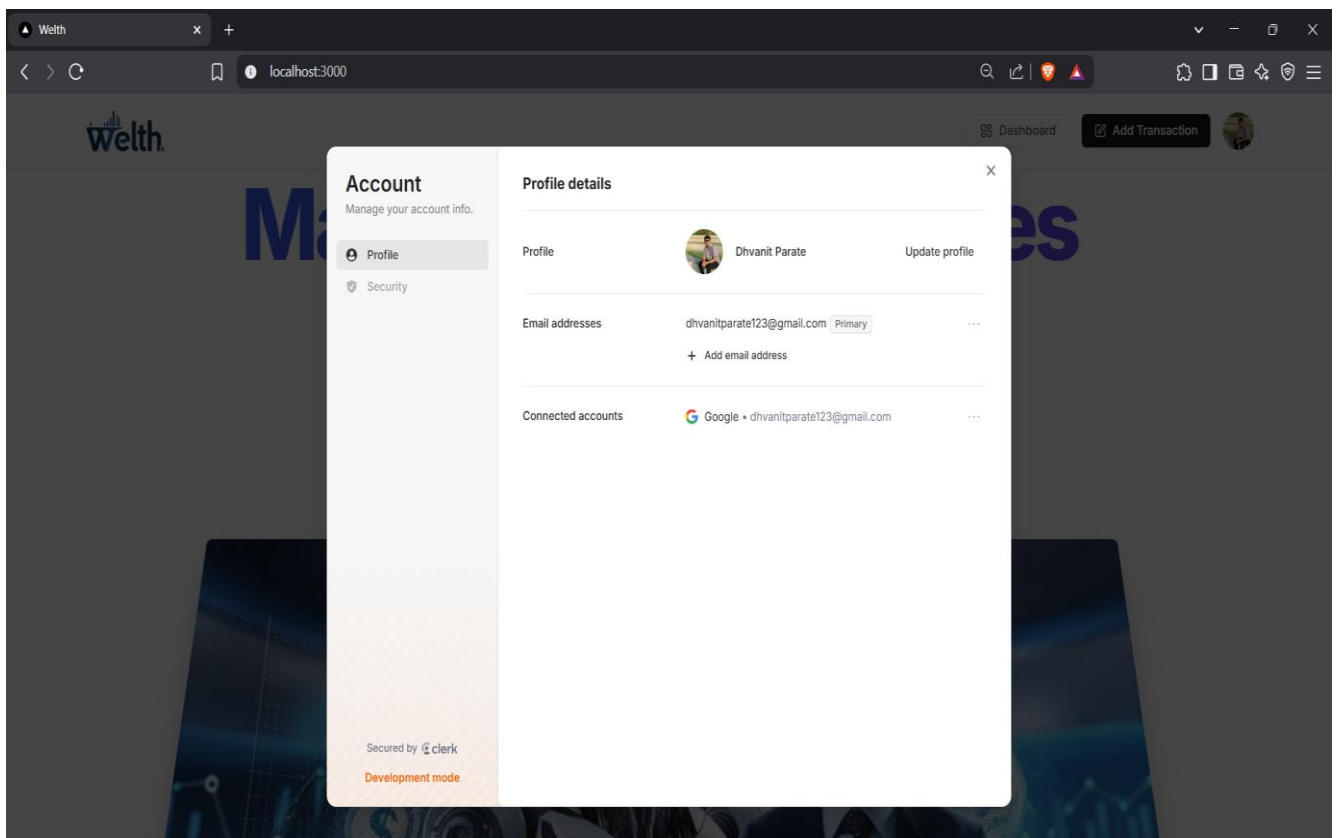


Figure 3: Screenshots

3. Login Verification through OTP via Email

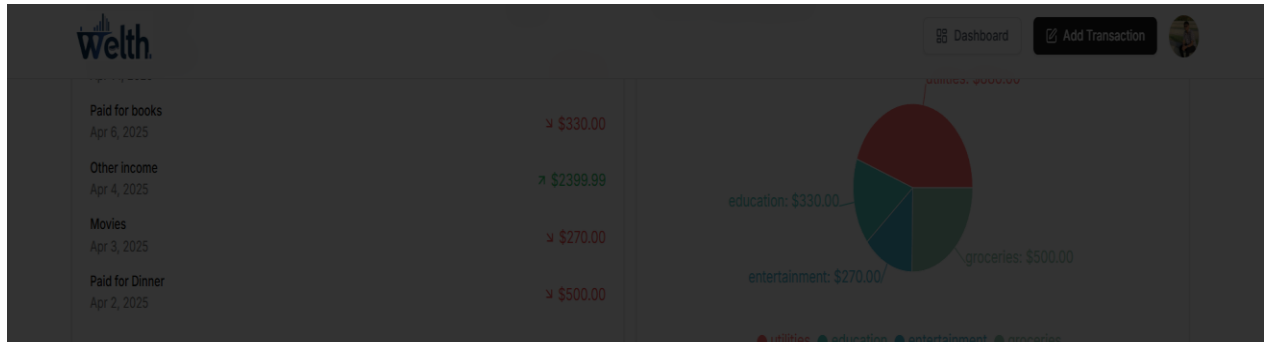


4. Manage Account Section





5. Create Account Page



Create New Account

Account Name

MEET SOLANKI

Account Type

Savings

Initial Balance

1000

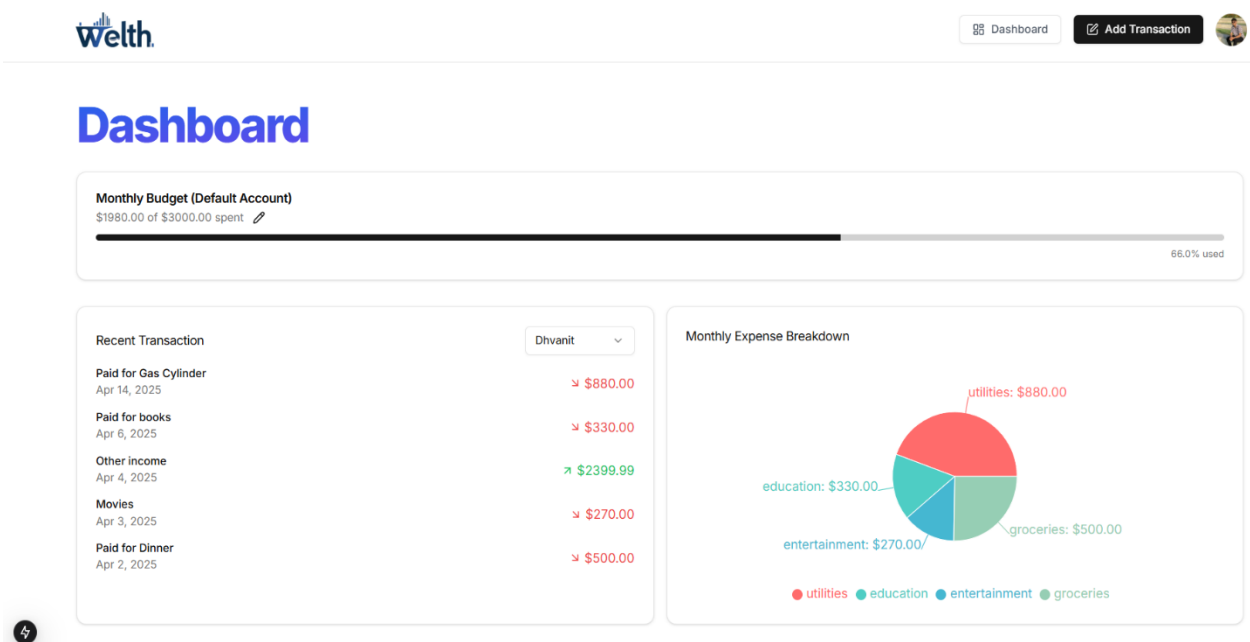
Set as Default

This account will be selected by default for transactions


Cancel

Create Account

6. Dashboard Page



7. Add Transaction Page



Dashboard
Add Transaction

Add Transaction

Scan Receipt with AI

Type

Expense

Amount

2000

Account

Dhvanit (\$190382.82)

Category

Groceries

Date

April 19th, 2025

Description

Paid for grocery

Recurring Transaction

Set up a recurring schedule for this transaction

☐

Cancel

Create Transaction

8. Transaction Statement Overview Page

Welth

Dashboard
Add Transaction


All Types
All Transaction


<input type="checkbox"/>	Date	Description	Category	Amount	Recurring	
<input type="checkbox"/>	Apr 14, 2025	Paid for Gas Cylinder	Utilities	-\$880.00	One-time	...
<input type="checkbox"/>	Apr 6, 2025	Paid for books	Education	-\$330.00	One-time	...
<input type="checkbox"/>	Apr 4, 2025	Other income	Investments	+\$2399.99	One-time	...
<input type="checkbox"/>	Apr 3, 2025	Movies	Entertainment	-\$270.00	One-time	...
<input type="checkbox"/>	Apr 2, 2025	Paid for Dinner	Groceries	-\$500.00	One-time	...
<input type="checkbox"/>	Mar 28, 2025	Paid for Petrol	Transportation	-\$249.98	One-time	...
<input type="checkbox"/>	Mar 22, 2025	2.77L of Petrol	Transportation	-\$254.97	One-time	...
<input type="checkbox"/>	Mar 15, 2025	Paid for Dinner	Food	-\$994.00	One-time	...
<input type="checkbox"/>	Mar 12, 2025	Netflix (Recurring)	Entertainment	-\$10.00	One-time	...
<input type="checkbox"/>	Mar 12, 2025	Received Salary (Recurring)	Salary	+\$6999.97	One-time	...

Page 1 of 22



9. Edit Transaction Page



DashboardAdd Transaction

Edit Transaction

TypeExpense

Amount1700

AccountDhvanit (\$190382.82)

CategoryGroceries

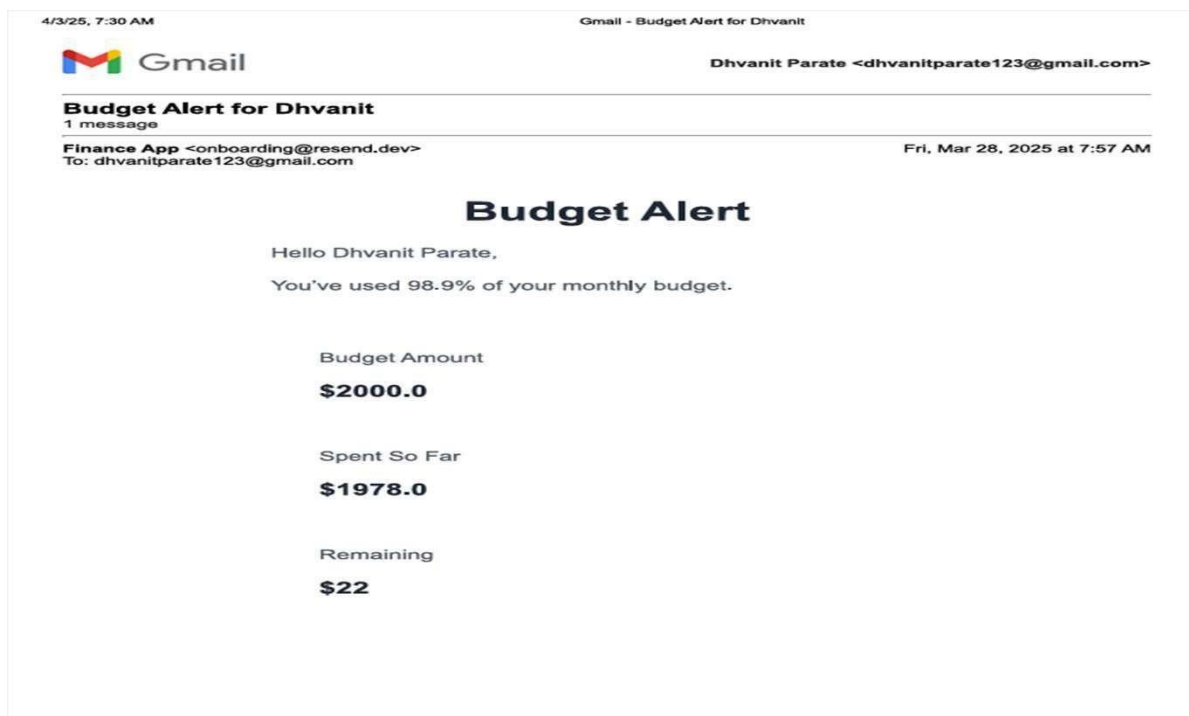
DateApril 18th, 2025

DescriptionPaid for Gas Cylinder

Recurring TransactionSet up a recurring schedule for this transaction

CancelUpdate Transaction

10. Budget Alert Email



11. Monthly Financial Report Email

4/19/25, 5:04 PM

Gmail - Your Monthly Financial Report - March

 **Gmail**

Dhvanit Parate <dhvanitparate123@gmail.com>

Your Monthly Financial Report - March
1 message

Finance App <onboarding@resend.dev>
To: dhvanitparate123@gmail.com

Sat, Apr 5, 2025 at 5:19 PM

Monthly Financial Report

Hello Dhvanit Parate,

Here's your financial summary for March:

Total Income

\$6999.97

Total Expenses

\$2178.54

Net

\$4821.43

Expenses by Category

groceries\$107.6

travel\$531.99

<https://mail.google.com/mail/u/0/?ik=f45cfb2ad7&view=pt&search=all&permthid=thread-f:1828563215943763185&simpl=msg-f:1828563215943763185> 1/2

4/19/25, 5:04 PM

Gmail - Your Monthly Financial Report - March

entertainment\$20

gifts\$20

food\$994

transportation\$504.95

Welth Insights

- Hey! You're saving a great chunk of your income, that's awesome. However, food and transportation seem a bit high. Consider meal prepping to reduce grocery and food costs.
- Your travel expense is significant. Could you explore cheaper travel options or cut back on trips for a while to boost savings even more?
- Entertainment and gifts are low, which is great for budgeting! Maybe allocate a bit more to entertainment each month for a better work-life balance, but keep it reasonable.

<https://mail.google.com/mail/u/0/?ik=f45cfb2ad7&view=pt&search=all&permthid=thread-f:1828563215943763185&simpl=msg-f:1828563215943763185> 2/2

9. Conclusion & Future Direction of Work

9.1 Conclusion

The AI Finance platform successfully delivers a comprehensive solution for personal financial management, leveraging modern web technologies and artificial intelligence. By integrating Next.js, React, Tailwind CSS, Clerk for authentication, and Supabase for database management, the platform provides a secure, responsive, and intuitive user experience.

The implementation of Gemini API for receipt scanning represents a significant advancement in automated transaction entry, reducing the manual effort required for expense tracking. The budget management system with automated alerts helps users maintain financial discipline, while the monthly reporting feature offers valuable insights into spending patterns and financial health.

The project demonstrates successful integration of multiple technologies to solve real-world problems in personal finance management, providing users with tools to make informed financial decisions and maintain better control over their finances.

9.2 Future Direction of Work

To enhance the platform further, the following features are suggested for future development:

- **Advanced Analytics:**
Implement machine learning algorithms for spending pattern analysis.
Develop predictive budgeting based on historical data.
Create personalized financial insights and recommendations.
- **Extended Integration:**
Direct bank connections for automatic transaction import.
Integration with payment platforms for real-time transaction tracking.
Tax preparation software integration for year-end financial planning.
- **Internationalization:**
Multi-currency support.
Region-specific financial rules and categories.
- **Multi-user Capabilities:**
Family accounts with customized access levels.
Business expense tracking with approval workflows.
Account sharing with financial advisors.



Bibliography/References

1. <https://nextjs.org/docs>
2. <https://react.dev/learn>
3. <https://supabase.com/docs>
4. <https://clerk.com/docs>
5. <https://tailwindcss.com/docs>
6. <https://ui.shadcn.com/docs>
7. <https://zod.dev/>
8. <https://react-hook-form.com/>
9. <https://www.typescriptlang.org/docs/>
10. <https://ai.google.dev/gemini-api>
11. <https://www.npmjs.com/package/arject>
12. <https://vercel.com/docs>
13. <https://www.postgresql.org/docs/>
14. <https://www.geeksforgeeks.org/nextjs-tutorial/>
15. https://www.tutorialspoint.com/tailwind_css/index.htm