A blue and black logo

Description automatically generated

Capstone Project Phase B

**Personal Budget Tracker**

25-1-D-26

Supervisor:

Dr. Zeev Frenkel

Noor Khateeb [Nour.El.Din.Khateb@e.braude.ac.il](mailto:Nour.El.Din.Khateb@e.braude.ac.il)

Shady Hijazy [Shady.hijazy@e.braude.ac.il](mailto:Shady.hijazy@e.braude.ac.il)

GitHub repository: <https://github.com/khatebnor29/Personal_Budget_Tracker>

Deployment Link : <https://pbt-project.vercel.app/>

[**1. Project Review 3**](#_sdkut3p7wjxh)

[1.1. Introduction 3](#_xm9lpx7dierr)

[1.2. Solution Description 3](#_xuc1ns7010zd)

[**1.2.1. Activity diagram 4**](#_ii8irhkcnsex)

[**1.2.2. System Architecture 4**](#_gx0up51xms9s)

[1.3. Research & Development Process 5](#_6harurh7k1g7)

[1.4. Tools & Client Interaction 5](#_jlzgqjm60znx)

[1.4.1. Tools 5](#_yx0z5lmyn7zz)

[1.4.2. Client/User Feedback 5](#_s29nr8ufjar3)

[1.5. Challenges & Solutions 5](#_gbjhml7fwbtd)

[1.6. Results & Conclusions 6](#_r4ggs85fpkl2)

[1.7. Lessons Learned 6](#_u86n6h5ym65x)

[1.8. Evaluation of Success Criteria 7](#_ftjpkfp3z9pd)

[**2. User Documentation 8**](#_5n134fm01nnq)

[2.1. User Guide 8](#_idn8tj87256v)

[2.2. Operating instruction 8](#_yg33unt8pc20)

[2.2.1. Homepage Overview 8](#_27339cyeb6r)

[2.2.2. Creating a New Account 9](#_2q800cdx4bm7)

[2.2.3. Logging In 10](#_aduk80b4bxuj)

[2.2.4. Dashboard Overview 10](#_oio5fxxv8h1n)

[2.2.5. Adding and Viewing Transactions 11](#_67yw28cc7f1t)

[2.2.6. Budget Management 11](#_ockp7gnddwmv)

[2.2.7. AI Financial Assistant 12](#_utphbvy579uk)

[2.2.8. Financial Insights Dashboard 13](#_z3gfmm6hghsf)

[2.2.9. Account Settings 13](#_3wuyke58npsg)

[**3. Maintenance Guide 14**](#_ke1dexoi7y52)

[3.1. System Overview 14](#_soa10xl7vue6)

[3.2 Directory Structure 14](#_be0hvgbd46r5)

[3.3. Required Tools and Environments 14](#_q5u63j39g9mx)

[Hardware Requirements: 14](#_iquwjr2hgqlj)

[3.4. Installation Instructions 15](#_frzpf6hpq5ga)

[3.5. Updating the System 15](#_9egcwj2e96ra)

[3.6. Security and Access 15](#_xcflq84qs6hy)

[3.7. Common Maintenance Tasks 15](#_2np02z3uvx9h)

[3.8. Future Enhancements 16](#_ow9zt0ze64l)

[**4. Key Application Usage & AI Behavior 16**](#_33lanwl72qi5)

[4.1. Database Changes When Adding a Transaction 16](#_28i9cqp272qi)

[4.2. Realtime Calculations and Budget Tracking 17](#_v5xvk0m3513v)

[4.3. AI Assistant Activation & Context 17](#_qyszdcdgssyn)

[4.4. Example Use Cases of AI Assistant 17](#_cumshrwaaanv)

# 1. Project Review

## 1.1. Introduction

The "Personal Budget Tracker" is a comprehensive web application built to assist users in managing their financial lives in a simple, intuitive, and effective way. The goal of the system is to help individuals track their income and expenses in real-time, analyze financial behavior, create custom budgets, and set and monitor savings goals.

The system is designed with a user-first mindset, targeting people with various levels of financial literacy:

1. Students who want to manage a tight budget
2. Adults managing household finances
3. Individuals interested in understanding and improving their spending behavior

The structure of the system includes a clean frontend interface, a secure and scalable backend, and integration with cloud databases and AI-based logic. The application supports user login and registration, personalized dashboards, interactive charts, and smart financial advice.

## 1.2. Solution Description

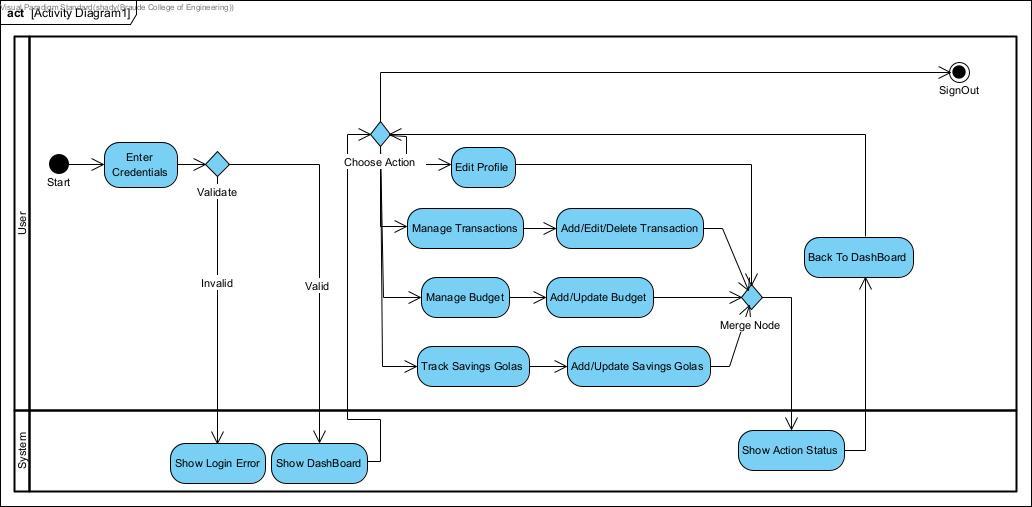
The application is built as a modern web platform using the following components:

* Frontend: React + Vite (with CSS)
* Backend: Node.js (server logic and API integration)
* Database: Firebase Firestore (cloud-hosted NoSQL DB)
* Authentication: Firebase Auth (user management)
* AI: OpenAI ChatGPT API (personalized financial advice)

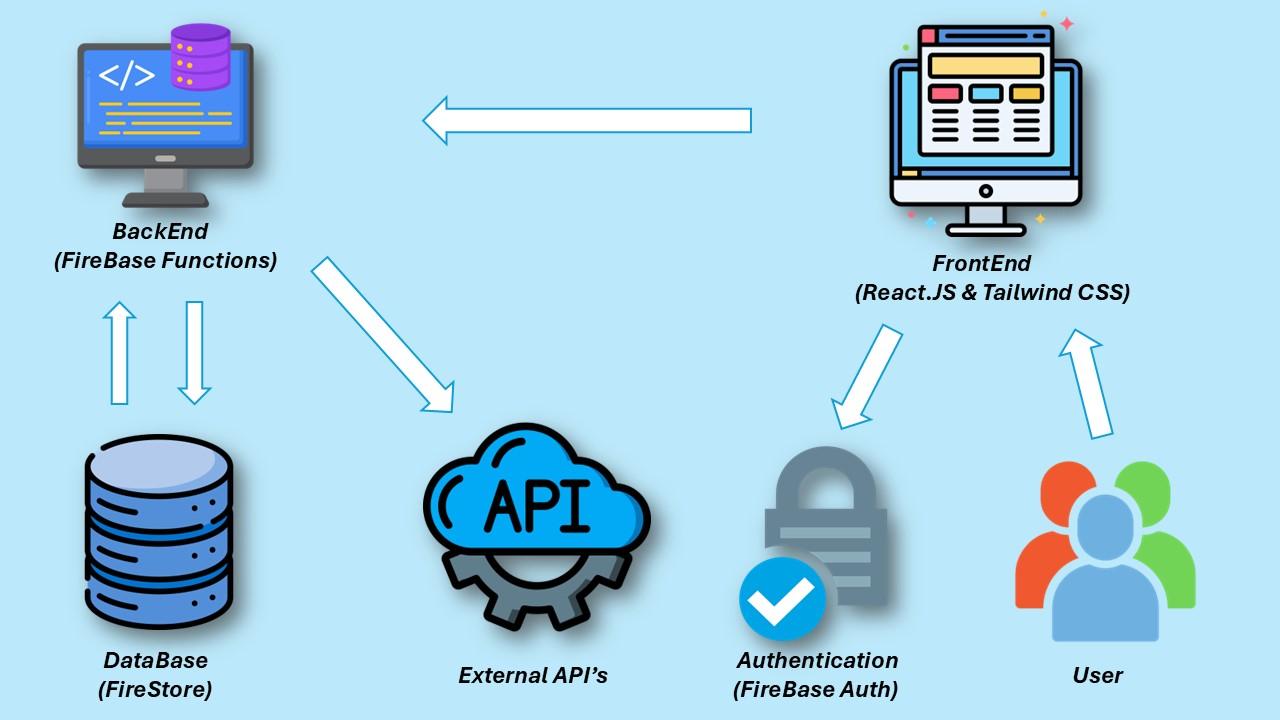
Component Overview:

* Login/Signup system
* Overview page describing the application's purpose
* Dashboard displaying income, expenses, and insights (in last days/months)
* Visual expense breakdown by categories (e.g., food, car, travel)
* Forms for users to input income and expenses, with live updates
* Data storage in Firebase for all transactions and budgets
* Real-time analytics and personalized algorithmic suggestions

### 1.2.1. Activity diagram



### 1.2.2. System Architecture



## 1.3. Research & Development Process

The project began with the requirements gathering and background research done in Phase A. We defined the market need, studied existing solutions, and proposed a feature-rich but simplified alternative. The development process followed these stages:

1. Setup and Planning: Defined UI/UX flow, database schema, and algorithm usage.
2. Frontend Development: Built with React and CSS; structured into reusable components.
3. Backend & Firebase Integration: Connected input forms and dashboard with Firestore.
4. AI Integration: Used ChatGPT API to generate savings tips and responses.

User testing occurred throughout the development using test accounts, and feedback was integrated into the design.

## 1.4. Tools & Client Interaction

### 1.4.1. Tools

* Development: React, Node.js, Vite, Firebase, Chart.js
* Database: Firebase Firestore
* Auth: Firebase Authentication
* Version Control: GitHub
* AI Services: OpenAI GPT API

### 1.4.2. Client/User Feedback

* Early feedback from peer testers helped refine the UI
* Login and dashboard flow was improved based on usability suggestions
* Real-time feedback for savings goals was enhanced after user input

## 1.5. Challenges & Solutions

During the development of the Personal Budget Tracker, the team encountered several technical and design-related challenges that required creative problem solving. One major challenge was ensuring real-time synchronization between the user interface and the Firebase backend. Initially, updates to data entries such as income or expenses did not reflect immediately on the dashboard. This was resolved by properly integrating Firebase listeners using real-time update mechanisms to ensure smooth data flow and consistency.

Another obstacle involved handling the user authentication process, particularly edge cases such as incorrect login attempts and password resets. Firebase Authentication simplified much of this, but custom feedback messages and additional form validation were added to improve usability.

On the front-end side, designing an intuitive and clean user interface using only raw CSS presented a challenge, especially without using UI libraries like Tailwind or Material UI. The team overcame this by creating modular CSS classes and maintaining a consistent visual structure across all pages.

Lastly, deployment to Firebase Hosting introduced configuration issues related to routing and static file handling. These were fixed by adjusting the Firebase hosting settings and adding rewrite rules to support the client-side routing used by React.

Each of these issues was addressed through iterative development, user testing, and collaboration, ensuring the application remained stable, responsive, and user-friendly.

## 1.6. Results & Conclusions

The application fully meets the goals defined in Phase A:

* All core features work smoothly
* Algorithms are integrated and output actionable insights
* Firebase ensures secure and real-time data storage
* User interface is intuitive and responsive
* Users can view visual summaries and export reports

Key Decisions:

* Use Firebase instead of full backend: saved time and added scalability
* Focus more on visualizations and actionable tips than advanced statistics
* Keep design simple and clean for ease of use

## 1.7. Lessons Learned

* Incremental development helped catch bugs early
* Firebase was a good choice for rapid MVP development
* Clear task division helped as a 2-person team
* Would add automated testing earlier if repeating
* Earlier UI prototyping would have saved rework

## 

## 1.8. Evaluation of Success Criteria

|  |  |  |
| --- | --- | --- |
| Metric | Met? | Explanation |
| Real-Time Expense Tracking | Yes | Firebase live updates, input works |
| Custom Budgets | Yes | Per-category limits and notifications |
| Algorithmic Insight | Yes | Visualized tips and suggestions based on data |
| User Experience | Yes | Simple, minimal, and tested UI |
| Secure Data Management | Yes | Firebase Auth + Firestore rules |
| Multi-platform Deployment | Yes | Web-based, works across browsers |

# 

# 2. User Documentation

## 2.1. User Guide

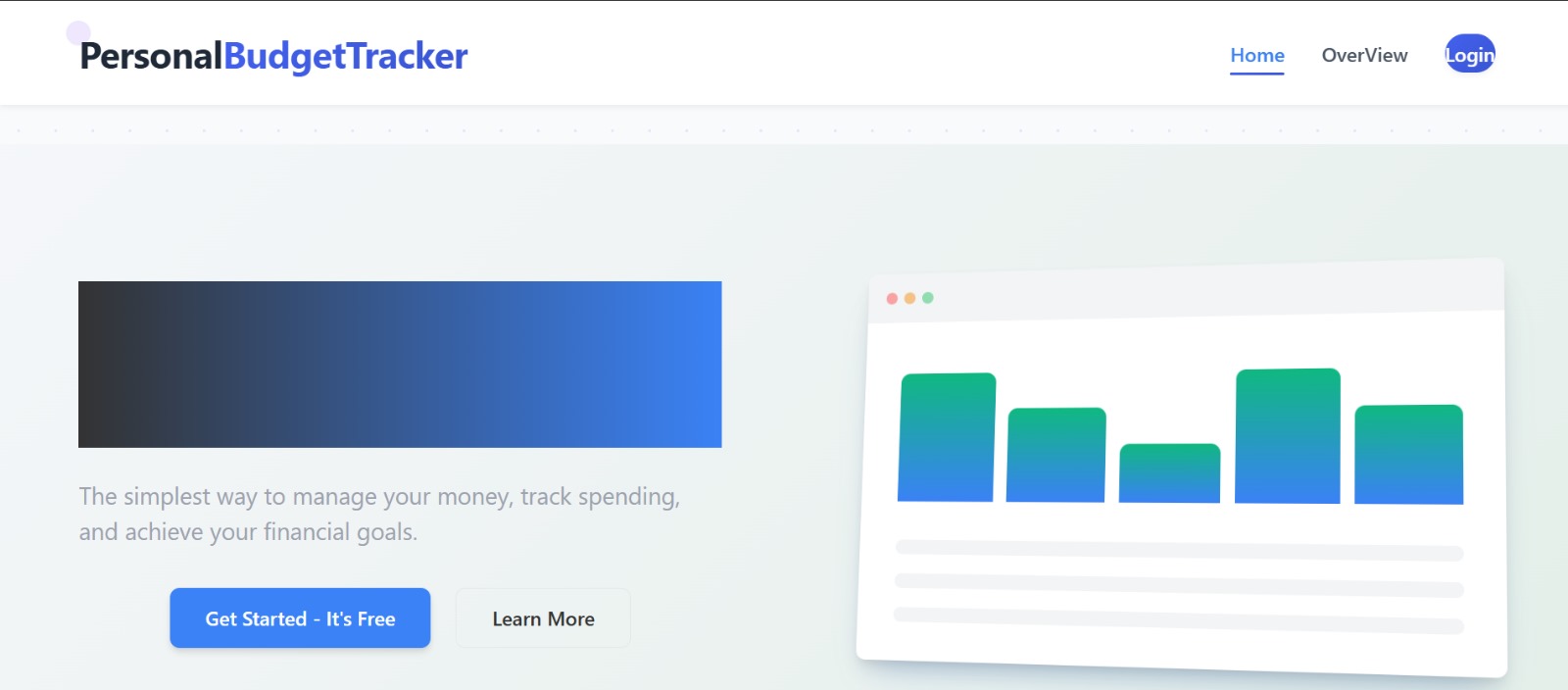
The Personal Budget Tracker is a user-friendly web application that helps individuals track their income, expenses, savings goals, and budget allocations. This guide provides clear instructions for navigating and using the application successfully.

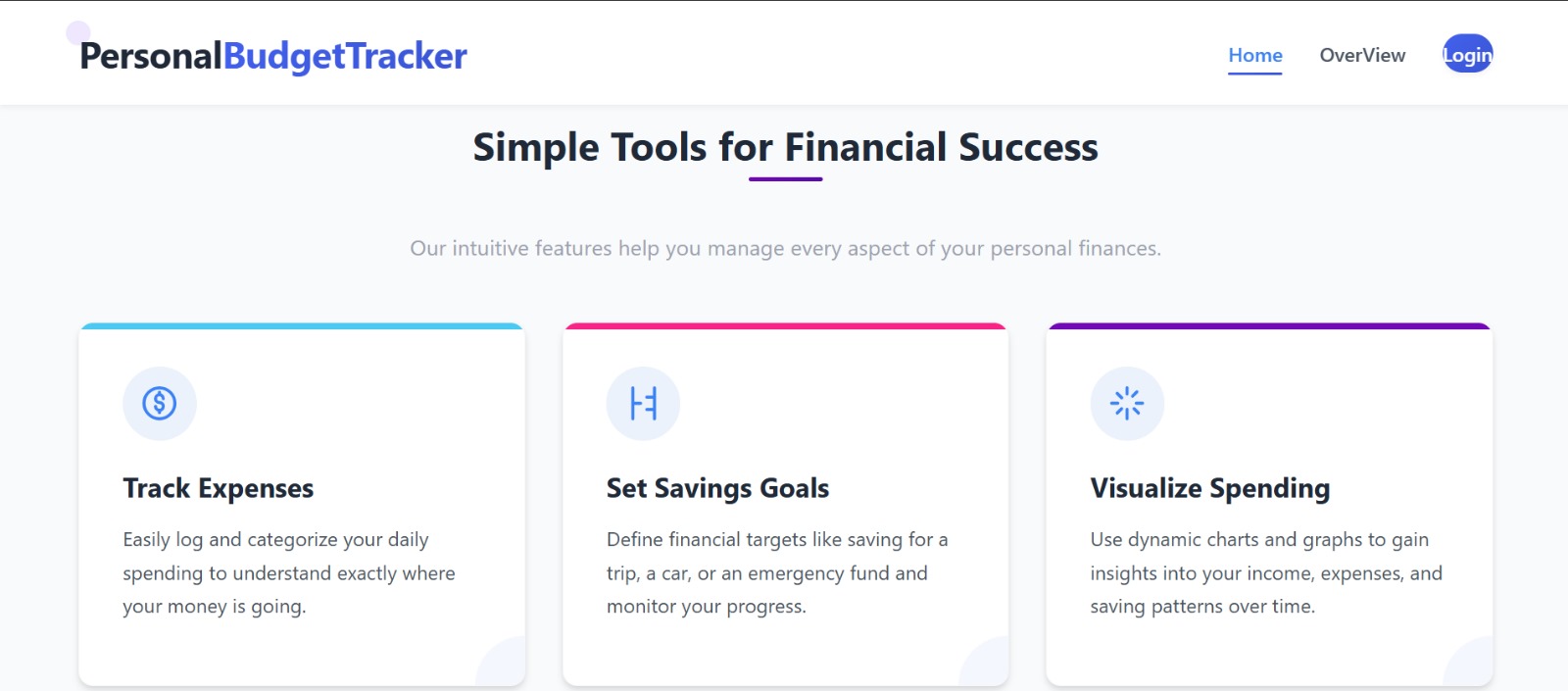
## 2.2. Operating instruction

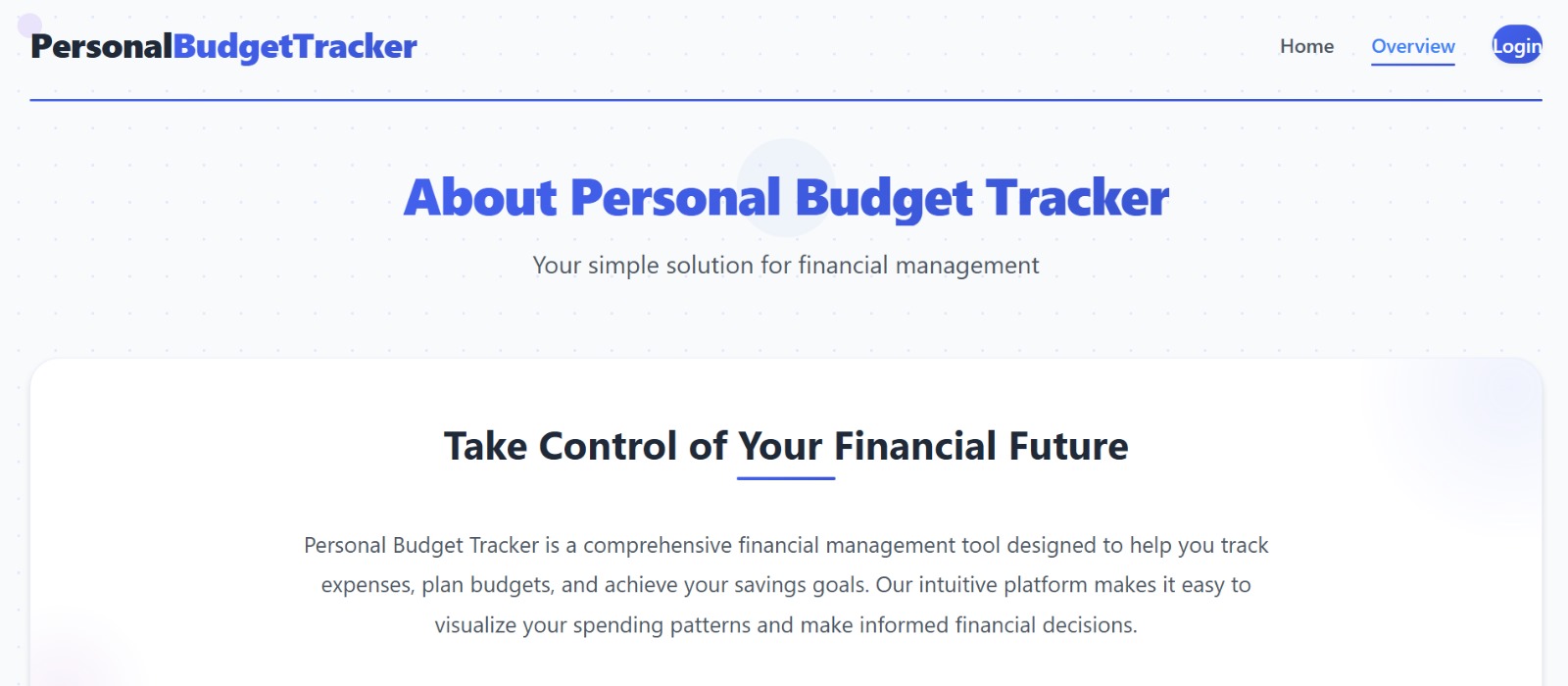
### 2.2.1. Homepage Overview

When visiting the website, users are welcomed with a clean homepage that presents the application's purpose and features. From here, users can:

* Click "Get Started - It's Free" to begin using the application.
* Navigate using the top menu bar to access the Overview or Login pages.



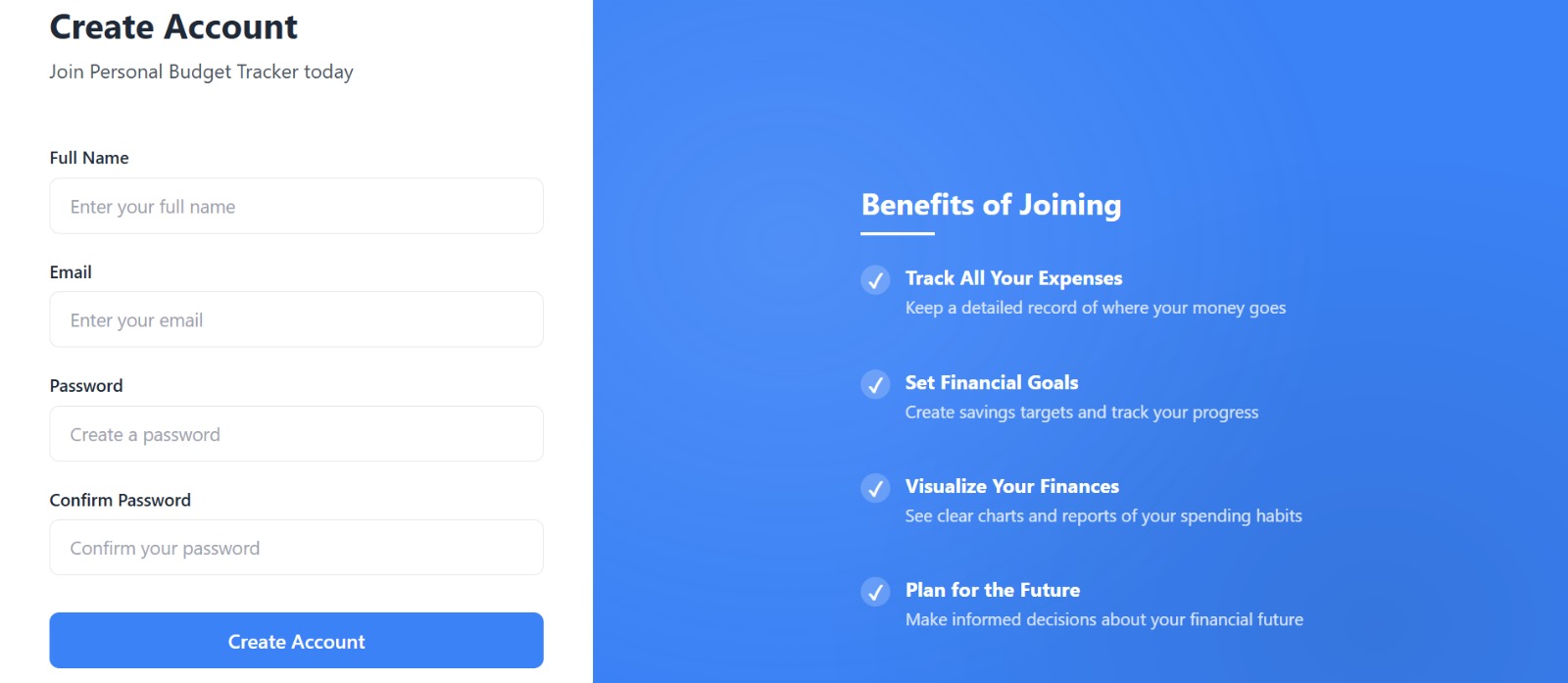




### 

### 2.2.2. Creating a New Account

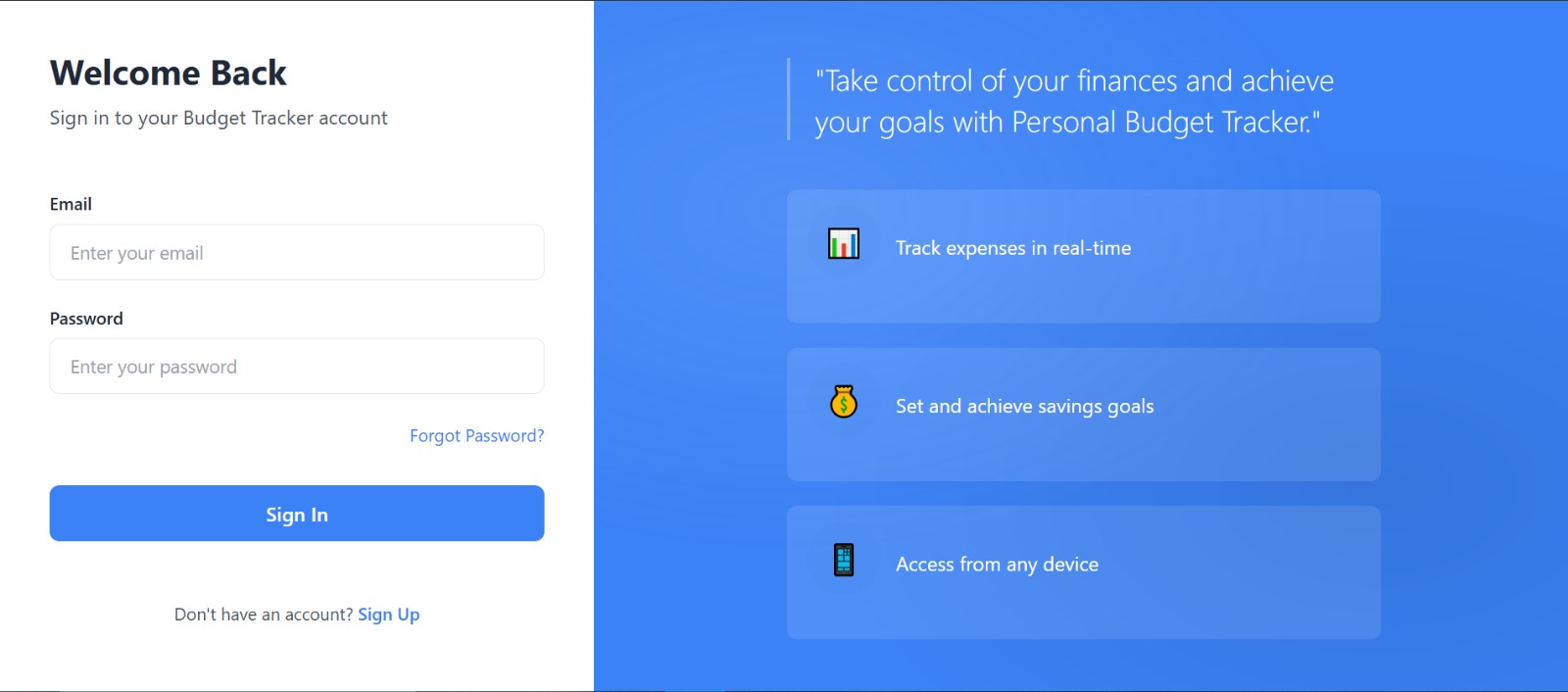
To start using the app, new users should click on the "Sign Up" link located on the Login page. They are then prompted to fill out a registration form including full name, email, password, and confirmation of the password. Once the form is complete, clicking the "Create Account" button finalizes the registration and redirects the user to their personal dashboard.



### 

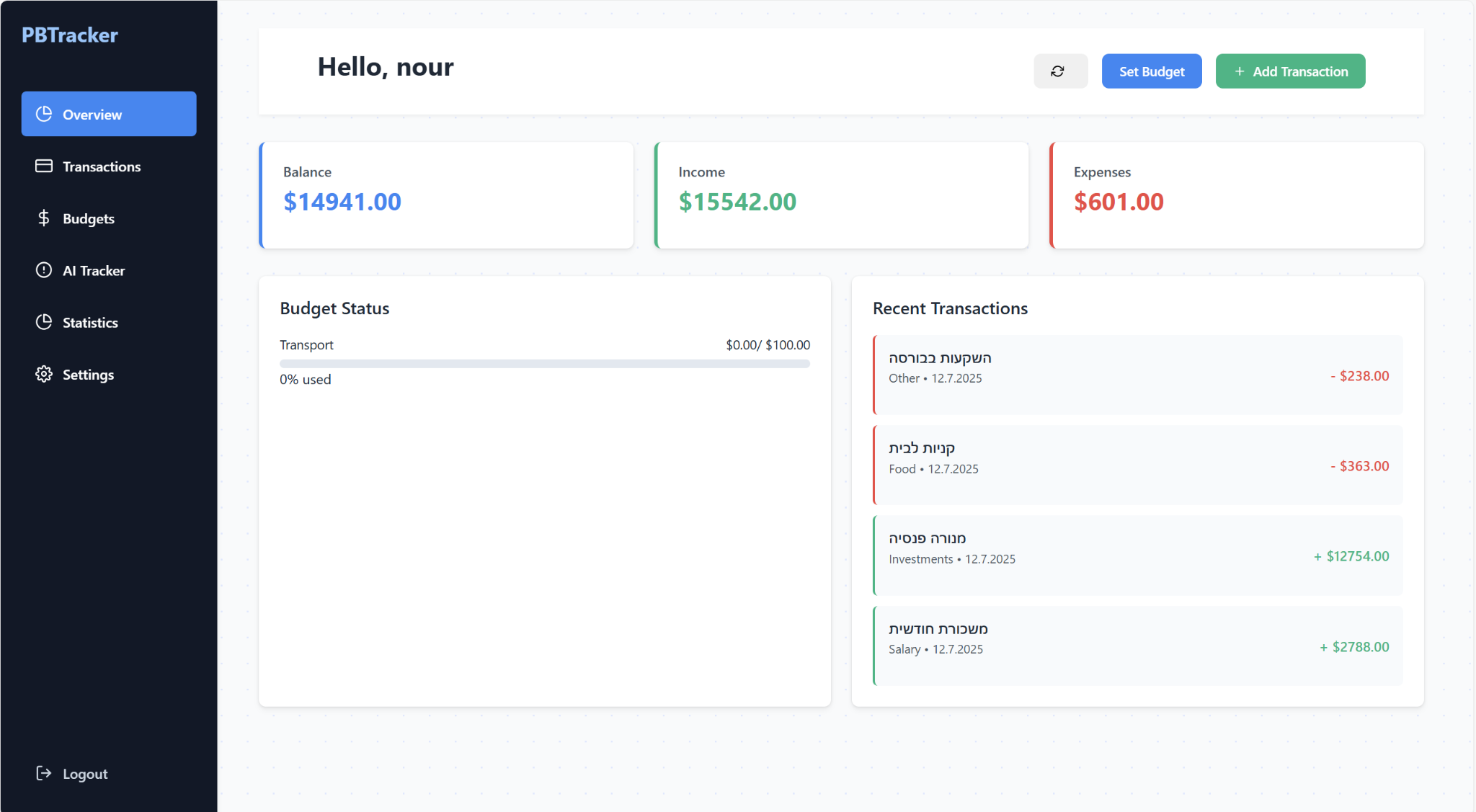
### 2.2.3. Logging In

Returning users can access their accounts by clicking the "Login" link from the homepage. After entering their registered email and password, clicking "Sign In" logs them into the application. Users who forget their password can easily reset it using the "Forgot Password?" link.



### 2.2.4. Dashboard Overview

Once logged in, users are greeted by name on the main dashboard. This dashboard provides a summary of their financial status, including current balance, total income, and total expenses. It also features a Budget Status section that visually indicates how much of each budget category has been used. Users can click the "Set Budget" button to define monthly limits for different spending categories, and the "Add Transaction" button to manually record income or expenses. The navigation menu on the left allows users to easily move between key sections of the application, including Overview, Transactions, Budgets, AI Tracker, Settings, and Logout.



### 2.2.5. Adding and Viewing Transactions

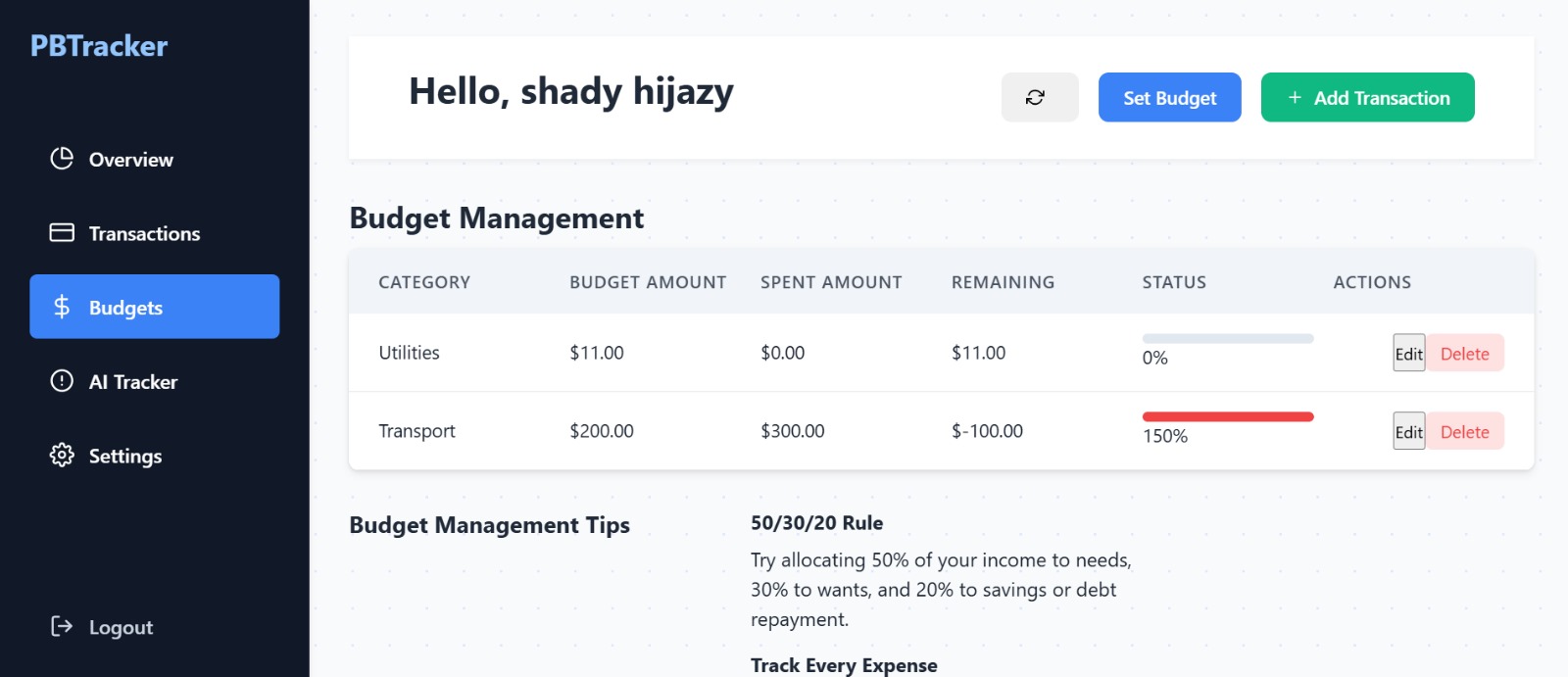
To track their financial activity, users should navigate to the Transactions section. By clicking the "Add Transaction" button, they can enter a new record with a description (e.g., "groceries"), category (e.g., Food, Transport), date, and amount. Income is recorded as a positive value and expenses as a negative one. All transactions are listed in a table with the option to delete any record as needed.



### 

### 2.2.6. Budget Management

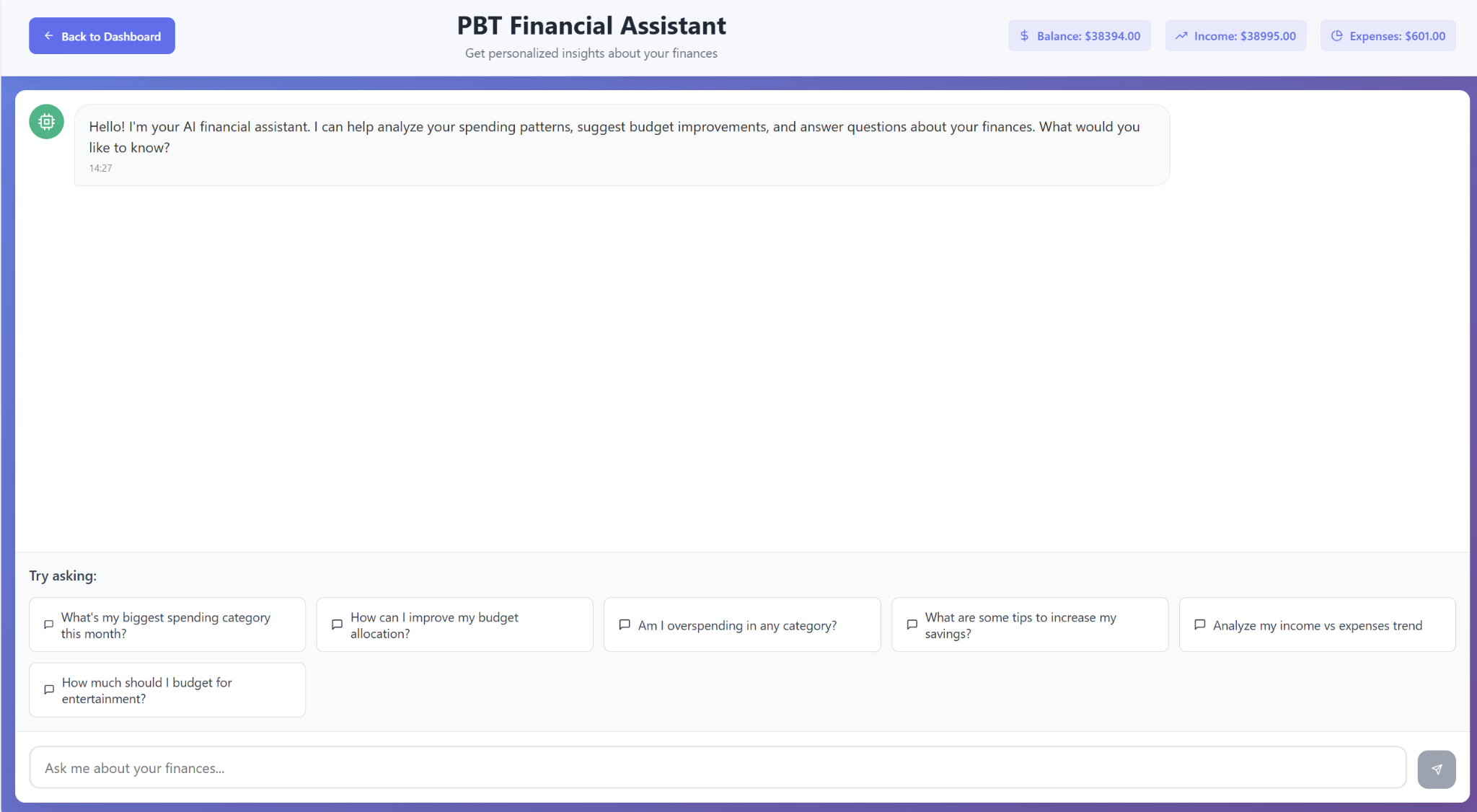
Users can manage their spending categories under the Budgets section. This view presents a table displaying each budget category alongside its allocated amount, current spending, remaining balance, and usage status. A visual bar indicates whether the budget is on track or overused, with excessive use highlighted in red. Users can edit or delete individual budgets and review helpful budgeting tips such as the 50/30/20 rule displayed below the table.

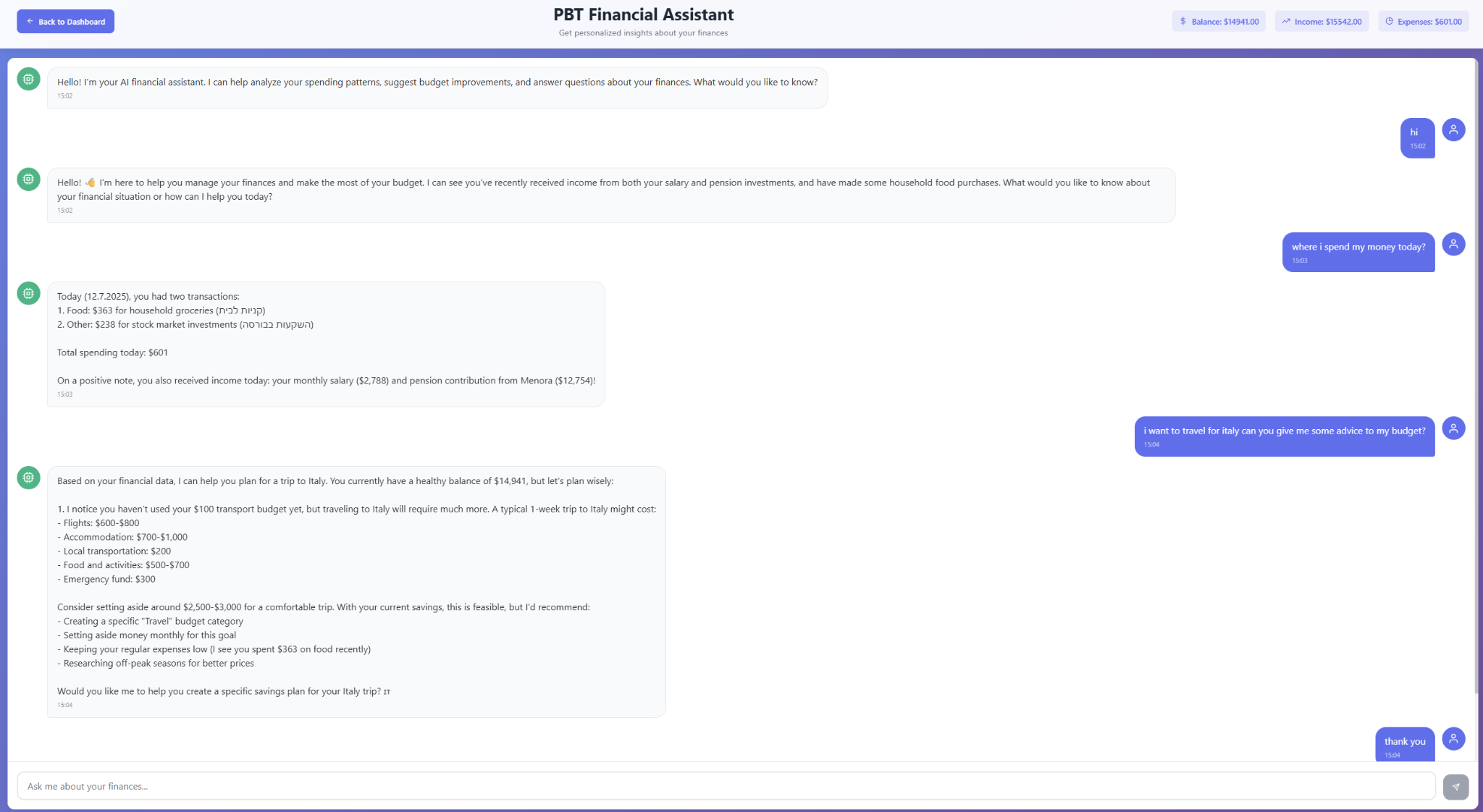


### 

### 2.2.7. AI Financial Assistant

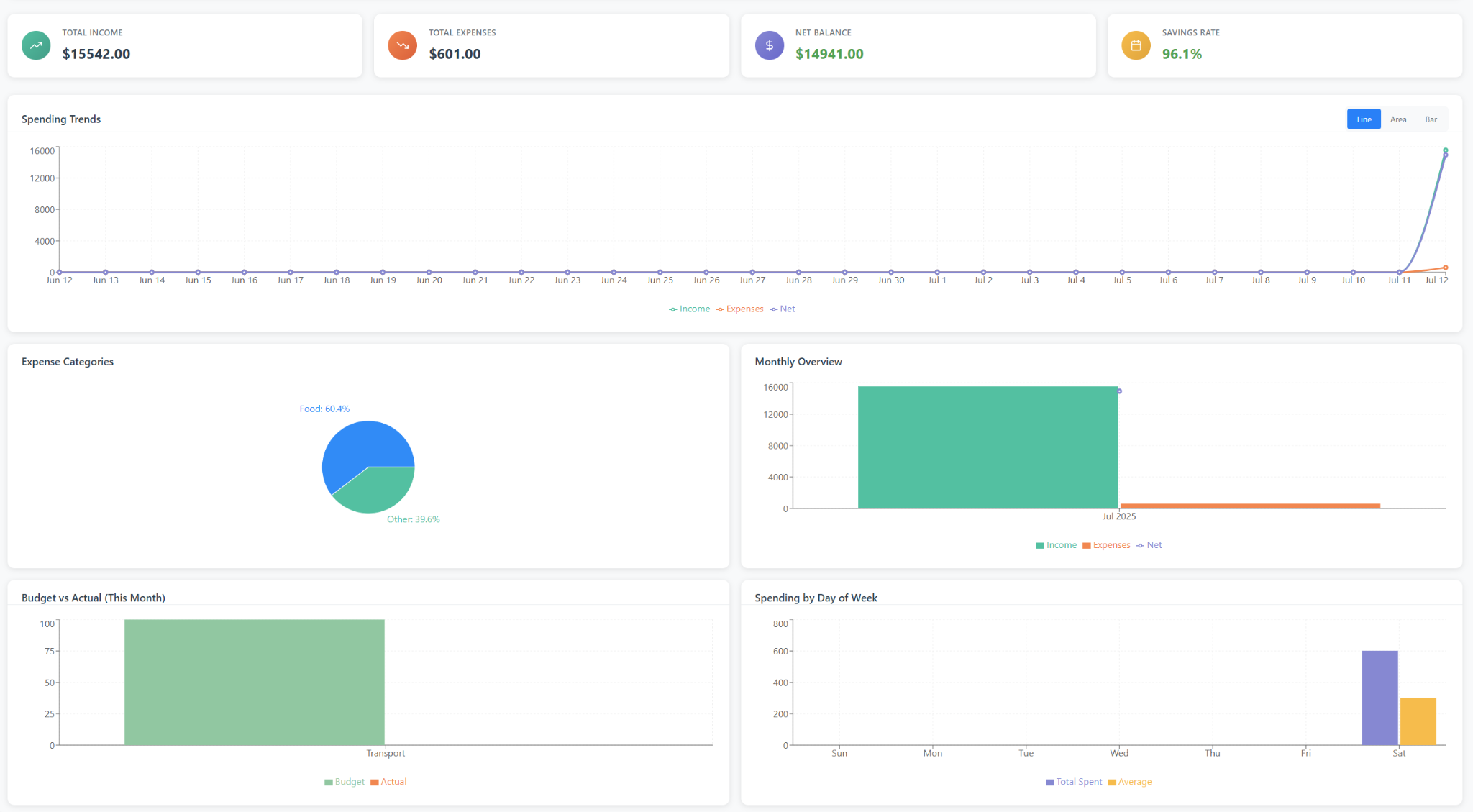
The AI Tracker section provides users with a smart financial assistant powered by GPT. Here, users can ask personalized questions about their financial habits, such as "How can I save more?" or "Am I overspending?" They may also choose from preset suggestions that explore spending trends, budget recommendations, and saving strategies. The AI assistant delivers instant, context-aware responses to help users improve their financial planning.





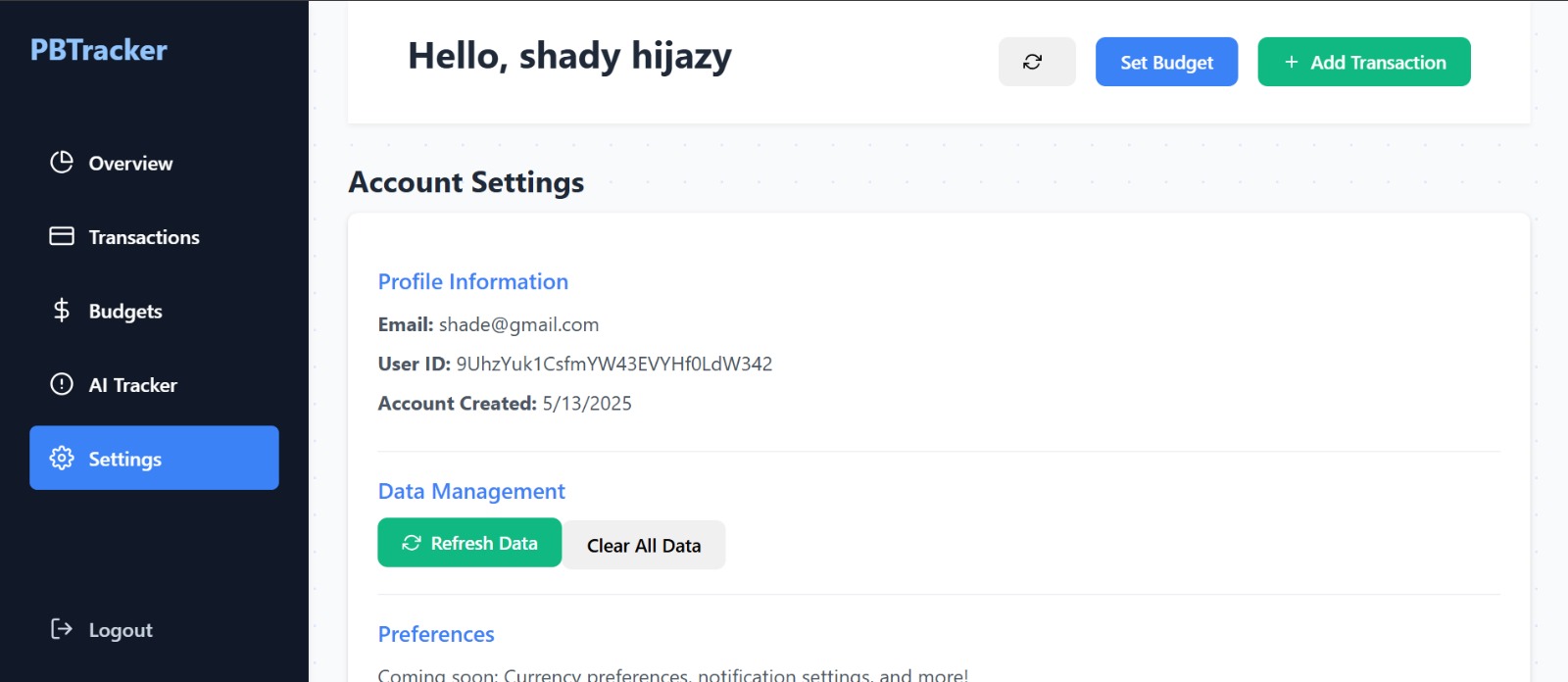
### 2.2.8. Financial Insights Dashboard

The Financial Insights Dashboard presents a comprehensive view of a user's income, expenses, and budget performance. It combines real-time data with interactive charts to provide personalized financial analytics. The top section highlights key statistics, including total income, expenses, net balance, and savings rate. Below, the user can explore visual trends in their spending over time, compare budget vs actual values, view expense distribution by category, and analyze spending behavior by day of the week. These insights help users better understand their financial habits and make smarter budgeting decisions. All data is dynamically updated from Firebase, ensuring real-time accuracy and seamless user experience.



### 2.2.9. Account Settings

Under the Settings section, users can view and manage their account information. The profile section displays the registered email, a unique user ID, and the account creation date. Users also have access to data management tools, including the ability to refresh their data or clear all data from their account. Preferences such as currency settings and notifications will be available in future updates. This section helps users maintain control over their account and ensures their data remains current.



# 3. Maintenance Guide

## 3.1. System Overview

The Personal Budget Tracker is a full-stack web application built using modern JavaScript technologies and cloud-based services. It provides users with tools to manage income, expenses, budgets, and financial insights using interactive charts and AI-powered features.

## 3.2 Directory Structure

The project follows a clean and modular directory structure designed to make development and maintenance intuitive and efficient. At the top level, the public/ folder holds static assets accessible to the browser. The src/ folder contains all source code, including components grouped into functional directories such as Home, Login, SignUp, BudgetDashboard, AiTrack, and Overview. The Firebase/ directory holds configuration logic for connecting to Firebase services.

The root also includes standard configuration files like .env, firebase.json, vite.config.js, and eslint.config.js. The entry point of the application is defined in main.jsx, which renders the main App.jsx component. Styling is managed through App.css and index.css. This structure enables clear separation of concerns, making the project scalable and maintainable.

## 3.3. Required Tools and Environments

To set up and maintain the Personal Budget Tracker system, developers need to have several software tools installed. These include Node.js (version 18 or later) for running the JavaScript runtime environment, and either npm or yarn in their latest stable versions for managing project dependencies. Firebase CLI is required for interacting with Firebase services such as Firestore, Authentication, and Hosting. Lastly, Git is used for version control and source code management.

### Hardware Requirements:

* Any modern development machine (Windows, macOS, or Linux)
* Recommended: 8 GB RAM or more for smoother experience

## 

## 3.4. Installation Instructions

Step 1: Clone the Repository (Personal\_Budget\_Tracker Folder)

Step 2: cd PhaseB

Step 4: cd PBT-PhaseB

Step 5: npm install

Step 6:cd backend

Step 7 : npm run dev (runs the server on render )

Step 7: cd..

Step 8: npm run dev

Step 9: Open The Local Host Link

Step 10 : Ready To Use

## 3.5. Updating the System

To keep the system up-to-date, developers should regularly update project dependencies using the npm update command. When upgrading core libraries like Vite or React, it is important to review their respective changelogs to ensure compatibility and apply any necessary code adjustments. Before implementing changes in Firebase (such as modifying Firestore rules or data structure), it's essential to test the modifications in a local environment to avoid disruptions in production. As a precaution, it is recommended to back up the Firestore database before making any schema changes.

## 3.6. Security and Access

User access to the system is managed using Firebase Authentication, which ensures secure sign-in and account protection. Access to user-specific data stored in Firestore is restricted through well-defined security rules that enforce document-level permissions. Sensitive configuration data, including Firebase API keys and the OpenAI key, should be stored in the .env file. This file must never be pushed to version control systems like GitHub to protect private credentials and maintain system integrity.

## 3.7. Common Maintenance Tasks

Routine maintenance ensures the continued stability and performance of the application. Developers should regularly monitor user data and activity through the Firebase Console, reviewing authentication logs, Firestore usage, and performance metrics. It is also advisable to rotate API keys and credentials on a scheduled basis to enhance security. When issues arise, error logs from both Firebase and browser development tools can provide valuable insight. Additionally, the AI Tracker should be periodically tested to confirm that the integration with the GPT API is functioning correctly and delivering relevant financial insights to users.

## 3.8. Future Enhancements

To expand the functionality and value of the Personal Budget Tracker, several future improvements can be considered. One enhancement could be to add support for handling multiple currencies, which would make the platform more accessible for international users. Implementing a dark mode theme would also improve usability and user comfort. The AI assistant could be extended with more advanced logic and personalized insights. Additionally, the system could offer options to import transactions directly from bank statements (via CSV or API), and automated email alerts could notify users when budget thresholds are exceeded, enhancing financial awareness and control.

# 

# 4. Key Application Usage & AI Behavior

## 4.1. Database Changes When Adding a Transaction

When a user adds a new transaction, the following data is updated in Firebase under the user's UID path:

* A new document is created under users/{uid}/transactions with the following fields:  
  + amount: Numeric value (positive for income, negative for expense)
  + category: Expense or income category (e.g., "Food", "Transport")
  + date: Timestamp of the transaction
  + description: Free-text description of the transaction
  + type: Either "income" or "expense"

These changes are instantly reflected in the UI using real-time Firebase listeners.

## 4.2. Realtime Calculations and Budget Tracking

Each time a transaction is added, the system performs several automatic calculations:

* **Total Income**, **Total Expenses**, and **Net Balance** are recalculated instantly and displayed on the dashboard.
* Each expense is checked against the corresponding **budget category** under users/{uid}/budgets. The system calculates how much of the budget has been used, and visual indicators (progress bars) update accordingly.
* If a budget is exceeded, the system shows **visual warnings**, such as red bars and alert messages, indicating the user has gone over their allocated limit.

This dynamic recalculation ensures users always have up-to-date financial insights.

## 4.3. AI Assistant Activation & Context

The AI Assistant is available through a dedicated "AI Tracker" section in the application. It is powered by the OpenAI GPT API and is **only activated when the user asks a question manually.**

When a question is submitted:

* The system fetches relevant user data (e.g., recent transactions, budgets, spending summaries).
* This context is included in the prompt sent to the OpenAI API to generate personalized, accurate financial advice.

The assistant supports both **free-form questions** and **preset suggestions** such as:

* "Am I overspending?"
* "How can I save more?"
* "Review my top expenses"

## 4.4. Example Use Cases of AI Assistant

Below are typical examples of how users can interact with the AI Assistant and the type of valuable feedback it provides:

**Example 1 – Overspending Alert**

* **User:** “Am I spending too much on transportation?”
* **AI:** “In the last 30 days, you spent ₪220 on Transport, which exceeds your budget of ₪150. You may consider using public transit more frequently.”

**Example 2 – Saving Strategy**

* **User:** “How can I save 500₪ in two months?”
* **AI:** “You could save ₪250 per month by reducing discretionary expenses such as dining out and shopping. Consider setting a strict limit for non-essential categories.”

These examples show that the AI assistant provides actionable and context-aware insights, even in simple budget-related scenarios.

