**SmartBudget**

CMPS 4910 Senior Project 1

Group Members: Ryan Reddemann, David Carter, Omer Rohina, Josef Chicas

**Project Objective:**

To design and implement a browser-based personal finance app that allows users to register, log in securely, add and categorize income/expenses, view real-time analytics, and track their financial habits over time. The app will feature a user-friendly UI, dynamic data visualization, and persistent storage using a cloud-hosted MongoDB database. Once logged in, users will be able to add, edit, and categorize their income and expenses. The application will offer real-time, interactive dashboards that provide insightful analytics and visualizations, enabling users to identify spending patterns, monitor savings goals, and track financial habits over time. To ensure data integrity and availability, the app will utilize a cloud-hosted MongoDB database for persistent storage, supporting secure, scalable, and efficient management of user financial records. The user experience will be enhanced through a responsive and modern UI, designed for both desktop and mobile platforms, with dynamic visual elements powered by charting libraries and asynchronous data updates.

**Target Market:**

The target audience for this application is anyone who may need to keep track of their spending habits and decisions. This can include students, freelancers, and young professionals who may need extra assistance with their finances. People more experienced in making financial decisions can also use our application for some of its features, such as quick visualization of spending habits with graphs, as well as savings goals. The application is mainly intended to be utilized on a PC, but can also be accessed on mobile for quick convenience.

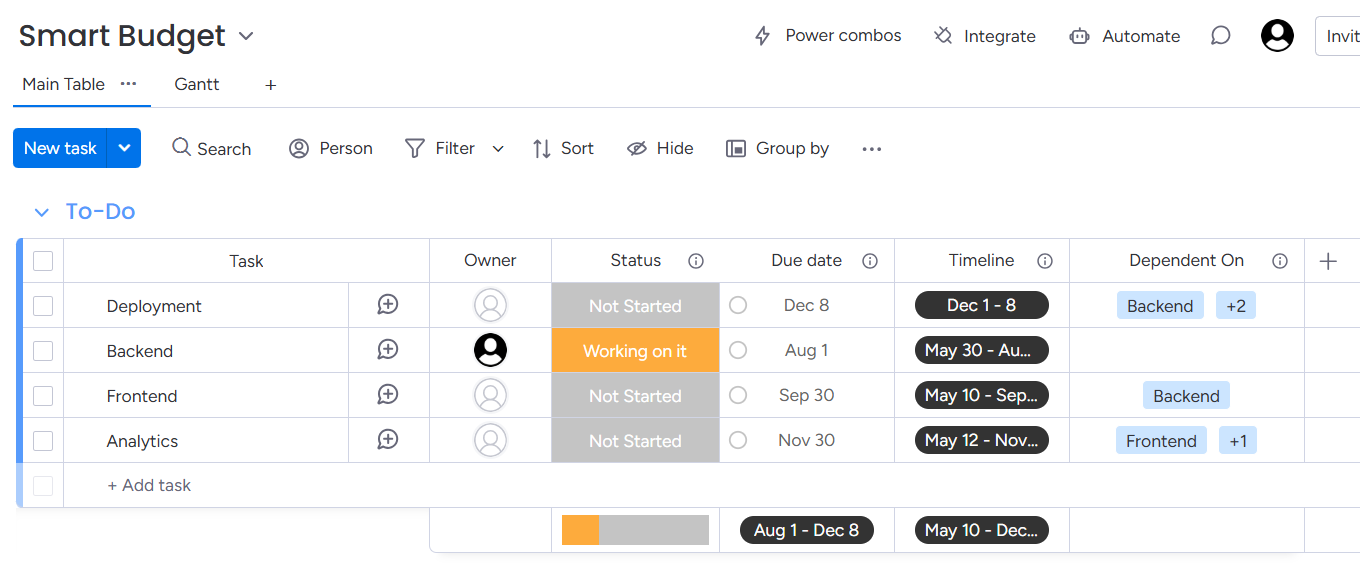
**Competitors:**

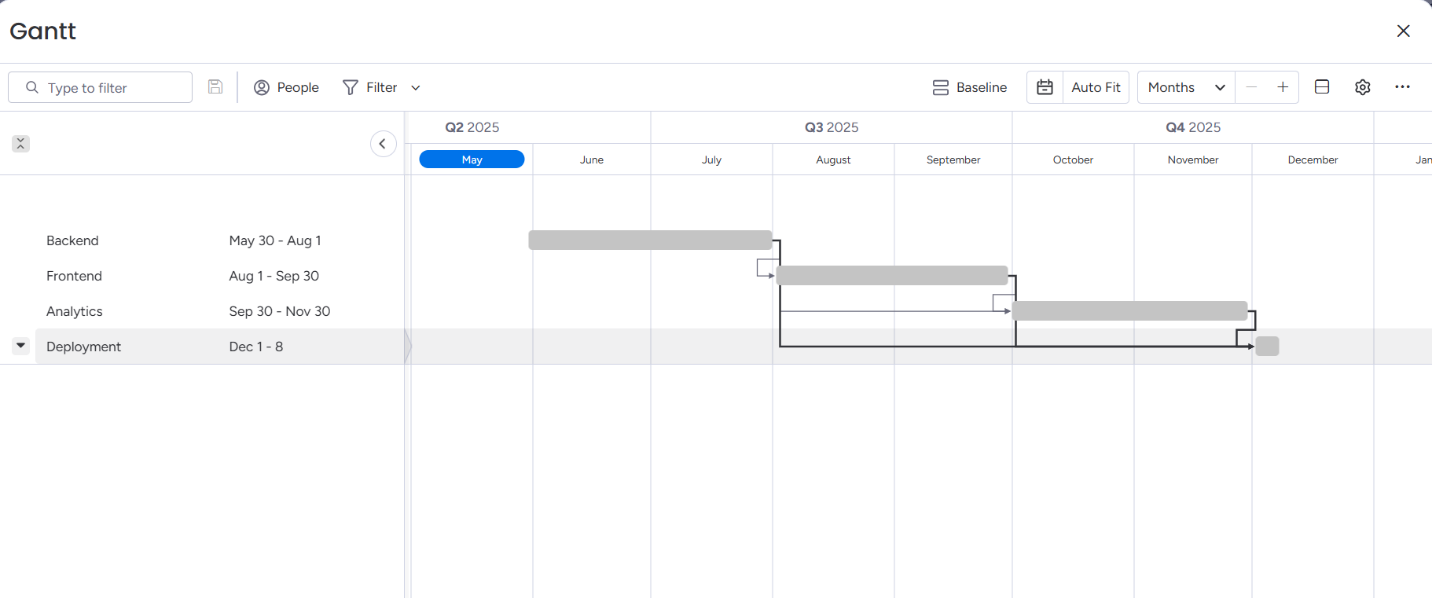
Similar competitors in this market include Mint, PocketGuard, and Excel Sheets. These applications have similar functionality to the application we are developing, such as creating charts to visualize the user's spending habits as well as savings goals that the user can create. However, these applications have their own fair share of issues that may make it intimidating for people first starting to make their own financial decisions. For example, PocketGuard has had harsh price increases of up to over 200%, and Mint was moved to Credit Karma, which lacked some of the previous features that Mint previously had. Decisions such as these are hostile towards beginners who may choose to forgo any financial planning app and may make poor decisions as a result. Our application plans to create a more friendly yet feature-rich application intended for beginners to make the best possible financial decisions they can make.

**Team Description:**

Our team is comprised of four people, each assigned a specific task in the project, but can also help in other tasks as needed.

**Working Plan:**

****

****

**Tech Stack Overview:**

| **Layer** | **Technology** | **Purpose** |
| --- | --- | --- |

| Frontend | React.js | Component-based UI |
| --- | --- | --- |

|  | Material-UI (MUI) | Ready-to-use UI components (buttons, forms, cards, modals) |
| --- | --- | --- |

|  | Chart.js | Data visualization (pie, line, bar charts) |
| --- | --- | --- |

| Backend | Express.js | REST API server |
| --- | --- | --- |

|  | Node.js | JavaScript runtime |
| --- | --- | --- |

| Database | MongoDB Atlas | NoSQL database for user data & transactions |
| --- | --- | --- |

| Auth | JSON Web Token (JWT) | Secure user authentication |
| --- | --- | --- |

| Security | bcryptjs | Password hashing |
| --- | --- | --- |

**Major Components & Features:**

**1. User Authentication**

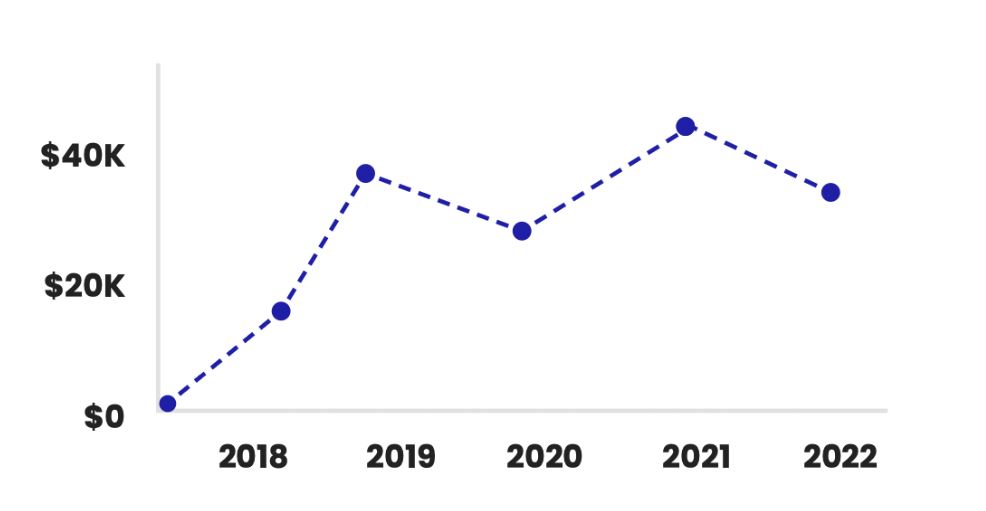
* Registration and Login forms
* Encrypted password storage using *bcryptjs*
* JWT-based authentication system
* Protected routes (only authenticated users can access the dashboard)

**2. Transaction Management**

* Add transactions with:
  + Type (income or expense)
  + Amount
  + Category (e.g., Food, Rent, Salary, Shopping)
  + Date
* View all transactions in a styled table
* Delete transaction functionality
* Transaction form using MUI TextField, Select, and Button

**3. Analytics Dashboard**

* Summary cards (Total Income, Total Expenses, Balance)
* Charts using Chart.js:
  + Pie chart: Spending by category
  + Line/Bar chart: Expenses over time
* Real-time updates based on the selected date range

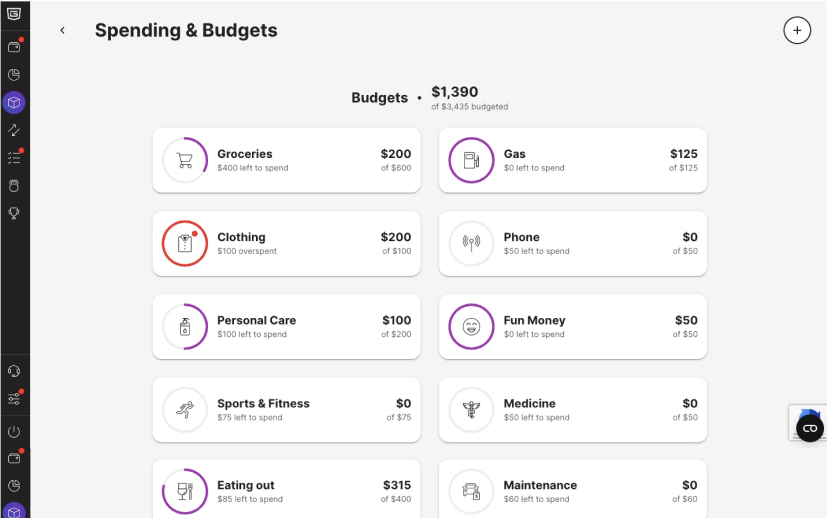


A pie chart with numbers and text

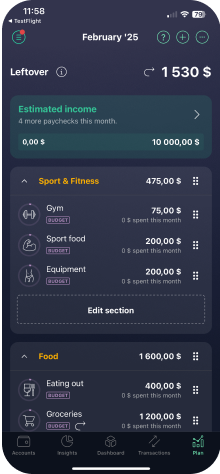
AI-generated content may be incorrect.

**4. UI/UX Design**

* Responsive design using Material-UI Grid, Card, Paper
* Consistent theme and color palette
* Navigation bar and sidebar with icons



*Example interface from PocketGuard listing expenses the user has listed in the application for the PC client*



*The UI for the mobile client for Pocketguard, also listing the Users expenses.*

**5. API Routes**

**Handled in Express (routes/)**

* POST /auth/register → Register new user
* POST /auth/login → Authenticate and return JWT
* POST /transactions/add → Add a transaction
* GET /transactions → Fetch all transactions for the logged-in user
* DELETE /transactions/:id → Delete specific transaction

**Team Roles:**

**Backend + API Developer (Omer Rohina)**

**Responsibilities:**

* Set up Express.js server
* Connect to MongoDB using Mongoose
* Create and test all API routes:
  + POST /transactions/add
  + GET /transactions
  + DELETE /transactions/:id
  + POST /auth/register & POST /auth/login
* Implement JWT Authentication
* Manage .env configuration for secrets

**UI Designer + Frontend Setup (Josef Chicas)**

**Responsibilities:**

* Set up React.js project
* Choose and implement a design library (Tailwind CSS)
* Design and build main pages:
  + Login / Signup
  + Dashboard
  + Add Transaction form
  + List of Transactions

**Budget Analytics + Chart Visuals**

**Responsibilities:**

* Build the dashboard view with analytics:
  + Total income / expenses / balance
  + Filter by date/category
* Use Chart.jsto show:
  + Pie chart of expenses by category
  + Line/bar chart over time
* Fetch data from backend and update charts dynamically

**Final Deliverables**

* Fully working web app (live link)
* GitHub repo with full codebase
* Documentation (README, screenshots)
* Live demo