**SmartBudget**

CMPS 4910 Senior Project 1

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

**Table of Content**

**Section Title Page**

1 Executive Summary 3

2 Introduction 3

2.1 Target Market 3

2.2 Competitors 4

2.3 What Makes Our Product Unique 4

2.4 Team Description 5

2.4 Work Plan (with Gantt Chart) 5

2.4.1 Major Goal 7

3 Specific Requirements 8

3.1 User Interfaces 8

3.2 Communication Interfaces 9

3.3 Universal Design Elements 10

3.4 Database Requirements 10

3.5 Testing 10

3.6 Security 11

4 Works Cited 11

## **Executive Summary**

SmartBudget is a mobile app designed to help individuals take control of their personal finances through intuitive budgeting, real-time tracking, and visual analytics. Our goal is to build a tool that makes financial management simple, secure, and accessible — particularly for students, freelancers, and everyday users who want to understand their money better.

SmartBudget will feature user authentication, transaction tracking (income and expenses), category breakdowns, filtering tools, and chart-based analytics — all within a lightweight and responsive mobile app. We will use **React Native** to build cross-platform apps (iOS/Android), **Node.js with Express** for the backend, and **MongoDB Atlas** for cloud-based storage. Authentication will be handled using **JWT tokens**, and user data will be protected using **bcrypt hashing.**

## **2. Introduction**

This project aims to create a fully functional mobile budgeting app that stores all user data securely and displays it in easy-to-read charts. Users will be able to view their financial progress, set spending goals, and maintain better financial habits through constant visibility and insights.

### **2.1 Target Market**

Our target market includes:

* College students managing limited budgets
* Freelancers tracking irregular income and expenses
* Entry-level professionals who want to avoid overspending
* Individuals who want an alternative to ad-heavy or subscription-based apps

According to Forbes, over 80% of millennials use budgeting apps. However, many are overly complex, cluttered with ads, or require paid subscriptions. SmartBudget focuses on simplicity, speed, and privacy.

**2.2 Competitor Analysis**

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.

|  |  |  |
| --- | --- | --- |
| **App** | **Drawbacks** | **How We’re Better** |
| Mint | Ad-heavy, data sold to 3rd parties | No ads, no selling user dataNo ads, no selling user data |
| PocketGuard | Premium features behind paywall | |  | | --- | | All core features are free |  |  | | --- | |  | |
| GoodBudget | Envelope system may be too rigid | Flexible categorization + charts |
| Excel/Spreadsheets | Manual, non-visual | Real-time and graphical tracking |

### **2.2.3 What Makes Our Product Unique**

* **Mobile-first** design — optimized for smartphones, not just responsive web apps.
* **Cross-platform (Android + iOS)** using React Native.
* **No paid plans, no ads.** Full functionality is free and private.
* **Data visualization** through pie/bar charts.
* **Accessible UI** with universal design best practices.

**2.3 Team Description**

Our team consists of four members, each assigned to a specialized role: frontend development, backend development, analytics & visualization, and deployment & documentation. Together, we aim to build a secure, user-friendly mobile budgeting app using modern technologies. We collaborate through weekly syncs, shared GitHub repos, and agile task tracking. We will also help each other out if we run into issues to make sure the progress of our project is staying up to date.

|  |  |  |
| --- | --- | --- |
| Group Member | Role | Skill |
| Omer & David | Backend + Auth + API | Strong Node.js, Express, Mongoose, experience with REST APIs |
| Ryan & Josef | Frontend UI + Forms + Routing + Dashboard + Analytics | Skilled in React, JS, Material Design, mobile UX |
| Everyone | Integration & Deployment | Render deployments, testing, documentation |

## **2.4 Work Plan and Timeline**

* Set up repo, install dependencies, assign roles – Already done
* Backend: Auth, MongoDB models, transaction API – By August of 2025
* Frontend: Login, dashboard, form UI – By ending of September
* Analytics: Chart.js graphs, filters – By November of 2025
* Deployment, testing, polish, documentation – By December of 2025
* Even though we have these specific dates when we want to finish these parts of the project’s individuals will also separate work on their different parts of the project at the same time to make sure we’re all working.

**Timeline Chart:**

**A screenshot of a computer

AI-generated content may be incorrect.**

Figure 1: A screenshot of the timeline chart of how we’re planning out the timelines for the project

**Gantt Chart:**

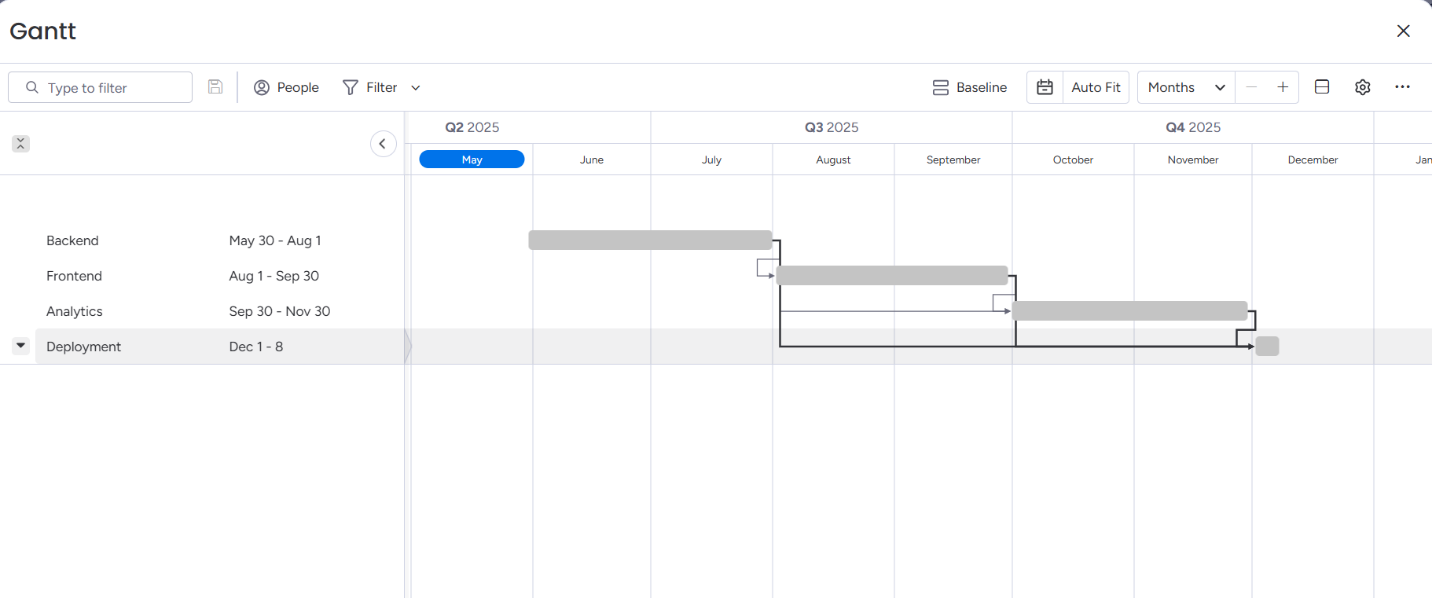
****

Figure 2: Here’s a GANTT chart that we made using our timeline to show the breakdown of how we want to complete the different parts of the project.

## **2.4.1 Major Goals**

1. Secure registration/login system (JWT + bcrypt)
2. Add/view/delete financial transactions
3. Categorize and filter transactions
4. Show balance, income, and expenses summary
5. Visual analytics with Chart.js
6. Deploy backend to Render and mobile frontend via APK builds

**1. User Authentication**

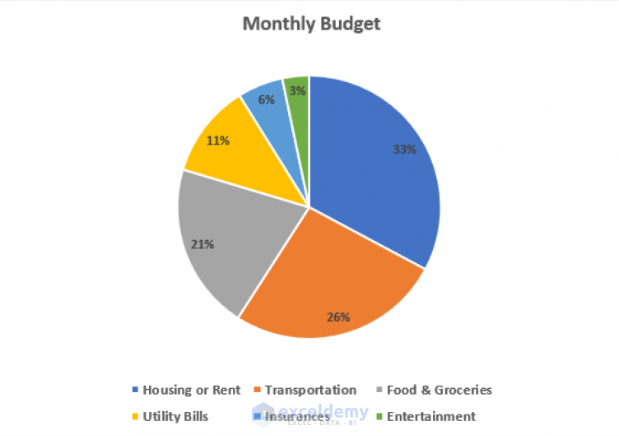
* Registration and Login forms
* Encrypted password storage using *bcryptjs*
* JWT-based authentication system
* Protected routes (only authenticated users can access the dashboard)

**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 line graph with dotted lines and numbers

AI-generated content may be incorrect.

Figure 3 & 4: These are two examples of what the analytics charts will look like. We will have a line chart and a pie chart.

## **3. System Requirements**

### Software:

* React Native (frontend)
* Node.js + Express (backend)
* MongoDB Atlas (database)
* Mongoose (ORM)
* JWT + bcrypt (security)
* Chart.js (visualization)

### Hardware:

* Android phone or IOS (test device)

## **3.1 User Interfaces**

* Welcome/Login/Signup
* Home Dashboard (summary cards + pie chart)
* Add Transaction (form with dropdowns)
* Transaction History (list with delete option)
* Filter by date/category
* Settings screen (theme toggle, logout)

**UI Concept :**

A screenshot of a mobile application

AI-generated content may be incorrect.

Figure 3: Here is an example screenshot of what our app would look like when a user is using it

## **3.2 Communication Interfaces**

* Frontend app sends HTTP requests to backend API (REST):
  + POST /auth/register
  + POST /auth/login
  + GET /transactions
  + POST /transactions/add
  + DELETE /transactions/:id
* Backend returns JSON responses.
* Token-based authentication using HTTP headers.

## **3.3** Universal **Design Elements**

* High-contrast color scheme
* Support for system-wide font size preferences
* Large, touch-friendly buttons
* Visual indicators for focus/selection
* Alt text for icons and symbols
* All features accessible without sound or animations
* No flashing or blinking effects

## **3.4** **Database Requirements**

We will use **MongoDB Atlas**, a cloud-hosted NoSQL database.

### Collections:

1. **Users**
   * \_id (ObjectId)
   * name
   * email
   * password (hashed)
2. **Transactions**
   * \_id (ObjectId)
   * userId (ref to Users)
   * type: income/expense
   * amount
   * category
   * date

Data will be accessed securely using Mongoose models.

**3.5 Testing**

Once the project is fully functioning, we will begin testing. During the testing we will challenge our app to make sure there are no flaws. Whether that is testing different characters for passwords or usernames or just making sure user’s data is being saved to the database we will need to test everything to make sure it works properly.

**3.6 Security**

Security is very important for our project because we handle users’ money. Making sure user’s credentials are safe and can’t be breached is something we will work on when thinking about security for our project.

**Work Cited**

J. Chamary, "Why Pokémon GO is the world's most important game," Forbes, 2018.

MongoDB Atlas, "Cloud Database as a Service," <https://www.mongodb.com/cloud/atlas>

React Native Docs, [https://reactnative.dev](https://reactnative.dev/)