University of Glasgow | School of Computing Science

Honours Individual Project Dissertation

Evergreen Budgeting: A Personal Budget Tracker

Kirsty Balfour

September 24, 2024

Abstract

Budgeting is an essential tool for maintaining financial stability, achieving personal savings goals, and building long-term financial resilience. While technology offers powerful means to support budgeting, existing tools are often either too flexible, resulting in complexity and users being overwhelmed, or too restrictive to accommodate user needs.

This project focused on implementing a personal budgeting application called Evergreen Budgeting. It was built using Angular, Node.js, and a PostgreSQL database, and it allows users to create a category-based budget, log and track income and expenses, set and manage savings goals, and generate financial reports.

The application was evaluated against its original requirements, focusing on usability, functionality, and overall experience. Results showed that most participants would prefer using Evergreen Budgeting over their current budgeting methods. However, issues related to mobile usability and the savings allocation approach were identified, highlighting key areas for future work.

Acknowledgements

Firstly, I would like to thank my project supervisor, Dr Sofiat Olaosebikan, for the opportunity to take on this project and her constant help and support over these last six months. Secondly, thanks go to my academic advisor, Professor David Manlove, who was always willing to provide advice and feedback.

I would also like to thank all my friends and family for their encouragement and for always being there to talk things through when needed. Lastly, special thanks go to my mother, who patiently dealt with far too many phone calls throughout this project, and my father, who shared his financial expertise and provided valuable feedback on the app's features and design.

Education Use Consent

I hereby grant my permission for this project to be stored, distributed and shown to other University of Glasgow students and staff for educational purposes. **Please note that you are under no obligation to sign this declaration, but doing so would help future students.**

Signature: Kirsty Balfour Date: 24 September 2024

Contents

1 Introduction

1.1 Motivation

Budgeting is a fundamental financial practice that allows individuals and families to manage their money effectively, enabling them to achieve personal goals such as saving for a holiday, purchasing a home, or preparing for retirement. At the same time, it plays a crucial role in day-to-day financial stability, helping people cover regular expenses, avoid debt, and handle unexpected costs. By providing a structured approach to tracking income and expenses, budgeting offers a sense of financial control and security, reducing uncertainty and stress (Mohd Azril Shukri 2024).

Despite its importance, financial literacy is often overlooked in formal education, leaving many young people unprepared to manage their finances when they first become financially independent (Stillwell 2016). Without a foundational understanding of budgeting principles, individuals may struggle to develop effective money management habits, leading to financial stress and potential long-term difficulties.

Technology presents a powerful solution to this challenge by offering intuitive tools that simplify budgeting, automate expense tracking, and provide insightful spending analysis. Unlike traditional manual methods, such as paper-based tracking or wholly mental estimation, digital solutions can leverage automation and data-driven insights to improve financial awareness and decision-making (Sonjaya 2024).

While various technological solutions already exist, such as spreadsheets and specialised budgeting applications, they come with notable drawbacks. Spreadsheets offer high customisability but can be overwhelming for users unfamiliar with either financial concepts or spreadsheet functionalities. Many budgeting apps, while simplifying the process, often lock essential features behind paywalls, focus primarily on reporting rather than budgeting, and tend to lack flexibility.

These limitations highlight the need for a new tool that finds a balance between usability and flexibility, providing an accessible, structured, and adaptable budgeting solution without complexity or cost barriers.

1.2 Goals

The aim of this project is to develop a mobile-friendly budgeting web application that enables users to track their spending, income, and savings goals while providing insightful financial analysis. The application should prioritise usability, flexibility, and accessibility, ensuring that both experienced budgeters and beginners can use it effectively.

To achieve this, the following key goals have been set:

- Users should be able to log, edit, and delete income and expenses with relevant details, including categorisation for better financial tracking.
- The application should implement a category-based budgeting system, allowing users to allocate funds effectively and monitor their spending within set limits.

- Users should be able to set and track savings goals, helping them work towards both short-term and long-term financial objectives.
- The application should generate reports and visualisations, such as spending breakdowns and trend analyses, to help users understand their financial habits.
- The interface should be mobile-friendly, intuitive and easy to navigate, ensuring that both experienced budgeters and beginners can use the application effectively.
- The application should feature a secure login system to protect user data, ensuring that financial information remains private and accessible only to the user.
- User data should be stored persistently, allowing them to access their financial history even after logging out or switching devices.
- Users should be informed when they exceed their budget, overspend in specific categories, or need to plan for upcoming expenses.

By fulfilling these goals, the project aims to create a user-friendly, secure, and flexible budgeting tool that empowers individuals to take control of their day-to-day spending and long-term financial planning without complexity or cost barriers.

1.3 Summary

This dissertation presents the development of a budgeting application aimed at enhancing financial management. The remainder of this paper is structured as follows:

- **Chapter 2** discusses the background and context of budgeting, exploring its importance in personal finance and examining how technology can address common challenges. It also provides a brief description of current solutions.
- **Chapter 3** details the analysis and requirements-gathering process, including idea generation, existing app analysis, user surveys, and MoSCoW prioritisation to establish the Minimal Viable Product (MVP) requirements.
- **Chapter 4** outlines the design of the system, covering the architecture, wireframes, database schema, and technology choices, while also discussing key design decisions.
- **Chapter 5** describes the implementation of the application, including software engineering practices, feature development, UI considerations, database integration, security measures, and deployment.
- **Chapter 6** evaluates the system's features and reliability, presenting the results of a user study and aligning them with the initial project requirements. This chapter also discusses the testing approaches used.
- **Chapter 7** summarises the project outcomes, reflects on the development process, and suggests areas for future work to enhance the application's capabilities.

1.4 Relevant Links

For the reader's convenience, relevant links can be found below:

- Link to the hosted application: `https://evergreen-budgeting-app.web.app/`
- Link to the application repository: `https://github.com/kirstyb2003/evergreen-budgeting-app`
- Link to the project repository, containing meeting minutes, evaluation data, and other administrative documents: `https://github.com/kirstyb2003/Personal-Budget-Tracker`

2 Background

2.1 What is Budgeting and How is it Used?

Budgeting, as defined by Shim et al. (2011), is the process of planning how to manage your finances in order to meet goals and needs. It involves setting spending limits, tracking income and expenses, and making deliberate decisions about how to use money over specific periods. While originally developed in the context of business budgets, applying Shim's definitions to personal finances suggests that a budget can help people stay in control of their money, avoid overspending, prepare for future expenses, and ensure that their financial choices align with their priorities and goals.

There are several common methods for managing personal budgets. For example, the 50/20/30 rule, discussed by the University of Pennsylvania's Financial Wellness section (Penn Student Registration and Financial Services 2025), allocates 50% of income to needs, 20% to savings, and 30% to wants. Another popular strategy is "Pay Yourself First," where a predetermined amount is immediately set aside into savings at the start of each month, treating it like a recurring bill. This approach is widely endorsed by financial advisors, including Ramsey Solutions (Ramsey Solutions 2024*b*), a trusted personal finance education company. The zero-based budget method assigns every dollar or pound of income to a specific purpose, ensuring income minus expenses equals zero by the end of the period. While this method requires detailed planning, it is highly effective for avoiding impulse spending. Lastly, envelope budgeting involves allocating specific amounts into physical or digital "envelopes," each representing a different spending category. This approach improves spending awareness and helps prevent overspending by making it clear when funds in a particular category have run out (Ramsey Solutions 2024*a*).

Each one of these methods has been well established and widely recommended for years, whether by financial advisors, universities, or academic literature. For example, an article by Gorshkova et al. (2015) demonstrated how implementing a family budgeting system, incorporating techniques like envelope budgeting and "Pay Yourself First," can serve as an effective foundation for personal accounting.

The widespread use and credibility of these budgeting techniques help explain their frequent adoption in commercial budgeting tools. Their adaptability to a range of personal budgeting needs makes them a strong foundation for many technological solutions, as their familiarity can ease the transition for users looking to take advantage of the benefits that digital budgeting offers.

2.2 Why is Budgeting Important?

Budgeting is an essential practice for achieving and maintaining financial stability while reducing stress. A well-planned budget enables individuals to gain control over their spending and saving habits, making them more aware of where their money is going (Lake 2022). Furthermore, effective budgeting helps people meet their financial and savings goals by building a safety net. By accumulating savings, individuals can better prepare for unexpected expenses and avoid falling into debt (Ganti 2024).

Not only can budgeting help achieve and manage tangible goals, but it can also significantly improve mental and financial well-being. The Consumer Financial Protection Bureau defines financial well-being as a spectrum, from extreme financial stress to high satisfaction with financial situations, and emphasises that financial security depends on more than just income levels. In fact, financial well-being is described as

> A state of being wherein a person can fully meet current and ongoing financial obligations, can feel secure in their financial future, and is able to make choices that allow enjoyment of life.

(Consumer Financial Protection Bureau 2015)

Research investigating the impact of savings and personal budgeting on financial literacy and well-being shows that solid budgeting practices can significantly improve both objective and subjective financial well-being (Versal et al. 2023). Additionally, a study by Mohd Azril Shukri (2024) examining the effects of the COVID-19 pandemic on personal financial planning and mental health states that financial stress can negatively impact both mental and physical well-being. The study also emphasises that effective budgeting is "crucial for managing financial stress and promoting mental well-being."

Budgeting also plays a crucial role in enhancing financial literacy. Developing budgeting skills enables individuals to make informed decisions and adapt effectively to unexpected life changes (Gorshkova et al. 2015). According to Navickas et al. (2014), financial literacy can positively influence daily decision-making and can lead to higher savings rates, thereby improving the quality of life in the long term.

In summary, budgeting is a crucial practice that not only aids in managing finances but also improves overall well-being. By strengthening financial literacy, individuals can navigate their financial journeys more effectively, leading to a more secure and fulfilling life.

2.3 How Technology is Important in Budgeting

Technology has increasingly become a vital part of how people and organisations manage their budgets. While traditional budgeting methods often relied on paper-based ledgers and manual calculations, technological advancements have transformed budgeting into a more flexible, efficient, and data-driven process (Sonjaya 2024).

Budgeting apps, which are readily available through app stores, are central to this shift from traditional to technological methods. A functionality review of 45 top-rated budgeting apps found that many offer a range of features, including support for various transaction types and accounts, the ability to create and manage budgets, secure data handling, and transaction tracking (Alenazi and Sas 2024). These apps do not just make budgeting more convenient, they can also help individuals improve their financial literacy through practical use.

One of the key benefits of using technology to budget, especially for those who are just beginning to manage their own finances, is that it removes much of the complexity from the process. Understanding how to create and manage a budget can be a steep learning curve, and many people, particularly young adults, struggle to know where to start. Budgeting apps help ease the pressure in those early stages by simplifying the process and allowing users to focus on logging their spending and savings, rather than trying to fully grasp complex financial planning straight away.

This is particularly important given research highlighting that while many people are reasonably good at managing money day to day, younger adults are often much less confident in planning ahead and predicting future financial needs. In fact, studies on financial education in the UK show that people tend to lack the skills and knowledge needed to make informed financial decisions

(Stillwell 2016). These findings emphasise the need for more support in the early stages of financial management, especially around budgeting and future planning. In this context, budgeting apps offer an accessible and supportive starting point, helping to build confidence and ease users into financial literacy at a manageable pace.

That said, while these tools offer much potential, some studies have noted that many budgeting apps still focus more on tracking than on helping users build strong financial approaches. A review presented at the BCS Human-Computer Interaction Conference pointed out that although tracking is useful, more structured support for actual budgeting practices is often neglected (Alenazi and Sas 2023). This demonstrates the need for improvement in these tools, with a greater emphasis on the budgeting methodology to help users develop consistent budgeting habits without overwhelming complexity.

Overall, technology does not replace the need for financial awareness, but it can make budgeting less daunting and more effective, particularly for those just starting their financial journey.

2.4 Relevant Existing Work & Technologies

Budgeting apps are a popular way for individuals to manage their finances, with many apps offering differing levels of support, from fully manual entry of transactions to automated approaches that involve users linking their bank accounts. Most apps adopt one of the methodologies described above, such as envelope, zero-based, pay-yourself-first, or percentage-based budgeting. More recently, some apps have begun making use of the data processing capabilities of modern technology, implementing predictive budgeting methods based on recurring expenses and the user's financial activity, such as the forecasting tools used by MoneyPatrol (n.d.).

From a technical standpoint, budgeting tools often involve technologies such as open banking APIs - as outlined by Open Banking's approved list of budgeting apps (Open Banking n.d.) - to allow secure linking of the app to bank accounts, cloud-based data storage for syncing across devices, and data visualisation techniques such as bar charts and line graphs. Some apps also offer more advanced features, such as automatic categorisation of transactions through the use of machine learning or personalised financial insights based on the user's behaviour.

The options for budgeting apps are vast, ranging from basic tools that rely on manual data entry to advanced platforms that integrate with multiple bank accounts, and in some cases, provide their own financial services such as savings accounts or investment products. A more detailed analysis of these apps, their strengths and limitations, and how they align with user needs is provided in Section 3.2.

3 Analysis/Requirements

3.1 Refining the Initial Specification

At the beginning of the project, a brief specification overview was provided by the project's supervisor, Dr Sofiat Olaosebikan. This outlined the broad goal of the project and some key features to include. The initial objective was to "develop a personal budget tracker application that helps users manage their finances by tracking income, expenses and savings." The key features requested included the ability to log transactions, categorise expenses, generate financial summaries and set savings goals. Some general technical specifications were also provided to guide the project, with the main requirements being the implementation of a secure login system and the persistent storage of data.

From this, an idea generation session was conducted. The main outcomes were that the app should be mobile-friendly; users should be able to allocate income across budget categories; surplus income should be automatically directed into savings; transactions should be schedulable and repeatable; and users should be able to create custom budget categories. The app would also generate reports over different time periods and allow users to set simple financial goals, such as monthly spending limits or savings targets.

To further inform the design, a comparison of existing budgeting apps was carried out (see Section 3.2), along with a user survey to explore expectations and feature preferences (see Section 3.3).

3.2 Existing Products Comparison

There are many budgeting apps already available on the market and in app stores, many of which share similar features and the aim to achieve financial goals such as managing spending, tracking income, and building savings. According to Forbes, who listed their top budgeting apps of 2024 (Thornhill 2024), these apps can be broadly split into two main categories: those that can be linked to users' bank accounts and those that cannot.

To better understand the features, limitations, and user experience offered by these apps, several examples from both categories were explored in September 2024. Where possible these apps were downloaded and tested with time taken to experiment with their functionalities. For both GoodBudget and Spending Tracker, user feedback from the Google Play Store was reviewed, noting positive and negative reviews, which can be found in Appendix B. Apps that rely on manual input were reviewed first, followed by those that support bank account integration and offer more advanced automation features. A summary of this review is presented in the tables below, while a more detailed analysis, along with screenshots of the apps, can be found in Appendix A.

3.2.1 Manual-Entry Budgeting Apps

App Name	Key Features	Strengths	Limitations
GoodBudget	• Envelope budgeting system • Basic charts • Data syncing (paid only) • Debt tracker	• Easy to use • Visual envelope progress bars • Secure (no bank link) • Encourages accountability	• Limited reports • Key features behind a paywall • Inconsistent mobile/web experience • Limited customisability to reports and transactions
Spending Tracker	• Blackboard-style UI • Basic customisable charts • CSV export • Tallies up income & expenses • No login required • Can schedule & repeat transactions	• Simple to use • Intuitive for beginners • No limits on number of bank accounts or custom categories	• Lacks detailed reporting • Best features behind a paywall • Cannot retrieve data without login • Data backup requires Dropbox account • Ad-heavy
Monefy	• Pie chart visualisation • Multi-device access • Quick transaction entry • Intuitive interface	• Easy transaction logging • Clean UI • Good overview of expenses	• Core features behind a paywall • No recurring payments or custom categories without paying • Only visualises expenses, not income

Table 3.1: *Manual-Entry Apps Feature Comparison*

	GoodBudget	Spending Tracker	Monefy
Free Version Available	✓	✓	✓
Charts & Reports	✓	✓	✗
Budgeting Focus	✓	✗	✓
Multiple Bank Accounts	✗	✓	✓
Custom Categories	✓	✓	✗
Exportable Data	✗	✓	✓
Data Syncing	✓	✗	✓

3.2.2 Bank-Integrated Budgeting Apps

The focus of this app study was on simple, manual-entry applications. This was due to the complexities and challenges associated with implementing open banking features, such as increased security requirements, greater emphasis on data protection, and the short six-month project timeframe. Some more advanced, bank-integrated apps were reviewed for comparison, though

in less detail, primarily to gather ideas on simpler design elements and features that could be adapted.

Table 3.2: *Comparison of Bank-Integrated Budgeting Apps*

App Name	Key Features	Strengths	Limitations
Snoop	• Spending insights • Savings tips • Automatic categorisation	• User-friendly UI • Free access to core features • Insightful suggestions	• Frequent miscategorisation of transactions • Limited customisation
PocketGuard	• Budget spending limits • Debt planning • Automatic categorisation	• Simple overview of spending • Custom budgets	• No free version • User-reported glitches • Unreliable bank syncing • Reliability concerns
YNAB	• Zero-based budgeting • Loan calculator • Net worth reports • Data syncing • Goal tracking	• Advanced features • Highly rated • Can share subscription between people	• High subscription cost • Limited transaction customisation • Steep learning curve
Rocket Money	• Financial insights • Subscription tracking • Handles bill negotiation • Offers a credit card	• Advanced subscription and bill management features	• Key features behind a paywall • Not available in the UK

3.2.3 Key Insights & Gaps Identified

From this comparison of both manual-entry and bank-integrated budgeting apps, several key insights and recurring themes emerged.

Key Insights:

- Manual-entry apps are simpler to use, allow greater user control and privacy, but often lack deeper financial insights.
- Bank-integrated apps provide more advanced features like data syncing, insights, and automatic categorisation, but they often struggle with reliability, data accuracy, and require the user to trust in the app's data security.
- Bank-integrated apps often promote additional financial products, which some users may perceive as prioritising the provider's commercial interests over user needs.
- Customisation and flexibility are valued by users, yet these aspects are often limited, especially in free versions.
- UI clarity and ease of use affect user satisfaction more than the number of features available.

- The budgeting methodology or theory behind the app can impact usability, as seen with users struggling to adopt YNAB's zero-based budgeting approach.

Gaps Identified:

- Many apps lock core features behind paywalls, reducing accessibility for budget-conscious users.
- Manual apps often lacked detailed reporting or visualisation tools, limiting users' ability to reflect on their financial behaviour.
- Several apps offered limited savings goal tools beyond basic budgeting.
- Syncing and backup features were either lacking or exclusive to paid versions, risking data loss or outdated views across multiple devices.
- Most apps did not offer custom views tailored to individual needs (e.g. payday-to-payday tracking, renaming transactions, viewing reports by payment method, etc.).

3.3 User Survey

3.3.1 Survey Design

The aim of this survey was to gather both quantitative and qualitative data on how people budget and how technology supports this process. Specifically, it sought to understand current budgeting habits, tools used, challenges faced, and which features are most useful or unhelpful. This information was intended to guide requirements gathering and design for the budgeting application.

The survey was shaped by the initial project ideas and insights from the comparison of existing budgeting apps. It aimed to validate whether the issues and priorities identified in the app comparison were reflected in user feedback while further exploring end-user needs.

Questions accommodated both those who budget and those who do not. Budgeters were asked about their budgeting frequency, methods, and satisfaction levels, while non-budgeters were asked why they do not budget and what might encourage them to start. Both groups were also asked about desired features in a budgeting app and which they considered most important.

Background information such as age range, tech comfort, and preferred devices for financial management was also gathered to identify correlations between characteristics and budgeting habits.

The full survey, including the introduction, consent form, and complete list of questions, is included in Appendix C.

3.3.2 Results & Conclusions Relevant to App Design

The survey received 51 responses, and the findings highlighted several key areas that should shape the design of the budgeting app. Full details of the results can be found in Appendix D.

Firstly, it is clear that the app needs to be mobile-friendly, as 63% of respondents manage their finances on their phones. However, laptops were also commonly used (24%), so offering a web-accessible version is equally important to ensure accessibility across devices (see Figure 3.1).

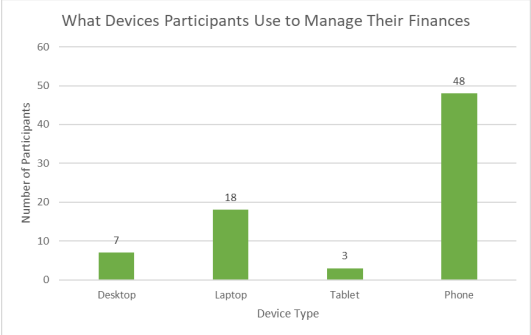


Figure 3.1: *The types of devices respondents use to manage their finances.*

Users typically manage their budgets on a weekly (36%) or monthly (43%) basis, so the app should prioritise these reporting periods to align with users' existing habits. The ability to toggle between these timeframes would also support flexibility for different budgeting styles.

Many respondents still rely on spreadsheets (32%) or pen and paper (29%), largely due to their simplicity, flexibility, and the control they offer (see Figure 3.2). This reinforces the importance of making the app intuitive and customisable while avoiding the rigidity that discourages users from switching to digital solutions.

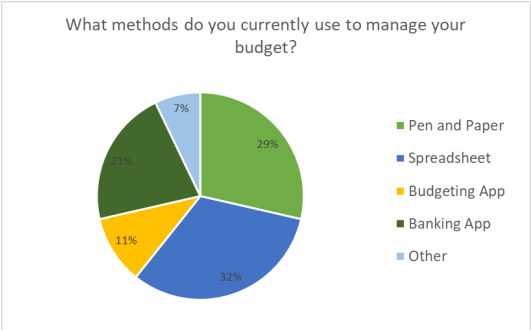


Figure 3.2: *Chart showing the different tools and methods respondents use to budget.*

Only 11% of participants currently use a budgeting app, with the main reasons for avoiding them being a lack of awareness (44%), preference for existing methods (35%), and a perception that budgeting apps are too complex or stressful (17%). Despite this, 92% of respondents said they would either consider or be open to trying a budgeting app in the future, showing strong potential demand for an approachable and helpful solution.

The most requested features included:

- Spending and income tracking
- Categorisation of expenses (e.g. "money pots" or envelope-style budgeting)
- Charts and reports to visualise spending habits
- Savings goal tracking
- Notifications and alerts for limits, bills, and weekly summaries

These should form the foundation of the minimum viable product (MVP), especially as they were ranked as the most important by users. Additional suggestions such as banking integration, educational budgeting tips, and customisable reports were also mentioned but may be better suited as future enhancements.

Ease of use is especially important, as many users found existing apps too complicated or overwhelming. Several respondents who do not currently budget said the process felt like too much effort or that they were unsure where to begin. However, many also indicated that the right tool, one that is clear, useful, and simple, would encourage them to start budgeting.

In conclusion, the app should focus on simplicity, flexibility, and usability while delivering core budgeting features. It should also aim to reduce the perceived friction around budgeting, particularly for beginners, by making the experience feel achievable and helpful from the outset.

3.4 Initial Requirements

The following requirements were derived from the initial specification and idea generation session; the comparison of existing budgeting apps; and the user survey. All potential requirements were organised into functional and end-user categories. Functional requirements describe what the system must do to enable user actions, while end-user requirements outline the actions and goals users wish to accomplish (GeeksForGeeks 2024*a*).

3.4.1 Functional Requirements

During early planning, it was discovered that deploying a mobile app through the Google Play Store or Apple App Store required separate developer accounts and various testing and verification processes (Google 2024; Apple 2024). Since the aim was to minimise project costs and ensure the application remained easily accessible, it was decided to implement a Progressive Web Application (PWA). A PWA is a web application that can be installed on a user's device and behaves like a native app (Mozilla 2025). This mitigated platform-specific concerns, allowed simultaneous development of web and mobile versions, and ensured users could access the app directly through a browser, without relying on an app store.

The resulting functional requirements are as follows:

- **Tech Stack**: React.js frontend, Node.js backend, PostgreSQL database, and deployment via GitHub Pages.
- **System Security and Data Protection**:
  - Users must be able to securely log in and access their personal financial data.
  - Sensitive data, such as passwords, should be protected using secure practices such as hashing before storage.
  - The system must prevent unauthorised access and comply with data protection principles such as GDPR and the Data Protection Act (DPA).
  - User data must persist between sessions.
- **Database Requirements**:
  - A relational database (PostgreSQL) will be used to store user financial data.
  - Key entities include: User, Transaction, Bank Account, Category, and Budget.
  - The database must support full CRUD operations and maintain appropriate relationships between entities.
- **Persistent Storage**: User data must be stored securely on the server and persist between sessions.
- **Device Support**: The application must be optimised for both mobile and desktop use, with responsive UI components to ensure usability across screen sizes.
- **Automatic Categorisation**: Transactions should be categorised based on their name or shop, using rule-based logic or optionally enhanced with machine learning.

3.4.2 End-User Requirements

Users should be able to:

- Securely log in and manage multiple bank accounts
- Add, edit, delete, and repeat income, expense, and savings transactions
- Set budget limits by category and receive alerts when overspending occurs
- Manage and track savings goals:
  - Identify categories to reduce spending on in the next month
  - See increases in specific expenses over time (e.g. rising electricity bills)
  - Set specific savings targets (e.g. save £200 this month)
  - Receive reminders and prompts related to savings goals
- Generate financial reports based on:
  - Transaction categories, shops, bank accounts, or transaction types
  - Comparisons between two different time periods (e.g. this month vs the previous month)
  - Time periods such as daily, weekly, monthly, pay-check to pay-check, or yearly
- Filter/sort transactions by date, type, category, shop, or bank account
- Search transactions by name or keyword
- Export transaction reports and data as PDF or Excel files

3.5 MoSCoW Prioritisation & the Minimal Viable Product (MVP)

MoSCoW prioritisation is a method of organising requirements into categories based on their importance and feasibility (ProductPlan n.d.). The categories are:

- **Must Have**: essential product features required for a usable MVP
- **Should Have**: important, but not vital for MVP, but still add significant value
- **Could Have**: features that can improve user experience but have minimal effect if left out
- **Won't Have**: features that were considered but excluded from this version

This method was applied to categorise the initial requirements, from the most essential features to the least critical ones.

3.5.1 The Must Haves & The MVP

The MVP is made up of all the Must Have requirements, which are listed below:

- The user should be able to:
  - Securely login to retrieve their data and be confident others cannot view their sensitive financial data
  - Set their default currency when setting up their account
  - Log their spending and income with appropriate detail
  - Edit/delete any of their transactions (with specific options for handling repeated transactions)
  - View reports/charts on their finances (e.g. pie charts for categories, line graphs for trends) in weekly, monthly, and yearly periods
  - Set and track budgets and savings goals
  - Check if they have enough funds to pay bills/upcoming expenses
  - Clearly see if they have overspent on a category
  - Receive a notification or alert if they overspend on their overall budget
- The app should have persistent storage of the users data even after they leave the app
- The application should be mobile friendly and have PWA capability

3.5.2 The Should & Could Haves

These features would be included in the MVP if time permits, but they were deprioritised to non-essential for the initial version of the app.

Should Have

The user should be able to:

- Set the currency of each transaction
- Create custom categories
- Create and track multiple bank accounts
- Edit/delete their bank accounts
- View additional chart types (e.g, spending per shop)
- View charts in additional reporting periods (e.g. fortnightly)
- Generate bank-account specific reports
- Set concrete savings goals e.g. put £200 away in a savings account this month
- Make use of advanced transaction filtering and sorting

Could Have

The user could:

- Convert transactions in currencies different from their default currency into their default currency using current exchange rates
- Log repeated transactions where the payment amount may change each time
- View charts in additional time periods (daily, pay-day-to-pay-day)
- Export reports (PDF, Excel)
- Search transactions by name
- Generate cash flow visualisations
- View suggestions on which categories they should reduce spending
- Receive notifications when their bills increase
- Attach interest rates to their bank accounts so that they can view how much interest each account is earning

3.5.3 The Won't Haves

This version of the application will not include:

- Automatic categorisation of transactions
- Bank account integration

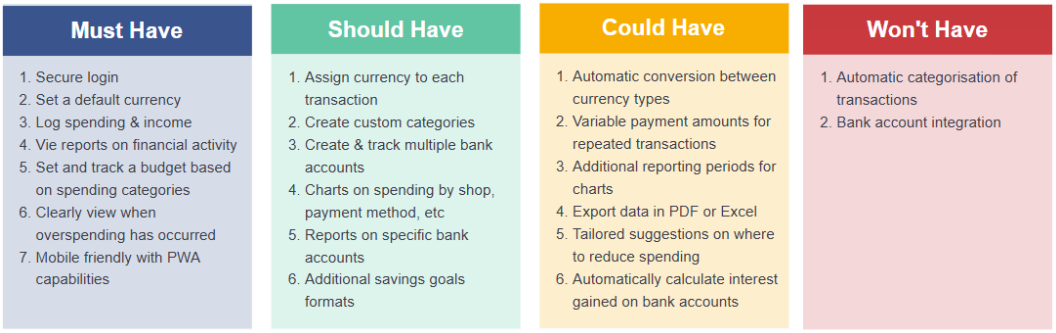


Figure 3.3: *Summarised MoSCoW prioritisation*

4 Design

4.1 Architecture Design

4.1.1 Initial Design

The initial architecture was designed as a web application using the React.js framework for the frontend, a Node.js server for middleware, and a PostgreSQL database for backend storage. The frontend was also intended to include Progressive Web App (PWA) capabilities. This setup supported the goals of creating a system that was scalable, accessible, and cost-effective to deploy across multiple platforms.

React was selected as it is one of the most widely adopted JavaScript libraries for building dynamic and responsive UIs. Its large developer community, comprehensive documentation, and reusable component structure make it a strong choice for rapid development (Komodo 2022). The component-based architecture also promotes clear separation of concerns and simplifies long-term UI maintenance.

The decision to use a PWA aimed to improve accessibility and usability, especially on mobile devices. PWAs combine aspects of web and native apps, allowing users to install the application directly onto their devices without an app store (Mozilla 2025). This reduces friction for users and avoids the overhead of maintaining separate iOS and Android apps.

Node.js was chosen for the backend due to its strong performance with asynchronous operations and seamless integration with JavaScript-based frontends like React. It is lightweight, efficient, and well-suited for building RESTful APIs (GeeksForGeeks 2024*b*).

PostgreSQL was selected as the database system for its robust relational capabilities, support for complex queries, and strong data integrity via its ACID-compliant architecture. It integrates well with Node.js through tools like node-postgres and supports advanced data types and indexing (PostgreSQL n.d.; npm 2025).

4.1.2 Final Design

Before development began, the frontend framework was switched from React to Angular. The primary motivation was Angular's more structured setup, which simplified project management and scaling as the application became more complex.

A key advantage of Angular is its built-in support for features such as routing, form validation, and state management which are critical for a financial system handling significant volumes of user data. In contrast, React would have required third-party libraries to achieve similar functionality. Angular also offers native PWA support, streamlining the addition of offline and installable capabilities compared to React's more manual configuration (Sakthi S 2024).

The decision was further influenced by prior industry experience using Angular in a financial dashboard project, which also required strict data validation, modularity, and security. This background provided greater confidence in using Angular to build a reliable and secure system.

Angular is commonly adopted in enterprise applications, especially those handling sensitive data. It promotes best practices such as strong typing through TypeScript, automatic protection against common vulnerabilities, and a consistent project structure. Its use in platforms like PayPal demonstrates its suitability for financial applications (Sols n.d.).

The final system architecture is shown in Figure 4.1, and initial sketches of both the React and Angular versions can be found in Appendix E.

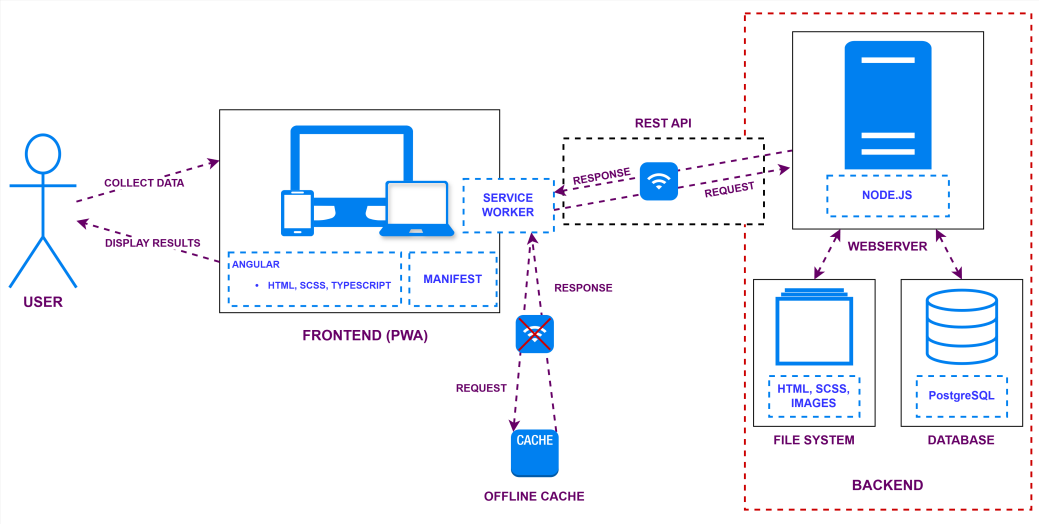


Figure 4.1: *Final architecture design using an Angular front-end with the PWA labelled.*

Service Worker *- manages offline access, push notifications and caching of the PWA.*
Manifest *- enables installable app on mobile and desktop.*

4.2 Database Design

4.2.1 Data Types & Storage Considerations

When designing the database schema, a number of informed decisions were made about the data types used and the structure of the entities.

Adherence to normalisation principles was important for reducing redundancy and dependency between records and fields. This was particularly important given the likelihood of strict storage constraints from a free hosting service, so space needed to be used efficiently. This focus on normalisation also helps to maintain data integrity, as it enforces constraints and relationships between entities. The theory outlined in Elmasri (2017) *Fundamentals of Database Systems* was followed, which resulted in the schema being in Boyce-Codd Normal Form (BCNF). This means that all inherent dependencies are removed, and each attribute in the entities is dependent only on its associated primary key. BCNF is considered the industry-standard level of normalisation that database designers strive to achieve, representing that the schema has been broken down into efficient and well-structured relations. Although these methods are often used to improve existing databases, the definition of BCNF was used to guide the design from the beginning, keeping in mind normalisation principles such as reducing null values, avoiding duplicate information, and ensuring that properties are only dependent on their necessary primary key.

The emphasis of data normalisation on reducing redundancy, along with the awareness of storage space limitations, led to the decision to not store calculated or intermediate values such as the user's current bank balance, category totals, total amount spent, and so on. Instead, it was planned for this responsibility to be handled by the front-end. While this approach may result in higher levels of processing and potentially impact performance, it allows more users to enter additional transactions and other financial data within the available storage constraints.

Security and compliance with data protection laws remained a priority throughout the design process, and it was established that user passwords would need to be hashed and stored securely. While hashing alone provides some protection against data breaches, the addition of a randomised salt to the hashing of users' passwords makes it significantly more difficult for attackers to use tools such as rainbow tables or offline dictionary attacks to access user accounts if the password data were compromised. Salts are important because they prevent duplicate passwords from being recognised or mixed up within the relation, and also help protect users who may reuse passwords across different systems from being identified (Salomon 2011).

Research and consideration were given to selecting appropriate data types for key fields. The serial data type was chosen for primary key IDs in each entity as this allows for up to 2 billion unique values that are automatically incremented. As this is a small-scale project, the assumption was made that none of the entities would ever contain more than 2 billion entries. For currency fields, guidance from Crunchy Data (n.d.) was followed, allowing for 9 digits, with 2 of those reserved for decimal places. Only two decimal places were used, as intermediate or calculated values were not stored, so further precision was not necessary. This choice allows money values up to 9,999,999.99, which should be more than sufficient for users using the app to manage their real-life finances.

4.2.2 MVP Database

Using the considerations mentioned earlier, the schema for the MVP was developed as shown in Figure 4.2. The data type decisions made for each property are outlined in Table 4.1.

The User entity stores login credentials, the user's default transaction currency, and their starting bank account balance upon sign-up. The Category entity stores fixed values for the different transaction and budgeting categories and cannot be edited by users. The Transaction entity holds all information about the user's transactions, including whether a particular transaction is recurring. The Budget table includes each category in the user's budget alongside the limit they have set for that category. Finally, the Savings_Goal entity stores the user's goal amount, their target date for reaching it, and how much they have saved towards that goal when creating it.

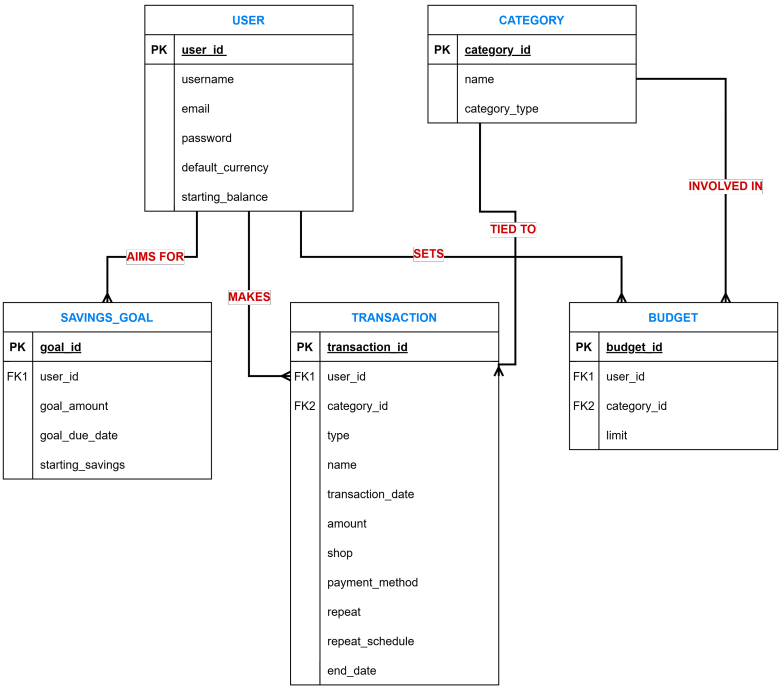


Figure 4.2: *ER diagram for the MVP. "PK" and underlines indicate primary keys and "FK" indicates foreign keys.*

4.2.3 Should Have & Could Have Databases

Schemas for the Should Have and Could Have requirements were also designed to ensure that, if these features were implemented, the necessary database structures would already be available.

Figure 4.4 shows both the Should Have and Could Have ER diagrams, with changes from the MVP highlighted in yellow. The new or updated properties are summarised in Table 4.3.

The additions in the Should Have schema are as follows:

- A new Bank_Account entity, allowing users to track multiple bank accounts
- A user_id property added to the Category entity, enabling users to create custom categories
  - This property would be set to null for the default categories available to all users
- A currency property added to the Transaction entity, so users can specify if a payment was made in a different currency than their default
- A bank_account_id property added to the Transaction entity, so users can indicate which of their bank accounts the transaction relates to

The further enhancements shown in the Could Have schema include:

- A variable_value property in the Transaction entity, indicating whether a repeated transaction may have different amounts each time
- A notes_and_tags property in the Transaction entity, allowing users to include additional details or tags about the transaction
- A spend_less property in the Budget entity, which indicates that the user wants to reduce spending in that category
- An interest_rate property in the Bank_Account entity, allowing the app to calculate and display the interest earned for each account.

Table 4.1: *MVP Property Table*

Entity	Property	Data Type	Constraints	Other
USER	user_id	serial	PK	
USER	username	varchar(30)	Unique, Not Null	
USER	email	varchar(255)	Unique, Not Null, Valid Email Format	
USER	password	varchar(255)	Not Null, Encrypted with a Salt	
USER	default_currency	varchar(3)	Not Null, in ISO 4217 Currency Codes format	
USER	starting_balance	numeric(9,2)	Not Null, default to 0.00	
CATEGORY	category_id	serial	PK	
CATEGORY	name	varchar(255)	Not Null	
CATEGORY	category_type	enum('expense', 'income', 'savings')	Not Null	
SAVINGS_GOAL	goal_id	serial	PK	
SAVINGS_GOAL	user_id	serial	Not Null	FK
SAVINGS_GOAL	goal_amount	numeric(9,2)	Not Null	
SAVINGS_GOAL	goal_due_date	datetime	Not Null	
SAVINGS_GOAL	starting_savings	numeric(9,2)	Not Null, default to 0.00	
TRANSACTION	transaction_id	serial	PK	
TRANSACTION	user_id	serial	Not Null	FK
TRANSACTION	category_id	serial	Not Null	FK
TRANSACTION	type	enum('expense', 'income', 'savings')	Not Null	
TRANSACTION	name	varchar(255)	Not Null	
TRANSACTION	transaction_date	datetime	Not Null	
TRANSACTION	amount	numeric(9,2)	Not Null	
TRANSACTION	shop	varchar(255)		
TRANSACTION	payment_method	enum('credit', 'debit', 'cheque', 'cash')		
TRANSACTION	repeat	Boolean	Default to False	
TRANSACTION	repeat_schedule	enum('daily', 'weekly', 'monthly', 'yearly')	Default to Null	
TRANSACTION	end_date	datetime	Default to Null	
BUDGET	budget_id	serial	PK	
BUDGET	user_id	serial	Not Null	FK
BUDGET	category_id	serial	Not Null	FK
BUDGET	limit	numeric(9,2)	Not Null	

4.3 UI Design

4.3.1 Wireframes

Since the application was going to be developed as a web app but also needed to be mobile-friendly, wireframes were designed for each page in both phone and desktop formats. An example of these two layouts can be seen in Figure 4.5.

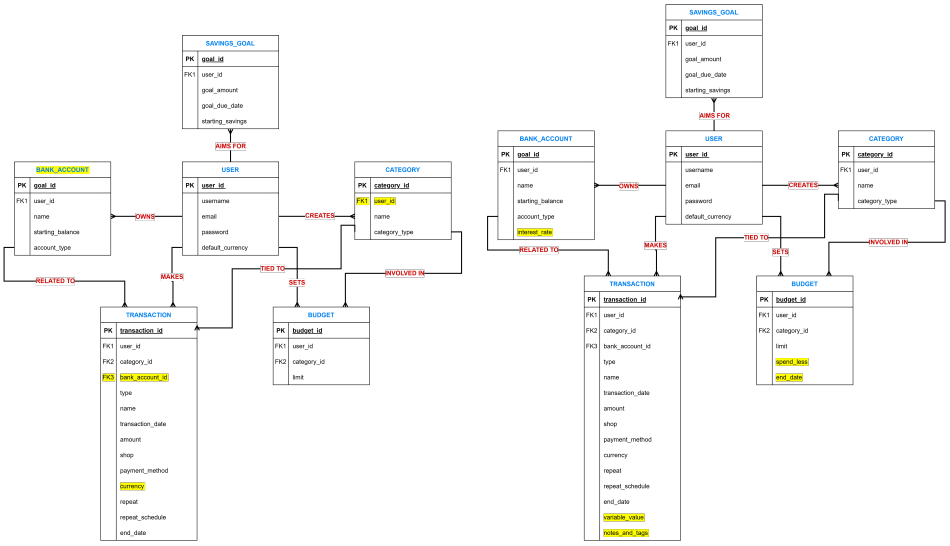
(a) *Should Have Property Table - New and modified properties since the MVP.*

Entity	Property	Data Type	Constraints	Other
CATEGORY	user_id	serial	Optional	FK
TRANSACTION	bank_account_id	serial	Not Null	FK
TRANSACTION	currency	numeric(9,2)	Not Null, default to the user default_currency	
BANK_ACCOUNT	bank_account_id	serial		PK
BANK_ACCOUNT	user_id	serial	Not Null	FK
BANK_ACCOUNT	name	varchar(255)	Not Null	
BANK_ACCOUNT	starting_balance	numeric(9,2)	Not Null, defaults to 0.00	
BANK_ACCOUNT	account_type	enum('savings', 'current', 'joint', 'ISA')	Not Null	

(b) *Could Have Property Table - Additional potential enhancements beyond Should Have.*

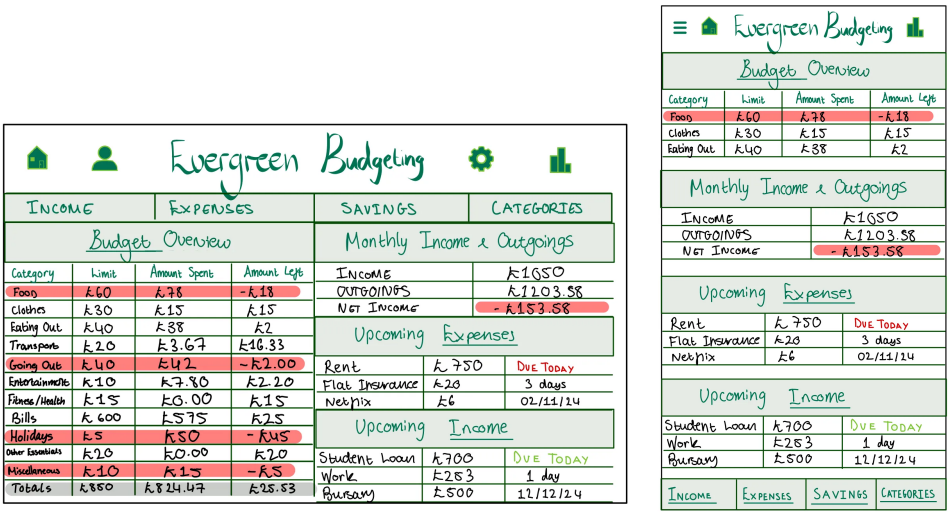
Entity	Property	Data Type	Constraints	Other
TRANSACTION	variable_value	Boolean	Default to False	
TRANSACTION	notes_and_tags	varchar(500)	Optional	
BUDGET	spend_less	Boolean	Defaults to False	
BUDGET	end_date	datetime	Default to Null	
BANK_ACCOUNT	interest_rate	decimal(5,4)	Not Null, default to 0.0000	

Figure 4.3: *New and modified properties introduced since the MVP. The Should Have table lists essential changes, while the Could Have table includes optional enhancements.*



(a) Should Have ER Diagram. *(b) Could Have ER Diagram.*

Figure 4.4: *Yellow highlights show the changes from the previous ER diagram. I.e. highlights in the Should Have diagram are changes from the MVP and highlights in the Could Have diagram are changes from the Should Have.*



(a) Desktop View *(b) Phone View*

Figure 4.5: *Wireframes for the app's home page.*

Before designing the look and feel of the UI, a name for the app was needed. A soothing and calming concept was chosen based on feedback from the user study, which highlighted that one barrier to using budgeting apps is that users can find them overwhelming. This led to a nature-themed concept.

The idea of "money trees" was considered initially, but this felt a bit too obvious. Eventually, the name Evergreen Budgeting was selected, a title that maintained the nature theme while subtly referencing the money tree idea. The word evergreen also evokes a sense of longevity and financial stability, as evergreen trees retain their leaves year-round. Additionally, green is commonly associated with money, making the name feel both relevant and approachable.

Once the name was chosen, the app's aesthetic and colour palette were tailored to suit it, with many elements adopting a green or nature-inspired look. A clean and minimal interface was still prioritised, as many potential users expressed a preference for simplicity and ease of use.

For displaying data, a table-based approach was chosen. The user survey indicated that spreadsheets are a popular budgeting tool, so this design choice aimed to mimic a familiar and preferred format for users. These tables are shown in Figure 4.6.

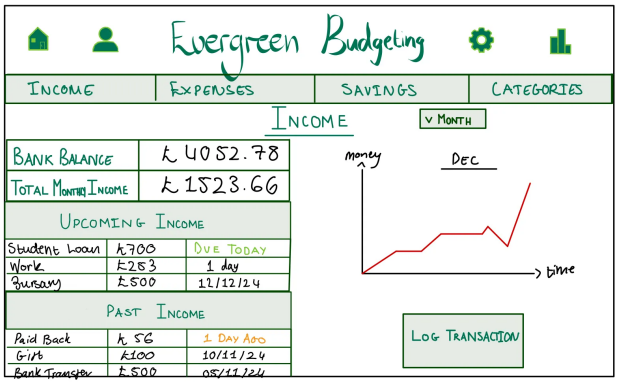


Figure 4.6: *Wireframe for the income page.*

For the reports page, the intention was to allow users to view multiple financial reports, each highlighting different aspects of their finances (see Figure 4.7). The charts were also designed to be customisable, allowing users to filter the data by time period and transaction type. Three charts were included:

1. A pie chart showing a breakdown of transactions by category. Users can toggle between expense, income, and savings categories, and select the time period they want to view.
2. A line chart showing income, expenses, and net balance over the chosen time period.
3. A bar chart displaying total spending across different time units (e.g. monthly totals for the past year).

For the budget page (Figure 4.8), a more visual and engaging approach was used rather than another table. In keeping with the nature theme, watering cans were designed to act as progress bars for each category in the user's budget. If a user overspends in a category, the watering can overflows and is highlighted in red to clearly indicate this.

Wireframes for the remaining pages (both phone and desktop views) can be found in Appendix F.

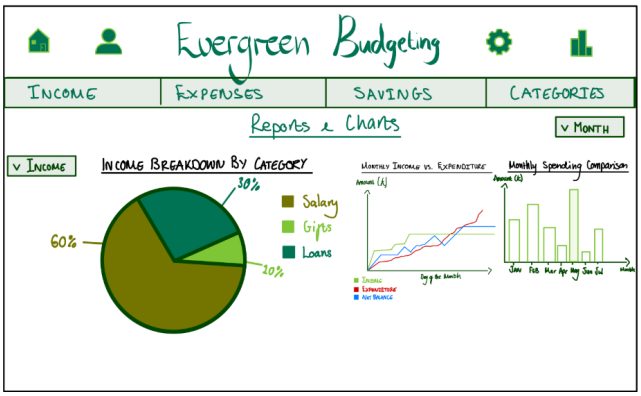


Figure 4.7: *Wireframe for the reports page.*

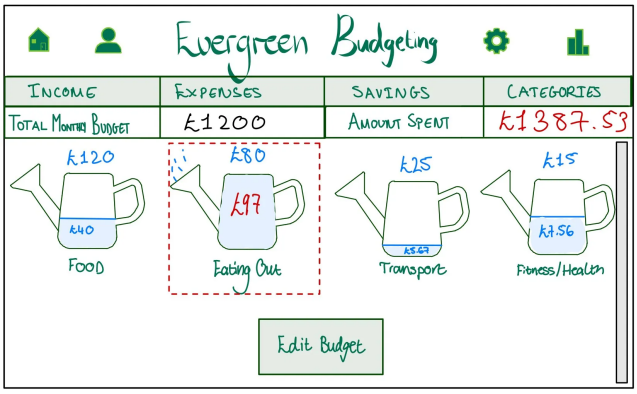


Figure 4.8: *Wireframe for the budget page.*

4.3.2 Logo

Time was also spent designing the logo for the application, following the wireframe stage, as a distinctive icon was needed for users' home screens. The process began by looking at logos from other budgeting apps and various tree-inspired designs. The aim was to steer away from money imagery and instead focus on themes of tracking and reporting, while still tying in with nature.

The final logo (Figure 4.9) reflects the app's overall colour palette and aesthetic. The tree trunk and main structure are styled to resemble a bar chart, subtly tying in the financial tracking theme. Logo inspirations can be found in Appendix G.

Figure 4.9: *The logo for Evergreen Budgeting.*

4.4 Design Changes

4.4.1 Database Changes

During implementation, a number of changes were made to the initial database design as it became clearer what worked in practice and what did not. Some of these were smaller adjustments, for example:

- The limit attribute in the Budget entity had to be renamed, since "limit" is a reserved keyword in PostgreSQL.
- The user ID and category ID in the Budget entity were combined to create a composite key.
  - This ensures that a user cannot have duplicate budget categories in their budget.
- The data type for all money-related fields was changed to numeric(20,2), allowing for much larger values.
  - Originally, only nine digits had been allocated, but currencies where values are typically much higher, even for low purchasing power, had not been considered.

In addition to these minor tweaks, more structural changes were also made:

- A goal_name field was added to the Savings_Goal entity, allowing users to assign meaningful names to each of their goals.
- A ranking attribute was introduced in the Savings_Goal entity to allow users to prioritise their savings goals.
  - With this ranking system, a user's available savings are distributed in order of priority, starting with the top-ranked goal, and any surplus then trickling down to the next goals in the list.

Another key addition was the repeat_group_id field in the Transaction entity. This is used to link together transactions that are part of a recurring schedule. For instance, if a user logs a transaction that repeats on the 1st of every month from 01/01/2025 to 01/12/2025, the system will automatically generate 12 transactions with the same details but different dates. Each of these transactions is tagged with a shared repeat_group_id, making it easy to identify and manage all transactions within that group, such as when editing or deleting all future instances of that recurring transaction.

4.4.2 Navigation Bar Redesign

Some time was also spent during implementation redesigning the navigation bar. Initially, the original design was implemented, but it quickly became apparent that there would not be a profile or settings page at this stage, and that many key pages, such as the budget and transaction pages, were inaccessible through the nav bar.

Because of this, the design was updated so that the logo, home, transaction, budget, and report pages were visible and accessible from all pages, allowing for easier navigation. Since the income, expense, and savings pages shared a similar layout and functionality, they were grouped under a single "Transactions" dropdown menu.

The updated navigation bar can be seen in Appendix F.7.

The mobile version of the nav bar required fewer changes, as the necessary options were simply added to the hamburger menu. However, the "Evergreen Budgeting" text was replaced with the logo, and the home and reports icons were removed.

5 Implementation

5.1 Software Engineering Process

5.1.1 Version Control & Continuous Integration

Git and GitHub were selected for version control in order to more easily and efficiently manage changes to the codebase, and to maintain a record of the project's progress.

The project development followed a feature branch workflow. For each major milestone or new feature, a dedicated branch was created, separate from the master branch. This made it easier to work on individual features without introducing instability into the main application. Once a feature was completed, it was then merged back into the master branch.

Commits were made at the end of each coding session, once the code was functional and stable. This helped to ensure that the repository was updated regularly with small, incremental changes. It also provided a record of the development history, which made it easier to identify when and where certain changes were made. Each commit was linked to a relevant GitHub issue, helping to maintain documentation between tasks and code changes.

These version control practices were selected to improve code readability, reduce the risk of integration conflicts, and contribute to a more streamlined and maintainable development process (Chacon and Straub 2014).

5.1.2 Issue Management

Effective issue management was essential to maintaining focus and organisation throughout the project. Milestones were used to represent major features, while individual tasks and sub-features were tracked as issues within each milestone.

A Kanban board was implemented using GitHub Projects to track and coordinate these tasks (see Figure 5.1). The board was structured around four columns: Backlog, To Do, In Progress and Done. The Backlog served as a holding area for tasks not included in the current unit of work, while To Do listed items ready to be started. Issues were moved to Done either after the code's deployment to production or, for non-code-related tasks, following the addition of an explanatory comment.

Each issue contained key information to assist with planning and execution, including the associated milestone; a detailed summary of the task; a time estimate; and a priority label (see Appendix H.1). Priority was categorised into three levels: low, medium and high. These distinct, yet limited, options allowed for clear distinctions between urgent and lower-priority work. For time estimation, a t-shirt sizing system was used with three options: small, medium and large. This approach allowed for quick approximation of task complexity, particularly in cases where precise time predictions were difficult, and helped facilitate realistic workload distribution across the development timeline.

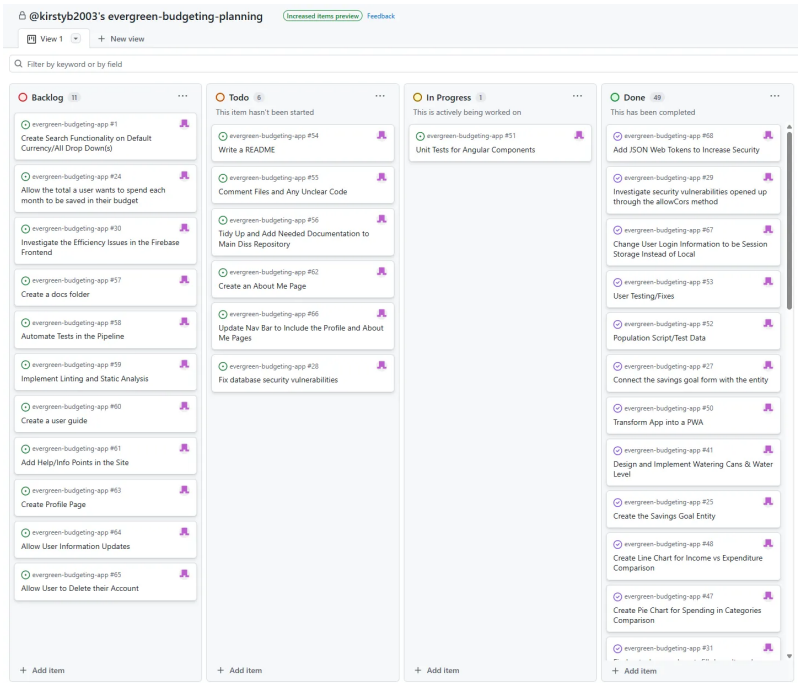


Figure 5.1: *Kanban board implemented with GitHub Projects*

5.1.3 Continuous Deployment

The application was continuously deployed using a CI/CD pipeline powered by GitHub Actions (n.d.). Whenever changes were pushed to the master branch, an automated workflow was triggered to deploy the latest version of the application.

The master branch served as the production branch, and no changes were merged into it until they had been tested and reviewed on their respective feature branches. Once a feature was complete, the branch was merged into master, triggering the deployment pipeline.

Deployment automation was configured using a deploy.yaml file, which outlined the necessary steps for building and deploying both the frontend and backend components. Authentication for deployment was managed using GitHub Secrets to ensure secure and automated access to both of the hosting platforms.

The benefits of continuous deployment strategies are discussed in software engineering literature. Humble and Farley (2010) highlight how automation leads to faster feedback loops, reduced deployment errors, and more reliable software delivery. Similarly, Fowler (2024) emphasises the importance of CI/CD in streamlining development workflows and enabling rapid, iterative releases.

5.2 Features

This section describes the features successfully implemented and how these related to the initial set of requirements. Further screenshots of the final product can be found in Appendix I.

5.2.1 Login & Registration

A key requirement of the MVP was that users should be able to log in securely to access their personal data. This functionality was fully implemented. All pages, other than the login and registration pages, are protected using an Angular AuthGuard, which ensures that only authenticated users can access routes that display sensitive data.

When a user logs in, a JSON Web Token (JWT) is generated by the backend and stored in session storage. This token is used to validate the user's identity on future requests, preventing unauthorised access to sensitive data. Passwords are securely handled using hashing with a salt before storage. During login, the entered password is hashed and compared server-side within the database query, ensuring plaintext passwords are never exposed. These methods are discussed further in Section 5.5.

Another key requirement was allowing users to select a default currency during the registration process. This value is collected via a dropdown input on the registration form, where users choose from a list of currency options represented by their corresponding ISO 4217 currency codes (e.g. GBP, EUR, USD).

5.2.2 Setting & Managing a Budget

When implementing the "creating a budget" functionality, a number of considerations were taken. First, the approach would involve the categorisation of spending, while also placing emphasis on savings. Second, the design aimed to allow the user to approach their budget setup from different perspectives. As a result, the budget form was implemented with two distinct methods: the user could either build their budget by allocating values to individual spending categories based on expectations or needs, or alternatively, start with a fixed overall budget amount and divide it across categories, with any remaining balance automatically allocated to savings. This flexibility supports both income-focused and spending-focused approaches to financial planning.

Form validation was handled using Angular's built-in features, with reactive warnings to alert users if they exceeded their total budget (see Figure 5.2). In the income-focused approach, expenses and savings are subtracted from the total budget, with any leftover funds automatically assigned to the "Miscellaneous" savings category. This approach supports a savings-oriented mindset, while still allowing users to adjust all category values, including those auto-calculated, supporting both structure and flexibility.

Displaying the user's budget progress also required careful consideration. From the existing app comparison, it was clear that users appreciate visual representations of spending progress. To cater to this, watering cans were used to visually represent spending across each category. Each watering can fills in proportion to the amount spent and overflows when the budget is exceeded (see Figure 5.3). These were implemented using SVG graphics, with the water level calculated dynamically based on the percentage of the budget spent. If the user has spent more than the budgeted amount, a conditional check within the component displays a blue ellipse below the watering can, representing a spill. Additionally, a red dotted line surrounds the SVG container to further highlight overspending and to fulfil the MVP requirement for a clear visual indication in categories where the user has exceeded their budget.

The user can also view a table overview of their budget on the home page, providing an alternative, more data-driven view (see Figure 5.4). This table contains additional details such as how much the user has left to spend in each category, as well as a totals row summarising the amount spent overall and how much money remains for the current month.

In this display, any rows representing categories that have been overspent in are coloured red and appear at the top of the grid, prioritising visibility of categories requiring attention. Additionally,

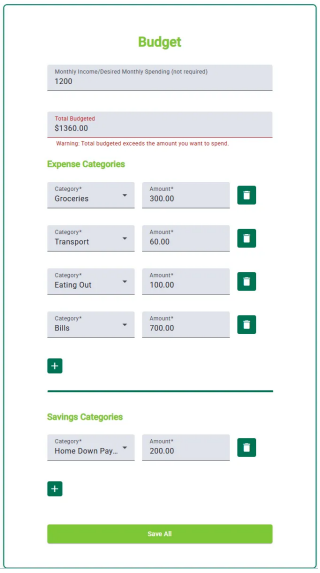


Figure 5.2: *The implemented budget form. Note the inputted desired spending amount and how this has led to a warning when the user's budget exceeds this amount.*

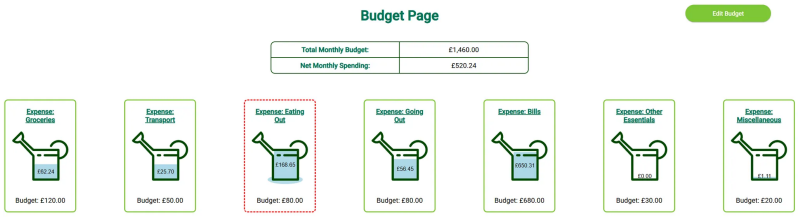


Figure 5.3: *The implemented budget page. Make note that each category in the users budget has their own watering cans and that categories that have been overspent in are illustrated clearly.*

if the user has exceeded their overall budget, the totals row also turns red, offering a clear and immediate indication that spending limits have been surpassed.

5.2.3 Setting & Managing Savings Goals

When designing the functionality for setting and monitoring savings goals, it was decided that contributions towards these goals would be based on a ranking system. Rather than assigning specific amounts to each goal manually, the total available savings are treated as a single pool of money that is distributed across the goals based on the user's prioritisation. To facilitate this ranking mechanism, a drag-and-drop feature was implemented using an Angular Material (2025) component, allowing users to easily reorder their goals. When the order is changed, the distribution logic recalculates and the amount saved towards each goal updates instantly.

Each goal visualises the amount saved using a progress bar (see Figure 5.5), providing users with a quick visual reference of their progress. Goals can be edited or deleted after creation, with delete actions triggering a confirmation modal implemented using Angular Material's dialog components.

A simplified overview of savings progress is also shown on the home page. This includes a progress bar representing how close the user is to reaching their combined savings targets.

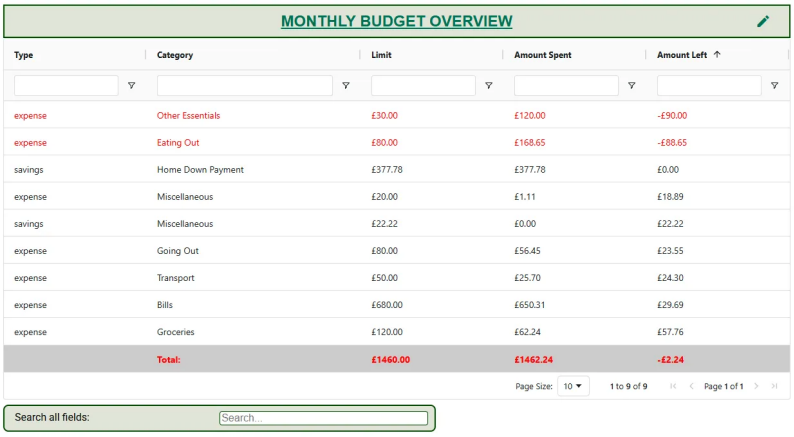


Figure 5.4: *The budget display from the home page. This is laid out like a data grid instead of the visual representation using the watering cans. Note that because the user has overspent on their entire budget, the totals row has turned red.*

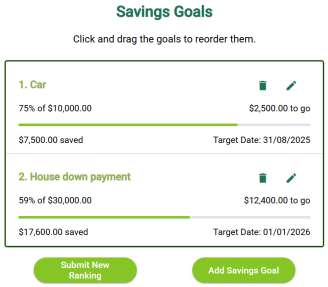


Figure 5.5: *The savings goal area on the Savings page. Each goal can be dragged to change its ranking and the progress bar indicates how far the user is in completing their goal.*

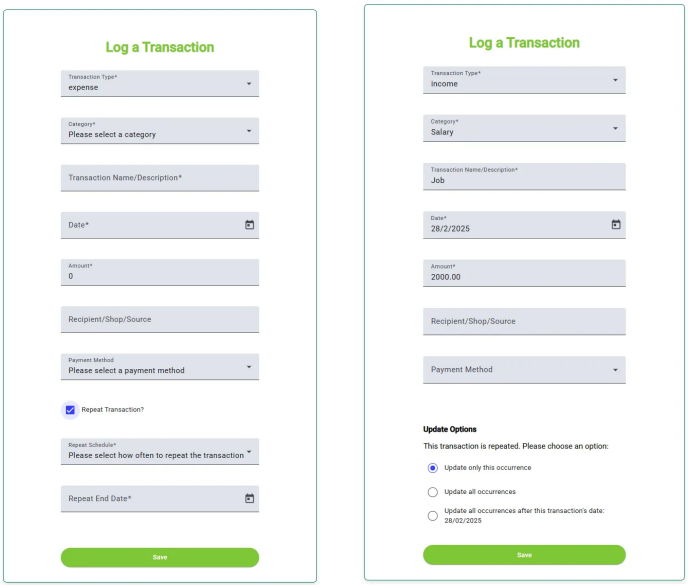
5.2.4 Logging Transactions

Users can log transactions with detailed information along with if the transaction is repeated (see Figure 5.6a). These transactions are displayed using the AG Grid (2025) library and are separated into two dynamically rendered tables: upcoming transactions and past transactions (see Figure 5.7). This helps users easily distinguish between historical and future activity.

Handling repeated transactions required server-side logic to expand the initial input into multiple database entries. The user selects a repeat interval and an end date, and the server calculates each individual payment occurrence by mapping time intervals between the start and end date. Each occurrence is then expanded into a full transaction object with the original metadata and is inserted into the database in a single batch query. A unique repeat_group_id is generated and assigned to all instances, enabling batch operations (e.g. mass deletion or edits) later.

Both tables are sorted by transaction date, with custom sorting logic ensuring that transactions nearest to today's date appear first. Since Angular defaults to the American date format of MM-DD-YYYY, a custom Moment.js data adapter was used to enforce the British format (DD-MM-YYYY).

AG Grid's built-in column filtering and sorting capabilities were extended through custom comparator functions to ensure accurate date ordering and search functionality. Date filters were



(a) *The transaction form with each of the possible inputs displayed.*

(b) *The display shown when editing a transaction that is repeated.*

Figure 5.6: *How the form is displayed for logging either a (a) new transaction or (b) editing an existing, repeated transaction. Note in (b) the update options at the bottom of the page that allow the user to customise how many transactions the changes are applied to.*

custom developed using a comparator-based IDateFilterParams implementation to allow date cells to be compared between rows. Although AG Grid supports individual column filtering, a custom search bar was implemented to enable full-table filtering via the quickFilterText property in the grid's API.

For editing and deleting transactions, a reusable TableActionComponent was created to display edit and delete symbols and to support row-level modification of data. When modifying or deleting an entry that is part of a repeated transaction group, users are presented with multiple options: apply the action to a single transaction, to all occurrences of the transaction, or only to future occurrences (see Figures 5.6b). These options are displayed in the log transaction form if the user attempted to edit the transaction and in a Material dialog popup component if attempting to delete one.

5.2.5 Reports & Charts

Reporting and financial visualisation were highlighted as a priority during the user survey, so it was essential that the charts implemented were not only accurate but also easy to interpret. Three types of charts were developed (see Figure 5.8): a pie chart showing a breakdown of categories; a line graph tracking changes in the user's net balance over time; and a bar chart displaying which time periods involved the highest spending. Each chart supports customisation by reporting period (weekly, monthly or yearly) while the pie chart can also be filtered to show income, expenses, or savings categories.

To handle situations where there might be no transactions in certain periods, the system uses Common Table Expressions (CTEs) with generate_series(). This creates a full list of dates (or weeks/months), even when no data exists for them. By doing this, charts always show complete time ranges without gaps, and zero values are filled in using COALESCE() to keep the data

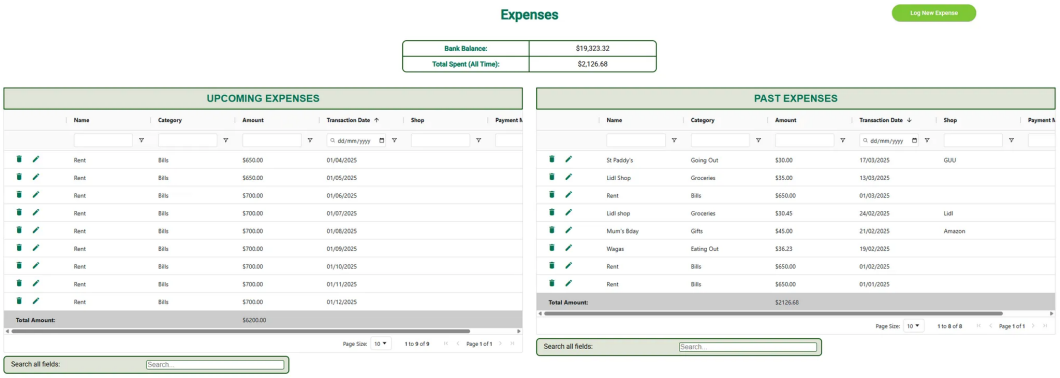
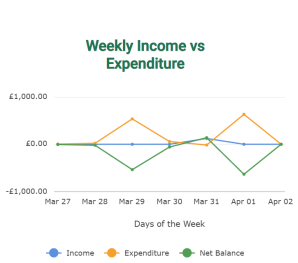
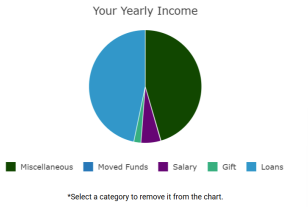


Figure 5.7: *The implemented expenses page. The savings and income pages look very similar. Note that transactions are split between two time periods and the custom search bar at the bottom of each table.*



(a) *A graph depicting the user's net balance over the last week.*

(b) *A bar chart illustrating how much has been spent each week for the last month.*



(c) *A pie chart that shows the breakdown of the users income in the last year by category.*

Figure 5.8: *The reports/charts found on the reports page. Each show the chart in a different reporting period. The pie chart can also show expenses and savings breakdowns.*

consistent.

Each query also calculates income, expenses, and net totals (income minus expenses) directly in the database. This reduces the amount of processing needed on the frontend and makes the system faster and easier to work with.

5.3 User Interface

5.3.1 Phone vs Desktop Views

To ensure the maximal display on both phones and desktops, alternative styling was developed for each page to accommodate different screen sizes. This was achieved through the use of responsive components such as flexboxes, CSS grids and flexible sizing units such as fr, rem and percentages. These methods, combined with media queries, which trigger different CSS styling when the screen size drops below a set limit, allowed components to adjust depending on the device being used. CSS grids were particularly useful, as they allowed for structured positioning and provided a more reliable layout than some of the other methods such as flexboxes.

Challenges arose when using external libraries like AG Grid to display data. Although AG Grid is designed to efficiently handle large datasets, its styling was difficult to override, resulting in poor display on smaller screens. Similar issues were encountered with other libraries, such as Angular

Material's drag-and-drop elements, which had fixed minimum sizes that exceeded mobile screen widths and were difficult or impossible to customise.

The data grids themselves also became harder to use on mobile, as only a few columns were visible at once. To improve readability, the past and upcoming transaction tables were stacked vertically on smaller screens. While rotating the phone to landscape mode alleviated some issues, relying on this is not ideal. Overall, the use of external libraries limited layout flexibility and mobile responsiveness.

Angular Material provided a consistent, modern UI across the app with reusable components like buttons, forms, popups, and drag-and-drop features (see Figure 5.5). Combined with Angular's component-based structure, this enabled reusable modules, ensuring visual consistency and reducing redundancy through dynamic, flexible components.

5.3.2 Scalable Data Management & Performance Optimisation

Handling large volumes of data was anticipated from the beginning of the project, particularly for users viewing their full transaction history.

AG Grid was chosen for its high-performance design, especially in handling large datasets. Beyond performance, AG Grid also helped reduce development time, as it came with many of the features required for displaying and interacting with transaction data, such as searching, sorting, and filtering, already built in. Although, AG Grid provided strong performance benefits and detailed documentation, the tables were hard to customise. Altering default styling or certain behaviours, such as filter interactions, was often challenging and lacked flexibility.

When it came to the data visualisation portion of the application, the AG Charts (2025) library was used. This library is designed to work efficiently with complex datasets and supports interactive, high-performance charts. However, similar to AG Grid, AG Charts also limited the level of customisation available, particularly in terms of styling. As a result, certain design requirements involved complex workarounds, making the customisation process more difficult than expected.

5.4 SQL Queries & Server-Database Interaction

An Express.js server was used to handle backend logic and database interactions, structured with a clear separation of concerns. All database logic was organised within an api folder, with a dedicated database-queries directory where each file, contained the SQL logic for a specific entity.

To keep the codebase modular and maintainable, the routing layer simply handled HTTP requests and passed data to these query files. This followed the Single Responsibility Principle, where each function focused on one task, such as inserting a transaction or updating a budget, making the code easier to test and debug.

Performance was prioritised through optimised SQL queries, indexing key columns like user_id and transaction_date, and using aggregation functions such as SUM, COUNT, and JOIN to offload calculations to the database. Queries were written to retrieve only the required data.

The backend followed RESTful API conventions, and a central query service was created on the frontend to manage API calls, keeping components cleaner and the codebase more maintainable.

Connection pooling was implemented using PostgreSQL's Pool to efficiently manage and reuse database connections. This, along with careful handling of concurrent requests, ensured good scalability and responsiveness under load.

5.5 Security

Given that the application handles user financial information, security was at the forefront of the design and development process. This section details the measures taken to protect the system against common attacks and security breaches.

5.5.1 Protection Against SQL Injection

SQL injection is a common attack in which a malicious actor manipulates SQL queries by injecting malicious code into form inputs or URL parameters. If not handled properly, this can allow attackers to access, modify, or delete database records, sometimes even bypassing user authentication altogether (W3 Schools n.d.).

To reduce this risk, all database communication in the system is handled using parameterised queries. These queries use placeholders (e.g. $1, $2) rather than embedding values from the user directly into the SQL query strings. Values are bound to these placeholders at execution time, preventing any injected SQL code from being interpreted as part of the query logic.

Additionally, routes on the server are protected by an authentication middleware which verifies the user's identity before the database query is even reached. This adds an additional layer of protection, as even valid SQL queries cannot be executed unless the user provides a valid access token.

5.5.2 Authentication & Session Management

User authentication is handled using JSON Web Tokens (JWTs), which are securely generated and signed when a user logs in. Upon successful login, the server creates a token using a secret key, embedding the user's ID in the payload. This token is then returned to the client and used to authenticate future requests. This system was implemented with the help of a GeeksForGeeks (2025) tutorial.

Each protected route on the server checks the incoming request for a valid token using an authentication middleware. The token is extracted from the Authorization header, and is verified using the same secret key that was used at login (see Appendix I.8 for the middleware code). If the token is missing or invalid, the server immediately rejects the request, returning an error response and then logs the user out.

These JWTs automatically expire after one hour. This helps to minimise the risk of long-term session hijacking, even in cases where the token is somehow illegitimately accessed. While storing the token in an HTTP-only cookie was attempted for added protection against client-side attacks, it was eventually stored in sessionStorage due to compatibility issues and to maintain simplicity in the Angular frontend. Although sessionStorage is accessible to JavaScript, it is automatically cleared when the browser tab is closed, making it a safer option than localStorage, which keeps data indefinitely.

5.5.3 Password Security

Password logic and security are handled entirely in the database. When a user creates an account or logs in, their password is hashed using pgcrypto's crypt function and a unique salt, generated using the Blowfish algorithm. This process was developed with the help of a Vultr (2023) tutorial on securely saving passwords using PostgreSQL. Only the resulting hash is stored in the database, and plain-text passwords are never saved.

Earlier versions of the system temporarily stored the hashed password in localStorage to simplify login handling. However, this approach was deemed insecure, as client-side storage of even

hashed passwords increases exposure to cross-site scripting (XSS) attacks (OWASP n.d.). This practice was removed in favour of token-based session management. At no point in the current implementation is the user's password or hash ever stored on the client or even transmitted from the database after the initial login.

5.5.4 Cross-Origin Resource Sharing (CORS)

CORS configuration plays an important role in determining which frontend domains are allowed to access the server (AWS n.d.).

The server checks the origin of each request against a list of trusted domains. Only requests from the deployed frontend or from localhost (when in development or test mode) are allowed. If a request comes from an unrecognised domain, it is immediately rejected with a 401 Unauthorised response. This ensures that external scripts or unauthorised web applications cannot access the server APIs even if they somehow acquire a valid token.

This CORS configuration is handled in a dedicated middleware file, which sets the appropriate response headers and performs origin checks before any further processing takes place. In addition to this, all production traffic is served over HTTPS, ensuring that sensitive data, including JWTs, is encrypted during transit.

5.5.5 Managing Environment Variables

Sensitive credentials such as database passwords, JWT secrets, and environment-specific configuration values are never hardcoded into the application source code. Instead, they are stored in environment variables, accessed through process.env.

In development, these values are defined in a local .env file, which is excluded from version control via .gitignore to prevent accidental disclosure. In production, environment variables are managed securely through the Vercel deployment dashboard. This ensures that critical secrets remain confidential and are not exposed on the client-side or version history.

5.6 Deployment

The initial deployment plan involved GitHub Pages, but it was quickly discovered that it only supports static web applications (GitHub n.d.). As the project included a Node.js server, this option was not viable.

Alternative hosting solutions were explored for the front-end, backend, and database. The first platform tested was Vercel (n.d.). Although the front-end rendered correctly, Vercel treated the project as a static site, breaking Angular's routing. After several troubleshooting attempts, a modular deployment strategy was adopted.

The project was reinitialised without server-side rendering to simplify deployment. Firebase (n.d.) was then chosen to host the front-end, offering an easy setup for static content. However, it was not suitable for hosting the full-stack architecture.

The final deployment approach used Firebase for the Angular front-end, Vercel for the Node.js server, and Supabase (n.d.) for the PostgreSQL database. This introduced the need for cross-service HTTP communication and raised CORS issues due to separate domains. These were resolved through configuration changes, as discussed in Section 5.5.4.

Although deploying across multiple services added complexity, it allowed the use of free hosting platforms, aligning with the project's accessibility goals. It also offered flexibility and scalability for potential future development.

6 Evaluation

6.1 Testing

Unit tests were conducted to verify the system's functionality at the component level, ensuring that each part of the application worked as expected in isolation. These tests are crucial for identifying issues early and maintaining stability throughout development, especially when implementing larger changes that could affect existing features.

Angular testing utilises the Jasmine and Karma frameworks. Jasmine provides a behaviour-driven development (BDD) framework for writing and structuring tests, while Karma acts as a test runner to execute them in real browsers.

All components underwent unit testing, achieving over 90% coverage across the frontend, exceeding the industry standard of 80%. A few components fell short of this threshold due to challenges with mocking external libraries, particularly Angular Material. Code coverage metrics are detailed in Appendix J.1.

Frontend services, responsible for tasks like authentication, data querying, and component communication, were also thoroughly tested to ensure reliable system performance and minimise integration issues.

The backend server was tested using the Jest testing framework. Each file was rigorously tested, including those managing CORS access, middleware authentication, error handling, query pooling, database routes, and database queries. Integration tests were implemented to ensure smooth communication between the frontend, middleware, and database. These tests achieved over 90% coverage, and the metrics are detailed in Appendix J.2.

Automated testing was integrated into the CI/CD pipeline using GitHub Actions. This setup ensures that unit and integration tests are automatically executed whenever code is committed or a merge request is made, maintaining code quality and stability. By automating these tests, any issues introduced by new changes can be promptly identified, reducing the risk of regressions.

Beyond these formal testing strategies, usability testing was conducted through manual interaction with the application from an end-user perspective. This helped verify that common tasks, such as creating budgets and logging expenses, were intuitive and functioned correctly in real-world scenarios. User testing was also performed as part of the system evaluation and is discussed further in the next section.

Additionally, the user evaluation was used to test the database storage capabilities, as it was unclear how much data the system could handle. By recruiting participants for the study, it was possible to observe database performance under load. Even with 20 participants, each inputting close to 100 records, the database never approached its capacity and continued to operate effectively without noticeable performance degradation.

6.2 Evaluation & User Testing

6.2.1 Aims

This evaluation assessed how well the application met the MVP requirements. Since these were based on end-user feedback from a user study and app reviews of existing budgeting applications, meeting them indicates alignment with user needs.

It also served as user testing, enabling direct observation of interactions with the system. In addition to assessing functionality, the study evaluated usability and overall experience. Through structured tasks and participant feedback, it identified strengths, usability issues, and areas for improvement.

6.2.2 Evaluation Design

A task-based evaluation was conducted to assess ease of use, feature effectiveness, and overall user perception.

The evaluation had two phases: a supervised phase, where participants' actions, challenges, and preferences were observed, followed by an unsupervised phase, where refined instructions were tested via a Microsoft Form. Participants completed a streamlined version of the same tasks independently, without real-time guidance during the unsupervised phase.

Participants provided demographic data on age, tech comfort, and budgeting habits before completing eight core tasks covering registration, budget creation, savings management, transaction logging, and financial reporting. Post-task questions gathered insights on usability, pain points, and overall experience. The full task list and questions are in Appendix L and links to the complete form results can be found in Appendix L.3.

The evaluation followed university ethical guidelines. Participants gave informed consent, could withdraw at any time, and no personally identifiable data was stored. The signed ethics checklist is in Appendix K.

6.2.3 Results

The evaluation included 20 participants, 11 supervised and nine unsupervised, with most aged 18–25. Across all participants, banking apps were the most common method for managing finances, while only two participants used budgeting apps, specifically Emma, Lumio, and Snoop. Most people also reported that they review their budget weekly. Charts can be found in Appendix M.

In the remote evaluation, five of nine participants used a computer, while three used their phones. Of these, two accessed the website, and only one downloaded the app.

Task 1 – Registration & Login

All participants successfully created an account and logged in. A bug was discovered when a participant selected Czech Koruna as their currency, causing NaN errors in calculations due to the system assuming currency symbols were a single character. This was promptly fixed before the next evaluation.

Users also highlighted a navigation issue, with seven of the 11 supervised participants expecting the Transactions heading in the navigation bar to be a link rather than a dropdown.

Task 2 – Creating a Monthly Budget

All participants successfully created budgets with the required amount by adding, editing, and deleting categories. 85% found the process intuitive, with many highlighting the real-time budget total as particularly helpful. Positive feedback included 65% finding it easy to use, 30% appreciating the wide range of categories, and 25% praising the clean UI and colour palette.

However, some issues were raised. The field for setting a desired budget total was unclear to users, and warnings for exceeding this amount lacked visibility. Additionally, premature form submission when pressing Enter and dropdown text being cut off on mobile were noted.

Tasks 3 & 7 – Setting & Managing Savings Goals

Participants found the UI intuitive, with the drag-and-drop feature for reprioritising goals being well received. However, several participants expected automatic saving of the reorder instead of manual confirmation. While 90% understood how goal order affected distribution, only 35% fully grasped where the redistributed money originated. Several participants requested the ability to allocate specific transactions to savings goals rather than relying solely on automatic fund distribution.

Some struggled to find the Savings page under the Transactions drop-down and suggested a separate menu item. Additionally, participants had difficulty finding the Add Savings Goal button as it was often positioned lower down and required scrolling to be seen.

Tasks 4 & 5 – Adding Income & Expenses

Most participants found adding both singular and recurring transactions easy. However, some attempted to enter expenses directly into the grid rather than using the Add Transaction button.

Filtering by date was a major usability issue, with all supervised participants struggling to locate or understand the filter button. Recurring transactions also presented challenges, with users finding the delete workflow unintuitive and requesting more flexibility in editing repeat logic after transaction creation.

Task 6 – PWA Mobile App

Among supervised participants, 80% attempted to use the PWA mobile app, but 50% switched back to desktop due to layout issues, difficulties with tables, and a preference for larger screens.

Task 8 – Viewing the Budget & Home Page

Preferences were mixed between the Budget and Home pages for viewing financial summaries. The Budget page was preferred for clarity, while the Home page provided more information but could feel overwhelming, particularly on mobile.

No participants found the Add Savings Goal button on the Home page, as the flag icon used for this action was unintuitive.

Task 9 – Viewing Reports & Charts

Most users successfully interacted with the reporting features, though some struggled with removing categories due to unclear selection functionality. Suggestions included a bar chart alternative for the pie chart and clearer differentiation of net income/expenditure trends.

6.2.4 Discussion

The evaluation demonstrated that the budgeting system is generally intuitive and user-friendly, with a clean design and strong visual appeal. However, several usability issues were identified,

particularly around navigation, mobile usability, and transaction management.

Navigation & Mobile Usability

Navigation was largely well received, but some users found certain elements unintuitive, particularly the placement of savings under Transactions and the flag icon for adding savings goals. The mobile version was useful for quick access but had layout issues, especially with tables and dropdowns, which impacted usability. Enhancements such as clearer icons, better spacing, and improved table responsiveness would improve the mobile experience.

Budgeting & Savings Features

While creating a budget was straightforward for most, the lack of visibility for overspending warnings and the unclear nature of the budget total field were recurring concerns. Addressing these by improving warning placement could enhance user understanding.

Savings goals were well received overall, but participants desired greater manual control over savings allocation. The ability to assign transactions directly to specific goals could increase user flexibility and engagement.

Transaction Management & Filtering

Filtering transactions was one of the most significant usability challenges, with multiple participants struggling to locate or use the filtering options. A more prominent filter button, clearer explanations, and a simplified rule-setting interface could address this issue.

Recurring transactions also required improvement. Many users instinctively attempted to delete recurring transactions by setting their value to zero rather than using the Delete button. Additionally, users wanted more flexibility in scheduling transactions (e.g. repeating on "the last Thursday of the month"). Enhancing the edit functionality and introducing a clearer deletion workflow would improve user experience.

Reporting & Data Visualisation

Participants appreciated the reporting features, but interaction with charts was sometimes unclear. Providing alternative chart types and improving interactivity, such as hover tooltips and checkboxes for filtering categories, would improve the experience.

Feature Requests

Many participants who did not already budget found the app helpful and indicated they would use it. However, spreadsheet users preferred their existing methods due to familiarity and flexibility. Bank integration was highlighted as a key factor that would encourage adoption, along with customisable categories and CSV transaction imports.

Despite some usability challenges, overall reception was positive, with 55% of participants stating they would use the application over their current method. Participants appreciated the app's visuals, intuitive design, and user-friendly approach. One participant mentioned that the app made them want to budget and asked when the final product would be available. Several others expressed interest in continued use.

6.2.5 MVP Requirements Evaluation

Requirement	Relevant Tasks	Met?	Comments
Secure login and data protection	1	✓	Login worked as expected with several security measures in place.
Set default currency on account setup	1	✓	Initially buggy with non-standard currency symbols but resolved.
Log spending and income with appropriate detail	4, 5, 7	✓	Process was intuitive, but some users expected direct grid entry for expenses.
Edit/delete transactions (including repeated transactions)	4, 5, 7	✓	Users could edit/delete, but recurring transaction deletion was unclear and led to confusion.
View reports/charts on finances	9	✓	Users found graphs helpful but suggested more interactivity and alternative chart types.
Set and track budgets	2, 8	✓	Budgeting was well-received, with both the grid and watering can formats praised for different reasons.
Set and track savings goals	3, 7	✓	Many appreciated the process, however, goal handling did receive some criticism, including how savings were assigned to each goal.
Persistent storage of user data	*	✓	User data remained after leaving, meeting expectation.
Indicate overspending in a clear way (category level)	2, 8	✓	Both the watering can approach and the table rows turning red to indicate overspending, were noted as positives of the system.
Indicate overall budget overspending clearly	2, 8	✗	No clear notification system for budget overspending; users were unsure of where this information would be displayed.
Check funds for upcoming expenses	4, 5, 8	✓	Users could check upcoming expenses, but filtering by date was confusing.
Mobile-friendly design with PWA capability	6	✗	Mobile layout had usability issues, particularly with tables and navigation.

6.2.6 Implications for App Improvements

The evaluation demonstrated that the budgeting system is intuitive and user-friendly, with strong visual appeal and useful features. However, improvements are needed in:

- Enhancing mobile usability, especially for tables and navigation.
- Improving feature visibility, such as the savings goals icon on the Home page and overspending warnings.
- Refining transaction workflows, including filtering and recurring transaction edits
- Providing more user control over savings allocation and enhancing reporting flexibility.

Despite these challenges, participants showed enthusiasm for the app. With targeted refinements, it has strong potential for adoption and user satisfaction.

7 Conclusion

7.1 Summary

Budgeting is an essential life skill that can significantly improve quality of life and mental well-being by giving individuals greater control over their money and spending. Technology is a powerful tool for financial management, leveraging data analysis and the ubiquity of mobile devices to enable detailed and informed tracking on the go.

Many applications already utilise these capabilities but often fall short in different ways. Some are overwhelmingly flexible (e.g. spreadsheets), while others require payment and abstract the budgeting process too far, resulting in rigid solutions. By comparing existing solutions, gathering feedback through personal usage, and analysing app store reviews, a concrete set of requirements for a new system was established. These requirements were further refined through an end-user survey, gathering insights into current budgeting practices and desired features.

The chosen architecture was a Progressive Web Application (PWA) comprising an Angular frontend, a Node.js server, and a PostgreSQL database. This approach enabled free web deployment while offering a mobile app experience, ensuring accessibility without financial barriers. The design prioritised a mobile-friendly, intuitive system with a clean UI that contained all necessary features without being overwhelming. Visualisation was a key aspect, with watering can icons designed as progress bars for each budget category.

Implementation followed sound software engineering practices, including version control with Git and GitHub, feature branching, a Kanban board for issue tracking, and continuous deployment to Firebase for the frontend, Vercel for the server, and Supabase for the database.

All planned features were implemented, with adjustments to savings goals and navigation to enhance usability. Due to the sensitive nature of financial data, security was prioritised through measures such as password hashing with salts, JSON Web Tokens for authentication, parameterised queries to guard against SQL injection, and CORS policies to prevent unauthorised server access.

Both monitored and unmonitored evaluations were conducted to assess the system's performance, identify areas for improvement, and carry out user testing. This led to resolving issues like multi-character currency code handling. Evaluations indicated that the system effectively met the requirements, with many users expressing interest in continued use. Participants praised the clean UI, clear visualisation of savings goals and budget progress, and the availability of financial reports. However, some found certain features confusing, such as filtering transaction tables by date, and the mobile version had usability issues like unintended touch interactions and layout problems on smaller screens. While users generally liked the system, some remained hesitant due to being comfortable with existing methods and the lack of bank integration.

In conclusion, the project successfully achieved its goal of creating a personal budgeting application that supports spending, income, and savings goal tracking, along with reporting and budget management. However, further improvements are needed to fully meet the diverse needs of a broader user base and ensure usability on mobile devices.

7.2 Future work

The following section discusses potential changes and enhancements that could be implemented in the app if there was more time in the project or if the project is developed further in the future.

7.2.1 Enhancing Mobile Usability

The current application does not work effectively on smaller touchscreen devices, which was a major requirement for the project. Improvements could include resizing elements to fit the screen, potentially involving switching libraries, or blocking column movement in tables to prevent accidental rearrangement and loss of columns.

Larger changes could also enhance the mobile experience. Since grids are difficult to view on smaller screens, the table view could be redesigned to display just the transaction name, date, and amount, with more details shown on click. This would present essential information at a glance while allowing users to view more as needed.

7.2.2 Expanding Reporting & Customisation Options

Currently, users can only view reports for the current week, month, or year. It would be beneficial to allow viewing of past periods, such as previous months or years, enabling better financial analysis and decision-making.

Reports could also be made more customisable. Rather than just breaking down spending by category, users could choose to see breakdowns by shop/payee or payment method. Additionally, more reporting periods could be offered, such as fortnightly, payday-to-payday, or custom. Users could also select their preferred chart type (e.g. bar graph instead of pie chart).

Preferences like chart types, reporting periods, and transaction table ordering could be saved to the user's account to provide a consistent experience when logging in.

7.2.3 Custom Categories & Multiple Bank Account Support

Additional features from the "should have" and "could have" sections of the MoSCoW prioritisation could be added to increase flexibility. Allowing users to create custom categories would enable them to tailor budgeting and tracking to their specific needs.

Implementing multiple bank account support could also help users track spending across various accounts (e.g. savings, current, ISAs). Interest rates could be linked to these accounts, allowing the system to automatically calculate interest payments.

7.2.4 Linking Savings to Specific Goals

Currently, savings are treated as one collective pool that trickles down through goals based on prioritisation. It would be beneficial to let users allocate savings to specific goals, creating custom savings categories that give more control over where transactions are directed. Generic savings could still follow the trickle method for those who prefer it.

If combined with the ability to track multiple bank accounts, users could tie specific accounts to particular savings goals or allocate a percentage of an account balance to a goal.

7.2.5 Optional Bank Integration

Some evaluation participants expressed interest in Open Banking integration to avoid manual transaction logging. However, the initial decision to exclude this feature was due to time constraints and feedback indicating that manual entry increases accountability and trust. Additionally,

many existing apps that require Open Banking are are unusable without this integration, which was a main complaint during the existing app comparison.

To meet the desire for bank integration without compromising flexibility, this feature could be made optional, allowing users to decide whether to connect their accounts. Enhanced security measures, such as two-factor authentication and robust data protection, would be necessary to handle the additional sensitive information.

7.3 Reflection

This project was a valuable learning experience in designing, implementing, and managing a full-stack application. Simultaneously developing frontend, server, and database capabilities required significant organisation and thorough checks to ensure that all components behaved as expected. The adoption of sound software engineering practices greatly supported this process. In particular, the Kanban board proved invaluable for tracking issues, keeping tasks on schedule, and maintaining clarity in the future workflow. Additionally, continuous deployment allowed new features to be tested in production environments, ensuring compatibility with production technologies and preventing unexpected breakages.

A substantial amount of time was dedicated to analysing requirements and designing the system, which proved beneficial once implementation began. However, some challenges could not be fully predicted or planned for, making it essential to remain flexible and make informed decisions throughout development. Despite the extensive planning, some key objectives lost focus during implementation, most notably, mobile usability. Since development primarily took place on a computer, much of the styling and features were optimised for desktop use, with insufficient testing on mobile devices. In hindsight, a more mobile-focused technology stack would have maintained emphasis on mobile usability, even at the cost of ease of deployment and accessibility.

Testing practices also emerged as a weakness during the project, as they were not consistently incorporated from the outset. Although continuous deployment was maintained, the absence of early testing meant that bugs occasionally reached production or were not identified promptly. Future projects would benefit from adopting a test-driven development approach by writing tests alongside the code and establishing an automated testing pipeline from the beginning. This would ensure that no code could be deployed to production without passing essential checks.

Overall, many of the project requirements were successfully met, and the evaluation received largely positive feedback from the 20 participants. Throughout the process, valuable technical skills were developed, including making database queries through HTTP calls, implementing secure page access with Angular routing, and applying security measures to protect sensitive data. Important soft skills were also cultivated, such as feature prioritisation as deadlines approached and maintaining consistent documentation of completed work.

Ultimately, the Evergreen Budgeting app achieved its primary goal of providing users with an effective tool for tracking and managing personal finances through a calming and aesthetic UI. The insights gained from this experience will significantly inform future development efforts.

A Detailed Existing App Comparison

A.1 GoodBudget

GoodBudget is a manual-entry budgeting app based on the envelope budgeting strategy. It is available in both free and paid versions, though the free version limits the number of envelopes users can create and the number of bank accounts they can track. There is also a web-based version of the app, which appears to be more complex and feature-rich than the mobile version. However, since this analysis focuses on mobile applications, the web version was not examined in detail.

Key features:

- "Envelopes" to categorise and track spending
- Debt-payment tracker
- In-app reports/charts to monitor budgeting progress
- Manual transaction entry including details such as payee, transaction amount, envelope category, and transaction date
- Ability to sync across devices (paid version only)
- Notifications to encourage progress or warn when overspending occurs

Strengths/positive feedback (Google Play Store reviews):

- Simplistic UI that is easy to use
- Informative and encouraging notifications
- No requirement to link to a bank account
- Manually adding transactions increases user accountability for spending
- Data syncs across devices (available in the paid version)

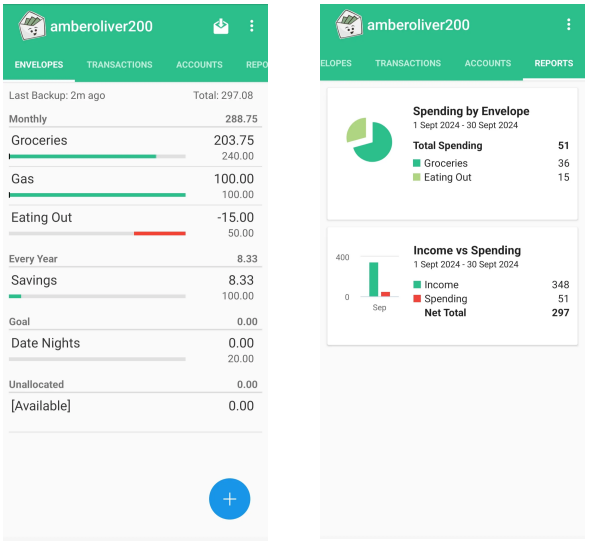
Limitations/negative feedback (Google Play Store reviews):

- Minimal reports/charts available and lack of customisability of these reports available
- Confusing process for editing envelope capacities on a one-time basis
- Inconsistencies between the mobile and web app experiences
- Many features are locked behind a paywall

Observations/personal experience:

- The envelope system was simple and easy to grasp.
- Progress bars for each envelope category were visually effective in communicating spending at a glance.
- The app does not require linking to a bank account, which may feel more secure for some users, especially when initially trying out the platform.
- The free version was rather limited.
- The lack of customisability in reports limited financial insights.

- Transactions were added to the bank balance even when the scheduled transaction date had not yet passed.
- Repeat payment schedules were limited and did not offer an end date, resulting in indefinite repetition.



(a) *GoodBudget's Envelope Budgeting Implementation.*

(b) *GoodBudget's Report Page.*

Figure A.1: *Screenshots of GoodBudget's UI. Progress bars can be seen for the amount spent of each envelope category in figure (a). Only 2 reports are available within the mobile app, as can be seen in (b).*

A.2 Spending Tracker

Spending Tracker is a manual-entry budgeting app that uses a blackboard-style UI to display income and expense categories and show how much has been spent in them. These values are then tallied up and displayed as the bank balance for that period. The app has no login, meaning the data is tied to the device. A one-time fee of £2.99 unlocks the paid version, which adds features like Dropbox sync, PDF export, priority support, and removes ads. While the budgeting concepts are simplistic, the UI can be confusing and limited.

Key features:

- Blackboard home page showing income and expense tallies for the selected period
- Manual entry of transactions, grouped into custom categories
- Multiple bank accounts supported, including transfers between accounts
- Transactions can be scheduled in advance and set to repeat (daily, weekly, or monthly)
- Basic reports and interactive charts (weekly, monthly, yearly)
- Budgeting goals showing leftover amounts carried into the next period
- CSV export (free) and PDF export with the paid version
- Dropbox backup and sync (paid feature)

Strengths/positive feedback (Google Play Store reviews):

- Simple to use, even without budgeting knowledge

- No limits on number of bank accounts or custom categories
- Scheduled transactions only affect the selected period, useful for "what if" planning
- One-off payment for Pro version is low-cost
- Does not require linking to a bank account
- Android users can back up data for free

Limitations/negative feedback (Google Play Store reviews):

- Best features are restricted to the paid version
- No login, data is device-specific
- Reports are limited to category breakdowns and cash flow only
- No way to show how a transaction was paid (e.g. card, transfer, etc)
- No income reports equivalent to the expenses charts
- Transactions sorted only by day with no way to change this
- No ability to assign budgets per category
- No payday-to-payday viewing option
- Dropbox is the only backup method, requiring a separate Dropbox account

Observations/personal experience:

- Switching between periods or adjusting the view date is not intuitive
- Reports page requires rotating the device, which is frustrating
- Budget mode must be manually enabled and is not clearly signposted
  - The Budget mode only allows for a budget total to be entered so only applies to the overall balance, not individual categories
- Ads are frequent and sometimes require watching 30-second videos
- While the app covers basic budgeting, it is almost too simplistic, offering little more than what a banking app already provides
- Useful for quick tracking, but lacks depth for long-term planning or goal-setting

A.3 Monefy

Monefy is a manual-entry budgeting app that displays spending in a clear, category-based pie chart format. It allows quick tracking of where the user's money is going by selecting a category and entering the amount spent. Transactions can be added either by choosing "Income", "Expense", or selecting a specific category directly.

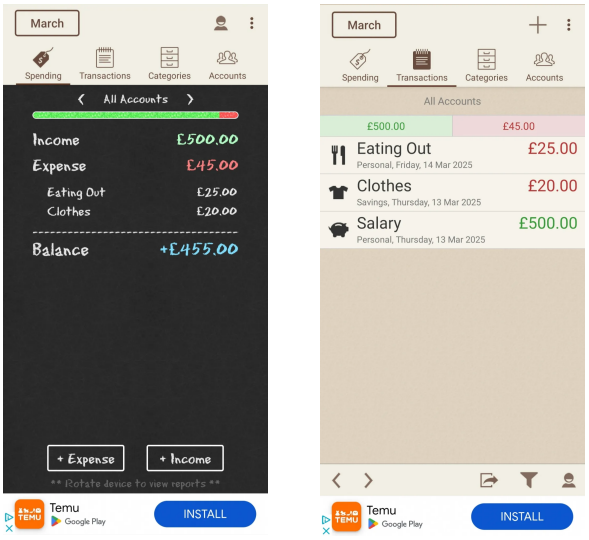
The free version is limited, with key features locked behind a paywall. A subscription of £26.49 per year is required to unlock the full set of features, including repeated expenses, custom spending categories, login/sync functionality, and multi-device access.

Key features:

- Pie chart visualisation of expenses by category
- Add transactions by selecting "Income", "Expense", or a category directly
- View transactions either in pie chart form or as a list
- When adding a transaction: enter amount, note, category, and whether it is an income or expense

Strengths/positive points:

- Simple, visual layout makes it easy to understand where money is going



(a) Spending Tracker's Blackboard UI.

(b) Spending Tracker's Transactions Page.

Figure A.2: *Screenshots of Spending Tracker's UI. Breakdown of spending in each category and income can be seen in (a). The amount spent and the category of each transaction can be seen in (b). Note that it is not possible to assign transactions names in this app.*

- Quick to enter transactions with minimal steps
- Clean and intuitive interface

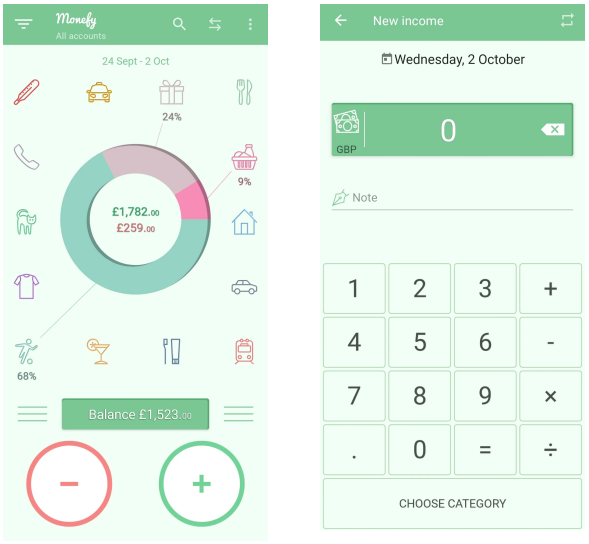
Limitations/negative points:

- Most key features are locked behind a paywall (£26.49 per year)
- Cannot add recurring expenses without paying
- Custom categories are not available in the free version
- No login or sync options unless you pay
- Pie chart only displays expenses, not income

A.4 Budgeting Apps with Bank Account Integration

Any app reviews mentioned can be found in Appendix B.

Snoop was the first open banking budgeting app looked at. It integrates directly with users' bank accounts to provide personalised insights and savings recommendations. By analysing spending data, Snoop offers tailored advice on how to save money, set financial goals, and improve overall financial health. It was found that while the app was easy to use, with a clean UI and interesting financial insights, it often categorised transactions incorrectly. This resulted in inaccurate insights and charts, meaning manual adjustments were needed for many transactions, even though one of the main points of linking a bank account is to reduce the need for manual input. A paid version is available, but it was found that most of the features interested in were accessible through the free version. Snoop is rated highly on the Google Play Store, with an average rating of 4.5. Many users appreciate that all key information is available in one place and that the interface is clear and easy to read. However, others also mentioned the inaccurate categorisation of transactions and the limited options for customisation of analytics and transaction information.



(a) Monefy's Pie Chart Spending Breakdown.

(b) Monefy's Add a Transaction Page.

Figure A.3: *Screenshots of Monefy's UI. Breakdown of spending in each category is displayed as a pie chart on the home page as seen in (a). To enter transactions, the user must use a built in calculator and can add very limited detail to the transaction, as can be seen in (b).*

PocketGuard was the next app explored. It does not offer a free version, so the amount of the app available to be assessed was limited. From what could be seen, the app provides a simple and clear overview of spending and automatically categorises transactions through a bank account connection. Users can customise spending limits, receive a personalised debt payoff plan, and access resources to help manage their personal finances. However, with an average rating of 3.4 on the app store, reviews suggest that the bank connection is often unreliable, with users frequently having to reconnect their accounts. Others also mentioned slow syncing between their bank and the app, along with various glitches such as missing information and deleted transactions.

YNAB (You Need a Budget) uses a zero-based budgeting methodology, where users assign every dollar or pound to a specific expense at the start of the month. The app connects to bank accounts and syncs across multiple devices. Although there is no free version, a YNAB subscription can be shared across multiple users, which makes it more cost-effective for couples, families, or housemates. In addition to core budgeting features, YNAB provides loan calculators, spending and net worth reports, and a range of financial help guides. The app is rated very highly on the Google Play Store, with an average rating of 4.9. Despite this, there are recurring criticisms, including the high subscription cost, the inability to rename transactions (which default to the payment processor's name), and the steep learning curve, as the zero-based budgeting method is not always intuitive to new users.

Rocket Money is a budgeting app with a specific focus on managing subscriptions and recurring payments. It aggregates bank account data to provide insights into spending trends and offers tools to help users cancel unwanted subscriptions and negotiate bills. Rocket Money even provides negotiators who will contact service providers on the user's behalf to try to reduce bills, and it also offers its own credit card with a focus on building mortgage savings. A free version of the app is available, along with a premium version that unlocks additional features such as subscription cancellation services, account sharing, and full credit reports. However, Rocket Money is not currently available in the UK or for users with UK-based bank accounts.

B App Store Reviews of Existing Budgeting Apps

B.1 GoodBudget Reviews

Positive Reviews:

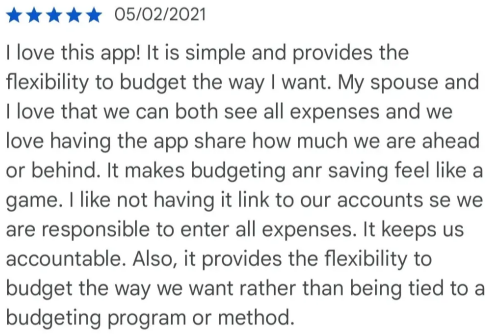


Figure B.1: *Review that mentions how they like manually entering transactions as it holds them accountable for their spending.*

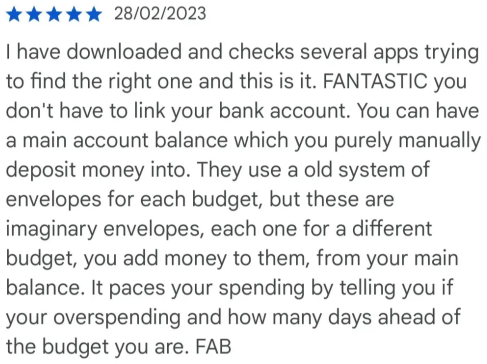


Figure B.2: *Review that mentions how they like that the app does not require them to link their bank account and that they like the envelope system.*

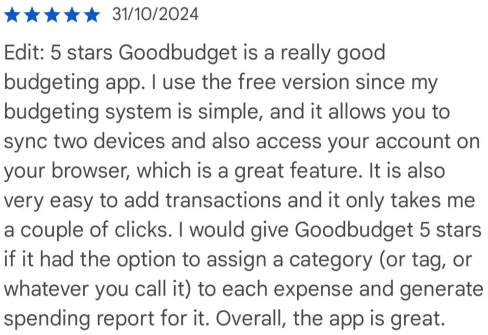


Figure B.3: *Review that mentions how they like that they can sync across devices but does not like that reporting is so rigid.*

Negative Reviews:

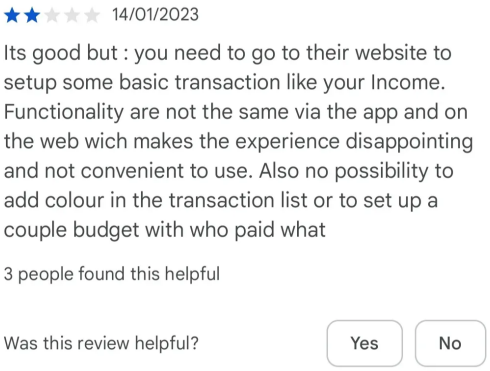


Figure B.4: *Review that mentions the lack of consistency between the mobile and web app.*

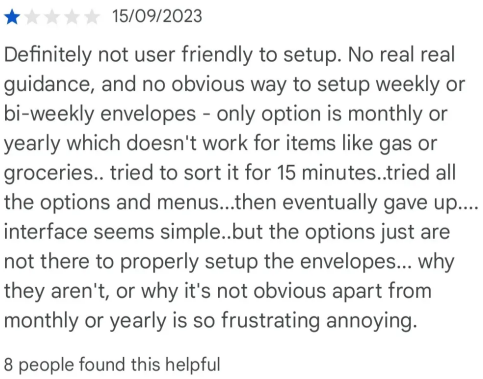


Figure B.5: *Review that mentions the lack of repeat periods for repeated transactions.*

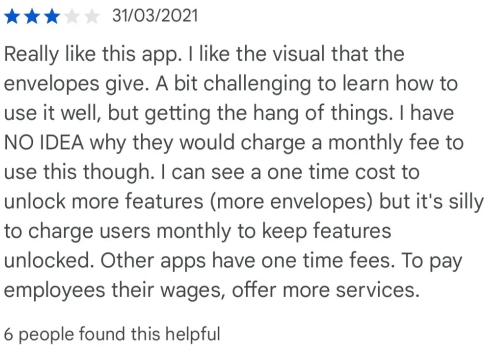


Figure B.6: *Review that complains about the price but they like the envelope approach.*

B.2 Spending Tracker Reviews

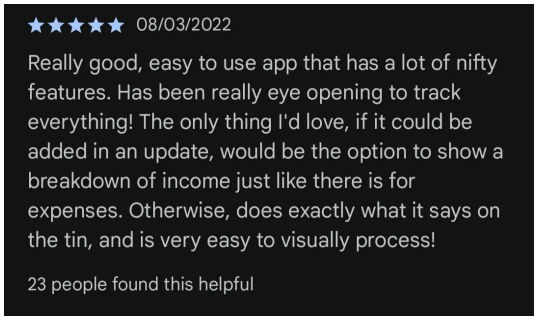


Figure B.7: *Review talking about how the app does not have income reports and breakdowns.*

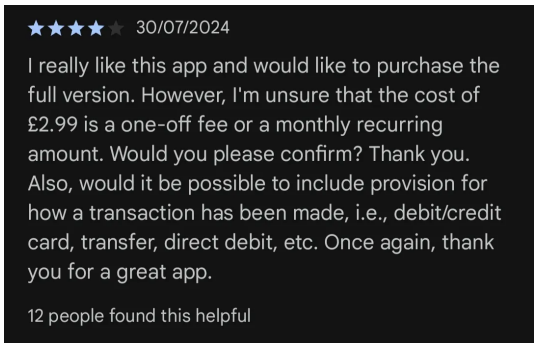


Figure B.8: *Review talking about how the user would like to be able to add a payment type detail to transactions.*

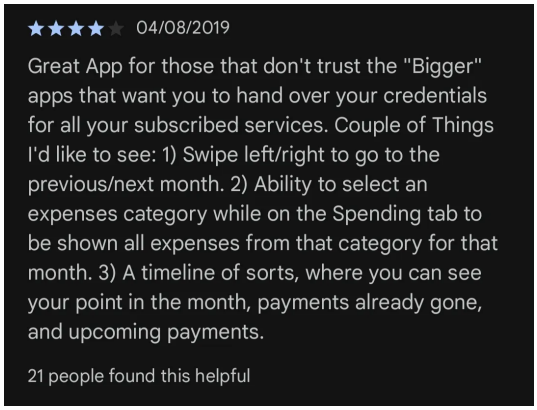


Figure B.9: *Review that mentions the user likes that they do not have to share their personal details or connect their bank account in order to use the app.*

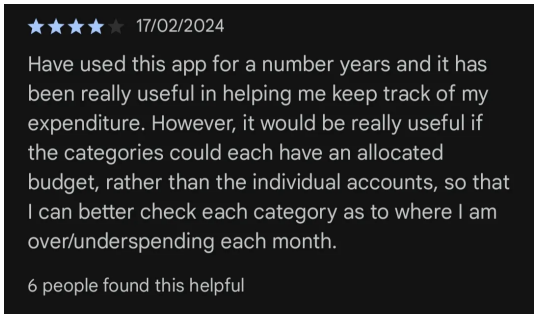


Figure B.10: *Review that mentions the inability to set budget limits for specific categories.*

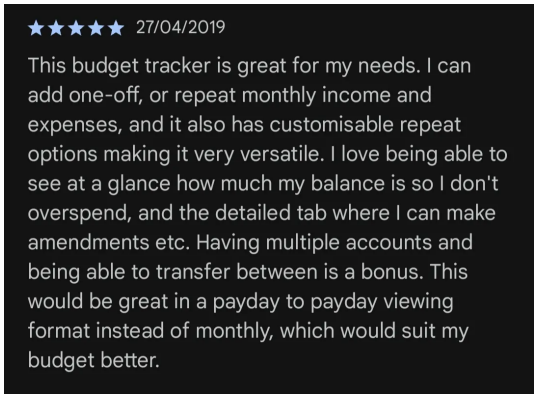


Figure B.11: *Review that talks about how the user liked being able to track multiple bank accounts but that they would like to have a payday-to-payday financial period. Also mentions how they like how customisable the repeat periods are.*

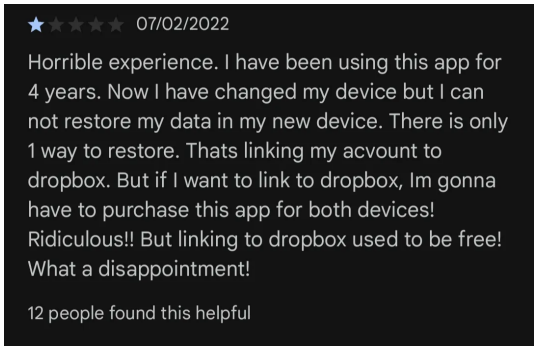


Figure B.12: *Review that complains about how you must use Dropbox in order to save data. Also mentions how if you pay for the pro version of the app, you need to pay for it on every device you use the app due to the lack of login.*

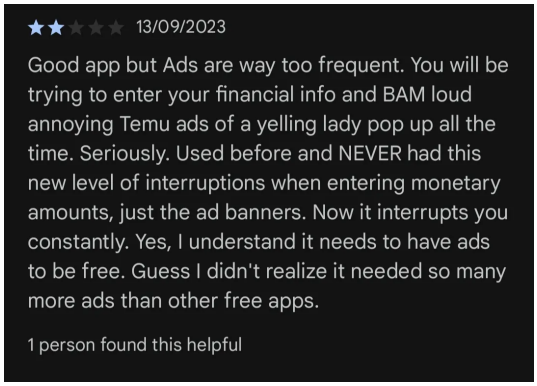


Figure B.13: *Review that complains about the amount and frequency of ads in the app.*

B.3 Snoop Reviews

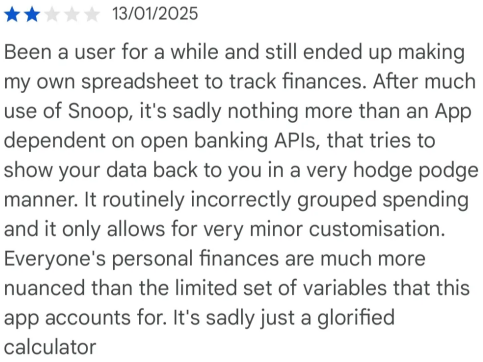


Figure B.14: *Review discussing how transactions are often miscategorised and how the app allows very little customisation.*

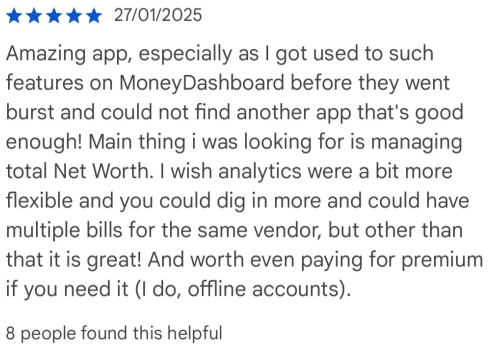


Figure B.15: *Review talking about how the analytics and reports are rigid.*

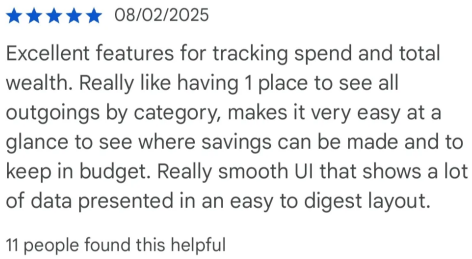


Figure B.16: *Review discussing how having all information in one place is really helpful and that the UI is clear and easy to use.*

B.4 PocketGuard Reviews

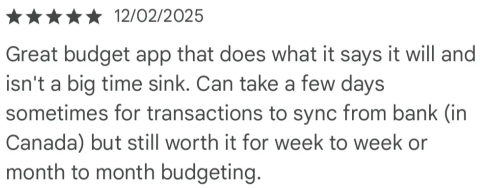


Figure B.17: *Review complaining how the app's link with bank accounts often disconnects. Also discusses loss of transactions and data when cards are cancelled.*

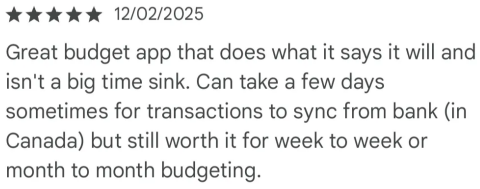


Figure B.18: *Review talking about how syncing data from the bank account can be slow.*

B.5 YNAB Reviews

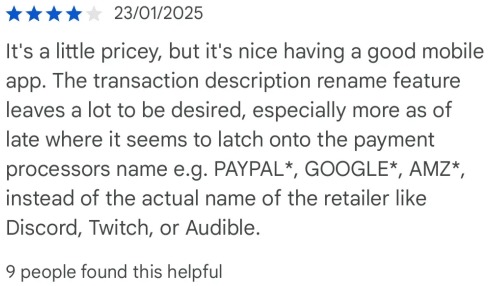


Figure B.19: *Review talking about how the app is expensive and how you cannot change the name of transactions.*

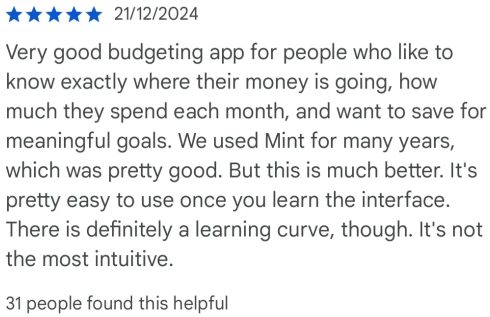


Figure B.20: *Review noting that the app has a rather steep learning curve.*

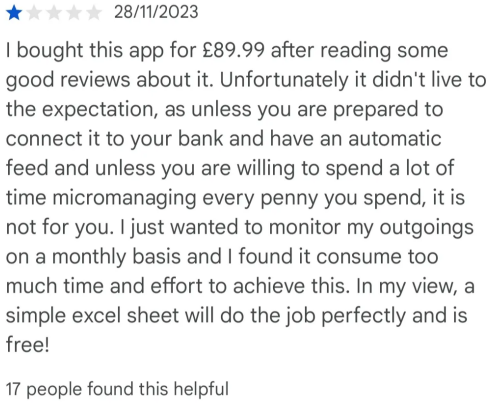


Figure B.21: *Review complaining that the app is too expensive and that the zero-budgeting approach was frustrating for this user.*

C User Research Questionnaire & Ethics Scripts

Budgeting with Technology

The aim of this experiment is to explore how people manage household budgeting and the role technology plays in this process. Understanding people's habits and preferences in using tools for budgeting will help me design a more effective budgeting application. This is why your participation is essential, as it allows me to collect diverse data on how individuals manage their finances and the use of technology to do so.

During this questionnaire, I will ask you several questions about your general budgeting process and any tools you use. These questions will not require you to share any private or confidential information, instead they focus on understanding your general approach to managing finances. **All responses are anonymous.**

Please feel free to ask any questions by emailing me at 2635375b@student.gla.ac.uk.

Please remember it is your general budgeting habits that are being evaluated, not you personally. Even if you do not currently use a budgeting system, that information is still valuable for me.

You are welcome to withdraw from the experiment at any time. If you do so, then it will not be possible for you to be debriefed about the purposes of the survey.

Required

1. Do you agree to participate in this questionnaire and consent to your answers being used in this project? *

- ◯ Yes
- ◯ No

Demographic Questions

2. What age range are you in?

- ◯ 18-25
- ◯ 26-35
- ◯ 36-50
- ◯ 50-60
- ◯ 60+

3. How comfortable are you using technology?

- ◯ Very comfortable
- ◯ Somewhat comfortable
- ◯ Somewhat uncomfortable
- ◯ Very uncomfortable

4. What type of device do you use when managing your finances?

- ☐ Phone
- ☐ Laptop
- ☐ Desktop
- ☐ Tablet
- ☐ Other

5. Do you currently do any form of budgeting? *

- ◯ Yes
- ◯ No

Budgeting Habits

6. How often do you review your budget or financial situation? *

- ◯ Daily
- ◯ Weekly
- ◯ Monthly
- ◯ When Making a Purchase
- ◯ Yearly
- ◯ Never

7. What methods do you currently use to manage your budget? *

- ◯ Pen and paper
- ◯ Spreadsheet
- ◯ Budgeting app
- ◯ Other

8. How competent do you feel in managing and understanding your finances? *

Not competent	Somewhat competent	Competent
◯	◯	◯

9. How satisfied are you with the budgeting tools you use? *

Very unsatisfied	Unsatisfied	Satisfied
◯	◯	◯

10. What are your main goals when budgeting? E.g. saving to buy a specific item, paying bills, tracking spending, etc.

11. What features do you find most useful in budgeting tools? E.g. expenses/income tracking; spending reports and charts; savings goals; bill alerts; etc.

12. Do you use a budget specific app? E.g. YNAB, Good Budget, Monefy, Rocket Money, etc. *

- ◯ Yes
- ◯ No

13. What prevents you from using a budget app? E.g. E.g. privacy concerns, difficulty of use, prefer manual methods, etc.

14. Would you be willing to use a budget app in the future?

- ◯ Yes
- ◯ No
- ◯ Maybe

15. If you answered no to question 14, why is this?

Barriers & Motivators for Budgeting

16. What stops you from budgeting? *

- ◯ I don't see the need for it.
- ◯ I find it difficult to understand.
- ◯ It feels overwhelming.
- ◯ It's too much effort.
- ◯ Other

17. What could get you to start budgeting? *

- ◯ Gaining more knowledge about budgeting.
- ◯ Having access to a tool.
- ◯ A change in my financial situation.
- ◯ Other

Budget App Features

18. Are there any particular features you'd like to see in a budgeting app?

19. Rank the following features in the order that are most important to you when budgeting (the top being most important):

- Savings goal setting
- Reports and charts on spending and income
- Envelope budgeting (i.e. set spending limits for each category of purchase)
- Tracking multiple bank accounts

20. Do you have any further suggestions for improving the experience of a budgeting app?

Debrief

The main aim of this experiment was to explore how people manage household budgeting and the role technology plays in that process.

In particular, I was interested in learning which features of budgeting technology you would find most useful, as well as what might encourage you to use a budgeting app if you do not already. I also wanted to understand the factors that may currently prevent you from budgeting.

If you have any questions or comments about the experiment, please feel free to reach out to me. You can contact me at 2635375b@student.gla.ac.uk. Thank you for your participation!

D User Research Questionnaire Results

D.1 Link To Microsoft Form Results

Microsoft Form summary results can be found here.

D.2 Detailed Results Summary

The survey received a total of 51 responses. The results showed that 88% of respondents were aged between 18 and 25, and only 6 respondents were over the age of 36 (see Figure D.1). Confidence with technology was very high as can be seen in Figure D.2a. Interestingly, there was a strong correlation between comfort with technology and the use of budgeting apps. Any respondents who currently use a budgeting app stated that they were very comfortable using technology and the only participants to say that were completely unwilling to even try using a budgeting app were over 36 years old.

The majority of respondents use their mobile phone to manage their finances, as can be seen in Figure D.2b, with the next most common device used being a laptop. These findings highlight the importance of designing a budgeting app that is mobile-friendly.

A slight majority of 55% of the respondents budget in some form, with 45% stating that they carry out no budgeting of any kind. Among those who do budget, most review their finances either monthly (43%) or weekly (36%), as seen in Figure D.4a. This shows that it is important that the budgeting app allows for these reporting periods.

The most common tools used to budget were reported to be spreadsheets (32%) and pen and paper (29%), as can be seen in Figure D.3. Other less frequent methods included banking apps, budgeting apps, notes apps, and mental calculations.

Respondents satisfaction with their budgeting methods can be seen in Figure D.4b. The majority reported that they were at least somewhat satisfied with their current methods, but a notable 25% were unsatisfied, presenting a clear opportunity to design a tool that addresses common shortcomings and improves user experience.

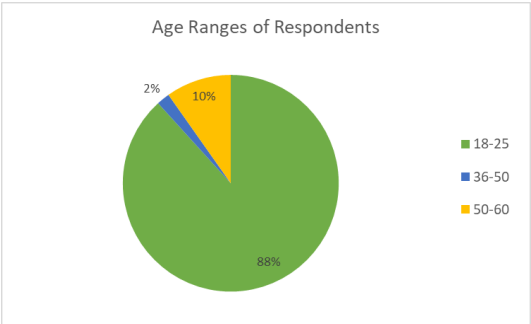
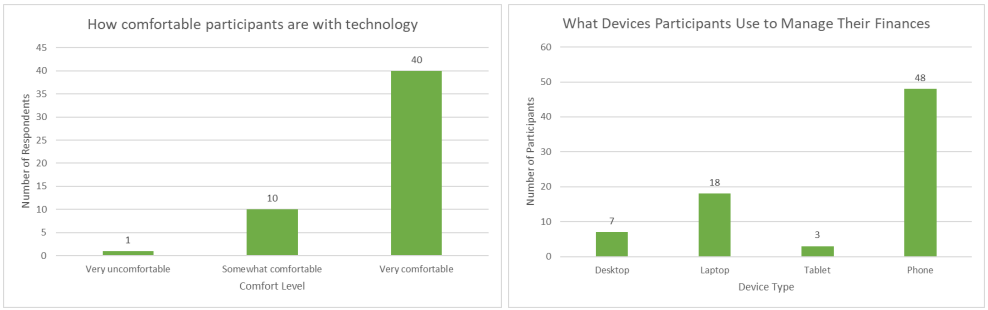


Figure D.1: *Chart showing the distribution of ages of respondents.*



***(a)** How comfortable respondents reported feeling with using technology.* ***(b)** The types of devices respondents use to manage their finances.*

***Figure D.2:** Column charts illustrating respondents comfort with technology and the devices they use to manage their finances.*

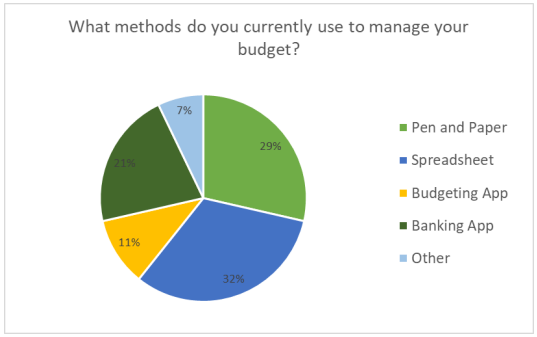


***Figure D.3:** Chart showing the different tools and methods respondents use to budget.*

Respondents who budget indicated a number of motivations, with the most common being:

- Tracking spending/visualising where money goes (56%)
- Saving, either long-term or for specific goals (52%)
- Ensuring bills are paid (29.6%)

Other goals included reducing unnecessary purchases, building an emergency fund, and having enough disposable income for hobbies/treat items.

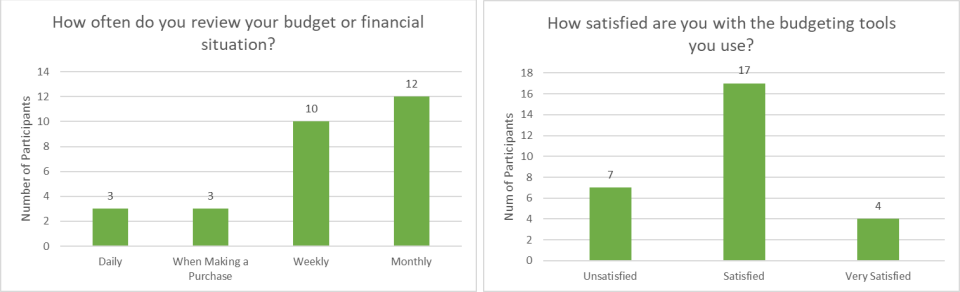
Despite the prevalence of budgeting habits, only 11% of those who budget currently use a budgeting app. The most common reasons given for not using one were:

- Lack of awareness or felt there was no need for one (43.5%)
- Preference for existing methods such as spreadsheets or banking apps (35%)
- The complexity or stress associated with budgeting apps (17.4%)
- Privacy concerns and lack of personalisation were also mentioned by some participants.

A large portion of respondents indicated openness to trying a budgeting app in the future: 40% said they would, and 52% said they might. Only 8% firmly rejected the idea.

Among those who do not currently budget, the most common barriers stated were:

- There is too much effort involved in using a budgeting app (35%)
- A lack of necessity based on personal financial circumstances (26%)
- Feeling overwhelmed by the process (22%)



(a) *How frequently respondents who budget review their finances.*

(b) *Respondents' satisfaction with their current budgeting methods.*

Figure D.4: *Column charts illustrating how often respondents review their finances and how satisfied they are with their current budgeting methods.*

In terms of motivators, the majority of non-budgeters indicated they would consider budgeting if:

- They had access to a helpful tool (45%)
- Their financial situation changed (36%)
- They gained more knowledge or confidence in budgeting (14%)

Participants provided detailed feedback on the features they would find most valuable in a budgeting application. These included:

- Linked banking integration (15%)
- Categorising expenses into different money pots (12%)
- Reports and charts on spending (12%), including predictive models showing how finances may evolve based on decisions
- Notifications and alerts, such as:
  - When limits are reached
  - Reminders about upcoming bills or payments
  - Weekly spending summaries and progress updates
- Budgeting tips and educational features, including personalised insights and advice on how to save more effectively.

Participants also expressed a strong preference for simplicity, clarity, and for the app to be free to use.

When asked to rank a number of features by what the respondents felt was most important to them in a budgeting app, the average ranking was:

1. Reports and charts on spending and income
2. Savings goal setting
3. Envelope budgeting (i.e., assigning spending limits by category)
4. Tracking multiple bank accounts

Additional suggestions included:

- Ability to visualise income vs. expenses dynamically throughout the month
- Exportable data and reports
- Insights into where money is spent (e.g., by shop or category)
- Educational resources or budgeting advice embedded within the app

E Architecture Diagrams

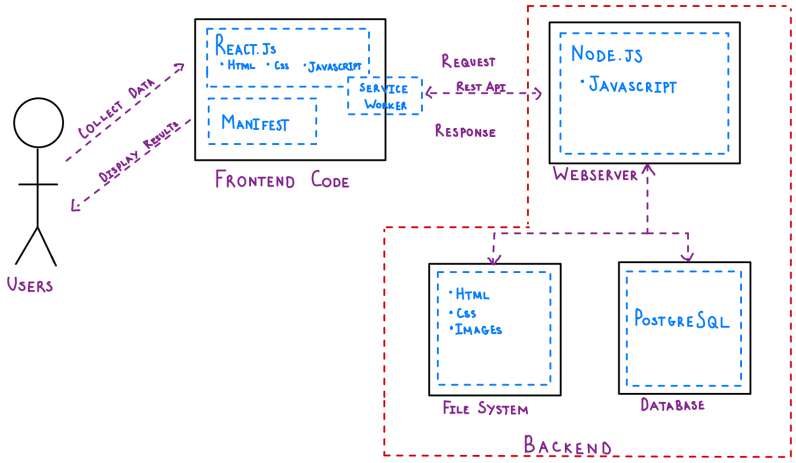


Figure E.1: *Initial architecture design using a React front-end.*

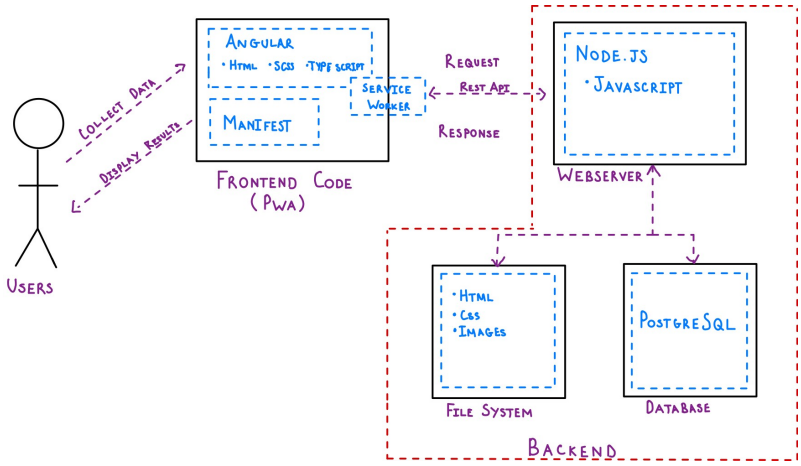


Figure E.2: *Final architecture design using an Angular front-end with the PWA labelled.*

Service Worker *- manages offline access, push notifications and caching of the PWA.*
Manifest *- enables installable app on mobile and desktop.*

F Wireframes

F.1 Login Page

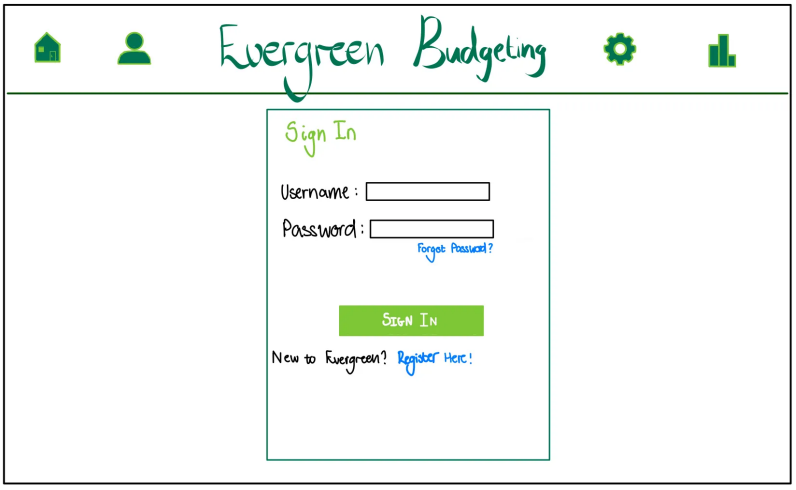


Figure F.1: *Desktop wireframe for the login page.*

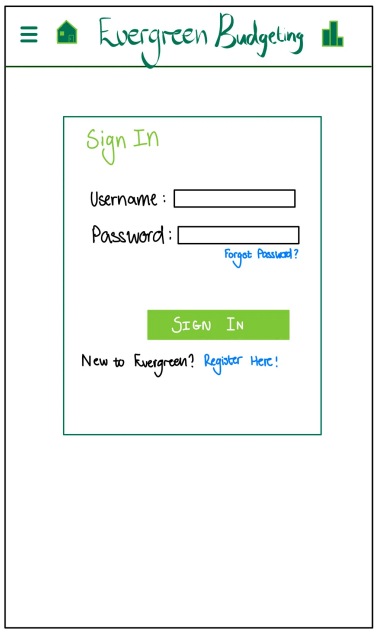


Figure F.2: *Phone wireframe for the login page.*

F.2 Home Page

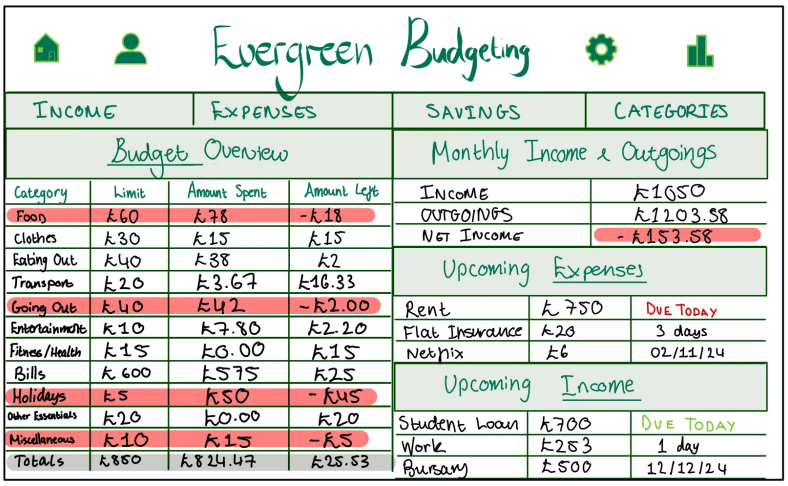


Figure F.3: *Desktop wireframe for the home page.*

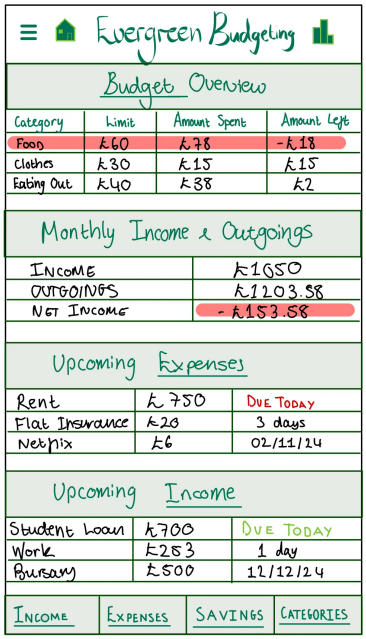


Figure F.4: *Phone wireframe for the home page.*

F.3 Income Page

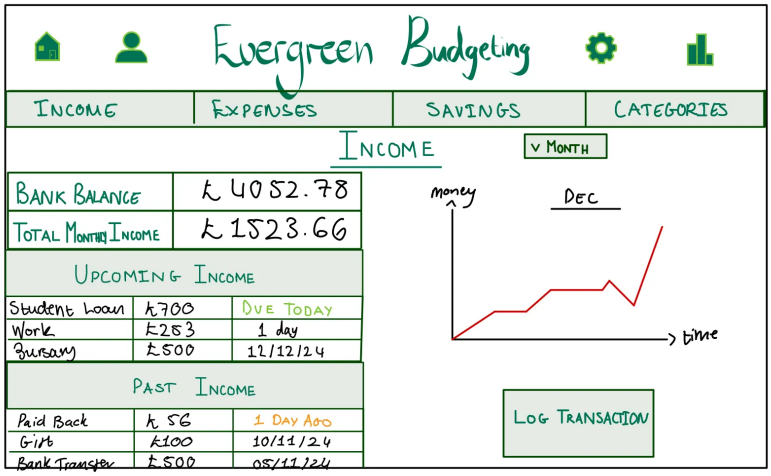


Figure F.5: *Desktop wireframe for the income page.*

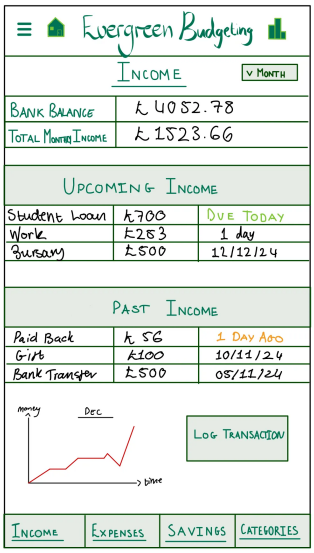


Figure F.6: *Phone wireframe for the income page.*

Note: The UIs for the expenses and savings pages are very similar.

F.4 Log a Transaction Page

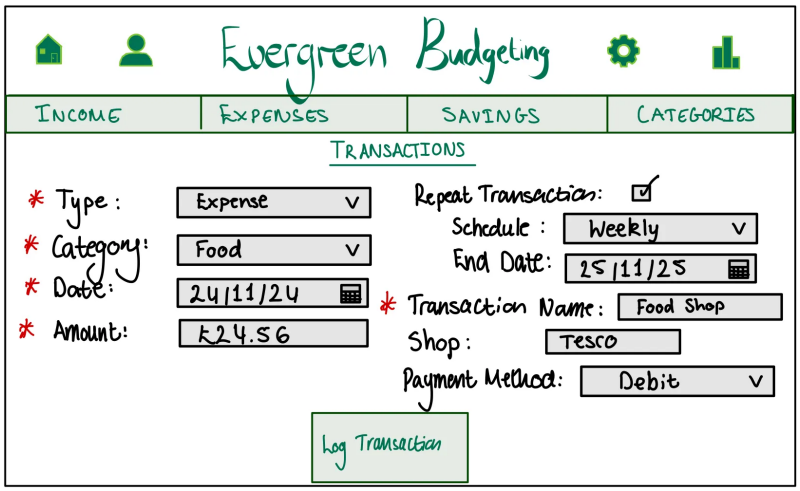


Figure F.7: *Desktop wireframe for the log a transaction page.*

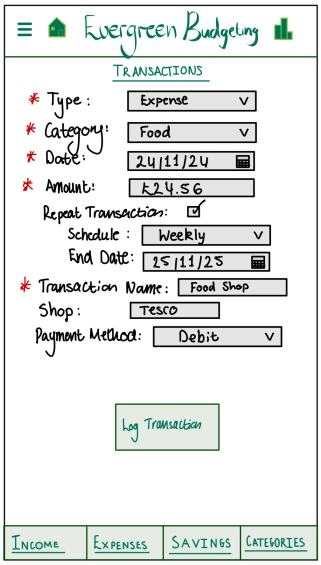


Figure F.8: *Phone wireframe for the log a transaction page.*

F.5 Budget Page

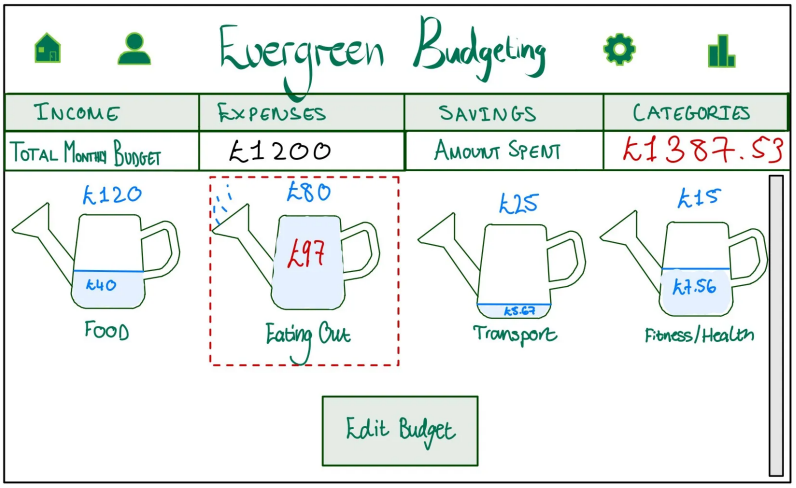


Figure F.9: *Desktop wireframe for the budget page.*

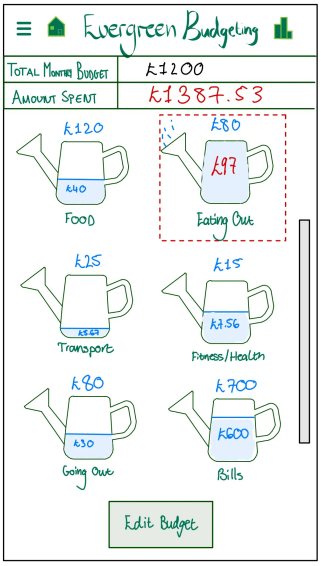


Figure F.10: *Phone wireframe for the budget page.*

F.6 Reports Page

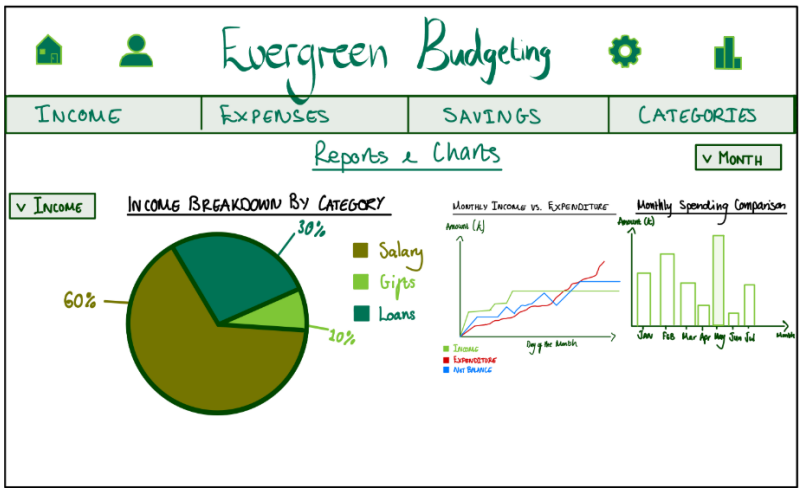


Figure F.11: *Desktop wireframe for the reports page.*

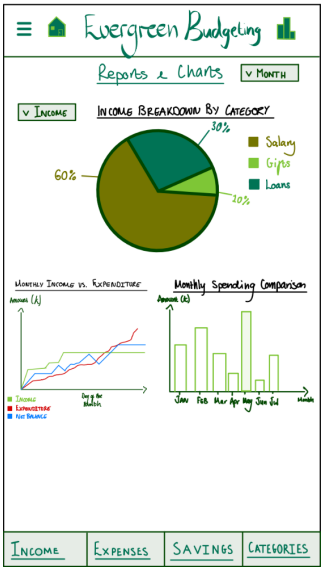


Figure F.12: *Phone wireframe for the reports page.*

F.7 Redesigned Nav Bar

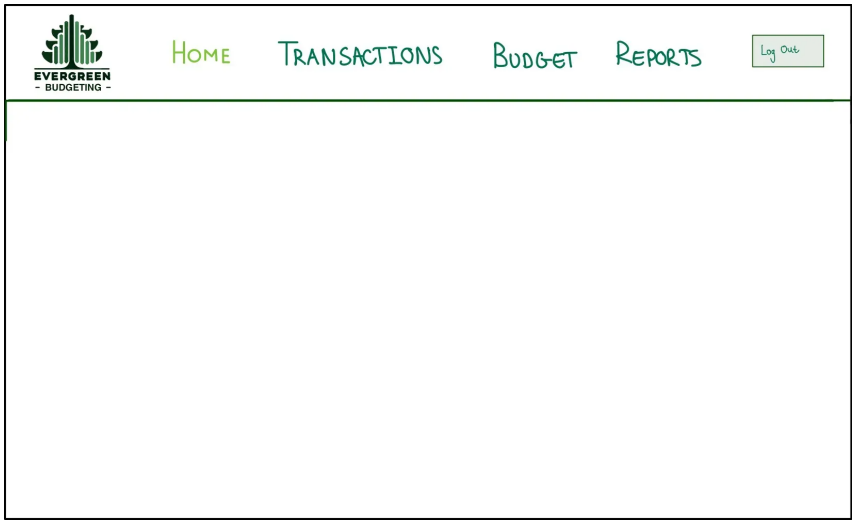


Figure F.13: *Updated Nav Bar Wireframe.*

G Logo Inspiration

Disclaimer: The following images were sourced via Google and are used here for illustrative purposes only. I do not claim ownership of these images.

Figure G.1: *Image found on Dreamstime.*

Figure G.2: *An Adobe Stock image.*

Figure G.3: *Logo of YNAB.*

Figure G.4: *Logo of Evergreen Building Co.*

H Software Engineering Processes

H.1 Issues

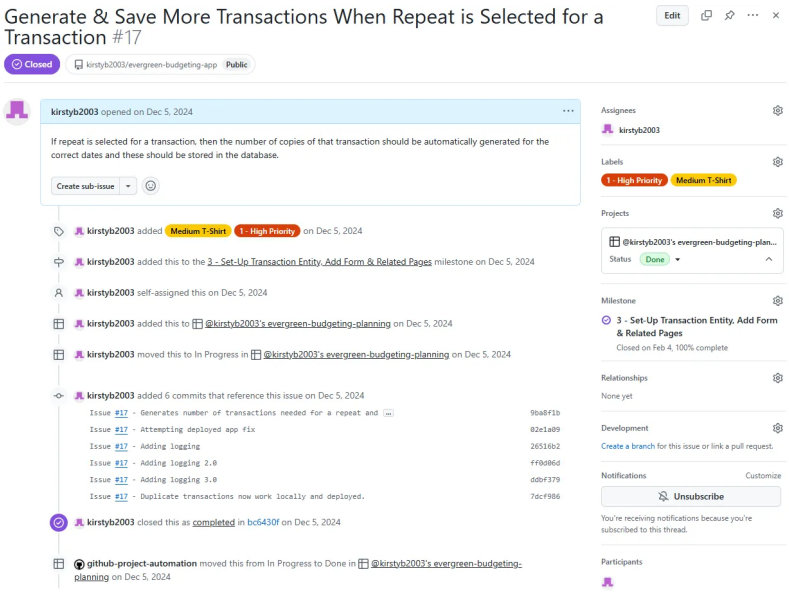


Figure H.1: *An issue showing the description, milestone, priority tag, time estimation tag and the history of commits related to the issue.*

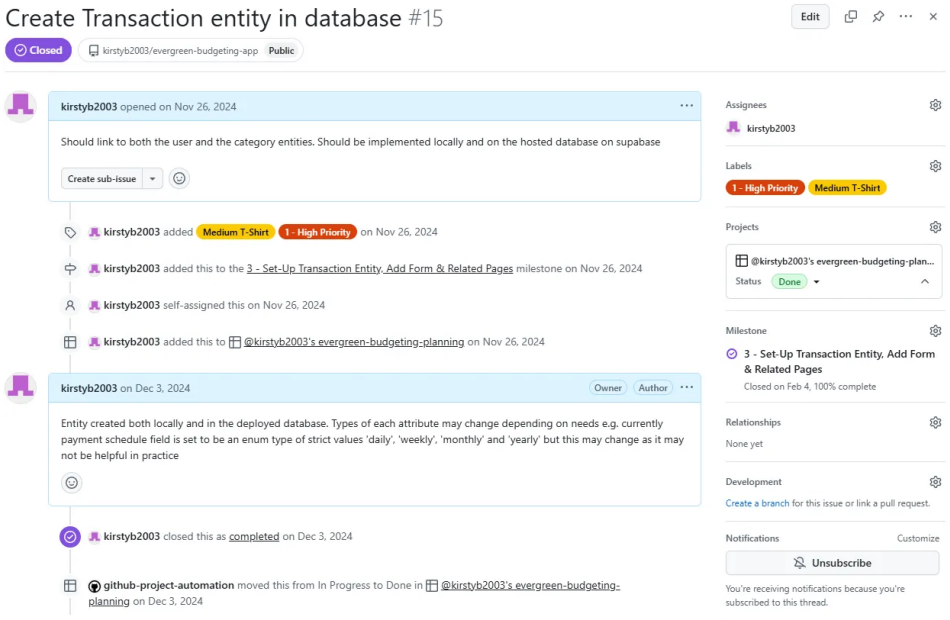


Figure H.2: *Issue showing a non-code task being closed with a comment that explains the actions taken.*

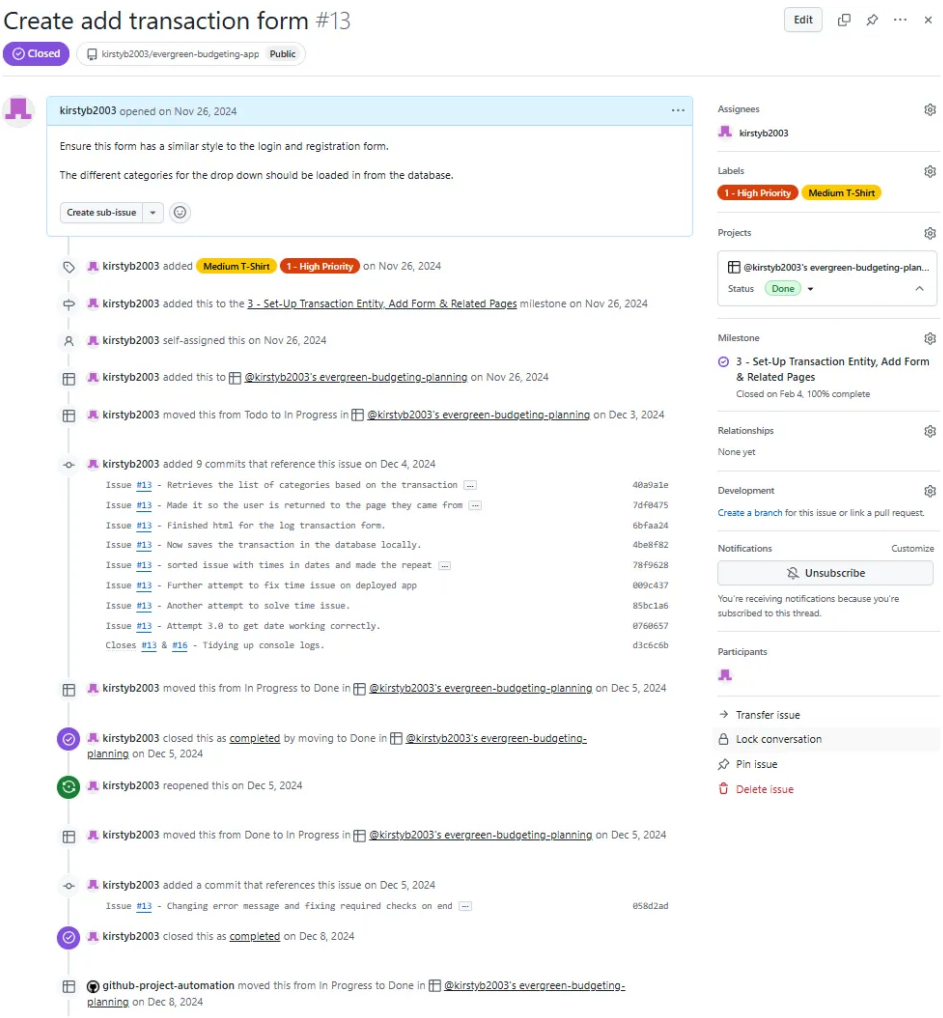


Figure H.3: *Issue illustrating documentation through commits.*

I UI Screenshots

I.1 Login & Registration Pages

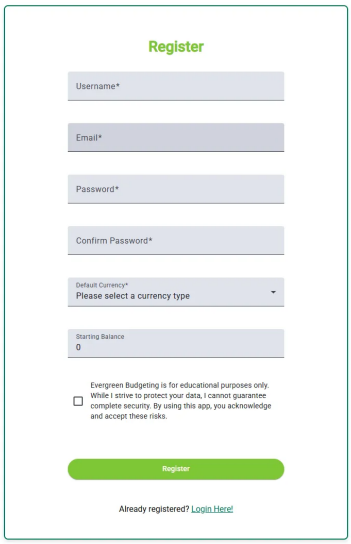


Figure I.1: *The implemented registration page. Note the default currency field.*

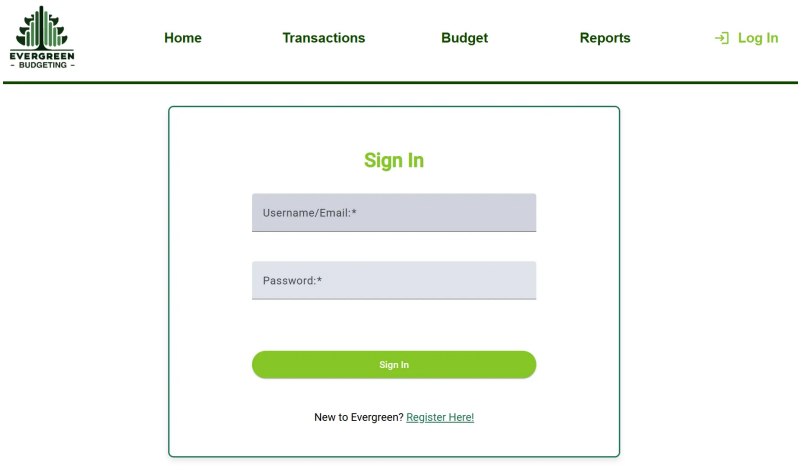


Figure I.2: *The implemented login page. Note that the nav bar matches the redesign.*

I.2 Home Page

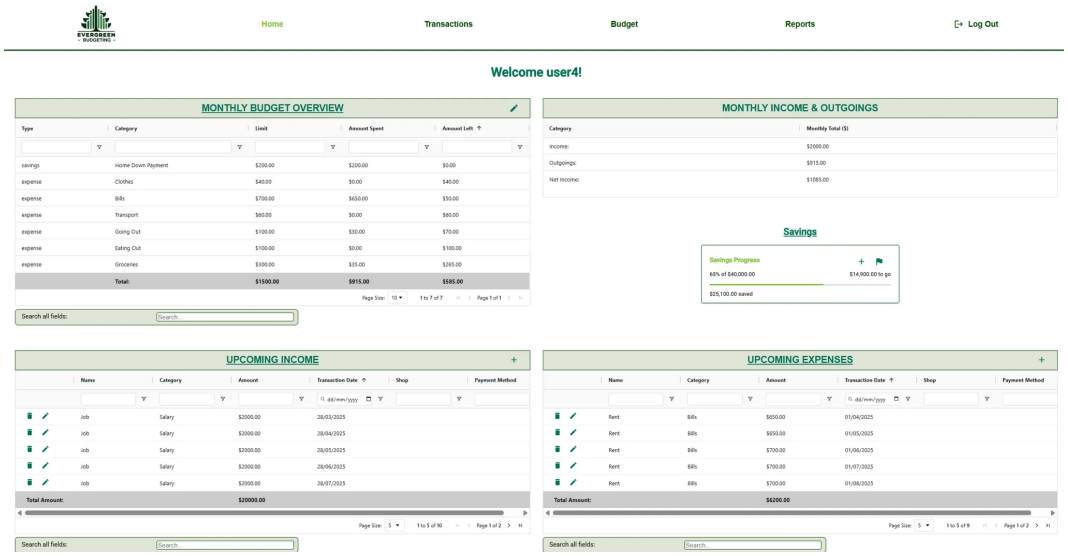


Figure I.3: *The implemented home page. Each main page of the app has a small overview on the home page. There is also a table in the top-right that gives an overview of the users income and outgoings for the month. Note the new nav bar design.*

I.3 Expenses/Income Page

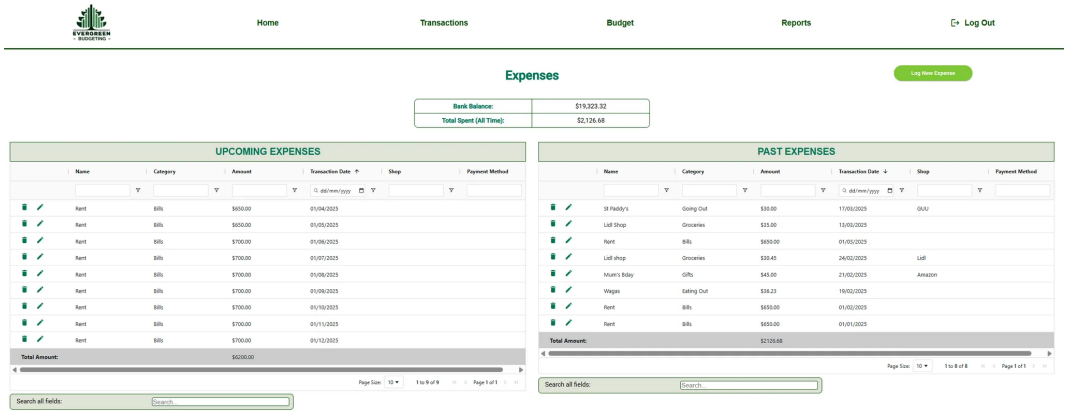


Figure I.4: *The implemented expenses page.*

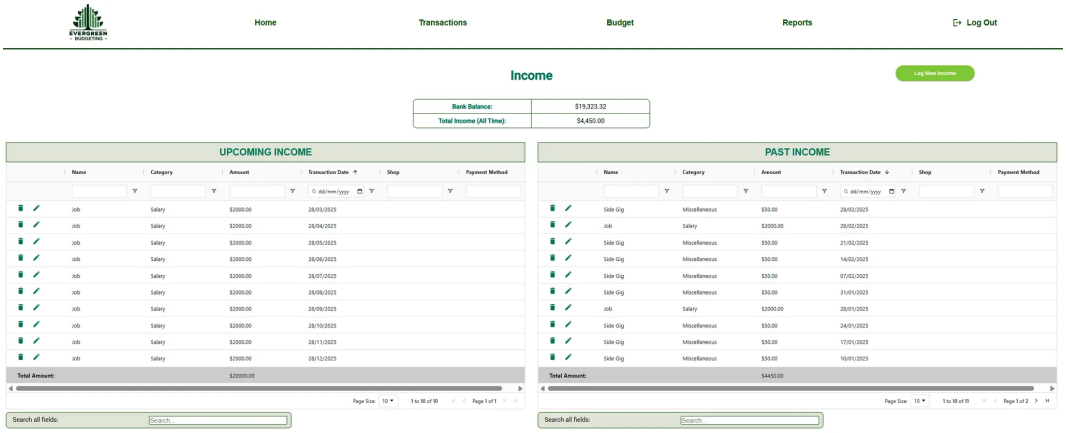


Figure I.5: *The implemented income page.*

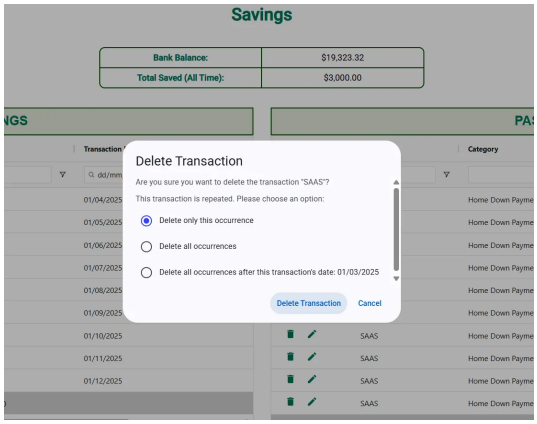


Figure I.6: *The popup that confirms the deletion of a transaction. For repeated transactions, the user is given the options as seen in the checkbox list.*

I.4 Savings Page

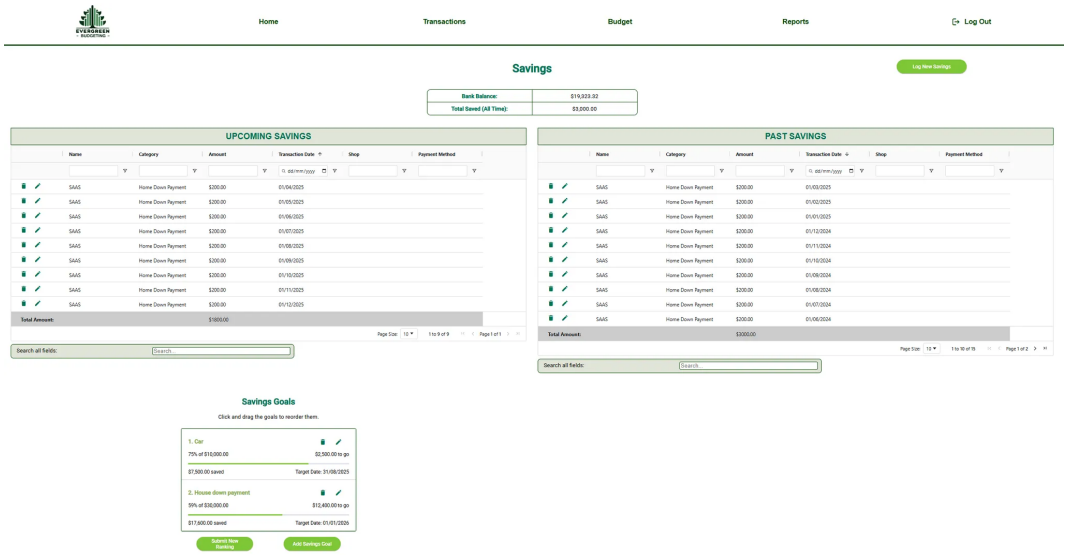


Figure I.7: *The implemented savings page. Note the savings goal area at the bottom of the screen.*

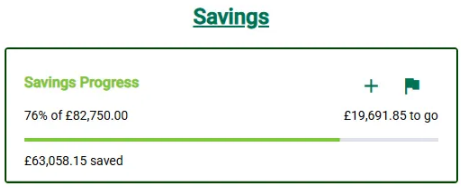


Figure I.8: *The savings section on the home page. The progress bar indicates how close the user is to meeting their aggregated savings goal.*

I.5 Budget Page

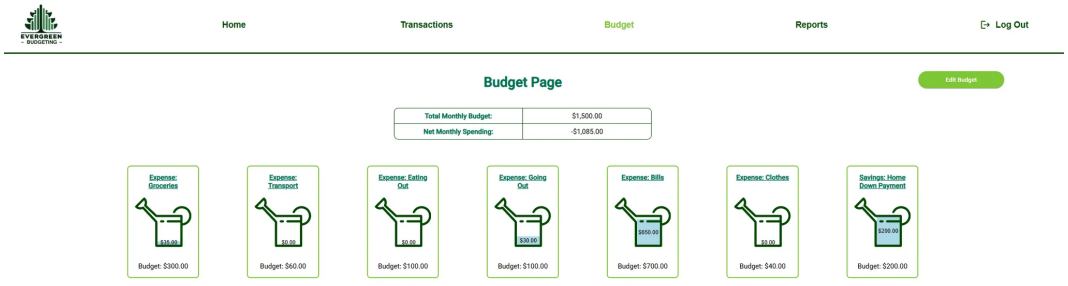


Figure I.9: *The implemented budget page with watering cans representing spending in each budget category.*

I.6 Reports Page

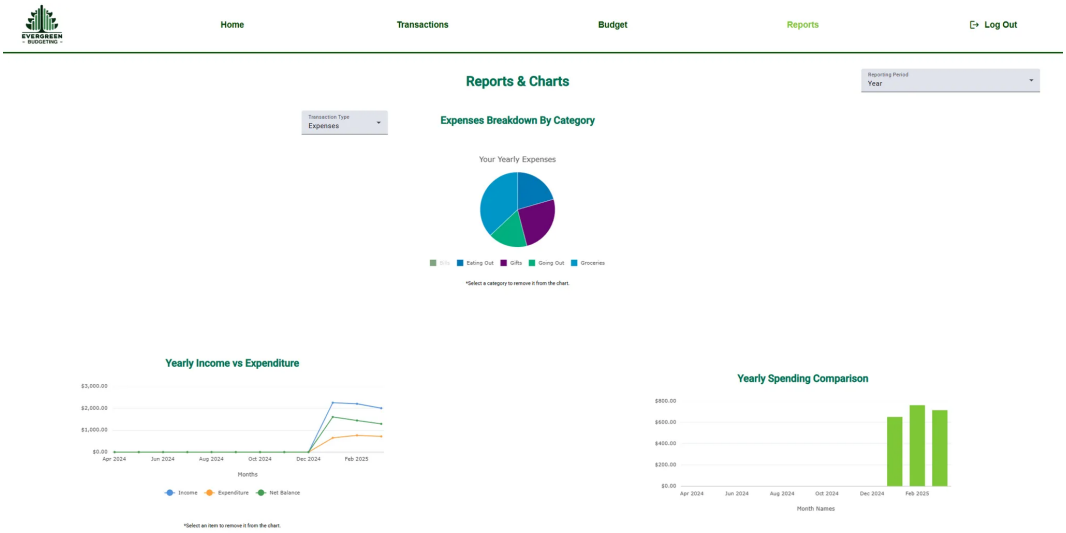
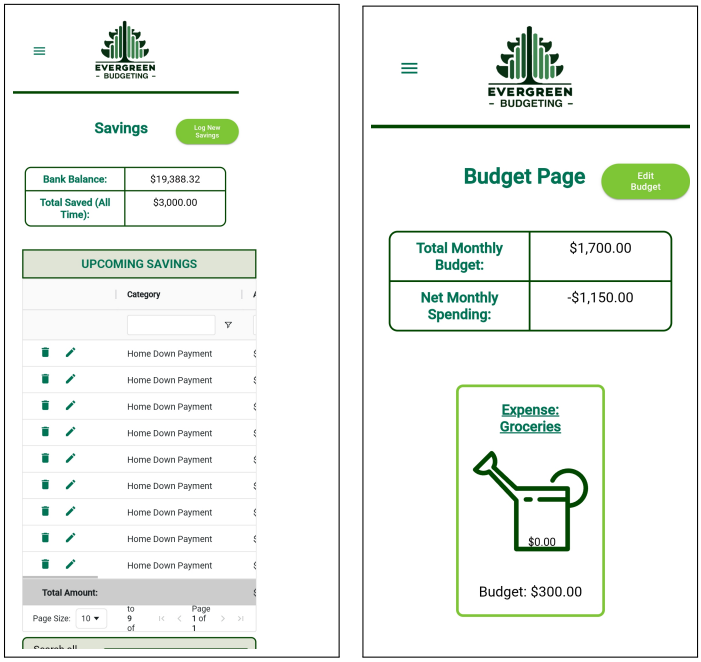


Figure I.10: *The implemented reports page.*

I.7 Mobile-App & Issues



(a) *Illustrates the disjointed alignment of components.*

(b) *Shows the navigation bar on smaller screens.*

Figure I.11: *(a) This misalignment is due to the sizing of the AG Grid tables, see the "Upcoming Savings" table, and the drag-and-drop areas, which are not visible as these are lower down the web page. (b) shows the nav bar for smaller screens, note that only the logo and hamburger menu are displayed.*

```
const authenticateToken = (req, res, next) => {

   const authHeader = req.headers['authorization'];
   const token = authHeader && authHeader.split(' ')[1];
   if (!token) return res.status(401).json({ error: 'Access token required'});

   jwt.verify(token, secretKey, (err, user) => {
      if (err) return res.status(403).json({ error: 'Invalid token' });
      req.user = user;
      next();
   });
};
```

Listing I.1: *The code in the authentication middleware which is used in the router.post calls as seen in Listing* ??*. If a token is provided, this code verifies it matches the token generated at login. If it does not, it throws an error.*

I.8 Authentication Middleware Code Listing

J Testing Code Coverage

J.1 Frontend Metrics

All files

92.98% Statements 967/1040 **85.83%** Branches 200/233 **91.24%** Functions 417/457 **93.18%** Lines 916/983

Press *n* or *j* to go to the next uncovered block, *b*, *p* or *k* for the previous block.

Filter:

File	Statements		Branches		Functions		Lines	
app	100%	3/3	100%	0/0	100%	1/1	100%	2/2
app/bank-balance	100%	56/56	100%	8/8	100%	24/24	100%	54/54
app/bar-chart	100%	44/44	100%	12/12	100%	21/21	100%	43/43
app/budget-page	100%	26/26	100%	2/2	100%	13/13	100%	24/24
app/budget-table	92.45%	49/53	92.85%	13/14	89.28%	25/28	98%	49/50
app/data-structures	100%	3/3	100%	0/0	100%	1/1	100%	2/2
app/display-savings-goals	90.62%	58/64	90%	9/10	88.88%	24/27	89.83%	53/59
app/home-page	100%	8/8	100%	0/0	100%	3/3	100%	7/7
app/line-graph	100%	43/43	100%	12/12	100%	21/21	100%	42/42
app/log-transaction-page	80.76%	84/104	62.96%	17/27	79.31%	23/29	80%	80/100
app/login-page	100%	15/15	100%	4/4	100%	7/7	100%	13/13
app/nav-bar	100%	12/12	100%	1/1	100%	5/5	100%	11/11
app/net-income-table	100%	24/24	100%	0/0	100%	15/15	100%	24/24
app/pie-chart	97.67%	42/43	100%	6/6	95.23%	20/21	97.67%	42/43
app/register-page	100%	43/43	100%	16/16	100%	21/21	100%	41/41
app/reports-page	100%	10/10	100%	0/0	100%	4/4	100%	10/10
app/savings-area	100%	24/24	100%	2/2	100%	12/12	100%	22/22
app/services	88.07%	96/109	100%	3/3	83.54%	66/79	87.73%	93/106
app/set-budget-page	92.02%	127/138	74.07%	40/54	89.65%	52/58	92.18%	118/128
app/set-savings-goal-page	97.61%	41/42	88.88%	8/9	100%	17/17	97.36%	37/38
app/table-action	70.96%	22/31	50%	3/6	55.55%	5/9	67.85%	19/28
app/transaction-display-page	91.3%	21/23	75%	3/4	85.71%	6/7	90.47%	19/21
app/transaction-table	93.81%	91/97	94.11%	32/34	89.28%	25/28	95.55%	86/90
app/watering-can	100%	24/24	100%	9/9	100%	6/6	100%	24/24
environments	100%	1/1	100%	0/0	100%	0/0	100%	1/1

Figure J.1: *Coverage metrics for unit tests of Angular components.*

All files app/services

88.07% Statements 96/109 **100%** Branches 3/3 **83.54%** Functions 66/79 **87.73%** Lines 93/106

Press *n* or *j* to go to the next uncovered block, *b*, *p* or *k* for the previous block.

Filter:

File	Statements		Branches		Functions		Lines	
authentication.service.ts	100%	22/22	100%	2/2	100%	8/8	100%	22/22
component-communication.service.ts	100%	5/5	100%	0/0	100%	2/2	100%	4/4
query.service.ts	84.81%	67/79	100%	1/1	82.35%	56/68	84.61%	66/78
window.service.ts	66.66%	2/3	100%	0/0	0%	0/1	50%	1/2

Figure J.2: *Coverage metrics for the frontend Angular services.*

J.2 Server Metrics

All files

96.17% Statements 628/653 **87.03%** Branches 94/108 **98.91%** Functions 91/92 **96.1%** Lines 617/642

Press *n* or *j* to go to the next uncovered block, *b*, *p* or *k* for the previous block.

Filter:

File	Statements		Branches		Functions		Lines	
api	99.13%	342/345	82.14%	23/28	97.77%	44/45	99.11%	337/340
api/database-queries	92.85%	286/308	88.75%	71/80	100%	47/47	92.71%	280/302

Figure J.3: *The overview of coverage metrics for the backend server.*

All files api

99.13% Statements 342/345 **82.14%** Branches 23/28 **97.77%** Functions 44/45 **99.11%** Lines 337/340

Press *n* or *j* to go to the next uncovered block, *b*, *p* or *k* for the previous block.

Filter:

File	Statements		Branches		Functions		Lines	
allow-cors.js	95%	19/20	70%	7/10	100%	2/2	94.73%	18/19
authMiddleware.js	100%	13/13	100%	6/6	100%	2/2	100%	11/11
database-routes.js	100%	282/282	100%	2/2	100%	37/37	100%	282/282
error-handler.js	100%	4/4	100%	4/4	100%	1/1	100%	4/4
index.js	90%	18/20	75%	3/4	66.66%	2/3	88.88%	16/18
pool.js	100%	6/6	50%	1/2	100%	0/0	100%	6/6

Figure J.4: *Coverage metrics for the API server files.*

All files api/database-queries

92.85% Statements 286/308 **88.75%** Branches 71/80 **100%** Functions 47/47 **92.71%** Lines 280/302

Press *n* or *j* to go to the next uncovered block, *b*, *p* or *k* for the previous block.

Filter:

File	Statements		Branches		Functions		Lines	
budget.js	100%	33/33	100%	6/6	100%	4/4	100%	33/33
categories.js	100%	14/14	100%	0/0	100%	3/3	100%	14/14
savings_goal.js	95.45%	63/66	100%	12/12	100%	9/9	95.38%	62/65
transactions.js	89.08%	155/174	85%	51/60	100%	27/27	88.75%	150/169
users.js	100%	21/21	100%	2/2	100%	4/4	100%	21/21

Figure J.5: *Coverage metrics for the server query files.*

K | Evaluation Ethics Checklist

School of Computing Science
University of Glasgow

Ethics checklist form for 3rd/4th/5th year, and taught MSc projects

This form is only applicable for projects that use other people ('participants') for the collection of information, typically in getting comments about a system or a system design, getting information about how a system could be used, or evaluating a working system.

If no other people have been involved in the collection of information, then you do not need to complete this form.

If your evaluation does not comply with any one or more of the points below, please contact the Chair of the School of Computing Science Ethics Committee (matthew.chalmers@glasgow.ac.uk) for advice.

If your evaluation does comply with all the points below, please sign this form and submit it with your project.

---

1. Participants were not exposed to any risks greater than those encountered in their normal working life.

   Investigators have a responsibility to protect participants from physical and mental harm during the investigation. The risk of harm must be no greater than in ordinary life. Areas of potential risk that require ethical approval include, but are not limited to, investigations that occur outside usual laboratory areas, or that require participant mobility (e.g. walking, running, use of public transport), unusual or repetitive activity or movement, that use sensory deprivation (e.g. ear plugs or blindfolds), bright or flashing lights, loud or disorienting noises, smell, taste, vibration, or force feedback

2. The experimental materials were paper-based, or comprised software running on standard hardware.

   Participants should not be exposed to any risks associated with the use of non-standard equipment: anything other than pen-and-paper, standard PCs, laptops, iPads, mobile phones and common hand-held devices is considered non-standard.

3. All participants explicitly stated that they agreed to take part, and that their data could be used in the project.

   If the results of the evaluation are likely to be used beyond the term of the project (for example, the software is to be deployed, or the data is to be published), then signed consent is necessary. A separate consent form should be signed by each participant.

   Otherwise, verbal consent is sufficient, and should be explicitly requested in the introductory script.

4. No incentives were offered to the participants.

   The payment of participants must not be used to induce them to risk harm beyond that which they risk without payment in their normal lifestyle.

5. No information about the evaluation or materials was intentionally withheld from the participants.
   Withholding information or misleading participants is unacceptable if participants are likely to object or show unease when debriefed.

6. No participant was under the age of 16.
   Parental consent is required for participants under the age of 16.

7. No participant has an impairment that may limit their understanding or communication.
   Additional consent is required for participants with impairments.

8. Neither I nor my supervisor is in a position of authority or influence over any of the participants.
   A position of authority or influence over any participant must not be allowed to pressurise participants to take part in, or remain in, any experiment.

9. All participants were informed that they could withdraw at any time.
   All participants have the right to withdraw at any time during the investigation. They should be told this in the introductory script.

10. All participants have been informed of my contact details.
    All participants must be able to contact the investigator after the investigation. They should be given the details of both student and module co-ordinator or supervisor as part of the debriefing.

11. The evaluation was discussed with all the participants at the end of the session, and all participants had the opportunity to ask questions.
    The student must provide the participants with sufficient information in the debriefing to enable them to understand the nature of the investigation. In cases where remote participants may withdraw from the experiment early and it is not possible to debrief them, the fact that doing so will result in their not being debriefed should be mentioned in the introductory text.

12. All the data collected from the participants is stored in an anonymous form.
    All participant data (hard-copy and soft-copy) should be stored securely, and in anonymous form.

Project title ___**Personal Budgeting Application**_____

Student's Name __**Kirsty Balfour**_____

Student Number __**2635375B**_____

Student's Signature 

Supervisor's Signature 

Date _16/03/2025__

L Evaluation Questionnaires & Results

L.1 In-Person Evaluation

Evergreen Budgeting App Evaluation - In-Person

This form is used to record the results of the evaluation of my dissertation budgeting app "Evergreen Budgeting". If there are any issues or questions, I can be contacted at 2635375B@student.gla.ac.uk

* Required

Introduction

Hello, and thank you for agreeing to take part in the evaluation of the Evergreen Budgeting App. The aim of this experiment is to assess the usability, functionality, and overall user experience of the budgeting app. I need the involvement of real users because I can't fully evaluate how well the app meets its goals without feedback from the people who are likely to use it. Your input will help refine the app and ensure it meets user expectations.

During this session, I will ask you to complete a set of tasks within the app, such as creating a budget, logging transactions, and setting savings goals. As you perform these tasks, I will be observing how you interact with the system, noting any challenges or points of confusion. Afterward, I will ask you some questions about your experience. The data collected will include:

- Your interaction with the app during the tasks.
- Your responses to the post-evaluation questions.
- Any comments or feedback you provide.

Please note that this is **not** a test of your ability, there are no right or wrong answers. The goal is to evaluate the app, not you. Feel free to ask me any questions during the session, and if at any point you feel uncomfortable or wish to stop, you are free to withdraw from the evaluation without any consequences. I can be contacted at 2635375B@student.gla.ac.uk if you have any questions or queries after the end of the evaluation.

1

Do you have any questions at this point?

2

Do you agree to take part in this evaluation and for your data to be used in my dissertation? *

- ◯ Yes
- ◯ No

3

If you consent, please sign below: *

Pre-Evaluation Questions & Demographics

4

What age range are you in? *

- ◯ 18-25
- ◯ 26-35
- ◯ 36-50
- ◯ 50-60
- ◯ 60+

5

How often do you review your budget or financial situation? *

- ◯ Daily
- ◯ Weekly
- ◯ Monthly
- ◯ When making a purchase
- ◯ Yearly
- ◯ Never

6

What methods do you currently use to manage your budget? *

- ◯ Pen and paper
- ◯ Spreadsheet
- ◯ Budgeting app
- ◯ Other

7

Have you used budgeting apps before? If yes, please list them *

Task 1 - Registration & Login

You can find the app here: https://evergreen-budgeting-app.web.app/

Using your laptop/desktop:

1. Navigate from the login page to the registration page.
2. Enter a username, email, password, default currency and a starting balance of your choice.
    - Nothing will be done with this information so it can be real or fake.
    - Make note of your username/email and password as this information will be needed at a later point.

1. Login with these details

8

Did the user manage to successfully login and access the other pages of the site? *

- ◯ Yes
- ◯ No
- ◯ Other

9

Was the user's chosen username visible on the home page? *

- ◯ Yes
- ◯ No
- ◯ Other

10

Could the chosen currency symbol be seen displayed on the home page (e.g. in the income and outgoings table)? *

- ◯ Yes
- ◯ No
- ◯ Other

11

Navigate to one of the transaction pages. Can their starting balance be seen as their bank balance? *

- ◯ Yes
- ◯ No
- ◯ Other

Task 2 - Creating a Monthly Budget Part A

1. Navigate to the budget page
2. Set the desired monthly spending amount to be 2000
3. You spend 40 on clothes each month, add this category to the budget
4. For the remainder of the budget, you can enter any amounts you like in any categories you like. Only do this in the expense categories currently.
5. Once your budget is at 2000 add 200 in any savings category you want

12

What is your total budgeted amount currently? *

13

Was it clear that you had gone over your desired spending amount after adding the savings? *

- ◯ Yes
- ◯ No
- ◯ Other

Task 2 - Creating a Monthly Budget Part B

1. Delete or edit any of your expense categories, other than clothing, so that you're back within your desired spending amount.
2. When you're happy, save the budget.

14

Did the user successfully create a budget that displays on the budget page? *

- ◯ Yes
- ◯ No
- ◯ Other

15

Did the user manage to add/delete/edit their budget categories as needed? *

- ◯ Yes
- ◯ No
- ◯ Other

16

Was there a warning at the top of the screen when the user went over their desired budget amount? *

- ◯ Yes
- ◯ No
- ◯ Other

17

Could the user view each of their budget categories as watering cans once created? *

- ◯ Yes
- ◯ No
- ◯ Other

18

Was the user's monthly budget amount displayed as 2000 when they submitted their budget? *

- ◯ Yes
- ◯ No
- ◯ Other

19

Did this process feel intuitive to how you'd usually think to break down your finances for the month? *

- ◯ Yes
- ◯ No
- ◯ Other

20

What did you like about this process? *

21

Was there anything you didn't like about this process? *

Task 3 - Setting Savings Goals Part A

1. Navigate to the savings page
2. Add a savings goal using the following information
    1. You're saving for a house down payment
    2. You want to save 30,000.00 and you currently have 17,600.00
    3. You're hoping to have this saved by the beginning of January next year.
    4. Save this goal

22

View the savings goal part of the page. How much do you have left to save? *

23

What percentage of the goal have you saved so far? *

Task 3 - Setting Savings Goals Part B

1. Add a new savings goal:
    1. You're saving for a second-hand car and want to buy this before the end of August
    2. You think this could cost about 10,000.00 in total and you currently have 4,500.00 put aside for it
    3. Save this goal
2. You forgot to take into account the savings you have in a separate account.
    1. There's 5,000.00 in this account.
    2. Edit one of the goals and add this amount onto the starting amount

24

Did the user manage to enter both savings goals with no issues? *

- ◯ Yes
- ◯ No
- ◯ Other

25

Were the created goals able to be viewed with the correct information? *

- ◯ Yes
- ◯ No
- ◯ Other

26

Did the user manage to edit the starting amount of a goal from the goals list? *

- ◯ Yes
- ◯ No
- ◯ Other

27

Is there any other information you would want to be able to add/display about your savings goals that isn't currently there? *

Task 4 - Adding Expenses Part A

1. Navigate to the expenses page
2. Your rent/mortgage each month is 650 and is taken out of your account on the 1st. Enter this repeating transaction from 01/01/2025 to 01/12/2025

28

Was it easy to figure out how to add a new expense? *

- ◯ Yes
- ◯ No
- ◯ Other

29

Was it clear what fields in the log transaction form were mandatory and which weren't? *

- ◯ Yes
- ◯ No
- ◯ Other

30

Please list the mandatory fields (feel free to go back and look at the form in order to answer this) *

31

How easy was it to understand how to add repeated transactions? *

- ◯ Extremely easy
- ◯ Somewhat easy
- ◯ Somewhat tricky
- ◯ Extremely tricky

32

Are there any repeat schedules (e.g. monthly, yearly) that you feel would be useful that aren't included? *

Task 4 - Adding Expenses Part B

1. Your electricity is 75 a month and council tax is 50, enter these for the same repeat period as you did for rent (i.e. from 01/01/2025 to 01/12/2025)
2. You went for a food shop yesterday at Lidl. You spent 28.98 and spent this on your credit card. Log this transaction.
3. Sorry! You actually spent 30.45 at Lidl as you had to go back in for milk. Can you update this?
4. You bought a gift for your Mum 4 days ago. You spent 45 at Amazon. Log this transaction
5. You went out to dinner with some friends last Wednesday at Wagamama's. Your meal cost 36.23. Log this transaction
6. You've been told by your landlord that your rent is going up by 50 a month from June onwards. Edit the rent transaction from the 1st of June onwards so that it's actually 700 a month instead
    1. Check this change to 700 has been done correctly by searching the upcoming expenses table for transactions named rent and for transaction dates from the 1st of June onward.
7. You went clothes shopping today and spent 64.58 at H&M and you bought this using your debit card. Log this transaction

33

Did the user manage to create all transactions correctly? *

- ◯ Yes
- ◯ No
- ◯ Other

34

Did each transaction appear in the correct upcoming or past tables with all the correct information? *

- ◯ Yes
- ◯ No
- ◯ Other

35

Did the repeat transactions create the correct number of transactions (i.e. 12)? *

- ◯ Yes
- ◯ No
- ◯ Other

36

Were any edits to transactions shown immediately in the tables? *

- ◯ Yes
- ◯ No
- ◯ Other

37

Did the user manage to successfully edit information about repeating transactions? *

- ◯ Yes
- ◯ No
- ◯ Other

38

Is the rent transaction on the 1st of May 650? *

- ◯ Yes
- ◯ No
- ◯ Other

39

Is the rent transaction on the 1st of June and the 1st of August 700? *

- ◯ Yes
- ◯ No
- ◯ Other

40

Is there any information that you feel is missing from the transaction form? If so, what? *

41

How do you feel about the way the transactions are displayed e.g. the information shown, the ordering, etc? *

42

Would you prefer the transactions in one big table rather than split between past and future transactions? If so, why? *

43

How did you find searching the tables? Was this easy enough to do? *

Task 5 - Adding Income

1. Navigate to the income page
2. You get 2,000 a month from your job. Your pay day is the 28th of each month. Log this transaction from January to December of this year.
3. You actually make 1,800 each month. Update all these transactions
4. You've just gotten a side gig that pays 50 a week each Friday. Log this from January to December this year
5. Oh no, this fell through, delete any transactions that occur past February
6. Your gran has very generously given you 500 on the 4th of February. Log this transaction

44

Did the user manage to create all transactions correctly? *

- ◯ Yes
- ◯ No
- ◯ Other

45

Does each transaction get displayed in the correct past/future table with the correct information? *

- ◯ Yes
- ◯ No
- ◯ Other

46

Were the correct number of transactions generated by the repeated transactions? (i.e. 12 for the monthly and 52 for the weekly) *

- ◯ Yes
- ◯ No
- ◯ Other

47

Were any edits to transactions displayed immediately? *

- ◯ Yes
- ◯ No
- ◯ Other

48

Was the user able to edit multiple repeated transactions in one go? *

- ◯ Yes
- ◯ No
- ◯ Other

49

Did the delete operation only delete the transactions after the selected date? *

- ◯ Yes
- ◯ No
- ◯ Other

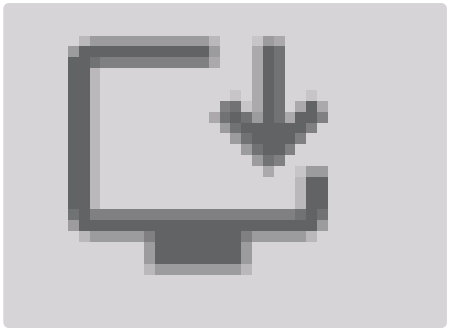
Device Usage

50

Will the user be making use of their phone for the next part of the evaluation? *

- ◯ Yes
- ◯ No

Task 6 - PWA Mobile App



1. Using your phone, visit https://evergreen-budgeting-app.web.app and download the app
    1. For Android, this should appear as a notification at the top of the page or as the symbol seen above
    2. For iPhone, go to share and click add to home screen
    3. If these steps don't work, feel free to use the browser site instead of downloading the app
2. Login to the account you made using the laptop/desktop

Task 7 - Add Savings Transactions

1. Navigate to the savings page
2. You put 200 a month into savings and have been doing so since the beginning of January last year. Log these transactions until the end of this year.
    1. You can choose which date this repeats on and what category the savings fall under.
3. You've realised that you want to prioritise getting a car before a house. Reprioritise your savings goals and save this new ordering

51

Did the user manage to successfully log each transaction correctly? *

- ◯ Yes
- ◯ No
- ◯ Other

52

Did the user manage to successfully reorder their savings goals? *

- ◯ Yes
- ◯ No
- ◯ Other

53

Did the user find it easy to tell how to reorder these goals? *

- ◯ Yes
- ◯ No
- ◯ Other

54

Can you tell me how reordering your goals has effected how much you have saved towards each goal? *

55

Is it obvious where this extra money towards to goals has come from? *

Task 8 - Viewing the Budget & Home Page Part A

1. Navigate to the budget page

56

What categories have you overspent in (if any)? *

57

Roughly, as a percentage, how much have spent out of your budget for a category of your choice? (write the category and the percentage) *

Task 8 - Viewing the Budget & Home Page Part B

1. Navigate to the home page

58

Which page do you feel displayed the budget/category information more clearly? *

- ◯ Budget Page
- ◯ Home Page
- ◯ Other

59

Why did you find this page clearer? *

60

What would you click to change your budget from the home page? *

61

I want you to add a new savings goal from the home page. What would you click to do this? *

62

Can you show me what buttons you would click to add a new expense transaction from the home page? *

63

Can you show me what buttons you would click to add a new income transaction from the home page? *

64

Can you show me what buttons you would click to add a new savings transaction from the home page? *

65

Could the user easily tell at a glance what each category in the budget page is telling them? *

- ◯ Yes
- ◯ No
- ◯ Other

66

Could the user easily determine how to edit their budget and add transactions from the home page? *

- ◯ Yes
- ◯ No
- ◯ Other

Task 9 - View Reports & Charts

1. Navigate to the reports page
2. You want to view how much you've spent in each of your expenses categories in the last year. Choose the relevant options in the drop downs to get this report.

67

The chart isn't very clear because the bills section is taking up most of the space. Can you remove this category from the chart? *

68

What category have you spent the most in outside of bills? *

69

What day have you spent the most money in the last week? *

70

What's been your biggest income category in the last month? *

71

What's been your biggest income category in the last week? *

72

What does the bottom left/middle graph represent? *

73

What does this graph tell you about your spending and income? *

74

Could the user easily read each chart? *

- ◯ Yes
- ◯ No
- ◯ Other

75

Could the user easily customise what data was displayed by the charts? *

- ◯ Yes
- ◯ No
- ◯ Other

76

Did you like the charts used to display this information? Or would you have found other chart types clearer for certain information? E.g. a bar chart to display category information instead *

Post-Evaluation Questions

77

How easy was it to navigate between pages of the app/site? *

- ◯ Extremely easy
- ◯ Somewhat easy
- ◯ Somewhat tricky
- ◯ Extremely tricky

78

Were there any features you found confusing or difficult to use? *

79

Was there any feature that didn't work how you expected it to? E.g. a button taking you somewhere you didn't expect; savings page coming under transactions; symbols used having a different meaning than you expected; etc *

80

What did you like most about the app? *

81

Would this app replace your current budgeting method? Why or why not? *

82

Are there any features that you would/expect that aren't there? If so, what are these features? *

83

Did you make use of the mobile version of the app during this evaluation? *

- ◯ Yes
- ◯ No

84

Did you prefer the computer or phone version of the app? *

- ◯ Computer
- ◯ Phone
- ◯ Other

85

Was it convenient to use the app on your phone? Why or why not? *

86

Is there any other feedback or comments you'd like to make? *

87

Would you like your account to be kept after the evaluation ends or can I delete it? *

- ◯ I would like to keep it
- ◯ Delete it
- ◯ I don't mind

Debrief

Thank you for participating in this evaluation of my budgeting app. The main aim of this experiment was to assess the usability and functionality of the application and gather feedback on your experience. Your insights will be invaluable in improving the system.
Please take note of my contact details and my supervisor's details in case you have any further questions or concerns.

- Kirsty Balfour: 2635375B@student.gla.ac.uk
- Sofiat Olaosebikan: sofiat.olaosebikan@glasgow.ac.uk

Thank you once again for your time and input.

88

Do you have any final comments or questions about the evaluation?

L.2 Remote Evaluation

Evergreen Budgeting App Evaluation - Remote

This form is used to record the results of the evaluation of my dissertation budgeting app "Evergreen Budgeting". If there are any issues or questions, I can be contacted at 2635375B@student.gla.ac.uk

* Required

Introduction

Hello, and thank you for agreeing to take part in the evaluation of the Evergreen Budgeting App. The aim of this experiment is to assess the usability, functionality, and overall user experience of this app. I need the involvement of real users because I can't fully evaluate how well the app meets its goals without feedback from the people who are likely to use it. Your input will help refine the app and ensure it meets user expectations.

During the session, you will be asked to complete a set of tasks within the app, such as creating a budget, logging transactions, and setting savings goals. As you perform these tasks, you will be asked how the tasks went and if you faced any challenges or points of confusion. Afterward, you will be asked some questions about your experience. The data collected will include:

- Your interaction with the app during the tasks.
- Your responses to the post-evaluation questions.
- Any comments or feedback you provide.

Please note that this is **not** a test of your ability, there are no right or wrong answers. The goal is to evaluate the app, not you. If at any point you feel uncomfortable or wish to stop, you are free to withdraw from the evaluation without any consequences, though by doing so, you acknowledge that you won't be able to be debriefed at the end of the evaluation. I can be contacted at 2635375B@student.gla.ac.uk if you have any questions or queries during or after the evaluation.

Please Note: all task instructions will be contained within this form but you will be required to make use of my website/app so you may find it more convenient to either have 2 devices (e.g. a phone and a laptop) or a bigger screen while completing this evaluation.

1

Do you have any comments to make at this point?

2

Do you agree to take part in this evaluation and for your anonymised data to be used in my dissertation? *

- ◯ Yes
- ◯ No

Pre-Evaluation Questions & Demographics

3

What age range are you in? *

- ◯ 18-25
- ◯ 26-35
- ◯ 36-50
- ◯ 50-60
- ◯ 60+

4

How often do you review your budget or financial situation? *

- ◯ Daily
- ◯ Weekly
- ◯ Monthly
- ◯ When making a purchase
- ◯ Yearly
- ◯ Never

5

What methods do you currently use to manage your budget? *

- ◯ Pen and paper
- ◯ Spreadsheet
- ◯ Budgeting app
- ◯ Other

6

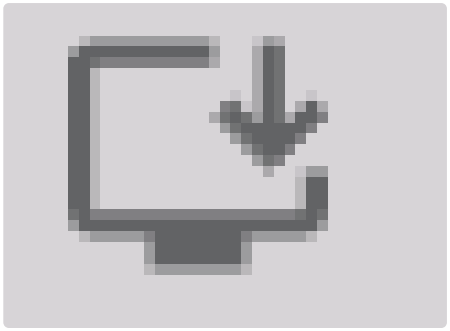
Have you used budgeting apps before? If yes, please list them *

7

Which device will you be using to conduct the evaluation today? *

- ◯ Phone
- ◯ Laptop/desktop
- ◯ Tablet
- ◯ Other

PWA Mobile App



1. Using your phone, visit https://evergreen-budgeting-app.web.app and download the app
    1. For Android, this should appear as a notification at the top of the page or click the symbol as seen above
    2. For iPhones, go to share and click add to home screen
    3. If these steps don't work, feel free to use the browser site instead of downloading the app

8

Are you making use of the website or downloaded app? *

- ◯ Website
- ◯ App
- ◯ Other

9

Why are you not making use of the app? E.g. couldn't download the app, didn't want to download it, etc *

Task 1 - Registration & Login

You can find the app here: https://evergreen-budgeting-app.web.app/

1. Navigate from the login page to the registration page.
2. Enter a username, email, password, default currency and a starting balance of your choice.
    - Nothing will be done with this information so it can be real or fake.
    - Make note of your username/email and password as this information will be needed at a later point.

1. Login with these details

10

Did you manage to successfully login and access the home page of the site? *

- ◯ Yes
- ◯ No

11

If you had issues logging in/registering, what was the issue?

12

Was there any error messages? And if so, were these informative?

13

Was your chosen username visible on the home page? *

- ◯ Yes
- ◯ No
- ◯ Other

14

If no, what was displayed instead?

15

Could your chosen currency symbol be seen displayed on the home page (e.g. in the income and outgoings table)? *

- Yes
- No
- Other

16

If no, what was displayed instead? Or could you not find where the symbol should be displayed?

17

Navigate to one of the transaction pages. Does the bank balance displayed at the top of the page match the starting balance you entered when registering? *

- Yes
- No
- Other

18

If no, what was displayed instead? Or could you not find one of the transaction pages?

Task 2 - Creating a Monthly Budget Part A

1. Navigate to the budget page and create a budget
2. You spend 40 on clothes each month, add this category to the budget
3. For the remainder of the budget, you can enter any amounts you like in any categories you like, but it must total 1,500.
    1. Only do this in the expense categories currently.
4. Once your budget is at 1,500 add 200 in any savings category you want

19

What is your total budgeted amount currently? *

20

Was it easy to tell how much your budget added up to when creating it? *

- ◯ Yes
- ◯ No
- ◯ Other

21

If no, why wasn't this easy? *

22

Did you make use of the desired monthly spending amount field (top)? *

- ◯ Yes
- ◯ No

23

What value did you put in this field? *

24

Did you get a warning that you had gone over your desired spending amount at any point? And was this message obvious? *

25

Did you manage to successfully create and save the budget? *

- ◯ Yes
- ◯ No
- ◯ Other

26

If no, what problems did you run into/what were the error messages? *

27

Were each of the categories you'd created in the budget viewable as watering cans on the budget page when the budget had been saved? *

- ◯ Yes
- ◯ No
- ◯ Other

28

If no, what was displayed on the budget page instead? *

29

Did this process feel intuitive to how you'd usually think to break down your finances for the month? Why or why not? *

30

What did you like about this process? *

31

Was there anything you didn't like about this process? *

Task 3 - Setting Savings Goals Part A

1. Navigate to the savings page
2. Add a savings goal using the following information
    1. You're saving for a house down payment
    2. You want to save 30,000.00 and you currently have 17,600.00
    3. You're hoping to have this saved by the beginning of January next year.
    4. Save this goal

32

View the savings goal part of the page. How much do you have left to save? *

33

What percentage of the goal have you saved so far? *

Task 3 - Setting Savings Goals Part B

1. Add a new savings goal:
    1. You're saving for a second-hand car and want to buy this before the end of August
    2. You think this could cost about 10,000.00 in total and you currently have 4,500.00 put aside for it
    3. Save this goal

34

Did you manage to enter and save each savings goal? *

- ◯ Yes
- ◯ No
- ◯ Other

35

If no, what went wrong? *

36

Is there any other information you would want to be able to display about your savings goals that isn't currently there? *

37

What did you like/dislike about this process and the way the goals are displayed? *

Task 4 - Adding Expenses Part A

1. Navigate to the expenses page
2. Your rent each month is 650 and is taken out of your account on the 1st of each month. Enter this repeating transaction from 01/01/2025 to 01/12/2025

38

Was it easy to figure out how to add a new expense? *

- ◯ Yes
- ◯ No
- ◯ Other

39

If no, why wasn't this easy? *

40

Was it clear what fields in the log transaction form were mandatory and which weren't? *

- ◯ Yes
- ◯ No
- ◯ Other

41

Why wasn't this clear? *

42

How easy was it to understand how to add repeated transactions? *

- ◯ Extremely easy
- ◯ Somewhat easy
- ◯ Somewhat tricky
- ◯ Extremely tricky

43

Are there any repeat schedules (e.g. monthly, yearly) that you feel would be useful that aren't included? *

Task 4 - Adding Expenses Part B

1. You went for a food shop yesterday at Lidl. You spent 28.98 and spent this on your credit card. Log this transaction.
2. Sorry! You actually spent 30.45 at Lidl as you had to go back in for milk. Can you update this?
3. You bought a gift for your Mum 4 days ago. You spent 45 at Amazon. Log this transaction
4. You went out to dinner with some friends last Wednesday at Wagamama's. Your meal cost 36.23. Log this transaction
5. You've been told by your landlord that your rent is going up by 50 a month from June onwards. Edit the rent transaction from the 1st of June onwards so that it's actually 700 a month instead
6. You went clothes shopping today and spent 64.58 at H&M and you bought this using your debit card. Log this transaction

44

Did you manage to edit multiple transactions in one go? I.e. task 5 *

- ◯ Yes
- ◯ No
- ◯ Other

45

If you answered no, what went wrong/what did you struggle with? *

46

Is the rent transaction on the 1st of May 650? *

- ◯ Yes
- ◯ No
- ◯ Other

47

Is the rent transaction on the 1st of June 700? *

- ◯ Yes
- ◯ No
- ◯ Other

48

If you answered no to either of the previous questions, what were these values instead?

49

Is there any information that you feel is missing from the transaction form? If so, what? *

50

How do you feel about the way the transactions are displayed e.g. the information shown, the ordering, etc? *

51

Would you prefer the transactions in one big table rather than split between past and future transactions? If so, why? *

52

How did you find searching the tables? Was this easy enough to do? *

Task 5 - Adding Income

1. Navigate to the income page
2. You get 2,000 a month from your job. Your pay day is the 28th of each month. Log this transaction from January to December of this year.
3. You've just gotten a side gig that pays 50 a week each Friday. Log this from January to December this year
4. Oh no, this fell through, delete any transactions that occur past February

53

Did you manage to delete the correct transactions in task 4? *

- ◯ Yes
- ◯ No
- ◯ Other

54

If no, what did you struggle with/what went wrong? *

55

Overall, what did you like and dislike about the process of adding income and expenses? *

Task 7 - Add Savings Transactions

1. Navigate to the savings page
2. You put 200 a month into savings and have been doing so since the beginning of January last year. Log these transactions until the end of this year.
    1. You can choose which date this repeats on and what category the savings fall under.
3. You've realised that you want to prioritise getting a car before a house. Reprioritise your savings goals and submit this new ordering

56

Did you manage to successfully reorder your savings goals? *

- ◯ Yes
- ◯ No
- ◯ Other

57

If no, what went wrong? Did you remember to click the submit this new ranking? *

58

Did you find it easy to tell how to reorder these goals? *

- ◯ Yes
- ◯ No
- ◯ Other

59

Can you tell me how reordering your goals has effected how much you have saved towards each goal? *

60

Is it obvious where this extra money towards the goals has come from? If so, can you tell me where you think the money comes from? *

Task 8 - Viewing the Budget & Home Page Part A

1. Navigate to the budget page

61

What categories have you overspent in (if any)? *

62

Roughly, as a percentage, how much have spent out of your budget for a category of your choice? (write the category and the percentage) *

63

Was it easy to tell at a glance the amount you'd spent of a category? *

- ◯ Yes
- ◯ No
- ◯ Other

64

If no, why wasn't this clear? *

Task 8 - Viewing the Budget & Home Page Part B

1. Navigate to the home page

65

Which page do you feel displayed the budget/category information more clearly? *

- ◯ Budget Page
- ◯ Home Page
- ◯ Other

66

Why did you find this page clearer? *

Task 9 - View Reports & Charts

1. Navigate to the reports page
2. You want to view how much you've spent in each of your expenses categories in the last year. Choose the relevant options in the drop downs to get this report.

67

The chart isn't very clear because the bills section is taking up most of the space. Can you remove this category from the chart? Note down how you did this or if you didn't *

68

What category have you spent the most in outside of bills? *

69

What day have you spent the most money in the last week? *

70

What's been your biggest income category in the last month? *

71

Did you like the charts used to display this information? Or would you have found other chart types clearer for certain information? E.g. a bar chart to display category information instead of the pie chart *

Post-Evaluation Questions

72

How easy was it to navigate between pages of the app/site? *

- ◯ Extremely easy
- ◯ Somewhat easy
- ◯ Somewhat tricky
- ◯ Extremely tricky

73

Were there any features you found confusing or difficult to use? *

74

Was there any feature that didn't work how you expected it to? E.g. a button taking you somewhere you didn't expect; savings page coming under transactions; symbols used having a different meaning than you expected; etc *

75

What did you like most about the app? *

76

Would this app replace your current budgeting method? Why or why not? *

77

Are there any features that you would want/expect that aren't there? If so, what are these features? *

78

Did you make use of the mobile version of the app during this evaluation? *

- ◯ Yes
- ◯ No

79

Was it convenient to use the app on your phone? Why or why not? *

80

What would you like done with your account after this evaluation has ended? *

- ◯ Keep it
- ◯ Delete it
- ◯ I don't mind

81

What was the username/email you used when creating the account? *

82

Is there any other feedback or comments you'd like to make? *

Debrief

Thank you for participating in this evaluation of my budgeting app. The main aim of this experiment was to assess the usability and functionality of the application and gather feedback on your experience. Your insights will be invaluable in improving the system.
Please take note of my contact details and my supervisor's details in case you have any further questions or concerns.

- Kirsty Balfour: 2635375B@student.gla.ac.uk
- Sofiat Olaosebikan: sofiat.olaosebikan@glasgow.ac.uk

Thank you once again for your time and input.

83

Do you have any final comments or questions about the evaluation?

L.3 Links to Results

L.3.1 Supervised Evaluation Microsoft Form Results

https://forms.office.com/Pages/AnalysisPage.aspx?AnalyzerToken=1GuUBmawOfbSi2YK6Qt061tDcgVVl
id=KVxybjp2UE-B8i4lTwEzyLzaCqY070xHgthppmcrVEpUOVpYNElCVUZXQ0JQRkI5T1A1MjNaREcyVy4u

L.3.2 Unsupervised Evaluation Microsoft Form Results

https://forms.office.com/Pages/AnalysisPage.aspx?AnalyzerToken=wz7TPfdjadrJxikAVfc1UFHjoNqL9
id=KVxybjp2UE-B8i4lTwEzyLzaCqY070xHgthppmcrVEpUMjhPUjBYMEs3MVQ3RFNNRVFRWUgzTDAwUi4u

L.3.3 Combined & Summarised Results

Hosted with Notion: https://dolomite-account-777.notion.site/Evaluation-Discussion-1c11f372b4378
pvs=4

M | Evaluation Charts

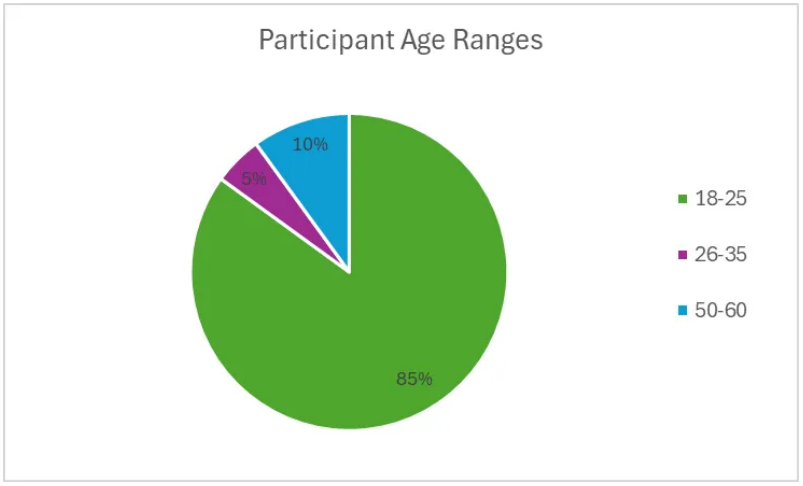


Figure M.1: *Chart showing the distribution of participants' ages.*

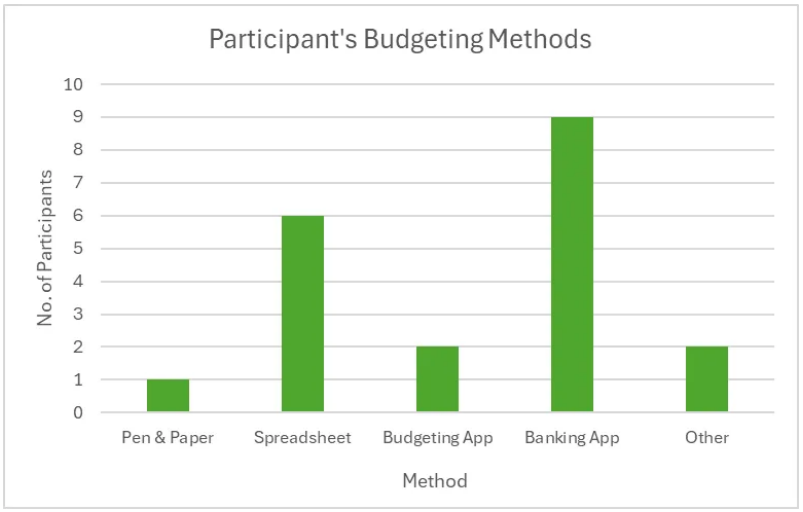


Figure M.2: *Chart showing the most common budgeting methods amongst the participants*

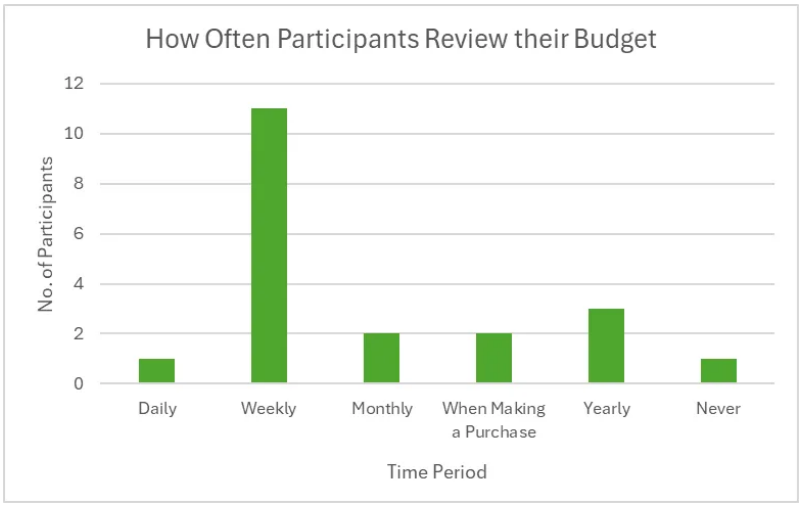


Figure M.3: *Chart showing how often the participants review their budgets.*

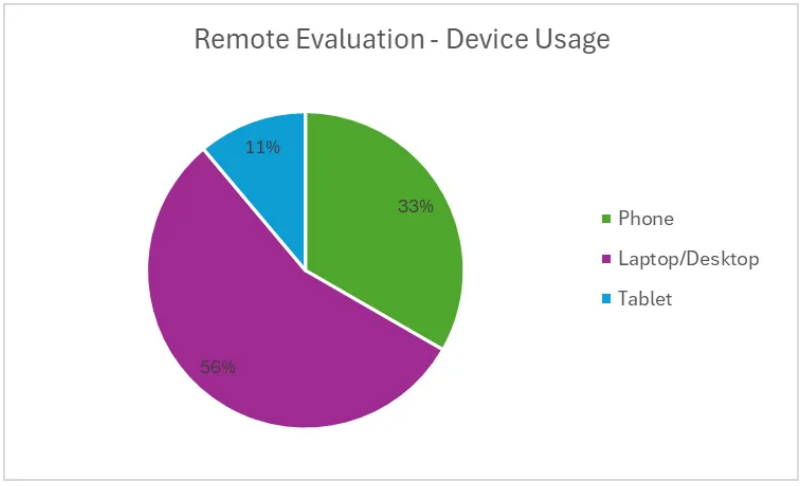


Figure M.4: *Chart showing the devices used to conduct the evaluation by unsupervised participants.*

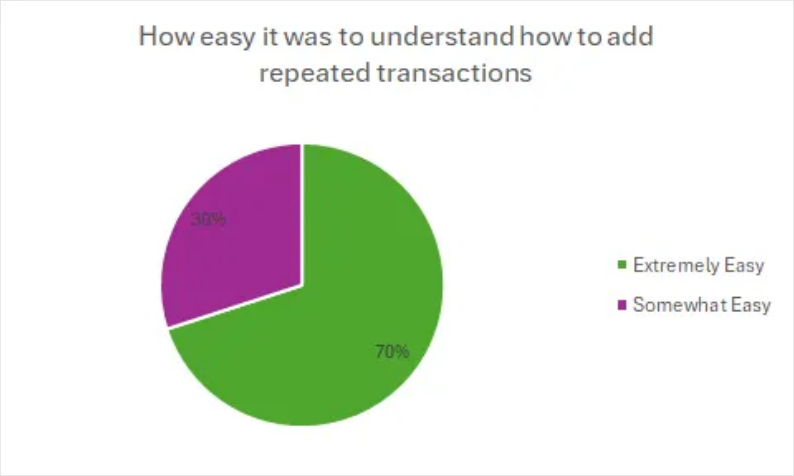


Figure M.5: *Chart depicting how easy participants found adding repeat transactions.*

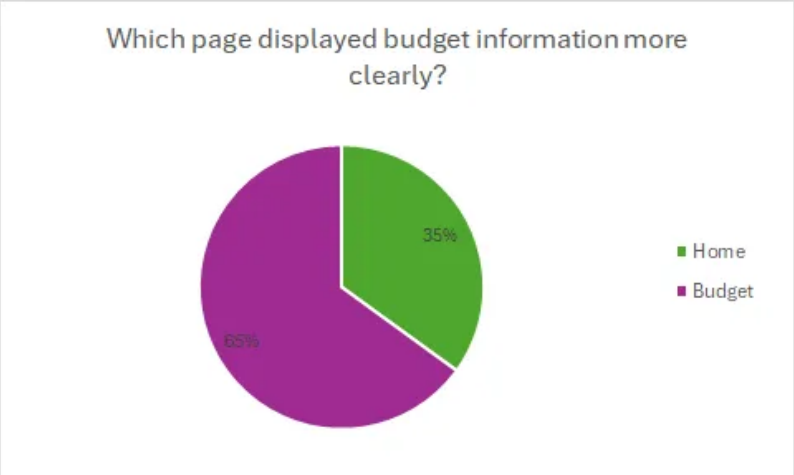


Figure M.6: *Chart showing the preferred page to view budget information.*

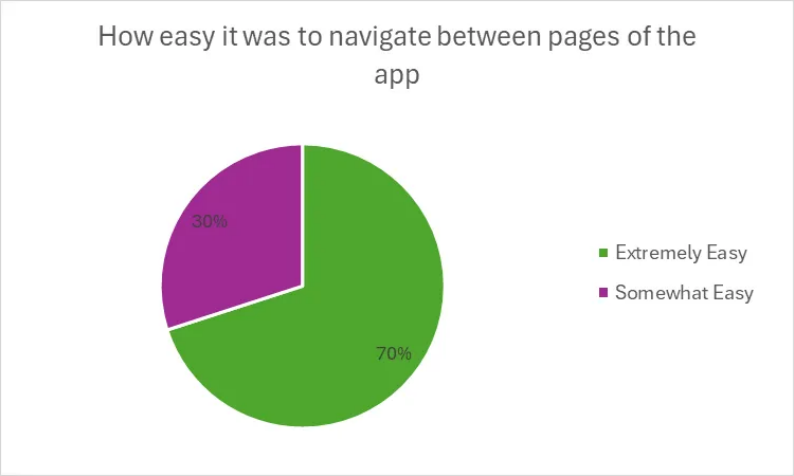


Figure M.7: *Chart illustrating ease of navigation across the site.*

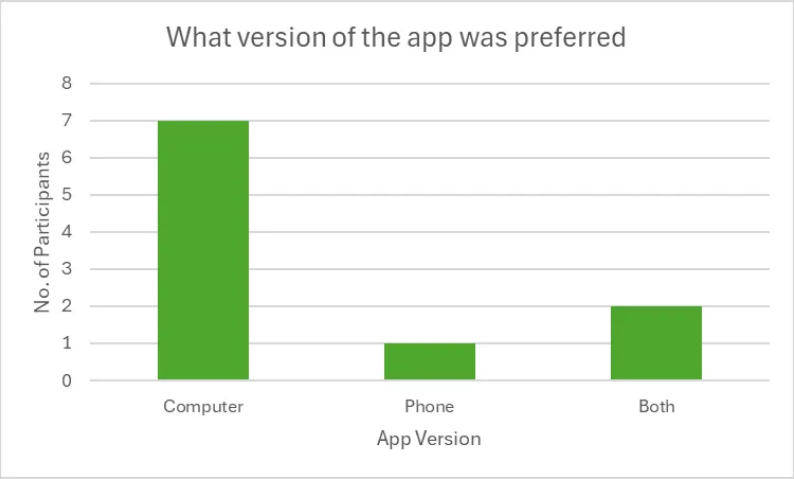


Figure M.8: *Participants' preferences for the computer or phone version of the application.*

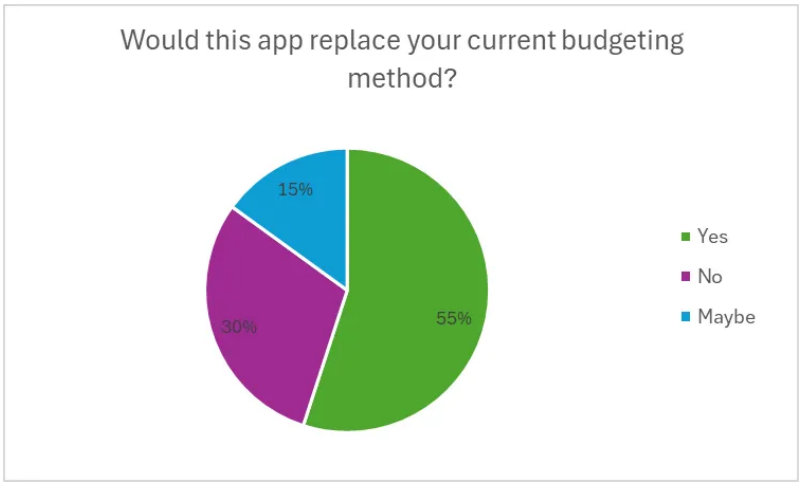


Figure M.9: *Chart showing the proportion of participants who would use the app to manage their finances.*

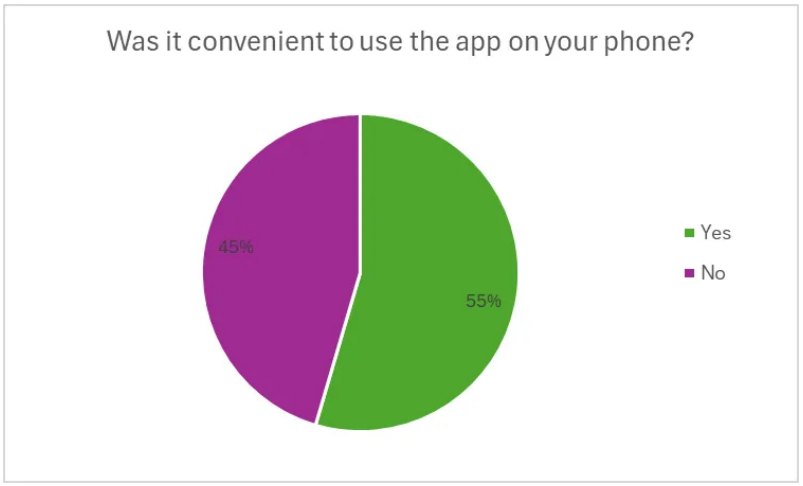


Figure M.10: *Chart showing if participants found using the mobile version of the app convenient.*

N Guides

N.1 Manual

Running the Application

App Architecture

The app has an Angular frontend, found in /client, and a Node.js server, found in /server.

To get the application up and running, a couple of steps are first needed.

Database Setup

You must first set-up a local instance of the database. This can be done by

1. Downloading PgAdmin here.
2. Set up a server and remember the username and password you use during set-up.
3. Create the entities:
    - Set-Up SQL:

```
CREATE DATABASE budgetingapp

CREATE EXTENSION pgcrypto;

CREATE TYPE public.payment_type AS ENUM (
    'credit',
    'debit',
    'cheque',
    'cash',
    'bank_transfer'
);

CREATE TYPE public.repeat_type AS ENUM (
    'daily',
    'weekly',
    'monthly',
    'yearly'
);

CREATE TYPE public.transaction_type AS ENUM (
    'expense',
    'income',
    'savings'
);

CREATE TYPE public.category_type_enum AS ENUM (
    'expense',
    'income',
    'savings'
);
```

- Users:

```
create table public.users (
    user_id serial not null,
    username character varying(30) not null,
    email character varying(255) not null,
    password character varying(255) not null,
    default_currency character varying(3) not null,
    starting_balance numeric(20, 2) not null default 0.00,
    constraint users_pkey primary key (user_id),
    constraint users_email_key unique (email),
    constraint users_username_key unique (username)
) TABLESPACE pg_default;
```

- Category:

```
create table public.category (
    category_id serial not null,
    name character varying(255) not null,
    category_type public.category_type_enum not null,
    constraint category_pkey primary key (category_id)
) TABLESPACE pg_default;
```

- Budget:

```
create table public.budget (
    budget_id serial not null,
    user_id serial not null,
    category_id serial not null,
    amount numeric(20, 2) not null,
    constraint budget_pkey primary key (budget_id),
    constraint unique_user_category unique (user_id, category_id),
    constraint budget_category_id_fkey foreign KEY (category_id)
references category (category_id),
    constraint budget_user_id_fkey foreign KEY (user_id) references
users (user_id) on delete CASCADE
) TABLESPACE pg_default;
```

- Savings_Goal

```
create table public.savings_goal (
    goal_id serial not null,
    user_id serial not null,
```

```
    goal_amount numeric(20, 2) not null default 0.00,
    starting_savings numeric(20, 2) not null default 0.00,
    goal_due_date date null,
    name character varying(255) not null,
    ranking bigint not null,
    constraint savings_goal_pkey primary key (goal_id),
    constraint savings_goal_user_id_fkey foreign KEY (user_id)
references users (user_id) on delete CASCADE
) TABLESPACE pg_default;
```

- Transaction:

```
create table public.transaction (
    transaction_id serial not null,
    user_id serial not null,
    category_id serial not null,
    type public.transaction_type not null,
    name character varying(255) not null,
    transaction_date date not null,
    amount numeric(20, 2) not null,
    shop character varying(255) null,
    payment_method public.payment_type null,
    repeat boolean null,
    repeat_schedule public.repeat_type null,
    end_date date null,
    repeat_group_id uuid null,
    constraint transaction_pkey primary key (transaction_id),
    constraint transaction_category_id_fkey foreign KEY (category_id)
references category (category_id),
    constraint transaction_user_id_fkey foreign KEY (user_id)
references users (user_id) on delete CASCADE
) TABLESPACE pg_default;
```

Setting Up The Env File

In /server create a .env file. You will need to fill in the following environment variables:

```
DB_USER=[your pgadmin username]
DB_PASSWORD=[your pgadmin password]
DB_HOST=[your pgadmin database host]
DB_PORT=[the port your pgadmin database is running on]
DB_DATABASE=budgetingapp
JWT_SECRET=[will come back to this]
NODE_ENV=development
```

To create a JWT secret, run the following in a terminal and copy the output into the JWT_SECRET value:

```
node -e "console.log(require('crypto').randomBytes(64).toString('hex'));"
```

Downloading Dependencies

cd into client and run:

```
npm install
```

Do the same in the server folder.

Running the Application

When in the client folder, run:

```
npm run start
```

When in the server folder, run:

```
npm run start-dev
```

Testing the Application

Frontend

To run the unit tests for the frontend, once in the client folder, run:

```
npm test
```

To get code coverage for frontend tests, run:

```
npm run test-coverage
```

The coverage report can be found at client/coverage/evergreen-budgeting-app/index.html

Server

To run tests for the server, run:

```
npm test
```

This will produce the coverage too, which can be found in server/coverage/lcov-report/index.html.

Bibliography

AG Charts (2025), 'JavaScript Charts | AG Charts', https://www.ag-grid.com/charts. Last accessed: 2025-03-18.

AG Grid (2025), 'AG Grid: High-Performance React Grid, Angular Grid, JavaScript Grid', https://www.ag-grid.com/. Last accessed: 2025-03-18.

Alenazi, M. and Sas, C. (2023), Evaluating budgeting apps: Limited support for budgeting compared to tracking, *in* '36th International BCS Human-Computer Interaction Conference', BCS Learning and Development, pp. 1–12.

Alenazi, M. and Sas, C. (2024), 'Creating and managing transactions and budgets: Analysis of marketplace descriptions and functionality review of budgeting apps', *Interacting with Computers* p. iwae041.

Angular Material (2025), 'Angular Material', https://material.angular.io/.

Apple (2024), 'Enrollment - support - apple developer', https://developer.apple.com/support/enrollment/. Last accessed: 2025-03-15.

AWS (n.d.), 'What is CORS? - Cross-Origin Resource Sharing Explained - AWS', https://aws.amazon.com/what-is/cross-origin-resource-sharing/.

Chacon, S. and Straub, B. (2014), *Pro Git*, 2nd edn, Apress. Chapter 1: Getting Started. Accessed Online At: https://git-scm.com/book/en/v2.

Consumer Financial Protection Bureau (2015), 'Financial well-being: The goal of financial education', https://www.consumerfinance.gov/data-research/research-reports/financial-well-being/. Last accessed: 2025-03-11.

Crunchy Data (n.d.), 'Working with money in postgres | tutorials', https://www.crunchydata.com/developers/playground/working-with-money-in-postgres. Last accessed: 2024-10-21.

Elmasri, R. (2017), *Fundamentals of Database Systems*, seventh, global edn, Pearson. Chapter 14: Basics Dependencies of Functional and Normalization for Relational Databases. Pages 489-518.

Firebase (n.d.), 'Firebase | google's mobile and web app development platform', https://firebase.google.com/. Last accessed: 2025-03-17.

Fowler, M. (2024), 'Continuous integration', https://martinfowler.com/articles/continuousIntegration.html. Last accessed: 2025-03-17.

Ganti, A. (2024), 'What is a budget? plus 11 budgeting myths holding you back', https://www.investopedia.com/terms/b/budget.asp. Last accessed: 2025-03-11.

GeeksForGeeks (2024*a*), 'Classification of software requirements - software engineering', https://www.geeksforgeeks.org/software-engineering-classification-of-software-requirements/. Last accessed: 2025-03-15.

GeeksForGeeks (2024*b*), 'The pros and cons of node.JS in web development', https://www.geeksforgeeks.org/the-pros-and-cons-of-node-js-in-web-development/. Last accessed: 2025-03-15.

GeeksForGeeks (2025), 'How To Implement JWT Authentication in Express App?', https://www.geeksforgeeks.org/how-to-implement-jwt-authentication-in-express-js-app/.

GitHub (n.d.), 'About GitHub pages', https://docs-internal.github.com/en/pages/getting-started-with-github-pages/about-github-pages. Last accessed: 2025-03-17.

GitHub Actions (n.d.), 'Deploying with GitHub actions', https://docs-internal.github.com/_next/data/E-ePrqo9YsAcY1X1qhs66/en/free-pro-team%40latest/actions/use-cases-and-examples/deploying/deploying-with-github-actions.json?versionId=free-pro-team%40latest&productId=actions&restPage=use-cases-and-examples&restPage=deploying&restPage=deploying-with-github-actions. Last accessed: 2025-03-08.

Google (2024), 'Get started with play console - play console help', https://support.google.com/googleplay/android-developer/answer/6112435?hl=en&ref_topic=3450769&sjid=15894356692705775063-EU#zippy=%2Cstep-sign-up-for-a-play-console-developer-account%2Cstep-accept-the-developer-distribution-agreement%2Cstep-pay-registration-fee%2Cstep-choose-a-developer-account-type. Last accessed: 2025-03-15.

Gorshkova, N., Mytareva, L., Perekrestova, L., Glushchenko, A. and Fisher, O. (2015), 'System of family budgeting as a methodological basis for personal accounting and guarantee for growth of financial literacy of the russians', *Mediterranean Journal of Social Sciences* **6**.

Humble, J. and Farley, D. (2010), *Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation*, 1st edn, Addison-Wesley Professional.

Komodo (2022), '9 reasons why react is still popular in 2022', https://www.komododigital.co.uk/insights/9-reasons-why-react-is-still-popular-in-2021/. Last accessed: 2025-03-15.

Lake, R. (2022), 'Budgets: Everything you need to know', https://www.thebalancemoney.com/how-to-make-a-budget-1289587. Last accessed: 2025-03-11.

Mohd Azril Shukri, N. A. M. (2024), 'Personal financial planning and mental wellbeing after covid-19', *SSRN Electronic Journal* .

MoneyPatrol (n.d.), 'Moneypatrol', https://www.moneypatrol.com/moneytalk/budgeting/personal-budget-forecasting-software/. Last accessed: 2025-03-12.

Mozilla (2025), 'Progressive web apps', https://developer.mozilla.org/en-US/docs/Web/Progressive_web_apps/Guides/What_is_a_progressive_web_app. Last accessed: 2025-03-15.

Navickas, M., Gudaitis, T. and Krajnakova, E. (2014), 'Influence of financial literacy on management of personal finances in a young household', *Business: Theory and Practice* **15**(1), 32–40.

npm (2025), 'node-postgres', https://www.npmjs.com/package/pg. Last accessed: 2025-03-15.

Open Banking (n.d.), 'Open banking app store - personal finance tools', https://www.openbanking.org.uk/app-store/?filter-apps=consumer&filter-app-categories%5B%5D=personal-finance-tools&filter-sort=0. Last accessed: 2025-03-12.

OWASP (n.d.), 'HTML5 Security - OWASP Cheat Sheet Series', https://cheatsheetseries.owasp.org/cheatsheets/HTML5_Security_Cheat_Sheet.html#Local_Storage.

Penn Student Registration and Financial Services (2025), 'Popular budgeting strategies', https://srfs.upenn.edu/financial-wellness/browse-topics/budgeting/popular-budgeting-strategies. Last accessed: 2025-03-10.

PostgreSQL (n.d.), 'PostgreSQL: About', https://www.postgresql.org/about/. Last accessed: 2025-03-15.

ProductPlan (n.d.), 'MoSCoW prioritization', https://www.productplan.com/glossary/moscow-prioritization/. Last accessed: 2025-03-15.

Ramsey Solutions (2024*a*), 'How to budget with the cash envelope system', https://www.ramseysolutions.com/budgeting/envelope-system-explained. Last accessed: 2025-03-10.

Ramsey Solutions (2024*b*), 'How to make a budget: Your step-by-step guide', https://www.ramseysolutions.com/budgeting/how-to-make-a-budget. Last accessed: 2025-03-10.

Sakthi S (2024), 'Unlocking angular benefits: 10 key advantages in 2024', https://www.calibraint.com/blog/benefits-of-angular-web-application-development. Last accessed: 2025-03-15.

Salomon, D. (2011), *Foundations of Computer Security*, 1st edn, Springer Publishing Company, Incorporated. Chapter 8: Authentication. Pages 196–202.

Shim, J. K., Siegel, J. G. and Shim, A. I. (2011), *Budgeting Basics and Beyond*, John Wiley & Sons. Google-Books-ID: FsioDgAAQBAJ.

Sols, M. (n.d.), 'Angular vs. react—which one to choose for your web app? | monterail', https://www.monterail.com/blog/angular-vs-react. Last accessed: 2025-03-15.

Sonjaya, Y. (2024), 'Exploring the evolution of budgeting practices from traditional to technology', *Advances in Management & Financial Reporting* **2**(1), 36–45.

Stillwell, S. (2016), Financial education in the UK: A case study of practice, *in* C. Aprea, E. Wuttke, K. Breuer, N. K. Koh, P. Davies, B. Greimel-Fuhrmann and J. S. Lopus, eds, 'International Handbook of Financial Literacy', Springer, pp. 357–368.
URL: *https://doi.org/10.1007/978-981-10-0360-8*$_2$4

Supabase (n.d.), 'Supabase | the open source firebase alternative', https://supabase.com/. Last accessed: 2025-03-17.

Thornhill, J. (2024), 'Our pick of best free budgeting apps for 2024', https://www.forbes.com/uk/advisor/banking/best-budgeting-apps/. Last accessed: 2024-09-24.

Vercel (n.d.), 'Vercel documentation', https://vercel.com/docs. Last accessed: 2025-03-17.

Versal, N., Honchar, I., Balytska, M. and Erastov, V. (2023), 'How do savings and personal budgeting matter on financial literacy and well-being', *Business, Management and Economics Engineering* **21**(2), 190–203.

Vultr (2023), 'How to Securely Store Passwords Using PostgreSQL | Vultr Docs', `https://docs.vultr.com/how-to-securely-store-passwords-using-postgresql`.

W3 Schools (n.d.), 'SQL Injection', `https://www.w3schools.com/sql/sql_injection.asp`.