**GoodBudget**

**Links**

* **GitHub Repository:** *https://github.com/AF1cpt/Goodbudget-budget-app.git*
* **YouTube Demo:** *https://youtube.com/shorts/jdFVaMvBbuI?feature=share*

**Introduction**

**GoodBudget** is a mobile financial management tool that helps users track their expenses, record income, and maintain personal budgets. The app allows users to create an account and then log daily transactions (both expenses and incomes) in an intuitive interface. With GoodBudget, users can monitor their overall financial health at a glance – including total balances and spending across different categories – and develop better budgeting habits. The application’s goal is to empower users to take control of their finances by providing clear insights into where their money goes and how it aligns with their budget goals.

**Contributors**

* **Dishwanelo Tumane – ST10311615**
* **Nicol Black – ST10391534**

**Full Project Team:**

* Nicol Black – ST10391534
* Delcio Tona – ST10284604
* Harvey Skekwa – ST10339985
* Dishwanelo Tumane – ST10311615
* Hopewell Mhlaba – ST10218234

*(Dishwanelo Tumane and Nicol Black are primary contributors for this documentation, with the full group listed above.)*

**Programming Language and Tools**

This project is an **Android application built with Kotlin**. It leverages Android Jetpack libraries and follows modern Android development practices. Key tools and libraries used include:

* **Kotlin** – All development is done in Kotlin, providing concise and safe code for Android.
* **Android Jetpack & SDK** – The app uses components like **Room** for the database layer (providing an abstraction over SQLite) and **AndroidX** libraries for UI and navigation. It also uses **ViewModel** and **LiveData** (or similar architecture components) to manage UI data, ensuring responsive and lifecycle-aware behavior.
* **Material Design Components** – Standard Android UI components and Material design elements are used for a consistent and user-friendly interface (buttons, forms, cards, etc.).
* **MPAndroidChart** – *(Included for future use)* A charting library is added to the project, intended for visualizing financial data (for example, showing spending distribution or balance trends in charts). While not heavily used in the current implementation, it is available for implementing graphs of expenses vs. income or category-wise spending in future updates.

**Colour Scheme**

The **color scheme** of GoodBudget is simple, modern, and focused on clarity:

* **Primary Color – Orange:** The app’s primary accent is an energetic orange hue (for example, #FF9800 and a lighter variant #FFB74D). This orange is used for buttons, highlights, and important UI elements (such as the “Let’s Get Started” button and section titles) to draw attention. It gives the app a vibrant and motivating feel, fitting for a financial app encouraging action.
* **Secondary Accent – Deep Green:** A deep green shade (around #1B5E20) is used as a secondary accent in some places. Green typically represents positive financial health, so this color may highlight successful actions or positive balances. For instance, green could be used to indicate a paid goal or a safe budget status.
* **Neutral Colors – Black and White:** The background is primarily white (#FFFFFF) for a clean look, with black (#000000) and dark gray tones (#444444, #666666, etc.) for text to ensure high contrast and readability. Important text like headings are in solid black, whereas secondary information or placeholders might use lighter grays.
* **Additional Colors – Gray and Red:** Lighter gray shades (e.g., #757575, #888888) appear in hint text or less prominent UI elements (such as form hints or descriptions). A bright red (#FF0000) is reserved for error messages or warnings – for example, indicating an invalid input or an over-budget alert – to immediately catch the user’s attention when something needs fixing.

Overall, the palette balances a **bright orange** for emphasis with **calm neutrals** for backgrounds and text, creating a UI that is engaging but not overwhelming. The colors are chosen to make the important information stand out (like actionable buttons and financial highlights) while keeping the overall look clean and professional.

**Database Storage**

GoodBudget uses a **local database** on the device to store all user data securely, employing Android’s Room library (SQLite under the hood). The app defines several data entities and DAO (Data Access Object) interfaces to manage persistence:

* **User Accounts:** User registration info (first name, surname, username, email, and password) is stored in a **User** table. This allows the login system to verify credentials against locally stored users.
* **Expenses (Purchases):** Expenses are stored in a **Purchase** table with fields such as amount and category. Every time the user logs a new expense, a record is inserted via Room. The app can sum up all expenses to show total spending (“debt”) using a DAO query (for calculating the user’s total expenses).
* **Income:** Income entries are stored in an **Income** table. Similarly, the app tracks all sources of income added by the user and can compute the total income sum.
* **Budget Categories:** Categories (each with a name and a spending limit) are stored in a **Category** table. Users can create budget categories (like Groceries, Rent, etc.) with a defined limit, which is saved in the database for reference when budgeting.

Using Room ensures that all financial data is persisted across app launches and is structured. **All data is currently stored locally (on-device)**, which means it is available offline.

**App Functionality**

GoodBudget provides a range of screens and features to cover the entire budgeting workflow. Below is a detailed breakdown of each screen/feature in the app and what the user can do there:

1. **App Launcher Screen (Splash Screen):** When the app starts, users are greeted with a splash screen that showcases the app’s branding and tagline. This screen includes a friendly illustration (like a piggy bank graphic) and a motivational subtitle (e.g., *“Saving for your future”* along with a brief description encouraging good financial habits). The splash screen sets the tone and purpose of the app. A prominent **“Let’s Get Started”** button is provided on this screen – when tapped, it takes the user into the app’s login flow. (This button immediately leads to the Login Screen, ensuring new or returning users can proceed to authentication.) The launcher screen essentially serves as a welcome and onboarding prompt, engaging the user before they sign in or sign up.
2. **Register Screen:** The Register Screen allows new users to create an account in the app. It presents a form with multiple input fields to collect user details:
   * **First Name and Surname:** so the app can address the user by name.
   * **Username and Email:** unique identifiers for the account. The user can choose a username and provide an email address (which could be used for account recovery in the future).
   * **Password and Confirm Password:** for account security. The form likely includes basic validation, ensuring that the passwords match and possibly that they meet certain complexity criteria.

The screen ensures all required information is provided before submission. There may also be visual cues for password strength or requirements (e.g., indicators if a password includes a number, special character, etc., based on the presence of those criteria in the UI). Once the user fills in the form and taps the **“Sign Up”** button, the app validates the input, creates a new User entry in the local database (via Room), and then navigates the user forward. Typically, after successful registration, the app directs the user to the Login Screen (so they can log in with their new credentials). The Register Screen also likely includes a shortcut for users who might have an account already – for example, a text like “Already have an account? Log in,” which when tapped will switch the user to the Login Screen.

1. **Login Screen:** The Login Screen is where returning users enter their credentials to access their budget data. This screen has fields for **Username/Email** and **Password**. Users type in the account credentials they created, and tap the **“Login”** button. The app then checks the entered credentials against the local user database (using Room queries). If the username/email and password match an existing user record, the login is successful and the user is granted access to the main app (navigating to the Home/Dashboard screen). If the credentials are incorrect, an error message (in red text) is shown to inform the user (for example, “Invalid login, please try again”).

The Login Screen also provides convenient links for account recovery and new users:

* + A **“Forgot Password?”** link: If the user taps this, it will initiate the password recovery process (see below on Forgot Password flow). This lets users reset their password in case they cannot remember it.
  + A **“Sign Up”** prompt: If the user doesn’t have an account yet, a message like “Don’t have an account? Sign up” is displayed. Tapping this link will take the user to the Register Screen to create a new account. This ensures smooth navigation between login and registration for user convenience.

1. **Get Started Screen:** *(Get Started Prompt)* After the splash screen, the app prompts the user to begin using GoodBudget. In practice, this “Get Started” step is embodied by the splash screen’s call-to-action button. When the user taps **“Let’s Get Started,”** it effectively serves as the Get Started Screen, transitioning the user into the app’s sign-in/sign-up process. In other words, there is no lengthy multi-page tutorial – the app opts for a quick start. The idea here is to streamline onboarding: the splash (launcher) screen doubles as an introduction and a get-started prompt. By tapping the button, users proceed directly to log in or register, thus *“getting started”* with using GoodBudget immediately. *(This step is essentially part of the Splash Screen experience, ensuring the user knows how to proceed into the app.)*
2. **Home Screen (Dashboard Overview):** Upon logging in, the user is taken to the Home screen, which serves as the **main dashboard** and overview of their finances. This screen is designed to give a snapshot of the user’s financial status and recent activity. Key elements of the Home Screen include:
   * **Welcome Message:** At the top, the app greets the user by name (e.g., “Welcome, [User] 👋”). This personal touch makes the experience more engaging.
   * **Total Balance (Net Worth):** A prominent section displays the user’s **total balance or net worth**, calculated as total income minus total expenses. This figure updates as the user logs new income or spending. It gives a quick sense of overall financial standing. For example, it might show *“Total Balance: R12,300.00”* in large bold text, representing how much money the user has after accounting for expenses.
   * **Spending Highlights:** The Home screen can include a summary of recent spending patterns. For instance, it might show a line like “You’ve spent R1,200 this week – Top category: Groceries.” (This insight tells the user how much they spent in the current week and what they spent the most on.) This kind of highlight helps users identify where their money is going at a glance.
   * **Budget Summary:** There could be a section listing the user’s budget categories and how they’re doing in each. For example, small cards or lines might show categories like **Groceries, Rent, Transport, etc.**, each with something like “Budget: R2,000, Spent: R1,500” to indicate usage. This would rely on the user having set category budgets in the Budget screen. Even if full budget tracking isn’t implemented yet, the framework is there to present category-wise info.
   * **Quick Actions:** The Home screen may present shortcuts or buttons for common actions (like *“Add Expense”* or *“Add Income”*) so that users can quickly navigate to record a new transaction right from the dashboard. It keeps the most frequent tasks readily accessible.
   * **Expense Name/Description:** A brief description of the expense (e.g., “January Grocery Shopping” or “Electricity Bill”). This helps the user remember what the expense was for. *(In the current app design, this might be a field labeled “Purchase Name.”)*
   * **Amount:** How much was spent. The input is numeric (with a decimal allowed for cents). The user enters the amount in their currency (e.g., *“250.00”* for R250).
   * **Category:** What budget category this expense falls under (such as Groceries, Rent, Transport, etc.). The user either selects from a list of predefined categories or enters the category name. GoodBudget has a concept of categories stored in its database; linking the expense to a category allows the app to count this expense against the relevant budget limit. *(Currently, the app may allow typing the category name directly. Ideally, it would be a dropdown of existing categories or include an “Add new category” option.)*

After filling in the details, the user hits a **“Add Expense”** or **“Save”** button. The app then saves this expense to the local database (in the Purchase table via Room). Upon saving, a confirmation (such as a toast message “Expense Added!”) is shown, and the app updates relevant totals. For example, the total expenses sum will increase, and the total balance/net worth will decrease accordingly. The user is typically returned to a relevant screen (maybe back to the Home/Dashboard or to the list of expenses) after adding the expense. The Add Expenses Screen simplifies the data entry process so users can quickly record spending on the go, ensuring they keep an accurate record in the app.

* + **Income Source/Name:** A short description of the income source (e.g., “August Salary” or “Gift from John”). *(In the UI it might be labeled “Income Name.”)* This field is more for the user’s reference, as the internal data model may or may not store the name explicitly.
  + **Amount:** The amount of money received. The user enters the value (numeric). For example, *“5000.00”* could represent a R5,000 income.

After entering the details, the user taps **“Add Income.”** The app saves the new income record to the Income table in the database. A success message like “Income Added!” confirms the action. Immediately, the total income sum is updated in the app’s calculations, and consequently the user’s total balance/net worth reflects this added income (increasing their available balance). Typically, after saving, the app might return the user to the main dashboard or a list of incomes. The Income Screen ensures that users track all money coming in, not just expenses, giving a complete picture of their finances. It’s particularly useful for users who have multiple income sources or want to log one-off earnings to see how those contribute to their budget.

* + **Total Income vs. Total Expenses:** It clearly lists the cumulative total income and total expenses recorded to date (or for the current budget period). For example: *“Total Income: R15,000”* and *“Total Expenses: R12,000.”* This lets users see both sides of the equation in one place.
  + **Net Balance (Net Worth):** It then highlights the difference between the two – in this example, *“Net Balance: R3,000”*. If the user has more income than expenses, this number is positive (and perhaps styled in a reassuring color like green). If expenses exceed income (negative balance), the number might be highlighted in red to alert the user.
  + **Visual Representation:** Because understanding finances can be easier with visuals, the Dashboard Screen might include a progress bar. For instance, a pie chart showing the ratio of expenses to remaining balance, or a bar chart comparing income vs expenses over time. *(The project included the MPAndroidChart library precisely to enable such visualizations, so a bar chart or pie chart could be integrated here to give the user an at-a-glance understanding of their financial trajectory.)*

In summary, the Dashboard Screen is where a user can go to confirm the **big picture**: “How much have I earned? How much have I spent? And where do I stand now?” It’s a crucial part of budgeting, as it tells the user if they are living within their means. GoodBudget Dashboard Screen consolidates this info, giving users confidence and clarity about their financial status.

* + **Expense Name/Description:** A label for what the expense was (e.g., “Grocery Shopping”).
  + **Amount:** How much was spent, formatted in currency (e.g., “R250.00”). To make it clear, expense amounts might be styled in a standard color or with a “-” sign to denote money spent.
  + **Category:** The budget category of the expense (such as Groceries, Rent, etc.), possibly accompanied by a small icon. *(Icons for common categories like groceries cart, house for rent, car for transport are included in the project resources and could be shown here next to each expense item.)*

1. **Dashboard Screen:** The dashboard Screen revolves around the user’s **budget categories and limits** – essentially the planning side of personal finance. On this screen, users manage and view their predefined budgets for different categories of spending. Key aspects of the Budget Screen include:
   * **List of Budget Categories:** All the categories the user has created (or that come by default) are displayed. For example: Groceries, Rent, Transport, Utilities, Entertainment, etc. Each category entry shows the **budget limit** set for that category. (These are pulled from the Category data stored in the database, which includes a name and a limit amount for each category.)
   * **Used vs. Remaining:** For each category, the screen can show how much has been spent in that category so far (by summing all expenses labeled with that category) versus the set limit. This might be depicted with a progress bar or text, e.g., “Spent R1,500 of R2,000 budget.” If a category is close to the limit or over it, that entry might be highlighted (perhaps in red if over budget).
   * **Add/Edit Categories:** The Main Activity screen provides a way to add a new category or adjust an existing one. For instance, the **“Add Category”** button allowing the user to input a new category name and assign a limit to it. The app then saves this via the CategoryDao (inserting a new Category).

The Main Activity Screen is all about **planning and limits**. It complements the other parts of the app (which track actual numbers) by letting the user set targets or limits on their spending per category. By comparing what one *planned* to spend with what one *actually* spent (visible on this screen), the user can adjust their habits accordingly. For example, if the Entertainment budget is R500 and the user already spent R400, they know to be cautious for the rest of the month. GoodBudget uses this screen to instill discipline and foresight in financial habits, not just after-the-fact tracking.

1. **Bottom Navigation:** The app features a persistent **bottom navigation bar** to enable easy switching between its main sections. This navigation menu appears at the bottom of most screens (especially after login) and typically contains three primary tabs in the current design:
   * **Home:** (represented with a home icon) – Tapping this takes the user to the Home/Dashboard screen (overview of finances). This is the default view when logged in.
   * **Budget/Dashboard:** (represented with a dashboard or stats icon) – This might correspond to the Budget screen or a detailed Dashboard view. In the current implementation, this was labeled “Dashboard,” intended to show budget insights or possibly a breakdown of expenses. It could be the Budget screen described above or a similar overview section.
   * **Notifications:** (represented with a bell or notification icon) – This section is reserved for any alerts or updates. For instance, the app could use this area to show notifications like “Budget limit nearly reached for Groceries” or reminders like “Don’t forget to log your rent payment.” In the existing template, a Notifications tab was included as a placeholder. In the context of GoodBudget, this can be repurposed to show important budget alerts, tips, or even a list of recent activities. *(If the project scope did not implement actual notifications, this tab would currently show a placeholder message such as “No notifications” or similar.)*

The bottom nav bar provides **quick access** to the core functionalities without needing to use the back button excessively. Users can jump between the Home overview, their Budget details, and Notifications with a single tap. The icons and labels for each navigation item make the app intuitive to explore. As the app grows, these sections can be adjusted (for example, one of the tabs could become a direct link to the Expenses list or Profile if those are deemed more important than notifications). For now, it follows a common pattern that users will find familiar in Android apps.

By covering all these screens and components, **GoodBudget** provides a comprehensive budgeting experience. Users can create accounts, log in securely, input their incomes and expenses, set budgets for different categories, and always see where they stand financially. The intuitive layout (with clear sections accessible via bottom navigation and an overflow menu) and the consistent color scheme make the app user-friendly. All data is stored locally with future-ready plans for cloud integration, meaning the app works offline while looking to expand online. This README gives an overview of the project structure and features as implemented, serving as a guide for anyone reviewing or trying out the GoodBudget app.