Github Repo for KashDaddy

https://github.com/ST10191686/KashDaddy.git

Purpose of the application: KashDaddy

KashDaddy is a personal finance management application designed to empower users to achieve financial freedom and security. Its purpose is to provide a user-friendly platform for tracking expenses, managing budgets, and setting financial goals, all while offering personalized financial insights and education. By integrating features such as goal setting with progress tracking, expense logging, and budget management, KashDaddy helps users stay on top of their finances in a structured and informed manner.

Additionally, KashDaddy incorporates biometric authentication for security, offline mode with sync for convenience, and customizable financial categories to suit individual needs. The application goes beyond basic finance tracking by delivering educational resources and tips, ensuring users can also learn how to improve their financial habits, making it a comprehensive tool for personal finance management.

In short, KashDaddy's purpose is to offer an innovative, secure, and educational environment where users can manage their financial life with ease, even without a constant internet connection.

Design Considerations:

1. User-Centric Design:

Intuitive UI: The application must be easy to use, with a clean interface that provides a smooth and user-friendly experience. This includes clear navigation, quick access to key features, and responsive design to ensure usability across different devices and screen sizes【8†source】.

Customization: Allow users to personalize their experience by offering customizable categories, preferences, and financial goals

2. Security:

Biometric Authentication: Implementing fingerprint or facial recognition to ensure users' financial data is protected.

Secure API Communication: Use encryption protocols to secure communication between the app and backend servers.

Data Privacy: Compliance with data protection regulations and secure handling of user data, such as using hashing techniques for storing passwords

3. Offline Mode with Sync:

Seamless Synchronization: One of the innovative features of KashDaddy is the ability to operate offline and sync data when internet access is restored.

4. Performance Optimization:

Fast Loading Times: Ensure the app's performance is optimized with quick loading, efficient database queries, and minimal network request latency

5. Financial Education Integration:

In-App Educational Resources: The design includes integrating articles, tips, and videos within the app to offer users financial education. This content needs to be well-organized and easily accessible

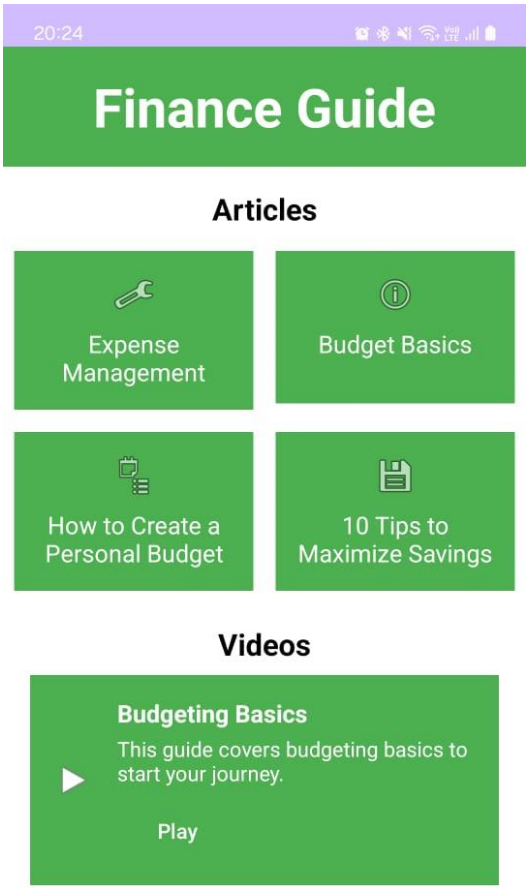
6. Visual Progress Tracking:

Goal Setting with Visual Elements: The application includes a feature for users to set financial goals and track progress using visual indicators like graphs and progress bars. This adds an engaging and motivating element to the app.

User Defined Features :

1.In App Educational Resources :

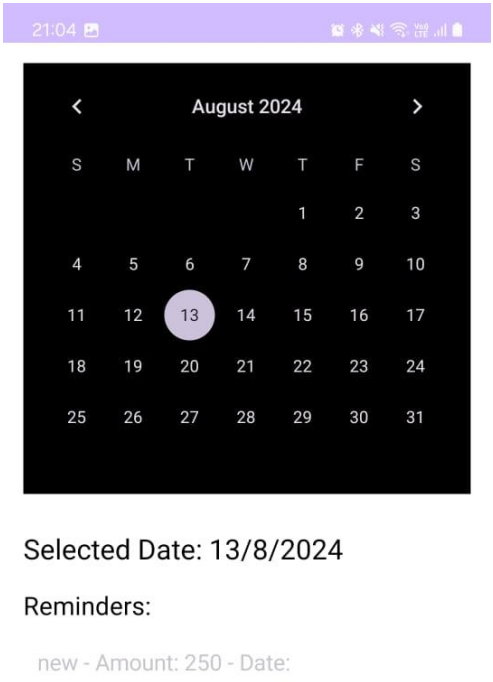
On this screen the user will be able to find and access articles as well as videos that have to do with budgeting. There will be multiple resources available to the user to help them through their budgeting process.



2.In App Calendar for Reminders and Scheduling:

This screen is where the user will be able to select a date from the calendar and in the "upcoming bills" a reminder will be shown to the user notifying them of what payments are

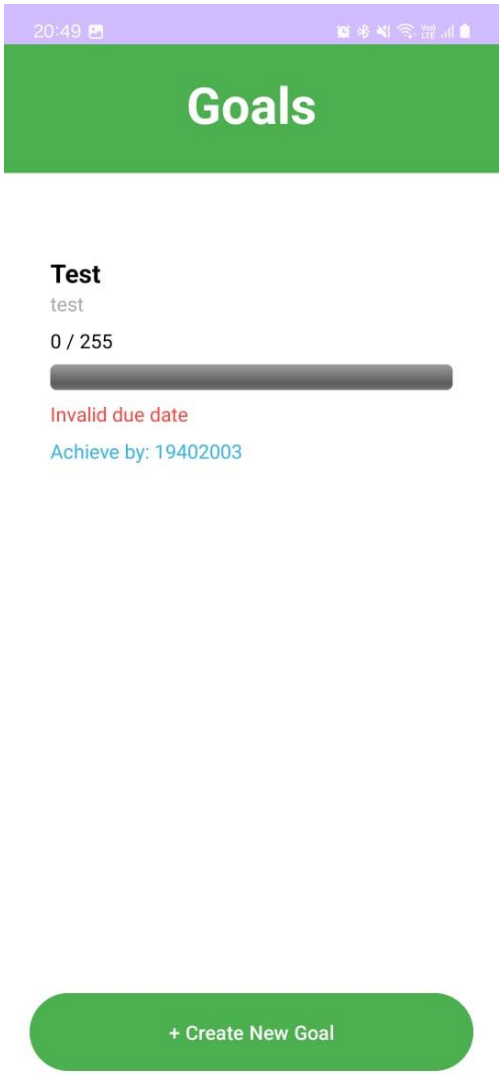
upcoming This feature will help the user keep up with all their payments.



3.Budgeting Goals :

These 2 screens are for the “goals’ feature of the application. The user will be able to view all recent goals they have added, displaying details such as the amount saved along with a progress bar towards the saving goal. There is also the date of when the goal needs to be

achieved and the amount of days remaining. The user can also add new goals by entering in and selecting the relevant details as displayed .



GitHub Utilization

1.Repository Creation and Version Control

- A **GitHub repository** was created to host and manage the source code of the KashDaddy application. This repository served as the central location where all group members could collaborate on the project.
- **Version control** was managed through **Git**, allowing each team member to work on individual features in separate branches. This enabled us to track changes, merge code effectively, and resolve conflicts.
- **Commit history** was maintained for transparency, with detailed commit messages documenting the purpose of each update or fix. This helped us track the progress of the project and revert to previous versions if necessary.

2.Continuous Integration (CI) with GitHub Actions

- **GitHub Actions** was used to automate the testing and building of the application. Every time code was pushed to the repository, GitHub Actions automatically triggered a workflow to test the code and ensure it was functioning as expected.
- The setup ensured that the project was compatible across different environments and not just the local development machines of the team members.

3.Branch Management and Pull Requests

- We implemented a branching strategy where each team member worked on individual features in their own branches. This allowed us to isolate new functionality or fixes without affecting the main codebase.
- **Pull Requests (PRs)** were submitted to merge code back into the main branch, after which the code was reviewed by other team members to ensure quality and correctness. Once approved, the code was merged into the main branch.

GitHub Actions Utilization

1.CI/CD Pipeline Setup

- **GitHub Actions** provided a platform to automate key workflows, such as running tests and building the application on each push or pull request. This allowed us to ensure that the code remained stable and functional after every change.
- We defined a workflow in the .github/workflows/ directory using a **YAML** file. This file outlined the steps for the CI pipeline, such as:
    - **Running tests** (using JUnit for unit testing and Espresso for UI testing)
    - **Building the project** to ensure that it compiles successfully
    - **Code quality checks** using static analysis tools (e.g., **Detekt** for Kotlin)

2.Automated Testing and Build

- GitHub Actions automatically executed our test suite after every push. This included:
    - **Unit tests**: Testing the core functionality of features like user registration, login, and financial goal setting.
    - **UI tests**: Ensuring the user interface behaves correctly and intuitively.
- The automated build ensured that any potential errors were detected early, minimizing

3.Workflow Failures and Fixes

- If any of the tests or builds failed, the team was immediately notified via email. This ensured quick identification and resolution of issues, maintaining the integrity of the code.
- The failure logs provided by GitHub Actions made it easy to trace and fix bugs, improving the overall development workflow.