**Software Requirements Specification for Expensify**

(The Expense Management System)

**Prepared by:**

Jenil Gajera-B.tech(CE)

Nenis Rudani- B.tech(CE)

Krupansu Sorathiya- B.tech(IT)

Date:- 12 july 2024

Software Requirements

Specification

for

Expensify

(The Expense Management Sys

EXPENSE  MANAGER

* ABSTRACT
* Expense trackers are smartphone apps that keep track of a user's spending and provide a detailed picture of their daily, monthly, or annual spending. This study looks into spending tracker apps that are deficient in terms of user experience and data collection.
* The proposed expense tracker will provide a better experience for the user by incorporating information from text messages about financial transactions such as debit and credit of his/her funds. The user can also enter an initial budget for a week, month, or year, and the expense tracker will keep him up to date or alert him if his expenses exceed his weekly, monthly, or yearly budget.
* OBJECTIVE
* The main objective of the Project on Expense Management System is to manage the details of Expense, Category, Type, User, and Registration. It manages all the information about Expense, Payments, Registration, and expenses. The project is totally built at the administrative end and thus only the administrator is guaranteed the access. The purpose of the project is to build an application program to reduce the manual work for managing the Expenses, Category, Payment, and Type. It tracks all the details about the Type, User, and Registration.
* **INTRODUCTION**

1. PURPOSE OF THIS DOCUMENT

The purpose of this document is to provide a detailed description of the requirements for the Expense Tracker application developed using Flutter. It outlines the functional and non-functional requirements, user interface requirements, and other critical aspects needed for the development and deployment of the application.

1. SCOPE OF THIS DOCUMENT

This document covers all the necessary requirements to develop the Expense Tracker application, including:

* To utilize resources in an efficient manner by increasing their productivity through automation.
* The system generates types of information that can be used for various purposes
* It satisfies the user's requirement
* Be easy to understand by the user and operator
* Be easy to operate
* Have a good user interface
* Be expandable
* Delivered on schedule within the budget.

1. BUSINESS CONTEXT

The Expense Manager is targeted at individuals seeking effective tools to manage personal finances. It aims to provide a user-friendly interface, real-time alerts, and comprehensive data visualization tools to aid users in making informed financial decisions. Built using Flutter, the application ensures a consistent and smooth user experience across different platforms.

* **GENERAL DESCRIPTION**
* PRODUCT FUNCTION
* The Expense Management application will provide the following key functions:
* **Transaction Tracking:** Users can add, edit, and delete transactions, categorizing them as expenses or income.
* **Budget Setting:** Users can set budgets for specific categories and time periods (weekly, monthly, yearly).
* **Importing Transactions:** The app can import transaction data from text messages, recognizing financial transactions such as debits and credits.
* **Alerts and Notifications:** Users will receive alerts when their spending exceeds set budgets or when important financial events occur.
* **Data Visualization:** The app will provide visual reports, charts, and graphs to help users understand their spending patterns and financial status.
* **Data Syncing:** The app will sync data with a cloud database when an internet connection is available, ensuring data is accessible across devices.
* USER OBJECTIVES
* **Effective Budget Management:** Set, monitor, and adhere to budgets for various spending categories.
* **Accurate Expense Tracking:** Easily track and categorize all financial transactions to understand spending habits.
* **Timely Alerts:** Receive notifications when budgets are exceeded or significant financial transactions occur.
* **Data Integration:** Seamlessly import and manage transaction data from text messages.
* **Financial Insights:** Access visual reports and analytics to make informed financial decisions.
* **FUNCTIONAL REQUIREMENTS**

1. Dashboard panel

The system shall authenticate the user and then display the panel based on the particular identified user.

2. Add bill

The system shall allow the user to add bill details based on the user's need to track the type of expenses

3. Expense planner

The system should graphically represent the current month figure based on current month expenses and the user's own budget share. on user's

4. Expense tracker

The system should graphically represent the yearly expense numbers in the form of a report

5. Add notes

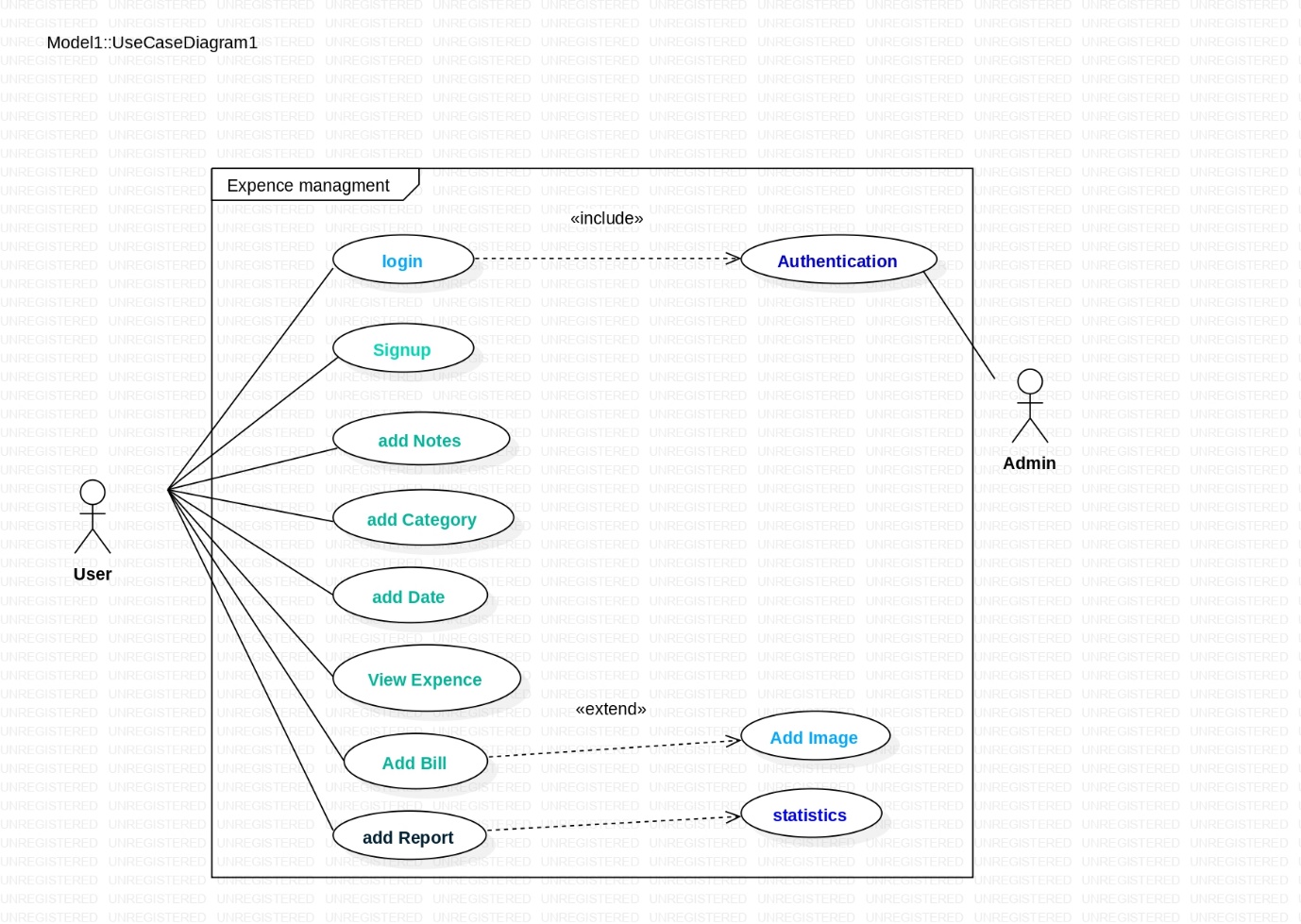
The system shall allow users to add notes to their expenses.

6.Calender

The system shall allow users to add the date to their expenses

7. Category

The system shall allow users to add categories of their expenses



|  |  |  |
| --- | --- | --- |
| **Name** | **Description** | |
| initiating actor | user | |
| Login | to provide identification details and enter the system | |
| Authentication | To validate the details entered by the user | |
| View expenses | Provide the updated log of expenses | |
| Add bill | To add new expenses | |
| Add image | To add an image of the bill (optional) | |
| Track report | Display the record | |
| Statistical analysis | Internal scrutinized evaluation of the data | |
| Add Notes | Can add further more information about the expense | |
| Add Category | To add category where the user is supposed to spend more | |
| Add date | To add the date of the expenditure |

* NOTIFICATION AND ALERTS
* The system shall send notifications to users when their spending exceeds the set budget.
* The system shall send reminders to users for upcoming bills or payments
* The system shall allow users to configure the frequency and type of notifications they receive.
* REPORTING AND ANALYTICS
* The system shall generate reports showing spending patterns over different periods (e.g., daily, weekly, monthly, and yearly).
* The system shall provide visual representations of financial data, such as pie charts, bar graphs, and line charts.
* The system shall allow users to filter reports by categories, periods, and transaction types (income or expense).
* SECURITY AND PRIVACY
* The system shall encrypt all sensitive data, both in transit and at rest.
* The system shall comply with relevant data protection regulations (e.g., GDPR, CCPA).
* The system shall allow users to delete their accounts and all associated data.
* **INTERFACE REQUIREMENTS**
* USER INTERFACES
* LOGIN SCREEN

1. Fields for email and password entry.
2. Options for OAuth login (Google, Facebook) using Flutter packages.
3. Links for password recovery and registration.

* REGISTRATION SCREEN

1. Fields for name, email, and password entry.
2. Terms of service and privacy policy agreement.

* DASHBOARD

1. Overview of total income, expenses, and remaining budget.
2. Quick access to recent transactions and budget summaries.
3. Navigation to other sections (transactions, budgets, reports, settings) using Flutter's navigation system.

* TRANSACTION MANAGEMENT SCREEN

1. View, add, edit, and delete transactions.
2. Fields for amount, date, category, and description.
3. Filtering options by date, category, and type.

* SETTINGS SCREEN

1. Update profile information (name, email, password).
2. Configure notifications, link bank accounts, and manage data sync.

* HARDWARE INTERFACES

1. MOBILE DEVICES(Compatibility with Android and iOS smartphones and tablets.)

* COMMUNICATIONS INTERFACES
* INTERNET CONNECTIVITY

1. Required for data synchronization with the cloud.
2. Offline functionality for adding, editing, and viewing transactions.

* SMS INTEGRATION

1. Import transaction data from text messages using device's SMS service.
2. Parse SMS content for financial transaction details.

* BANKING API INTEGRATION

1. Secure import of transaction data from banking APIs using HTTPS.

* SOFTWARE INTERFACES
* OPERATING SYSTEM

1) Compatibility with Android 6.0+ and iOS 12.0+.

* CLOUD DATABASE

1. Integration with a cloud database (e.g., Firebase Firestore).
2. Secure authentication for database access.

* NOTIFICATION SERVICES

1. Integration with push notification services (e.g., Firebase Cloud Messaging).
2. User-configurable notification preferences.

* **PERFORMANCE REQUIREMENTS**
* Performance Requirements
* **Response Time**
* **Dashboard Load:** Main dashboard loads within 2 seconds.
* **User Input Response:** User actions respond within 200 milliseconds.
* **Transaction Retrieval:** Display transaction data within 1 second.
* Data Synchronization
* **Sync Time:** Sync data with the cloud within 5 seconds.
* **Background Sync:** Data sync does not affect user interactions.
* Scalability
* **Concurrent Users:** Support up to 10,000 users without performance issues.
* **Transaction Management:** Handle up to 10,000 transactions per user efficiently.
* Resource Utilization
* **Memory Usage:** Use no more than 100 MB of RAM during normal operation.
* **CPU Usage:** Minimize CPU usage to avoid battery drain.
* Offline Performance
* **Offline Access:** Allow transaction management offline.
* **Data Availability:** Provide access to the last 30 days of transactions offline.
* Security Performance
* **Data Encryption:** Encrypt sensitive data without affecting performance.
* **Authentication:** Secure authentication methods complete within 2 seconds.
* Notification Performance
* **Notification Delivery:** Deliver push notifications within 5 seconds of the event.
* Usability
* **Frame Rate:** Maintain at least 60 frames per second during interactions.
* **NON-FUNCTIONAL ATTRIBUTES**
* **Security**
* **Data Protection:** All user data, including financial information, shall be encrypted both in transit and at rest using industry-standard encryption protocols.
* **Authentication:** The application shall implement secure authentication mechanisms, including support for biometric authentication (fingerprint, facial recognition) and OAuth login (Google, Facebook).
* **Authorization:** Access to different parts of the application shall be restricted based on user roles and permissions.
* **Data Privacy:** The application shall comply with data privacy regulations (e.g., GDPR) to ensure user data is handled securely and confidentially.
* **Reliability**
* **Uptime:** The application shall maintain a minimum uptime of 99.9%, ensuring availability to users with minimal downtime.
* **Error Handling:** The application shall handle errors gracefully, providing informative error messages and logging errors for further analysis.
* **Data Integrity:** The application shall ensure the integrity of user data through transactional operations and regular backups.
* **Maintainability**
* **Code Modularity:** The application codebase shall be modular, making it easier to update, debug, and enhance specific components without affecting the whole system.
* **Documentation:** Comprehensive documentation shall be provided for the codebase, APIs, and overall system architecture to facilitate the maintenance and onboarding of new developers.
* **Automated Testing:** The application shall include automated tests (unit, integration, UI) to ensure that changes do not introduce new bugs and that the system functions as expected.
* **Portability**
* **Cross-Platform Support:** The application shall be built using Flutter to ensure compatibility with both Android and iOS platforms.
* **Platform Independence:** The application shall abstract platform-specific features, ensuring that minimal changes are required to support additional platforms (e.g., web, desktop).
* **Extensibility**
* **Modular Architecture:** The application shall be designed with a modular architecture, allowing new features and functionalities to be added with minimal impact on existing components.
* **API Integration:** The application shall support integration with third-party APIs, allowing for future extensions such as additional authentication methods, financial data import/export, and more.
* **Reusability**
* **Reusable Components:** The application shall be developed with reusable components, such as UI widgets and utility functions, to facilitate code reuse across different parts of the application.