CSCE 361 Software Engineering

BudgetPal

Kyle Crowder, Jacob Norton, Harrison Hruby, Jon Glodowski (Team 5)

Software Requirements Specification

Document

Version: (1.3) **Date: (04/23/2017)**

Table of Contents

1. Introduction

1.1 Purpose

The purpose of this document is to explain the requirements and features of the BudgetPal budgeting app.

1.2 Scope

BudgetPal will be an Android application that will assist users in organizing and managing the budget and expenses. The app will allow users to create a profile and login to that profile to manage their finances. Once they have entered their profile they will have several options for management. The first will be to set/modify the budget. The second will be to add expenses. The third will be to add financial events to the calendar. The app will not be able to interface with other services and will rely solely on user input to track expenses. BudgetPal is intended to provide individuals with an easier way to track their budget. Often times, people have difficulties managing their finances so we are designing this app to make that easier.

1.3 Definitions, Acronyms, and Abbreviations.

Android - Mobile operating system developed by Google.
Firebase - NoSQL technology developed by Google.
Google Play Store - Application store for android devices.
Navigation Drawer - A menu that slides out from the left side of the screen.
Hash Technique - A technique by which passwords are secured by sending it through a secure hashing algorithm.

1.4 Overview

The rest of this document will include an overall description of the application, the requirements in specific detail, and also an appendix for this document. The overall description will explain the purpose of this application.

2. The Overall Description

The system will have a login system which allows a user to create a new account or login to an existing account. There will also be a Landing Page which will host an overview of the user's information. The user can then navigate with the following actions: Budget Management, Expense History, Event Schedule, Logout Button. Budget Management will let the user set and update their personal budget and add expenses. Expense History will allow the user to view a detailed listing of their previous expenses. Event Schedule will list upcoming financial events and deadlines and allow the user to create future

events. The logout functionality will log the user out of the application.

2.1 Product Perspective

The system is totally self-contained and does not rely on any external services.

2.1.1 System Interfaces

The BudgetPal app will not be an extension of or interface with any other systems on any android devices.

2.1.2 Interfaces

The system will use a GUI to interact with the user using the touchscreen and standard input buttons of the Android. The app will allow for pressing and sliding with the touchscreen as well as pressing the standard input buttons to interact with UI elements.

Future Iteration: The system will use a GUI to interact with the user using the mouse, keyboard, and monitor. The app will allow for point and click as well as typing interaction.

2.1.3 Hardware Interfaces

All Android phones and tablets with Android 6.0 or higher.

Future Iteration: All computers running Linux, Mac, and Windows.

2.1.4 Software Interfaces

This app will be available for download on the Google Play Store and compatible with Android 6.0 and higher.

Future Iteration: The web app will be accessible on Google Chrome, Firefox, Safari, Edge, and Internet Explorer.

2.1.5 Communications Interfaces

The app is self contained and does not require any communications to operate.

2.1.6 Memory Constraints

The system will require a minimum of 512MB of ram be installed on the device. As well

as 4 GB of ROM storage.

2.1.7 Operations

There is only one mode of operation for BudgetPal. All users will access the application in the same way and will all have access to the same features.

2.2 Product Functions

2.2.1. Budget Management

2.2.1.1. Users will be able to set their personal budget, add expenses, and track their personal expense history.

2.2.2. Financial Events

2.2.2.1. Users will be able to add and track financial events.

2.2.3. Web Interface (future)

2.2.3.1. After the main functionality of the application is completed, a separate web interface will be implemented - However, this interface will retain all functionality of the app in addition to advanced statistical analysis of expenses.

2.3 User Characteristics

The intended users are individuals between the ages of 18 and 30 who have incomes below $70,000 annually. They will be familiar with the basic operation of a stock android smartphone.

2.4 Assumptions and Dependencies

The is self contained and runs on the Android operating system and therefore the only dependency is the Android operating system itself.

2.5 Apportioning of Requirements.

Budget Management, Expense Management, and Financial Events will be prioritized for the initial release. For future releases the online account storage and enhanced data analysis will be implemented.

3. Specific Requirements

3.1 External Interfaces

- Touchscreen Inputs and Standard Android Input Buttons
- User will use Touchscreen Inputs and the Standard Input Buttons to specify selections and operate the app.
- The source will come from the touchscreen or the input buttons and the input will be used within the app.

3.2 System Features

3.2.1. Login System

3.2.1.1. Allows users to create an account or login so they can associate their data with an account to help with privacy. (Note: For the first iteration, a user's data will only be accessible from the device on which the created the account.)

3.2.1.2. ***Stimulus/Response sequence***

3.2.1.2.1. A user will choose to create an account. The system will provide them a registration form and will store their information for future login.

3.2.1.2.2. A user will choose to login. The system will provide the login form, check their input with the stored credentials, and either accept or deny the request to login.

3.2.1.3. ***Associated functional requirements***

3.2.1.3.1. Create An Account will require the user to provide their username and create a password.

3.2.1.3.2. Login will require the user to provide their username and password.

3.2.1.3.3. Future Iteration: Logging in will trigger the database update described in section 3.4 for all accessible databases.

3.2.2. Budget Management

3.2.2.1. Will allow users to make and modify a budget.

3.2.2.2. ***Stimulus/Response sequence***

3.2.2.2.1. User will specify how much money they intend to spend over a user defined period of time (They will select from daily, weekly, biweekly, and monthly.) The system will respond by adjusting how it tracks the user's expenses and how it determines statistics.

3.2.2.3. ***Associated functional requirements***

3.2.2.3.1. Make/Modify Budget

3.2.2.3.1.1. Users shall enter both an amount of money and a length of time.

3.2.2.3.1.2. The system will use the amount and time period to manage the user's budget when expenses are added.

3.2.2.3.1.3. Future Iteration: Modifying the budget updates the timestamp for the budget and triggers an update with the cloud database if online.

3.2.3. Expense System

3.2.3.1. This will allow user to add and view expenses.

3.2.3.2. ***Stimulus/Response sequence***

3.2.3.2.1. The user will be able to add an expense which will be stored in the system. The system will keep a running total of the expenses in the time period and will display this sum as well as the difference of the budget and the sum.

3.2.3.2.2. The user will be able to select past expenses to view, edit, and delete them.

3.2.3.3. ***Associated functional requirements***

3.2.3.3.1. User will be able to add an expense by providing a name, specifying the amount, categorizing the expense from a preset set of categories and can include a comment if desired.

3.2.3.3.2. The user will be able to view past expenses with details. The details will include the amount, category, date of the entered expense, and a comment if the user included one in the expense.

3.2.3.3.3. Future Iteration: Modifying expenses updates the timestamp for the expense and triggers an update with the cloud database if online.

3.2.4. Financial Events

3.2.4.1. Users will be able to create and add upcoming financial events and deadlines that the system will store and track for later use.

3.2.4.2. ***Stimulus/Response sequence***

3.2.4.2.1. User must input information which will be used by the system to create, store, and display upcoming events.

3.2.4.2.2. User can edit an event.

3.2.4.3. ***Associated functional requirements***

3.2.4.3.1. The user must input the name and date of the aforementioned. The user may then optionally add a small description of the event if they so choose.

3.2.4.3.2. *Future Iteration: Modifying financial events updates the timestamp for the event and triggers an update with the cloud database if online.*

3.2.5. Landing Page

3.2.5.1. The user will be directed to the Landing Page immediately after successfully logging into the application.

3.2.5.2. ***Stimulus/Response sequence***

3.2.5.2.1. The user, after a successful login, will be directed to the Landing page. The system will then display the user's current remainder of the budget.

3.2.5.2.2. The landing page displays current active events.

3.2.5.3. ***Associated functional requirements***

3.2.5.3.1. No additional requirements are needed for this feature.

3.2.6. Navigation

3.2.6.1. Allows the user to navigate between the various interfaces for managing their finances.

3.2.6.2. ***Stimulus/Response sequence***

3.2.6.2.1. The user will access the menu and choose an interface he or she wants to access. The system will navigate to to the respective interface.

3.2.6.3. ***Associated functional requirements***

3.2.6.3.1. The navigation menu will provide access to the Landing, Add Expense, View Expenses, Add Event, View Events, and Change Budget pages as well as the logout functionality.

3.3 Performance Requirements

3.3.1. Static Numerical Requirements
- 3.3.1.1. Only one terminal is usable with the first iteration.
- 3.3.1.2. The numbers of users on one device is virtually infinite but the number of simultaneous users on one device is one.
- 3.3.1.3. The type of information handled will be numerical, text, and selectors only.
- 3.3.1.4. 99% of the time, usage per one device should remain under 512MB but will not have a strict upper bound.

3.3.2. Dynamic Numerical Requirements
- 3.3.2.1. None

3.3.3. The system should take no longer than 5 seconds to respond to input given from the user for at least 98% of the time.

3.4 Logical Database Requirements

The app will use a Firebase database which will allow for both local and remote database access. The database will be used to store many types of data including: names, usernames, and other numerical and string data. The database stores accounts, and all the data specific to each account such as expenses, budget information, and event/deadline information. The database will need to be frequently used to access budget data and also stored information such as expenses.

In future iterations there will be a database in the cloud that will allow users to access their information by logging in on different devices. When logging in the application will check to see whether the local database is up to date if it exists then update the local and cloud databases if login is successful. The date last modified fields will be used to determine which entries need to be updated by having each database store the most recent version of each entry between the two databases.

Below is the list of the database tables and a description of their contents:

Users - Contains the records for each user profile		
Name	Type	Description
Username	Text	Username of the user
Hashed Password	Text	Hashed and Salted Password Digest
UserKey	String	Automatically generated key (primary key)
Date Last Modified	Datetime	Identifies the datetime the latest update was made to this entry

Budgets - Contains the records for each budget created		
Name	Type	Description
Time Period Code	Integer	Identifies the length of the budget cycle
Reset Code	Integer	Identifies when the cycle resets (meaning varies based on Time period code)
Active	Boolean	Identifies whether this entry is the currently active budget.
Anchor Date	Datetime	Identifies the date of the beginning of the current cycle
Date Created	Datetime	Identifies the date the budget was created
BudgetKey	String	(primary key) Automatically generated key
UserKey	String	(Foreign key) Links entry to the user that created it.
Amount	Float	Identifies the amount in dollars of the budget for each cycle
Date Last Modified	Datetime	Identifies the datetime the latest update was made to this entry

Expenses - Contains the records for every expense entered		
Name	Type	Description
UserKey	String	(Foreign key) Links entry to the user that created it.
BudgetKey	String	(Foreign key) Links entry the the budget that it is associated with.
ExpenseKey	String	(Primary key) Automatically generated key
Amount	Float	Identifies the amount in dollars of the purchases
Description	Text	A text description of the purchase entered by the

		user
Category	Text	A category used to identify the purchase contents chosen by the user
Date Created	Datetime	The date the purchase occurred
Exempt	Boolean	Identifies whether the expense should subtract from the budget.
Date Last Modified	Datetime	Identifies the datetime the latest update was made to this entry

Events - Contains the records for each event created		
Name	Type	Description
Start Date	Datetime	Identifies the start date of the event
End Date	Datetime	Identifies the end date of the event
UserKey	String	(Foreign key) Links entry to the user that created it.
EventKey	String	(Primary key) Automatically generated key
Description	Text	A text description of the event entered by the user
Date Last Modified	Datetime	Identifies the datetime the latest update was made to this entry

3.5 Design Constraints

The application must be able to be interacted with entirely using a touchscreen and the standard input buttons provided on a stock android device.

3.5.1 Standards Compliance

There are no standards which this application must comply with.

3.6 Software System Attributes

3.6.1 Reliability

The system should be able to make accurate calculations 99.99% of the time. It should also be able to store user data for at least one year.

3.6.2 Availability

The app will be available to run at anytime. In the case of a crash or error of the app, the user will be able to close and open the app to restart.

3.6.3 Security

Passwords will be stored using a secure hash technique.

3.6.4 Maintainability

The app will be built in a way that future maintenance and/or new features will be easily implemented. The organizational method used in the code will make the program very usable which will ease maintenance.

3.6.5 Portability

The application will only be expected to run on stock android devices with no expectation of it running on any other devices though nothing will be done to prevent it from running on non-stock implementations of android.