**Expenditure Management Web Application**

**Version 1.0 approved**

**Prepared by**

**SWAPNIL CHAVAN**

**RAHUL A R**

**RAMKUMAR R GHASTI**

**<organization>**

**<date created>**

**Table of Contents**

**Table of Contents ii**

**Revision History ii**

**1. Introduction 1**

1.1 Purpose 1

1.2 Document Conventions 1

1.3 Intended Audience and Reading Suggestions 1

1.4 Product Scope 1

1.5 References 1

**2. Overall Description 2**

2.1 Product Perspective 2

2.2 Product Functions 2

2.3 User Classes and Characteristics 2

2.4 Operating Environment 2

2.5 Design and Implementation Constraints 2

2.6 User Documentation 2

2.7 Assumptions and Dependencies 3

**3. External Interface Requirements 3**

3.1 User Interfaces 3

3.2 Hardware Interfaces 3

3.3 Software Interfaces 3

3.4 Communications Interfaces 3

**4. System Features 4**

4.1 System Feature 1 4

4.2 System Feature 2 (and so on) 4

**5. Other Nonfunctional Requirements 4**

5.1 Performance Requirements 4

5.2 Safety Requirements 5

5.3 Security Requirements 5

5.4 Software Quality Attributes 5

5.5 Business Rules 5

**6. Other Requirements 5**

**Appendix A: Glossary 5**

**Appendix B: Analysis Models 5**

**Appendix C: To Be Determined List 6**

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Date** | **Reason For Changes** | **Version** |
|  |  |  |  |
|  |  |  |  |

# **Introduction**

## **Purpose**

The purpose of this document is to give a detailed description of the requirements for the

Expenditure Management Web Application. It will explain system constraints, interface and

interactions with other external applications. It defines what problems the software

development process will face and the solutions. This document is primarily intended to be

proposed to a customer for its approval and a reference for developing the first version of

the system for the development team and it also serves to ensure that the software is

traceable throughout its software development life cycle.

## **Document Conventions**

|  |  |
| --- | --- |
| User | Enters budget, expenditure details, items in ToDo list, list |
| Admin | Manages database, checks user authenticity, updates |
| App | Abbreviated form of Application |
| Node.Js | Open source server framework |
| MongoDB | Open-source cross-platform document-oriented |
| GPS | Short for Global Positioning System |

## **Intended Audience and Reading Suggestions**

This project is a prototype for the expenditure management system college premises. This has been implemented under the guidance of college professors. This project is useful for the employees to manage their daily and monthly expenses and as well as to the students.

## **Product Scope**

*The application is free to use as a website on a desktop PC or as a web application on a*

*smartphone. It can be used by people of any profession. Children can use it to manage their*

*pocket money and save up to buy something they like. It can be used by house-makers to*

*manage the house expenses. It can be used by people who have jobs or small businesses*

*to manage their daily expenses. The application requires the user to enter their weekly*

*budget and the details of expenditures and it automatically calculates the savings. All*

*information of the users will be securely stored in a database. An administrator will*

*administer the database and ensure the correctness of the data. The administrator can*

*verify the users and also receive feedback for the application and update the application or*

*fix bugs. The application will be able to provide the users their expenditure history and*

*savings history and also notify users if they have enough savings to buy an item from their*

*wish-list.*

## **References**

*[1] IEEE Software Engineering Standards Committee, “IEEE Std 830-1998, IEEE*

*Recommended Practice for Software Requirements Specifications”, October 20, 1998.*

# **Overall Description**

## **Product Perspective**

Expenditure Management App enables users to note down their expenditure cost, time and place whenever they spend money so that they do not forget about that later, view their savings and also review their expenditure transactions. It requires the user however to have a device with him/her with web browser installed and connected to the internet. This means that the users of the system do not need to invest in any other software to get the most out of the software system as most of of the devices come with a browser installed.

## **Product Functions**

The important functions of this software are as follows:

● Authenticate user

● Log down details of expenditure

● Review the transaction history

● Log To Buy details and notify when user can buy

● ToDo list on that day

## **User Classes and Characteristics**

There will be three types of users using the software. Guest user, Signed In user and admin.

The Guest user will have access to login / sign up page and homepage about the software.

In short they will have to login / sign up.

The signed in user will have access to all the features of the software. It will enable him to add and view transactions, view savings, add things to to buy list, add / delete stuff from ToDo list and also to give feedback.

The admin user will have access to User Database. Thus can remove fake users or reported users. He will also have access to feedback database. They will have access to the server to modify the actual software whenever necessary.

## **Operating Environment**

*<Describe the environment in which the software will operate, including the hardware platform, operating system and versions, and any other software components or applications with which it must peacefully coexist.>*

## **Design and Implementation Constraints**

Since the software will be an web application the user must a have web browser installed in the device. The software should work well in all major browsers like Google Chrome, Mozilla Firefox, Microsoft Edge and Internet Explorer, but some functions may not be the same in all as different browsers are coded differently.

Internet Connection is a must since the application fetches data from the database over the internet.

The amount of data our software can hold is constrained by the capacity of the database. Since there will be no credit / debit card linked to the software the user will have to type the transaction information manually.

## **User Documentation //**To be filled

*<List the user documentation components (such as user manuals, on-line help, and tutorials) that will be delivered along with the software. Identify any known user documentation delivery formats or standards.>*

## **Assumptions and Dependencies**

● The system will be running on a windows computer with NodeJs and mlab will be used as our database (MongoDB).

● The user must enter all the transactions he makes regularly.

● The user must have his device connected to the internet while updating/viewing transactions, ToDo list, viewing savings and giving feedback.

● The user must type the information about every expenditure correctly to get the full advantage of the software.

# **External Interface Requirements**

This section specifies the detailed requirements which the system shall meet.

**3.1 External Interfaces**

**3.1.1 User Interfaces**

The user interface will be a GUI implemented using html, css and javascript. It will be responsive so that there will be convenience in using it on any device of user’s choice. The guest user of the application will see the log-in page when he/she opens the application. If the user is not registered he will have to sign up by providing Name, username, password and email id or else login by providing username and password in the username and password fields.

Once the guest user is signed in, he will be given the option related to Expenditure, Savings, ToDo List, ToBuy List, Feedback and account settings. In Expenditure route there will be options for the user to add a new Expenditure which will consist of amount, place, optional description. Next there will be an option to view expenditure history and at last the net amount remaining this month.

In Savings route there will be an option to see savings history and total amount saved.

In ToDo list the logged in user will be able to add or delete ToDo items.

In ToBuy list the logged in user will be able to add or delete ToBuy items. If a item is feasible to buy, the item will have a background colour of light green.

In Feedback route, the user can give number of stars and a form to fill feedback.

In account settings route, the user gets to change budget of the week, add user

description and profile picture.

**3.1.2 Hardware Interfaces**

The web application does not have any direct hardware interfaces.

**3.1.3 Software Interfaces**

To get the location the application communicates with GPS application to get geographical information. MongoDB database will be used to store the information and the server will be a Node.Js server running on a windows system.

**3.1.3 Communication Interfaces**

The communication between the different parts of the system is important since they depend on each other. However, in what way the communication is achieved is not important for the system and is therefore handled by the underlying operating systems.

# **System Features**

*<This template illustrates organizing the functional requirements for the product by system features, the major services provided by the product. You may prefer to organize this section by use case, mode of operation, user class, object class, functional hierarchy, or combinations of these, whatever makes the most logical sense for your product.>*

## **System Feature 1**

*<Don’t really say “System Feature 1.” State the feature name in just a few words.>*

4.1.1 Description and Priority

*<Provide a short description of the feature and indicate whether it is of High, Medium, or Low priority. You could also include specific priority component ratings, such as benefit, penalty, cost, and risk (each rated on a relative scale from a low of 1 to a high of 9).>*

4.1.2 Stimulus/Response Sequences

*<List the sequences of user actions and system responses that stimulate the behavior defined for this feature. These will correspond to the dialog elements associated with use cases.>*

4.1.3 Functional Requirements

*<Itemize the detailed functional requirements associated with this feature. These are the software capabilities that must be present in order for the user to carry out the services provided by the feature, or to execute the use case. Include how the product should respond to anticipated error conditions or invalid inputs. Requirements should be concise, complete, unambiguous, verifiable, and necessary. Use “TBD” as a placeholder to indicate when necessary information is not yet available.>*

*<Each requirement should be uniquely identified with a sequence number or a meaningful tag of some kind.>*

REQ-1:

REQ-2:

## **System Feature 2 (and so on)**

# **Other Nonfunctional Requirements**

## **Performance Requirements**

*Static*

*● Data in database should be updated in 1s.*

*● UI must load within 3s.*

*● At least 10000 users can use the application at the same time.*

*● Login validation will be done within 2s.*

*Dynamic*

*● The software will be should be able to handle at least 1000 requests / second.*

*● Average response time for each response should be 0.5s with worst case being 2s*

*which can occur only 5% of the time.*

## **Safety Requirements**

*<Specify those requirements that are concerned with possible loss, damage, or harm that could result from the use of the product. Define any safeguards or actions that must be taken, as well as actions that must be prevented. Refer to any external policies or regulations that state safety issues that affect the product’s design or use. Define any safety certifications that must be satisfied.>*

## **Security Requirements**

*<Specify any requirements regarding security or privacy issues surrounding use of the product or protection of the data used or created by the product. Define any user identity authentication requirements. Refer to any external policies or regulations containing security issues that affect the product. Define any security or privacy certifications that must be satisfied.>*

## **Software Quality Attributes**

*<Specify any additional quality characteristics for the product that will be important to either the customers or the developers. Some to consider are: adaptability, availability, correctness, flexibility, interoperability, maintainability, portability, reliability, reusability, robustness, testability, and usability. Write these to be specific, quantitative, and verifiable when possible. At the least, clarify the relative preferences for various attributes, such as ease of use over ease of learning.>*

## **Business Rules**

*<List any operating principles about the product, such as which individuals or roles can perform which functions under specific circumstances. These are not functional requirements in themselves, but they may imply certain functional requirements to enforce the rules.>*

# **Other Requirements**

*<Define any other requirements not covered elsewhere in the SRS. This might include database requirements, internationalization requirements, legal requirements, reuse objectives for the project, and so on. Add any new sections that are pertinent to the project.>*

**Appendix A: Glossary**

*● Guest User:*

*Any user who visits the web app.*

*● Registered User:*

*Users how have registered / subscribed the software and have logged in into the app.*

*● Admin User:*

*They are generally developers, maintainers and testers of the application, who have*

*special rights.*

*● Database:*

*A database is a collection of information that is organized so that it can be easily*

*accessed, managed and updated.*

*● MongoDB:*

*MongoDB is an open source database that uses a document-oriented data model.*

*● Collection:*

*In MongoDB a set of similar documents is called a collection.*

*● Documents:*

*Any Object of a specific structure that holds information is called a document. In*

*mongoDB they are BSON objects.*

*● BSON:*

*BSON is a binary representation of JSON documents, though it contains more data*

*types than JSON.*

*● JSON:*

*JavaScript Object Notation or JSON is an open-standard file format that uses*

*human-readable text to transmit data objects consisting of attribute–value pairs and*

*array data types.*

*● Node Js:*

*JavaScript run-time environment for executing JavaScript code server-side.*

**Appendix B: Analysis Models**

*<Optionally, include any pertinent analysis models, such as data flow diagrams, class diagrams, state-transition diagrams, or entity-relationship diagrams*.>

**Appendix C: To Be Determined List**

*Expenditure Management Application intends to ship all the features in its initial release*

*itself. However there is some functionality that could be released at a future date if needed.*

*● The primary functionality that could be held back for future release is the Email*

*Notification feature. The ability to send out emails to users when a transaction is made*

*or notifications of to buy. It is however a feature that the system can be used without and*

*still function in its original scope and requirement.*

*● If it is required that the Email Notification feature is postponed, it is highly recommended*

*that this feature be developed and added on to the system as quickly as physically*

*possible. As it is highly important to the ease of use by the customer and a big selling*

*point which will help keep the system ranked high.*

*● Debit card information gathering of users and automatically adding card usage on*

*expenditure directly to database.*

*● Using OAuth and OAuth2.0 also along with token based authentication system allowing*

*users to login through their Google / Facebook accounts.*