SOFTWARE REQUIREMENTS SPECIFICATION

**For**

**EXPENSE TRACKER**

**Prepared by: -**

P. Vedha Divya Sri

S. Saranya

K. Deepika

# Introduction

## Purpose

The main objective of this document is to illustrate the requirements of the project Expense Tracker. The document gives the detailed description of the both functional and non-functional requirements proposed by the user. The purpose of this project is to provide a friendly environment to maintain the details of expenses and income. The main purpose of this project is to maintain easy circulation system using computers and to provide different reports. This project describes the hardware and software interface requirements using ER diagrams and UML diagrams.

## Document Conventions

* + - Entire document should be justified.
    - Convention for Main title

Font face: Times New Roman Font style: Bold

Font Size: 14

* + - Convention for Sub title

Font face: Times New Roman Font style: Bold

Font Size: 12

* + - Convention for body

Font face: Times New Roman Font Size: 12

## Scope of Development Project

## The scope of the expense tracker app project encompasses the development of a comprehensive financial management solution, designed to assist users in tracking and managing their expenses efficiently. The core features include a user-friendly interface allowing seamless entry, editing, and deletion of expenses, along with robust category management capabilities. The scope emphasizes a thorough testing process, consideration for scalability, compliance with relevant regulations, and documentation creation for user and developer guidance. Continuous maintenance and updates will be implemented to ensure the app remains current and responsive to user needs.

## Definitions, Acronyms and Abbreviations

JAVA -> platform independence SQL-> Structured query Language ER-> Entity Relationship

UML -> Unified Modeling Language

IDE-> Integrated Development Environment SRS-> Software Requirement Specification

## References

* + - Books

 Software Requirements and Specifications: A Lexicon of Practice, Principles and Prejudices (ACM Press) by Michael Jackson

Software Requirements (Microsoft) Second Edition by Karl E. Wiegers

Software Engineering: A Practitioner’s Approach Fifth Edition by Roger S. Pressman

* + - Websites

[**http://www.slideshare.net/**](http://www.slideshare.net/)

[**http://ebookily.net/doc/srs-library-management-system**](http://ebookily.net/doc/srs-library-management-system)

# Overall Descriptions

## Product Perspective

Use Case Diagram of Library Management System

*searches*

1

1 *request*

1

1

1.. \*

\*

search book



1.. \*

check limit

check availability

User 1

issue book

*request renews*

<<include>>

*monitors request*

1

*monitors renew* 1

1

*performs*

*give book*

<<include>>

0.. \*

1.. \*

renew book

verify member

<<include>>

*take book*

1

1 Librarian

Student

0.. \*

1.. \*

*adds\_new\_book*

*perform\_transaction\_updation*

Staff

\*

\*

return\_book

View\_logs

<<extend>>

add\_book

\*

calculate\_fine

update\_record

This is a broad level diagram of the project showing a basic overview. The users can be either staff or student.. This System will provide a search functionality to facilitate the search of resources. This search will be based on various categories viz. book name or the ISBN. Further the library staff personnel can add/update the resources and the resource users from the

system. The users of the system can request issue/renew/return of books for which they would have to follow certain criteria.

## Product Function

Entity Relationship Diagram of Library Management System



The Online Library System provides online real time information about the books available in the library and the user information. The main purpose of this project is to reduce the manual work. This software is capable of managing Book Issues, Returns, Calculating/Managing Fine, Generating various Reports for Record-Keeping according to end user requirements. The Librarian will act as the administrator to control members and manage books. The member’s status of issue/return is maintained in the library database. The member’s details can be fetched by the librarian from the database as and when required. The valid members are also allowed to view their account information.

## User Classes and Characteristics

* Manages personal expenses.
* Records various types of expenses (e.g., groceries, utilities, entertainment).
* Sets budget limits for different expense categories.
* Views personal expense history and trends.
* Manages user accounts and permissions.
* Configures system settings and preferences.
* Monitors system performance and security.
* Resolves technical issues and provides support. Works with financial data for auditing purposes.
* Ensures compliance with accounting standards.
* Manages reconciliation of financial records.
* Generates financial statements and reports.
* Manages expenses for a larger organization.
* Handles expense approval workflows.
* Generates detailed financial reports.
* Monitors and analyzes expense patterns across the organization.
* Tracks business expenses separately.
* Manages expenses related to business operations, supplies, and services.
* Generates reports for tax purposes.
* Monitors cash flow and financial health of the business.

## Operating Environment

## Android apps run on the Android Runtime (ART), which is a virtual machine that executes Java bytecode. The ART is responsible for loading and running the app's code, managing its memory, and providing security sandboxing. When you install an Android app, it is packaged as an APK (Android Package Kit) file. The APK file contains the app's code, resources, and manifest file. The manifest file describes the app, including its name, version, permissions, and components.

## Assumptions and Dependencies

The assumptions are: -

* + The app is intended for individuals who want to track their personal expenses.
  + Users have a basic understanding of using mobile apps and navigating through menus.
  + Users are willing to enter their expense information manually into the app.

**App Functionality Assumptions:**

* + The app allows users to add new expenses with details such as amount, date, category, and description.
  + The app categorizes expenses into groups such as food, transportation, entertainment, etc.
  + The app provides a summary of expenses by category and date range.
  + The app generates reports or charts to visualize expense trends.
  + The app may include additional features such as budget setting, reminders, and synchronization with online services.

**Data Storage Assumptions:**

* + The app stores expense data locally on the user's device in a database or file system.
  + The app may offer options to back up or export expense data to external storage or cloud services.
  + The app ensures the privacy and security of user data.

**Performance Assumptions:**

* + The app is responsive and efficient, with minimal loading times and smooth transitions.
  + The app consumes a reasonable amount of battery power and storage space.
  + The app works reliably on various Android devices and operating system versions.

**Accessibility Assumptions:**

* + The app is designed with accessibility in mind, making it usable for people with disabilities.
  + The app supports features such as voice control, screen reader compatibility, and high-contrast color schemes

**Android SDK:** The Android SDK provides the basic tools and libraries for developing Android apps. It includes the Android Runtime (ART), which is the virtual machine that executes Java bytecode, as well as a set of APIs for accessing features of the Android platform, such as the UI, sensors, and network connectivity.

**Support libraries:** The Android Support libraries provide additional functionality and compatibility across different Android versions. They include libraries for UI components, such as Recycler View and Material Design, as well as libraries for compatibility with older Android versions.

**Room:**  Room is a persistence library that simplifies the process of storing and retrieving data in an Android app. It provides an abstraction layer on top of SQLite, making it easier to manage database schema, perform CRUD operations, and handle data migrations.

**ViewModel:** ViewModel is a class that is designed to store and manage UI-related data in a manner that is independent of the UI lifecycle. This helps to prevent memory leaks and ensures that data is preserved when the UI configuration changes.

**LiveData:** LiveData is an observable data holder that can be observed by UI components. When the data changes, the observers are notified automatically, which simplifies the process of updating the UI in response to data changes.

**Navigation Component:** Navigation Component is a library that simplifies the process of navigating between different screens in an Android app. It provides a declarative way to define the app's navigation flow, and it handles tasks such as saving the current state, transitioning between screens, and managing the back stack.

## 2.5 Requirement

Software Configuration: -

This software package is developed using java as front end which is supported by sun micro system. Microsoft SQL Server as the back end to store the database.

Operating System: Windows NT, windows 98, Windows XP Language: Java Runtime Environment, Net beans 7.0.1 (front end) Database: MS SQL Server (back end)

Hardware Configuration: - Processor: Pentium(R)Dual-core CPU Hard Disk: 40GB

RAM: 256 MB or more

## Data Requirement

**Expense Date:** The date on which the expense occurred. This is a critical piece of information for tracking spending over time and identifying trends.

**Expense Amount:** The amount of the expense. This is typically entered in the currency of the user's choice.

**Expense Category:** The category of the expense. This allows users to classify their expenses and see how much they are spending in different areas, such as food, transportation, and entertainment.

**Expense Merchant:** The name of the merchant or vendor where the expense was made. This can be helpful for tracking recurring expenses and identifying spending patterns.

**Expense Method:** The method of payment used for the expense, such as cash, credit card, or debit card. This can be useful for tracking spending limits and managing payment accounts.

**Expense Notes:** Optional notes about the expense. This can be helpful for providing additional context or justification for the expense.

# External Interface Requirement

## GUI

The graphical user interface is a user-friendly dashboard:

* Summarizing expenses
* An input form for adding expenses with options for categories and tag
* A list view for easy management and graphical reports displaying spending patterns.
* Settings for customization, authentication screens, support options, consistent design elements, responsiveness across devices and feedback mechanisms complete the interface.
* Prioritizing simplicity and clarity is key for a smooth expense-tracking experience.

# System Features

System features for an expense tracker app include expense logging, category management, dashboard summaries, detailed reports, budgeting tools, receipt capture, cross-device synchronization, reminders, secure data storage, integration with financial services, search/filter options and export/sharing capabilities. These features empower users to efficiently track, manage and analyze their expenses.

# Other Non-functional Requirements

## Performance Requirement

* Responsiveness: The app should load quickly and respond promptly to user interactions, ensuring a seamless experience.
* Scalability: It must handle increased data without compromising speed or performance, accommodating a growing number of users.
* Offline functionality: Partial or complete functionality when offline, syncing data once the connection is restored.
* Minimal Resource Consumption: Efficiently utilize device resources(CPU, memory, battery) to optimize performance.

## Safety Requirement

* Data security: Implement robust measures to secure user data, including encryption techniques and secure data storage.
* Error handling: Gracefully manage errors or exceptions to prevent data loss or system crashes.
* Backup and recovery: Provide mechanisms for data backup and recovery to prevent data loss in case of system failure.

## Security Requirement

* + - System will use secured database
    - Normal users can just read information but they cannot edit or modify anything except their personal and some other information.
    - System will have different types of users and every user has access constraints
    - Proper user authentication should be provided
    - No one should be able to hack users’ password
    - There should be separate accounts for admin and members such that no member can access the database and only admin has the rights to update the database.

## Requirement attributes

* + - There may be multiple admins creating the project, all of them will have the right to create changes to the system. But the members or other users cannot do changes
    - The project should be open source
    - The Quality of the database is maintained in such a way so that it can be very user friendly to all the users of the database
    - The user be able to easily download and install the system

## Business Rules

Business rules for an expense tracker app revolve around categorizing expenses, setting budget alerts, supporting multiple currencies, enabling recipt attachments, facilitating multi-user access with varying permissions, automating transaction tracking, maintaining an audit trail, offering detailed reports

## User Requirement

# Other Requirements

## Data and Category Requirement

There are different categories of users namely teaching staff, Librarian, Admin, students etc. Depending upon the category of user the access rights are decided. It means if the user is an administrator, then he can be able to modify the data, delete, append etc. All other users except the Librarian only have the rights to retrieve the information about database. Similarly, there will be different categories of books available. According to the categories of books their relevant data should be displayed. The categories and the data related to each category should be coded in the particular format.

## Appendix

## Glossary

The following are the list of conventions and acronyms used in this document and the project as well:

* + - Administrator: A login id representing a user with user administration privileges to the software
    - User: A general login id assigned to most users
    - Client: Intended users for the software
    - SQL: Structured Query Language; used to retrieve information from a database
    - SQL Server: A server used to store data in an organized format
    - Layer: Represents a section of the project
    - User Interface Layer: The section of the assignment referring to what the user interacts with directly
    - Application Logic Layer: The section of the assignment referring to the Web Server. This is where all computations are completed
    - Data Storage Layer: The section of the assignment referring to where all data is recorded
    - Use Case: A broad level diagram of the project showing a basic overview
    - Class diagram: It is a type of static structure diagram that describes the structure of a system by showing the system’s cases, their attributes, and the relationships between the classes
    - Interface: Something used to communicate across different mediums
    - Unique Key: Used to differentiate entries in a database

## Class Diagram

A class is an abstract, user-defined description of a type of data. It identifies the attributes of the data and the operations that can be performed on instances (i.e., objects) of the data. A class of data has a name, a set of attributes that describes its characteristics, and a set of operations that can be performed on the objects of that class. The classes’ structure and their relationships to each other frozen in time represent the static model. In this project there are certain main classes

which are related to other classes required for their working. There are different kinds of relationships between the classes as shown in the diagram like normal association, aggregation, and generalization. The relationships are depicted using a role name and multiplicities. Here ‘Librarian’, ‘Member’ and ‘Books’ are the most important classes which are related to other classes.

