Software Requirements Specification

for

Calendar Management App

Version 1.0 approved

Prepared by Keshav Kumar (PES2201800207),

Shivam Singh Rawat (PES2201800095),

Jeetraj Choudhury (PES2201800349)

PES University – Electronic City Campus

1/2/21

Table of Contents

Revision History

Name	Date	Reason For Changes	Version

1. Introduction

1.1 Purpose

The purpose of this project is to build a calendar management system. Calendars are useful tools for keeping track of upcoming meetings, deadlines, and milestones. They can help us visualize our schedule and remind us of important events, such as holidays and vacation time.

1.2 Intended Audience

This project is a prototype for a calendar management system and it is restricted within the college premises. This has been implemented under the guidance of college professors.

1.3.1 Developer

Developers will be in charge of analysis, design, implementation, integration, and testing of all artifacts produced for the application. Developers will reference this document to make sure that they are complying to all functional and nonfunctional requirements outlined in this SRS and to make sure that the vision of the application is being fulfilled.

1.3.2 Supervisor (Mrs. Aisha Begum)

Mrs. Aisha will use this document to obtain the intended vision for this application. Suggested Readings:

1. *Section 2: Overall Description*
2. *Section 3: Use Cases*

The primary target of this application will be people who want to manage their day-to-day busy lives and structure it in an efficient manner.

1.3 Product Scope

The purpose of this project is to place all tasks for an individual's multiple projects or tasks and appointments in one location. Additionally the user must be able to see whether the year is leap or not and the calendar for any month of any year.

Above all, we hope to provide a nice user experience.

1.4 References

1.Google Calendar (for feature inspiration)
2.SRS Format (for elaborate explanation of each heading)
3.Use case diagram Format (for drawing Use case diagram)

2. Overall Description

2.1 Product Perspective

Calendars are one of the basic necessities for anyone. Integrating an event management system with a calendar is easily one of the best applications for a calendar management system. There are many existing event management systems and calendars but this project integrates these features in a single location. This could further be bundled with other products targeting office management somewhat like microsoft office.

2.2 Product Functions

The major functions for this calendar management system are:

- *Ability to add and manage events/tasks.*
- *Ability to check whether a year is a leap or not.*
- *Ability to view calendar for a given month/year.*
- *Ability to differentiate between working and non-working days.*

2.3 User Classes and Characteristics

User

This product can be used by almost anyone who needs a calendar and event/task management system. They can use it for viewing their calendar and manage tasks accordingly. Most users just need to be signed in with their accounts to get their details synced up and saved. User data can be accessed and modified by the user only.

Developers

Developers will be able to check for any errors or warnings and solve them. Developers can add new features or even remove an existing feature if found redundant or obsolete. Developers have the highest security privilege for the core working of the project but they cannot access user data for privacy purposes.

2.4 Operating Environment

Minimum Hardware Requirements:

- *Intel Pentium Processor*
- *100MB of free hard disk space*
- *512 MB - 1GB of RAM*

Minimum Software Requirements:

- *Windows 7 or above*
- *G++ (if compiling), since project is in C++ language*
- *Command Prompt or relevant shell*

2.5 Design and Implementation Constraints

The main issues for developers can be to scale according to the users. Since, there is no limit on users so the software and the hardware should be able to keep up with it. Also, there is no limit to adding events for each user, but there should be restrictions present that developers should know to avoid any warnings and errors which can render the whole system completely useless. One of the challenges for developers can be providing support for different platforms or devices. There should not be any issues when the user accesses the system from a different platform or device if they are supported.

2.6 Assumptions and Dependencies

Let us assume there is a certain user signed in and using the system to manage events and is faced with the following situation:

- *The user scheduled events/tasks on a certain device but somehow, they lose access to that device. The user should be able to recover their data for their new device and that also seamlessly without any technical issues.*
- *Use of a third-party user authentication service will improve the service and reduce coding time and effort.*
- *Project will not require an official administrator or moderator, they can be created in the machine in which they are executed.*

These are some of the situations where the developers need to focus and provide a solution through various methods/techniques. Similar situations like this should be considered before deployment.

3. External Interface Requirements

3.1 User Interfaces

- *Front-end software: Terminal*
- *Back-end software: OOP oriented software*

The first interface that the user would see would ask him whether he is a regular customer or an admin. The next interface would ask the customer to provide his login credentials. If successfully logged in, The customer would then have access to the application.He will then be presented a menu with various options to check leap year, print the calendar or mark appointments.

3.2 Software Interfaces

Following are the software used for the system:

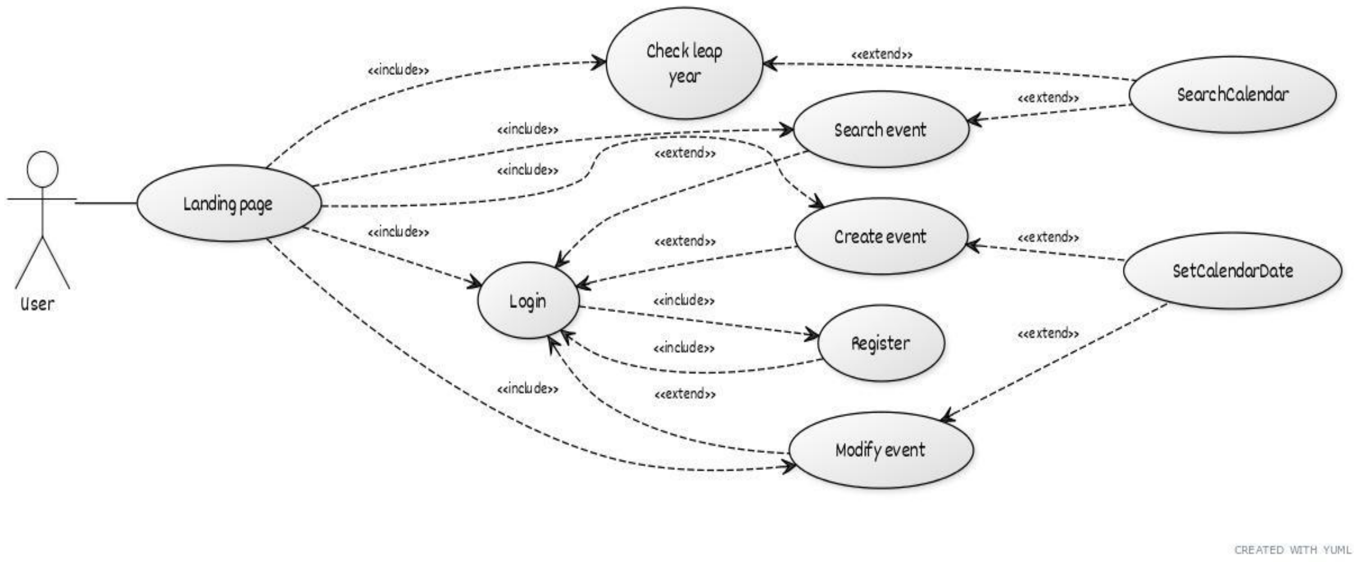
- *OS: Windows is chosen for best support and user experience*
- *Database: Basic file storage methods present in OOP language.*
- *Programming Language: Any Object-Oriented Programming language.*

3.3 Communications Interfaces

This project should support all types of compilers and the operating system it is meant for running seamlessly without any issues.
When the user marks an appointment he can see the event on a calendar printed on the user interface and similarly for other features as well. Any exceptions generated like wrong input will be handled by the code in a corrective/suggestive manner.

4. Analysis Models

Overall structure:



5. System Features

5.1 Check If Leap Year

5.1.1 Description and Priority

This allows the user to check if any random year is a leap year or not . This is a low priority feature.

5.1.2 Stimulus/Response Sequences

Can be triggered by running the .exe file and typing the marked option then input the year.

5.1.3 Functional Requirements

If the user enters a year that does not exist or is in wrong format , they will be prompted with a message to correct their input if they wish to proceed further.

5.2 Print Calendar

5.1.1 Description and Priority

This allows the user to print a calendar of any random year and month whether it's leap or not and check the appointments for that month. This is a high priority feature.

5.1.2 Stimulus/Response Sequences

Can be triggered by running the .exe file and typing the marked option after logging in, then input the month and year for which the calendar is needed.

5.1.3 Functional Requirements

If the user enters a year or month that does not exist or is in wrong format , they will be prompted with a message to correct their input if they wish to proceed further.

5.3 Add Appointment

5.1.1 Description and Priority

This allows the user to mark an appointment for any random year,month and day along with the description of the appointment. This is a high priority feature.

5.1.2 Stimulus/Response Sequences

Can be triggered by running the .exe file and typing the marked option after logging in, then input the day, month and year on which the appointment exists.

5.1.3 Functional Requirements

If the user enters a year ,month or day that does not exist or is in wrong format , they will be prompted with a message to correct their input if they wish to proceed further. Weekends and appointments are displayed in different colors so that they can be differentiated easily.

5.3 Login and security

5.1.1 Description and Priority

This allows the user to differentiate themselves as either the administrator/developer or a normal user. This is a high priority feature.

5.1.2 Stimulus/Response Sequences

Can be triggered by running the .exe file and typing the marked option after logging in, then input their ID and password.

5.1.3 Functional Requirements

If the user enters the wrong id or password they will be prompted to retry again and if they fail multiple times then they will be barred from entering it again.

6. Other Nonfunctional Requirements

6.1 Performance Requirements

This system must only be scaled considering the performance aspect. It should use less memory and should be easy to access and work around even for a naive user. Memory management should be done wisely so that none of the memory is blocked unnecessarily.

6.2 Safety Requirements

Safety of the system and user data is of utmost importance since any lapse in this aspect will not only lead to a compromised system but also loss of property and finance for this reason login credentials are necessary and will be aptly implemented. This will not only grow and garner the trust from users but also maintain the integrity of such applications further developed.

6.3 Security Requirements

User Account ID and Password protection occurs at the first level within the Calendar management application. To access their personal calendar, users are required to enter an Account ID and password. Without these, the user can only check if a year is leap or not.

6.4 Software Quality Attributes

This project can be reused as part of a bigger suite of applications, the maximum part of the code will be portable to ubuntu,mac and windows .Since the software will be repeatedly tested before production it will be guaranteed that it is robust and usable. Maintenance could be performed easily since most of the code will be commented appropriately and written in a conventional manner.

6.5 Business Rules

Normal users can only have the privilege to work with their appointments and cannot view or edit other user's data.
Administrators can view and edit data throughout the application, but this may change accordingly to comply with privacy concerns.

This can be implemented in a variety of domains like corporate offices, transport departments, education ,etc. Such applications already find a large use in day-to-day lives encompassing various phases of human lives.

7. Other Requirements

A file will be created containing the credentials for different accounts, it is crucial that the permissions for this file do not change at any point of time since this will be re-read by the program for authentication purposes. For a seamless performance requirements stated above should be fulfilled.

Appendix A: Glossary

User: *A person who only uses the software for its basic purpose and can only modify and work on the entities that they created and that too only through the GUI.*

Developer: *A person who has access to the application code and can modify it according to their will.*

Administrator: *A person who has the rights to delete and alter user entries through the GUI.*

Login: *Basic authentication needed to use full services*

Register: *Account registration required for a new user to use full services*

G++: *Compiler necessary for compiling C++ code and will only be required by developers and maintainers.*

C++: *A general purpose object oriented programming language.*

OOP: *Refers to object oriented programming that is being used here.*

Terminal: *A command-line tool that can run shell scripts.*

Appendix B: Field Layouts

Tables containing field layouts and properties/attributes and report requirements.

Information required for login / registration.

Field	Length	Data Type	Description	Is Mandatory
Username	6+	Text	unique username for login	Y/N
Password	6+	Alphanumeric with special symbols		Y/N

Information required for checking leap year.

Year	4	Date	year that is checked	Y

Information required for booking appointment

Date	2	Date	Date for booking appointment	Y/N
Day	10	Date	Day of booking appointment	Y/N
Event Description	25	Alphanumeric	brief description of event	Y

Appendix C: Requirement Traceability Matrix

Sl. No	Requirement ID	Brief Description of Requirement	Architecture Reference	Design Reference	Code File Reference	Test Case ID	System Test Case ID
1	Req 1	Register through landing					
2	Req 2	Login as user or admin					
3	Req 3	Enter year for checking leap					
4	Req 4	Enter date to create/modify/search events					
5	Req 5	Enter description for event					

Rest of the cells will be filled as the project progresses.