Software Requirements Specification Template

CptS 322—Software Engineering

9 February 2005

The following annotated template shall be used to complete the Software Requirements Specification (SRS) assignment of WSU-TC CptS 322. The instructor must approve any modifications to the overall structure of this document.

**Template Usage:**

Text contained within angle brackets (‘<’, ‘>’) shall be replaced by your project-specific information and/or details. For example, <Project Name> will be replaced with either ‘Smart Home’ or ‘Sensor Network’.

Italicized text is included to briefly annotate the purpose of each section within this template. This text should not appear in the final version of your submitted SRS.

This cover page is not a part of the final template and should be removed before your SRS is submitted.

**Acknowledgements:**

Sections of this document are based upon the IEEE Guide to Software Requirements Specification (ANSI/IEEE Std. 830-1984). The SRS templates of Dr. Orest Pilskalns (WSU, Vancover) and Jack Hagemeister (WSU, Pullman) have also be used as guides in developing this template for the WSU-TC Spring 2005 CptS 322 course.

MEETME

Software Requirements Specification

2

14-11-2015

Jeme John

Nitesh Thalli

Tharun Niranjan

Vishal Johri

Yousaf Hussain

Prepared for

SER 515—Software Enterprise

Instructor: DR. Kevin Gary.

Fall 2015

# Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Description** | **Author** | **Comments** |
| 14-11-15 | Version 1 | Jeme John | Final Draft |

**Table of Contents**

Revision History ii

Document Approval ii

1. Introduction 1

1.1 Purpose 1

1.2 Scope 1

1.3 References 1

1.4 Overview 1

2. General Description 2

2.1 Product Perspective 2

2.2 Product Functions 2

2.3 User Characteristics 2

2.4 General Constraints 2

2.5 Assumptions and Dependencies 2

3. Specific Requirements 2

3.1 External Interface Requirements 3

3.1.1 User Interfaces 3

3.1.2 Hardware Interfaces 3

3.1.3 Software Interfaces 3

3.1.4 Communications Interfaces 3

3.2 Functional Requirements 3

3.2.1 Functional Requirement 1 3

3.2.2 Functional Requirement 2 3

3.2.3 Functional Requirement 3 3

3.2.4 Functional Requirement 4 4

3.2.5 Functional Requirement 5 4

3.2.6 Functional Requirement 6 4

3.2.7 Functional Requirement 7 4

3.2.8 Functional Requirement 8 5

3.2.9 Functional Requirement 9 5

3.2.10 Functional Requirement 10 5

3.3 Use Cases 5

3.3.1 Use Case 1 5

3.3.2 Use Case 2 6

3.5 Non-Functional Requirements 7

3.5.1 Performance 7

3.5.2 Reliability 8

3.5.3 Availability 8

3.5.4 Security 8

3.5.5 Maintainability 8

3.5.6 Portability 8

3.6 Design Constraints 8

3.7 Logical Database Requirements 8

4. Analysis Models 9

4.1 Data Models 10

4.2 State-Transition Diagrams (STD) 10

5. Change Management Process 10

# 1. Introduction

## 1.1 Purpose

*MeetMe is an online meeting/event scheduler application developed with the intention of scheduling meetings easily, by handling and solving the difficulties and constraints encountered in other meeting schedulers applications like Doodle. The system suggests the meeting time slots, with maximum availability, from the host's synced calendar, without compromising on the security and privacy of the host or participants.*

## 1.2 Scope

*(1) The system under consideration is a web application, which has been designed to work on major browser and operating environments without issues.*

*(2)The purpose of the project is to develop a user friendly meeting scheduler for easy scheduling*

*of meetings and events.*

*(3) MeetMe allows both members and guests to schedule meetings easily without having to worry about the availability of participants. It just requires the host to enter some primary information, such as title, dates, participant ids, the rest is taken care of by the system. The system also takes into consideration the availability of the participants before arriving at a final meeting time, by asking them to attend a poll.*

## 1.3 References

*This subsection should:*

*(1) Provide a complete list of all documents referenced elsewhere in the SRS, or in a separate, specified document.*

*(2) Identify each document by title, report number - if applicable - date, and publishing organization.*

*(3) Specify the sources from which the references can be obtained.*

*This information may be provided by reference to an appendix or to another document.*

## 1.4 Overview

*(1) The purpose of this specification is to briefly describe the requirements of MeetMe- Scheduler and thereby provide a documentation regarding the functionality of each component and the purpose and qualities (complete, consistent) of each requirement, for traceability purposes.*

*(2) The document includes functional and nonfunctional requirements. And encloses diagrammatic representation of the analytical models.*

# 2. General Description

*The requirements stated below were finalized from the project description and were finalized after several rounds of group meetings and brainstorming sessions. The correctness of the design was later verified with the sponsor of the project though email communication.*

## 2.1 Product Perspective

*The product has been designed with the intention to include maximum functionalities, which, similar products in market have been missing out on and thereby improve the user experience and ease while scheduling meetings.*

## 2.2 Product Functions

*The system allows a logged in user or a guest to send meeting invites to members, where the meeting time is selected by the host and the participants from the suggestion provided by the system. The system takes care of the concept of user security by maintaining privacy of user calendar information. The participants are also given a chance to voice their selection from the suggested slots, all of which is combined at the final stage to send meeting invites.*

## 2.3 User Characteristics

The user of the system is expected to be willing to share his calendar information with the system. Even though the system works best when the user has been registered or is willing to do so, it also allows guest users to schedule meetings.

## 2.4 General Constraints

*(1)Design Constraints: Meteor Framework has been used for development of the system. The frontend code is in JavaScript and the backend uses NodeJs, with backend support of Mongo DB for data storage.*

*(2) Implementation Constraints: The current implementation of the system requires the user scheduling the meeting to have an up to date google calendar synced with the system, for efficient scheduling of the meeting.*

*(3) Operating Environment Constraints: This system does not have any operating environment constraints as such and has been tested to work fine on windows, OSX systems, and all major web browsers (Google Chrome, Firefox, Safari, IE).*

## 2.5 Assumptions and Dependencies

*(1)The system depends on Google API for calendar data. The system makes API calls to Google calendar for retrieving the time schedule information and updating the calendar with meetings setup though the application.*

*(2) The system also makes use of third party Mail Server for sending meeting requests and reminders to the participants*

# 3. Specific Requirements

*The requirements attached to this document are mainly derived from the project description on website (http://score-contest.org/2016/projects/meetme.php) and from the conversation with the sponsor.*

In this final version, each requirement abides to the IEEE requirements quality specification (Correct, Traceable, Unambiguous, Verifiable, Complete, Consistent, and Uniquely Identifiable with proper numbering).

## 3.1 External Interface Requirements

### 3.1.1 User Interfaces

### 3.1.2 Hardware Interfaces

### 3.1.3 Software Interfaces

### 3.1.4 Communications Interfaces

## 3.2 Functional Requirements

### 3.2.1 Functional Requirement 1

*FR 1. Implementation of login and signup functionality.*

*Acceptance Tests:*

*(1) An existing user should be able to login to his account, on entering the correct email address and password.*

*(2) A new user should be able to sign up to the system and later be able to login with the newly created account.*

### 3.2.2 Functional Requirement 2

*FR 2. The user should have an option to delete his account whenever he wants to.*

*Acceptance Tests:*

*(1) All the user related information (profile, upcoming meetings, calendar synchronization) should be removed from system database.*

### 3.2.3 Functional Requirement 3

*FR 3. Implementation of the profile page for the user, with user profile information (pending meeting invites, scheduled meeting etc).*

*Acceptance Tests:*

*(1)The profile page should have the user account related information displayed on it. The pending meeting section should show the meeting yet to be finalized on and the upcoming section should be a section for already finalized meeting with information about the number of confirmed participants.*

### 3.2.4 Functional Requirement 4

*FR 4. Implementation of meeting scheduling section on the profile page, which allows the user to request slot suggestion on entering the required information (title, description, duration, date, venue, and participant).*

*Acceptance Tests:*

*(1) The profile page should have valid input fields for title, description, meeting dates, venue, duration, participants email. All the fields should be made mandatory, with validation on date (should not be a past date), email ids.*

### 3.2.5 Functional Requirement 5

*FR 5. The user should be given a choice to synchronize one of his calendar application with system either during sign up or after logging in.*

*Acceptance Tests:*

*(1) The user while creating an account should be successfully able to sync his google calendar with his Meetme account.*

*(2) If the user doesn’t already have a synced calendar or wishes to change the calendar he should be able to do it on the profile page after logging in.*

### 3.2.6 Functional Requirement 6

*FR 6. Implementation of guest user functionality, which includes asking for email address before submitting and storing of the guest user activity as well for tracking purposes.*

*Acceptance Tests:*

*(1) When the user chooses to proceed as the guest user, he should be asked to enter his email address that has an associated calendar and the system then proceeds to fetch the calendar data corresponding to the email address. This should be a mandatory field with proper email validation.*

*(2) The meeting scheduled by a guest user and its status should be stored on the database. And this should be of the form of a state machine (where the meeting states can vary from Invalid Email -> Pending -> Scheduled -> Finished/Cancelled)*

### 3.2.7 Functional Requirement 7

*FR 7. Implementation of meeting slots suggestion logic, by extracting information from the synchronized calendar.*

*Acceptance Tests:*

*(1) Once the user submits the request for scheduling a meeting, the following actions need to be validated*

*(a) A database entry is created with status “Created”.*

*(b)The calendar information on the meeting, should be transformed into a poll table where the host is suggested available time slots and busy timeslots) and should be given the option to select the desired timeslots.*

*(c)The meeting status should get updates to indicate that slot selection has been done.*

### 3.2.8 Functional Requirement 8

*FR 8. The system should take into consideration the participants' votes before finalization of the meeting dates.*

*Acceptance Tests:*

*(1)The change of a meeting status to “Slots Selected” should trigger mail sending to the participants, and the mail should direct the participants to a poll page where they can record their time slot choices. If no timeslot is feasible a participant should be able to mark his vote as “cannot make it”.*

### 3.2.9 Functional Requirement 9

*FR 9. The user should receive reminders regarding pending polls to be taken before deadline to inform the host about their availability.*

*Acceptance Tests:*

*(1) A user should get email reminders for pending polls to be attended until the deadline, after which his vote should be marked to be unrecorded.*

### 3.2.10 Functional Requirement 10

*FR 10. The users should receive email notification on the finalization of meeting date and time.*

*Acceptance Tests:*

*(1) On finalization of the meeting date, emails should be sent the participants informing them about the meeting dates.*

## 3.3 Use Cases

### 3.3.1 Use Case 1

|  |  |
| --- | --- |
| *Use Case Number* | *1* |
| *Use Case Name* | *Schedule Meeting* |
| *Actors* | *Logged In/ Signed Up/Guest Host* |
| *Description* | *The user enters title, date, venue, duration information and clicks on submit to schedule the meeting* |
| *Precondition* | *The existing user should either sign in and new user should sign in after signing up for an account* |
| *Basic Flow* | *The host is given a lists of available time slots, from which he selects the slots he wants and this triggers email request with polls sent to the participants who reply and on receiving the response or on reaching deadline, the decision is made.* |
| *Alternative Flow* | *If the user fails to enter the mandatory fields,*  *or cancels in between , the meeting request is reset and he is taken to the starting page.* |
| *Post Condition* | *Once the decision is made the user should be able to see the status of the decision on the profile page and it should match* |

### 3.3.2 Use Case 2

|  |  |
| --- | --- |
| *Use Case Number* | *2* |
| *Use Case Name* | *Accept/Reject Meeting* |
| *Actors* | *Logged In/ Signed Up/Guest Participant* |
| *Description* | *The user receives a mail that directs him to a page, where he should be able to record his vote or reject the invite in case he will not be able to make it.* |
| *Precondition* | *The user receives a mail invite to a meeting from the system.* |
| *Basic Flow* | *The user as a participant, should be able to record his votes, which gets saved in the database. These votes are later used in making the final decision.* |
| *Alternative Flow* | *If the user doesn’t record his vote before the deadline, his response status is set to “unrecorded” and the final decision is made.* |
| *Post Condition* | *Once the final decision is made he will receive a notification email with updates about the meeting and will be able to see the same if he logs in to his profile page.* |

## 3.5 Non-Functional Requirements

### 3.5.1 Performance

### 3.5.1.1 Performance Requirement 1

*PR 1. The user should be able to invite any number of participants to the meeting with ease*

### 3.5.1.2 Performance Requirement 2

*PR 2. The system database should be able to support the storage of large amounts of calendar information of each user.*

### 3.5.1.3 Performance Requirement 3

*PR 3. The system should always use the latest calendar information for calculating the available time slots of a meeting.*

### 3.5.1.4 Performance Requirement 4

*PR 4*. *The system should be able to synchronize multiple meeting requests involving the same participants at around the same time.*

### 3.5.2 Reliability

### 3.5.2.1 Reliability Requirement 1

*RR 1*. *The user should have his account and calendar information protected.*

### 3.5.2.2 Reliability Requirement 2

*RR 2*. *The hosts should not be given access to view the participant calendar data and vice versa.*

### 3.5.2.3 Reliability Requirement 3

*RR 3*. *The system should retrieve and update the calendar information in a secure and authenticated manner.*

### 3.5.3 Availability

### 3.5.4 Security

*S.R 1 The user passwords should be encrypted before storage in database.*

### 3.5.5 Maintainability

*MR 1: The system should be designed in a way which makes it scalable and future maintenance easy.*

### 3.5.6 Portability

*Po R 1: The system should be built in a portable fashion, be it with respect to the database or application server. We should be able to migrate the system with ease and minimal change at minimum cost.*

## 3.6 Design Constraints

*D R 1.Meteor Framework has been used for development of the system. The frontend code is in JavaScript and the backend uses NodeJs, with backend support of Mongo DB for data storage.*

## 3.7 Logical Database Requirements

DR 1. A database should be used which can store all the user account information, meeting details, calendar information, guest user details. The database transaction should be made reliable by following the ACID principle.

# 4. Analysis Models

## 4.1 Data Models

C:\Users\Jeme\Desktop\ER diagram.png

## 

## 4.2 State-Transition Diagrams (STD)

C:\Users\Jeme\Desktop\State_Machine.png

# 5. Change Management Process

*The change management process involved adding the new requirements to the requirement documents for traceability purposes. And each code change was regressively tested, and if an existing feature was broken by the new change, the person making the new change fixed it as well.*