Final Project Report

For

MeetMe

Product Owner: Michal Young

Contributors: Aditya Kanakamedala, Syed zafar shah, Dhawal Soni, Praful Gaikwad, Viplav Kadam

**1.**    **Final Project Summary**

**//Summary should be written here**

* 1. **Key Decisions**

**Scope**

The key decision for choosing this system is to create a better meeting scheduler planner which is easier and efficient than other existing meeting planners. It allows the users to schedule meetings or respond to a meeting availability request in a simple, easy to use environment. The user can view scheduled events on his dashboard. The application can draw from other calendar sources such as google calendar in order to provide the user with a much more convenient way to schedule events taking into consideration other calendar events. The synced calendars do not violate user privacy.

**Development Process**

The process flow is modelled on scrum, which is and in incremental agile process methodology. The primary advantage of the process model is flexibility, and its ability to incorporate the learning curve efficiently into the process model. The model helps to adapt in cases of constantly evolving and changing requirements. The changing requirements are easy to incorporate in scrum. The entire product backlog was divide to be completed in five sprints. The first four sprints were two weeks long, with a major part of the development happening in these sprints. The final sprint is a week-long sprint which was primarily dedicated to stabilizing the system.

**Requirements: Problem Statement**

The meeting planner titled MEETME is expected to meet the following constraints at the very minimum:

* It should draw user’s data from Google Calendar or other Calendaring applications to help schedule prospective meeting times
* It should respect the privacy of user while fetching data and the meeting times of other participants should not be visible to that particular participant
* The scheduler should work for user’s google calendar or any other calendaring application that he is using.

There was a scope for a lot of enhancements, and we could get around some of them:

* Include well defined protocols while fetching user’s data from other calendaring applications.
* Support web based and email based responses from prospective meeting times on multiple platforms.
* Supporting complex attendees requirements like A and B should compulsory be present and whereas presence of C optional.
* Distinguish between user’s possible meeting and ranked meeting time.
* Handle different time zones of the users where the users have a possibility of selecting time slots of different parts of the day
* As a user system, I want to show the calendar in a grid view of the synced google calendar data, so that I don’t have to separately manage the calendar application for Meetme and google calendar.
* Integrate the various landing pages to create a seamless flow of control through the website
* Put proper validation in place to check each and every field
* Implement the forgot password functionality so that a user gets an email to reset the password.
* Design a home page after successful login.
* Use facebook login to authenticate to the website. As a user system, I want to research on the Outlook API's and deploy my project on Windows Azure so that my project shall extract the Outlook Calendar API's

**1.2 Requirement Specification:-**

The requirements that we had were framed in the form of user stories. These user stories were distributed among 5 sprints. We made use of scrum board present on Taiga to look at the requirements and make necessary changes and add meta - data for each of the user story.

We maintain a product backlog which contains all our requirements in the form of user stories and sprint by sprint we move the requirements in the current sprint. The product backlog is what will ultimately be delivered, ordered into the sequence in which it should be delivered. Product Backlogs are broken into manageable, executable chunks named sprints. At sprint is initiated to deliver a certain number of user stories from the backlog, in accordance with their skills, abilities and resources. The project advances as the backlog becomes depleted.

**Product Backlog:-**

The product backlog consisted of the set of user stories that were elicited by having one of the team member proxy as product owner. Limited availability of product owner has forced us to use this elicitation technique, however, efforts were made to get the user stories from the sponsor himself. The user stories were regularly reviewed and updated after each sprint by having a team member proxy as the product owner. The priority user stories are moved to the sprint backlog at the beginning of each sprint. The user stories are then decomposed into a set of tasks which are assigned to the team members. The aim is to close all or most of these tasks before the end of the sprint. An important feature of user story elicitation is attributing each user story with IEEE The user stories that went into the final product backlog are:

* Create login page for logging into the system.
* Create form for user registration and store in database.
* As a user system, I want to show the calendar in a grid view of the synced google calendar data, so that I don’t have to separately manage the calendar application for Meetme and google calendar.
* Integrate the various landing pages to create a seamless flow of control through the website
* Put proper validation in place to check each and every field
* Implement the “forgot password” functionality so that a user gets an email to reset the password.
* Design a home page after successful login.
* Use facebook login to authenticate to the website. As a user system, I want to research on the Outlook API's and deploy my project on Windows Azure so that my project shall extract the Outlook Calendar API's
* As a user system, I want to make my azure application extract Outlook calendar Api's so that it can benefit in accessing user's outlook calendar data
* As a stakeholder, I would like that the complex attendance requirement be supported so that we can get a consensus from people that are required in the meeting.
* Persons A,B, and C, are necessary; persons X,Y, and Z are invited but not necessary for the meeting; at least two of M,N, and O should be present
* As a user, I want to schedule events in a custom web-site calendar, so that i don't have to sync calendaring applications.
* As a user I wish to have my data private and secure regardless of any calendaring applications I choose so that I can use my schedule and settings conveniently.
* As a user, I wish to have the data from my google calendar synced in to my meetme account so that I don’t have to switch back and forth between the applications.
* As a system user I want to perform validations on New account sign up so that only valid users are allowed to create account
* Creating pages which are visually appealing, polished and professional with product logo, product details and product features.
* As a system user I want to perform validations on New account sign up so that only valid users are allowed to create account

**1st Sprint- 21st Sep, 15 – 5th Oct, 15:-**

The user stories that were moved from the current product backlog into this sprint are:

1. Create login page for logging into the system.
2. Create form for user registration and store in database.
3. User wants to sync the google and the outlook calendar with Meet Me.
4. When the user logs into the system, if he clicks on remember me, then the next time he logs in again, he should be able to do it without putting in the credentials
5. Use bcrypt for storing the crypted password in the database
6. Create the database keeping in mind all the possible scenarios
7. The User does not use the primary calendar apps viz. Microsoft Outlook and Google Calendar. So there should be other interface to take the input from the user.
8. Start working on front end design and integrating the pages created.
9. To add the related documents and updates to the asu website.
10. All team members research on MVC, node.js, jquery, MySQL, Spring MVC Framework, calendaring applications, google calendar API documentation, Outlook API documentation and write up important points to contribute to the sprint planning meeting.

**2nd Sprint- 5th Oct, 15 – 19th Oct, 15:-**

The user stories that we pushed into this sprint are:

1. User wants to sync the google and the outlook calendar with Meet Me.
2. When the user logs into the system, if he clicks on remember me, then the next time he logs in again, he should be able to do it without putting in the credentials
3. Use bcrypt for storing the crypted password in the database
4. Create the database keeping in mind all the possible scenarios
5. The User does not use the primary calendar apps viz. Microsoft Outlook and Google Calendar. So there should be other interface to take the input from the user.
6. Start working on front end design and integrating the pages created.
7. To add the related documents and updates to the asu website.
8. All team members research on MVC, node.js, jquery,mySQL, Spring MVC Framework, calendaring applications, google calender API documentation, Outlook API documentation and write up important points to contribute to the sprint planning meeting.

Our midsemester was graded till Sprint 2 and we received our grade based on our user stories framed and the functionality of each of them that we implemented. After sprint 2 our user stories for the succeeding sprints took a different format as “As a system user, I want to \_\_\_\_\_\_\_\_\_\_\_ so that it can be advantage to \_\_\_\_\_\_\_”

**3rd Sprint-19th Oct, 15 to 2nd Nov, 15:-**

The user stories that we pushed in this sprint are :-

1. As a user system, I want to show the calendar in a grid view of the synced google calendar data, so that I don’t have to separately manage the calendar application for Meetme and google calendar.
2. Integrate the various landing pages to create a seamless flow of control through the website
3. Put proper validation in place to check each and every field
4. Implement the forgot password functionality so that a user gets an email to reset the password.
5. Design a home page after successful login.
6. Use facebook login to authenticate to the website. As a user system, I want to research on the Outlook API's and deploy my project on Windows Azure so that my project shall extract the Outlook Calendar API's

**4th Sprint 2nd Nov- 16th Nov:-**

The user stories that we pushed in this sprint are :-

1. As a user system, I want to make my azure application extract Outlook calendar Api's so that it can benefit in accessing user's outlook calendar data
2. As a stakeholder, I would like that the complex attendance requirement be supported so that we can get a consensus from people that are required in the meeting.
3. Persons A,B, and C, are necessary; persons X,Y, and Z are invited but not necessary for the meeting; at least two of M,N, and O should be present
4. As a user, I want to schedule events in a custom web-site calendar, so that i don't have to sync calendaring applications.
5. As a user I wish to have my data private and secure regardless of any calendaring applications I choose so that I can use my schedule and settings conveniently.
6. As a user, I wish to have the data from my google calendar synced in to my meetme account so that I don’t have to switch back and forth between the applications.
7. As a system user I want to perform validations on New account sign up so that only valid users are allowed to create account
8. Creating pages which are visually appealing, polished and professional with product logo, product details and product features.
9. As a system user I want to perform validations on New account sign up so that only valid users are allowed to create account

**1.3 Requirement Analysis:-**

**1.4 Architectural Design:-**

The framework on which our application is being built is Spring MVC architecture where M, V, C means Model View and Controller. MVC is popular as it isolates the application logic from the user interface layer and supports separation of concerns. Here the Controller receives all requests for the application and then works with the Model to prepare any data needed by the View. The View then uses the data prepared by the Controller to generate a final presentable response. The MVC architecture from spring’s perspective is a below:

* The **Model** encapsulates the application data and in general they will consist of Plain Old Java Objects (POJOs).
* The **View** is responsible for rendering the model data and in general it generates HTML/JSP output that the client's browser can interpret.
* The **Controller** is responsible for processing user requests and building appropriate model and passes it to the view for rendering.

The MeetMe application is divided into Model, View and Controller modules.

1. Register/Login Module
2. Event Creation
3. Event Display
4. Event Pooling
5. **Register/Login Module:-** Initially the user has to sign up with MeetMe by providing the personal details, registering his email id and agreeing with the terms and conditions of MeetMe. As the user signs up there are certain fields which are mandatory and certain fields where the user has to enter data according to a specific format. If not done then the page prompts an alert to the user to enter his details in proper format. As the user registers his email-id can be a google email or a non- google email. When the user registers he has to login and enter the application with the proper login credentials. If a user does not prefer to register, he can directly login using Google or Facebook Authentication. Required permissions are taken from the user to access the data.
6. **Event Creation:-** Once the user logs in to the application, he can create an event. On the event creation page the user enter the parameters of the event i.e event title, event description, a probable time slot (these can be any number of entries). The user then sends the event details to the invitees (guests or MeetMe google users). The invitees then send their preferred time slot. Once the preferred time slot is sent the algorithm is triggered which takes all the time slots as input and performs computations considering the fact that required users should respond and be present in the meeting. The Algorithms then sends preferred time and ranked time as output to the participants. For the Google users that are part of the MeetMe application, no form of communication through email takes place. The probable timings are generated and ranked from the first to the last date. Once done, an email is being sent to all the people involved in the meeting with the list of timings.
7. **Event Display:-** The signed in user gets to see all the events that are scheduled for him on the dashboard. If the user gives permission to access his google calendar then all the events from his google calendar are fetched and displayed on the dashboard. This was achieved by first storing the google calendar events in database and then displaying them on the calendar dashboard.
8. **Event Polling:-** In this module we are populating the participants with all the events that he has been invited to attend. The link is sent via email and after clicking on the link the user selects one particular time slot from the given time slots and that is sent as an input to the trigger\_algorithm.

**1.5 Project Planning**

All members of the team have responded, contributed, and interacted well with each other throughout the project. We have arrange meetings on regular basis to discuss about the status of the project and tasks to be done in the next sprint. We have focused on the project deliverables of one sprint at a time and discussed them in detail. After that, we have distributed the work among the team. The tasks have completed the assigned work in a timely manner to finish the sprint on the time. We have updated the sprint status on the taiga and updated source code on GitHub on the regular basis.  We have described above the important User stories as per each sprint.

* Sprint 1: It will be more of a getting used to an agile system Sprint. We will go through the problem statement presented by the Product Owner and try to form user stories in the scrum-board. We will then research on the possible technologies that are good to work on with respect to the security aspects and also how comfortable each and every team member is with the development in the technology used. We will try to code some of the functionalities by the end of the Sprint1 so that we can get a good start and continue the implementation in the coming sprints.
* Sprint 2: After Sprint 1, we assessed the progress with respect to getting used to the technologies,
* Requirements. We will be taking forward the implementation of all the functionalities that have been implemented and try to add even more. We will work on the sync of google and outlook data and check the feasibility if it can be done or not. We will include a calendar view to display events of the users. We will also keep working on the event creation on the calendar.
* We also will concentrate on creating the documents, the ER diagram, presentation for the mid semester review and also a video on the current status of our working project.
* Sprint 3: Now as the Sprint 2 is over, we have assessed the outcome of it. We will now further plan to include all the tasks to be done in Sprint 3. Now that we know of correct format on how to write a user story, we will adhere to that and start writing user stories as per the correct format. Till now we had been treating user stories and tasks as same. So now we will be changing the approach and include the tasks for the user stories. We will plan to complete the google authentication and also try to see if we can authenticate a user using facebook. Due to issues faced in dealing with event creation in the calendar view, we will plan to create a new manual page for the complete flow.
* Sprint 4: Now as the Sprint 3 is complete, we have assessed how the implementation went. We will now device out a plan for sprint 4. During the Sprint 3 we found that, due to some API issues, we were not able to retrieve the values of Outlook Calendar. So we will be leaving it and concentrate on the remaining functionalities. We will complete the flow of creating an event and sending invites to the user. We will also work on the algorithm to rank the meeting times. We will be integrating the google authentication done as a Java and Google Application Engine to Spring.We will also create event creation mechanism for the users that are logged in as google. We will also start
* Sprint 5: Now as the Sprint 4 is completed, we are left with one week of the last Sprint. Now as we assessed the fourth Sprint, we found that almost all the tasks that we planned to complete, were done. We had written the algorithm to be implemented. We also were done with synching the google events. So in this sprint, we will be connecting all the dots and stabilize the system. We will check the algorithm output with the events that we are creating and also for the events created with google meet me users. We will also be making certain changes to the UI and do a round of bug testing and improve the system functionality. We will deploy the website on a server so that we can be independent of all the dependencies related to the localhost.

**Milestones:**

* Sprint 1: We completed the login and registration process for the MeetMe users.
* Sprint 2: We implemented forgot password, remember me, password crypt and a new bootstrapped UI to improve the look and feel of the website.
* Sprint 3: We created the validations on each of the pages that we created so that users cannot input garbled data. We also achieved facebook and google authentication. We created a new event creation flow. We also were able to retrieve the google calendar data.
* Sprint 4: We completed the complete flow of event creation from creating event to sending mails to storing user responses and testing the algorithm on a manual data.
* Sprint 5: We improved our UI and fixed the bugs in the system. We also include the google meet-me user event creation. We deployed our website on open-shift server successfully

**1.6.**    **Management Plan**

1.6.1 Sprint planning meetings

Sprint planning meetings took place on the beginning of each sprint in order to discuss the plans for the next sprint. Planning meetings also consisted of reviewing the previous sprint and the change in directions, if needed were decided in these meetings.

1.6.2 Daily stand up meetings

Daily stand up meetings lasting no more than ten minutes were held thrice a week (on class days) to discuss individual progress on sprint tasks, and also team progress. These stand up meetings have been quite helpful to adapt and change the course of execution when faced with challenges.

1.6.3 Project Management

The team maintained a shared repository on GitHub, and a scrum board on taiga.io. Taiga was the primary project management tool. Due to changing roles in each sprint, a major portion of the sprint meetings were devoted to assigning roles for the coming sprint. The previous sprint was reviewed in sprint planning meetings and an action plan was arrived at the end of the meeting. The action plans were updated on the scrum board as team requirements.

1.6.4 Accountability

Each team member had to validate his work in the form of milestones, which each team member would set for himself. At the milestone, the team member has to commit code to the central repository. Even experimental code implemented as part of learning curve was considered as a valid milestone, and pushed to the repository. This allowed for the team members to plan his/ her tasks and deliver the task at their own pace. The milestones acted as accountability parameters for each team member. The milestones were peer reviewed in order to ensure that no team member creates fake milestones. Any failed milestones were discussed during the mid-sprint meeting, which was a Sunday in the middle of the sprint.

**1.7 Implementation**

We have used spring MVC to design this system. We have used bootstrap and javascript for the front end design and java for the backend language. We have used MYSQL database to store the project data. We have also developed an efficient algorithm to help scheduler to select a preferred time among the participants to schedule meetings.

The steps of the algorithm is as following:

1.       Create counters same as preferred time slots generated by the meeting scheduler.

2.       Send emails to the participants with preferred time slots.

3.       Insert preferred time slots of every user into database.

4.       Fetch time slots for the event and store into a list.

5.       Run a loop till the end of list.

6.       Compare list elements with the time slots provided by meeting scheduler.

7.       If matched, increase respective counters of the time slot.

8.       Sort the time slots with respect to counter.

9.       Select time slot with highest counter value as the preferred time slot.

10.   Rank remaining time slots from highest counter value to lowest counter value.

11.   Send the preferred time slot and remaining time slots with rank value to the participants.

For google User

1.       Fetch google calendar of every user.

2.       Create an array in which every element represent a time slot of the window preferred by the scheduler.

3.       Run a loop for each user.

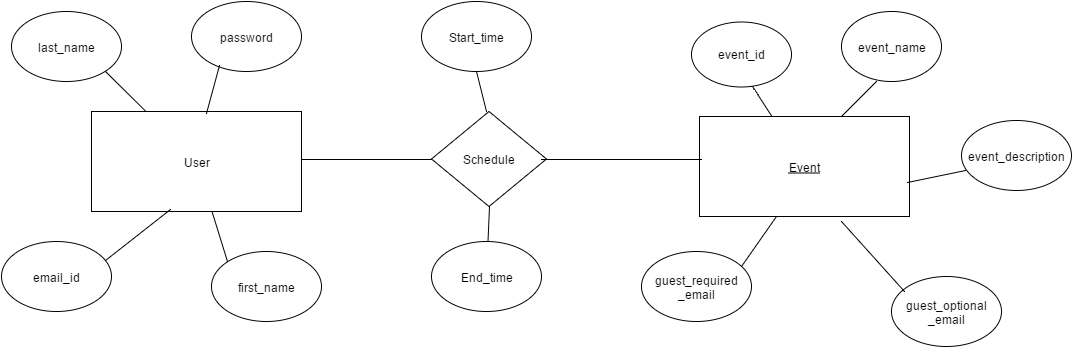
4.       Fetch time slots of events (busy time slots) during the given window and mark respective array element.

5.       After that, find unmarked array elements and its respective time slots.

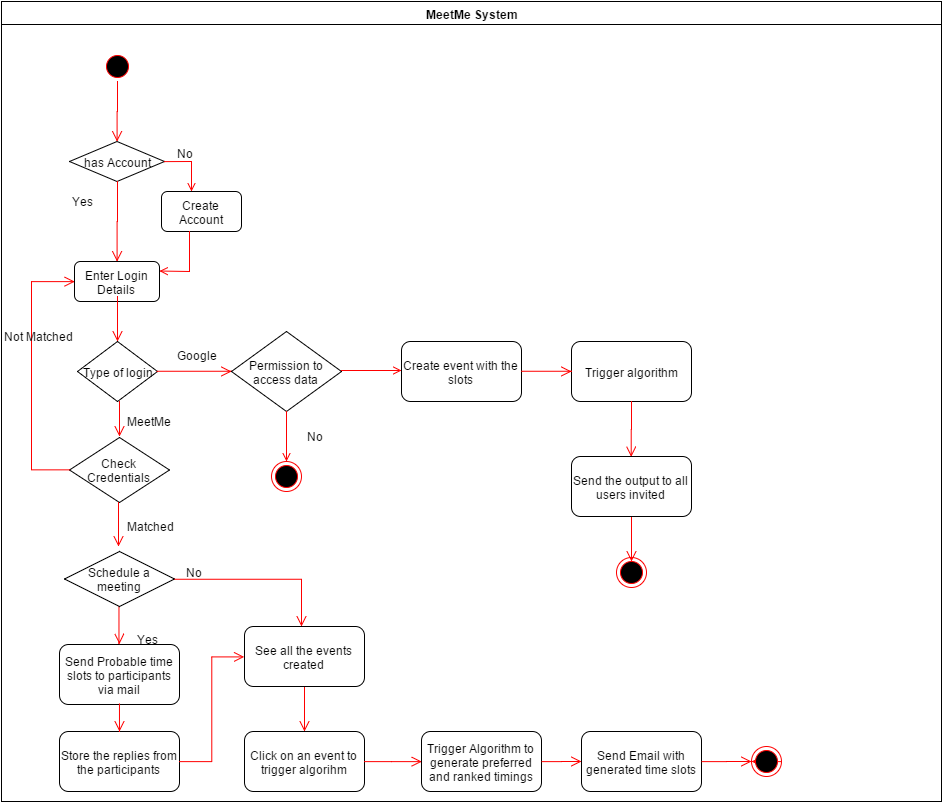
6.       Send the time slots to the participants.

Before starting implementation, we have developed diagrams to build the system such as ER diagram, Activity diagram and Use case diagrams.

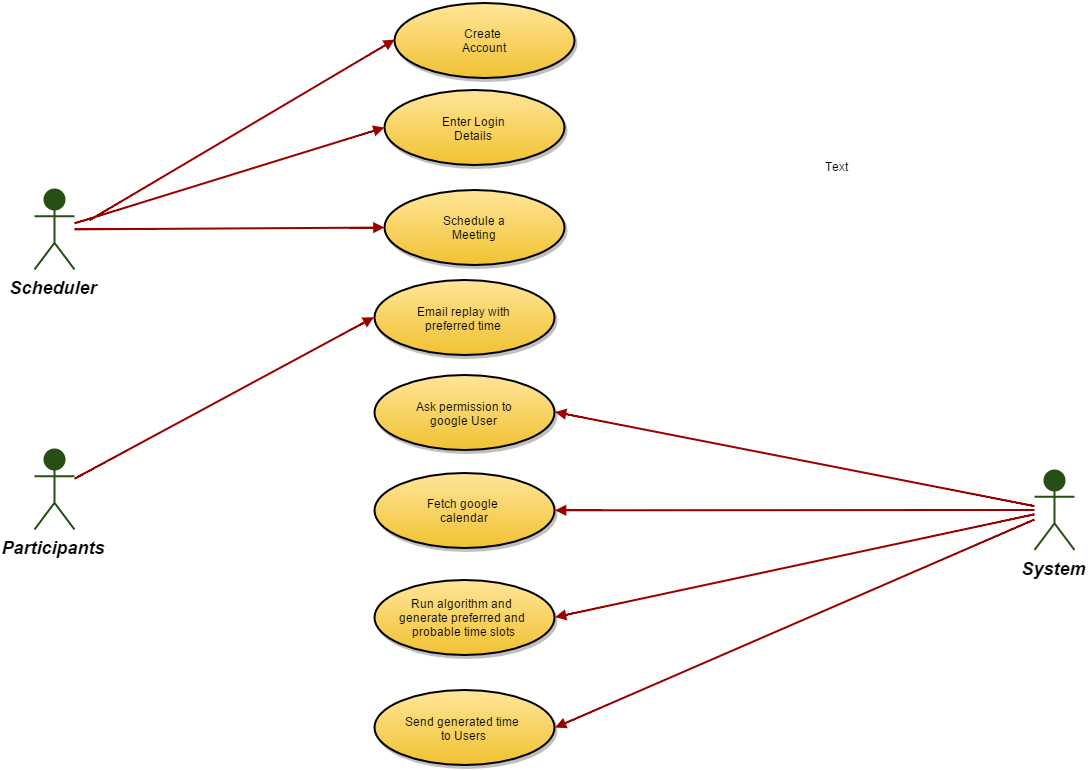
**ER Diagram:-**

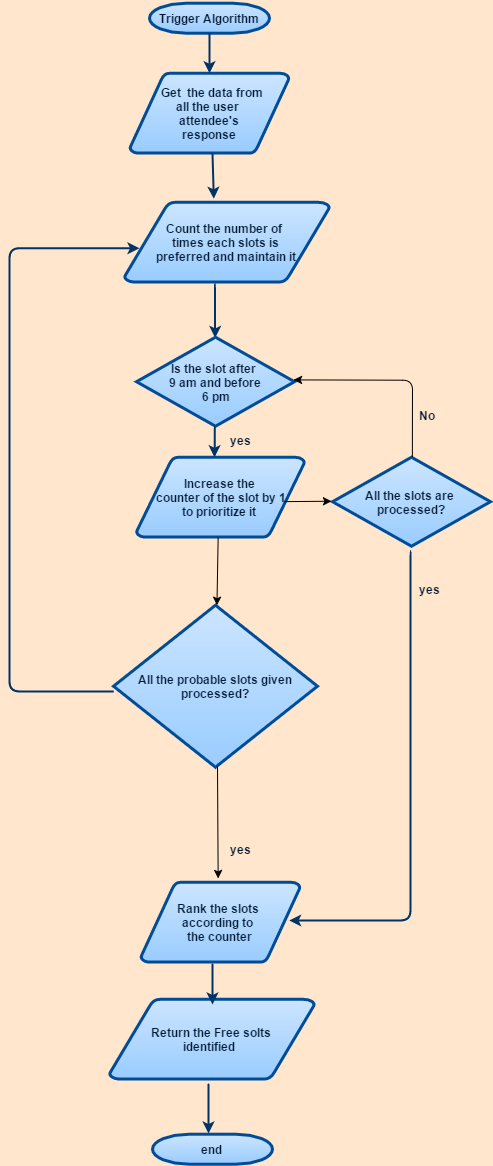
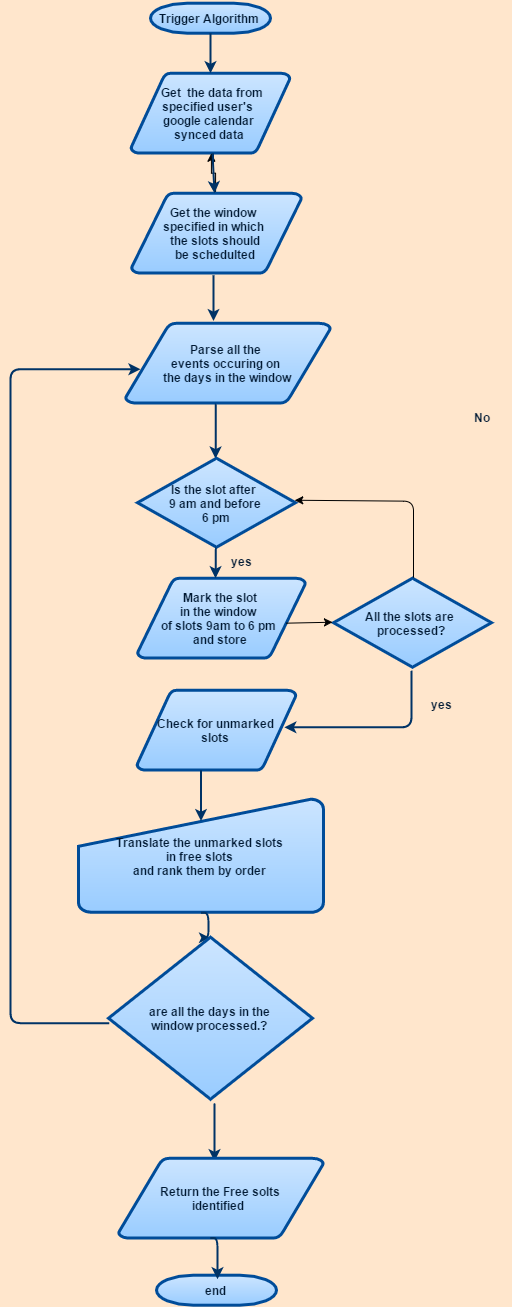


**Activity diagram:-**



**Use case Diagram:-**



Algorithm to Schedule meetings for Google users Algorithm to schedule meetings for manual scheduled meetings.

**FLOWCHART IMPLEMENTATION OF ALGORITHM IN BOTH CASES.**

**1.8**    **Validation and Verification**

We have provided validation on the login credentials so that only authorised user will have access to the system. We have also taken into account user’s privacy. We have asked user to give permission before fetching google calendar. For testing, We have performed Unit testing, Integration testing and functional testing. In unit testing, we have tested every small testable part of the application. After that, we have done the integration testing at time of combining modules. In the end, we have taken into account validation testing and checked weather system meets its intended purpose. In this testing, we have tried to improve quality of the system.

**1.9.**    **Outcome & Lesson learned**

The project implemented a lot of latest and popular technologies used for developing web applications. Following are the learnings in the technical domain:

1. Spring MVC architecture:
   1. Hands on work with spring MVC architecture has unraveled most of the work Spring does behind the curtains to ensure convenience for the developers.
2. OAuth2 authorization;
   1. The protocol used for authorizing users from other social websites like Google+, Facebook etc. was used to maintain security and privacy concerns.
3. Jquery and Bootstrap:
   1. Implementing a website compliant with all the web browsing devices enabled us to learn bootstrap and the functionality to implemented in Jquery.
4. Design process is important than the Developing process:
   1. Analysis of the requirements and their tentative translation in the source code is a significant thing

The process has been quite a learning experience giving huge insights about the Agile process of software development. Apart from the lessons in the technology used for implementing the web application project, lessons concerning the AGILE software development process were noteworthy. Following are these lessons concerning the AGILE process experienced by our team.

1. Quick decision Making –
   1. Considering the Spring framework, it was decided to use the Microsoft outlook and Google calendar application to draw the users data to find out the meeting times.
   2. Microsoft outlook apparently was a bit difficult to implement in the given timeframe.
   3. We were a bit slow to decide and rule out the integration of outlook which consumed a lot of time and energy.

Lesson Learnt: Taking such decisions early helps to save a lot of time and energy of the team members and also clears the way to concentrate on other requirements efficiently.

1. Wise choice of Technology :
   1. We chose Spring framework owing to its security promises which were perceived to be thought as a security wrapper without much of a do.
   2. The Spring framework has a great learning curve and given the project duration, we found ourselves struggling with learning the things to be implemented along with Spring.

Lesson Learnt: Considering the project duration and the technologies used highly influence the performance of the developers and progress of the project.

1. Turnaround Time:
   1. The tasks assigned to developers concentrated on how good task are done rather than how fast they can be done.
   2. This introduced huge latencies in turnaround thus affecting performance.

Lesson learnt: Focus should be on getting the tasks done by either an appropriate workaround or with proper measures in the given period of time.