Final Project Report

For

MeetMe

Group Members

|  |  |
| --- | --- |
| **Name** | **Role** |
| Dr. Kevin Gary | Project Manager |
| Praful Gaikwad |  |
| Syed Zafar Shah |  |
| Dhawal Soni |  |
| Aditya Kanakamedala |  |
| Viplav Kadam |  |

**1.**    **Final Project Summary**

**1.1.**    **Key Decisions**

**Scope**

The key decision for choosing this system is create a better scheduling meeting planner which is easier and efficient to use than other existing meeting planners. It allows users to log in into the applications either as a scheduler to schedule meetings and send invites to participants or as a participant to accept invites and participate in meetings. Moreover the user can also view the events that has been scheduled in his calendar dashboard. The application also provides a feature to user to log in using google credentials and allow them to provide their google calendar to scheduler in order to decide preferred time slots for the meeting without violating user privacy.

**Development Process**

We have used scrum which is an incremental and interactive agile software development methodology. it provide flexibility for user throughout the project as the learning occurs and environment changes. Since we are using this methodology for the first time, we have faced some challenges. We have handled the changes in the requirements with great care and again estimated the time to finish the requirements within the pre decided sprint. We have used online tool called taiga to implement scrum methodology and control the flow of the project. We have also used github to push our source code so that it will be available for each team member.

**Requirements: Problem Statement**

The meeting planner that we were supposed to make need to have conditions that a minimum meeting planner is expected to have:-

* 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 were certain enhanced features that were asked to make it a part of application. We tried to work on some of them but couldn’t completer because of time constraints.

* 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

**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. Now let us know MVC from Spring perspective.

* The **Model** encapsulates the application data and in general they will consist of POJO.
* The **View** is responsible for rendering the model data and in general it generates HTML 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 giving his 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 sign up and enter the application with the proper login credentials.
6. **Event Creation:-** Once the user sign in into the application, he gets to Manually 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.
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 Pooling:-** 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.

**1.2.**    **Lessons Learned**

**1.3.**    **Learning Outcomes Summary**

**1.4.Project Plan**

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 tiaga and updated source code on the github on the regular basis.

**1.6.**    **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.

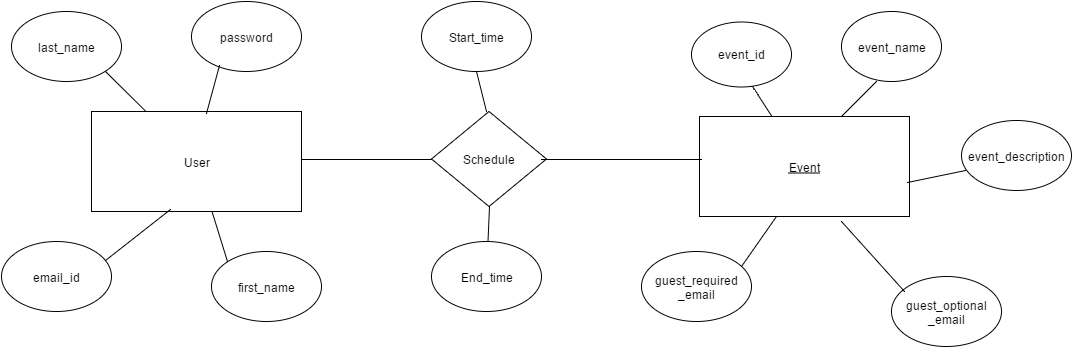
4.       Fetch time slots of events (busy time slots) during the given window and mark respective array element.

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

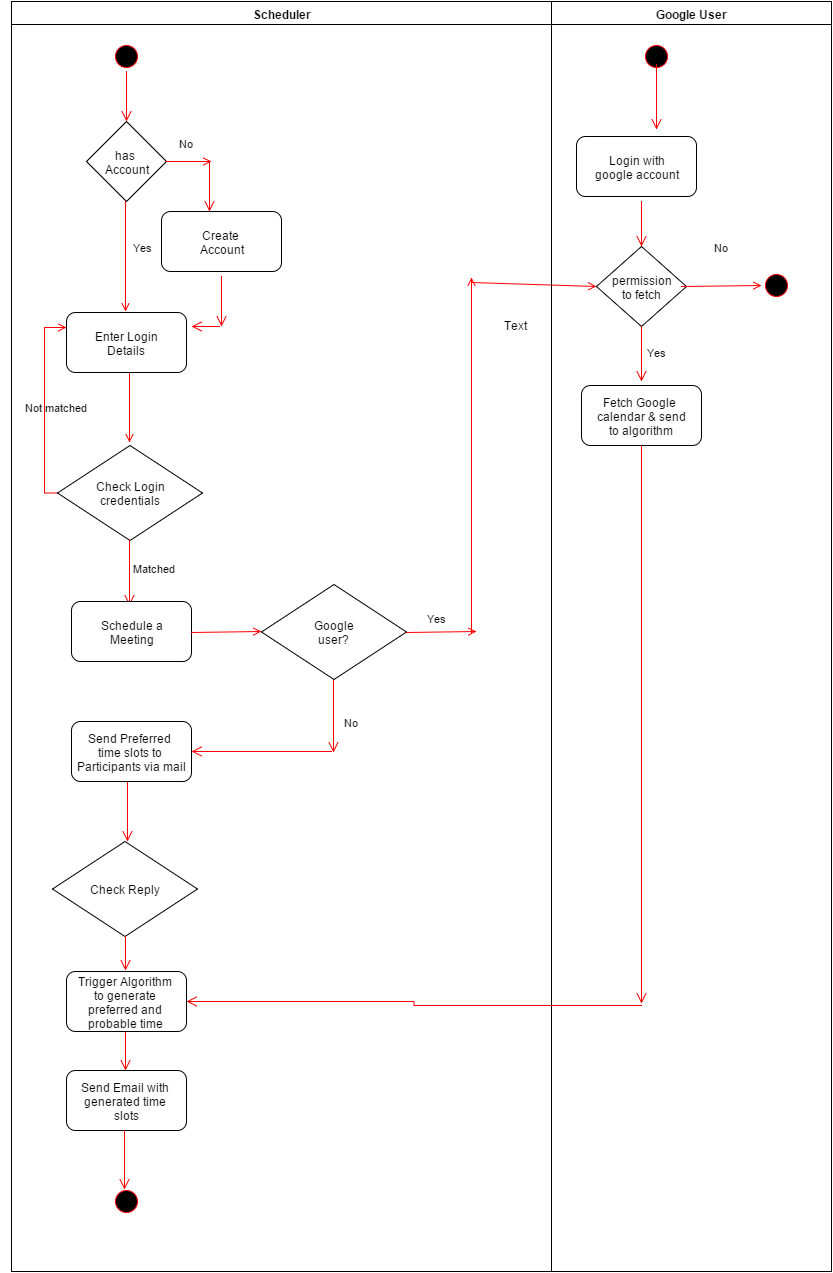
6.       Send the time slots to the participants.

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

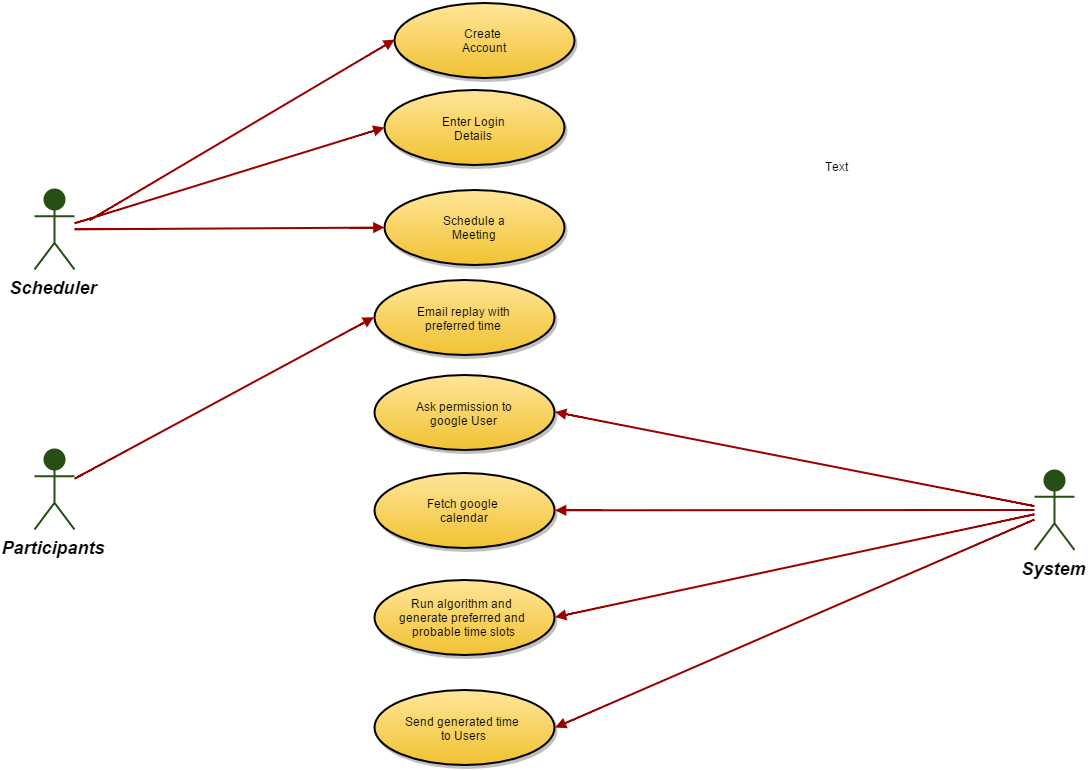
**ER Diagram:-**



**Activity diagram**



**Use case Diagram:-**



**1.7.**    **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.

**1.8.**    **Outcome & Lesson learned**

During this project, we have learned about different aspects of the project development. We have learned the importance of the requirement elicitation phase. We have learned how to gather requirements and how to create user stories corresponding to the requirements. We learned scrum agile methodology and how to use this methodology during this project. We also got opportunity to identify the advantages of this methodology as compare to other existing methodology. We also learned how the diagrams such that ER diagram, Activity diagram and Use case diagram helps to develop system easily. We also learn spring MVC architecture and how to implement this architecture to design the system. We also learned about front end languages such that bootstrap, javascript, back end languages such that java and MySQL database during this project.