Low Level Design (LLD)

BOOK MY CALENDAR

Revision Number: 0.1

Last date of revision: 06/06/2022

Saurav Gupta

# Document Version Control

|  |  |  |  |
| --- | --- | --- | --- |
| **Date Issued** | **Version** | **Description** | **Author** |
| **6th June 2020** | 0.1 | First Draft | Saurav Gupta |
|  |  |  |  |

**Contents**

[Document Version Control 2](#_TOC_250011)

[Abstract 4](#_TOC_250010)

1. [Introduction 5](#_TOC_250009)
   1. [Why this Low-Level Design Document? 5](#_TOC_250008)
   2. [Scope 6](#_TOC_250007)
2. [Technical speciﬁcations 7](#_TOC_250005)
   1. Input Schemas/Models 7
   2. Logging 8
   3. Database 8
   4. Deployment 9
3. [Technology stack](#_TOC_250004) 9
4. [User I/O workﬂow 1](#_TOC_250001)0
5. [Test Cases 1](#_TOC_250000)1
6. Key Performance Indicators 12

# Abstract

Due to covid, we have shifted most of our works to online platform. These includes classes, meetings and other stuff as well. Book My Calendar is an application which helps in booking the calendar of the owner of the event so that you can schedule meetings with the event organizer. The event organizer sets certain condition that needs to be met for the meeting to take place. If the meeting scheduler meets all the requirements of the meetings then the meeting scheduler can schedule a meeting with the event organizer. This helps in booking the calendar in advance.

# Introduction

## Why this Low-Level Design Document?

The purpose of this document is to present a detailed description of the Book My Calendar. It will explain the purpose and features of the application, the interfaces of the application, what the application will do, the constraints under which it must operate. This document is intended for both the stakeholders and the developers of the system and will be proposed to the higher management for its approval.

The main objective of the application is to create event so that other users with whom the event is shared can book the calendar of the event organizer for having meeting. The event organizer can provide the time when the meeting can be scheduled. The event organizer keeps certain condition that needs to be met for scheduling a meeting.

## Scope

The application will be web-based application that will be using MERN stack and Mongo DB. The user will be able to create event so that other users with whom the event is shared can book the calendar of the event organizer for having meeting. The event organizer can provide the time when the meeting can be scheduled. The event organizer keeps certain conditions that needs to be met for scheduling a meeting.

# Technical speciﬁcations

## Input Schema/Models

We have mainly three models namely

* **User**
* **Event**
* **Meeting**

### In User model, we store the information about the user.

It contains

* **name:** the name of the user
* **lastname:** the last name of the user
* **email:** the email of the user
* **encry\_password:** the password stored in encrypted way and not as normal text
* **salt:** string used in converting password to encry\_password
* **my\_events:** list of events created by the user
* **events\_shared\_with\_me:** list of events shared by other users with this user
* **meetings:** list of meetings scheduled by the user

### In Event model, we store the following information about the event

It contains

* **name:** the name of the event
* **max\_meeting\_length\_in\_minutes:** the maximum length of the meeting
* **event\_start\_date:** the start date of the event
* **event\_end\_date:** the end date of the event
* **event\_days:** the days of the week when the event will occur
* **event\_start\_time\_per\_day:** the start time of the event on a day
* **event\_end\_time\_per\_day:** the end time of the event on a day
* **visible\_days\_in\_calendar\_from\_today:** the visible days in calendar in which the meeting scheduler can schedule the meeting
* **schedule\_meeting\_after\_hours:** the meetings should be scheduled the given amount of hours later than the current time
* **time\_gap\_between\_meetings\_in\_minutes:** the time gap between two consecutive meetings
* **event\_creator:** the id of the user who created the event
* **event\_shared\_with\_users:** list of users with whom the event is shared so that they can schedule a meeting
* **meetings:** list of meetings scheduled under this event

### In Meeting model, we store the following information about the meeting

It contains

* **name:** the name of the meeting
* **meeting\_start\_time:** the start date and time of the meeting
* **meeting\_end\_time:** the end date and time of the meeting
* **meeting\_organizer:** the user who scheduled the meeting
* **participants:** the users who are invited to the meeting
* **meeting\_platform:** the platform where the meeting will take place

## Logging

We should be able to log every activity done by the user.

The logging over here is done using the JSON response given by the backend. We return the success message and error message accordingly as per the requirement

## Database

System needs to store every request into the database and we need to store it in such a way that it is easy to retrain the model as well.

We store the following information of the three models discussed above in the database namely,

* **User**
* **Event**
* **Meeting**

The system stores each and every data given by the user or received on request to the database. We have used MongoDB as our database.

## Deployment

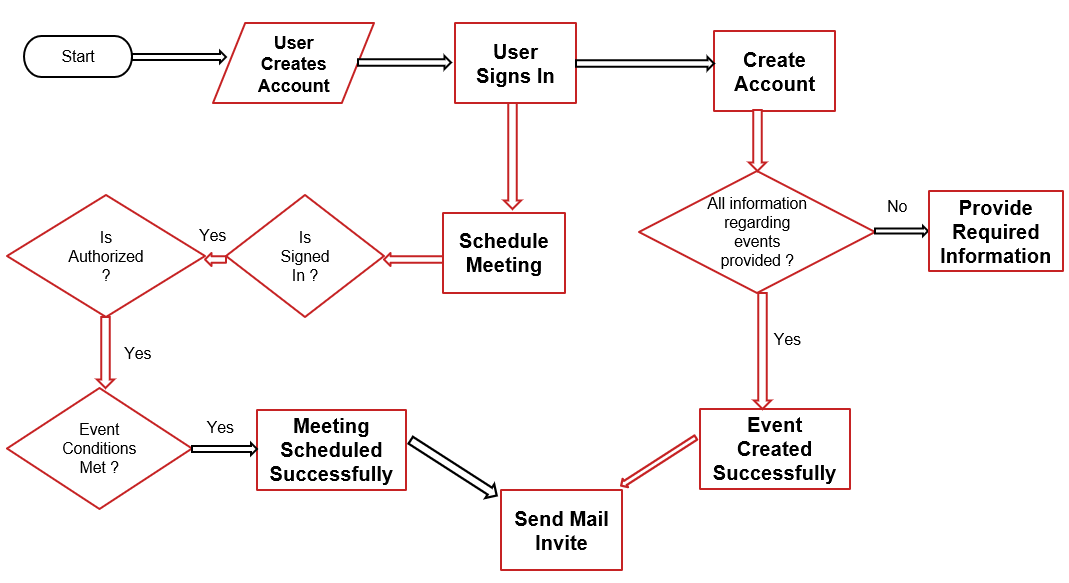
* 1. AWS
  2. GitHub
  3. Heroku GitHub



# Technology stack

|  |  |
| --- | --- |
| **Front End** | HTML/CSS/JS/React |
| **Backend** | NodeJS |
| **Database** | MongoDB |
| **Deployment** | AWS/Heroku Github |

# User I/O workﬂow



1. **Test cases**

|  |  |  |
| --- | --- | --- |
| **Test Case Description** | **Pre-Requisite** | **Expected Result** |
| Verify whether the API is accessible to the user | 1. API should be defined | API should be  accessible to the user |
| Verify whether the User is able to sign up in POSTMAN through API calls | 1. APIs are accessible | The User should be able to sign up in the application through API calls in POSTMAN |
| Verify whether user is able to successfully login to the application | 1. APIs are accessible 2. User is signed up to the application | User should be able to successfully login to the application |
| Verify whether user is able to see input fields on logging in | 1. APIs are accessible 2. User is signed up to the application 3. User is logged in   to the application | User should be able to see event creation request in POSTMAN |
| Verify whether user is able to provide values for all input fields in POSTMAN | 1. APIs are accessible 2. User is signed up to the application 3. User is logged in   to the application | User should be able to edit all input fields in POSTMAN |
| Verify whether user is presented with recommended response on clicking  send in POSTMAN | 1. APIs are accessible 2. User is signed up to the application 3. User is logged in   to the application | User should be presented with recommended response on clicking Send button in POSTMAN |
| Verify whether the data is stored in MongoDB database as per the inputs provided by user | 1. APIs are accessible 2. User is signed up | Data stored in MongoDB should be in accordance to what the user entered. |

1. **Key performance indicators (KPI)**

* Creating User Account
* Signing into the application
* Tokens attached to know whether the user is signed in and authorized
* Creating Events by the user
* Scheduling Meetings by the user
* Editing the details of Events by the authorized user
* Editing the details of Meetings by the authorized user
* Deleting the event by the authorized user
* Deleting the meeting by the authorized user