Implementation document

Team #05

March 26, 2022

Contents

I. Project Title and Authors

- Project title: ***Easy Team Up***
- Team number: **#05**
- Team Name: LFG
- Team meembers:
  - Yutong Li (3663393278)
  - Yuxuan Gao (4011350931)
  - Zhenbang Feng (8949822341)

II. Preface

This document includes the adjustments to previous documents we made during the implementation process.

III. Introduction

Following the design document, we fully implemented the app. Though most parts of the ideas proposed in the design document are adhered to, we made some changes to it, which either facilitates our implementation or contributes to the overall cleanness and usability of the app.

IV. Responsibilities

- *Feature 1: View Active Events*: Zhenbang Feng
- *Feature 2: Create Event Invitations*: Yutong Li
- *Feature 3: Manage Invitations*: Yuxuan Gao

V. Architectural Design Change

1. Backend framework

In our previous version, we plan to use **Springboot** as the backend framework. But we later realized most of the workload is on the frontend side, and there isn't a lot to do on the backend side except for retrieving data from MySQl and a few tasks involving basic logic. Therefore, we decided to switch to a more light-weight backend framework.

We choose the **express** framework using **NodeJS**, which provides us easy access to database and simple implementation for receiving HTTP requests from frontend.

2. Components correspondence

Since we moved to NodeJS and all functionalities are implemented in one single file, the current implementation is more of the functional programming style. We therefore list the correspondence between the components we proposed in the design document and the functions we actually coded. Wee will use th first line of each function to represent it. - Login/Signup Manager ⇒ `app.post('/login', (req, res) => {` `app.post('/signup', (req, res) => {` `app.get('/users/:id', (req, res) => {` - Event Manager ⇒ `app.post("/events", (req, res) => {` `app.post("/update_event", (req, res) => {` `app.post("/signup_event", (req, res) => {` `app.post("/withdraw_event", (req, res) => {` `app.post("/reject_event", (req, res) => {`

- History Manager ⇒

```
app.get('/events', (req, res) => {
app.get('/users/:id/invited_events', (req, res) => {
app.get('/users/:id/joined_events', (req, res) => {
```

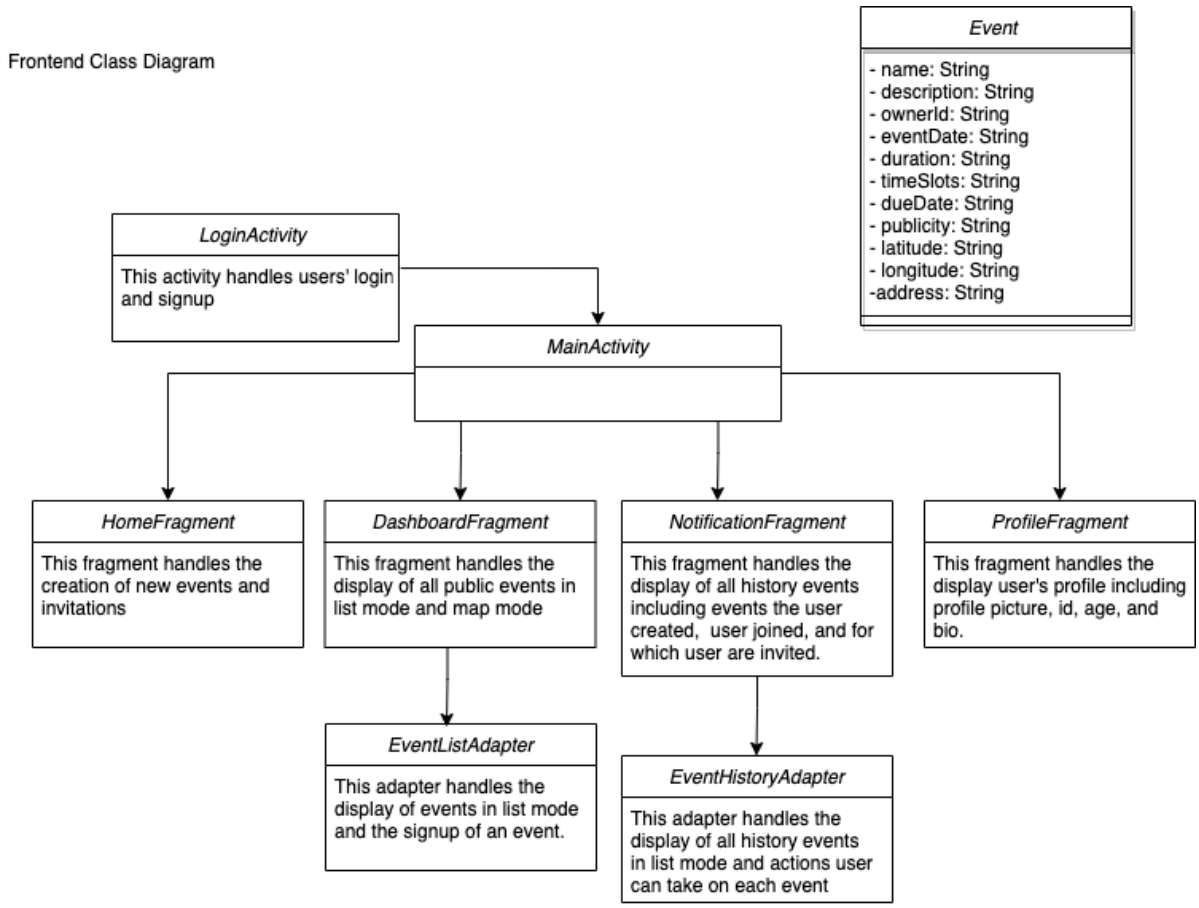
- Notification Manager ⇒

```
app.post("/add_notification", (req, res) => {
app.get("/get_notifications/:id", (req, res) => {
app.get('/determined_times', (req, res) => {
```

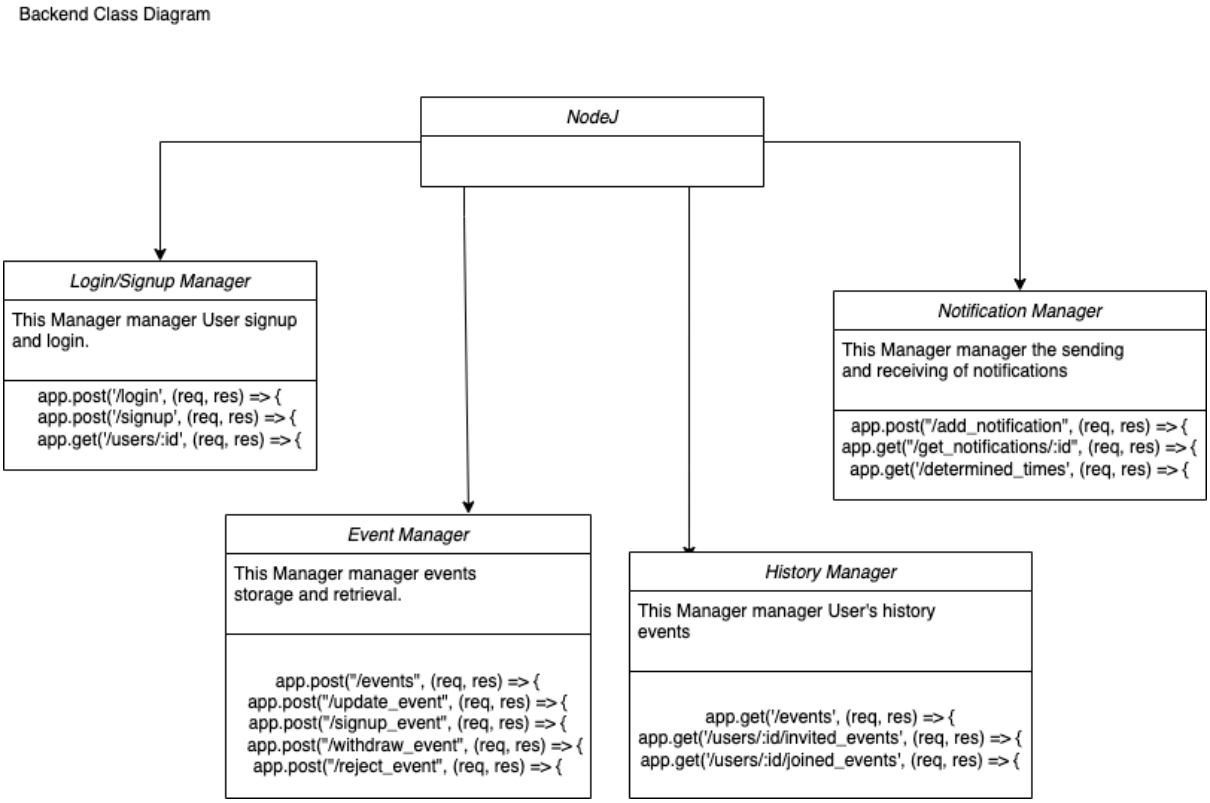
- Map Manager: Since we handle map-related functionalities on the frontend, we no longer need this component.

VI. Detailed Design Change

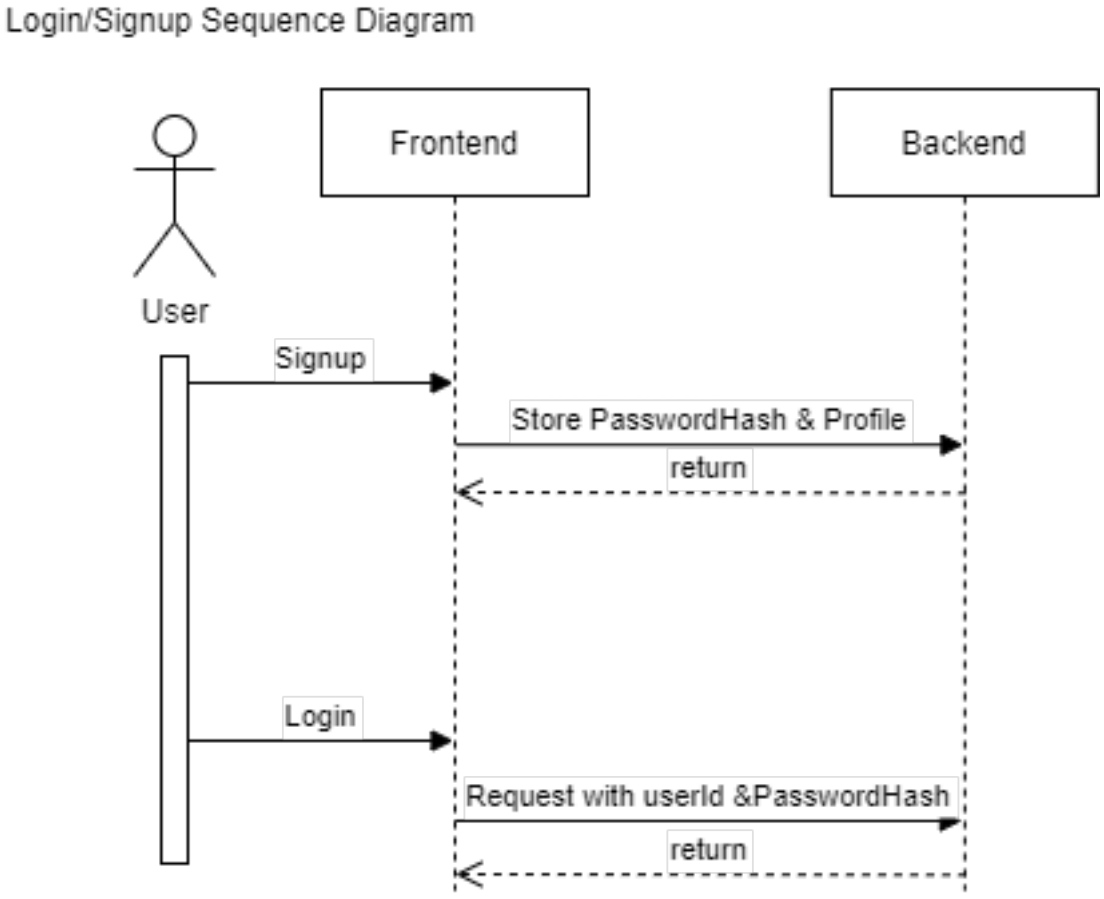
1. Frontend UML class diagram



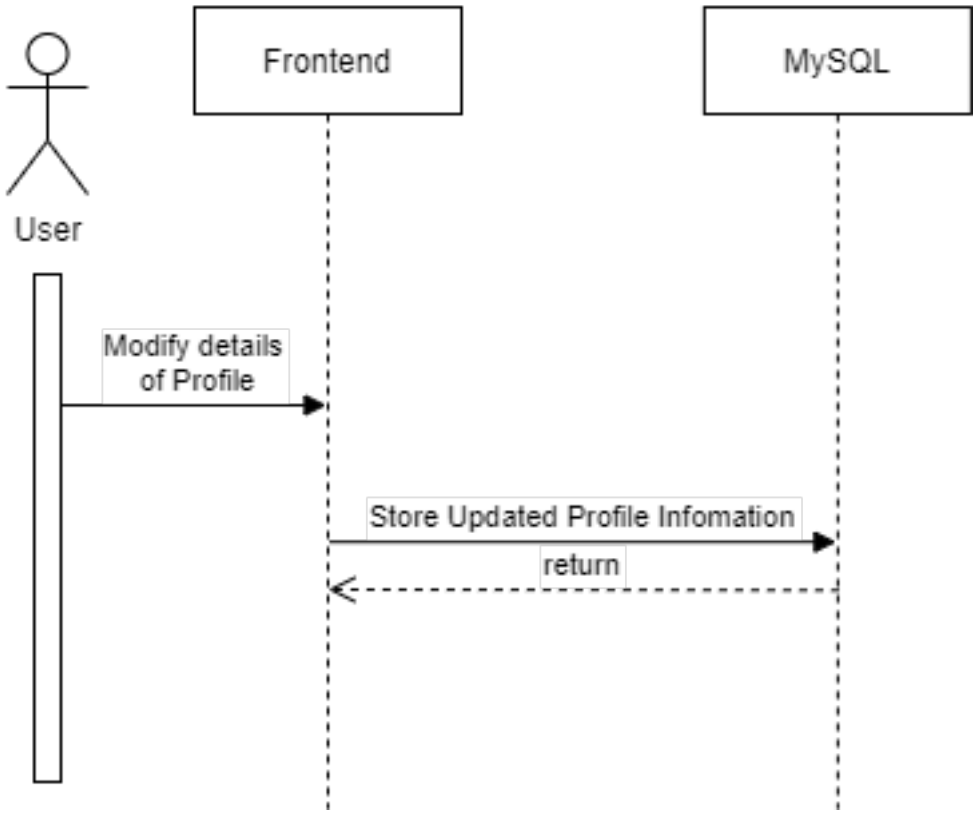
2. Backend design diagram



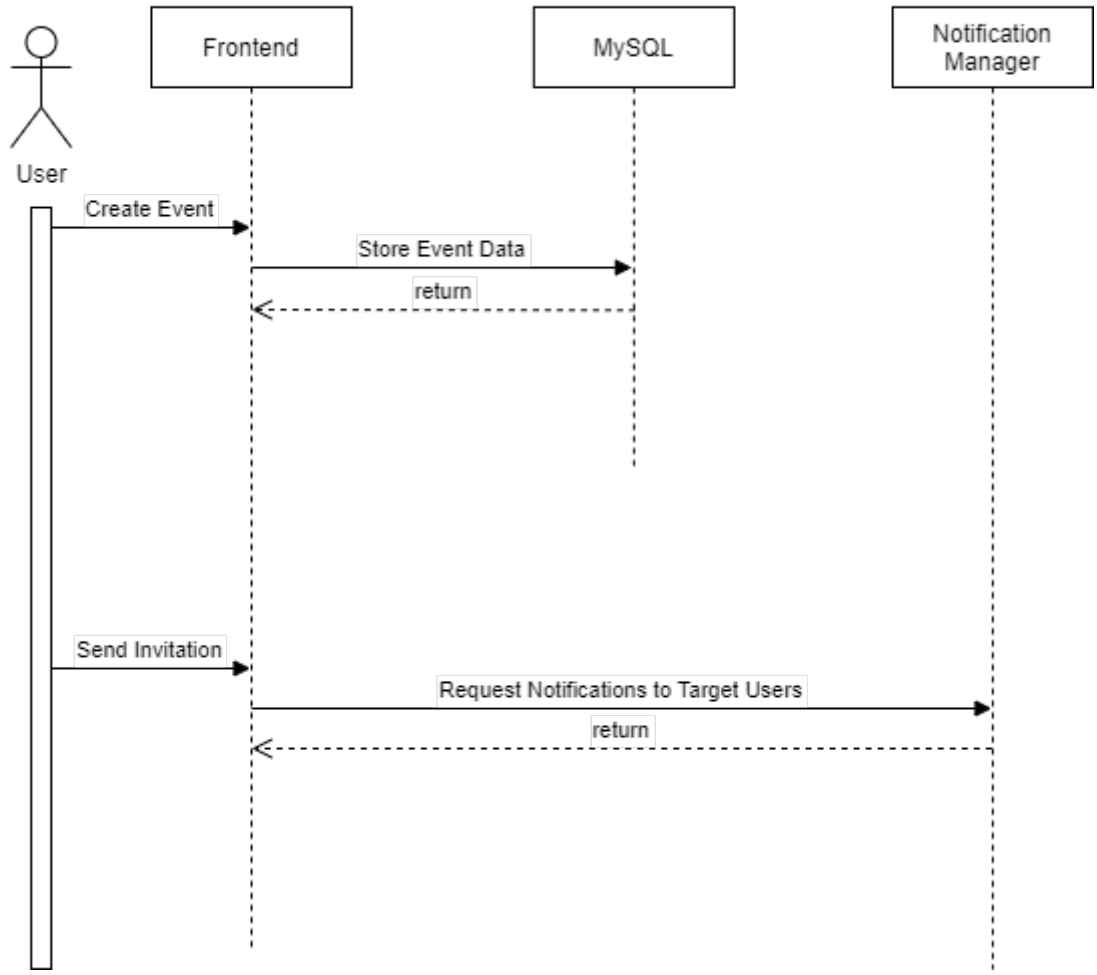
3. User case scenarios



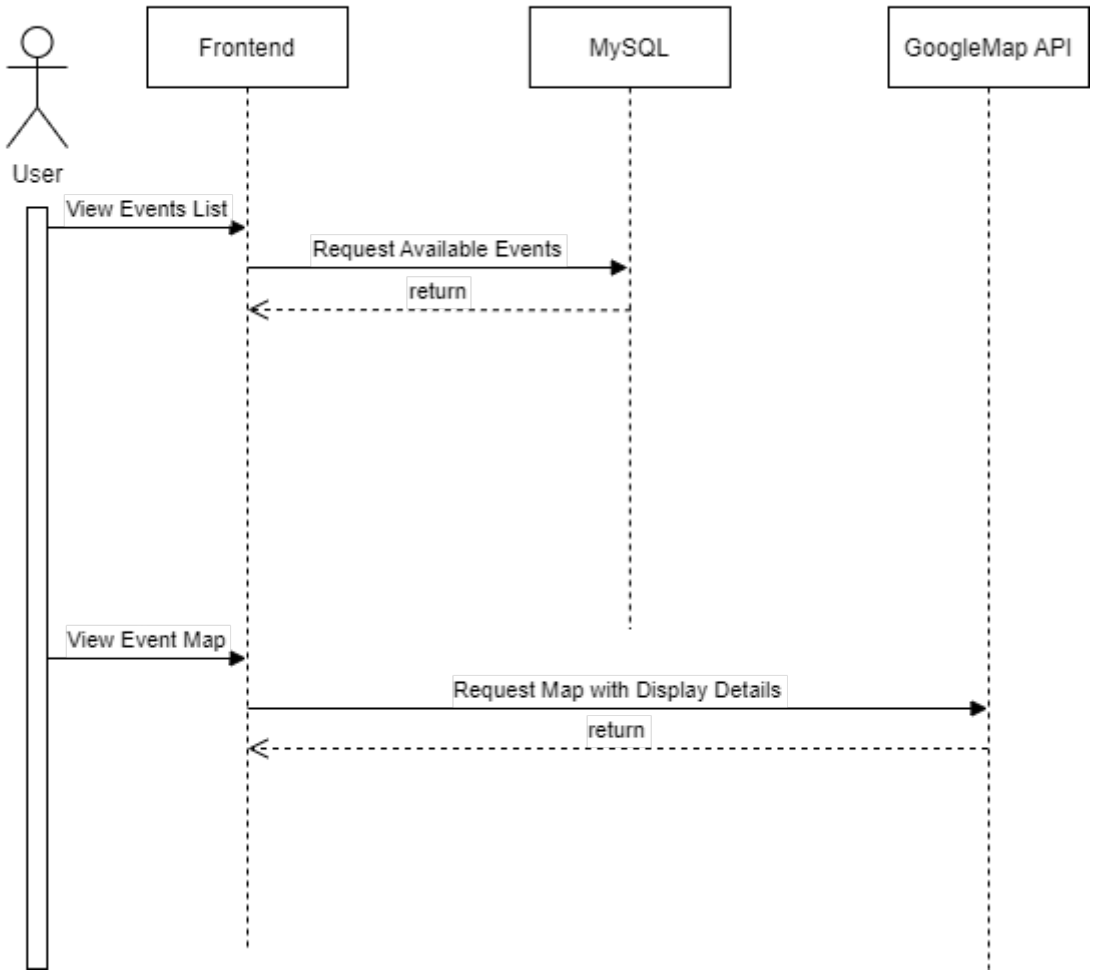
Modify Profile Sequence Diagram



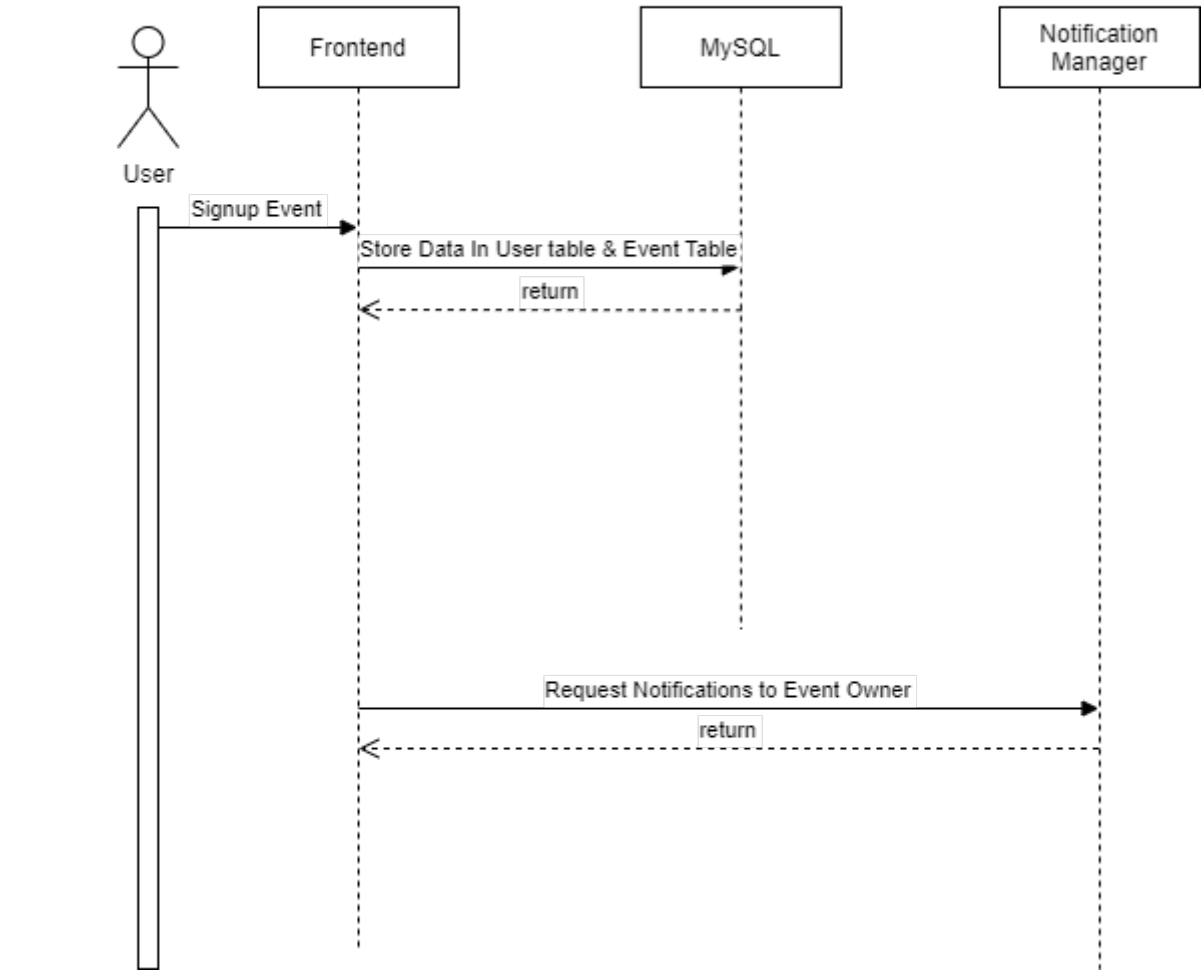
Create Event Sequence Diagram



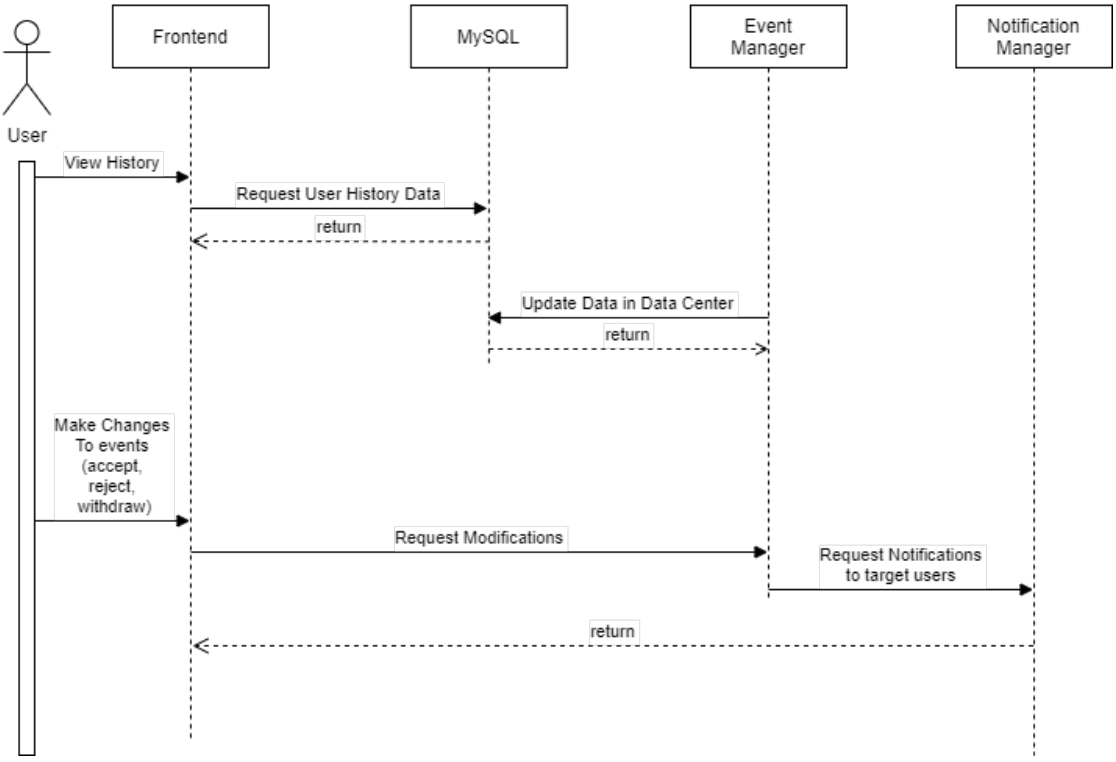
View Events Sequence Diagram



SIgnup Event Sequence Diagram



Track&Modify Event Sequence Diagram



VII. Requirements Change

1. Added missing requirement "modify user profile information". To accommodate this requirement, we added a new function to Login/Signup Manager:

```
@POST("edit_profile")
Call<Void> executeEditProfile(@Body HashMap<String, String> map);
```

2. In the registration phase, instead of letting users select profile pictures from gallery, we give them more flexibility by asking them to enter a url of the picture. The design change we made to accommodate this is adding a `String image` data member to the `Event` class to keep track of the url.
3. Rather than directly logging in users after their registration, the app will now direct them back to the log-in page and let them do the usual log-in process. This would make the design more consistent and render uniform user experience. We didn't make design change for this.
4. In the create event page, we added two more required items in addition to the ones we proposed in the requirement document: **event description** and **duration**. This gives event creators the opportunity to further elaborate on their event. We added two data members `String description` and `int duration` to the `Event` class.