# OpenJIO

Team Fivver

Tio Guo Yong | Chen Zihang | Zhang Jing Wen | Jayden Yeo He | Nema Aarushi

OPENJIO

# Table of Contents

## What is OpenJio?

OpenJio is a social networking platform for sports enthusiasts. It allows users to create open invitations of self-initiated activities to the public or join sports activities happening in their locality/ city.

OpenJio is a mobile based web-app that provides users with a 'smart way' to stay fit, deal with boredom, and build tight knit relationships through their favorite sporting activities.

## Features and requirements

1. Organize Activities

The app allows users to organize activities, which are categorized into sports, hangouts, etc.

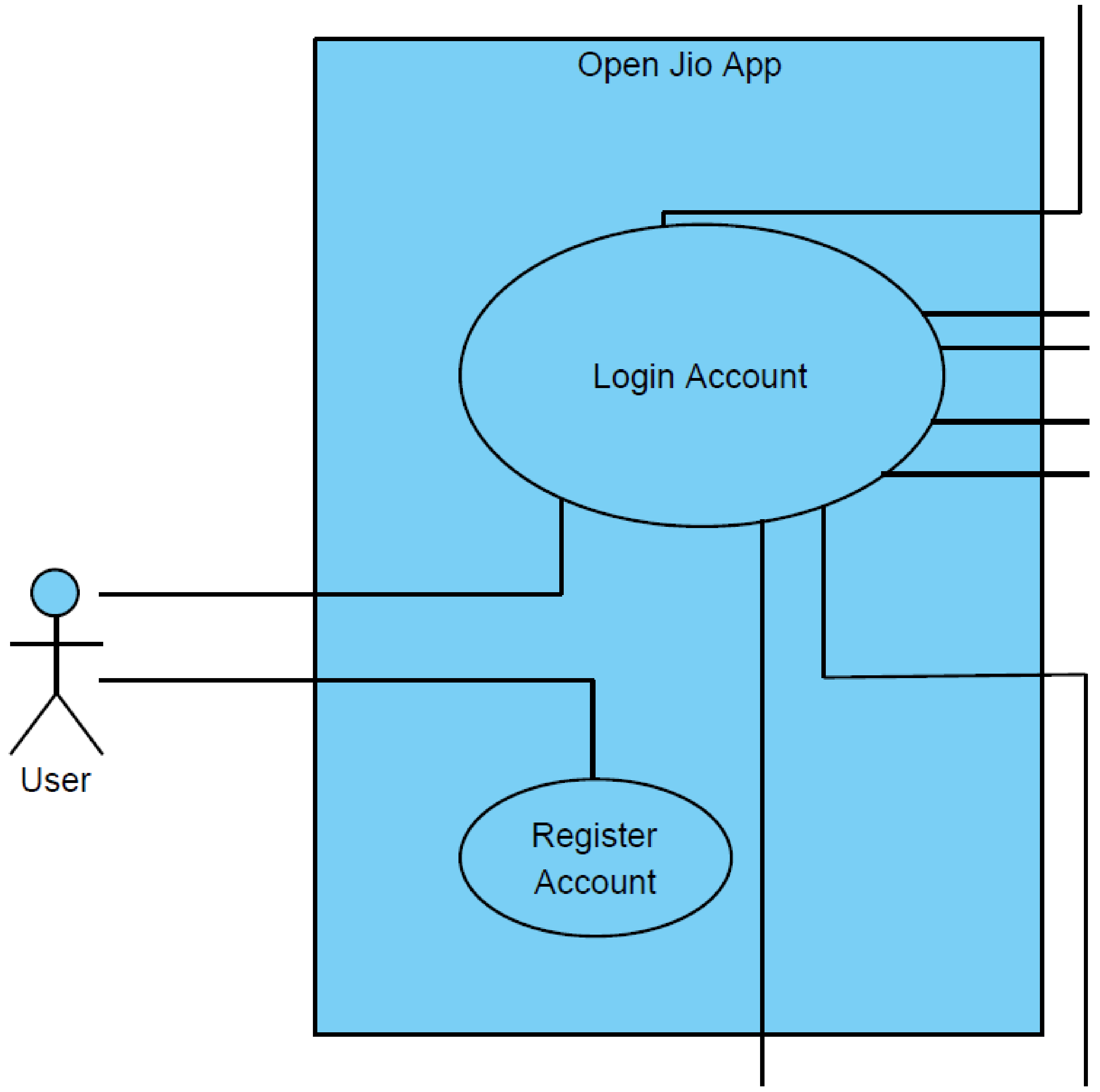
2. Explore Activities

Users can search for activities to join by category, location and time. Upcoming activities in their vicinity will also be automatically recommended to them.

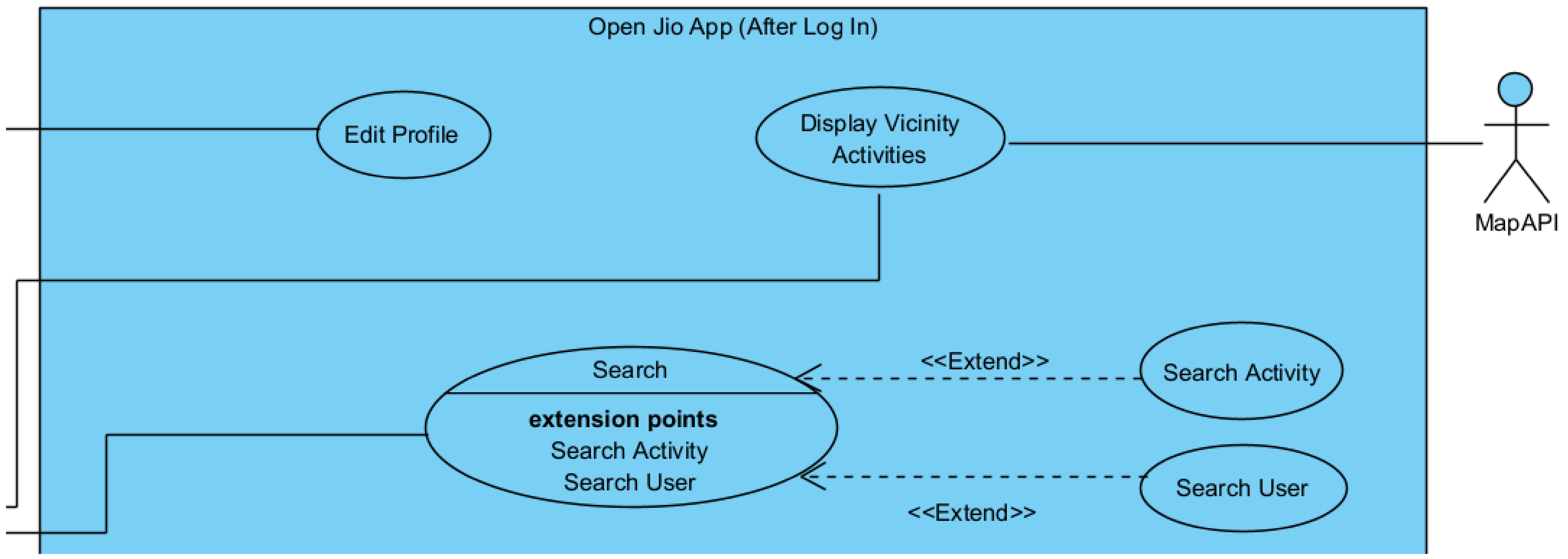
3. Friends

Users can add other users as friends, allowing them to chat and view what activities their friends are participating in.

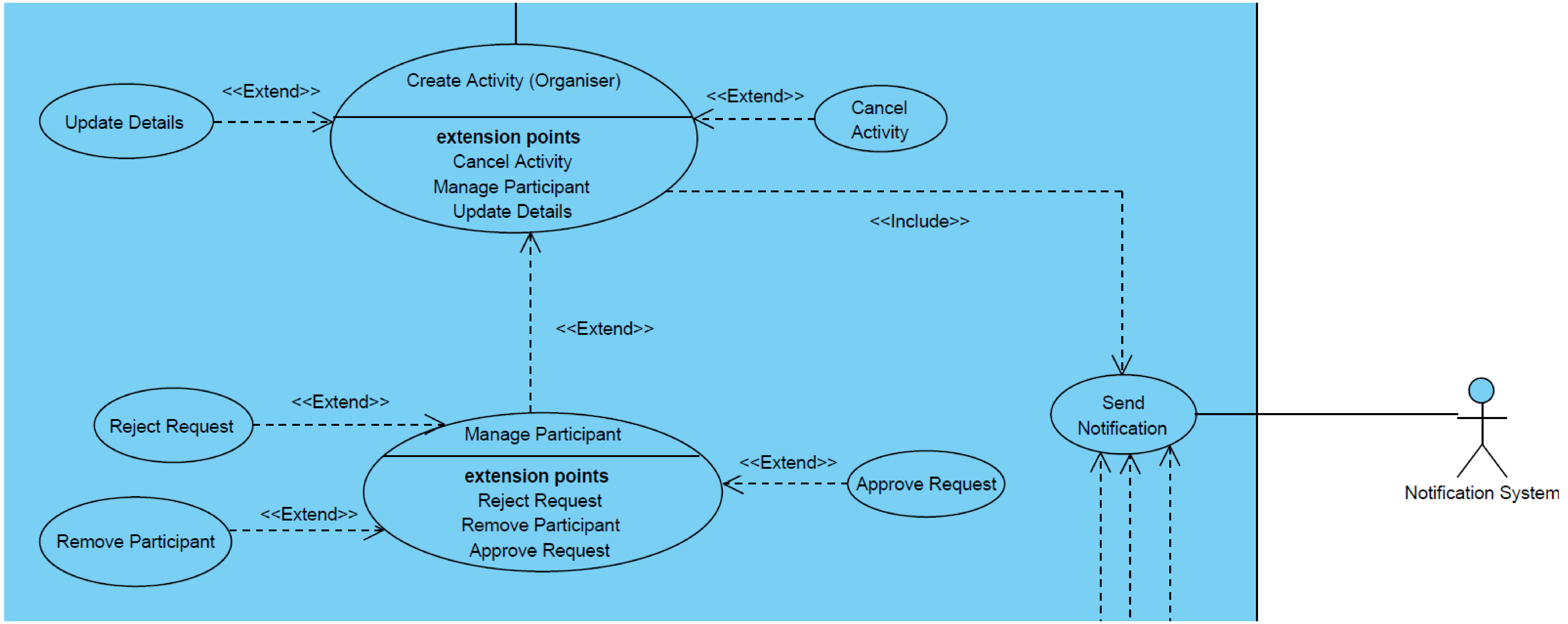
# Use-Case Diagram



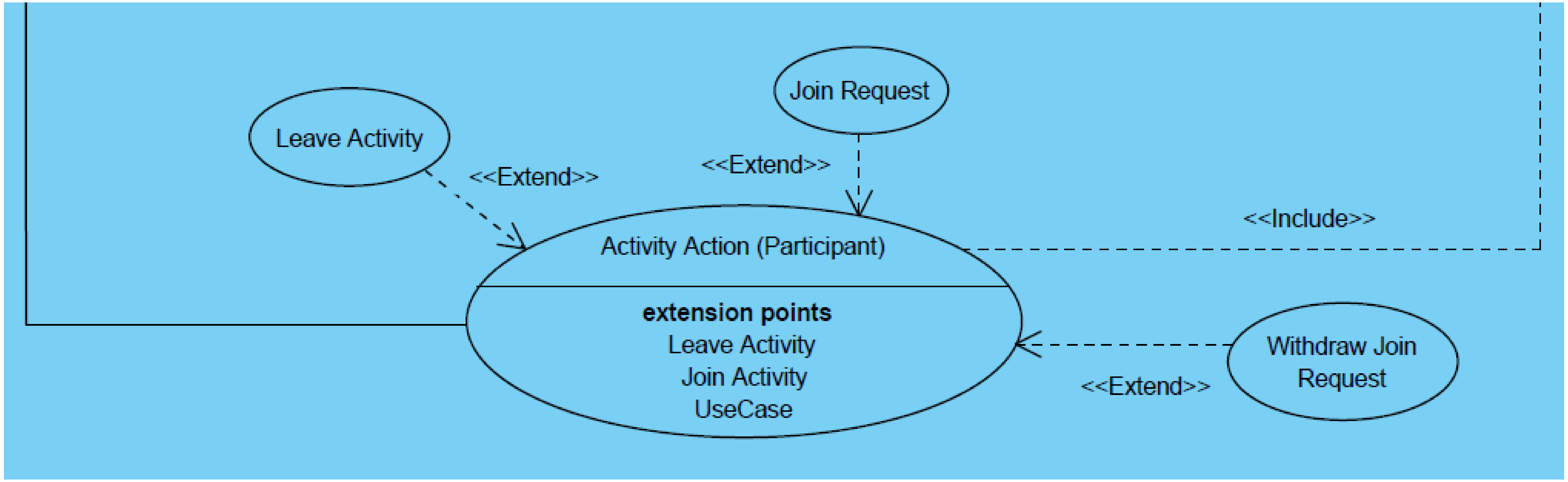
# Use-Case Diagram



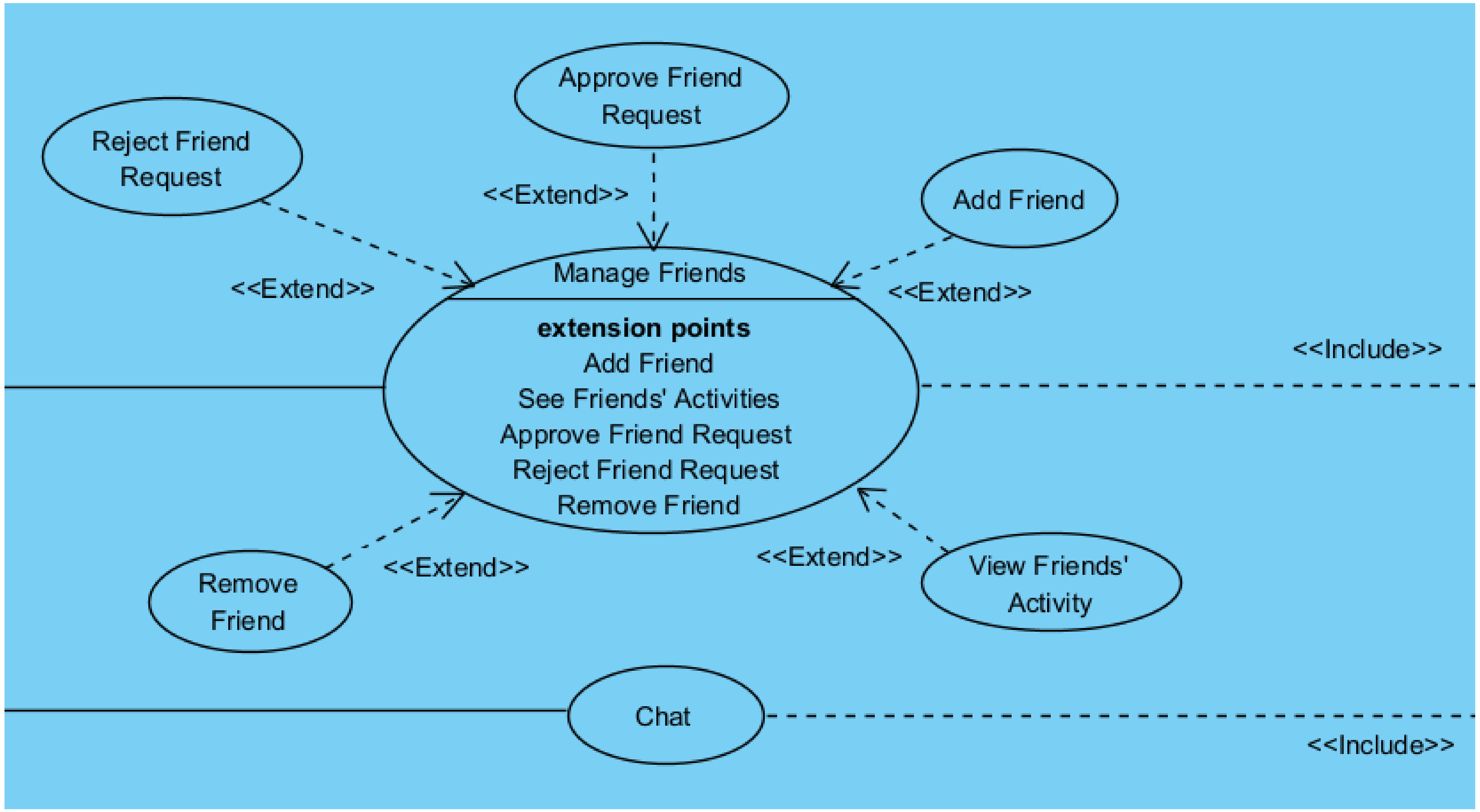
## Use-Case Diagram



# Use-Case Diagram



## Use-Case Diagram

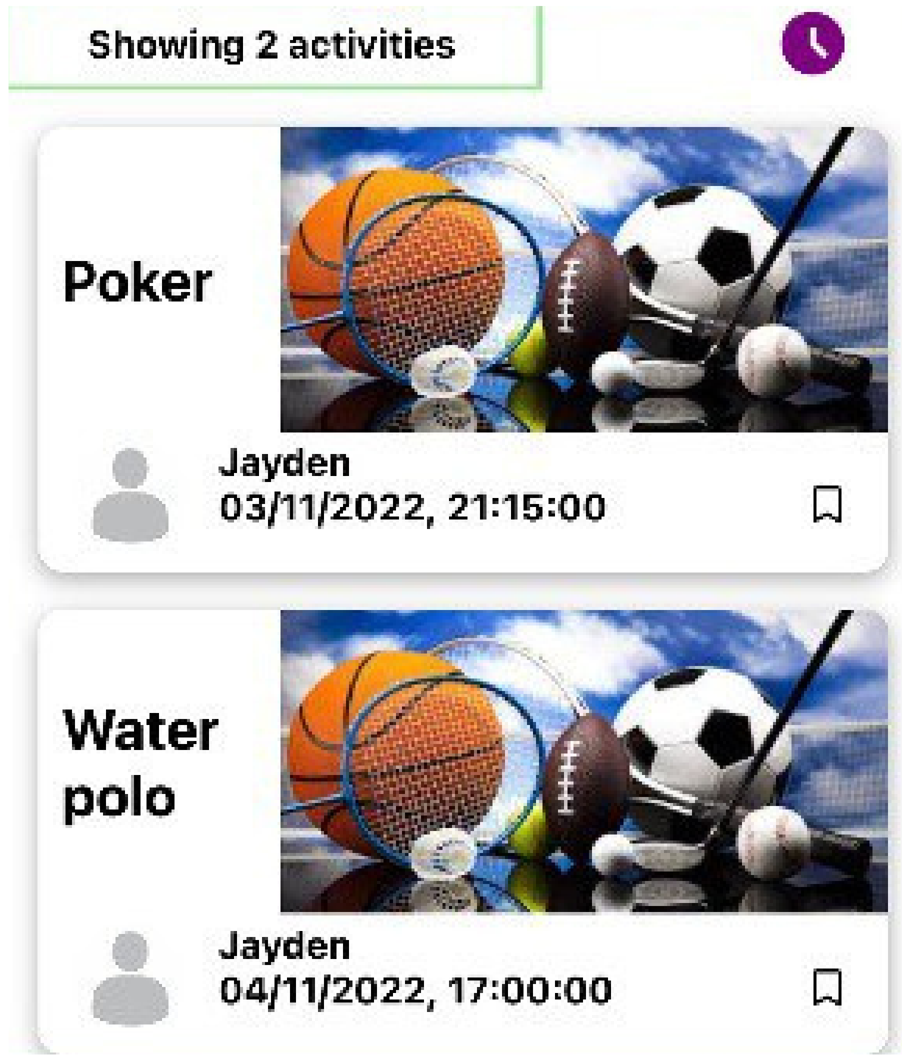


## Tech Solutions

**REACTJS**
FRONTEND

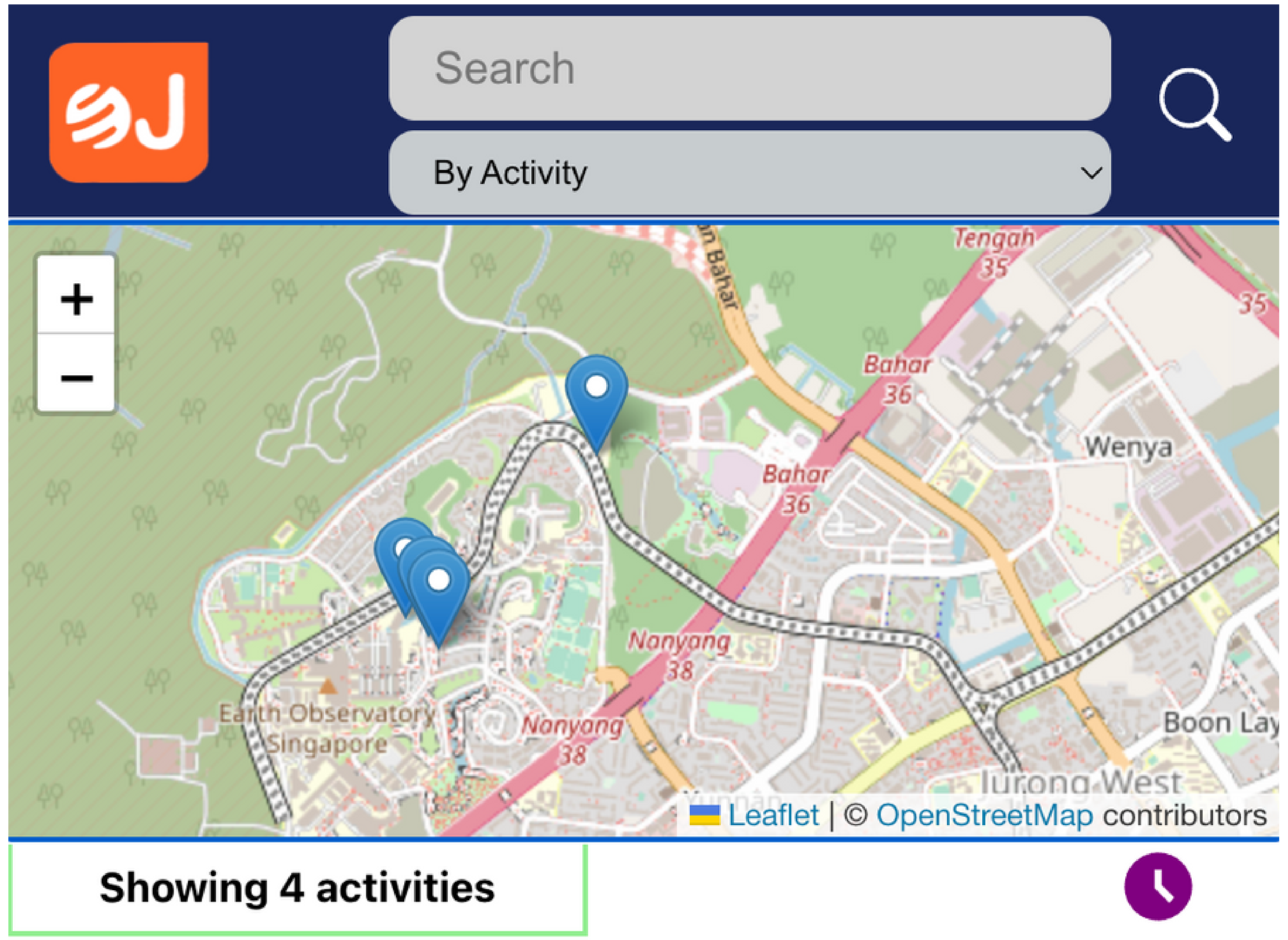
**FIREBASE**
BACKEND

# Cool Tech Features


Showing 2 activities

Poker
Jayden
03/11/2022, 21:15:00

Water polo
Jayden
04/11/2022, 17:00:00

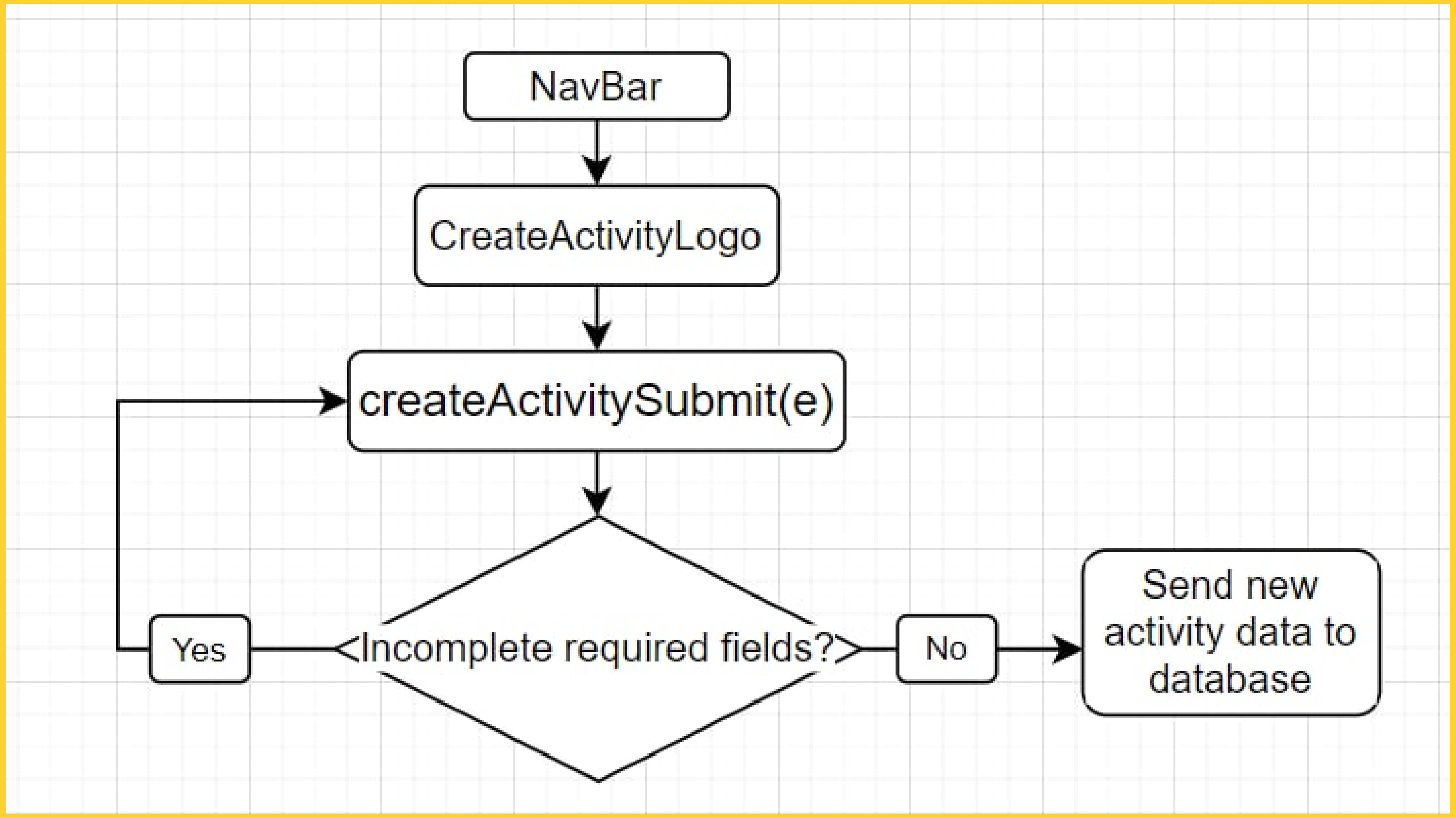

Search
By Activity
Tengah
Bahar
Wenya
Nanyang
Earth Observatory Singapore
Boon Lay
Jurong West
Leaflet | © OpenStreetMap contributors
Showing 4 activities

24/7 | Real time updates

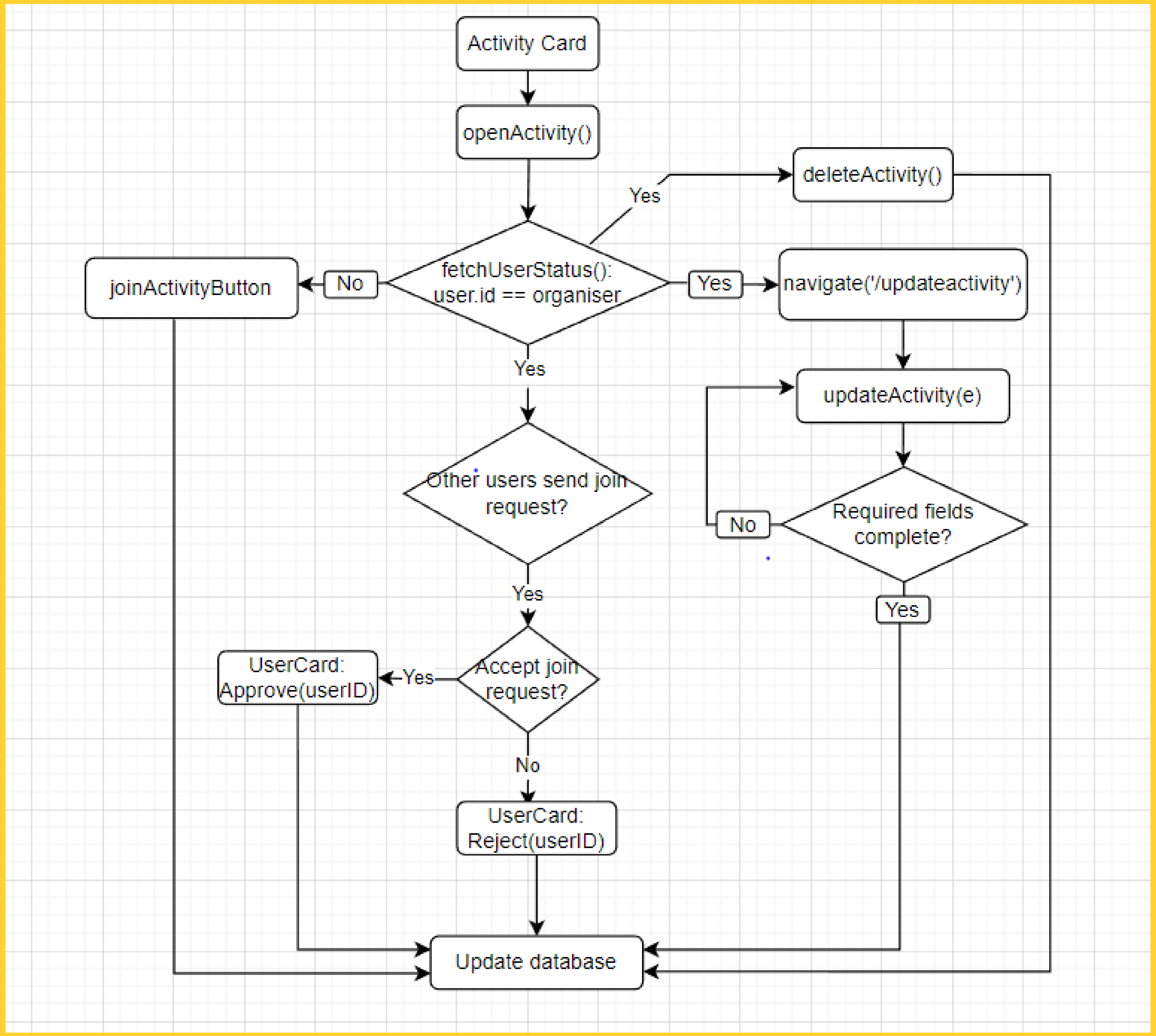
100% | Marked activities on map

## DEMONSTRATION

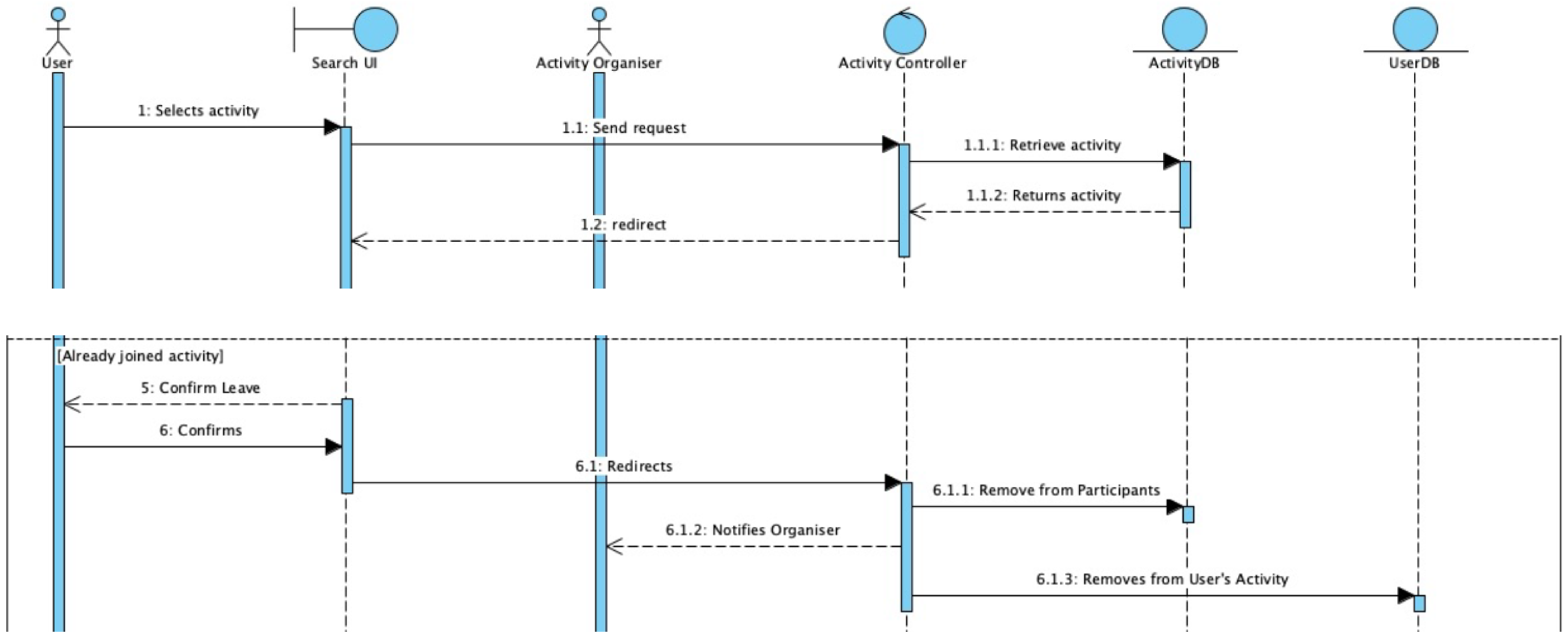
## ActivityController Control Flow - Create Activity


NavBar
CreateActivityLogo
createActivitySubmit(e)
Incomplete required fields?
Yes
No
Send new activity data to database

## ActivityController Control Flow - Update/Join/Cancel Activity and Approve/ Reject join request



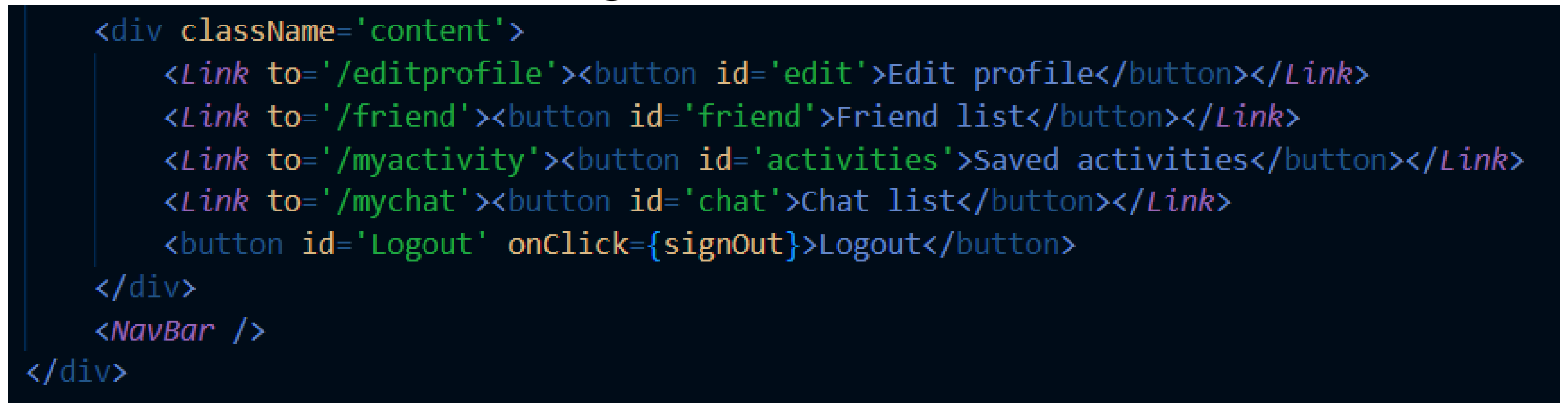
# Sequence Diagram: Join & Leave Activity



## REUSABILITY

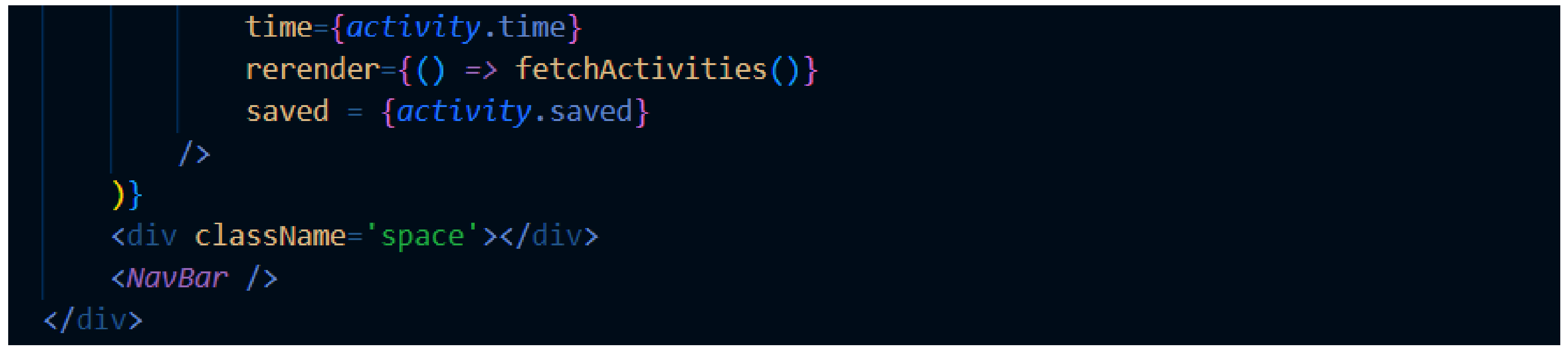
ReactJS - Component-based Framework in JavaScript.

### MyProfileUI



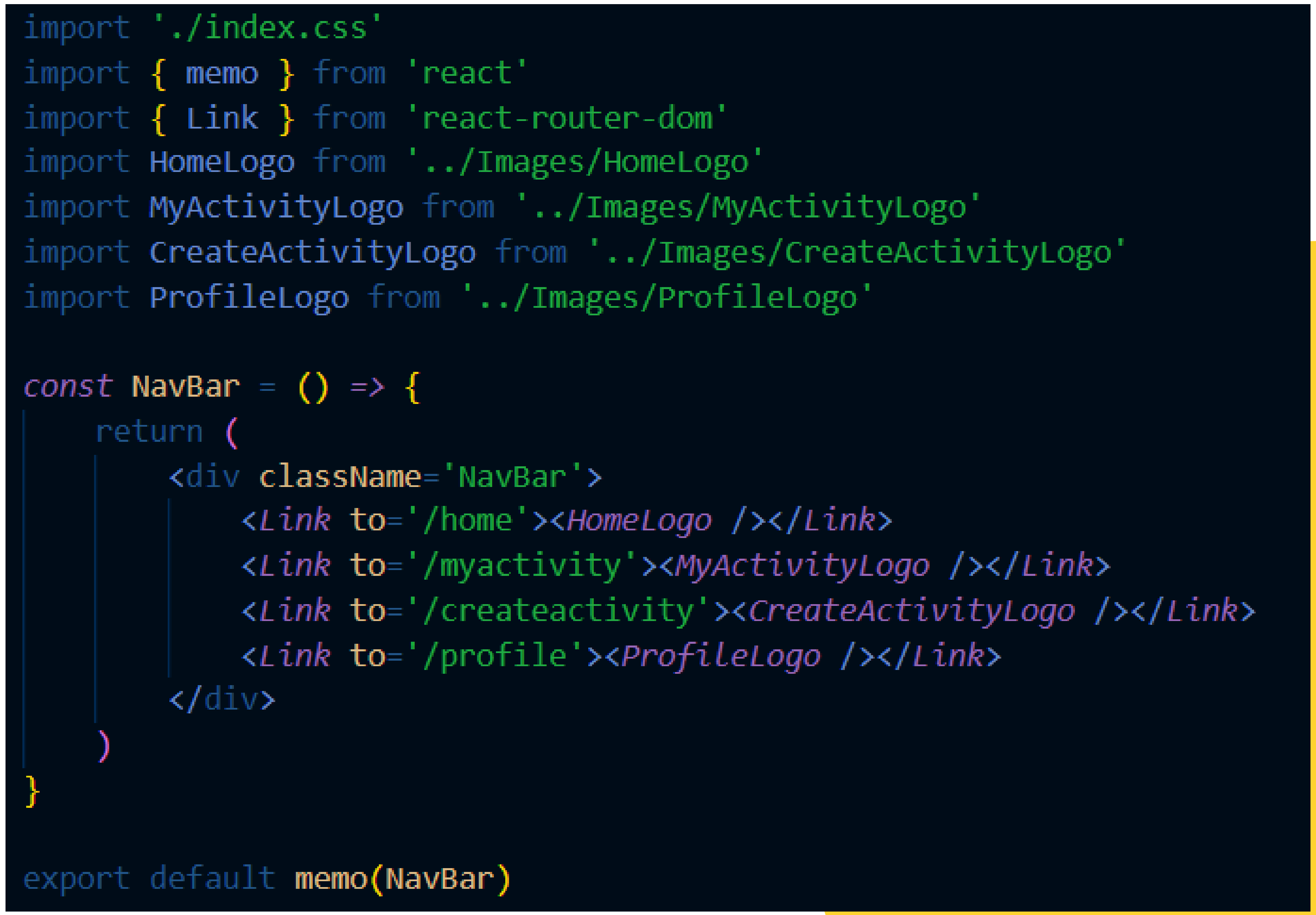
```
    <div className='content'>
        <Link to='/editprofile'><button id='edit'>Edit profile</button></Link>
        <Link to='/friend'><button id='friend'>Friend list</button></Link>
        <Link to='/myactivity'><button id='activities'>Saved activities</button></Link>
        <Link to='/mychat'><button id='chat'>Chat list</button></Link>
        <button id='Logout' onClick={signOut}>Logout</button>
    </div>
    <NavBar />
</div>
```

### HomeUI



```
            time={activity.time}
            rerender={() => fetchActivities()}
            saved = {activity.saved}
        />
    )}
    <div className='space'></div>
    <NavBar />
</div>
```

### Code Once, Multiple Usage



```
import './index.css'
import { memo } from 'react'
import { Link } from 'react-router-dom'
import HomeLogo from '../Images/HomeLogo'
import MyActivityLogo from '../Images/MyActivityLogo'
import CreateActivityLogo from '../Images/CreateActivityLogo'
import ProfileLogo from '../Images/ProfileLogo'

const NavBar = () => {
    return (
        <div className='NavBar'>
            <Link to='/home'><HomeLogo /></Link>
            <Link to='/myactivity'><MyActivityLogo /></Link>
            <Link to='/createactivity'><CreateActivityLogo /></Link>
            <Link to='/profile'><ProfileLogo /></Link>
        </div>
    )
}

export default memo(NavBar)
```

# TRACEABILITY

From and To Requirement Across SDLC

## Requirement

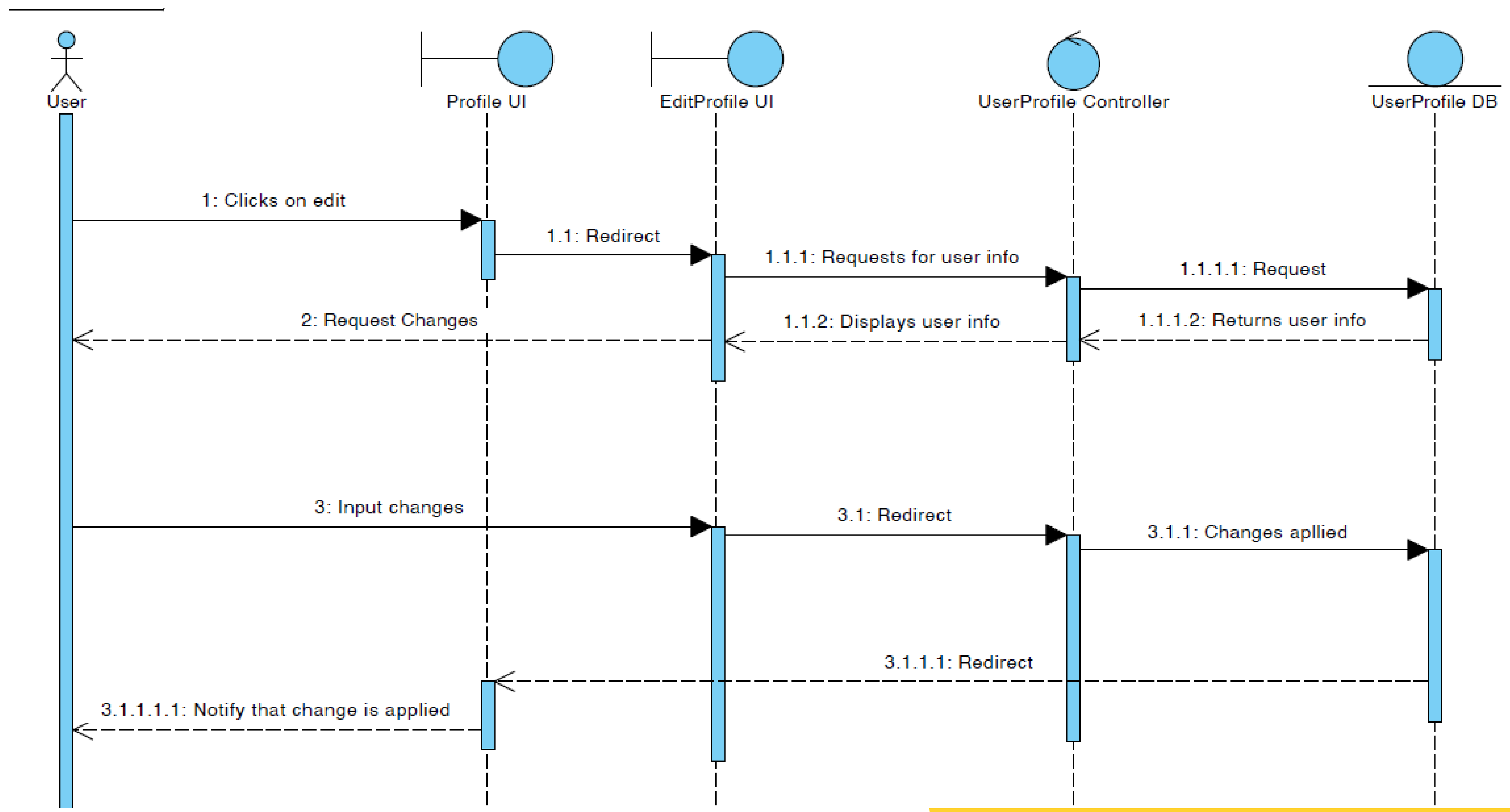
**Edit Profile**

**Description:** The app must allow the current user to edit the information displayed in their own profile page.

**Input:** Profile photo, username, name and/ or description.

**Output:** Profile information updated.

## Design

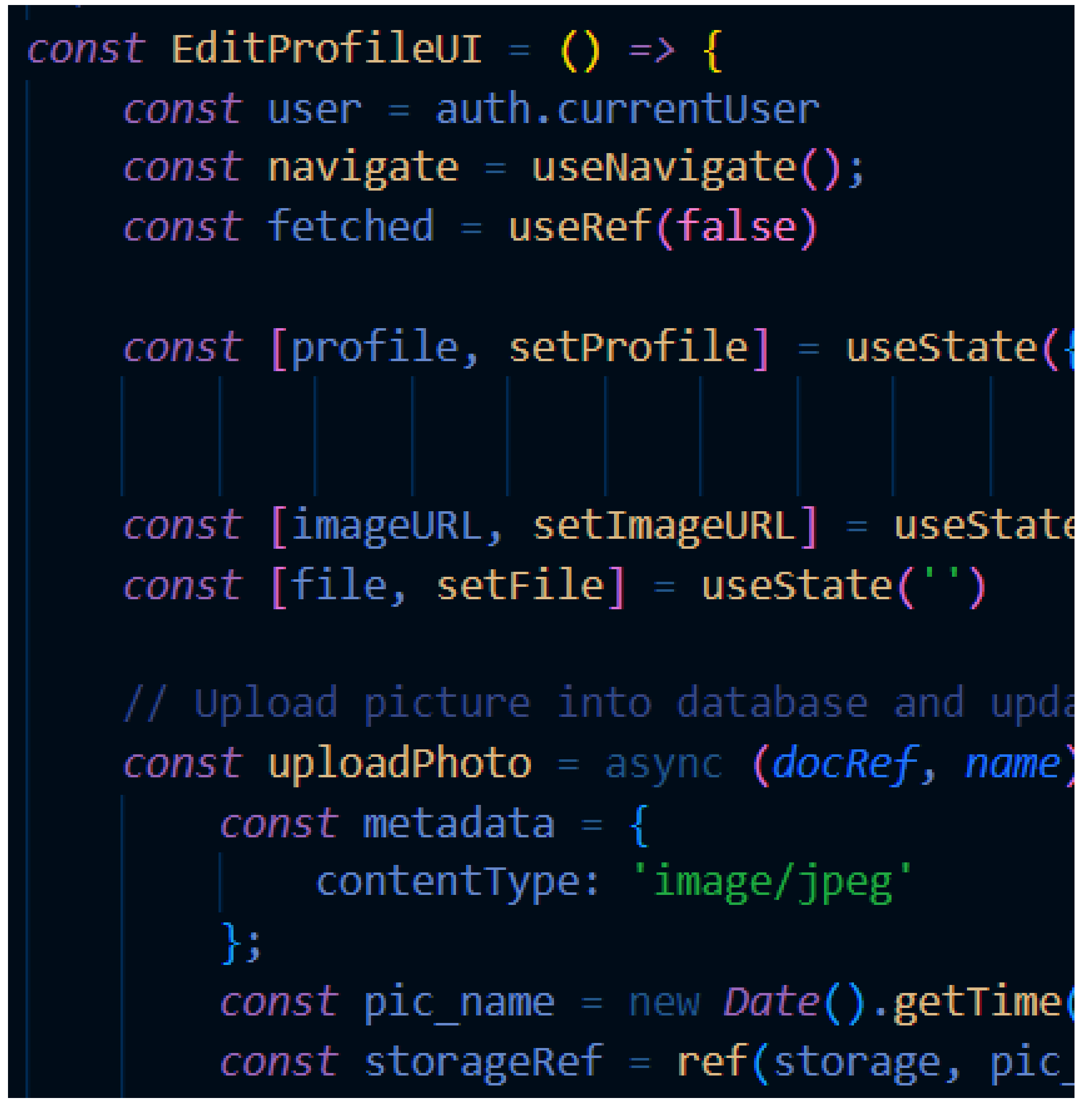


HOME OVERVIEW DEMO SE PRACTICES & FUTURE SP

## TRACEABILITY

From and To Requirement Across SDLC

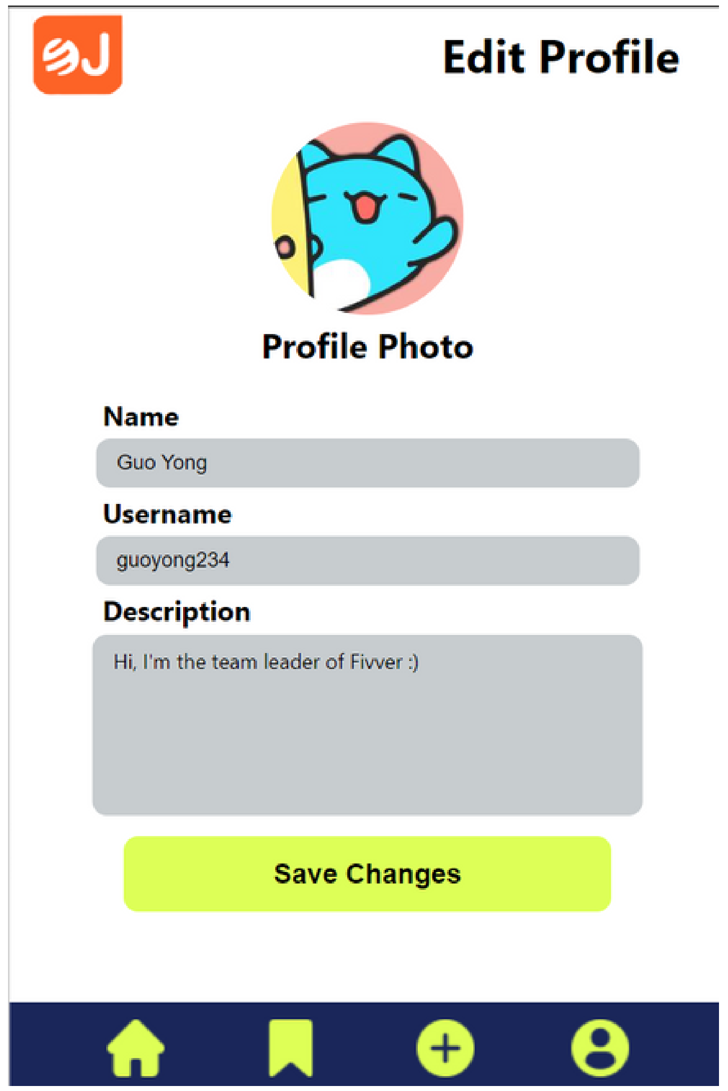
### Implementation


```
const EditProfileUI = () => {
    const user = auth.currentUser
    const navigate = useNavigate();
    const fetched = useRef(false)

    const [profile, setProfile] = useState(

    const [imageURL, setImageURL] = useState
    const [file, setFile] = useState('')

    // Upload picture into database and upda
    const uploadPhoto = async (docRef, name)
        const metadata = {
            contentType: 'image/jpeg'
        };
        const pic_name = new Date().getTime(
        const storageRef = ref(storage, pic_
```


**Edit Profile**

Profile Photo

Name: Guo Yong

Username: guoyong234

Description: Hi, I'm the team leader of Fivver :)

### Testing

**14. Functionality: Edit Profile**

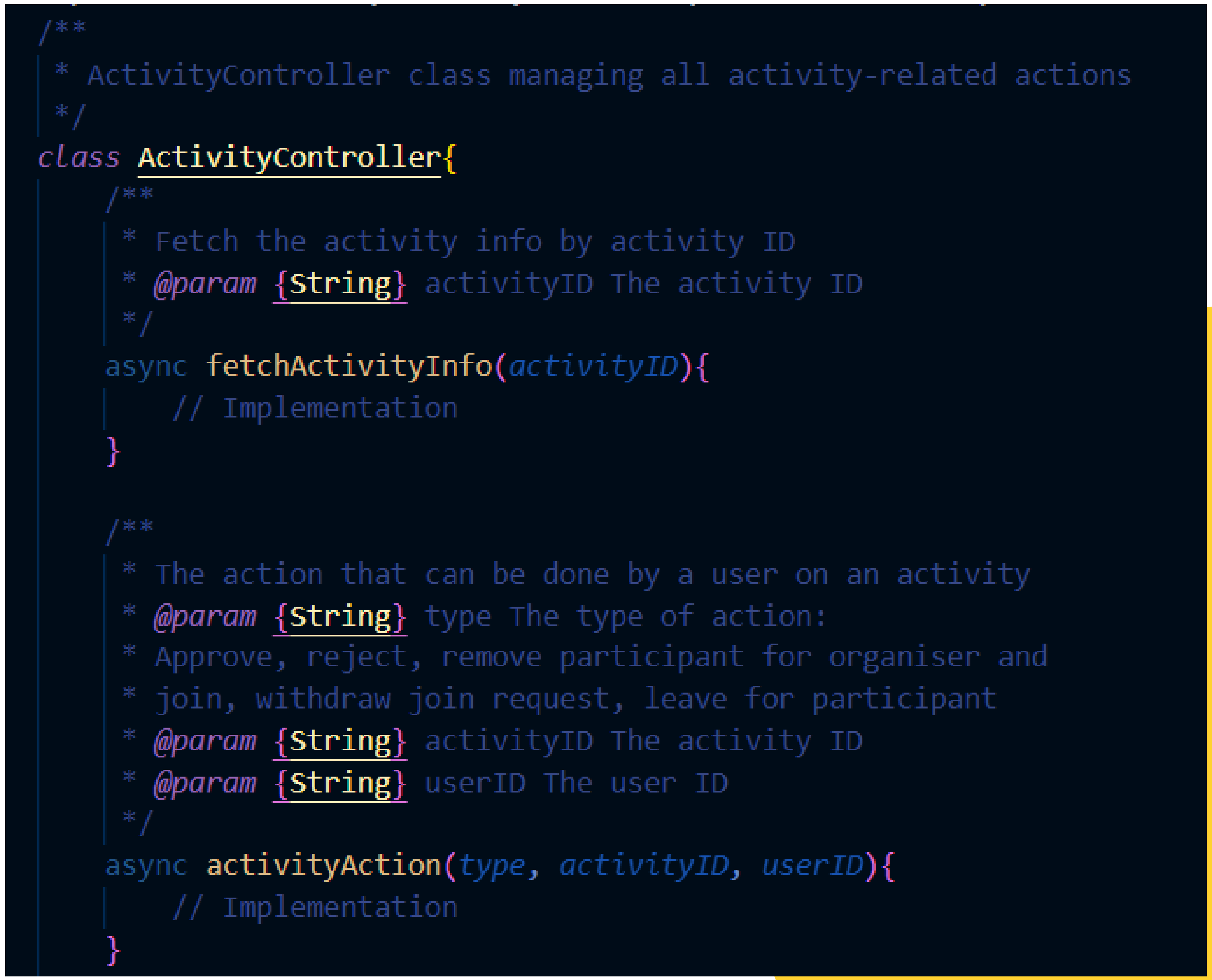
b. Specific cases:

| Test ID | Test Case | Expected Result | Actual Result |
|---|---|---|---|
| 13.1 | Input:<br>1. Username: Julie123 → EMPTY | The system prompts the user to enter missing field | The system prompts the user to enter missing field |
| 13.2 | Input:<br>2. Username: Julie123 → Julie12345 | When user views 'My Profile', updated fields are displayed | When user views 'My Profile', updated fields are displayed |

# EXTENSIBILITY

$$\begin{tikzcd}[row sep=large]
\text{ActivityUI/Any Boundary Class} \arrow[d, shift left, "\text{Request}"] \\
\text{ActivityController (Methods)} \arrow[u, shift left, "\text{Data}"] \arrow[d, shift left, "\text{Request}"] \\
\text{Activity/Any Entity Class} \arrow[u, shift left, "\text{Data}"]
\end{tikzcd}$$

## Boundary ⇔ Control ⇔ Entity



```
/**
 * ActivityController class managing all activity-related actions
 */
class ActivityController{
    /**
     * Fetch the activity info by activity ID
     * @param {String} activityID The activity ID
     */
    async fetchActivityInfo(activityID){
        // Implementation
    }

    /**
     * The action that can be done by a user on an activity
     * @param {String} type The type of action:
     * Approve, reject, remove participant for organiser and
     * join, withdraw join request, leave for participant
     * @param {String} activityID The activity ID
     * @param {String} userID The user ID
     */
    async activityAction(type, activityID, userID){
        // Implementation
    }
```

OpenJIO

Team Fivver

## Thank You!