Bilkent University

Department of Computer Engineering

Senior Design Project

Project short-name: Overfind

Analysis Report

Group Members:
Asena Rana Yozgatlı
Bartu Soykök
Burak Şenel
Mehmet Emre Arıoğlu

Supervisor:
Prof. Dr. Varol Akman

Jury Members:
Prof. Dr. Özgür Ulusoy
Assoc. Prof. Dr. Çiğdem Gündüz Demir

Innovation Expert:
Bora Güngören (Portakal Teknoloji)

November 6, 2017

This report is submitted to the Department of Computer Engineering of Bilkent University in partial fulfillment of the requirements of the Senior Design Project course CS491/2.

1. Introduction

We are living in an era of information. Every one of us is bombarded with information around us in a sense of intensity. Events are happening in a specific time frame so it is important to notice the events in a timely manner. Most of the people don’t have the time to look for events according to their interests or profession and sometimes they miss interesting events due to this intense information flow.

The aim of our project is to utilize different event platforms namely Eventbrite, Biletix by parsing information about events to make recommendation to our app’s users. Overfind users will be asked to fill a survey at the beginning of the application and later on they will be recommended events according to this survey and their following feedback. User interests will be used as labels and users will be able to search for events and they will be able to look at the recommended events which they can ask for more recommendation.

Overfind will use natural language processing with text retrieval from event platforms. By using natural language processing, we aim to analyze the text in event descriptions and get meaningful results. We plan to use machine learning algorithms for generating event labels and for matching event labels with user interest labels in order to give recommendations.

This report includes the scope and constraints of the project which determines the boundaries of the problem. We will define subproblems and explain how our solution will address them. We will explain the project boundaries and constraints which are needed to complete the project.

2. Current system

2.1. Eventbrite

Eventbrite is an online event organizing service [1]. Users have two roles in eventbrite system: organisers and attendants.

Organisers can create events and sell the tickets of these events if they wish to. Organisers can provide event details such as name, description, location, date, time, price, seating plan, etc. Organisers are encouraged to choose a topic and a subtopic for their events. These fields are treated as labels and are predefined. Organisers can promote their events using Eventbrite services.

Attendants can purchase tickets of the events found on eventbrite's database. They can see their purchased events. They are also given some recommendations based on the events they attended.

Recommendation service of Eventbrite considers both the event taste of attendants and the promotional need of organisers.

Our system, Overfind, will not allow the creation or promotion of events. It will search the web and display events that are publicly available. Overfind will create and assign its own labels to events based on the information provided. Overfind will use this self-generated information as labels of events. Overfind will address only the event taste of attendants (users of our system), in order to provide more personalised recommendations.

2.2. Event Recommendation Engine

This software[2] is designed as a model for the event recommendation engine challenge on Kaggle[3].

The software uses datasets including users' friends, friends' likes, their interests, etc. This includes rating of the neighborhood that consists of users with common interests such as information of who is attending which events, location similarities of events that are attended by users in that neighbourhood, and event similarities. This is an example of collaborative filtering model with user based approach.

Overfind will use content based filtering which is focused on using existing user profiles and getting information about users by initial surveys. In this process we compare events that user rated positively with the events that aren’t yet rated by users.

3. Proposed system

3.1. Overview

Overfind is an event recommendation system which will consist of two parts, a mobile(client) application and a server application. System will recommend events to users based on their personal information. Client application will collect the initial user data by making the user fill a small survey, it will collect the rest of user data from user feedback. Server application will collect event information from various websites.

Client will assign labels (tags) to user and change the values of these tags according to user feedback. It will request recommendations from server periodically. Server will parse the event data and generate tags. Server will store this data as a weighted graph, where the nodes represent tags and the weight values represent the relevancy between two tags. Server will get user data from client and generate recommendations using the tag relevancy information. Server will not store user data, it will only store event data.

System will also offer some functionalities to improve the quality of usage. User can choose an event that fits his/her interests to see the event details. User can open external links to websites of individual events if available. User can ask for more recommendations. User can search the database for events. User can can sort the search results by date or relevance. User can change the language and set a default location.

3.2. Functional Requirements

3.2.1. User Requirements

- Users should be able to see event details (title, description, date, time, location, relevant keywords).
- Users should be able to search for events by its details.
- Users should be able to see and open any external links of the events.
- Users should be able to mark events as interested or not interested.
- Users should be able to see the list of events that they are interested.
- Users should be able to see a list of recommended events.
- Users should be able to request a new lesser relevant list.
- Users should be able to mark the recommendations as good or bad.
- Users should be able to set and change their location preferences.
- Users should be able to see the results of his/her query sorted by date or relevance.

3.2.2. Client Application Requirements

- Client should be able to send query requests to server and display the received data.
- Client should be able to open links in web browser.
- Client should be able to store and modify user data.
- Client should be able to request recommendations from server.
- Client should be able to learn from user feedback (marked events and recommendations).

3.2.3. Server Application Requirements

- Server should be able to query and modify the database.
- Server should be able to give recommendations based on user data.
- Server should be able to get event information from external sources periodically.
- Server should be able to detect the language of events.
- Server should be able to classify events.

3.3. Nonfunctional Requirements

3.3.1. Confidentiality

- Users personal data will only be available to the system other than user him/herself.

3.3.2. Usability

- Functionalities of the system will be easily accessible through the mobile application.
- Functionalities of the system will be well defined and manageable in number for ease of use.

3.3.3. Scalability

3.3.3.1. Memory scalability

- Even though our database have a little amount of tables, the event and tag table will be huge. We can use our space efficiently by storing only needed information. If needed we can compress our archived events or add more hardware.

3.3.3.2. Performance scalability

- We expect our graph to converge to a certain size. We need an efficient graph traversing algorithm to evaluate our user data. Since the client application will request recommendations we will not need all of the graph. So we can only work with the relevant parts and return the result much faster.

3.3.4. Installability

- Mobile application will require Android OS version 4.4 (KitKat) or later.
- Mobile application can be downloaded from the Play Store.

3.4. Pseudo requirements

- System will consist of server application and mobile application.
- Database will be developed using MySQL.
- Backend services will be developed using Java with REST architecture.
- Mobile application will be developed for Android OS.
- Mobile application will be developed using Android Studio.
- Mobile application will be developed in Java.

3.5. Limitations

- System will support only English and Turkish languages.
- Event information will be limited to what we can gather from Eventbrite, Biletix etc.
- Initially, user information will be limited to the information gathered by a short survey.

3.6. System models

3.6.1. Scenarios

3.6.1.1. Fill Survey

Scenario Name: Fill survey

Primary Actor: User

Entry Condition: User has started the application for the first time.

Exit Condition: User has answered all of the questions in the survey.

Event Flow:

- User has opened the application.
- A quick survey is served to user.
- User has successfully finished the survey.
- User data is stored successfully.

3.6.1.2. Show Recommended Events

Scenario Name: Show recommended events

Primary Actor: User

Entry Condition: User chooses to view recommended events.

Exit Condition: Recommended events for user are listed.

Event Flow:

- User starts the application.
- User taps events button is tapped on homepage.
- Recommended events are listed.

3.6.1.3. Show Event Details

Scenario Name: Show event details

Primary Actor: User

Entry Conditions: User is browsing an event list (search results or recommendations)

Exit Condition: User has viewed details of a specific event.

Event Flow:

- User has tapped on an event to see its details.
- Event details are shown to user.

3.6.1.4. Mark Recommendation As Not-Interested

Scenario Name:	Mark recommendation as not-interested
Primary Actor:	User
Entry Condition:	User successfully opened homepage.
Exit Condition:	User marked an event as not-interested.

Event Flow:

- User has chosen recommended events option from homepage.
- List of recommendations are shown to user.
- User has tapped on an event to see its details.
- Event details are shown to user.
- User has marked the event as not-interested.
- User data is updated.

3.6.1.5. Ask For New Recommendations

Scenario Name:	Ask for new recommendations
Primary Actor:	User
Entry Condition:	User is in display mode.
Exit Condition:	User is shown a new recommendation list.

Event Flow:

- User opened homepage.
- User has chosen to view recommended events.
- User made a request by tapping “ask for more recommendations”.
- A new list of recommendations are shown to user.

3.6.1.6. Search Events

Scenario Name:	Search events
Primary Actor:	User
Entry Condition:	User is on search page.
Exit Condition:	Search results are shown to user.

Event Flow:

- User types a keyword in search bar
- User is shown the results of his/her query.

3.6.1.7. Advanced Search Events

Scenario Name:	Advanced search events
Primary Actor:	User
Entry Condition:	User is on search page.
Exit Condition:	Search results are shown to user.

Event Flow:

- User selects one or more of the following from search options:
    - Location
    - Date
    - Title
    - Relevant tags
- User types the details in corresponding areas.
- User is shown the results of his/her query.

3.6.1.8. Change The Order Of Results

Scenario Name:	Change the order of results.
Primary Actor:	User
Entry Condition:	User has made a successful search.
Exit Condition:	Search results are reordered.

Event Flow:

- User types a keyword in search bar.
- Results are shown as a list to user.

- User taps sort by date option.
- Results are returned in ordered by date.

3.6.1.9. Open Weblink

Scenario Name: Open weblink

Primary Actor: User

Entry Condition: User is browsing an event list (search results or recommendations)

Exit Condition: Weblink is successfully opened.

Event Flow:

- User has found an event which interests him.
- User has tapped on an event to see its details.
- Event details are shown to user.
- User has tapped to the link provided.
- Weblink is opened in a browser.

3.6.1.10. Change Language

Scenario Name: Change language

Primary Actor: User

Entry Condition: User decides to change language of the application.

Exit Condition: User changes language successfully.

Event Flow:

- User taps the settings button.
- User chooses either English or Turkish.
- User preferences are changed.

3.6.1.11. Change Location

Scenario Name: Change location

Primary Actor: User

Entry Condition: User wants to update his/her location information.

Exit Condition: User changes location successfully.

Event Flow:

- User taps the settings button.
- User changes the current default city.
- User preferences are changed.

3.6.1.12. Get Search Results From Server

Scenario Name: Get Search Results from Server

Primary Actor: Client

Entry Condition: Client has made a search request.

Exit Condition: Server has returned the query successfully.

Event Flow:

- Client makes a request.
- Serves handles the request from client.
- Server checks the input and decides that it is a search query.
- Server performs a database search and returns result to client.

3.6.1.13. Get Recommendations From Server

Scenario Name: Get recommendations from server

Primary Actor: Client

Entry Condition: Client has made a request for recommendation.

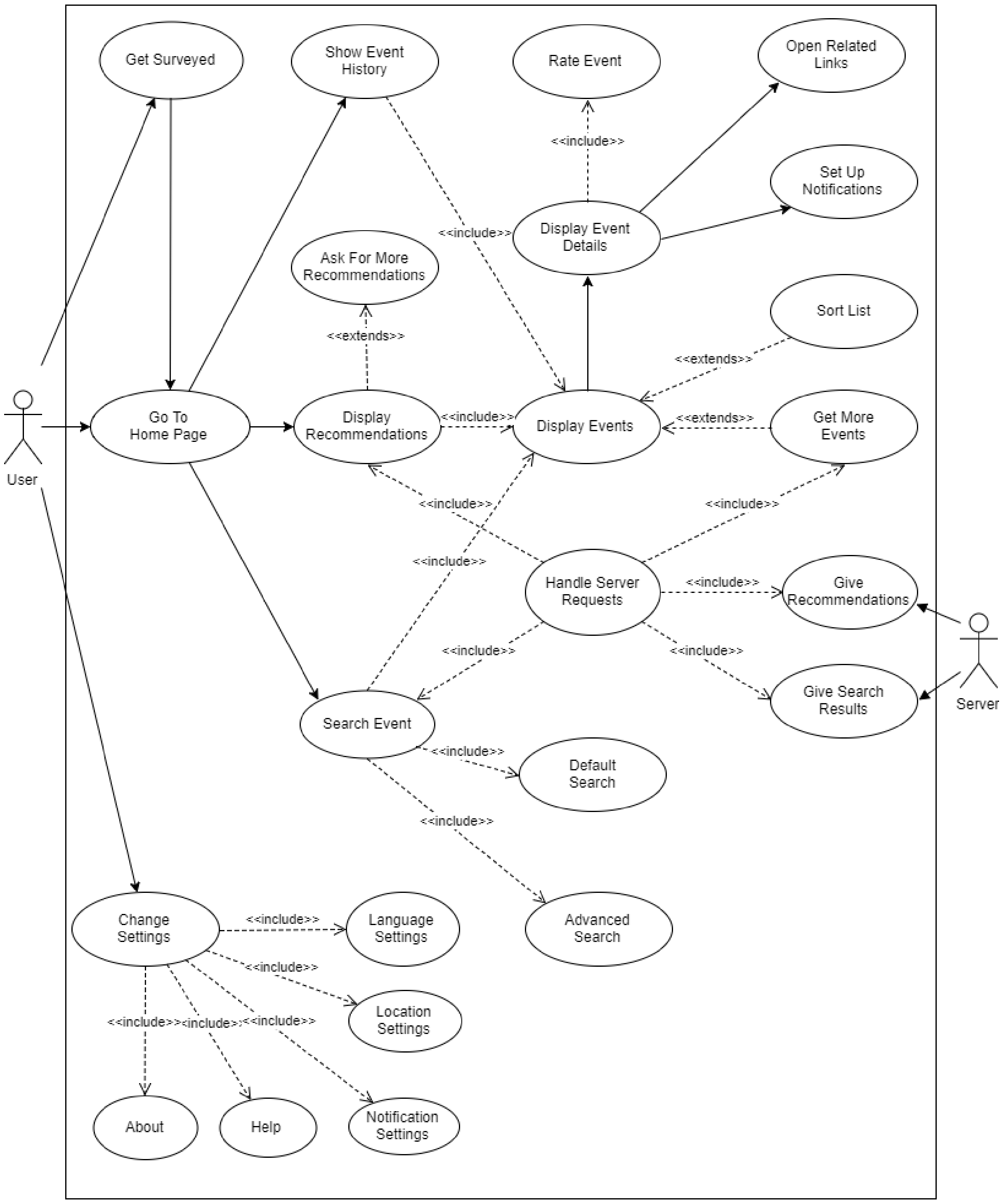
Exit Condition: Server has returned the recommendations successfully.

Event Flow:

- Client makes a request.
- Serves handles the request from client.
- Server checks the input and decides that it is a recommendation request.
- Server performs a get recommendation operation and returns result
to client.

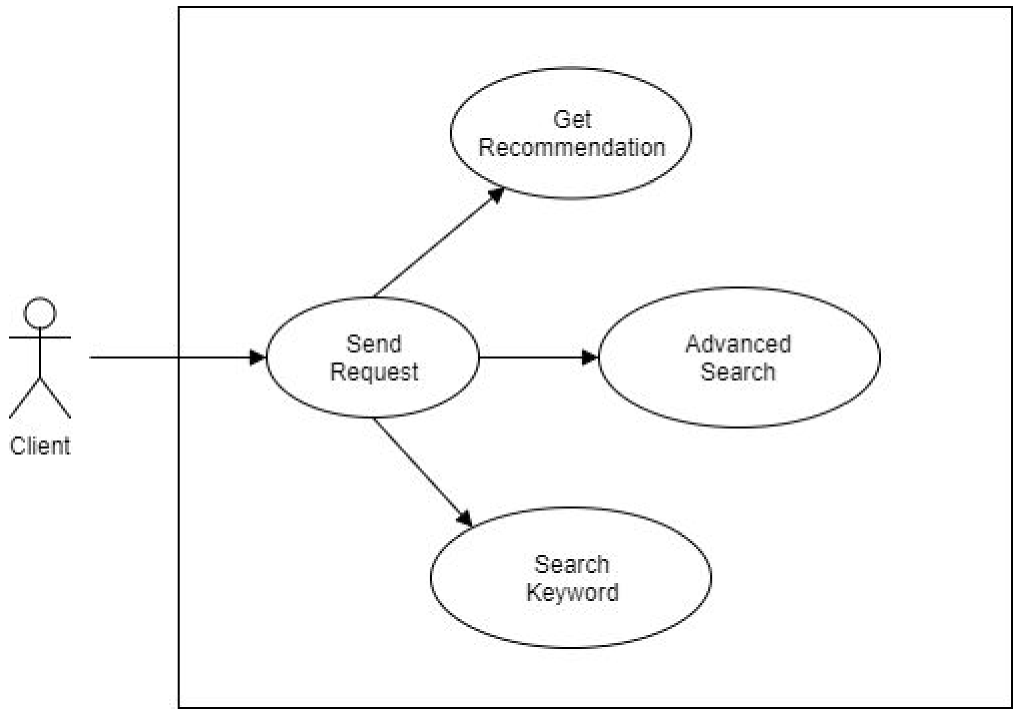
3.6.2. Use case model

3.6.2.1. Client Application Use Case Diagram



When a user opens the application for the first time s/he takes the survey. As a result system begins to learn about the user and forwards them to home page. In home page there are initially recommended events. User can request more recommendations, search a certain keyword or parameter, go to settings. Recommendations and search queries are handled by server. Then system will display the resulting event list, in this display user can vote events as interested or not. User also can tap on an event to get further information about the event . User can set up notification for an event and open related links. In the settings page user can change language, location and notification settings, can read about and help pages.

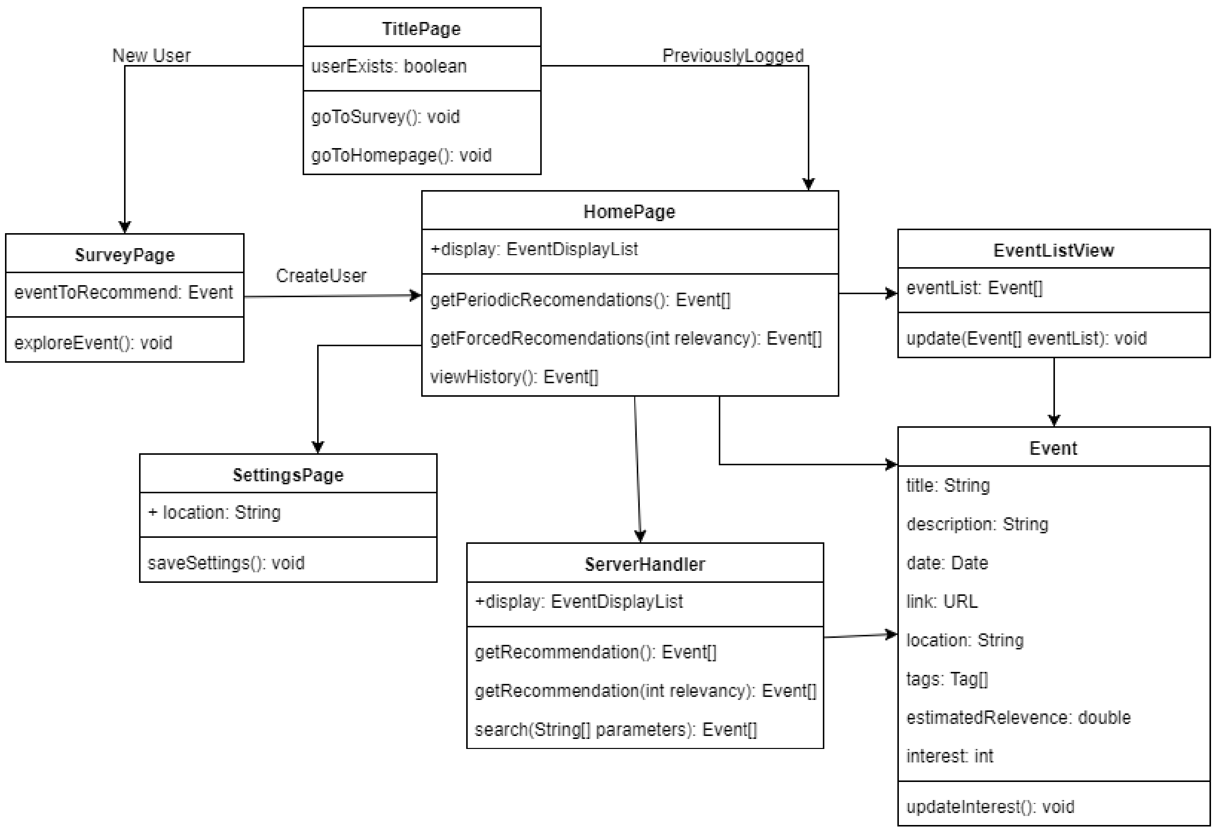
3.6.2.2. Server Application Use Case Diagram



Clients can use our server by requesting to search an event or to get recommendations. To search events clients give a single keyword or advanced parameters, such as date, location, tag, etc, then system gives the query results as a list of events. Recommandations that client get will be adjusted by giving the relevance parameter by the client. To get recommendations, client gives a list of tag-value pairs, then system gives a list of recommended events according to these pairs.

3.6.3. Object and class model

3.6.3.1. Client Application Class Diagram



TitlePage: If this is the first time user uses our application it forwards user to survey page, otherwise user goes to its usual homepage.

SurveyPage: Shows user variety of events for user to vote interested or not, so we can gather the initial user data.

HomePage: By default it shows recommended events. User can choose to go to settings page, search events, or request more recommendations.

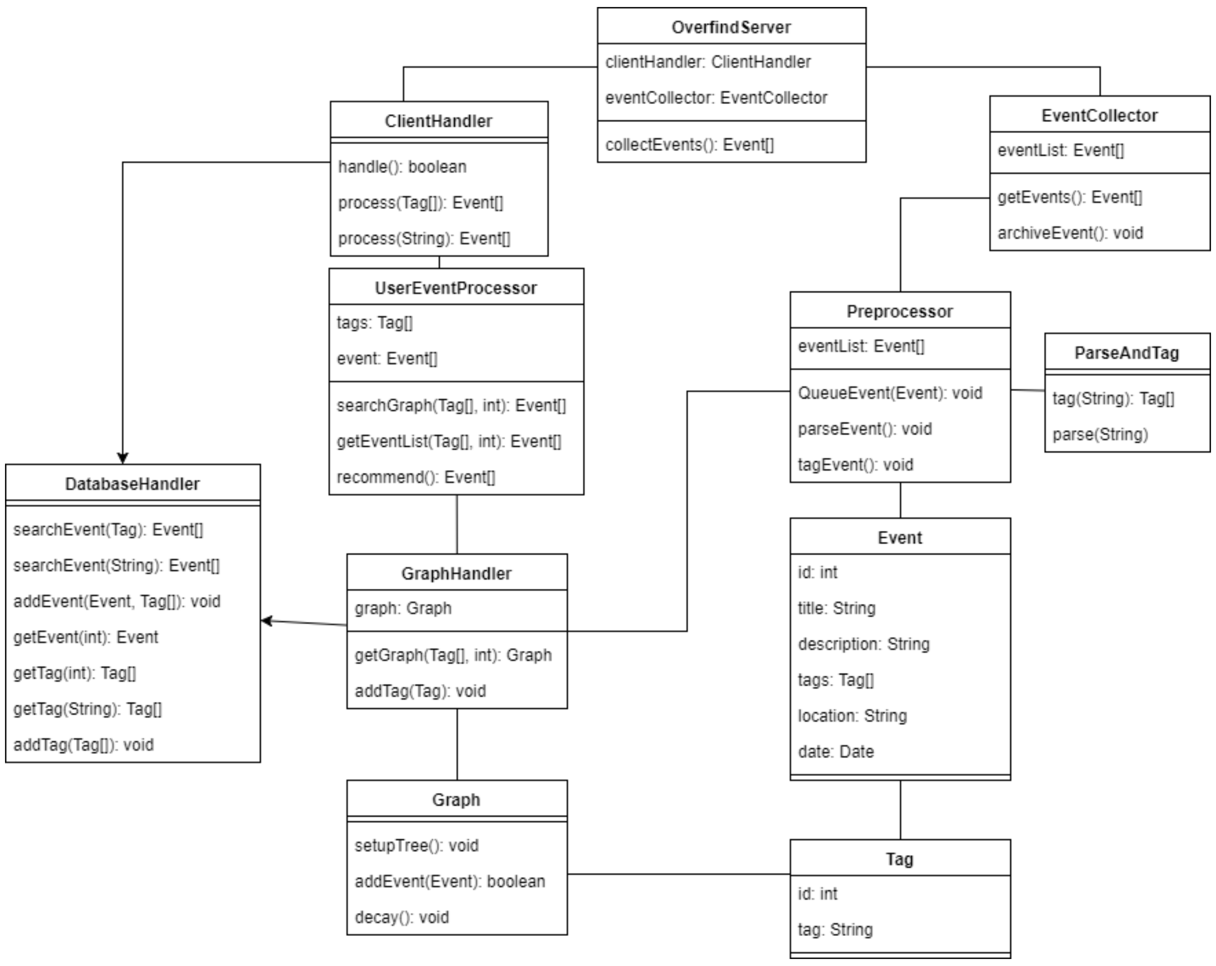
EventListView: It is a view class. Only displays the given event list.

SettingsPage: Gives the user the power to change settings.

ServerHandler: It can send a search query or request recommendations and get resulting event list.

Event: It is an entity class that holds relevant event information.

3.6.3.2. Server Application Class Diagram



OverfindServer: This is the main class of our server. One thread handles user requests, one thread handles periodic event fetch.

ClientHandler: Waits for requests and handles them.

UserEventProcessor: Give recommendations.

DatabaseHandler: Writes or retrieves data from database.

GraphHandler: Takes events and puts its tags to the graph, and sends the meaningful tags to database.

EventCollector: Waits for a period then starts to fetch events.

Preprocessor: Takes fetched events and processes them. This process consists of parsing and tagging.

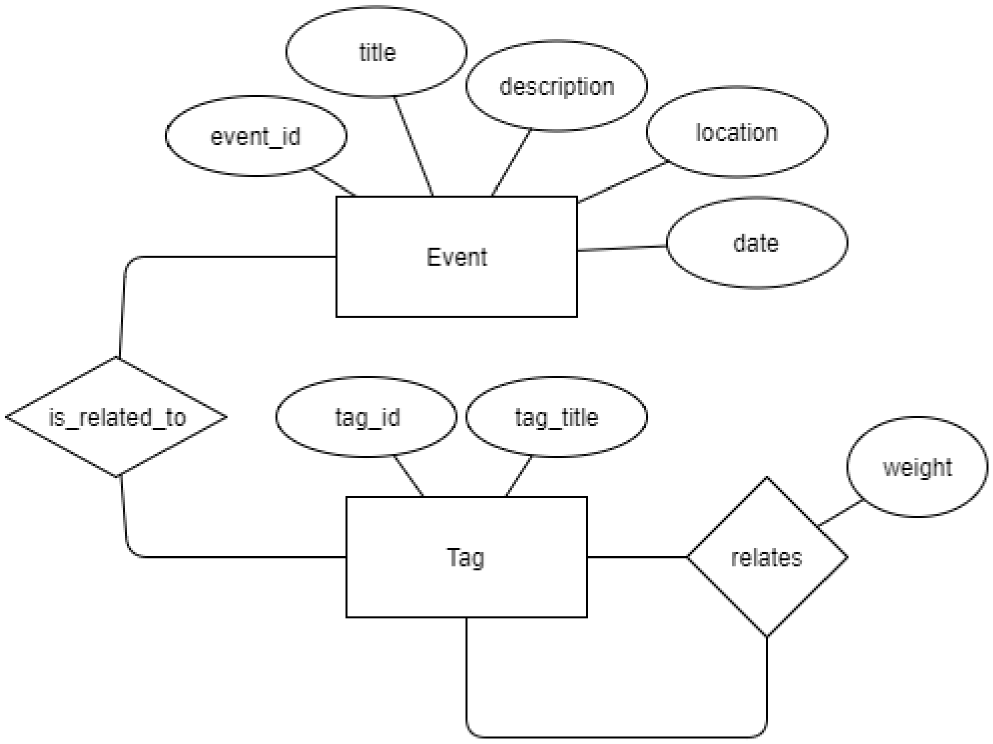
ParseAndTag: Parses and tags fetched events.

Event: Holds relevant event informations.

Tag: Holds relevant tag information.

Graph: Connects relevant tags together.

3.6.3.3. Database E/R Diagram



Event: Entity table for relevant event information.

Tag: Entity table for relevant tag information

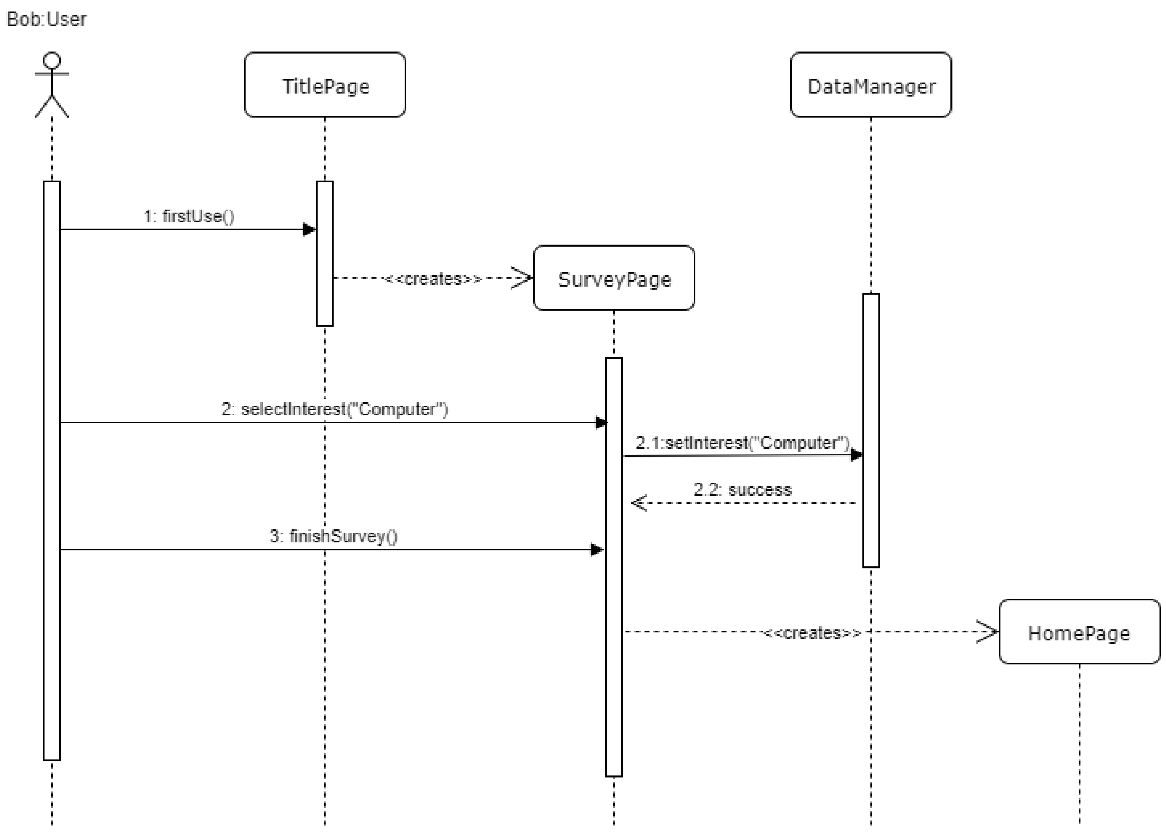
is_related_to: Relation table for tags of events. Many events can have many tags.

relates: Relation table for tags with their related tags. Many tags can be related to many tags.

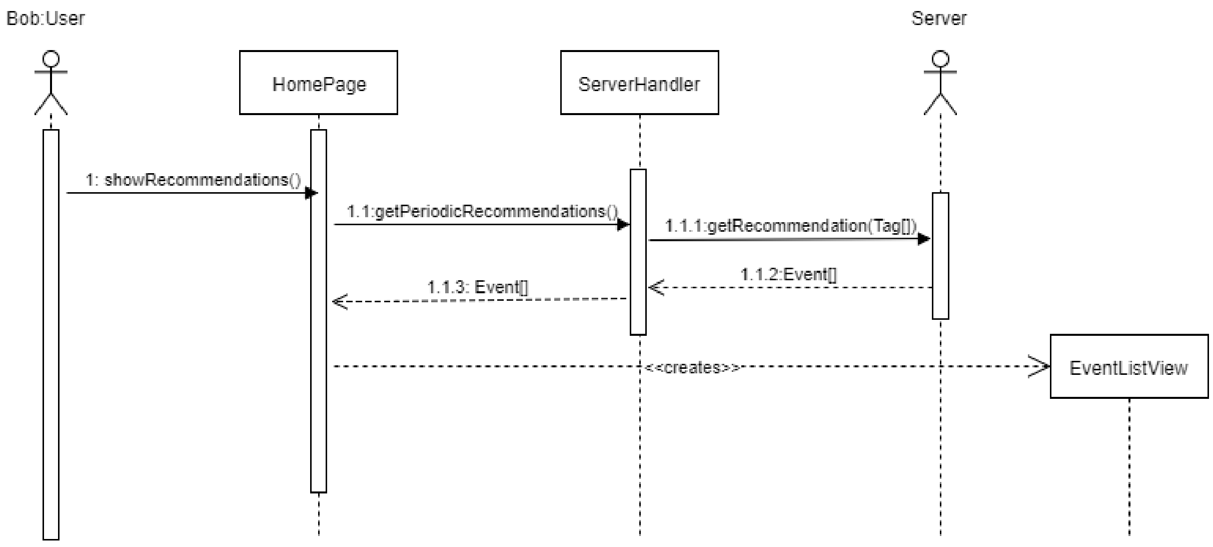
3.6.4. Dynamic models

3.6.4.1. Client Application Sequence Diagrams

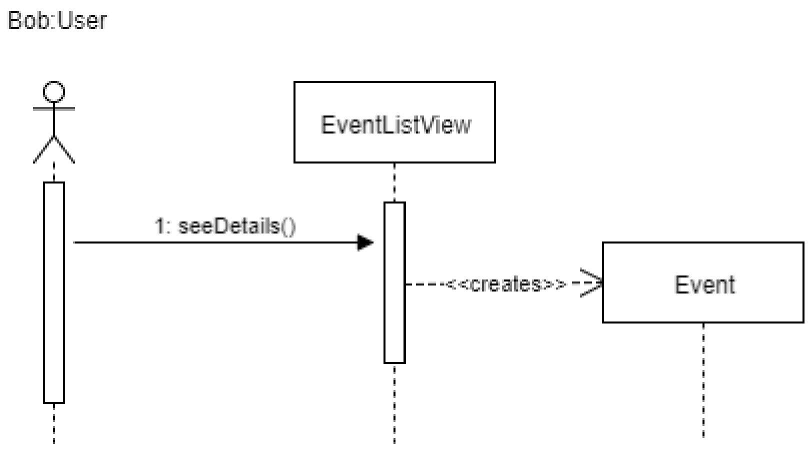
3.6.4.1.1. Fill Survey



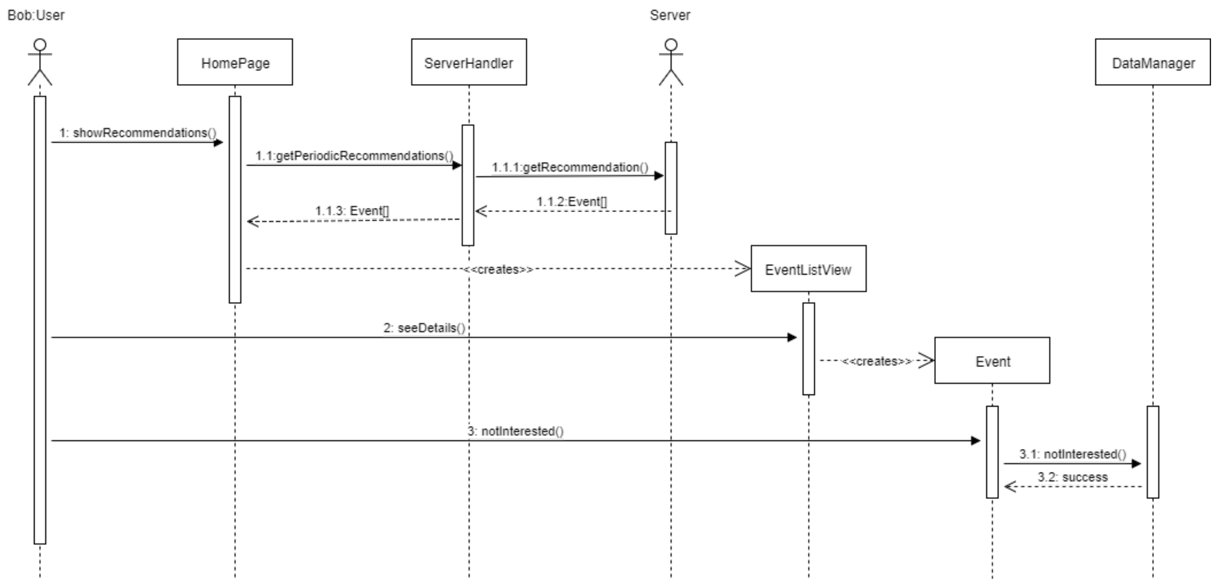
3.6.4.1.2. Show Recommended Events



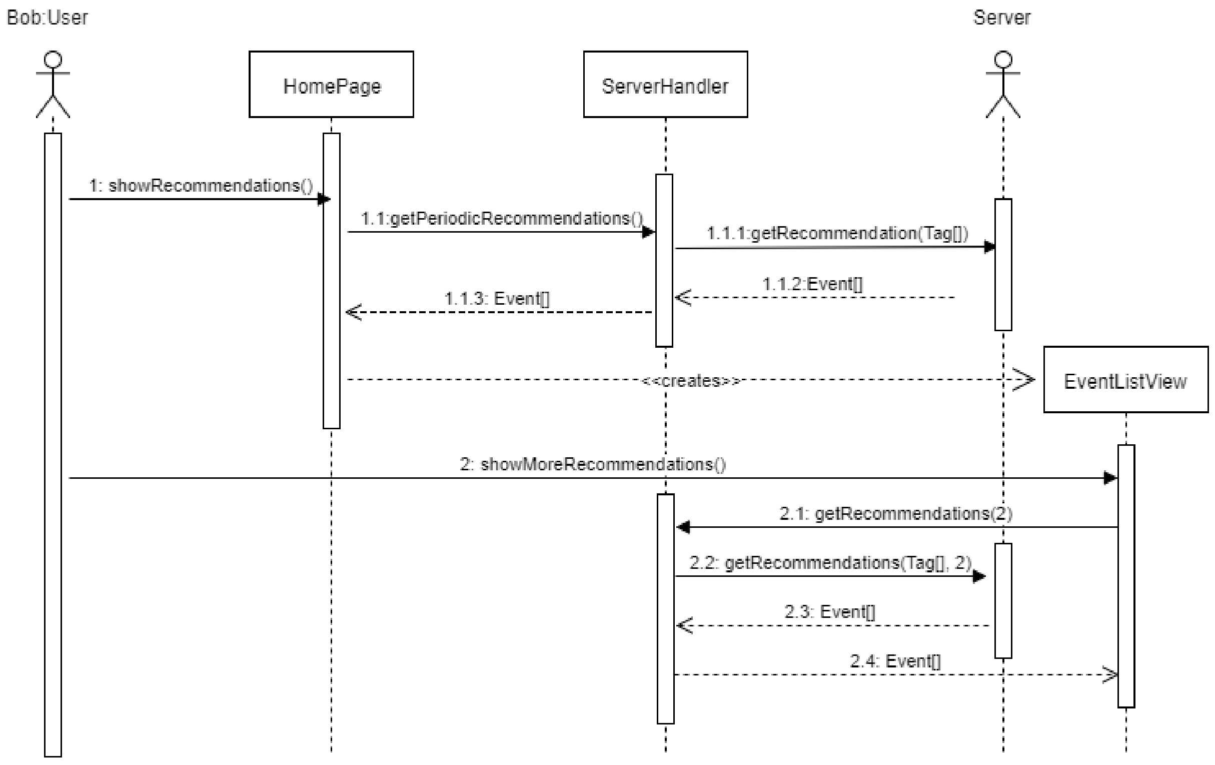
3.6.4.1.3. Show Event Details



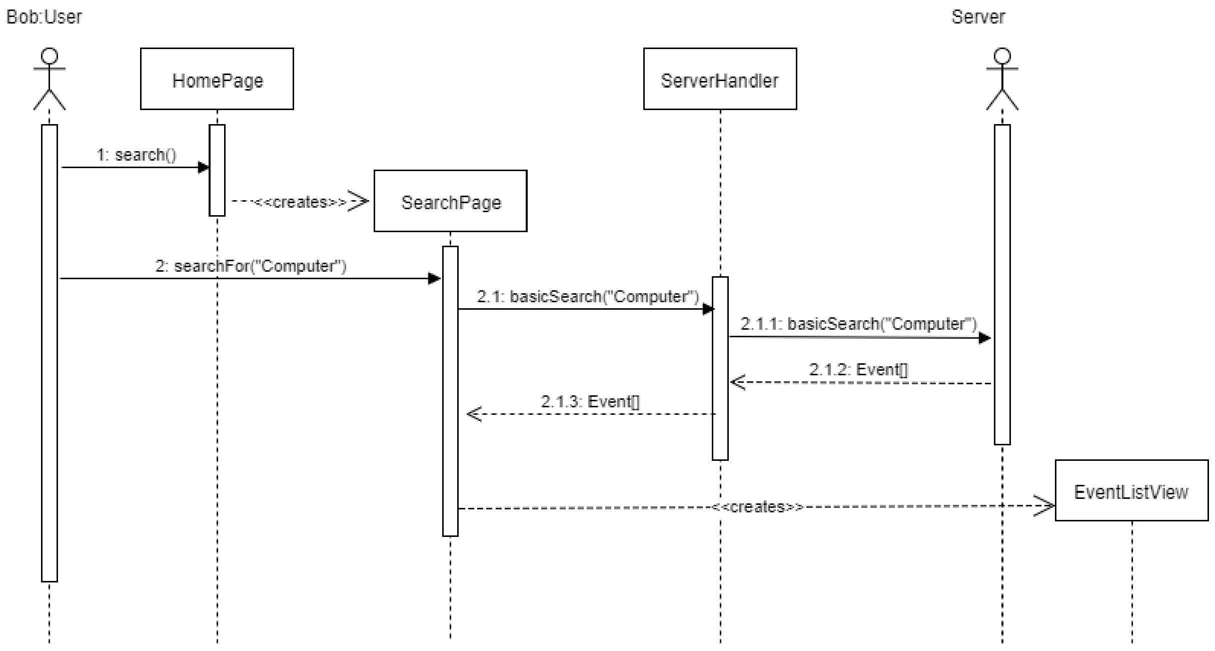
3.6.4.1.4. Mark Recommendation As Not-Interested



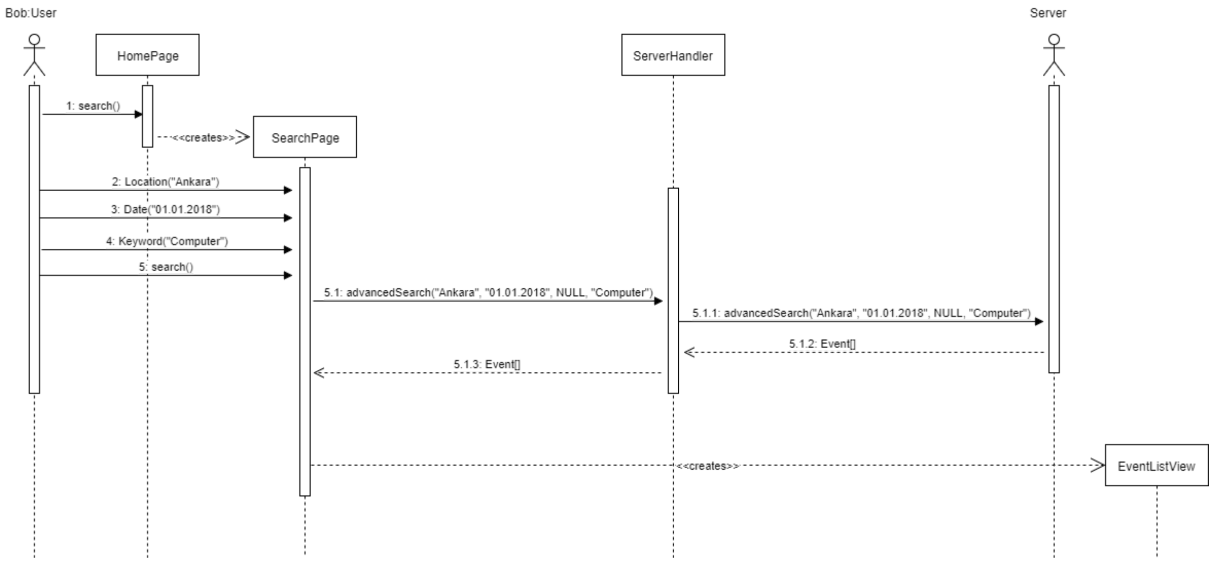
3.6.4.1.5. Ask For New Recommendations


Bob:User
HomePage
ServerHandler
Server
1: showRecommendations()
1.1:getPeriodicRecommendations()
1.1.1:getRecommendation(Tag[])
1.1.2:Event[]
1.1.3: Event[]
<<creates>>
EventListView
2: showMoreRecommendations()
2.1: getRecommendations(2)
2.2: getRecommendations(Tag[], 2)
2.3: Event[]
2.4: Event[]

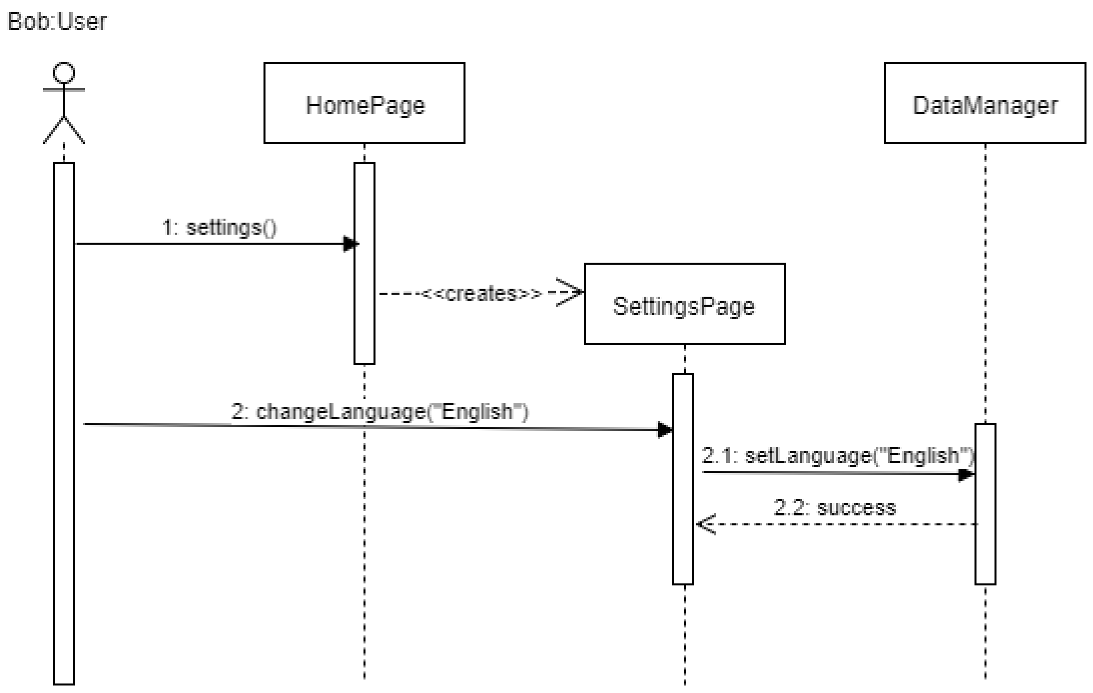
3.6.4.1.6. Search Events



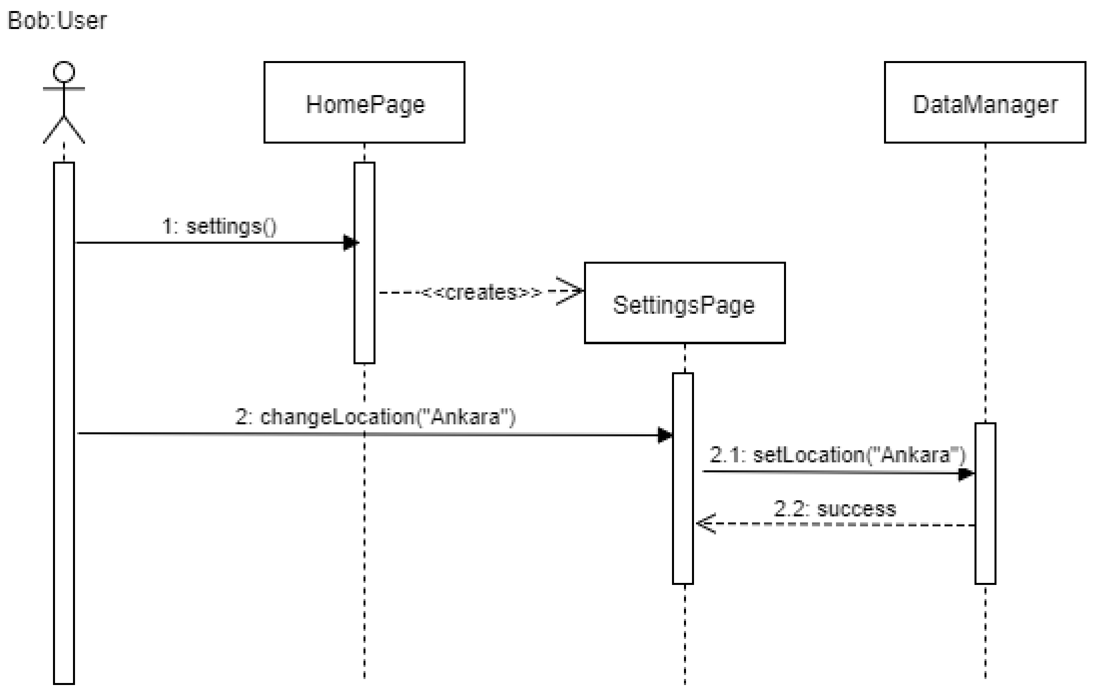
3.6.4.1.7. Advanced Search Events



3.6.4.1.8. Change Language

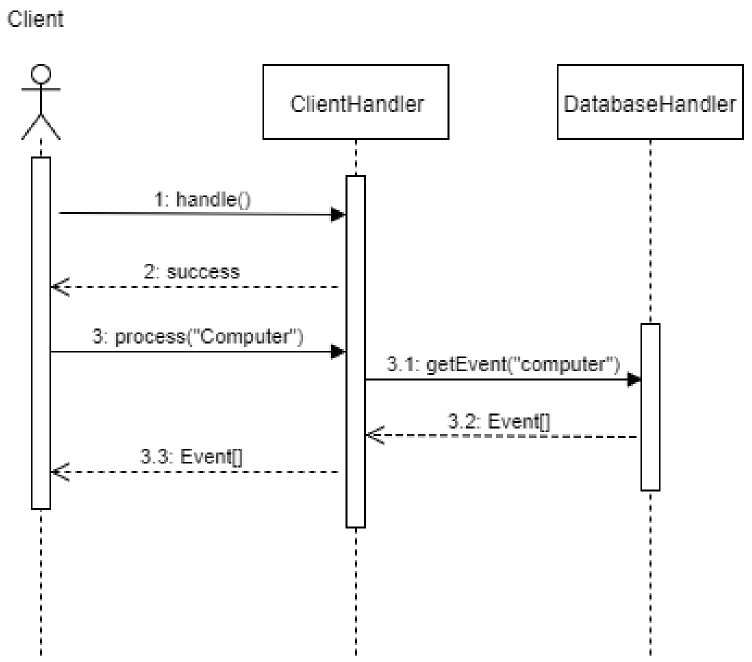


3.6.4.1.9. Change Location

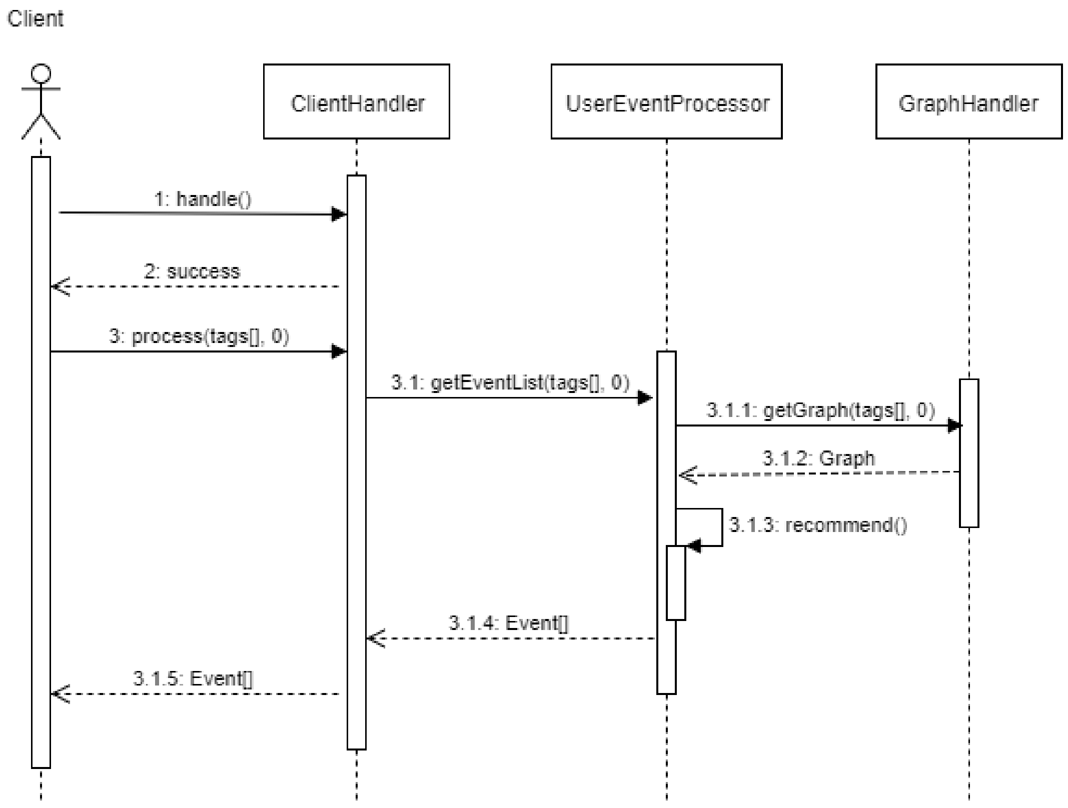


3.6.4.2. Server Application Sequence Diagrams

3.6.4.2.1. Get Search Results From Server



3.6.4.2.2. Get Recommendations From Server

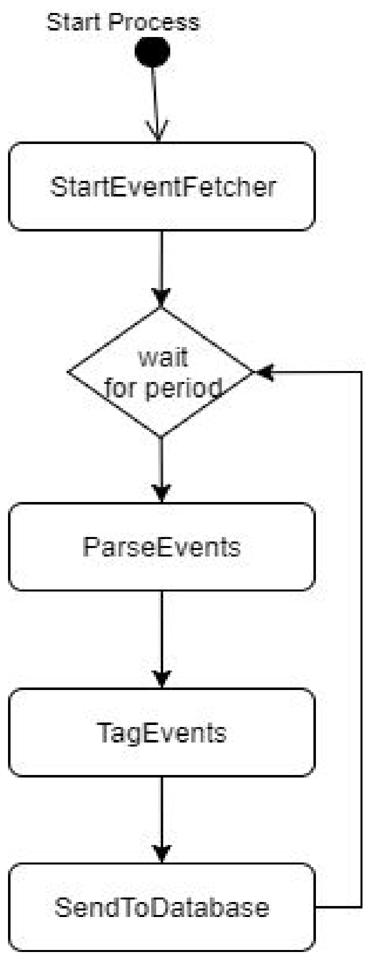


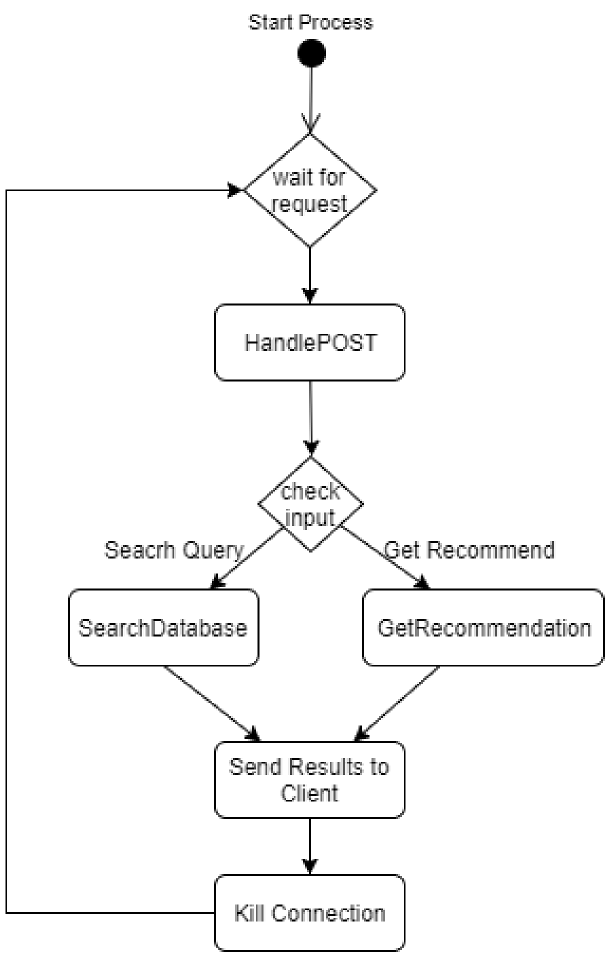
3.6.4.3. Activity Diagrams

3.6.4.3.1. Server Activity

Our server has two primary job. First one is fetching data periodically. A process waits for a period of time then fetches the events according to fetch list. Then server parses and tags the events using Natural Language Processing (NLP). This event and tag data will be sent to database and graph will be updated.

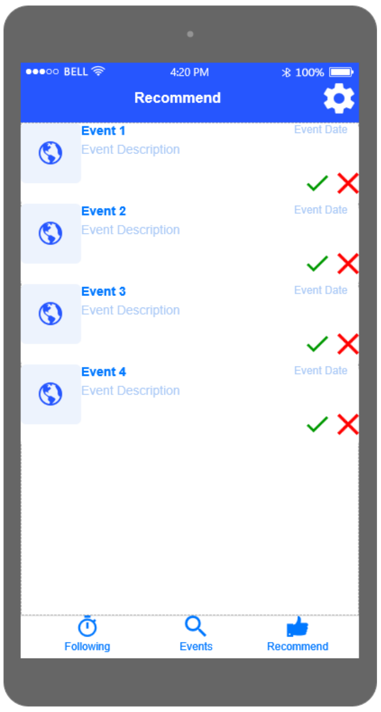
Second part is to listen and respond to client requests. Initially we only have two GET methods to search database and get recommendations.



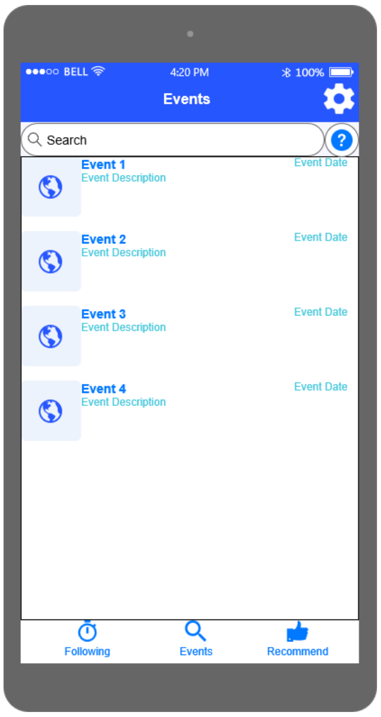


3.6.5. User interface

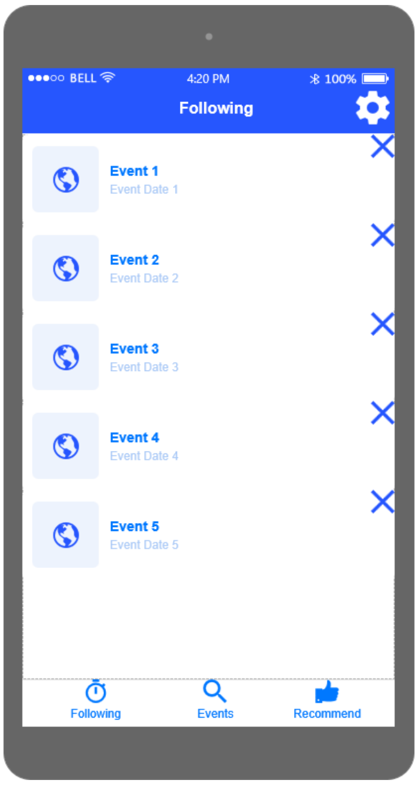
3.6.5.1 Recommended Events



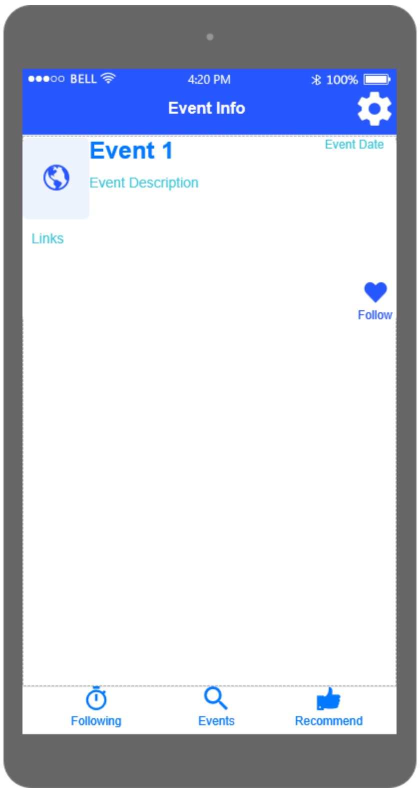
3.6.5.2. Search Events



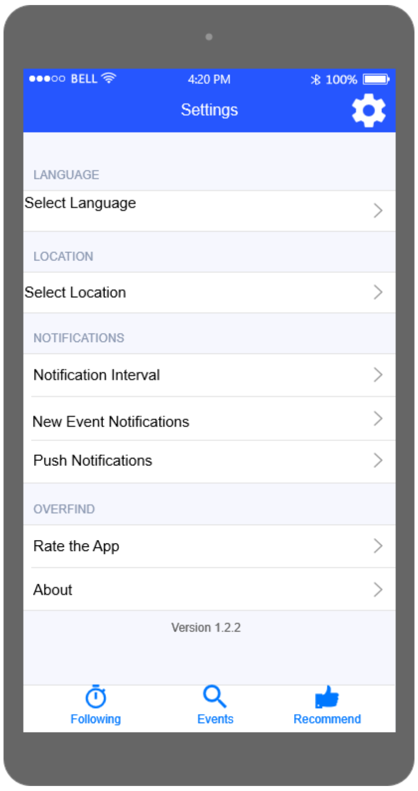
3.6.5.3. Followed Event List



3.6.5.4. Event View



3.6.5.5. Settings Page



●●●○○ BELL 4:20 PM 100%

Settings

LANGUAGE

Select Language

LOCATION

Select Location

NOTIFICATIONS

Notification Interval

New Event Notifications

Push Notifications

OVERFIND

Rate the App

About

Version 1.2.2

5. References

1. "Eventbrite - About Us”, Eventbrite, 2017. [Online]. Available: https://www.eventbrite.com/about/. [Accessed: 6-11-2017].
2. “Model for the Event Recommendation Engine Challenge on Kaggle.com”, GitHub, 2017. [Online]. Available: https://github.com/andreiolariu/kaggle-event-recommendation. [Accessed: 6-11-2017].
3. “Kaggle”, Kaggle, 2017. [Online]. Available: https://www.kaggle.com/kaggle. [Accessed: 6-11-2017].