Bilkent University

Department of Computer Engineering

Senior Design Project

Project short-name: Overfind

High Level Design 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)

December 22, 2017

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 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.

1.1. Purpose of the System

Overfind is an event recommendation system which will consist of three parts, a mobile application, a server application, and a database. System will recommend events to users based on their personal information. Mobile 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. Database will store information.

Mobile application 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. This data will be stored 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.

1.2. Design Goals

1.2.1. Security

We will use HTTPS for our connections with server. Clients have to have authentication to use the services of server.

1.2.2. Modularity

We decomposed system functionalities into smaller modules. Each module is capable of serving a specific need.

1.2.3. Confidentiality

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

1.2.4. 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.

1.2.5. Scalability

1.2.5.1. Memory Scalability

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

1.2.5.2. 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.

1.2.6. Installability

Mobile application will require Android OS version 4.4 (KitKat) [1] or later and can be downloaded from the Google Play Store [2].

1.3. Definitions, Acronyms, and Abbreviations

API: Application Programming Interface
HTTPS: Secure Hypertext Transfer Protocol
MVC: Model-Control-View
OS: Operating System
REST: Representational State Transfer
RDBMS: Relational Database Management System

2. Proposed Software Architecture

2.1. Overview

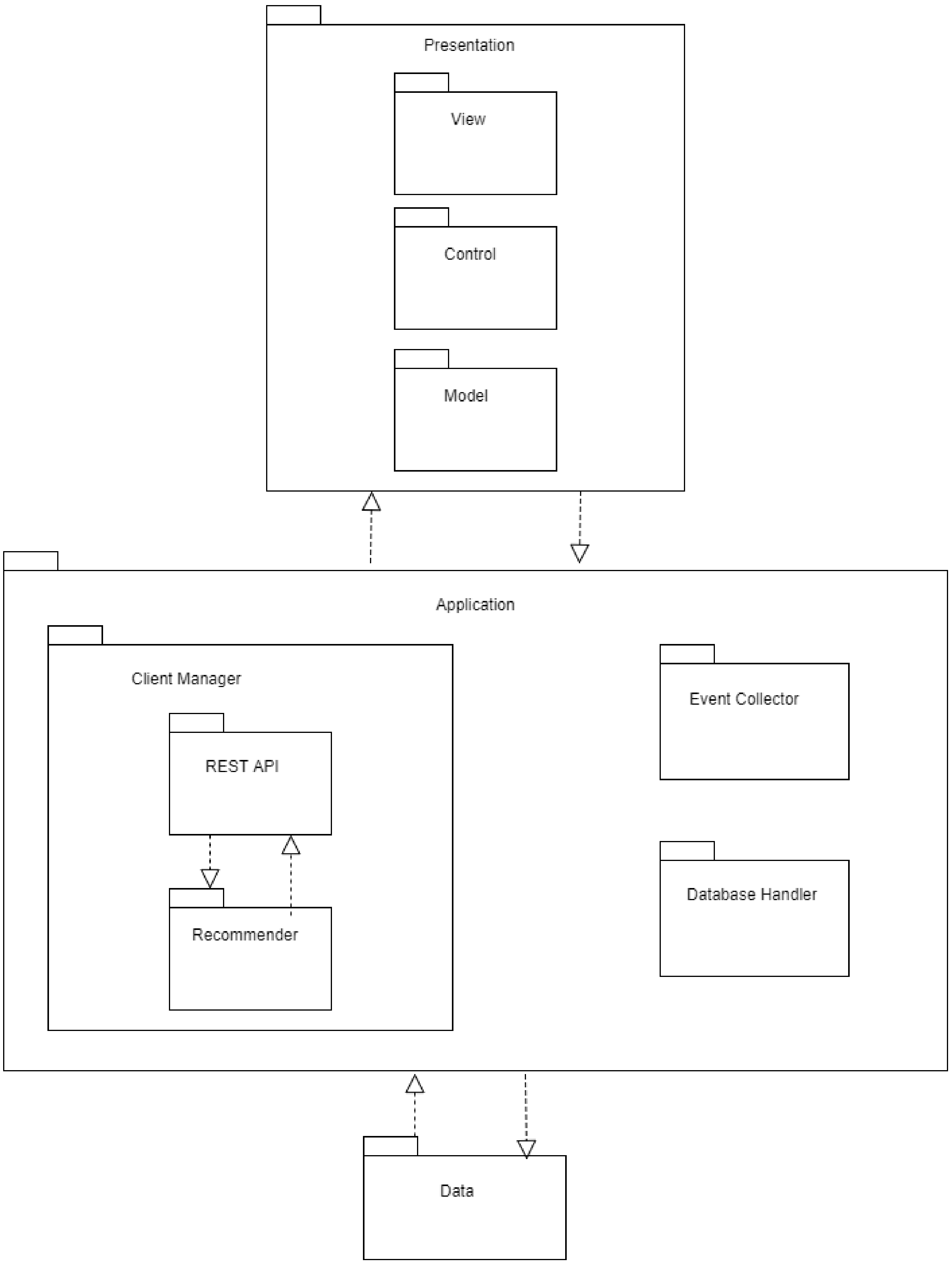
Overfind system is composed of 3 major parts. Client application that is the access point of the user; server application that provides services to clients; database server to store information. 3-tier architecture will be used.

Presentation layer represents the client application. This layer is composed of 3 parts: User interface for users to interact with the system; data manager to store and manipulate event and user information; controller that interacts with the server and manages data. MVC architecture will be used.

Application layer represents the server application. This layer is composed of 3 parts. Client manager that interacts with clients and handles client requests; event manager that interacts with web API's to collect and extract meaning from event information; database handler that interacts with the database server to read or write data.

Data layer represents the database. It will only interact with the server application.

2.2. Subsystem Decomposition



2.2.1. Presentation Layer

Presentation layer will be a mobile application and it will run on Android OS. This application will interact with user and application layer. It will have 3 subsystems.

2.2.1.1. View

This subsystem is responsible with user interaction. It will display and collect information.

2.2.1.2. Control

This subsystem is responsible with data manipulation and interaction with application server. It will update user related data based on data collected from view subsystem and it will update event related data based on data collected from application server.

2.2.1.3. Model

This subsystem is responsible with storing both user related and event related data.

2.2.2. Application Layer

Application layer represents the server application and it will work on a Linux machine. This application will interact with presentation and data layers. It will have 3 subsystems.

2.2.2.1. Client Manager

Client manager is responsible with client interactions and performing the client requests. It will have 2 subsystems.

2.2.2.1.1. REST API

REST API is responsible with client interactions. REST architecture will be used because of its flexibility.

2.2.2.1.2. Recommender

Recommender is responsible with recommending events based on data given by the client.

2.2.2.2. Event Collector

Event collector is responsible with collecting event information and extracting information from the data collected. It will communicate with web API's to collect and process data.

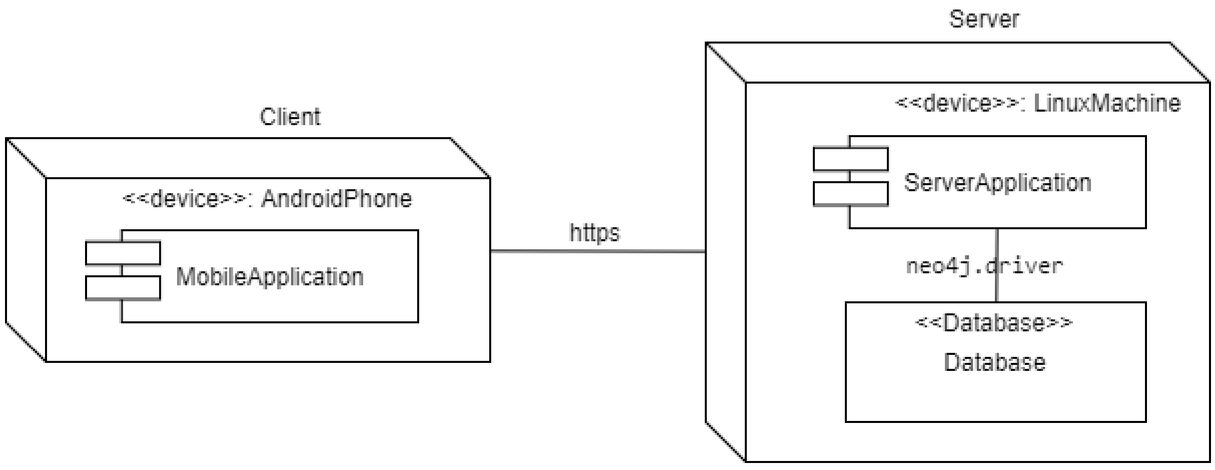
2.2.2.3. Database Handler

Database handler is responsible with interaction with the database server. It will collect information from database server when requested by client manager and will store data to database server when requested by event manager.

2.2.3. Data Layer

Data layer represents the database and it will work on a Linux machine and it will be a graph database. It will interact with application layer.

2.3. Hardware/Software Mapping



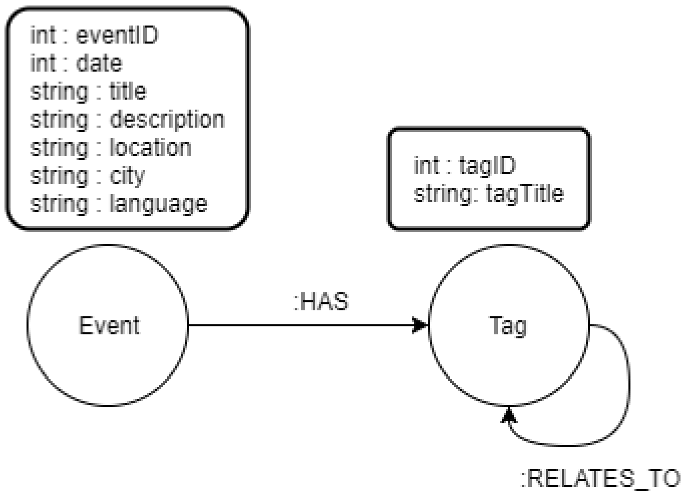
2.3.1. Client

Users have to supply their own devices to use our app. Client side of the application will take place on user's android phone. User can use recommendation and search services by communicating with our server by using https GET method and response will return in JSON format. User's device will store his/her likes locally in the device.

2.3.2. Server

Server will be run on a Linux machine. Server is responsible for listening client requests at any time and collect new events periodically. Client requests will be taken by https GET method and response will sent in JSON format. Server will query database by using neo4j.driver. Database will run in the same machine as server. Only server can reach the database.

2.4. Persistent Data Management



We decided not to use RDBMS since it represents data as sets of relations and tables. In our case, tables mainly consist of tags, relation between tags will increase greatly in number with time. The data will become hard to handle and execution of data will be expensive. For these reasons we choose not to use RDBMS. When we consider the difficulties above, a graph database becomes handy for our case since graph database is optimized for connected data. Since we will have massive amount of connected data we have chosen to use a graph database for our storage purpose. We prefer to use Neo4J [3] for it meets our needs.

2.5. Access Control and Security

Mobile application will use the device ID to create an initial record when the app launches for the first time. All user data will be stored locally on the device. When the mobile application is uninstalled from the device, it will not be possible to restore user data. Mobile application will require storage and network access of the device.

Since user data will be stored locally, mobile application will only send tag data to the server, when needed. Server will send event list to mobile client as JSON. All of the communication will be done through the REST API.

2.6. Global Software Control

2.6.1. Client Control

Client will be controlled by the client controller. When a user marks an event, client controller will update that users information accordingly. When user wants to do a search or asks for recommendations, client controller will send a request to server and save the response as a user list. User list will then update the view as necessary.

2.6.2. Server Control

Server will have two control mechanisms. First control mechanism is the client manager. REST API will listen to clients to figure out whether the user wants recommendations or doing a search. If user is doing a search, it will use database handler to communicate with database and gather the results of the query to create an event list. If the user is asking for recommendations, recommender will use the database handler to get a subgraph and uses the subgraph as well as the user info to create an event list. In both cases the event list will be send to the client application. Second control mechanism is the event collector. Which will periodically searches and gathers events from the web APIs. Then it will parse and tag these events. Lastly it will use the database handler to add events to the database.

2.7. Boundary Conditions

2.7.1. Initialization

At the initialization stage mobile application will first send the device ID and version information to server and waits for an acknowledgement. If mobile application is outdated, user will be notified to update the application. User may be denied access to the application until update is completed.

2.7.2. Termination

Mobile application can be terminated anytime by user. All data in the mobile application are updated at the time of modification. If application is terminated when a file is being modified, termination is halted until modification is finished. Otherwise termination will be instant. If termination occurs during communication with the server, queries will not be saved and user needs to send the query again.

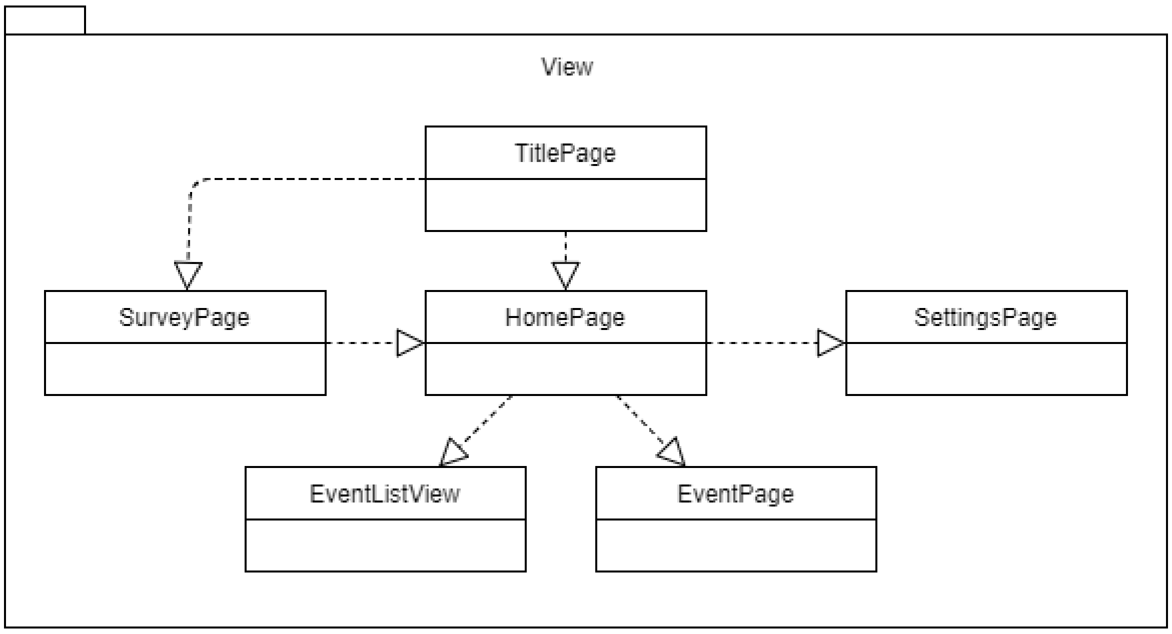
2.7.3. Failure

Since mobile application is network dependant any problem with the internet connection will create an error. Any problem related to the operating system will also create an error. Any problems that occur within the client server connection will cause client to retry to connect and create an error.

3. Subsystem Services

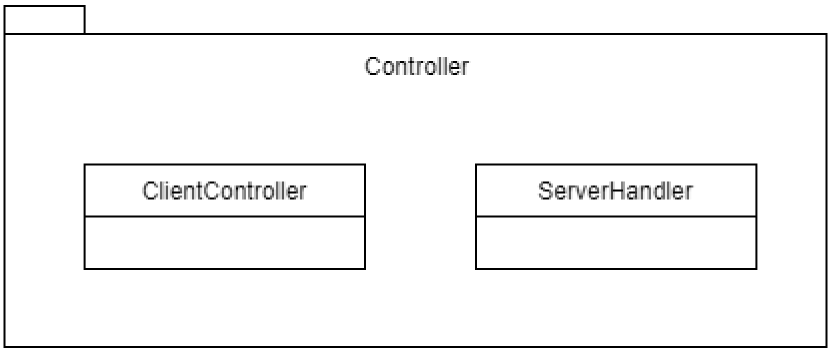
3.1. Presentation Layer

3.1.1. View



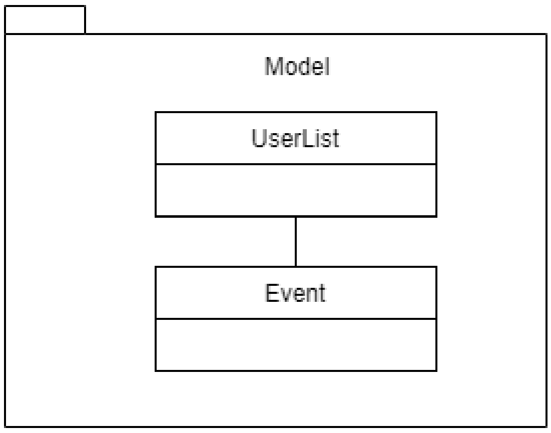
- **TitlePage:** Title page identifies if the users first time using the app, while loading the app. If user is new it forwards the user to Survey Page, otherwise forwards to Home Page.
- **SurveyPage:** App surveys the user so that it can identify users interests. Then forwards user to Home Page.
- **SettingsPage:** User can change language, notification settings, location settings and can see help page.
- **EventListView:** This is a fragment view that we can show the user list of events.
- **HomePage:** It is the main page that our users will see. Users can go to Settings Page, search/browse events and get event details by clicking them.
- **EventPage:** It displays the specific event and its details. User can mark event as interested or not interested. If the user marks interested, the app sets up a notification to alert user when event date is approaching.

3.1.2. Control



- **ClientController:** Updates the event lists based on user activity and manages settings for the user.
- **ServerHandler:** It handles the communication with the server.

3.1.3. Model



- **UserList:** It stores user lists such as event history, interested events, interested tags etc.
- **Event:** It stores event information.

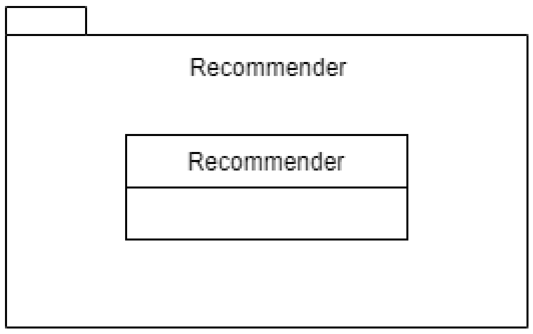
3.2. Application Layer

3.2.1. Client Manager

3.2.1.1. REST API

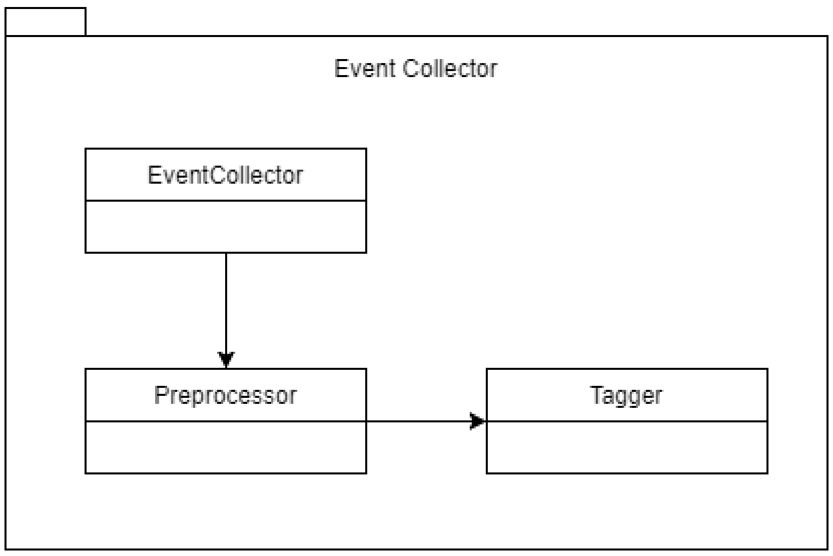
It will listen to client requests and respond accordingly. If client is requesting search results, it will use the database handler to get an event list. If client is requesting recommendations, it will use recommender to get an event list. When the event list is acquired, it will respond to client and send the list.

3.2.1.2. Recommender



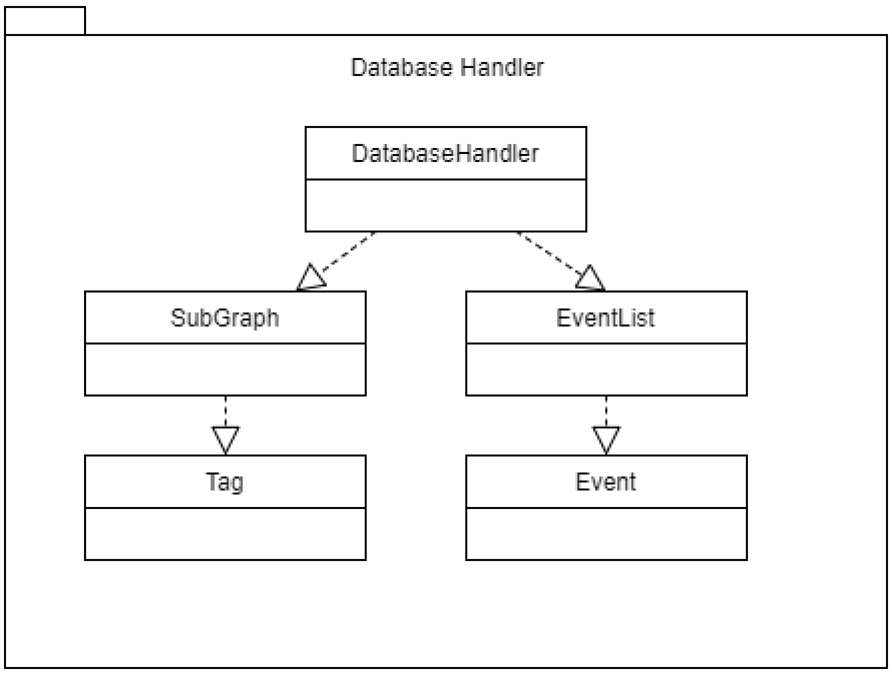
- **Recommender:** Evaluates the user data and uses database handler to get a subgraph according to user data. It then uses database handler to get an event list using both user data and the graph acquired.

3.2.2. Event Collector



- **EventCollector:** Fetches data periodically and if language detector allows, sends them to preprocessor to extract meaningful information. Then uses database handler to store them in database
- **Preprocessor:** Parses the data and extracts title, description, location, city, date, language and calls tagger for further processing.
- **Tagger:** Assigns tags based on description and title information.

3.2.3. Database Handler



- **Database Handler:** It communicates with database to store or gather data. It also stores subgraph and eventlist for the use of recommender.
- **SubGraph:** It stores a graph of tags.
- **Tag:** It stores tag information.
- **EventList:** It stores a list of events.
- **Event:** It stores event information.

4. References

1. "Android - 4.4 KitKat", Android, (n.d.). [Online]. Available: https://www.android.com/intl/tr_tr/versions/kit-kat-4-4/. [Accessed: 22-12-2017].

2. "Android - Play", Android, (n.d.). [Online]. Available: https://www.android.com/play/. [Accessed: 22-12-2017].

3. "The Neo4j Graph Platform – The #1 Platform for Connected Data", Neo4j Graph Database Platform, 2017. [Online]. Available: https://neo4j.com/. [Accessed: 22-12-2017].