



TEI of Crete
Technological Educational Institute of Crete

EventMate

Software Engineering & Big-data modeling,
Plan-Driven and Agile Programming

TEI Crete, Winter semester 2018/2019

EventMate	1
1. System specification	3
1.1. Introduction	3
1.2. Description	3
1.3. Requirements	4
1.3.1. Requirements list	4
1.3.2. Requirements diagram	5
1.4. Mindmap	5
1.5. Use Case	5
1.5.1. Use case diagram	5
1.5.2. Use case details	6
1.6. State machine diagram	8
1.7. Mockups	11
1.7.1. Event owner (Android)	11
1.7.2. Event owner (web)	11
1.7.3. Event assignee	12
2. Architecture design	12
3. Appendices	14
3.1. Attachment 1 – Use case diagram	1
Attachment 2 – Mindmap	2

1. System specification

1.1. Introduction

For human beings, it is natural to socialize and interact with each other. Technology has become more and more important in the social aspects of life. Nowadays people prefer to spend more time on portable devices. Social networking has moved to mobile platforms, which are accessible anywhere and anytime.

The objective of our work is to develop a social application for organizing events any type. As events can be considered birthday parties, new year eve, weddings, baby showers etc.

As we want to make this application more general for any kind of events, we have decided to call it by name “EventMate”. The main goal of the application is to provide tasks management (create, assign, close) and also to create a communication channel among event owners and guests. The application is going to content gamification elements such as a scoreboard and badges.

1.2. Description

The following chapter explains major features. In order to use the application it is necessary to create a user's account or log in via existing social accounts. In our application, there are various user roles such as owners, assignees, guests. As we mentioned earlier, the main goal is to provide task management for an event. For all practical purposes, it means that user can create a task with corresponding attributes such as name, deadline, and persons to be assigned to it. All these fields are saved and continuously maintained. During the whole event, gamification principles are applied which help users to feel that they are a part of the game. When the event has finished, event summary is provided to particular users.

List of roles

Event owner – it is considered as a user that has created an event. Event creation gives this user all permissions to manage event such as edit, delete, lock, start and close.

Task owner – it is a event guest that has created a task. As its owner has a right to edit, delete, start (in case of time limit task), assign points to assignees and close task. Task owner can also assign his own task to himself.

Assignee – it is a type of user that has been assigned to a particular task. He has a right to upload his answers and view results of others.

Guest – it is a regular user that has been invited to an event. He hasn't created any task yet or assigned to any task. He can view event detail with its tasks.

1.3. Requirements

This section describes requirements for EventMate application.

1.3.1. Requirements list

In this section there are listed all functional and also non-functional requirements.

1.3.1.1. Functional requirements

- User registration
- User registration via Facebook / Google account
- User login
- Create events
- Create events from template
- List events
- Filter events
- Modify events
- Delete events
- Change event state
- Create tasks
- List tasks
- Modify tasks
- Change task state
- Submit task results
- Edit task photo
- Assign points for accomplished tasks
- Create reports
- Share reports
- Send private messages
- Send group messages
- Show user profile

- Change their own settings
- Setup notifications

1.3.1.2. Non-functional requirements

- Supported Android version 6 – 8
- Supported Web version by all browsers
- Responsible Android frontend
- Responsible Web frontend
- Android app available in portrait mode
- General usable REST API
- Secured REST API
- Android app capable of working in offline mode
- Ownership permission policy
- Maximum response time of 2 seconds
- Multiplatform backend support
- Log rotation ability
- Account password encryption
- Multilanguage support

1.3.2. Requirements diagram

Fig. presents requirements diagram.

1.4. Mindmap

Created mindmap for this application can be found in Appendix 2.

1.5. Use Case

The following sections describes use case diagram with two actors. The principal use cases are described in detail. Particularly speaking about creating a new event, adding a new task and generating event summary.

1.5.1. Use case diagram

1.5.2. Use case details

Figure 1 - Add task user case detail

Conditions

⊖ Preconditions:

- [Register](#)
- [Log in](#)
- [User has permission to add task to specific event \(user is event owner or guest\)](#)
- [User has chosen specific event](#)

⊖ Post-conditions:

- [View event detail](#)

Flow of events

1. ♀ [User](#) clicks on "Add task" button
2. **SYSTEM** displays "New task" screen
3. ♀ [User](#) fills required fields (Task name, Description, ● [Add assignees](#), ● [Add points](#))
4. (optional) ♀ [User](#) performs ● [Set time limit](#), ● [Specify location](#)
5. ♀ [User](#) clicks on "Save task" button
- ⊖ 6. **while** Form contains errors
 - 6.1. **SYSTEM** Displays error message (form isn't valid)**end while**
7. **SYSTEM** displays "Task detail" screen

TODO prehodit

Figure 2 - Create event use case detail

Conditions

⊖ Preconditions:

- [Register](#)
- [Log in](#)

⊖ Post-conditions:

- [View event detail](#)

Flow of events

1. ♀ [User](#) clicks on "Add event" button
2. **SYSTEM** displays "New event" screen
3. ♀ [User](#) fills required fields (Event name, Date)
4. (optional) ♀ [User](#) performs ● [Invite guests](#)
5. ♀ [User](#) clicks on "Save event" button
6. **while** Form contains errors
 - 6.1. **SYSTEM** Displays error message (form isn't valid)**end while**
7. **SYSTEM** displays "Event detail" screen

Figure 3 - Generate event summary use case detail

Conditions









⊖ Preconditions:

Event is finished by owner

⊖ Post-conditions:

Event summary is generated

Flow of events

1.  Owner clicks on "Generate event summary" button
2. **SYSTEM** displays "Generate options" screen
- ⊖ 3.  User chooses options
 - 3.1.  User selects theme
 - 3.2.  User selects tasks
 - 3.3.  User selects achieved score
 - 3.4.  User selects guests
4.  User clicks on "Generate" button
5. **SYSTEM** generates report
6. **SYSTEM** displays screen for  Send summary

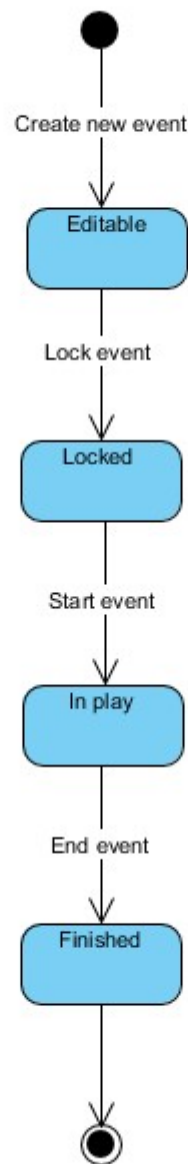
1.6. State machine diagram

The aim of this section is to clarify possible states of events and tasks using state machine diagram.

Firstly, event states will be presented. Any event can exist in four states. After its creation it is considered as 'editable' state. During this phase any user can add a new task to this event. Later on, the event owner has a right to lock his event to prevent guests from adding new tasks. Afterwards, the event owner is entitled to trigger his own event. The state is known as 'in play'. The event can changed its state to 'finished' when the owner close it or assign points for all tasks.

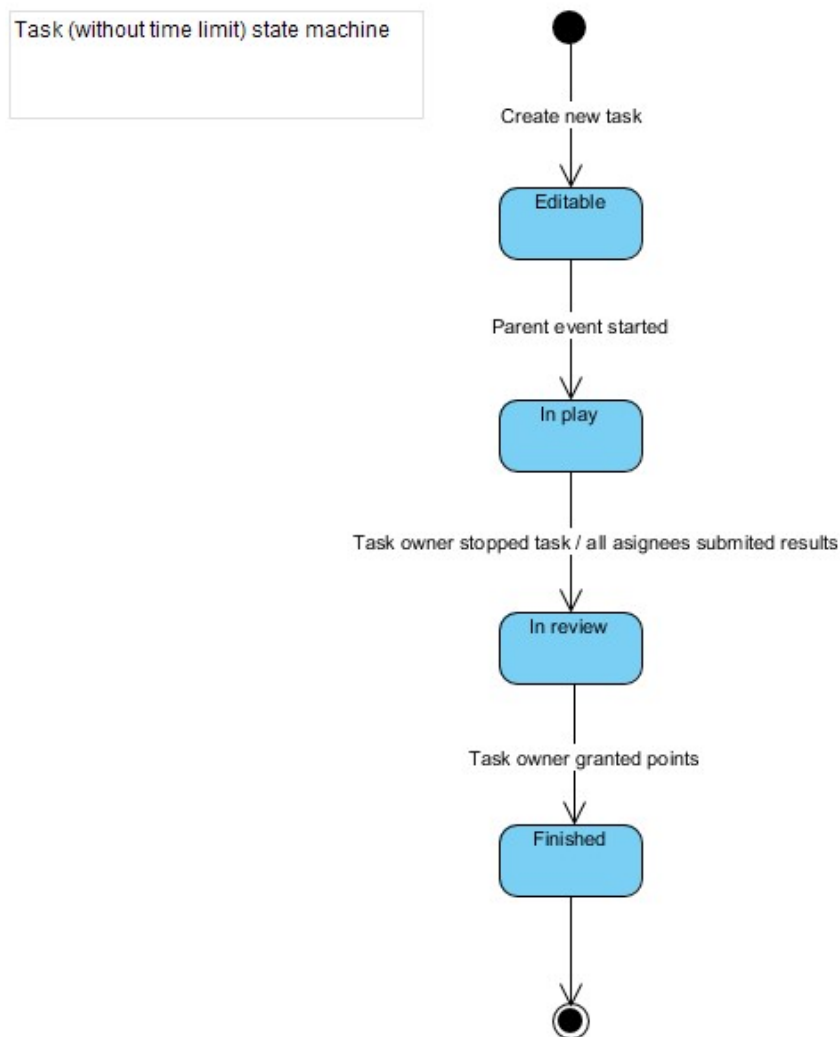
Figure 4 - Event state diagram

Event state diagram



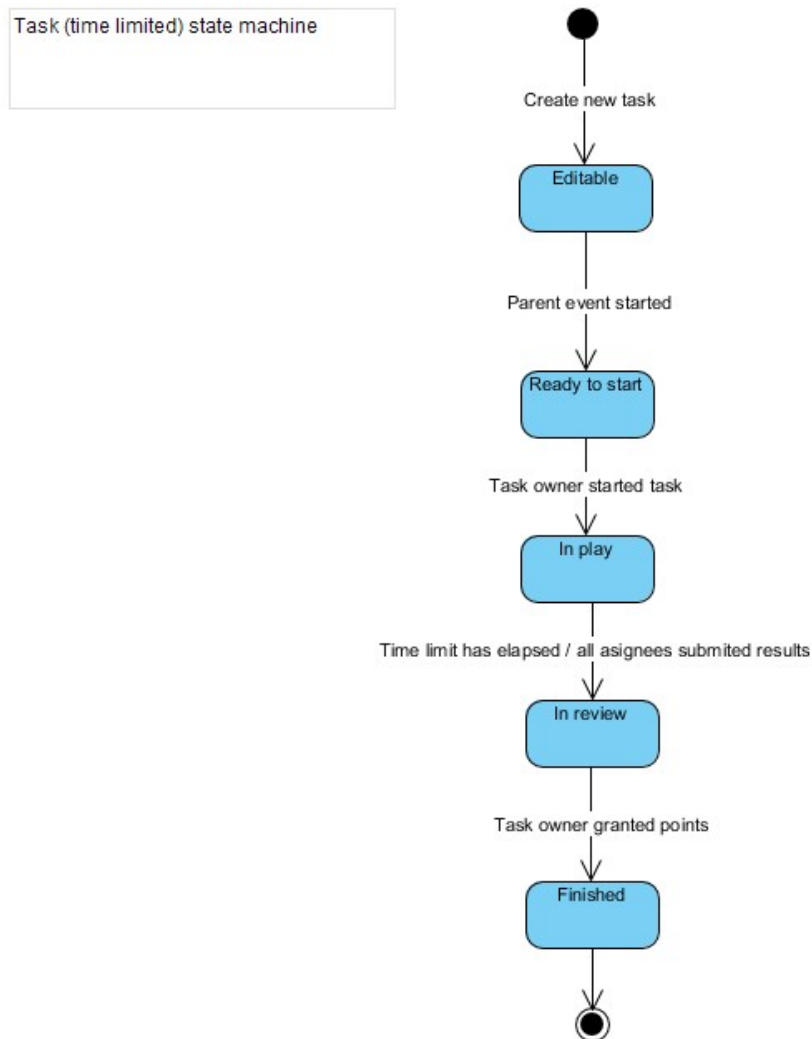
The application supports two types of tasks. Particularly speaking about tasks without defining time to finish and tasks with specified time limit. After its creation a task becomes editable. When its parent event has been triggered, the task passes to a new state 'In play' which supports result submission for assigned users. After submission of results or stopping the task by its owner, the task becomes only readable known as 'In review' state. Transition to last state requires granting points to assignees done by the task owner.

Figure 5 - Task state diagram



As Fig. 6 shown tasks with defined time limit contain one state more named 'Ready to start'. Transition to next state requires activation by its owner.

Figure 6 - Task (time limited) state diagram



1.7. Mockups

asda

1.7.1. Event owner (Android)

asd

1.7.2. Event owner (web)

asd

1.7.3. Event assignee

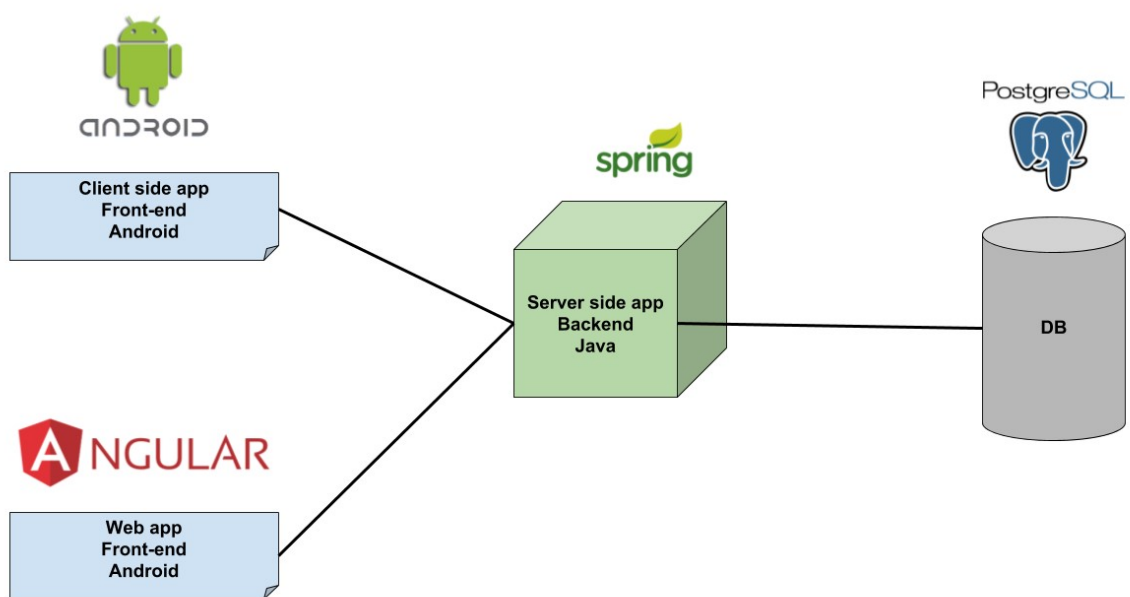
asd

2. Architecture design

Architecture style is based on **client-server**. The project consists of two client side parts and server side with a database.

2.1. Component diagram

2.2. Architecture diagram



Server side

Purpose of server-side application is to provide API, user authentication and persistence to the client side. The asynchronous notification system will be also provided by the server side.

Technology stack (early version)

- Java
- Spring Framework
- PostgreSQL

Client side

Client side - Android

First type of client-side is going to be implemented as a native mobile application for platform Android. There will be a huge emphasis on UX (User experience) and also on gamification techniques. Android application is going to adhere MVVM architecture that allows separating the user interface logic and the business logic.

Technology stack

- Kotlin
- LiveData
- Library Retrofit for API calls
- Dagger for dependency injection

Client side – Web

Another type of client-side is going to be implemented as a web application based on Angular framework.

Technology stack

- Angular 6 framework

TODO:

Chapter 1 – system specification (diagrams..)

Description, requirements list + diagram, textual description, use case diagram, mockups, short description,

Chapter 2 (Architecture design (title))

- Component diagram
- Architecture diagram

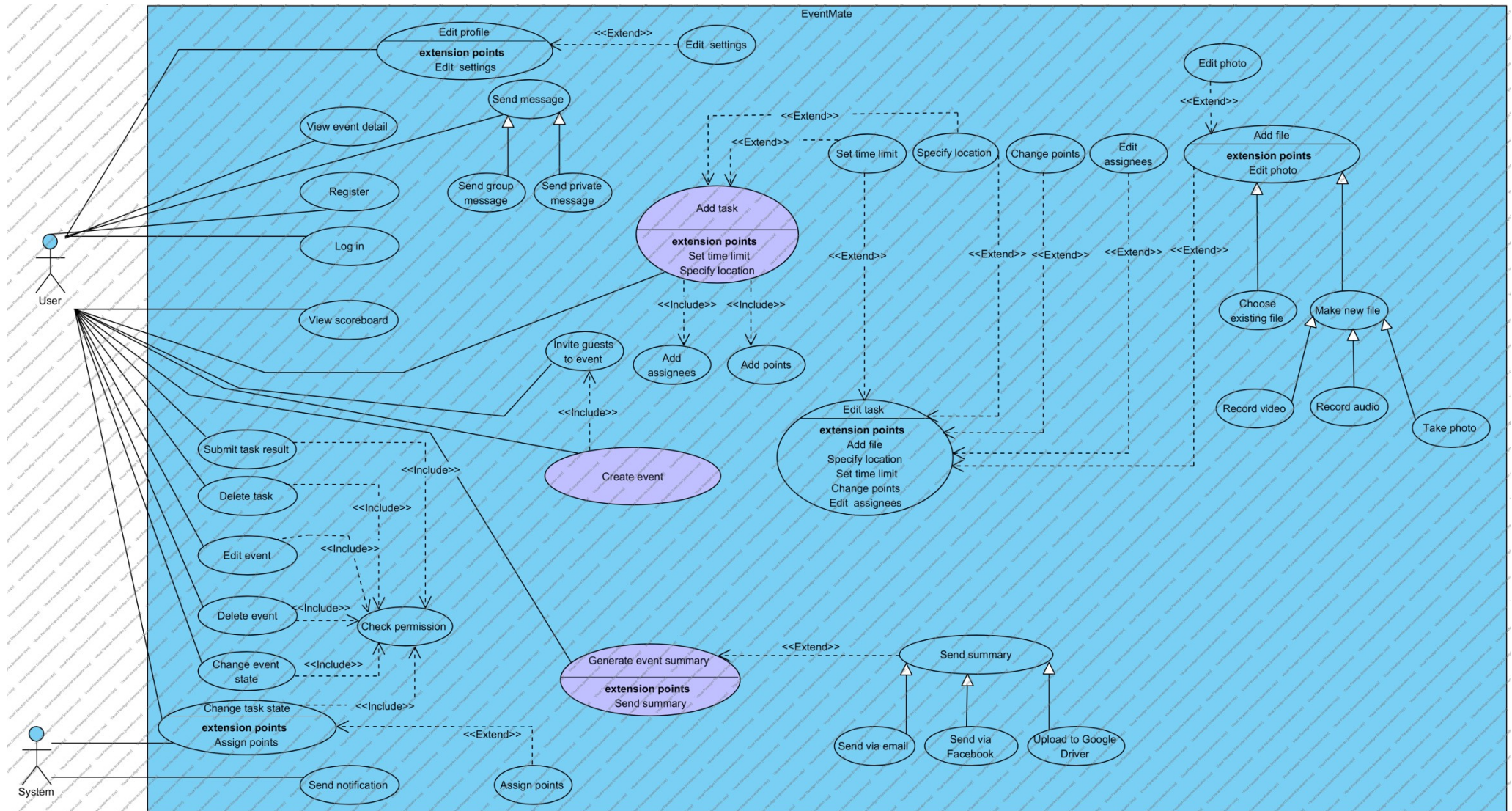
Presentace

1. Slide – mockup

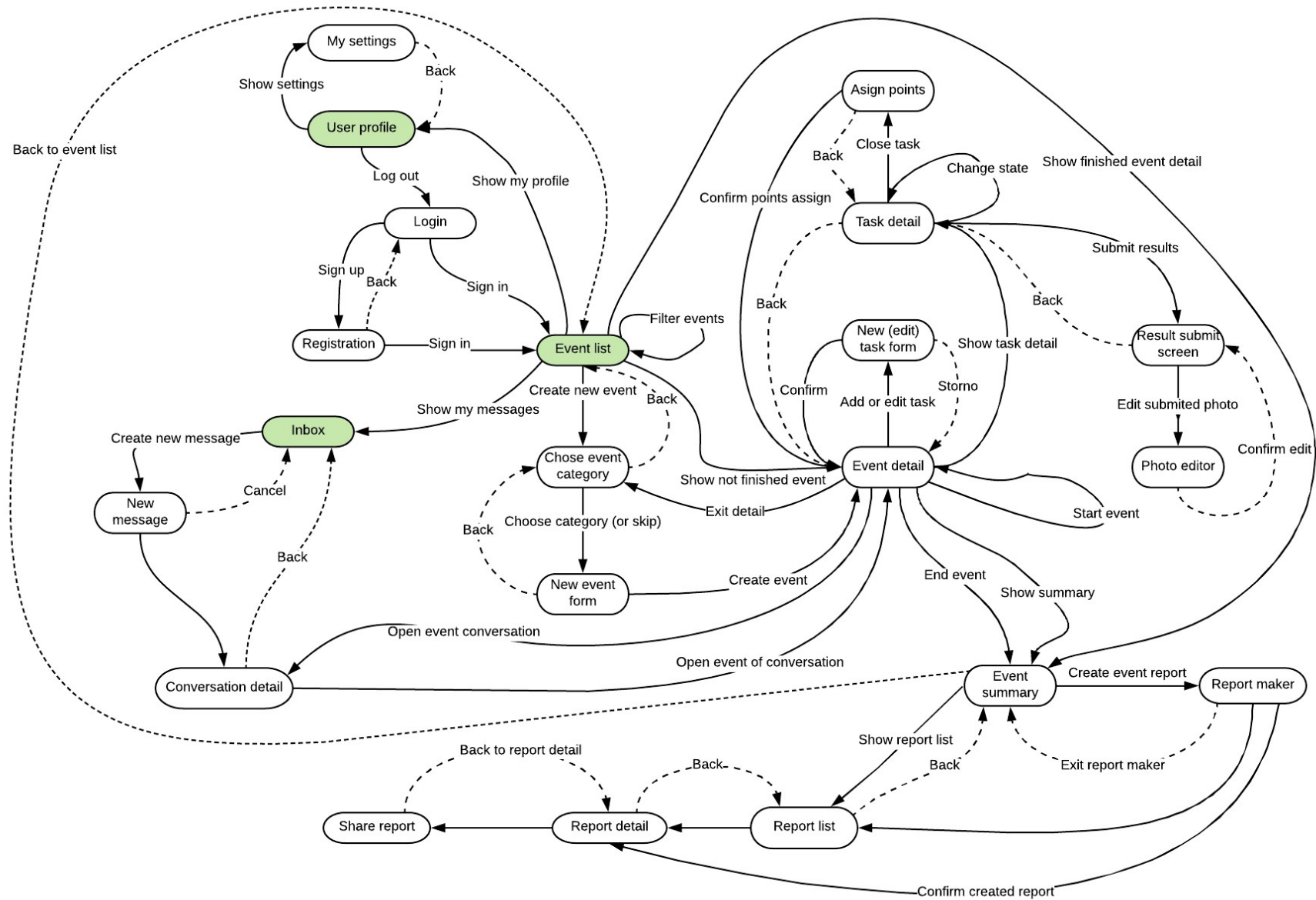
2. Slide – req. diagram
3. Slide – use case diagram

3. Appendices

3.1. Attachment 1 – Use case diagram



Attachment 2 – Mindmap




```

graph TD
    ManageEvents[Manage events  
Text = "Application should allow managing events"  
kind = "Functional"  
ID = "REQ001"]
    DeleteEvents[Delete events  
Text = "Managing events should allow deleting events"  
kind = "Functional"  
ID = "REQ001 REQ005"]
    FilterEvents[Filter events  
Text = "Managing events should allow filtering events"  
kind = "Functional"  
ID = "REQ001 REQ003"]
    CreateEvents[Create events  
Text = "Managing events should allow creating events"  
kind = "Functional"  
ID = "REQ001 REQ002"]
    ListEvents[List events  
Text = "Managing events should allow listing events"  
kind = "Functional"  
ID = "REQ001 REQ001"]
    ModifyEvents[Modify events  
Text = "Managing events should allow modifying events"  
kind = "Functional"  
ID = "REQ001 REQ004"]
    CreateReports[Create reports  
Text = "There should be option to create report from finished event"  
kind = "Functional"  
ID = "REQ001 REQ020"]
    ChangeEventState[Change event state  
Text = "Managing events should allow changing event state"  
kind = "Functional"  
ID = "REQ001 REQ006"]
    CreateEventsFromTemplate[Create events from template  
Text = ""  
kind = "Functional"  
ID = "REQ001 REQ002 REQ0"]
    ManageTasks[Manage tasks  
Text = "Application should allow managing events"  
kind = "Functional"  
ID = "REQ011"]
    ListTasks[List tasks  
Text = "Managing tasks should allow creating tasks"  
kind = "Functional"  
ID = "REQ011 REQ012"]
    ChangeTaskState[Change task state  
Text = "Managing tasks should allow changing task state"  
kind = "Functional"  
ID = "REQ011 REQ015"]
    SubmitTaskResult[Submit task result  
Text = "Managing tasks should allow to send task result"  
kind = "Functional"  
ID = "REQ011 REQ016"]
    AssignPoints[Assign points  
Text = "Managing tasks should allow assigning points for accomplished task"  
kind = "Functional"  
ID = "REQ011 REQ018"]
    ShareReports[Share reports  
Text = "There should be option to share created report via email"  
kind = "Functional"  
ID = "REQ001 REQ020 REQ021"]
    EditTaskPhoto[Edit task photo  
Text = "Submitting result should provide option for editing submitted photo"  
kind = "Functional"  
ID = "REQ011 REQ016 REQ017"]
    SendPrivateMessage[Send private message  
Text = "Application should provide messaging system"  
kind = "Functional"  
ID = "REQ022"]
    SendGroupMessage[Send group message  
Text = "Messaging should allow group messages"  
kind = "Functional"  
ID = "REQ022 REQ024"]
    RegistrationViaFacebookGoogle[Registration via Facebook / Google  
Text = "Application should provide user registration"  
kind = "Functional"  
ID = "REQ009"]
    SetupNotifications[Setup notifications  
Text = "User should have an option to change his own settings"  
kind = "Functional"  
ID = "REQ026"]
    BackendPlatformSupport[Backend platform support  
Text = "Backend application should be runnable on Linux and Windows"  
kind = "Non-functional"  
ID = "REQ037"]
    RESTAPISecurity[REST API security  
Text = "REST API will be secured with OAuth2 protocol"  
kind = "Non-functional"  
ID = "REQ033"]
    Login[Login  
Text = "Backend application will have to backup and rotation system"  
kind = "Non-functional"  
ID = "REQ038"]
    MultilanguageSupport[Multilanguage support  
Text = "There will be option for change font and language"  
kind = "Non-functional"  
ID = "REQ040"]
    PasswordSecurity[Password security  
Text = "Password should be secured using bcrypt"  
kind = "Non-functional"  
ID = "REQ039"]
    ShowUserProfile[Show user profile  
Text = "There should be ability to show user profile"  
kind = "Functional"  
ID = "REQ029"]
    ResponseTime[Response time  
Text = "Maximum 2 second response time"  
kind = "Non-functional"  
ID = "REQ036"]
    AndroidOfflineMode[Android offline mode  
Text = "Android app should be also working in offline mode"  
kind = "Non-functional"  
ID = "REQ034"]
    UserLogin[User login  
Text = "Application should provide login"  
kind = "Functional"  
ID = "REQ006"]
    OwnershipPermissionPolicy[Ownership permission policy  
Text = "Application should provide permission policy for managing events and tasks"  
kind = "Non-functional"  
ID = "REQ035"]
    AndroidSupport[Android support  
Text = "Android app frontends should be support on Android version 6.0"  
kind = "Non-functional"  
ID = "REQ028"]
    BrowserSupport[Browser support  
Text = "Web frontend should be supported on all major browser"  
kind = "Non-functional"  
ID = "REQ029"]
    AndroidScreenOrientation[Android screen orientation  
Text = "Android app should provide portrait screen orientation"  
kind = "Non-functional"  
ID = "REQ034"]

    ManageEvents --> DeleteEvents
    ManageEvents --> FilterEvents
    ManageEvents --> CreateEvents
    ManageEvents --> ListEvents
    ManageEvents --> ModifyEvents
    ManageEvents --> CreateReports
    ManageEvents --> ChangeEventState
    CreateEvents --> CreateEventsFromTemplate
    CreateEvents --> ManageTasks
    ManageTasks --> ListTasks
    ManageTasks --> ChangeTaskState
    ManageTasks --> SubmitTaskResult
    ManageTasks --> AssignPoints
    CreateReports --> ShareReports
    EditTaskPhoto --> SubmitTaskResult
    SendPrivateMessage --> SendGroupMessage
    SendGroupMessage --> RegistrationViaFacebookGoogle
    RegistrationViaFacebookGoogle --> SetupNotifications
    SetupNotifications --> BackendPlatformSupport
    RESTAPISecurity --> Login
    Login --> MultilanguageSupport
    MultilanguageSupport --> PasswordSecurity
    PasswordSecurity --> ShowUserProfile
    ShowUserProfile --> ResponseTime
    AndroidOfflineMode --> UserLogin
    UserLogin --> OwnershipPermissionPolicy
    OwnershipPermissionPolicy --> AndroidSupport
    AndroidSupport --> BrowserSupport
    BrowserSupport --> AndroidScreenOrientation
  
```