Software Design

Ac. yr. 2019./2020.

Festival organization

Documentation, Rev. *1*

Group: *Festival organisation*

Coordinator: *Bartol Bilic*

Turn-in Date: *15 . 11 . 2019.*

Supervisor: *Hrvoje Nuic*

Contents

1. Documentation Change Log

Rev.	Opis promjene/dodatka	Authors	Date
0.1	Template uploaded to git.	Bilic	20.10.2019.
0.2	Project Description written.	Ceple	6.11.2019.
0.3	Functional Requirements written.	Ceple	6.11.2019.
0.4	Class Diagram	Strbad	12.11.2019.
0.5	Use Cases finished	Ceple	13.11.2019.
0.6	Database description and diagrams added	Fribert	13.11.2019.
0.7	Architecture description added	Bilic	14.11.2019.
0.8	Sequence Diagrams added	Ceple	15.11.2019.
0.9	Class Diagrams added	Bilic, Sparemblek, and Strbad	15.11.2019.
1.0	Final revision 1 document generated	Bilic, Sparemblek, Strbad, and Ceple	15.11.2019.

2. Project Description

2.1 General Idea

The idea of this app is to enable a low to mid size festival organisation in a relatively simple and straight-forward manner that would be easily accessible and understandable even to non-technically educated Users.

The application would be open-source, and would run on the Android OS - a native mobile app.

2.2 Position in the market - the competition

Mainly, the competition consists of either high-profile professional apps, or non-native(non-mobile) apps. Thus, the point of this app is to fill that gap - it's supposed to be a native app, that's relatively simple to use, and very portable and easily deployable.

This would also imply, and goes hand in hand, with the fact that the application would be easy to use and easily accessible to a wide range of people - from highly-trained professional IT Users all the way to non-IT savvy amateur/inexperienced Users.

Because of the low difficulty, and relatively ad rem employability, this app would be more suitable to the lower skill level(entry to mid-level) Users, as it is likely that Pro Users would require a larger scale App for, probably, larger caliber Festivals that they deal with. With that in mind however, this app can be used as a mini, mobile device reminder version of whatever Pro tool is used for Festival organisation.

In the market there is presence of both event organisation, and user-event interface apps. This app is focused on organisation, and not festival-goers. Therefore, from a marketplace standpoint, there will be no impact on the demand of the application due to the selected specialisation.

2.3 In-Depth Description

Application would be used for Festival organisation - music festivals, film festivals, library events, food and alcohol festivals, parties, birthdays, and other kinds of meet-ups. By scope and complexity it would be used for smaller and medium scale Events/Festivals.

The system would be run and moderated by Administrators. The system/platform would allow concurrent usage by multiple Users. These Users are: Administrators, Creators, Organisers, Workers.

From here on, unregistered users and Users who aren't logged in will be referred to as Guests.

2.3.1 Account Data

Prior to registration, Guests must fill in a Registration form. This form consists of:

- Username
- Email
- Password
- Password Verification
- Phone Number
- Name
- Surname
- Desired Role
    1. Leader
    2. Organiser
    3. Worker

Aside from this data, Users can also upload their Profile Picture. In case they don't, a default anonymous one will be used. These Profile Pictures will be put on Users' Cards that serve as Festival entrance tickets.

Users can change some of this data. Specifically, they can alter:

- Email

- Password
- Phone Number
- Profile Picture

Users aren't allowed to modify their Usernames in order to prevent abuse and misuse. If Users want their Usernames, Names or Surnames changed, they can contact the Administrator, and if the appeal makes sense the Administrator can change those values for the User.

2.3.2 User roles

2.3.3 Administrators

Maintain and organize the platform. They curate the Creators. They have complete transparency and access to all data. They can veto and issue bans on: Festivals, Jobs, User registrations, comments, etc... Basically they have complete control over the platform.

2.3.4 Creators

Need to be verified by one of the Administrators. Have the ability to create and edit Festivals, as well as have complete transparency of the Festivals they have created. Can undertake certain actions regarding the details of their Festivals(and Jobs).

They appoint and verify Organisers. To these Organisers they assign Festivals that need to be organised - delegation of Festival planning and execution. Organisers cannot organise concurrent Festivals.

2.3.5 Organiser

Organisers are appointed by the Creators. They manage, plan and execute Festivals on both a macro and micro/detail scale. They are appointed to Festivals by Creators.

They can organise multiple Festivals, and be appointed to those Festivals by multiple Creators - it is only important that none of those Festivals are concurrent.

They organise and manage Jobs and Activities that need to be done for the Festival. Jobs are performed by Workers. Organisers also have the ability to write and read comments on the Worker's walls as to better organise these Jobs.

Job and Activity management is done via Auctions. First Workers apply to these Auctions, and their entries need to be verified by the Organisers. And then the Verified Entries can compete to receive the Job Offer. Should they deem it necessary, Organisers can extend Auctions by one day.

2.3.6 Worker

They perform Jobs and Activities. For these Jobs they first have to apply to the selection process(Auctions) during which their entries are curated by the Organisers(or eventually Administrators or Creators). Upon verification, their entries compete in order for the Worker to receive the Job Offer.

Some Jobs are necessary to be done by multiple Workers. Workers can specify in their Auction Entry how many more Workers would the Job require. Jobs can be done concurrently(parallelised), and Workers can perform Jobs with maximum freedom, so long as these Jobs aren't concurrent - in case they are, the system will immediately veto such entries.

Workers' accounts have certain specifics compared to other accounts. Their profiles contain:

1. Field of specialisation
2. Basic account data and information
3. Former Job information
4. Comment section on the Workers profile wall intended to allow the Workers co-workers, boss, to comment the Workers performance, characteristics, their satisfaction of working with the Worker, etc...

2.3.7 Functions and happenings of the system

Here some inner elements of the system will be defined, along with their attributes and characteristics.

2.3.8 Festival

Festivals are events that are being organised. Their attributes are:

1. Name
2. Description
3. Location
4. Planned start-end time

2.3.9 Auction

Auctions are part of the process where Workers apply to Jobs. First Workers need to turn in their entries, which are then verified by the Organisers/Creators, and moderated by Administrators. Upon successful verification/moderation, Workers' entries finally enter the Auction competition where the lowest bid ends upon the expiration of the Auction period. If need be, Organisers and Creators can extend the Auction period by one day.

Auctions have the following attributes:

1. Price
2. Comment
3. Number of Workers needed(Workforce Quantity)
4. Estimated time to Completion(ETC)

2.3.10 Timelines

Timelines are intended to provide an easy and graphical way of organising Jobs in the time-domain this is done in a way that the timeline itself is actually a flowchart organised against a time axis. The liable User can add, remove and manipulate the Objects located in the Timeline.

The beginning time, end time and the duration of a Timeline Object can be manipulated by moving and resizing the Objects corresponding box. This provides a visual way of organising Objects, and makes it easy to parallelise them.

The ability to parallelise implies the existence of multiple branches. Branches can be created, deleted, and moved.

There are 2 types of Timelines:

- Festival Job Timeline
- Job Auction Timeline

2.3.11 Job

Jobs are performed by Workers. They are constituted of Activities that need to be done in order for the Job to be completed. Jobs can require multiple Workers. The lowest bid Worker is assigned the Job. Auctions usually last 1 day, but if necessary Organisers and/or Creators can extend them by an additional day.

Jobs are to be given a sequential order of execution. They can be parallelised as well. Multiple Workers cannot work on concurrent Jobs.

2.3.12 Activity

Jobs consist of multiple Activities that need to be done in order for the Job to be completed - as already defined above. Activities help break down Jobs into manageable, and easily organised chunks. Since Jobs can be done by multiple Workers, as well as be parallelised, Activities can help with that task as they would be the elementary particle, the smallest unit of work that needs to be done and distributed throughout the network of Workers that would be performing the given Job.

2.3.13 Cards

*Participants in the organisation of the Festival would each receive a **SINGLE** card - with an exception of multiple Workers working on the same Job - that will be explained further below. Workers also have another specificity - their Cards feature some additional information.*

A single Card can be printed for a single User and a single Festival. Therefore, the same User is able to download and print multiple Cards for multiple Festivals.

2.3.14 Card Format

The cards can be downloaded in a .pdf format - with the dimensions 10x7cm

Everyone's Cards feature the following information:

1. User picture

2. Name and Surname
3. Name and Logo of the Festival
4. QR code(MD5 Hash):
   (a) User's Name
   (b) Name of the Festival

*Workers' Cards contain **ADDITIONAL** information:*

1. Time and Location of the Job
2. QR code(MD5 Hash):
   (a) ID of the Job
   (b) ID of the User

In case a Job needs multiple Workers - multiple Cards will be printed for all the Workers.

2.3.15 System implementation targets

*The Platform is meant to enable **MULTIPLE** Users **CONCURRENT** access to, and usage of the Platform. This would make festival organising a dynamic and fast environment. As such, a modern object-oriented programming language will be used for implementation.*

Since the application will be developed for Android OS, we have chosen Java as the programming language, and SQLite has been chosen for the database software.

2.3.16 Project Scope and Targeted Users

The software would be targeted towards people organising a festival and/or participating in its execution - administrators, technicians, investors, musicians, other kinds of artists, influencers, ...

The software, as said before, is aimed at primarily festivals of smaller sizes, but could be employed for festivals up to certain 'medium' size. Especially with improvements, it could be a viable free, open-source and easily protable alternative for both medium-sized festivals and small-sized festivals.

2.3.17 Changes, upgrades, adaptability of the Application

The application as it stands currently has some unfortunate restrictions on the freedom that Administrators, Creators and Organisers. It would be possible to modify these

restrictions as to allow a greater freedom of Festival organisation, but still keeping some in place as to prevent abuse and misuse. Certain functionalities could also be added in order to make the app more useful to mid-sized Festivals.

A few changes that would probably help tremendously is to implement an intra-platform messaging service, feedback system, and support/ticked service. This would allow the communication within the App between Users. It would also provide a way for the Developers to easily diagnose, track and reproduce bugs, as well as see what changes, ideas and updates Users would like to see in the App. Finally, a support/ticket system would help alleviate any frustration that Users could suffer due to bugs and/or failures of the platform.

A list of changes(actively tracked) to eventually, if time and will permits, be implemented:

- Email confirmation
- Email reset

3. Software Specification

3.1 Functional Requirements

Stakeholders:

1. Developers and Maintainers
2. Festival-goers
3. Festival investors and sponsors
4. Administrator
5. Creator
6. Organiser
7. Worker

Actors and their functional requirements:

1. Unregistered/Guest User(initiator) can:
   (a) Register a new account - fill in the form
      i. Username
      ii. Email
      iii. Password
      iv. Password Verification
      v. Phone Number
      vi. Name
      vii. Surname
      viii. Desired Role
         A. Leader
         B. Organiser
         C. Worker
   (b) Log in - fill in the form
      i. Username or Email
      ii. Password

2. Administrator can:
   (a) Access the list of Users
   (b) Verify Creators
   (c) Moderate Users' details and/or ban them as to alleviate abuse/misuse
   (d) Access the list of Festivals
      i. Access the list of Jobs
         A. Access the list of corresponding Activities
         B. Access the list of Workers
      ii. Moderate Festivals, Jobs and Activities and/or veto/delete them as to alleviate abuse/misuse
3. Leaders manage Festivals:
   (a) Create [multiple] Festivals
   (b) Modify or delete their Festivals
   (c) **Inherit Organiser functionalities for their own Festivals**
   (d) Appoint Organisers to their Festivals(check if the selected Organiser is organising any possibly concurrent Festivals)
   (e) Access the Jobs, Activities, Workers and other details of their Festival
4. Organiser organises the concrete Festival workflow:
   (a) Job management
      i. Select which Jobs need to be done - open their corresponding Job Auctions
      ii. Job Sequence - order and parallelise Jobs -¿ Festival Job Timeline
      iii. Ability to extend Job Auction lifetime by 1 day
      iv. View and modify Jobs
         A. Access Workers' profiles, details, comments, ...
         B. Access Job description, Time and Location - modify them as needed
         C. View each Job's list of Activities
   (b) Organise a Festival – Ability to organise multiple Festivals - check for concurrency!
5. Workers perform specific Jobs. If necessary, multiple Workers work on the same Job. They can
   (a) Select their fields of specialisation
   (b) Apply to Job Auctions

(c) Perform Job - can perform multiple Jobs - check for concurrency!

(d) Fill out Job information sheet

i. Job Description

ii. Job Location and Time

iii. Form a list of Activities that need to be done - ability to modify, add and/or delete the entries in this list

3.1.1 Use Cases

Use Cases Description

UC1 - Festivals Overview and Detail inspection

- **Main Stakeholders:** Administrators
- **Goal:** View the list of all the Festivals, ability to click on a specific Festival and view its details in a new Screen
- **Stakeholders:** Database
- **Conditions:** Must be logged in
- **Event flow description:**

  1. A list of Festivals is displayed(retrieved from the Database)
  2. Administrator selects a Festival of which he wants to inspect further details
  3. A new Screen appears depicting a detailed view of the Festival's information

  1.a Data not successfully retrieved
    1. An error message is displayed informing the User data couldn't be retrieved
    2. A 'Retry' button is displayed on the screen - upon clicking it the app attempts to fetch data again
    3. Upon successful data retrieval(if ever) the button and error message are removed and then the data is displayed

UC2 - Registration

- **Main Stakeholders:** Unregistered/Guest Users
- **Goal:** Register a new User Account
- **Stakeholders:** Database
- **Conditions:** Must not be logged in, and must be located at the Login screen.
- **Event flow description:**

  1. The User is located at the login screen, and taps the 'Create one' button, located next to the 'No account yet?' label
  2. A new Screen pops up - Guest fills the Registration Form
  3. Upon tapping 'Create Account' a new account is created and stored in the Database

2.a Illegal input
   1. An error message is displayed informing the User that they have entered illegal input
   2. The notification asks the User to re-format and re-enter the input. The expected format and rules are displayed
   3. The steps above are repeated if new input isn't accepted either

3.a Data not successfully sent and parsed on the Server
   1. An error message is displayed informing the User that an error has occurred
   2. A 'Retry' button is displayed on the screen - upon clicking it the form is resent and parsing tried again
   3. The User is notified upon success

UC3 - Log-In

- **Main Stakeholders:** Guest Users who have a registered account
- **Goal:** Log into the platform
- **Stakeholders:** Database
- **Conditions:** Must not be logged in, and must be located at the Login screen.
- **Event flow description:**
   1. User enter the email or username and their password
   2. User taps the 'Log-in' button
   3. Database checks the data and if login is successful the user is logged in

   3.a Email/Username and Password combination is wrong and the User isn't logged in.
      1. The fields Email/Username and Password are reset
      2. An error message is displayed
      3. User needs to enter the log-in data again

   3.a Data not successfully sent and parsed on the Server
      1. An error message is displayed informing the User that an error has occurred
      2. A 'Retry' button is displayed on the screen - upon clicking it the form is resent and parsing tried again
      3. The User is notified upon success

UC4 - Account Data Overview

- **Main Stakeholders:** User

- **Goal:** View the account data
- **Stakeholders:** Database
- **Conditions:** Must be logged in
- **Event flow description:**

1. User taps the sandwich button in the upper right screen corner, and then taps Account
2. Data is retrieved from the Database
3. A screen depicting Account data appears

2.a Data not successfully retrieved

1. An error message is displayed informing the User data couldn't be retrieved
2. A 'Retry' button is displayed on the screen - upon clicking it the app attempts to fetch data again
3. Upon successful data retrieval(If ever) the button and error message are removed and then the data is displayed

UC5 - Changing Account Data

- **Main Stakeholders:** User
- **Goal:** Change the account data
- **Stakeholders:** Database, Back-End(Server)
- **Conditions:** Must be logged in
- **Event flow description:**

1. User taps the sandwich button in the upper right screen corner, and then taps Account
2. Data is retrieved from the Database
3. A screen depicting Account data appears
4. User clicks the 'Change info' button and is taken to a new Screen
5. User enters the desired new data
6. User clicks the 'OK' button and the data is sent to the Database
7. Upon receiving a confirmation from the Server that change has taken place the success notification is displayed to the User

2.a Data not successfully retrieved

1. An error message is displayed informing the User data couldn't be retrieved

2. A 'Retry' button is displayed on the screen - upon clicking it the app attempts to fetch data again
3. Upon successful data retrieval(If ever) the button and error message are removed and then the data is displayed

2.a Illegal input

1. An error message is displayed informing the User that they have entered illegal input
2. The notification asks the User to re-format and re-enter the input. The expected format and rules are displayed
3. The steps above are repeated if new input isn't accepted either

6.a, 7.a Data not successfully sent or parsed on the Server

1. Field values are saved, and kept the same
2. An error is displayed to the User stating that data wasn't successfully sent and that User needs to resend the form
3. User resends the form. If still not successful, the same steps above apply.

UC6 - Account Deletion

- **Main Stakeholders:** User
- **Goal:** Delete their Account
- **Stakeholders:** Database, Back-End(Server)
- **Conditions:** Must be logged in
- **Event flow description:**

1. User taps the sandwich button in the upper right screen corner, and then taps Account
2. Data is retrieved from the Database
3. A screen depicting Account data appears
4. User clicks the 'Delete Account' button
5. A request is sent to the Server. Upon reception, the Server deletes the Account from the database.
6. The Server sends the confirmation to the User that his account has been deleted.
7. The User is logged out of the application

2.a Data not successfully retrieved

1. An error message is displayed informing the User data couldn't be retrieved

2. A 'Retry' button is displayed on the screen - upon clicking it the app attempts to fetch data again
3. Upon successful data retrieval(If ever) the button and error message are removed and then the data is displayed

5.a Data not successfully sent or parsed on the Server

1. Field values are saved, and kept the same
2. An error is displayed to the User stating that data wasn't successfully sent and that User needs to resend the form
3. User resends the form. If still not successful, the same steps above apply.

UC7 - Verifying a Leader

- **Main Stakeholders:** Administrators
- **Goal:** Check, and if all is in order, verify a Leader
- **Stakeholders:** Database, Leader
- **Conditions:** Must be logged in. There is a Leader awaiting confirmation.
- **Event flow description:**

1. Administrator opens the panel for verifying Leaders
2. Data is fetched from the Database
3. Administrator reads the info about the Leader and the Festival that the Leader wants to create
4. Administrator decides whether to verify the Leader or not

2.a Data not successfully retrieved

1. An error message is displayed informing the User data couldn't be retrieved
2. A 'Retry' button is displayed on the screen - upon clicking it the app attempts to fetch data again
3. Upon successful data retrieval(If ever) the button and error message are removed and then the data is displayed

UC8 - Create a Festival

- **Main Stakeholders:** Leader
- **Goal:** Create a Festival, and open it up to Workers so that they can start working
- **Stakeholders:** Database, Back-End(Server)

- **Conditions:** Must be logged in
- **Event flow description:**

1. The Leader opens the Screen for creating a Festival
2. The Leader fills in all the necessary info required for creating a Festival
3. The Leader submits the form and a Festival is created

2.a Illegal input

1. An error message is displayed informing the User that they have entered illegal input
2. The notification asks the User to re-format and re-enter the input. The expected format and rules are displayed
3. The steps above are repeated if new input isn't accepted either

3.a Data not successfully sent and parsed on the Server

1. An error message is displayed informing the User that an error has occurred
2. A 'Retry' button is displayed on the screen - upon clicking it the form is resent and parsing tried again
3. The User is notified upon success

UC9 - Appoint an Organiser to the selected Festival

- **Main Stakeholders:** Leader
- **Goal:** The Organiser is appointed and begins carefully managing and organising the Festival
- **Stakeholders:** Database, Back-End(Server), Organiser
- **Conditions:** Must be logged in, a Festival requires an Organiser
- **Event flow description:**

1. The Leader opens the Screen featuring their festivals
2. The Leader selects one of the Festivals
3. The Leader selects one of the Organisers, and appoints them to the selected Festival
4. This data is sent to the Server, which updates the Database
5. A notification is sent to the Organiser, who can either accept or reject the said Festival

4.a Data not successfully sent and/or parsed on the Server

1. An error message is displayed informing the User that an error has occurred

2. The Leader can try resending the form
3. The Leader is notified upon success

5.a Attempt to send a notification to the Organiser fails.

1. The Server will resend it until success
2. Upon success, the Leader is notified that the action of requesting an Organiser is successful

UC10 - Update Festival info

- **Main Stakeholders:** Leader
- **Goal:** Update/modify Festival details
- **Stakeholders:** Database, Back-End(Server), Organiser, Workers
- **Conditions:** Must be logged in
- **Event flow description:**

1. The Leader opens the Screen for creating a Festival
2. The Leader fills in all the necessary info required for creating a Festival
3. The Leader submits the form and a Festival is created

2.a Illegal input

1. An error message is displayed informing the User that they have entered illegal input
2. The notification asks the User to re-format and re-enter the input. The expected format and rules are displayed
3. The steps above are repeated if new input isn't accepted either

3.a Data not successfully sent and/or parsed on the Server

1. An error message is displayed informing the User that an error has occurred
2. A 'Retry' button is displayed on the screen - upon clicking it the form is resent and parsing tried again
3. The Leader is notified upon success

UC11 - Create a Job

- **Main Stakeholders:** Organiser, Leader
- **Goal:** Create a Job entry that would be visible to Workers who would apply to this Job via Job Auctions
- **Stakeholders:** Database, Back-End(Server), Worker
- **Conditions:** Must be logged in

- **Event flow description:**
  1. The Organiser opens the Screen for creating a Job
  2. The Leader fills in all the necessary info required for creating a Job
  3. The Leader submits the form, the Job is created and added to the list of Jobs(this list is visible to Workers who can then send their application to Organisers for the selected Job)

  2.a Illegal input
    1. An error message is displayed informing the User that they have entered illegal input
    2. The notification asks the User to re-format and re-enter the input. The expected format and rules are displayed
    3. The steps above are repeated if new input isn't accepted either

  3.a Data not successfully sent and/or parsed on the Server
    1. An error message is displayed informing the User that an error has occurred
    2. A 'Retry' button is displayed on the screen - upon clicking it the form is resent and parsing tried again
    3. The Organiser is notified upon success

UC12 - View the list of all the Jobs(Worker View)

- **Main Stakeholders:** Worker, Administrators
- **Goal:** View the list of Jobs. Includes the ability to filter the Jobs by Categories.
- **Stakeholders:** Database, Leaders, Organisers
- **Conditions:** Must be logged in
- **Event flow description:**
  1. The User opens the Screen for viewing the list of Jobs
  2. The data is fetched from the Database
  3. Optional: Filtering the Jobs according to the specified filter
  4. The User selects the Job and views the details about it(the 'View Job details' button)

  2.a Illegal input
    1. An error message is displayed informing the User that they have entered illegal input
    2. The notification asks the User to re-format and re-enter the input. The expected format and rules are displayed

3. The steps above are repeated if new input isn't accepted either

UC13 - View Job details(Worker view)

- **Main Stakeholders:** Worker, Administrators
- **Goal:** View Job details and specifics - such as Time, Location, Duration, Description, ...
- **Stakeholders:** Database, Leader, Organiser
- **Conditions:** Must be logged in, must have selected a Job
- **Event flow description:**

1. The User can read Job specifics
2. The Worker can click the 'Send Job application' button in order to apply to the given Job
3. The Administrator can click the 'Moderate this Job' button

1.a Data not successfully retrieved

1. An error message is displayed informing the User data couldn't be retrieved
2. A 'Retry' button is displayed on the screen - upon clicking it the app attempts to fetch data again
3. Upon successful data retrieval(If ever) the button and error message are removed and then the data is displayed

UC14 - Apply to the Job(Worker)

- **Main Stakeholders:** Worker
- **Goal:** Send his application to the Job's supervising Organiser/Leader
- **Stakeholders:** Database, Back-End(Server), Organiser, Leader
- **Conditions:** Must be logged in, the Job details Screen is opened
- **Event flow description:**

1. The Worker on this Screen fills out the details into the form(Application Form)
2. Upon filling out the form, he sends the form by pressing the 'Send Job Application' button
3. The Application is sent to the Server

1.a Illegal input

1. An error message is displayed informing the User that they have entered illegal input

2. The notification asks the User to re-format and re-enter the input. The expected format and rules are displayed
3. The steps above are repeated if new input isn't accepted either

3.a Data not successfully sent to the Server

1. An error message is displayed informing the User data wasn't successfully sent to the Server
2. A 'Retry' button is displayed on the screen - upon clicking it the app attempts to send the data again
3. Upon success(if ever) the Screen is closed. Otherwise the aforementioned procedure is again executed

UC15 - Moderate the Job

- **Main Stakeholders:** Administrators
- **Goal:** Moderating the selected Job - in case certain part of it violates the Rules
- **Stakeholders:** Database, Back-End(Server), Worker, Organiser
- **Conditions:** Must be logged in, and must have selected a Job to moderate
- **Event flow description:**

1. The Administrator alters certain part of the Job, or deletes it(by pressing the 'Delete Job' button)
2. The Administrator chooses whether the Job creator(corresponding Organiser or Leader) is notified or not
3. The changes are sent to the Server

1.a Illegal input

1. An error message is displayed informing the User that they have entered illegal input
2. The notification asks the User to re-format and re-enter the input. The expected format and rules are displayed
3. The steps above are repeated if new input isn't accepted either

3.a Data not successfully sent to the Server

1. An error message is displayed informing the User data wasn't successfully sent to the Server
2. A 'Retry' button is displayed on the screen - upon clicking it the app attempts to send the data again
3. Upon success(if ever) the Screen is closed. Otherwise the aforementioned procedure is again executed

UC16 - View the list of this Festival Jobs

- **Main Stakeholders:** Administrators, Organiser, Leader
- **Goal:** View the list of Jobs for the currently selected Festival
- **Stakeholders:** Database, Worker
- **Conditions:** Must be logged in
- **Event flow description:**

1. The data is fetched from the Database
2. The list is displayed to the User
3. The User can press the 'View Job details' button which will take them to a new Screen where they can view the Job details and specifics, modify or delete the Job

1.a Data not successfully retrieved

1. An error message is displayed informing the User data couldn't be retrieved
2. A 'Retry' button is displayed on the screen - upon clicking it the app attempts to fetch data again
3. Upon successful data retrieval(If ever) the button and error message are removed and then the data is displayed

UC17 - View Job details(Organiser view)

- **Main Stakeholders:** Administrators, Leader, Organiser
- **Goal:** View the details and specifics of the selected Job. Can also proceed to the Screen for modifying or deleting the Job
- **Stakeholders:** Database
- **Conditions:** Must be logged in and must have selected this Job by clicking on it on the list of Jobs
- **Event flow description:**

1. The data is fetched from the Database
2. The Job details and specifics are displayed to the User
3. Leaders and Organisers can click the 'Modify this Job details' or the 'Delete this Job' button in order to remove it, taking them to a new Screen
4. Leaders and Organisers can click the 'See Worker Applications' button taking them to a new Screen

5. The Administrator can click the 'Moderate this Job' button taking them to a new Screen

1.a Data not successfully retrieved
   1. An error message is displayed informing the User data couldn't be retrieved
   2. A 'Retry' button is displayed on the screen - upon clicking it the app attempts to fetch data again
   3. Upon successful data retrieval(If ever) the button and error message are removed and then the data is displayed

UC18 - Modify Job details

- **Main Stakeholders:** Leader, Organiser
- **Goal:** Edit the details and specifics of the selected Job
- **Stakeholders:** Database, Back-End(Server), Worker
- **Conditions:** Must be logged in and must have selected this Job by clicking on it on the list of Jobs
- **Event flow description:**
   1. A new Screen is shown to the User featuring all the Job details, but their editing is enabled
   2. The User arbitrarily edits the Job
   3. Upon being done, they can press the 'Save changes' button - the changes are sent to the Server

1.a Data not successfully retrieved
   1. An error message is displayed informing the User data couldn't be retrieved
   2. A 'Retry' button is displayed on the screen - upon clicking it the app attempts to fetch data again
   3. Upon successful data retrieval(If ever) the button and error message are removed and then the data is displayed

2.a Illegal input
   1. An error message is displayed informing the User that they have entered illegal input
   2. The notification asks the User to re-format and re-enter the input. The expected format and rules are displayed
   3. The steps above are repeated if new input isn't accepted either

3.a Data not successfully sent to the Server
   1. An error message is displayed informing the User data wasn't successfully sent to the Server
   2. A 'Retry' button is displayed on the screen - upon clicking it the app attempts to send the data again
   3. Upon success(if ever) the Screen is closed. Otherwise the aforementioned procedure is again executed

UC19 - View the list of all the Festivals

- **Main Stakeholders:** Administrators
- **Goal:** View the list of all the Festivals
- **Stakeholders:** Database, Back-End(Server)
- **Conditions:** Must be logged in
- **Event flow description:**
   1. Data is fetched from the Server
   2. A new Screen is shown to the Administrator featuring all the Festivals
   3. The Administrator arbitrarily selects the Festival
   4. They are taken to a new screen where Festival details are displayed - data retrieved from the Server

1.a, 4.a Data not successfully retrieved
   1. An error message is displayed informing the User data couldn't be retrieved
   2. A 'Retry' button is displayed on the screen - upon clicking it the app attempts to fetch data again
   3. Upon successful data retrieval(If ever) the button and error message are removed and then the data is displayed

UC20 - Moderate the Festival

- **Main Stakeholders:** Administrators
- **Goal:** Modify or delete the selected Festival
- **Stakeholders:** Database, Back-End(Server), Organiser, Leader
- **Conditions:** Must be logged in and must have selected the Festival to be moderated
- **Event flow description:**
   1. Data is fetched from the Server

2. A new Screen is shown to the Administrator featuring Festival details
3. The Administrator arbitrarily edits the data
4. The Administrator clicks the 'Save changes' button or the 'Delete Festival' button, or he just clicks the backward arrow and is taken to the previous Screen
5. Changes are sent to the Server and saved

1.a Data not successfully retrieved
   1. An error message is displayed informing the User data couldn't be retrieved
   2. A 'Retry' button is displayed on the screen - upon clicking it the app attempts to fetch data again
   3. Upon successful data retrieval(If ever) the button and error message are removed and then the data is displayed

3.a Illegal input
   1. An error message is displayed informing the User that they have entered illegal input
   2. The notification asks the User to re-format and re-enter the input. The expected format and rules are displayed
   3. The steps above are repeated if new input isn't accepted either

5.a Data not successfully sent to the Server
   1. An error message is displayed informing the User data wasn't successfully sent to the Server
   2. A 'Retry' button is displayed on the screen - upon clicking it the app attempts to send the data again
   3. Upon success(if ever) the Screen is closed. Otherwise the aforementioned procedure is again executed

UC21 - View the list of own Festivals(Leader, Organiser)

- **Main Stakeholders:** Leader, Organiser
- **Goal:** See the list of all the Festivals that the Leader created, or that the Organiser organises
- **Stakeholders:** Database, Back-End(Server), Workers
- **Conditions:** Must be logged in
- **Event flow description:**
   1. The User presses the sandwich button and then presses the 'View my Festivals' button

2. Data is fetched from the Server
3. A new Screen is shown, featuring the list of Festivals
4. The Leader/Organiser can tap on the Festival to open up its details
5. The Leader can create, remove, or edit Festivals

1.a, 4.a Data not successfully retrieved

1. An error message is displayed informing the User data couldn't be retrieved
2. A 'Retry' button is displayed on the screen - upon clicking it the app attempts to fetch data again
3. Upon successful data retrieval(If ever) the button and error message are removed and then the data is displayed

UC22 - View the Festival details(Leader/Organiser)

- **Main Stakeholders:** Leader, Organiser
- **Goal:** Inspect Festival details and specifics
- **Stakeholders:** Database, Back-End(Server), Workers
- **Conditions:** Must be logged in, must have selected a Festival
- **Event flow description:**

1. Data is fetched from the Server
2. Festival details are displayed to the User
3. The Leader/Organiser can click the 'Modify Festival details' button which will take them to a new Screen
4. The Leader can click the 'Delete the Festival' button. This will take them to a new Screen for Festival deletion.

1.a Data not successfully retrieved

1. An error message is displayed informing the User data couldn't be retrieved
2. A 'Retry' button is displayed on the screen - upon clicking it the app attempts to fetch data again
3. Upon successful data retrieval(If ever) the button and error message are removed and then the data is displayed

UC23 - Remove the Festival(Leader)

- **Main Stakeholders:** Leader
- **Goal:** Delete their Festival

- **Stakeholders:** Database, Back-End(Server), Organiser, Workers
- **Conditions:** Must be logged in, must have selected a Festival
- **Event flow description:**
  1. The Leader is asked if they're sure of deleting the Festival
  2. If yes is pressed again, they're asked if they're **REALLY** sure of deleting the Festival
  3. If yes is again pressed, then the Festival is deleted.
  4. All the stakeholders of the Festival(Organiser and Workers) are sent notifications, and their accounts are updated accordingly
  5. Changes are sent to the Server

4.a, 5.a Data not successfully sent to the Server
  1. An error message is displayed informing the User data wasn't successfully sent to the Server
  2. A 'Retry' button is displayed on the screen - upon clicking it the app attempts to send the data again
  3. Upon success(if ever) the Screen is closed. Otherwise the aforementioned procedure is again executed

UC24 - View Festival Job Timeline

- **Main Stakeholders:** Administrators, Leader, Organiser
- **Goal:** Inspect Festival Timeline regarding Jobs - provide Job overview
- **Stakeholders:** Database, Back-End(Server), Festival Job Timeline
- **Conditions:** Must be logged in, must have selected a Festival and then selected the button for viewing the Festival Job Timeline
- **Event flow description:**
  1. Data is retrieved from the Server
  2. The Festival Job Timeline is displayed to the User
  3. The User can click on a Job to open up its detail and manipulation Screen

1.a Data not successfully retrieved
  1. An error message is displayed informing the User data couldn't be retrieved
  2. A 'Retry' button is displayed on the screen - upon clicking it the app attempts to fetch data again
  3. Upon successful data retrieval(If ever) the button and error message are removed and then the data is displayed

UC25 - Remove a Job

- **Main Stakeholders:** Administrators, Leader, Organiser
- **Goal:** Remove the selected Job
- **Stakeholders:** Database, Back-End(Server), Worker
- **Conditions:** Must be logged in, must have selected a Job
- **Event flow description:**

1. The User is asked if they're sure of deleting the Job
2. If yes is pressed again, they're asked if they're **REALLY** sure of deleting the Job
3. If yes is again pressed, then the Job is deleted
4. All the stakeholders of the Job(Festival Leader, Organiser and affected Workers) are sent notifications, and their accounts are updated accordingly
5. Changes are sent to the Server

5.a Data not successfully sent to the Server

1. An error message is displayed informing the User data wasn't successfully sent to the Server
2. A 'Retry' button is displayed on the screen - upon clicking it the app attempts to send the data again
3. Upon success(if ever) the Screen is closed. Otherwise the aforementioned procedure is again executed

UC26 - Managing Festival Job Timeline

- **Main Stakeholders:** Administrators, Leader, Organiser
- **Goal:** Organise Jobs in an effective manner from a time point of view
- **Stakeholders:** Database, Back-End(Server), Workers, Festival Job Timeline
- **Conditions:** Must be logged in, must have selected a Festival and then its Festival Job Timeline
- **Event flow description:**

1. Data is retrieved from the Server
2. The User can add a Job to the Timeline by clicking the 'Add a Job to the Timeline' taking them to a new Screen
3. The User can remove a Job from the Timeline by selecting it on the Timeline and then dragging it out of the Screen - a confirmation dialogue will appear

4. The User can edit a Job by long pressing anywhere on it and then on the pop-up menu selecting the 'Edit Job' option
5. The User can create a new Timeline Branch by double tapping anywhere on the Timeline
6. The User can remove a Branch long pressing anywhere on it, and then on the pop-up menu selecting the 'Remove Branch' option
7. The User can change the Time of a Job or a Branch by either dragging them relative to the Time Axis, or by long pressing and selecting 'Change Time' option
8. The User can change the duration by changing the dimensions of the Job on the graph - elongating or shortening it relative to the time axis, or by long pressing and selectin the 'Change Duration' option
9. Data is sent to the Server, and changes there are saved

1.a Data not successfully retrieved due to connection timeout(can be caused by various network problems)
   1. An error message is displayed informing the User data couldn't be retrieved
   2. A 'Retry' button is displayed on the screen - upon clicking it the app attempts to fetch data again
   3. Upon successful data retrieval(If ever) the button and error message are removed and then the data is displayed

2.a Illegal input
   1. An error message is displayed informing the User that they have entered illegal input
   2. The notification asks the User to re-format and re-enter the input. The expected format and rules are displayed
   3. The steps above are repeated if new input isn't accepted either

8.a Data not successfully sent to the Server
   1. An error message is displayed informing the User data wasn't successfully sent to the Server
   2. A 'Retry' button is displayed on the screen - upon clicking it the app attempts to send the data again
   3. Upon success(if ever) the Screen is closed. Otherwise the aforementioned procedure is again executed

UC27 - Add a Job to the Festival Timeline(Screen)

- **Main Stakeholders:** Leader, Organiser
- **Goal:** Add a Job onto the Festival Timeline Screen - so that it can be further managed, organised and/or deleted
- **Stakeholders:** Database, Back-End(Server), Workers, Festival Job Timeline
- **Conditions:** Must be logged in, must have selected a Festival, its Festival Job Timeline and then selected the button 'Add a Job to the Timeline'
- **Event flow description:**

  1. The Job is added to the Festival Job Timeline
  2. The User is notified of the success
  3. Changes are sent to the Server
  4. The User is taken back

  3.a Data not successfully sent to the Server
     1. An error message is displayed informing the User data wasn't successfully sent to the Server
     2. A 'Retry' button is displayed on the screen - upon clicking it the app attempts to send the data again
     3. Upon success(if ever) the Screen is closed. Otherwise the aforementioned procedure is again executed

UC28 - View Job Worker Application Entries of the current Festival

- **Main Stakeholders:** Leader, Organiser
- **Goal:** View the list of all the Job Worker Application Entries, so that they can be verified in order for Workers to enter the Auction process
- **Stakeholders:** Database, Back-End(Server), Workers
- **Conditions:** Must be logged in, must have selected a Festival
- **Event flow description:**

  1. Tap the button to 'View the Worker Job Application Entries'
  2. The list is fetched from the Server
  3. The User is taken to a new Screen where all the entries are displayed
  4. Each Entry has an 'Inspect entry' button - upon pressing it the User it taken to a new Screen

  2.a Data not successfully retrieved
     1. An error message is displayed informing the User data couldn't be retrieved

2. A 'Retry' button is displayed on the screen - upon clicking it the app attempts to fetch data again
3. Upon successful data retrieval(If ever) the button and error message are removed and then the data is displayed

UC29 - Inspect Job Worker Application Entry

- **Main Stakeholders:** Leader, Organiser
- **Goal:** Inspect the Job Entry more closely as to be able to make a more quality decision on whether to verify this Entry or not
- **Stakeholders:** Database, Back-End(Server), Worker(s)
- **Conditions:** Must be logged in, must have selected a Festival and a Job Entry
- **Event flow description:**

1. Data is fetched from the Server
2. The User inspects the data
3. If deemed fit, the User can verify the Entry
4. The changes are sent to the Server
5. By tapping the button 'Back' the User can go back to the Job Application Entries List

1.a Data not successfully retrieved
1. An error message is displayed informing the User data couldn't be retrieved
2. A 'Retry' button is displayed on the screen - upon clicking it the app attempts to fetch data again
3. Upon successful data retrieval(If ever) the button and error message are removed and then the data is displayed

4.a Data not successfully sent to the Server
1. An error message is displayed informing the User data wasn't successfully sent to the Server
2. A 'Retry' button is displayed on the screen - upon clicking it the app attempts to send the data again
3. Upon success(if ever) the Screen is closed. Otherwise the aforementioned procedure is again executed

UC30 - Verify Worker Job Application entry

- **Main Stakeholders:** Leader, Organiser

- **Goal:** Upon inspection, if the Job Entry is deemed quality, verify this Entry so that it can compete with other Entries for the final approval
- **Stakeholders:** Database, Back-End(Server), Worker(s)
- **Conditions:** Must be logged in, must have selected a Job to verify
- **Event flow description:**

  1. Upon clicking the 'Verify this Job Application Entry' this Worker's Entry is verified
  2. It enters the Job Auction process
  3. Data is sent to the Server

  3.a Data not successfully sent to the Server
    1. An error message is displayed informing the User data wasn't successfully sent to the Server
    2. A 'Retry' button is displayed on the screen - upon clicking it the app attempts to send the data again
    3. Upon success(if ever) the Screen is closed. Otherwise the aforementioned procedure is again executed

UC31 - Manage Job Activities

- **Main Stakeholders:** Worker(s), Administrators
- **Goal:** Manage and modify the list of all the Activities that make up a Job
- **Stakeholders:** Database, Back-End(Server), Leader, Organiser
- **Conditions:** Must be logged in, must have selected a Job of which the Activities will be viewed and must have tapped the 'Manage Job Activities' button of the selected Job
- **Event flow description:**

  1. Data(the list) is fetched from the Server and displayed to the User
  2. The User can select a Specific Activity to inspect its Details
  3. The User can add or remove Activities
  4. The User can edit the Job Activity - if he is working with multiple Workers they are notified of this change via Push-Up Notifications
  5. The User can tap the 'View Job Activities Timeline' in order to manage the time-dimension of the aforementioned Activities

  1.a, 4.a Data not successfully retrieved
    1. An error message is displayed informing the User data couldn't be retrieved

2. A 'Retry' button is displayed on the screen - upon clicking it the app attempts to fetch data again
3. Upon successful data retrieval(If ever) the button and error message are removed and then the data is displayed

UC32 - Inspect Job Activity details

- **Main Stakeholders:** Worker(s), Administrators
- **Goal:** Further inspect and view the details of a specific Job Activity
- **Stakeholders:** Database, Back-End(Server), Leader, Organiser
- **Conditions:** Must be logged in, must have selected a Job and one of its Activities
- **Event flow description:**

1. Data(Job details) is fetched from the Server
2. The details are displayed to the User
3. The User can remove or modify this Activity
4. If any changes are made, they are sent to the Server

1.a Data not successfully retrieved

1. An error message is displayed informing the User data couldn't be retrieved
2. A 'Retry' button is displayed on the screen - upon clicking it the app attempts to fetch data again
3. Upon successful data retrieval(If ever) the button and error message are removed and then the data is displayed

3.a Data not successfully sent to the Server

1. An error message is displayed informing the User data wasn't successfully sent to the Server
2. A 'Retry' button is displayed on the screen - upon clicking it the app attempts to send the data again
3. Upon success(if ever) the Screen is closed. Otherwise the aforementioned procedure is again executed

UC33 - Modify Job Activity

- **Main Stakeholders:** Worker(s), Administrators
- **Goal:** Modify Job Activity details
- **Stakeholders:** Database, Back-End(Server), Leader, Organiser

- **Conditions:** Must be logged in, must have selected a Job and one of its Activities
- **Event flow description:**

1. Data(Job details) is fetched from the Server
2. The details are displayed to the User – editable
3. The User arbitrarily modifies the data
4. If any changes are made, they are sent to the Server

1.a Data not successfully retrieved
   1. An error message is displayed informing the User data couldn't be retrieved
   2. A 'Retry' button is displayed on the screen - upon clicking it the app attempts to fetch data again
   3. Upon successful data retrieval(If ever) the button and error message are removed and then the data is displayed

3.a Illegal input
   1. An error message is displayed informing the User that they have entered illegal input
   2. The notification asks the User to re-format and re-enter the input. The expected format and rules are displayed
   3. The steps above are repeated if new input isn't accepted either

4.a Data not successfully sent to the Server
   1. An error message is displayed informing the User data wasn't successfully sent to the Server
   2. A 'Retry' button is displayed on the screen - upon clicking it the app attempts to send the data again
   3. Upon success(if ever) the Screen is closed. Otherwise the aforementioned procedure is again executed

UC34 - Create Job Activity

- **Main Stakeholders:** Worker(s)
- **Goal:** Create a new Job Activity
- **Stakeholders:** Database, Back-End(Server), Leader, Organiser
- **Conditions:** Must be logged in, must have tapped the button to create a new Job Activity for a specific Job
- **Event flow description:**

1. The User fills the form out arbitrarily
2. Upon completion, they tap the ‘Create Job Activity‘
3. The updated data is sent to the Server and saved

1.a Illegal input
   1. An error message is displayed informing the User that they have entered illegal input
   2. The notification asks the User to re-format and re-enter the input. The expected format and rules are displayed
   3. The steps above are repeated if new input isn’t accepted either

3.a Data not successfully sent to the Server
   1. An error message is displayed informing the User data wasn’t successfully sent to the Server
   2. A ’Retry’ button is displayed on the screen - upon clicking it the app attempts to send the data again
   3. Upon success(if ever) the Screen is closed. Otherwise the aforementioned procedure is again executed

UC35 - Remove Job Activity

- **Main Stakeholders:** Worker, Administrators
- **Goal:** Remove the selected Job Activity
- **Stakeholders:** Database, Back-End(Server)
- **Conditions:** Must be logged in, must have selected a Job and the Activity to be deleted
- **Event flow description:**

   1. The User is prompted whether they are sure of the deletion
   2. They are again prompted, just in case
   3. The Activity is removed, and changes sent to the Server

   3.a Data not successfully sent to the Server
      1. An error message is displayed informing the User data wasn’t successfully sent to the Server
      2. A ’Retry’ button is displayed on the screen - upon clicking it the app attempts to send the data again
      3. Upon success(if ever) the Screen is closed. Otherwise the aforementioned procedure is again executed

UC36 - Manage Job Activities Timeline(Worker View)

- **Main Stakeholders:** Worker(s), Administrators
- **Goal:** Inspect and modify(in Administrator's case moderate) the list of Activities of the given Job
- **Stakeholders:** Database, Back-End(Server), Leader, Organiser, Job Activities Timeline
- **Conditions:** Must be logged in, must have selected the option to go to Job Activity Timeline Management Screen
- **Event flow description:**

1. Data is fetched from the Server
2. The time-relative graph is displayed to the User
3. Job Activity Timeline Management offers the same functionalities as the Festival Job Timeline Management interface
4. The User can add an Activity to the Timeline by tapping the 'Add an Activity to the Timeline' button taking them to a new Screen
5. The User can remove an Activity from the Timeline by selecting it on the Timeline and then dragging it out of the Screen - a confirmation dialogue will appear
6. The User can create a new Timeline Branch by double tapping anywhere on the Timeline
7. The User can remove a Branch long pressing anywhere on it, and then on the pop-up menu selecting the 'Remove Branch' option
8. The User can change the Time of an Activity or a Branch by either dragging them relative to the Time Axis, or by long pressing and selecting 'Change Time' option
9. The User can change the duration by changing the dimensions of the Activity on the graph - elongating or shortening it relative to the time axis, or by long pressing and selecting 'Change Duration' option
10. When done with changing the diagram, the User can tap the 'Save changes' button - data is sent to the Server and saved there
11.

1.a Data not successfully retrieved
    1. An error message is displayed informing the User data couldn't be retrieved
    2. A 'Retry' button is displayed on the screen - upon clicking it the app attempts to fetch data again

3. Upon successful data retrieval(If ever) the button and error message are removed and then the data is displayed

4.a Data not successfully sent to the Server

1. An error message is displayed informing the User data wasn't successfully sent to the Server
2. A 'Retry' button is displayed on the screen - upon clicking it the app attempts to send the data again
3. Upon success(if ever) the Screen is closed. Otherwise the aforementioned procedure is again executed

UC37 - Add Job Activity to Timeline

- **Main Stakeholders:** Worker(s), Administrators
- **Goal:** Add the selected Job Activity to the Timeline so that its Time parameters can be set and manipulated
- **Stakeholders:** Database, Back-End(Server), Leader, Organiser, Job Activities Timeline
- **Conditions:** Must be logged in, must have selected a Job and the Activity to be added to the Job Activity Timeline
- **Event flow description:**

1. The User is prompted whether they are sure of the addition
2. Upon confirmation, the Activity is added to the Timeline and can be manipulated there
3. The confirmation is sent to the Server in order to be parsed and saved

3.a Data not successfully sent to the Server

1. An error message is displayed informing the User data wasn't successfully sent to the Server
2. A 'Retry' button is displayed on the screen - upon clicking it the app attempts to send the data again
3. Upon success(if ever) the Screen is closed. Otherwise the aforementioned procedure is again executed

UC38 - Print Festival Card

- **Main Stakeholders:** All Users
- **Goal:** Festival Stakeholder can print Cards granting them an entrance to the Festival itself

- **Stakeholders:** Database, Back-End(Server)
- **Conditions:** Must be logged in
- **Event flow description:**

1. The User selects one of the Festivals or Jobs that they're responsible for
2. They download the .pdf of the Card from the Server

2.a Data not successfully retrieved

1. An error message is displayed informing the User data couldn't be retrieved
2. A 'Retry' button is displayed on the screen - upon clicking it the app attempts to fetch data again
3. Upon successful data retrieval(If ever) the button and error message are removed and then the data is displayed

UC39 - Job Entries Auction Process

- **Main Stakeholders:** Worker
- **Goal:** Select the Workers according to their Job Entries who will be given the task of performing the Job
- **Stakeholders:** Database, Back-End(Server), Leader, and Organiser
- **Conditions:** Must be logged in
- **Event flow description:**

1. After a specified period, according to the Job requirements, one or more Workers' entries are selected
2. The selected Workers are notified
3. The Jobs are added to the Worker's list of Jobs to be performed
4. Data is uploaded to the Server

4.a Data not successfully sent to the Server

1. An error message is displayed informing the User data wasn't successfully sent to the Server
2. A 'Retry' button is displayed on the screen - upon clicking it the app attempts to send the data again
3. Upon success(if ever) the Screen is closed. Otherwise the aforementioned procedure is again executed

UC40 - Extend Job Auction by 1 day

- **Main Stakeholders:** Leader, Organiser

- **Goal:** Extend the Job Auction by 1 day due to various reasons - up to Leader/Organiser discretion
- **Stakeholders:** Database, Back-End(Server), Workers
- **Conditions:** Must be logged in
- **Event flow description:**

  1. The User taps the button to extend the Auction by 1 day
  2. The Auction is extended by 1 day
  3. Changes are sent to the Server to be parsed and saved there

  3.a Data not successfully sent to the Server
     1. An error message is displayed informing the User data wasn't successfully sent to the Server
     2. A 'Retry' button is displayed on the screen - upon clicking it the app attempts to send the data again
     3. Upon success(if ever) the Screen is closed. Otherwise the aforementioned procedure is again executed

UC41 - View list of Job Auctions

- **Main Stakeholders:** Leader, Organiser, Administrators, Workers
- **Goal:** View the list of all the Job Auctions(both on-going and finished)
- **Stakeholders:** Database, Back-End(Server)
- **Conditions:** Must be logged in, and must have selected a Festival of which Job Auctions will be displayed
- **Event flow description:**

  1. The User taps the button to view the list of Job Auctions
  2. The list of Job Auctions is displayed to the User
  3. The User can add, remove or further inspect various Job Auctions

UC42 - Modify Job Activity

- **Main Stakeholders:** Worker(s), Administrator
- **Goal:** Modify Job Activities due to various reasons and necessities - up to lead Worker's discretion. Administrator can moderate the Job Activity.
- **Stakeholders:** Database, Back-End(Server), Leader, Organiser, Job Activities Timeline
- **Conditions:** Must be logged in, and must have selected a Job and one of its Activities to modify

- **Event flow description:**
  1. The User modifies the Activity data arbitrarily
  2. The User confirms the changes
  3. If an Administrator moderated, the affected Stakeholders are notified(Worker(s), Leader, and Organiser of the affected Festival)
  4. The changes are sent to the Server to be saved

  1.a Illegal input
    1. An error message is displayed informing the User that they have entered illegal input
    2. The notification asks the User to re-format and re-enter the input. The expected format and rules are displayed
    3. The steps above are repeated if new input isn't accepted either

  4.a Data not successfully sent to the Server
    1. An error message is displayed informing the User data wasn't successfully sent to the Server
    2. A 'Retry' button is displayed on the screen - upon clicking it the app attempts to send the data again
    3. Upon success(if ever) the Screen is closed. Otherwise the aforementioned procedure is again executed

UC43 - Job awarded to Worker

- **Main Stakeholders:** Worker(s)
- **Goal:** Upon winning the Auction, the Worker is awarded the Job
- **Stakeholders:** Database, Back-End(Server), Leader, Organiser
- **Conditions:** Must be logged in, and must have won a Job Auction
- **Event flow description:**
  1. The Worker(s) is awarded the Job
  2. Notify the Leader and Organiser of the corresponding Festival
  3. Update the Worker(s)' profile
  4. Send to, and save data on the Server

  4.a Data not successfully sent to the Server
    1. An error message is displayed informing the User data wasn't successfully sent to the Server
    2. A 'Retry' button is displayed on the screen - upon clicking it the app attempts to send the data again

3. Upon success(if ever) the Screen is closed. Otherwise the aforementioned procedure is again executed

Use Case Diagrams

Prikazati odnos aktora i obrazaca uporabe odgovarajuim UML dijagramom. Nije nuno nacrtati sve na jednom dijagramu. Modelirati po razinama apstrakcije i skupovima srodnih funkcionalnosti.

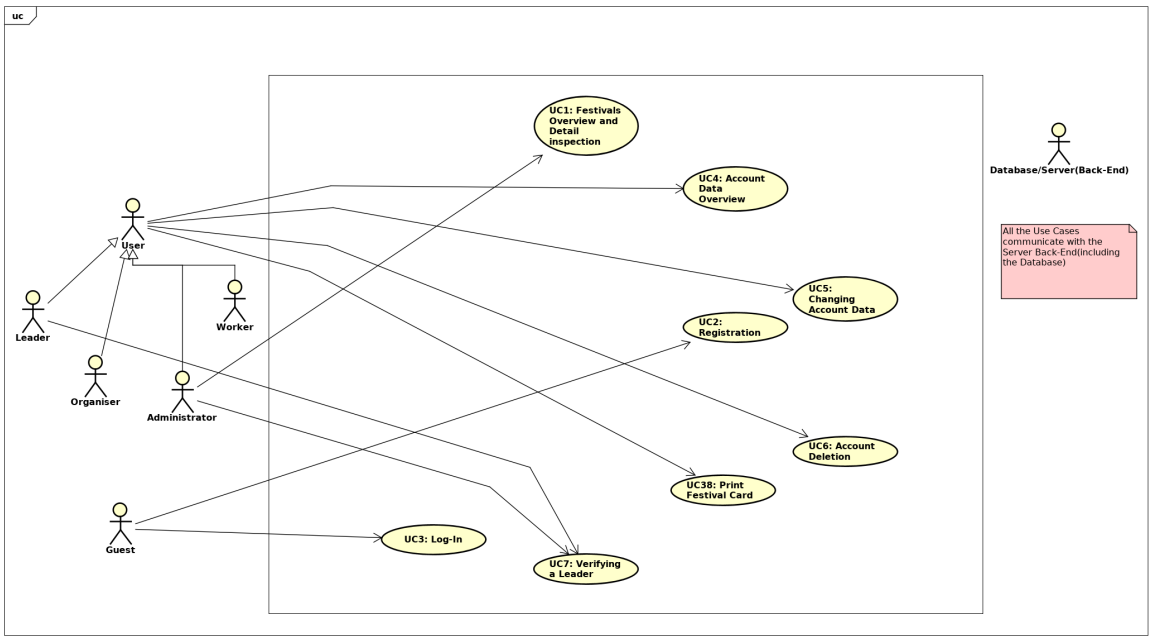

Figure 3.1: Use Case diagram - General Account Usage

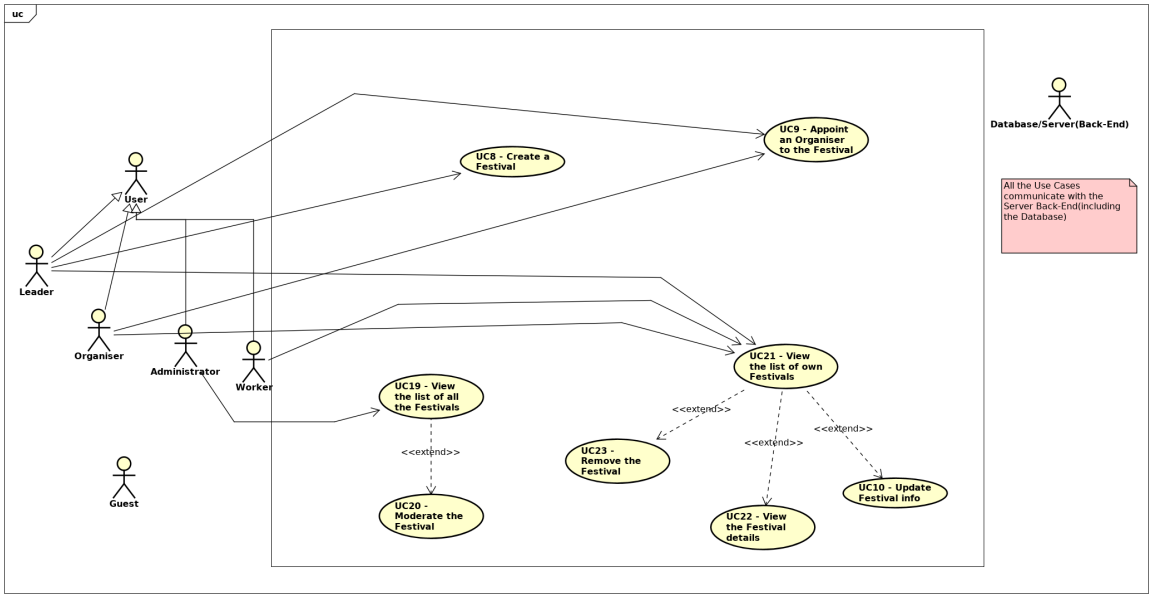

Figure 3.2: Use Case diagram - Festival

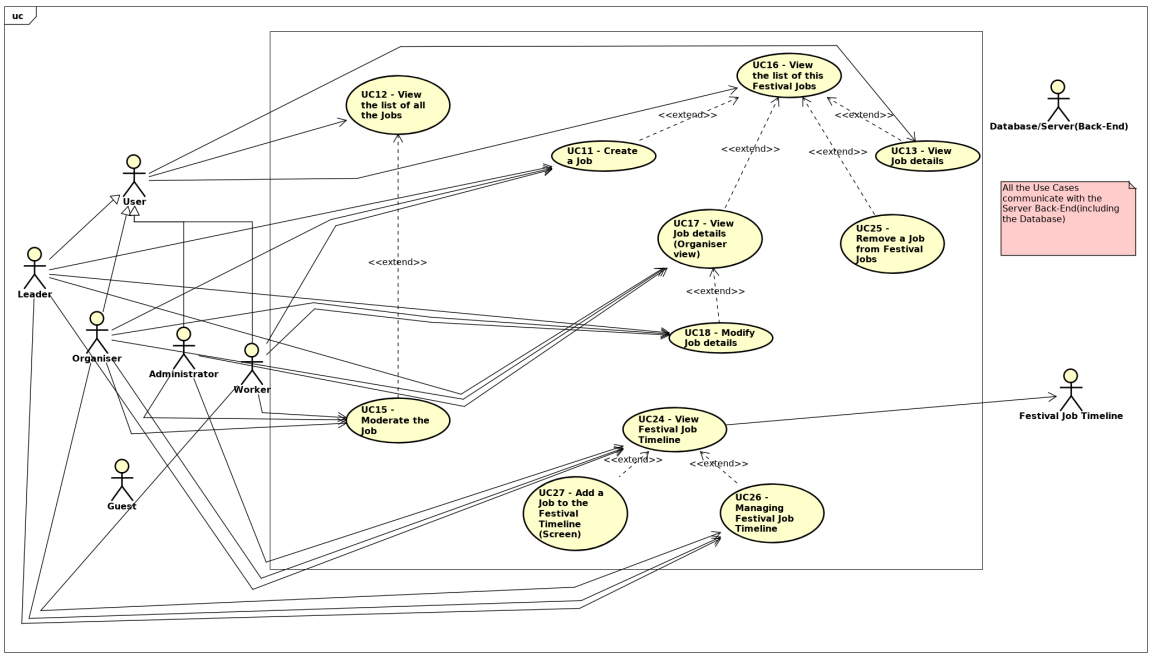

Figure 3.3: Use Case diagram - Job

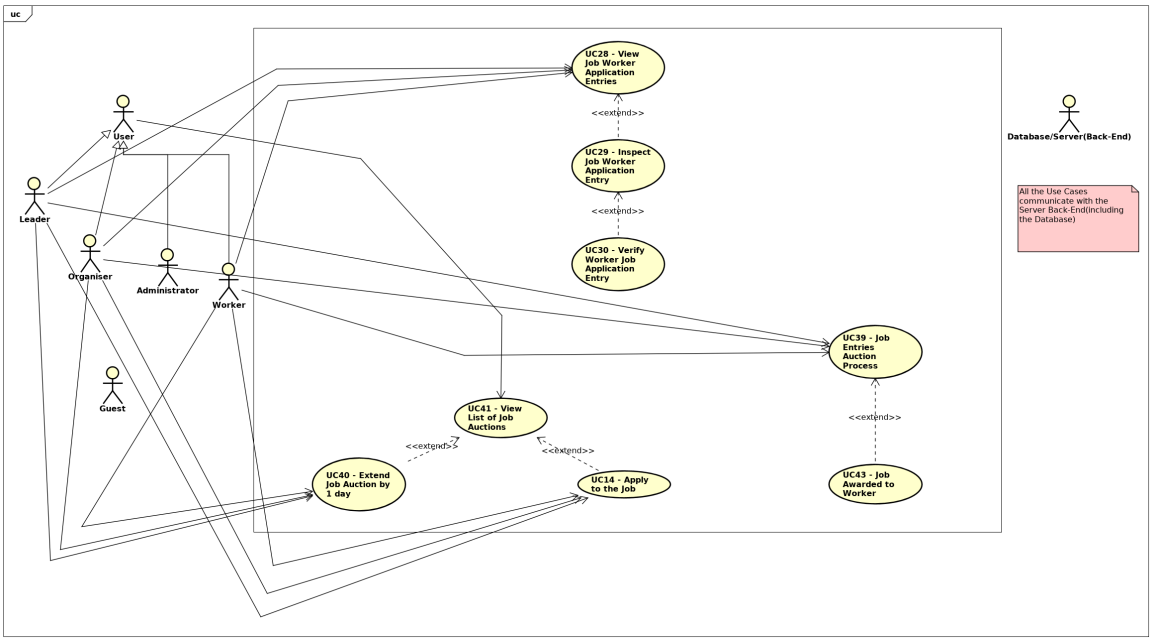

Figure 3.4: Use Case diagram - Auction

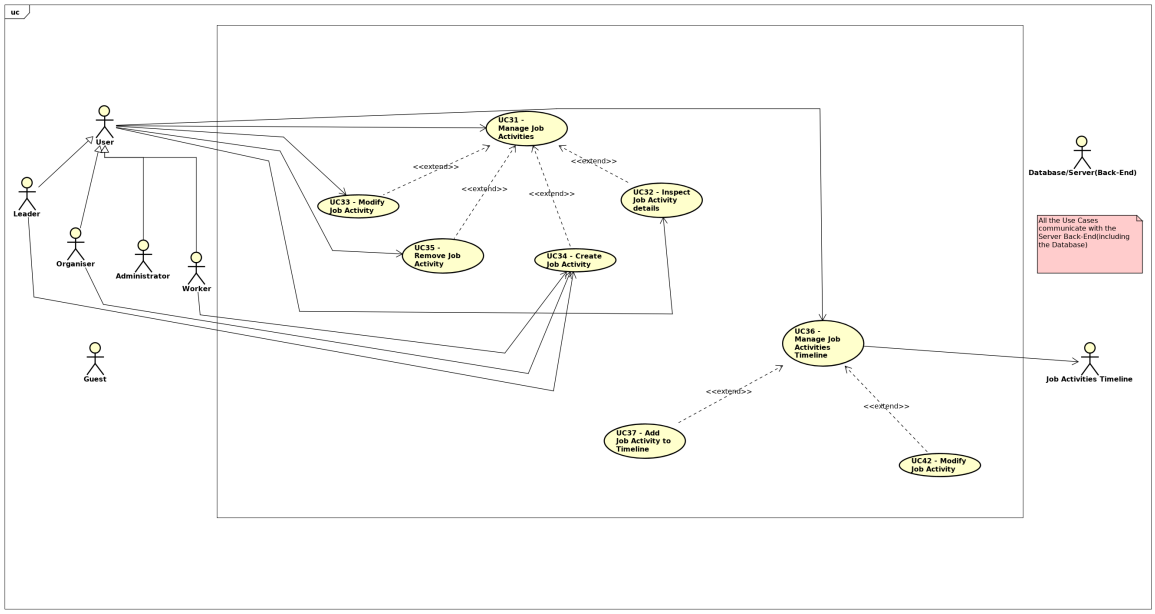

Figure 3.5: Use Case diagram - Activity

3.1.2 Sequence Diagrams

dio 1. revizije

Nacrtati sekvencijske dijagrame koji modeliraju najvanije dijelove sustava (max. 4 dijagrama). Ukoliko postoji nedoumica oko odabira, razjasniti s asistentom. Uz svaki dijagram napisati detaljni opis dijagrama.

Use Case name *Use Case description* Here goes the diagram

Use Case 31 Manage Job Activities(Worker view) *The Worker needs to organise the Job he's been assigned. He does this by creating, modifying and removing(= manipulating) Activities that compose individual Jobs.*

This Use Case depicts the management of such a list of Activities that make up a Job. These Activities can then be added to the Timeline, where their time domain will be well defined and visible - this will enable an easy visualisation of the way the Job will be done and organised.

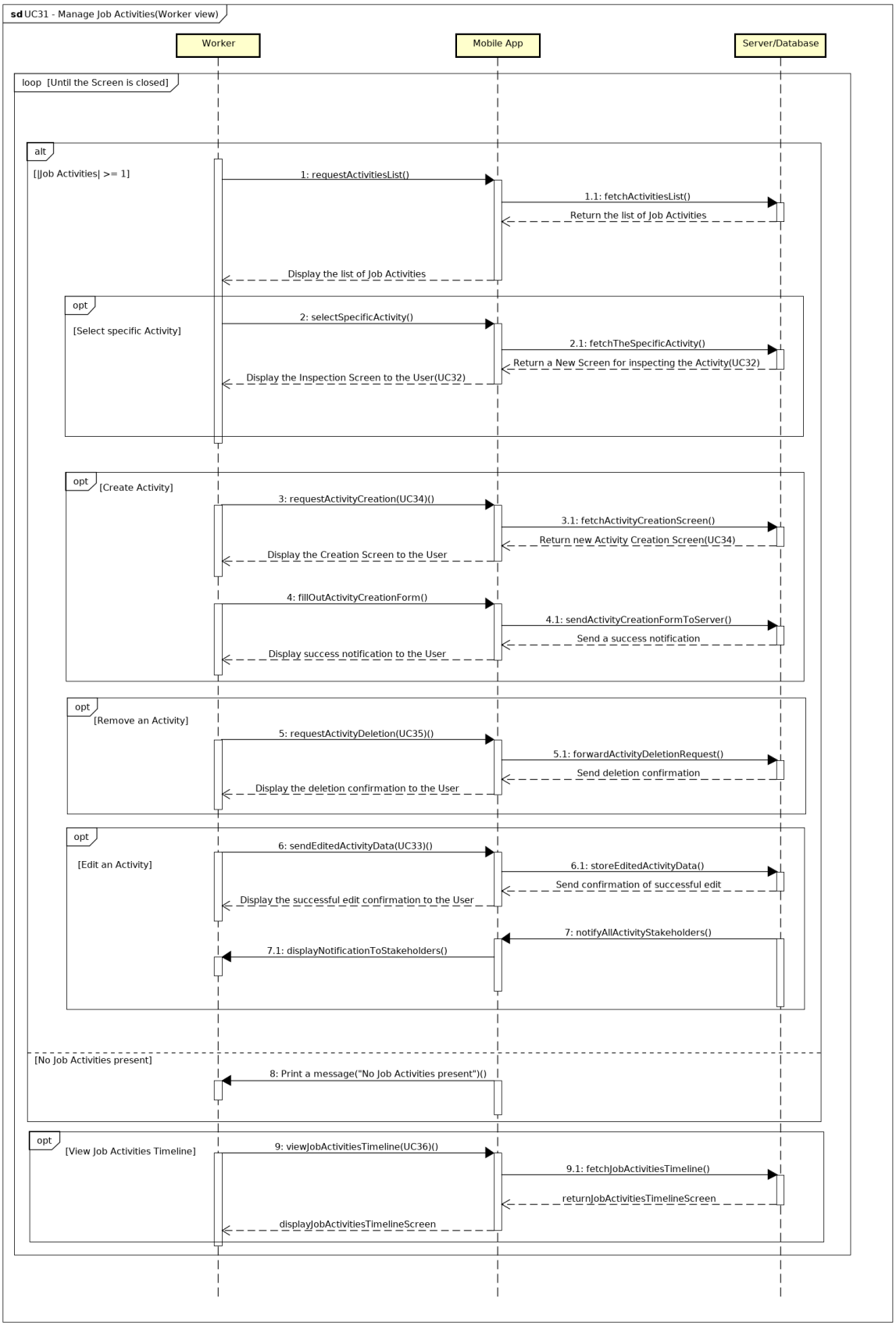

Figure 3.6: Job Activities Sequence Diagram

Use Case 26 Managing Festival Job Timeline *It is necessary for the Festival Leader or Organiser to properly manage the Festival Jobs with regard to Time organisation.*

This Use Case allows the Leader/Organiser to define the beginning and end times of certain Job Festivals, as well as modify their durations.

Timeline is a graphical flowchart - easy and intuitive Festival Job organisation. Concurrency and parallelisation can also be easily achieved.

Figure 3.7: Festival Job Timeline Sequence Diagram

Use Case 36 Manage Job Activities Timeline(Worker view) *Just as it is with Festival Leaders and Organiser and Festival Jobs Management so it is with Workers and Job Activities Management.*

Lead Workers must manage their Jobs' Activities in much the same way. They have to take into consideration the start and end times of each Activity, as well as parallelise those Activities.

The purpose of the Timeline here is, too, to provide an easy, graphical, and intuitive way of doing this Job Activities manipulation and organisation.

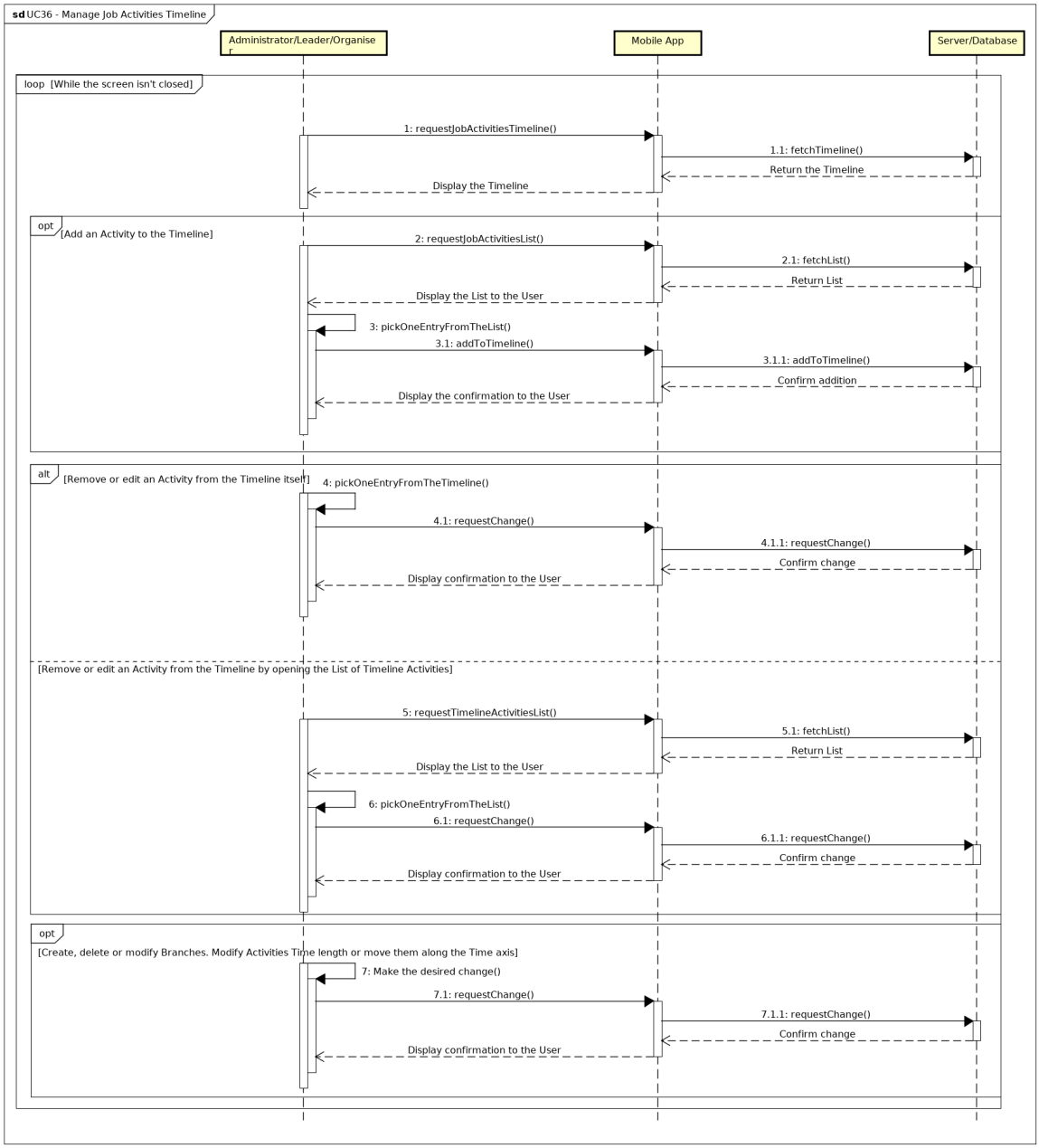

Figure 3.8: Job Activities Timeline Sequence Diagram

Use Case 21 View the List of own Festivals *Leaders and Organisers must view the List of the Festivals that they're reponsible for so they can further view the details of those Festivals, and then work on those Festivals.*

This Use Case is one of the pillars of the app, as it enables the management of the selected Festivals - Leaders and Organisers are in charge of those Festivals.

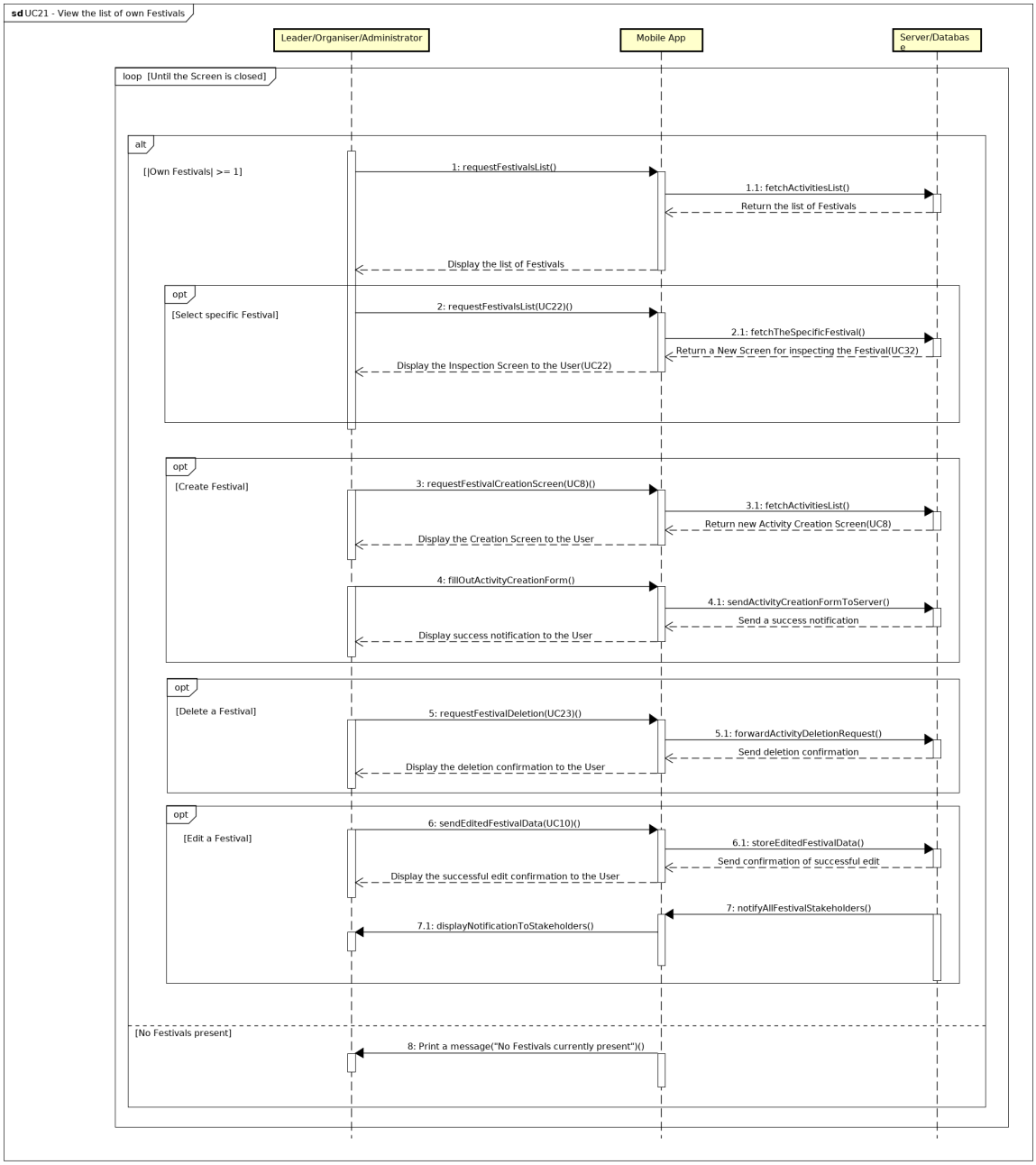

Figure 3.9: Organiser/Leader Festival List Sequence Diagram

3.2 Other requirements

- The System should allow concurrent usage by multiple Users
- The System should use UTF-8 - allow diacritical characters
- The System should use HRK and EUR as currency
- The System should be fast, reponsive and stable
- In case of some instability or error, the System should be able to recover gracefully and allow further usage
- User data should be stored safely and securely, as well as be accordingly encrypted
- The connection to the Server/Database should be reliable and secure
- The system should be easy and intuitive to use - not too complicated
- Implementation - using moder Object-Oriented Programming Languages such as Java or Kotlin
- The System should be as bug resistant as possible

4. Architecture and System Design

System architecture can be divided into three sub-systems :

- *Web server*
- *Android application*
- *Database*

Android application *It allows user to view and interact with the specific components of the graphic interface. In Android, graphical components are written in the XML (Extensible Markup Language) which is the language that can be understood by human and by the machine. On interaction with the GUI elements certain parts of the Android code are executed, mostly the ones that send HTTP requests to the Web Server. To send HTTP requests we are using Androids Volley library.*

Web Server *Web Server is the one who responds to the HTTP requests that are sent by Android application. Primary task of the web server is communication with the client (Android application) and fetching data from the database. Required data is then sent back to the client that displays data on the GUI. Response can be sent in the JSON (JavaScript Object Notation) format or as the plain String. Web server is also the one responsible for authentication and authorization of the user, server checks if the user has permission to access the certain data. If user does not have the required permission, Server will send Error response back to the client whose task is to handle the error properly.*

Programming language in which we have coded the Android application is Java, GUI components are written in the XML language and server-side code is written in Python. IDE (Integrated development environment) we are using is Android Studio for the client-side code and IDLE for the server-side code.

Design pattern we are using is MVC(Model-View-Controller). It is a common architectural pattern which is used to design and create interfaces and the structure of an application. This pattern divides the application into three parts that are dependent and connected to each other. These designs are used to distinguish the presentation of data from the way the data is accepted from the user to the data that is being shown. MVC design pattern consists of:

- *Model*
- *View*
- *Controller*

Model *The Model component corresponds to all the data-related logic that the user works with. This can represent either the data that is being transferred between the View and Controller components or any other business logic-related data. For example, a Customer object will retrieve the customer information from the database, manipulate it and update it data back to the database or use it to render data. In Android application Model is usually represented as independent Java class that contains data about e.g. Customer. That data is the one that is kept in the data base and displayed in the View component.*

View *The View component is used for all the UI logic of the application. For example, the Customer view will include all the UI components such as text boxes, dropdowns, etc. that the final user interacts with. In our application an example of View would be Activity for register, user interacts with the View, enters the necessary information that are then transferred to the Model and finally to the database via Controller.*

Controller *Controllers act as an interface between Model and View components to process all the business logic and incoming requests, manipulate data using the Model component and interact with the Views to render the final output. For example, the Customer controller will handle all the interactions and inputs from the Customer View and update the database using the Customer Model. The same controller will be used to view the Customer data.*

4.1 Database

Database used for this project is a relation based database. Relation is usually referred to as a table that has tuples. Tuple is an object that represents an information. Purpose of the database is easy and fast data manipulation, including saving, deleting, updating and sending data to the server. Database has relations:

- *User*
- *Festival*
- *Event*
- *Specialization*
- *WorkerSpec*
- *Auction*
- *Application*
- *Job*
- *JobSpec*
- *FestivalOrganizers*

4.1.1 Tables details

User *has entities for every user of the app. It has all needed personal information about the user and his role.*

User		
user_id	INT	User identification number
username	VARCHAR	Unique username
password	VARCHAR	User account password
firstname	VARCHAR	Users first name
lastname	VARCHAR	Users last name
picture	VARCHAR	Profile picture string
phone	VARCHAR	Users phone number
email	VARCHAR	Users email address
role	VARCHAR	Role user will persue in application

Festival *contains all needed information about the festival.*

Festival		
festival_id	INT	Festival identification number
creator_id	INT	ID of the user who created the festival
name	VARCHAR	Festival name
desc	VARCHAR	Short festival description, can be empty
logo	VARCHAR	Festivals logo string
duration	INTERVAL	Festival duration interval
active	BOOLEAN	True if festival is active, False if it's unactive

Event *contains information about the event of the festival.*

Event		
event_id	INT	Event identification number
festival_id	INT	ID of the festival that event belongs to
organizer_id	INT	ID of the user who created the event
name	VARCHAR	Event name
desc	VARCHAR	Short event description, can be empty
location	VARCHAR	Events location
start_Time	TIMESTAMP	Event start time

end_Time	TIMESTAMP	Event end time

Specialization *contains all different specializations and their names.*

Specialization		
specialization_id	INT	Specializations identification number
name	VARCHAR	Name of the specialization

WorkerSpec *contains specifications about the user who wants to apply as a worker and his specializations.*

WorkerSpec		
worker_id	INT	Workers identification number
specialization_id	INT	Specializations identification number

Auction *contains information about auctions.*

Auction		
auction_id	INT	Auction identification number
start_Time	TIMESTAMP	Auction start time
end_Time	TIMESTAMP	Auction end time

Application *contains information about applications for auctions.*

Application		
application_id	INT	Application identification number
auction_id	INT	Auction identification number that worker applies for
worker_id	INT	Workers identification number
price	FLOAT	Offered pay for the job
comment	VARCHAR	Additional comment for application, can be empty
approximate_time	INT	Time needed to complete the job, in days
number_of_people	INT	Number of people that will be doing the job

Job *contains information about jobs that had an auction.*

Job		
job_id	INT	Job identification number
event_id	INT	Events identification number that job is for
worker_id	INT	Workers identification number that does the job
auction_id	INT	Auctions identification number that auctioned the job
start_Time	DATETIME	Jobs start time
is_Completed	BOOLEAN	True if job is finished, false if it's still active

JobSpec *contains specializations that are needed for the job.*

JobSpec		
job_id	INT	Job identification number
specialization_id	INT	Specializations identification number

FestivalOrganizers *contains information which user, that is also an organizer, applied for which festival.*

FestivalOrganizers		
festival_id	INT	Festival identification number
organizer_id	INT	Organizers identification number
status	INT	1 when organizer is waiting on leader, 0 when rejected and 1 when accepted

4.1.2 Database diagrams

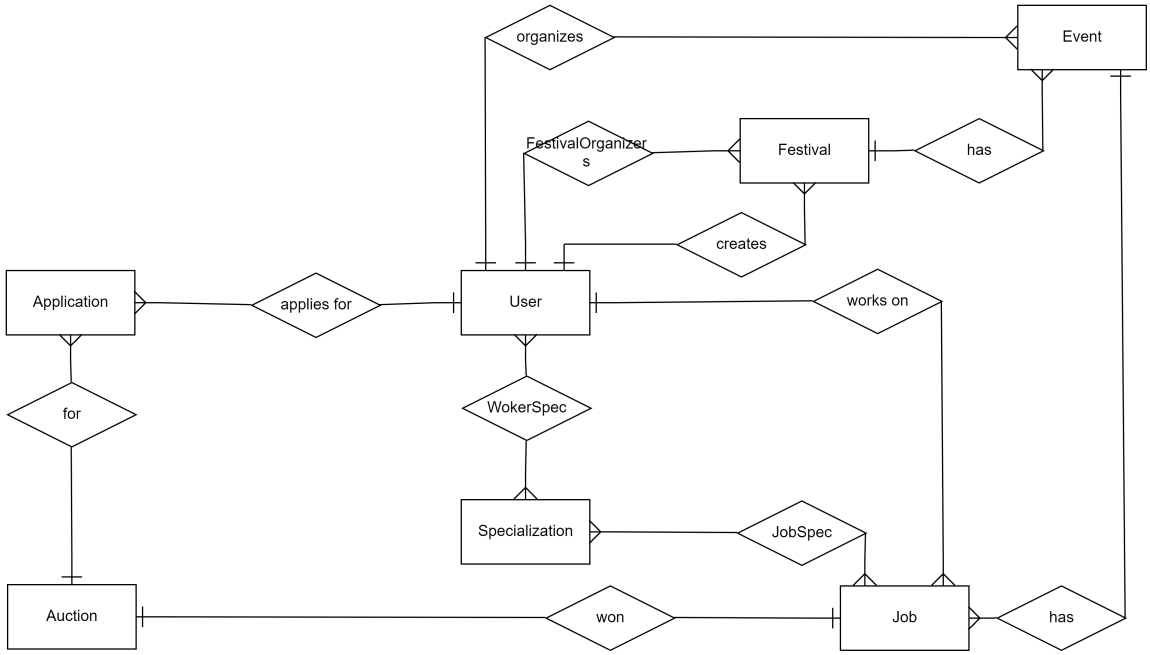

Figure 4.1: E-R Diagram

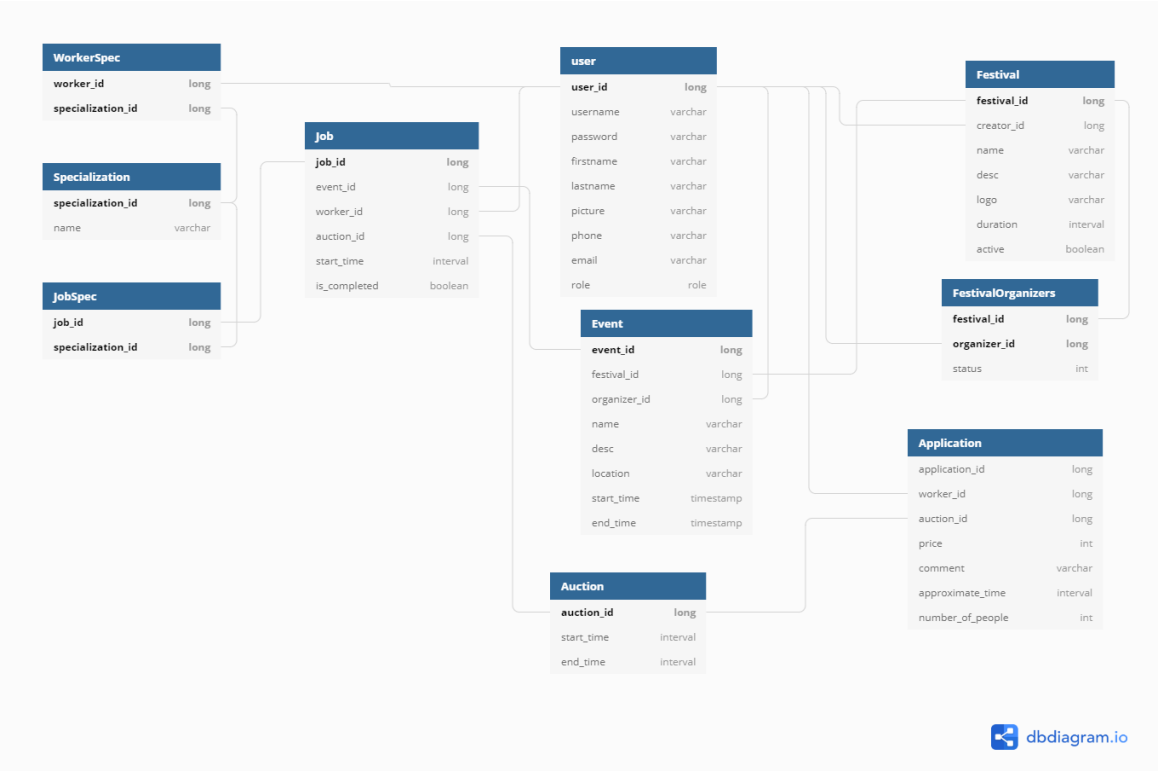

Figure 4.2: Database Diagram

4.2 Class Diagrams

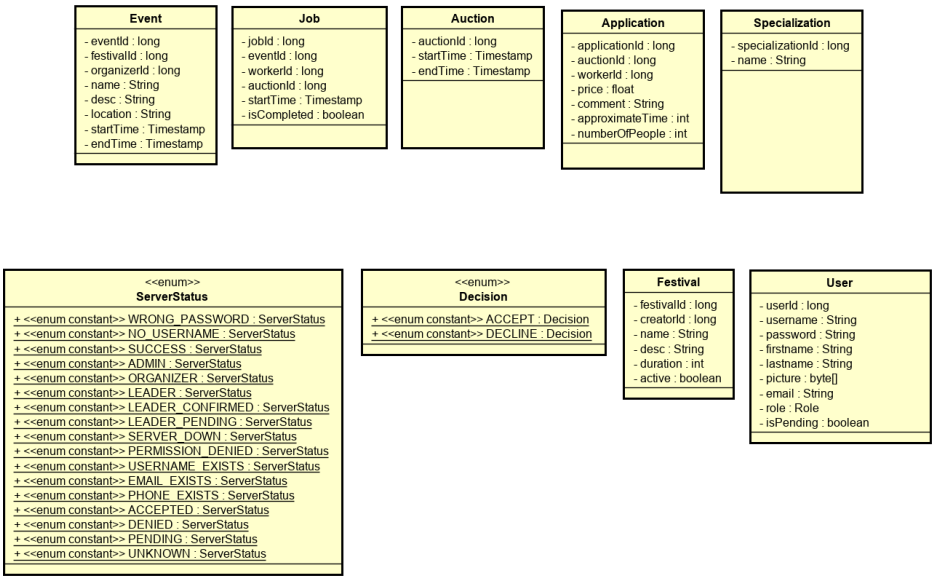

Figure 4.3: UML Model Class Diagram

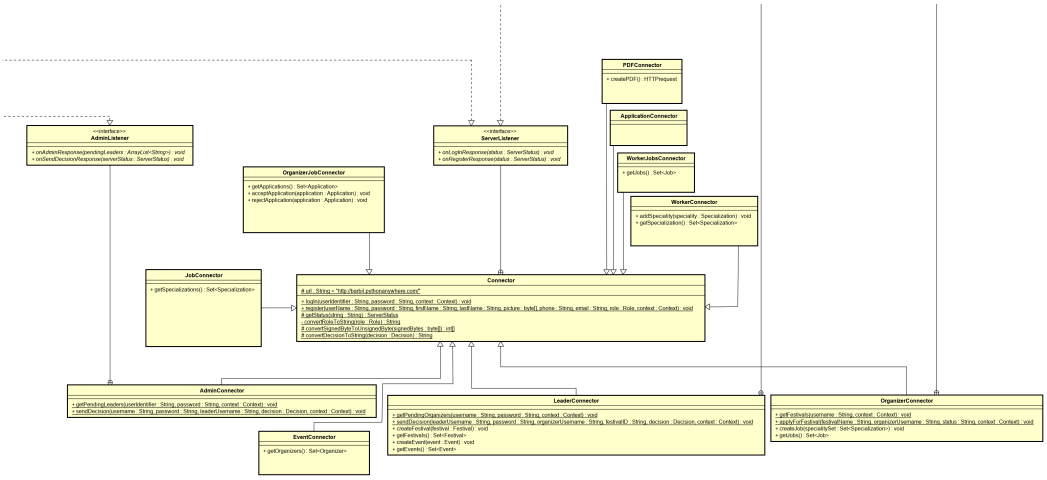

Figure 4.4: UML Connector Diagram

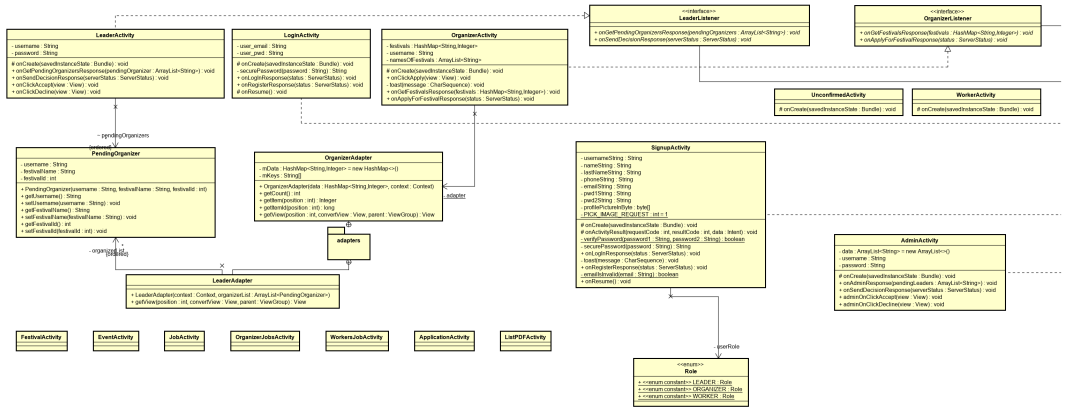

Figure 4.5: UML Activity Diagram

dio 2. revizije

Prilikom druge predaje projekta dijagram razreda i opisi moraju odgovarati stvarnom stanju implementacije

4.3 Dijagram stanja

dio 2. revizije

Potrebno je priloiti dijagram stanja i opisati ga. Dovoljan je jedan dijagram stanja koji prikazuje ***znaajan dio funkcionalnosti*** *sustava. Na primjer, stanja korisnikog suelja i tijek koritenja neke kljune funkcionalnosti jesu znaajan dio sustava, a registracija i prijava nisu.*

4.4 Dijagram aktivnosti

dio 2. revizije

Potrebno je priloiti dijagram aktivnosti s pripadajuim opisom. Dijagram aktivnosti treba prikazivati znaajan dio sustava.

4.5 Dijagram komponenti

dio 2. revizije

Potrebno je priloiti dijagram komponenti s pripadajuim opisom. Dijagram komponenti treba prikazivati strukturu cijele aplikacije.

5. Implementation and User Interface

5.1 Koritene tehnologije i alati

dio 2. revizije

Detaljno navesti sve tehnologije i alate koji su primijenjeni pri izradi dokumentacije i aplikacije. Ukratko ih opisati, te navesti njihovo znaenje i mjesto primjene. Za svaki navedeni alat i tehnologiju je potrebno ***navesti internet poveznicu*** *gdje se mogu preuzeti ili vie saznati o njima.*

5.2 Ispitivanje programskog rjeenja

dio 2. revizije

U ovom poglavlju je potrebno opisati provedbu ispitivanja implementiranih funkcionalnosti na razini komponenti i na razini cijelog sustava s prikazom odabranih ispitnih sluajeva. Studenti trebaju ispitati temeljnu funkcionalnost i rubne uvjete.

5.2.1 Ispitivanje komponenti

Potrebno je provesti ispitivanje jedinica (engl. unit testing) nad razredima koji implementiraju temeljne funkcionalnosti. Razraditi ***minimalno 6 ispitnih sluajeva*** *u kojima e se ispitati redovni sluajevi, rubni uvjeti te izazivanje pogreke (engl. exception throwing). Poeljno je stvoriti i ispitni sluaj koji koristi funkcionalnosti koje nisu implementirane. Potrebno je priloiti izvorni kd svih ispitnih sluajeva te prikaz rezultata izvoenja ispita u razvojnom okruenju (prolaz/pad ispita).*

5.2.2 Ispitivanje sustava

Potrebno je provesti i opisati ispitivanje sustava koristei radni okvir Selenium[1]*. Razraditi* ***minimalno 4 ispitna sluaja*** *u kojima e se ispitati redovni sluajevi, rubni uvjeti te poziv funkcionalnosti koja nije implementirana/izaziva pogreku kako bi se vidjelo na koji nain sustav reagira kada neto nije u potpunosti ostvareno. Ispitni sluaj se treba sastojati od ulaza (npr. korisniko ime i lozinka), oekivanog izlaza ili rezultata, koraka ispitivanja i dobivenog izlaza ili rezultata.*

Izradu ispitnih sluajeva pomou radnog okvira Selenium mogue je provesti pomou jednog od sljedea dva alata:

- *dodatak za preglednik* ***Selenium IDE*** *- snimanje korisnikovih akcija radi automatskog ponavljanja ispita*
- ***Selenium WebDriver*** *- podrka za pisanje ispita u jezicima Java, C#, PHP koristei posebno programsko suelje.*

Detalji o koritenju alata Selenium bit e prikazani na posebnom predavanju tijekom semestra.

[1] `https://www.seleniumhq.org/`

5.3 Dijagram razmjetaja

dio 2. revizije

*Potrebno je umetnuti **specifikacijski** dijagram razmjetaja i opisati ga. Mogue je umjesto specifikacijskog dijagrama razmjetaja umetnuti dijagram razmjetaja instanci, pod uvjetom da taj dijagram bolje opisuje neki vaniji dio sustava.*

5.4 Upute za putanje u pogon

dio 2. revizije

U ovom poglavlju potrebno je dati upute za putanje u pogon (engl. deployment) ostvarene aplikacije. Na primjer, za web aplikacije, opisati postupak kojim se od izvornog kda dolazi do potpuno postavljene baze podataka i posluitelja koji odgovara na upite korisnika. Za mobilnu aplikaciju, postupak kojim se aplikacija izgradi, te postavi na neku od trgovina. Za stolnu (engl. desktop) aplikaciju, postupak kojim se aplikacija instalira na raunalo. Ukoliko mobilne i stolne aplikacije komuniciraju s posluiteljem i/ili bazom podataka, opisati i postupak njihovog postavljanja. Pri izradi uputa preporuuje se ***naglasiti korake instalacije uporabom natuknica*** *te koristiti to je vie mogue* ***slike ekrana*** *(engl. screenshots) kako bi upute bile jasne i jednostavne za slijediti.*

Dovrenu aplikaciju potrebno je pokrenuti na javno dostupnom posluitelju. Studentima se preporua koritenje neke od sljedeih besplatnih usluga: Amazon AWS, Microsoft Azure ili Heroku. Mobilne aplikacije trebaju biti objavljene na F-Droid, Google Play ili Amazon App trgovini.

6. Conclusion and Outline of Planned Future Work

dio 2. revizije

U ovom poglavlju potrebno je napisati osvrt na vrijeme izrade projektnog zadatka, koji su tehniki izazovi prepoznati, jesu li rijeeni ili kako bi mogli biti rijeeni, koja su znanja steena pri izradi projekta, koja bi znanja bila posebno potrebna za bre i kvalitetnije ostvarenje projekta i koje bi bile perspektive za nastavak rada u projektnoj grupi.

Potrebno je tono popisati funkcionalnosti koje nisu implementirane u ostvarenoj aplikaciji.

Literature

Kontinuirano osvjeavanje
Popisati sve reference i literaturu koja je pomogla pri ostvarivanju projekta.

1. Oblikovanje programske potpore, FER ZEMRIS, http://www.fer.hr/predmet/opp
2. I. Sommerville, "Software engineering", 8th ed, Addison Wesley, 2007.
3. T.C.Lethbridge, R.Langaniere, "Object-Oriented Software Engineering", 2nd ed. McGraw-Hill, 2005.
4. I. Marsic, Software engineering book, Department of Electrical and Computer Engineering, Rutgers University, http://www.ece.rutgers.edu/~marsic/books/SE
5. The Unified Modeling Language, https://www.uml-diagrams.org/
6. Android documents, https://developer.android.com/docs
7. Head First Android Development , A Brain Friendly Guide, 2nd Edition
8. Astah Community, http://astah.net/editions/uml-new
9. https://stackoverflow.com
10. Framework introduction, https://www.tutorialspoint.com/mvc_framework/mvc_framework_introduction.htm
11. Flask documentation, http://flask.palletsprojects.com/en/1.1.x
12. Sqlalchemy, https://docs.sqlalchemy.org/en/13

Image and diagram index

Appendix: Preview of group activity

Meeting log

Continuous updating

1. Meeting
   - Date: October 4, 2019
   - Attended: P.Fribert, D.R.Sparemblek, B.Bilic, K.Ceple, D.Strbad, L.Lendel, K.Jezic
   - Meeting theme:
     - initial getting to know each other, individual skills
2. Meeting
   - Date: October 5, 2019
   - Attended: P.Fribert, D.R.Sparemblek, B.Bilic, K.Ceple, D.Strbad, L.Lendel, K.Jezic
   - Meeting theme:
     - creating back end, front end and server team
3. Meeting
   - Date: October 15, 2019
   - Attended:D.R.Sparemblek, B.Bilic
   - Meeting theme:
     - making first draft database
4. Meeting
   - Date: October 23, 2019
   - Attended:D.R.Sparemblek, B.Bilic, P.Fribert
   - Meeting theme:
     - making database in python
5. Meeting
   - Date: October 24, 2019
   - Attended:L.Lendel, B.Bilic, K.Jezic, D.R.paremblek, P.Fribert

- Meeting theme:
  - signup and login screen functions

6. Meeting
   - Date: November 6, 2019
   - Attended: P.Fribert, D.R.Sparemblek
   - Meeting theme:
     - foreign keys in database
7. Meeting
   - Date: November 7, 2019
   - Attended: P.Fribert, B.Bilic, K.Ceple, D.Strbad, L.Lendel, K.Jezic
   - Meeting theme:
     - server functions
     - front end: hashing passwords and checking emails
     - documentation update
8. Meeting
   - Date: November 10, 2019
   - Attended: D.R.Sparemblek, B.Bilic, L.Lendel, K.Jezic
   - Meeting theme:
     - debugging application, checking code
9. Meeting
   - Date: November 11, 2019
   - Attended: P.Fribert, D.R.Sparemblek, B.Bilic, K.Ceple, D.Strbad, L.Lendel, K.Jezic
   - Meeting theme:
     - final changes before commiting project
10. Meeting
    - Date: November 13, 2019
    - Attended: P.Fribert, K.Ceple
    - Meeting theme:
      - documentation specifics
11. Meeting
    - Date: November 15, 2019
    - Attended: B. Bilic, D.R. Sparemblek, D. Strbad and K. Ceple
    - Meeting theme:

- Create and add Class Diagrams to the Documentation
- Check the Documentation for slip-ups, fix and add some items to it
- Merge devdoc to master
- Generate final document

Activity table

Continuously refreshed

	Bartol Bilic	**Daniel Rey Sparemblek**	**Karlo Jezic**	**Danijel Strbad**	**Luka Lendel**	**Kristijan Ceple**	**Petra Fribert**
Upravljanje projektom	5	5					5
Opis projektnog zadatka						6	
Funkcionalni zahtjevi						5	
Opis pojedinih obrazaca						13	
Dijagram obrazaca						5	
Sekvencijski dijagrami						5	
Opis ostalih zahtjeva						1	
Arhitektura i dizajn sustava							6
Baza podataka							9
Dijagram razreda				5			
Dijagram stanja							
Dijagram aktivnosti							
Dijagram komponenti							
Koritene tehnologije i alati							
Ispitivanje programskog rjeenja							
Dijagram razmjetaja							
Upute za putanje u pogon							
Dnevnik sastajanja							5
Zakljuak i budui rad							
Popis literature						4	4
front end layout	1			9	12		
front end logic	7		13	8	9		5

	Bartol Bilic	**Daniel Rey Sparemblek**	**Karlo Jezic**	**Danijel Strbad**	**Luka Lendel**	**Kristijan Ceple**	**Petra Fribert**
izrada baze podataka	11	14					8
spajanje s bazom podataka	8	8	8				
back end	23	6					
razrada arhitekture	5	4					

Dijagrami pregleda promjena

dio 2. revizije

Prenijeti dijagram pregleda promjena nad datotekama projekta. Potrebno je na kraju projekta generirane grafove s gitlaba prenijeti u ovo poglavlje dokumentacije. Dijagrami za vlastiti projekt se mogu preuzeti s gitlab.com stranice, u izborniku Repository, pritiskom na stavku Contributors.