# Annex 1: Software Project Plan

In following annex, the organizational aspects of the chatbot’s development will be examined. More precisely, the software development process is described as well as the tools that were used to manage the process, followed by a detailed examination of the course of the project.

The second part of the annex examines the project’s viability, including the calculation of involved costs and possibilities for making profit.

# 2. Project Management

## 2.1 Scrum

The project’s management is inspired by *Scrum1*, an agile software development framework which facilitates managing tasks in teams. The Scrum framework is based on the experience that most projects are too complex to be planned completely in the early stages and therefore provides an agile and iterative alternative by structuring the project in smaller iterations.

However, the applied process during this project is only loosely based on the Scrum framework since there are several concepts of the original framework that were not applied. Scrum defines roles which describes the different members of a group (*Scrum Master*, *Project Owner* and *Development Team*) as well as several artefacts organizing the team’s interaction. Due to the nature of the project not having a development team in the proper meaning of the word, but only a single person realizing the bachelor thesis, only some artefacts were applied. The most important applied concept is the *Sprint*, a two-week time slot which is used to structure the project. At the beginning of each sprint, a *Sprint Planning* is realized between the author and the coordinator of project. The project’s coordinator can be seen as a *Product Owner* in the Scrum framework, prioritizing tasks and guiding the project’s direction.In the *Sprint Planning*, it is decided which tasks are going to be worked at during the two-week sprint. At the end of each sprint, a *Sprint Review* takes place where the development team, meaning the student, presents its results to the *Product Owner*.

## 2.2 Managing the project in GitHub

In order to facilitate the project’s management, the online project hosting tool *GitHub* with *ZenHub* as an extension was used. *ZenHub* provides several useful collaboration features such as a board to visualize tasks as well as an overview of the remaining workload and velocity. The tasks are defined in GitHub’s *Issues* where tasks can be named, described and estimated. Story Points are used to estimate the workload of each issue. In this project, one story point is seen as the equivalent of 2 – 3 hours of work. The iterations are planned and defined using *Milestones* which the different issues are assigned to.

## 2.3 Time Planning

The Kick-Off Meeting took place at 5th December 2016, where the elemental concepts of the project were discussed. Due to the fact that this project was held in the course of the semester parallel to usual classes and exam periods, there were some time breaks between the two week sprints in which the development on the project paused.The project consists of 11 iterations, the last one ending a day before the submission date of the thesis, the 2th July, 2017.

The following section gives an overview of the iterations during this project, finally using Burn-Down-Charts and velocity tracking to visualize the project’s progress.

### 2.3.1 Iteration #1 (30/12/2016 – 17/01/2017): Planning and Research

The first iteration mainly centered on setting up the project's infrastructure, including the establishment of the repository structure and the virtual machine set up. First research steps were taken by getting familiar with the used geo-information system.

### 2.3.2 Iteration #2 (17/01/2017 – 07/02/2017): Further Research

In this iteration, the knowledge of the used geo-information system was deepened by investigating how to extract essential tourist information from the database. On the other hand, a chatbot library was chosen after comparing different possible candidates. Besides, the project's documentation will be extended by describing theoretical concepts, used tools and the project's objectives.

### 2.3.3 Iteration #3 (07/02/2017 – 21/02/2017): Recommender System and Design

In this iteration, recommender system libraries were examined to find out which one was the most suitable for this project. Afterwards, the system's basic architecture was designed. On top of that, the project's documentation was extended.

### 2.3.4 Iteration #4 (21/02/2017 – 07/03/2017): Mockups

First mockups were implemented in which the design of certain components of the application were tested. At first, the interaction of the application's chatbot layer with its environment was set up, using the Telegram Bot API and the API.AI HTTP API.

The recommender component was examined in more depth, meaning that a recommender library was finally chosen and then used to mock up a POI recommender mechanism.

### 2.3.5 Iteration #5 (10/03/2017 – 24/03/2017): Recommender System Mockup

This iteration centered on the design and first implementation of the application's recommender system. The library Apache *Mahout* is used to implement a content-based filtering recommendation mechanism. The iteration includes the design of user and item profiles as well as the investigation of how to retrieve the data from *OpenStreetMap* into the recommender system.

### 2.3.6 Iteration #6 (03/04/2017 – 17/04/2017): Collaborative Filtering & Requirements

In this iteration, the collaborative filtering recommender was designed. Additionally, the project's requirements were examined as well as user-chatbot interaction.

### 2.3.7 Iteration #7(18/04/2017 – 02/05/2017) Conversational Interface Design

In this iteration, the chatbot's conversational interface was designed by modeling a conversation flow graph. The resulting knowledge was explained in the Theoretical Concepts chapter of the documentation. On the other hand, user ratings were generated for the collaborative filtering algorithm.

### 2.3.8 Iteration #8 (04/05/2017 -18/05/2017) Conversation Flow Implementation

The previously specified conversation flow was modeled in API.AI and the application's chatbot layer was implemented to handle the conversation. On the other hand, the recommender component was finished and documented.

### 2.3.9 Iteration #9 (18/05/2017 – 01/06/2017) Conversation Flow Refinement and Testing

The conversation flow was refined to improve the conversation between user and chatbot. The rating mechanism was implemented. Some performance tests for the recommendation mechanism were implemented and the software was evaluated using metrical code analysis tools.

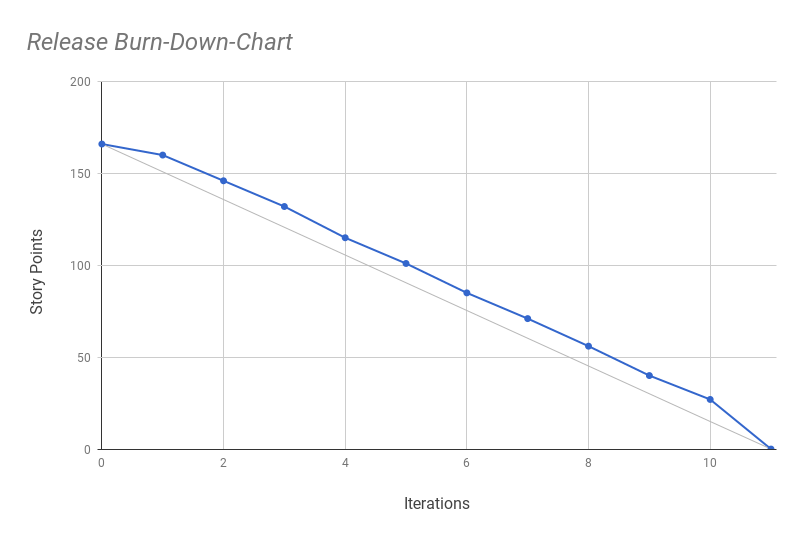
### 2.3.10 Iteration #10 (03/06/2017 – 17/06/2017) Latex Setup

This iteration dealt with setting up the latex file for the project’s documentation. Previously prepared drafts were modified and completed. Several chapters and parts of the annex were introduced into the latex document and the bibliography was formalized.

### 2.3.11 Iteration #11 (18/06/2017 – 02/07/2017) Documentation and Final Conversation Flow Adjustments

This iteration concentrated on completing the project's documentation, especially outlining the project's relevant aspects and come to final conclusions. The application's implementation was only adjusted in refining the conversation flow and small refactorings in order to improve the code quality.

### 2.3.12 Burn-Down Chart and Velocity Tracking



The Burn-Down-Chart shows the progression of story points in the development process. The ideal progression is represented in the grey straight line. As we can see, the actual story point progression does not differ largely from the shown ideal line, as the work load was divided equally to the different iterations.

# chart

The workload divided to the different iterations is shown in detail in the velocity tracking diagram. The average work load per iteration is measured with 15 story points, which can be seen in the red trend line. Most iterations do not differ largely from this average measure, except for the first and the last iteration. This observation can be explained as in the first iteration, research was done and a first overview of all the different aspects of the project was made. After this iteration, the necessary tasks to be done became clearer and a first task overview was created.

In contrast, the last iteration is outstandingly bigger than the previous ones. This can be explained by the fact that the student had terminated all of the exams and semester’s course work at that time and could completely focus on the preparation of the project.

## Viability Study

In this section of the paper, a viability study is made. It is examined whether the project is economically feasible, comparing the involved costs and potential sources of income. Additionally, the legal viability is checked to ensure that a realization of the application can be done without legal risks.

### Economic Viability

**Personnel Costs**

The application’s development was realized by the author of the presented bachelor thesis. As the author is a student of computer sciences without completed university degree, the hourly wage is estimated at 14€/hour. In order to calculate the total development costs, the time invested in the project is examined. As the realization of the bachelor thesis is evaluated with 12 ECTS points and a workload of 30 hours per ECTS point is assumed, a total amount of 360 hours were dedicated to the project’s development. This also coincides with the results of the velocity tracking, where a total number 166 story points was estimated to finish the project, one story point being roughly equivalent to 2-3 hours of working.

All in all, this leads to the following calculation:

**360 hours \* 14 €/hour = 5040 €**

As a German citizen, German tax law is applied in the following scenario: Assuming that the produced application is developed in a freelance project on behalf of a customer, the developer only charges the hourly fee. There are no additional social security or tax costs for the customer as freelance developers are liable to pay income taxes themselves. However, the development fees are below the tax allowance for unmarried persons in 2017, which is **8820 €. Therefore no income taxes have to be paid, assuming that the student does not have any other freelance income in the respective year.**

**Software Costs**

In the following, the costs of the services used in the project are examined.

|  |  |  |
| --- | --- | --- |
| **Software** | **Costs** | **Comments** |
| OpenStreetMap | 0,00 € | - |
| API.AI | 0,00 € | According to the API.AI terms of use, “services include basic services (“Basic Services”) provided free of charge and enhanced services (“Enhanced Services”), which, if available, must be purchased.” [x]  However, at the time of development, no fees or potential upgrades were evident. |
| Foursquare API | 0,00 € | According to Foursquare’s developer documentation, “the Foursquare API has a default limit of 1000 free requests per 24 hour period” and “500 requests/hour”, “whichever occurs first”. An enterprise option must be booked if more requests are needed. |
| Telegram API | 0,00€ | - |
| Apache Mahout | 0,00 € | - |
| Apache Maven |  |  |
| Heroku | 0,00 € | A “Free Plan” was used to host the application in Heroku. There are several pricing options, adjusted to the developer’s needs. The free plan comes with certain inconveniences, such as “sleep after 30 minutes of inactivity”. [h] |

As we can see, no software costs were charged in the way the third-party products are used in this project. However, there could be additional costs with an increasing number of users accessing the chatbot, as some services that are used apply a pricing model (Foursquare and Heroku).

**Hardware Costs**

Two different hardware devices were used during the project duration. For the software’s development, a laptop was used, in particular an Asus Zenbook, bought in 2015 for 899 €. In order to use the application and check the functionality in the productive environment, a mobile device is used, more precisely the Motorola Moto G4 Plus, bought in 2017 for 199 €. Due to the fact that the mentioned devices were not exclusively purchased for this project, their costs can be amortized as seen in the following calculation. A total amortization period of 3 years for the hardware is assumed:

Total Hardware Cost = 899 € + 199 € = 1098 €

Amortization period in hours = 3 years \* 8760 hours/year = 26280 hours

Development period in hours = 360 hours

Hardware Cost per hour = 1098 € / 26280 hours = 0.04 €/h

Hardware Costs during development = 0.04€/h\* 360 h= **14.4 €**

**Total Costs**

The previously calculated partial costs result are summarized in the following table:

|  |  |
| --- | --- |
| Personnel Costs | 5040,00 € |
| Software Costs\* | 0,00 € |
| Hardware Costs | 14.40 € |
| **Total Costs** | **5054,40 €** |

It must be noted that occurring office costs were not taken into account as it is assumed that the developer has worked in a home office which can be set off against tax liability.

Profit Analysis

The following section provides an understanding of possible ways to make profit from the developed application.

Due to the fact that the chatbot is publicly accessible in an instant-messenger, it is not very common to profit from the user directly, as there are no easy-to-use payment mechanisms integrated in the messenger.

A more interesting concept is to win tourist agencies, cultural institutions or restaurant owners as investors. The chatbot can be seen as an advertisement platform for those who want to promote their offers to tourists which in fact are a very promising target group as people tend to spend a lot of money on vacation.

Essentially, the idea is to present points of interest that belong to a customer in a more prominent, attractive way to the user. This could happen, for example, when a user asks for a recommendation and is situated in close proximity to a promoted point of interest. The presentation of the promoted point of interest could differ from the usual, non-promoted locations, in a way that more precise descriptions or more photos of the location are provided than usual. Additionally, the points of interests could be highlighted visually, for example by applying different font sizes or colors for the messages.

In return for this service, a monthly payment of the local investors would be conceivable.

This way of promoting points of interests is quite interesting for the local tourist industry as the chatbot collects user interests for the recommendation, therefore, the marketers are provided with a direct way to target exactly the user group they are interested in.

However, it must be noted that the commercial use of the application leads to a possible upgrade of pricing plans, for instance using the Foursquare API.

Legal Viability

The following legal viability concentrates on the third-party software licenses used in the project and in turn, an analysis of which license can be applied for the developed application:

|  |  |  |
| --- | --- | --- |
| Software | License | Comments |
| OpenStreetMap | Open Database License (ODbL) 1.0 | According to the ODbL license, the rights to use the OpenStreetMap data “explicitly include explicitly include commercial use, and do not exclude any field of endeavour.” |
| API.AI | API.AI END USER LICENSE AGREEMENT | According to API.AI’s license agreement, the “use of the Services within the commercial enterprise for internal purposes is expressly allowed.” [a] |
| Foursquare | FOURSQUARE LABS, INC. API AND DATA LICENSE AGREEMENT | For commercial use, an enterprise license is needed. The fees that are charged are not publicly available and are communicated directly by contacting Foursquare. |
| Telegram API | GNU General Public License v2.0 | According to Telegram, the commercial use of the Telegram API is permitted for anyone except for “large corporations, publicly traded companies and other businesses that exist to maximize shareholder value or sell stock.” [T] |
| Apache Mahout | [Apache License, Version 2.0](http://www.apache.org/licenses/LICENSE-2.0). | Permissive free software license that allows, among other things, sublicensing, commercial use, modifying and distributing of the software. |
| Apache Maven | [Apache License, Version 2.0](http://www.apache.org/licenses/LICENSE-2.0). | See above |
| Spark Framwork | [Apache License, Version 2.0](http://www.apache.org/licenses/LICENSE-2.0). | See above |
| Heroku | Heroku License Agreement | the granted license is not sublicenseable |

As some of the used third-party software is nonsublicenseable, for instance API.AI, it is not possible to apply a permissive free software license such as Apache License 2.0 or GNU General Public License v2.0 to the application. However, if the designed application aims to make profit, publishing the application under a free software license is not an option anyway. Therefore suitable terms of service need to be designed in order to publish the application.

# Bibliography

1: The Scrum GuideTM­, Ken Schwaber and Jeff Sutherland

[x]: api.ai terms of use <https://api.ai/terms/>

F: <https://developer.foursquare.com/overview/ratelimits>

H: <https://www.heroku.com/pricing>

[O]: <https://opendatacommons.org/licenses/odbl/1.0/>

HS: https://www.heroku.com/policy/tos