ASSIGNMENT COVERSHEET

Prague College

Student Name: Samuel A. Marti	
Class: Project Design Implementation and Evaluation TBL – Part2	
Assignment: Introduction to Java	
Lecturer: Bohus Ziskal	Semester: 1803
Due Date: 06.09.2018	Actual Submission Date: 06.09.2018

Evidence Produced (List separate items)	**Location (Choose one)**	
HumanaDBDocumentation.pdf	X	Uploaded to the Learning Center (Moodle)
		Submitted to reception
Note: Email submissions to the lecturer are not valid.		

Student Declaration	
I declare that the work contained in this assignment was researched and prepared by me, except where acknowledgement of sources is made. I understand that the college can and will test any work submitted by me for plagiarism. **Note:** The attachment of this statement on any electronically submitted assignments will be deemed to have the same authority as a signed statement	
Date: 05.09.2018	Student Signature: [illegible]

A separate feedback sheet will be returned to you after your work has been graded.
Refer to your Student Manual for the Appeals Procedure if you have concerns about the grading decision.

Student Comment (Optional)
Was the task clear? If not, how could it be improved?
Was there sufficient time to complete the task? If not, how much time should be allowed?
Did you need additional assistance with the assignment?
Was the lecturer able to help you?
Were there sufficient resources available?
How could the assignment be improved?
For further comments, please use the reverse of this page.

H³UMANA Database

Project Documentation

Author:	Samuel A. Marti
Place:	Prague College
Class:	Project Design Implementation and Evaluation TBL – Part2
Lecturer:	Bohus Ziskal
Assignment:	Project Specification and Planning
Date:	05.09.2018
Semester Code:	1803
Word Count:	10770

Contents

1 Introduction

This Document aims to be a concise project report of the assignment: *"Project Specification and Planning"*. It was composed for the Prague college course *"Project Design Implementation and Evaluation"* which was taught by lecturer Bohus Ziskal in 2018.

1.1 Glossary & Abbreviations

This glossary defines distinct terms used within this document.

Term	**Definition**
MVP	Minimal Viable Product. ([5] Agile Alliance, 2018)
Story	A synonym defined within this project for the term requirement. ([9] Margaret Rouse, -)
Product Backlog	Stack of stories sorted according to their priorities for the project. ([10] Robin Hackshall, 2014)
Product Owner	The person or entity which is maintaining the product backlog. ([11] Product Owner , 2018)
SP	Story Point ([12] Margaret Rouse, 2015)
MVC	Model View Controller ([13] Codecademy, 2018)
GUI	Graphical User Interface ([14], 2018)

Table 1 Glossary & Abbreviations

2 Project Introduction

This chapter will introduce the project and its background based on ([1] Samuel A. Marti, 2018).

2.1 Status Quo

In the Republic of Ecuador it is common practice for medical facilities to collect and maintain their client data on paper based documents. Such paper based system inflects several drawbacks in certain situations such as: Sharing of data, duplication of data, statistical investigation, loss of data and records of changes.

Bigger medical facilities such as hospital which are mostly located within cities, already started to implement digital client management systems. However smaller clinics or medical practices which are often located in rather rural areas still rely mostly on paper based documentation. Mainly due following reasons:

- Inflicted costs by implementing such a system.
- Missing network infrastructure.
- Missing awareness and expertise offer of such systems.

2.1.1 Sources

The stated status quo is based on following sources:

- A report carried out by a student of the university of Ambato stating facts in regards of medical database in regards of Ambato and diabetes. ([4] Pico, 2013)
- A Interview executed by the beginning of this year, by a H^{3}UMANA member and Dr. Jorge Isaac Sanchez Mino. Stating the personal observation and needs in regards of a medical database. ([3] Mino, 2018)
- The public ministry of Ecuador reporting on a project in order to increase the infrastructure of public health systems. ([2] Cordova, Ing. Miguel Antonio, 2018)

The cited sources shall be viewed critically since they do not base on statistical data or empirical evidence concerning the whole nation. Further research did not lead to any source of such information thus it can be assumed that such data has yet not been collected on a national level.

2.2 Owner

The project was requested by H^{3}UMANA.
H^{3}UMANA is an independent group of engineers, who's aim it is to disrupt the health industry and modernize it from within. The group was founded in Ecuador and the majority of its members poses a Ecuadorian citizenship. Additionally the members of H^{3}UMANA have direct access to several medical practices due their personal network. As such H^{3}UMANA sees the status quo of Ecuador as an opportunity which shall be address foremost by the group.

2.3 Problem Statement

H^{3}UMANA wants to provide a solution which addresses following problems:

- Patient data being lost or not findable in a sufficient amount of time.
- Patient data not able to be accessed by entities in case of emergencies.
- Patient data not being tracked.
- Insightful statistics not being executed due inflicted cost by summarizing data.

2.4 Proposed Solution

The solution to the problem statement [2.3] should allow medical institutions within Ecuador to maintain their client information digitally and so with allows them to tackle the problems inflicted by a paper based system.

Furthermore it shall also grant following benefits: Increased Efficiency, decreased maintenance costs, possible transparency and compliance with new government regulations [2.1.1].

Additionally H^{3}UMANA aims to gain a corner stone with the implementation of proposed solution on which further products for tackling the status quo of Ecuador [2.1] can be developed.

2.5 Aim

The project aims to develop a prototype for H^{3}UMANA addressing the stated problems in [2.3]. It would allow H^{3}UMANA to start a pilot project with an identified client of their own. The solution should be implemented as web application utilizing open source technologies which will allow H^{3}UMANA to own and modify any source code.

2.6 Objectives

The project objectives state specific goals which shall be achieved in order to satisfy the project aim.

ID	Objective
OBJ01	The project identifies the MVP(minimal viable product) of the solution.
OBJ02	The solution allows to manage client data.
OBJ03	The solution is tested automatically.
OBJ04	The modules of the solution are interchangeable.
OBJ05	The solution could be used for a pilot project (with constrains).

Table 2 Project Objectives

2.7 Deliverables

Based on the aim [2.5] and objectives [2.6] following deliverables can be defined:

- Prototype system implementing MVP requirements.
- A interface (User Interface) providing access to all implemented features.
- A data storage which stores patient data of the system.
- Logic functions which define the business rules of the system.
- Developer documentation of the system.

2.8 Exclusions

Following exclusions for the scope of this project were defined:

- Depending on the implementation progress desirable, optional and enhanced user stories may or may not be implemented into the system
- The system will not work on a distributed infrastructure. E.G Server machine and client computer.
- The system will not assess cybersecurity threads and implement security measures, despite those which are mentioned in the requirements.
- The project will not deliver a saleable system due missing security mechanism.
- The system will not provide user documentation.
- The interface will not be assessed with mobile devices.
- The project will not assess a data backup system.

2.9 Constraints

Following project constraints were identified:

- The system must function without any dependency on the internet.
- The system is based on open source technologies.
- The system shall work performant on single computer systems.

2.10 Assumptions

Following project assumptions were identified:

- The hardware and infrastructure for running the system is provided by the client or H^{3}UMANA.
- The client of H^{3}UMANA uses a windows machine with google chrome as installed browse

3 Methodologies & Practices

This chapter defines and introduce methods and practices which were applied in the project.

3.1 Agile Software Development

This project will be executed in an agile/scrum like manner. Agile is an umbrella term for a group of processes, principles and methods, which allow for agility in regards of changing requirements and objectives of a software project by delivering incremental steps towards a solution. ([6] Jonathan Rasmusson, -). More detailed information can be found in ([1] Samuel A. Marti, 2018).

Note: *As defined in the glossary [1.1] the terms requirement and story are interchangeable in the context within this project due the usage of agile.*

3.1.1 Justification

Following reasons give a justification for using agile.

- Each iteration includes all deliverables such as documentation, validation etc. which allows to scale the amount of implemented features for final product according to the sprint velocity.
- The current industry is eager to move the software development into agile frameworks and experience in that methodology is valuable.
- The assignee has spent a considerable amount of time doing waterfall ([7] Software Testing Help, 2018) or V-Model ([8] Andrew Powell-Morse, 2016) driven projects and wants to deepen the experience of agile.

3.1.2 Product Backlog Refinement

Product backlog refinement is a method where the product owner is sorting the story backlog according to priorities starting with the most important story. ([10] Robin Hackshall, 2014)

3.1.3 Story Points and working hours

The agile methodology discourage the usage of time in order to track progress or effort. However due requirements of the course "*Project Design Implementation and Evaluatio*n" it is necessary to do due so.

Hence a constant amount of time should be assigned for each story point. For simplicity reasons this project will use following rules:

1 Story Point = 60 Minutes of working time.

3.1.3.1 Relative Effort Definition

The table below defines the used story points within this project and their relative effort.

Value	Relative Effort	Example
1	Very small effort	Writing an elaborate email, implementing a unit test.
2	Small effort	Implementing a single function, preparing for an interview.
3	Medium effort	Create an architecture for a library, provide a presentation of the project.
5	Large effort	Research a technology, provide a training.
8	Very large effort	Implement an algorithm solving the 8 queens problem, Execute a market research.
13	Big effort	Create an architecture for a software system.
20	Huge effort	Create a single webpage for a café shop, provide a project specification.

Table 3 Story Points to Effort definition

3.2 Technical Solution Research

This method defines the way how technical solutions for given problems were researched and compared in order to gain a conclusion for a given problem.

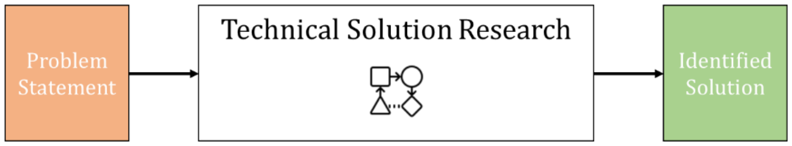


Figure 1 Technical Solution Research I/O

3.2.1 Procedure

The technical solution research follows following procedure.

Step	Description	Comment
1	Define Problem Statement	The statement should provide the necessary insights in order to find a solution.
2	Derive Key Parameters	Use the problem statement and the project context in order to identifying key parameter / features of a potential solution.
3	Define Timeframe for Solution Discovery	Define how much time shall be spend for solution discovery. The time shall be set in context with the effort estimation of the particular story.
4	Discover Potential Solutions	Find so many as possible solutions within the defined time frame.
5	Solution Research & Assessment	Research each solution so that an assessment of each defined key parameters can be done.
6	Solution Comparison	Compare the assessed solutions.
7	Conclusion	Draw a conclusion of the comparison and identify the best solution for given problem in regards of the key parameters.

Table 4 Technical Solution Research Procedure

3.2.2 Justification

The method was designed in order to reach a technical conclusion within a defined amount of time. As such It aims to ensure that an appropriate solution is found within a given time frame and thus tackles the problem of time management in regards of research where the time spend researching a technical problem is often disproportionate high compared towards the gained benefits or reduced implementation time.

3.3 Requirement Implementation Process

This process defines how a requirement/story is implemented within this project, while complying with the defined agile methodology in [3.1].

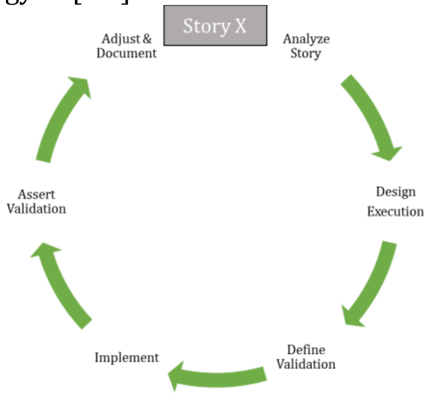


Figure 2 Implementation Process

Step	Description	Comment
1	Analyse Story	The story will be analysed and sub-tasks defined.
2	Design Execution	Asses the following: What & how will something change? What is affected? How can it be implemented/executed?
3	Define Validation	Assess the following: When is the story done? When is it implemented wrong? When is it implemented correct? Define criteria's for assessing the implementation.
4	Execution	Implement story according the design in regards of the validation.
5	Assert Validation	Research each solution so that an assessment of each defined key parameters can be done.
6	Adjust Documentation	Compare the assessed solutions.

Table 5 Requirement Implementation Process

3.3.1 Justification

The defined process shall ensure that each story is implemented with caution and quality. That is achieved by first analysing, designing the implementation and validation, so that execution is done tough full and can be asserted in the end.

3.4 Software Development Principles

Following applied practices are purely concerned with the craftsmanship of developing software.

3.4.1 Clean Code

Clean code defines several software design principles in order to make code more readable and maintainable. The principles used within this project are mainly based on the "Clean Code" book from Robert C. Martin. ([15] Martin, Robter C., 2008)

3.4.1.1 General Principles

ID	Name	Comment
CCP1	Follow Conventions	The code shall be design after an coding convention be consistent with its naming etc.
CCP2	Keep it Simple & Stupid	The solutions shall be always the simplest implementation possible and aim to reduce complexity.
CCP3	Boy Scout	Leave the code always cleaner than before.
CCP4	Polymorphism over if/else	Reduces complexity and increase reusability.
CCP5	Descriptive Naming	Variables, classes & functions shall be named unambiguously be not afraid of lengthy naming. Use pronounceable naming. No prefix or type information, that reduces readability.
CCP6	Small Functions	Small functions are understood faster.
CCP7	Function do only one thing	Function shall implement only one behaviour which can be conveyed solely by the function name.
CCP8	Functions have few arguments	Function shall implement few arguments which ensure simplicity, readability and maintainability.
CCP9	Avoid Commenting	Commenting code is redundant due obviosity of its meaning. If comment seems necessary the code might be not clean.

Table 6 General Clean Code Principles

3.4.2 Documentation

Software inline documentation shall be used in order to provide sufficient developer documentation which can be used in order to modify of further develop the software.

4 Requirements

During an interview and consultancy with H^{3}UMANA and one of their clients ([3] Mino, 2018). User stories were generated and agreed upon.

A user story aims to state a software requirement from a user perspective.
The user stories within this project have following structure.

As a <ROLE>, I want <FEATURE/PROBLEM/GOAL>, so that <REASON>

Figure 3 User Story Template

Example: A user story for a butcher knife could be like this:
As a Butcher, I want to cut my ham in pieces, so that I can sell it in smaller quantities.

Based on the identified user stories functional requirements, non-functional requirements and acceptance criteria's can be derived.

4.1 Prioritization

The user stories were prioritized in following manner

Sym.	**Priority**	**Description**
M	Mandatory	The sum of mandatory user stories result in the MVP (Minimal Viable Product). They must be implemented in order to gain a useable product.
D	Desirable	User stories which contain important features, but can be stripped away from the MVP. They shall be implemented unless the effort is too high.
O	Optional	User stories which were deemed as a useful but not important. They can be implemented if all desirable and mandatory features are present in the product and the project has still capacity left.
E	Enhancing	User stories which are clearly out of the scope of the project capacities, but still worthwhile being noted for future versions/continuations of the product/project.
L	Last	The last stories which shall be implemented in the end of the project. They are usually summarized in a release sprint.

Table 7 User Story Prioritization

4.2 Roles

During the interview following roles were identified:

User Symbol	**Role**	**Examples**	**Description**
U	Medical Personal	Doctors, Nurses	The Staff of the institution using the product in order to enter handle client data.
D	Dev Team	Developer, Tester, Product Owner	The Staff concerned with the implementation of the solution.

Table 8 User Roles Definition

4.3 Identified User Stories

Following user stories were identified during the interview with H^{3}UMANA and one of their clients ([3] Mino, 2018).

ID	User	Statement	Cat.
US01	U	As a User, I want to add new clients into my system, so that I can store their data.	M
US02	U	As a User, I want to modify client information in the system, so that I can keep their data relevant and up to date.	M
US03	U	As a User, I want to delete client information in the system, so that I can correct mistakes or remove client information.	M
US04	U	As a User, I want to retrieve specific client information from the system, so that I can use it for my work with the client.	M
US05	U	As a User, I want to retrieve a history log of entries in regards of specific client information, so that I can gain a historical context with the client which helps me to make decisions.	D
US06	U	As a User, I want to maintain an account of myself, so that I can share information of myself and my field of work with other users.	D
US07	U	As a User, I want to enforce a login system with user accounts, so that I can ensure just authorized users to have access to the system.	D
US08	U	As a User, I want to retrieve information of a specific user, so that I can see their profile and transfer clients correctly.	D
US09	U	As a User, I want to have a streamlined process for client interactions during a regular check-up, so that information can be gathered easily and consistently.	D
US10	U	As a User, I want to link vaccinations with a specific client, so that I can have an overview in regards of vaccinations done and missing.	O
US11	U	As a User, I want to link prescribed medicaments with a specific client, so that so that I can have an overview and also prevent conflicts with other medicaments.	O
US12	U	As a User, I want to link allergies with a specific client, so that I can have an overview and prevent conflicts.	O
US13	U	As a User, I want to add new medicaments to my system so that I can link them with my clients.	O
US14	U	As a User, I want to add new of vaccinations into my system so that I can link them with my clients.	O
US15	U	As a User, I want to add new allergies into my system so that I can link them with my clients.	O
US16	U	As a User, I want to add costume categories of medical information which can then be linked with specific clients, so that I can adjust the system towards my needs.	E
US17	U	As a User, I want to define costume client interaction processes, so that I can streamline other interactions types across my clinic.	E

Table 9 Identified User Stories

4.3.1 Requirement Assessment

Referring to This chapter assess the requirements necessary in order to implement each mandatory user stories. The desirable, optional and enhancing user stories will not be assessed, due their low probability of being implemented within this project.

4.3.1.1 Definitions

From each user story following specification will be derived: Functional Requirements, Non-Functional Requirements and Acceptance criteria. Furthermore the functional and non-functional requirements will be categorised depending on the area of effect in following categories: Interface (User-Interface), Logic, Data Storage.

4.3.1.2 Mandatory Requirements

ID	Statement
US01	As a User, I want to add new clients into my system, so that I can store their data.

Category	Functional Requirement
Interface	Provide a form for adding new clients to the system.
	Provide a feedback if the new client was added successfully or not.
Logic	Check if the new client already exists in the data storage.
	Check if the new client was stored successfully in the data storage.
	Check if the entered client information is valid.
Data Storage	The data storage is able to contain following client related information:

Name	Blood Pressure
Age	Allergies
Sex	Medicamentation
Social Security Number	Comments
Residence	Treatments
Family	Disease
Weight	Symptoms

Category	Non-Functional Requirement
System	The system takes X<1s for adding a new client to the data storage until providing the feedback to the user interface.

ID	Acceptance Criteria
US01AC01	The user can add new clients to the system.

Table 10 US01

ID	Statement
US02	As a User, I want to modify client information in the system, so that I can keep their data relevant and up to date.

Category	Functional Requirement
Interface	Provide a form for modifying the client information.
	Provide a feedback if the client information was modified correctly.
Logic	Check if the modified client information is valid.
	Check if the modified information was updated correctly in the data storage.
Data Storage	The data storage allows to update the client information defined in #US01.

ID	Acceptance Criteria
US02AC01	The user can modify client information.

Table 11 US02

ID	Statement
US03	As a User, I want to delete client information in the system, so that I can correct mistakes or remove client information.

Category	Functional Requirement
Interface	Provide a form for removing the client information.
	Ask the user to verify him or herself.
	Ask the user for additional confirmation
	Provide a feedback if the client information was removed correctly.
Logic	Check if the client information was removed correctly.
	Check if the user was verified correctly for removing the client data.
Data Storage	The data storage allows to remove client information defined in #US01.

ID	Acceptance Criteria
US03AC01	The user can remove client information.

Table 12 US03

ID	Statement
US04	As a User, I want to retrieve specific client information from the system, so that I can use it for my work with the client.

Category	Functional Requirement
Interface	Provide a form which allows to search for clients based on unique personal information.
	Provide a from which displays the information of the selected client.
Data Storage	The data storage allows to read client information defined in #US01.

ID	Acceptance Criteria
US04AC01	The user can search for clients based on personal information. Such as: Name, Address, Social Security Number. In less than 1 second for each 500 clients in the database.

Table 13 US04

4.4 Identified Implementation Stories

Implementation stories were generated by analysing the project objectives [2.6] and identified user stories [4.3]. They aim to address necessary tasks which need to be executed and are not being explicitly noted within the user stories.

Example: As a Dev, I want know the software architecture of the system to be build, so that I can find appropriate technologies to use.

ID	User	Statement	Cat.
IS01	D	As a User, I want know the software architecture of the system to be build, so that I can find appropriate technologies to use.	M
IS02	D	As a User, I want to know the technologies used within the project, so that I can build the software infrastructure.	M
IS03	D	As a User, I want to have the software infrastructure, so that I can start implementing user stories.	M
IS04	D	As a User, I want to assess software validation practices, so that I can implement appropriate and efficient software validation.	M
IS05	D	As a User, I want to implement software validation practices, so that the quality of my software is ensured throughout the project.	M
IS06	D	As a User, I want to assess software documentation practices, so that I can apply appropriate software documentation.	M
IS07	D	As a User, I want to implement software documentation practices, so that the produced software is easy to understand and maintainable.	M
IS08	D	As a User, I want to validate the produced software from a user perspective, so that I can ensure its value proposition.	M
IS09	D	As a Dev, I want to ensure that my documentation & Software is consistent and correct, so that further development can be executed with ease.	M
IS10	D	As a Dev, I want to ensure that the user documentation is consistent and describes all necessary aspects of the product.	M

Table 14 Implementation User Stories

4.5 General Acceptance Criteria's

Based on the defined deliverables [2.7] following acceptance criteria's apply in general, in disregard of a particular story.

ID	Acceptance Criteria
AC01	A functional prototype of the system implementing at least all mandatory user stories.
AC02	The interface is displayed correctly in the user system.
AC03	All implemented features are accessible over the interface.
AC04	The data storage works correctly and inputted data is stored accordingly.
AC05	In case of errors the user is notified with an appropriate message.

Table 15 General Acceptance Criteria's

5 Risk Assessment

The following risk assessment was generated by:

1. splitting the project into theoretical domains namely: Resources, Functionality, Quality, Safety and identifying.
2. Assess probability and impact for each risk.
3. Define mitigation actions for high scoring risks.

5.1 Scoring Risks

The risk score is calculated by multiplying the impact and the probability score.
The score is used for prioritizing and identifying huge risks for the project execution.
Following scoring scale was used for the Impact and Probability.

Score	Probability	Impact
1	Low	Low
3	Medium	Medium
5	High	High

Table 16 Risk Scoring Scale

5.2 Risk Assessment

ID	Risk	Probability	Impact	Score	Mitigation Strategy
R01	Project is not completed within given time.	3	5	15	1. Prioritize Core Functionality (MVP) 2. Minimize usage of technologies where staff is not experienced in.
R02	Functionalities not implemented in accordance to customer needs.	3	5	15	1. Collaborate and review functionality with H^3UMANA and their client for each sprint.
R03	Quality of implemented functionalities not sufficient.	3	3	9	1. Define and apply quality criteria 2. Define and meet acceptance criteria of stories. 3. Review quality with H^3UMANA. 4. Define and execute functional testing.
R04	Code is not maintainable.	3	3	9	1. Define coding principles and standards. 2. Define code documentation standards.
R05	Code quality is not consistent	3	3	9	1. Define code validation standards
R06	Wrong data result lead to hazardous situation	1	5	5	1. Ensure data integrity. 2. Define disclaimer towards user.
R07	Chosen technologies not suitable for solution	1	3	3	1. Choose technologies based on research and comparison. (keep staff experience in mind)
R08	Project staff misses experience in certain technologies	5	3	15	1. Account for training time in planning and effort estimations. 2. Minimize usage of technologies where staff is unexperienced in.

Table 17 Risk Assessment

6 Implementation Plan

The implementation plan consist of refining the backlog and creating a schedule for the implementation.

6.1 Backlog Refinement

In order to plan the implementation a backlog refinement [3.1.2] was executed on the current product backlog consisting of user [4.3] and implementation [4.4] stories.

ID	User	Text	Priority	Position	SP SUM	Research	Design	Implementation	Validation	Documentation
IS01	D	As a User, I want know the software architecture of the system to be build, so that I can find appropriate technologies to use.	M	1	**6**	2	2			2
IS02	U	As a User, I want to know the technologies used within the project, so that I can build the software infrastructure.	M	2	**1**	1				
IS03	D	As a User, I want to have the software infrastructure, so that I can start implementing user stories.	M	3	**7**	1	1	3	1	1
US01	U	As a User, I want to add new clients into my system, so that I can store their data.	M	4	**9**		2	5	1	1
IS04	D	As a User, I want to assess software validation practices, so that I can implement appropriate and efficient software validation.	M	5	**2**	2				
IS05	D	As a User, I want to implement software validation practices, so that the quality of my software is ensured throughout the project.	M	6	**6**		1	3	1	1
US02	U	As a User, I want to modify client information in the system, so that I can keep their data relevant and up to date.	M	7	**13**		3	7	1	2
IS06	D	As a User, I want to assess software documentation practices, so that I can apply appropriate software documentation.	M	8	**2**	1	1			
IS07	D	As a User, I want to implement software documentation practices, so that the produced software is easy to understand and maintainable.	M	9	**3**			2		1
US03	U	As a User, I want to delete client information in the system, so that I can correct mistakes or remove client information.	M	10	**4**			2	1	1
US04	U	As a User, I want to retrieve specific client information from the system, so that I can use it for my work with the client.	M	11	**8**		2	4	1	1
US05	U	As a User, I want to retrieve a history log of entries in regards of specific client information, so that I can gain a historical context with the client which helps me to make decisions.	D	12	**17**	1	4	9	2	1
US06	U	As a User, I want to maintain an account of myself, so that I can share information of myself and my field of work with other users.	D	13	**7**		1	4	1	1
US07	U	As a User, I want to enforce a login system with user accounts, so that I can ensure just authorized users to have access to the system.	D	14	**9**		1	6	1	1
US08	U	As a User, I want to retrieve information of a specific user, so that I can see their profile and transfer clients correctly.	D	15	**7**		1	4	1	1
US09	U	As a User, I want to have a streamlined process for client interactions during a regular check-up, so that information can be gathered easily and consistently.	D	16	**15**	1	2	10	1	1
US10	U	As a User, I want to link vaccinations with a specific client, so that I can have an overview in regards of vaccinations done and missing.	O	17	**6**		1	3	1	1
US11	U	As a User, I want to link prescribed medicaments with a specific client, so that so that I can have an overview and also prevent conflicts with other medicaments.	O	18	**6**		1	3	1	1
US12	U	As a User, I want to link allergies with a specific client, so that I can have an overview and prevent conflicts.	O	19	**6**		1	3	1	1
US13	U	As a User, I want to add new medicaments to my system so that I can link them with my clients.	O	20	**6**		1	3	1	1
US14	U	As a User, I want to add new of vaccinations into my system so that I can link them with my clients.	O	21	**6**		1	3	1	1
US15	U	As a User, I want to add new allergies into my system so that I can link them with my clients.	O	22	**6**		1	3	1	1
US16	U	As a User, I want to add costume categories of medical information which can then be linked with specific clients, so that I can adjust the system towards my needs.	E	23	**31**		2	26	2	1
US17	U	As a User, I want to define costume client interaction processes, so that I can streamline other interactions types across my clinic.	E	24	**43**		3	32	5	3
IS08	D	As a User, I want to validate the produced software from a user perspective, so that I can ensure its value proposition.	L	25	**4**				3	1
IS09	D	As a Dev, I want to ensure that my developer documentation & Software is consistent and correct, so that further development can be executed with ease.	L	26	**8**			3	2	3
IS10	D	As a Dev, I want to ensure that the user documentation is consistent and describes all necessary aspects of the product.	L	27	**5**				1	4

6.1.1 Backlog Analysis

The following figure plots the refined backlog in a story point relationship.

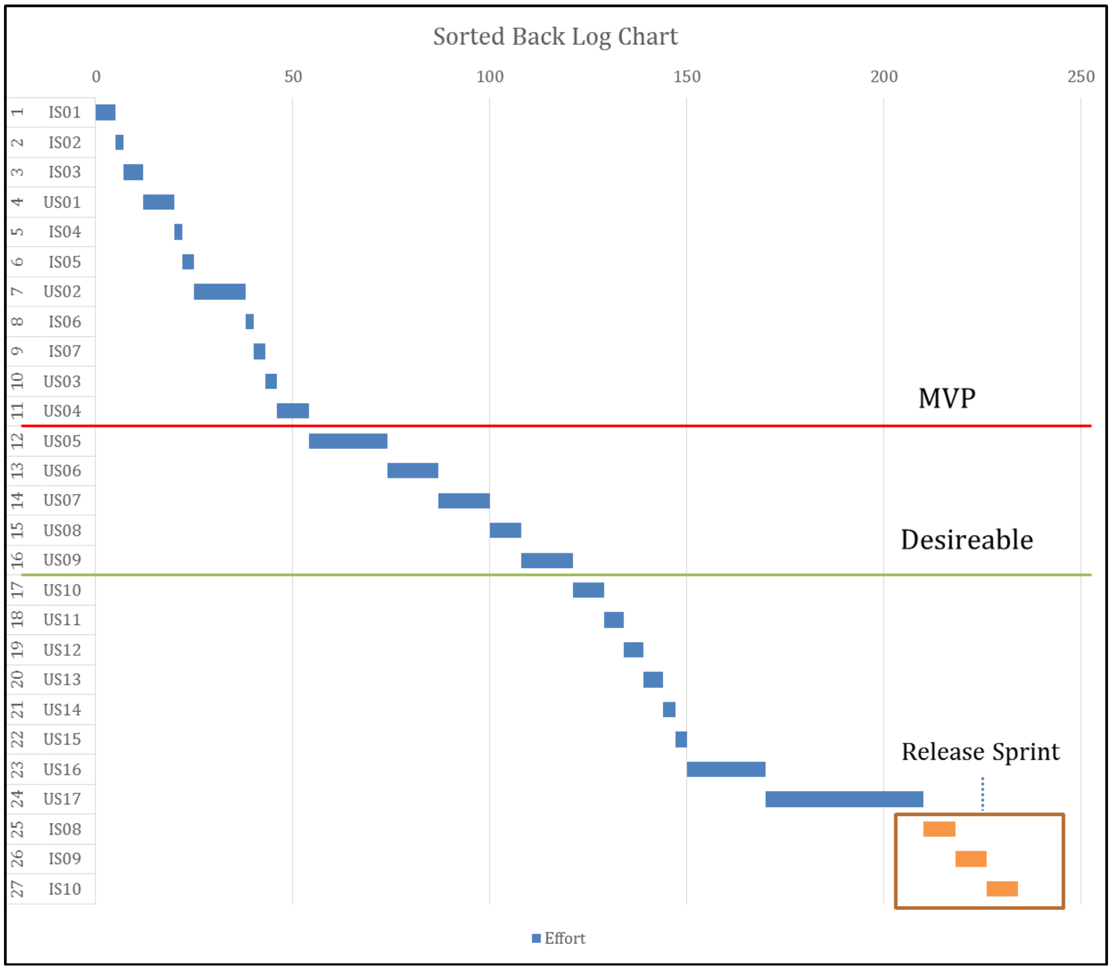


Figure 4 Backlog Chart

Using [Figure 4 Backlog Chart] different level of scopes can be defined:

Scope	Story Points	Hours
Whole Backlog	243	243
MVP	61	61
Desirable	116	116
Release Sprint	17	17

Table 18 Implementation Scopes

Due the risk assessment in [5, R1], the core functionality shall be focused. Thus in order to gain a solution satisfying the objectives [2.6] the MVP and release sprint must be implemented within this project. That scope would lead to a total 78 SP. (78 hours according to [3.1.3])

Following pie charts visualize the sum of story points spend for each activity, once for the whole refined product backlog and once for the MVP and release sprint.

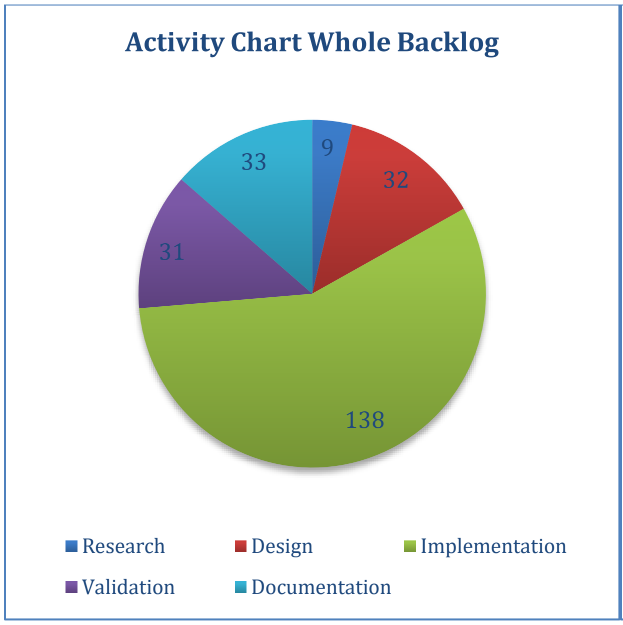


Figure 5 Activity Pie Whole Backlog

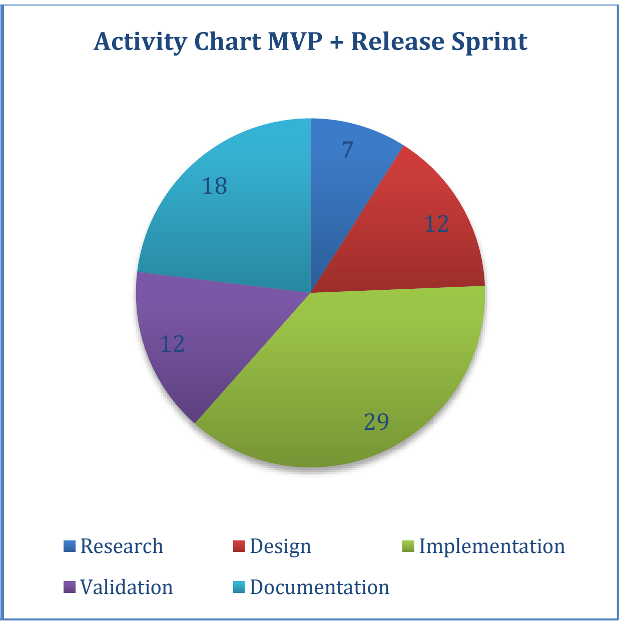


Figure 6 Activity Pie Minimal Scope

One can observe that in a scope of MPV + Release Sprint the amount of story points spent on implementation amounts for less than 50%, while with the whole backlog as scope the Implementation activity would account for more than the half. That difference is mainly due the start and end of the backlog which account for a fixed amount of research, validation & documentation.

6.2 Sprint Schedule

Based on [6.1.1] the sprint plan can be drawn as follows.
One can observe five sprints being scheduled with each accounting for two weeks.

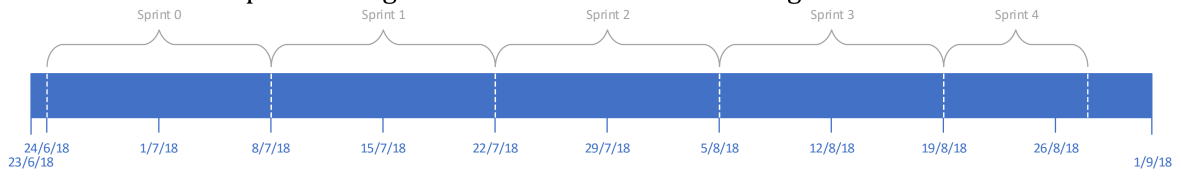


Figure 7 Sprint Schedule

6.3 Sprint Velocity

Based on the MVP defined in the refined backlog [6.1] The initial velocity for each sprint should be set to 16SP. That would lead to a product with ~80SP being implemented over all sprints.

Note: *Sprint velocity can change over the course of the project execution. However the sprint velocity shall never drop below 15SP so that the MVP scope is ensured.*

Due the low probability of implementation within this project all stories with an priority of optional and enhancing [4.1] were not assessed within this backlog refinement.

7 Implementation

The implementation was executed according to the sprint schedule [6.2] and backlog refinement [6.1]. However this chapter will describe the results of the implementation in a non-chronical manner in order group similar subjects into logical chapters.

7.1 Software Architecture

7.1.1 Statement

The chosen architecture follows aims to comply with following statements:

- The mitigation strategies of the risk assessment [5], R01, R08 require to minimize the introduction of new technologies or patterns.
- According the objective [2.6, OBJ04] the software shall be implemented in a modular fashion. In order to be interchangeable.
  Deducted statements:
  - GUI components shall be split into separate modules.
  - The GUI is build up by code generation/construction in which the HTML is generated.
  - A single webpage is constructed based on an aggregation of GUI elements.

7.1.2 Pattern

Concluding on [7.1.1] the architecture will orientate itself around the MVC pattern, mainly due the familiarity and simplicity of it.

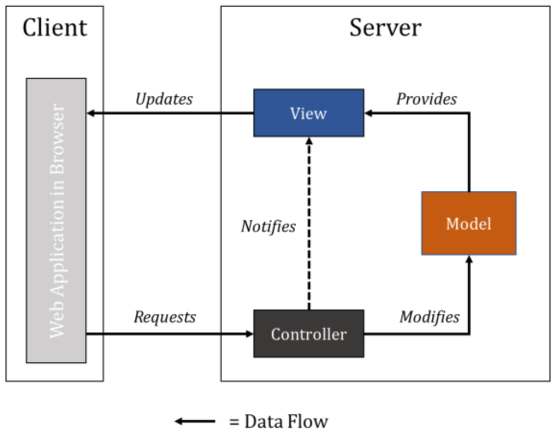


Figure 8 MVC Pattern

View: The user interface used for data visualization and providing the user with user controls.
Controller: Running the business logic of the application and is initiated by requests.
Model: Represents the data model and is updated by the controller while being used by the.

The in [Figure 8 MVC Pattern] described architectural pattern allows to split the concerns of the web application between defined components. Following described pattern will increases the maintainability and modularity of the resulting software architecture and so with comply with the architectural statement.

7.1.3 Components

Based on [Figure 8 MVC Pattern], the software architecture statements [7.1.1] and identified user stories [4.3] following software architecture was designed.

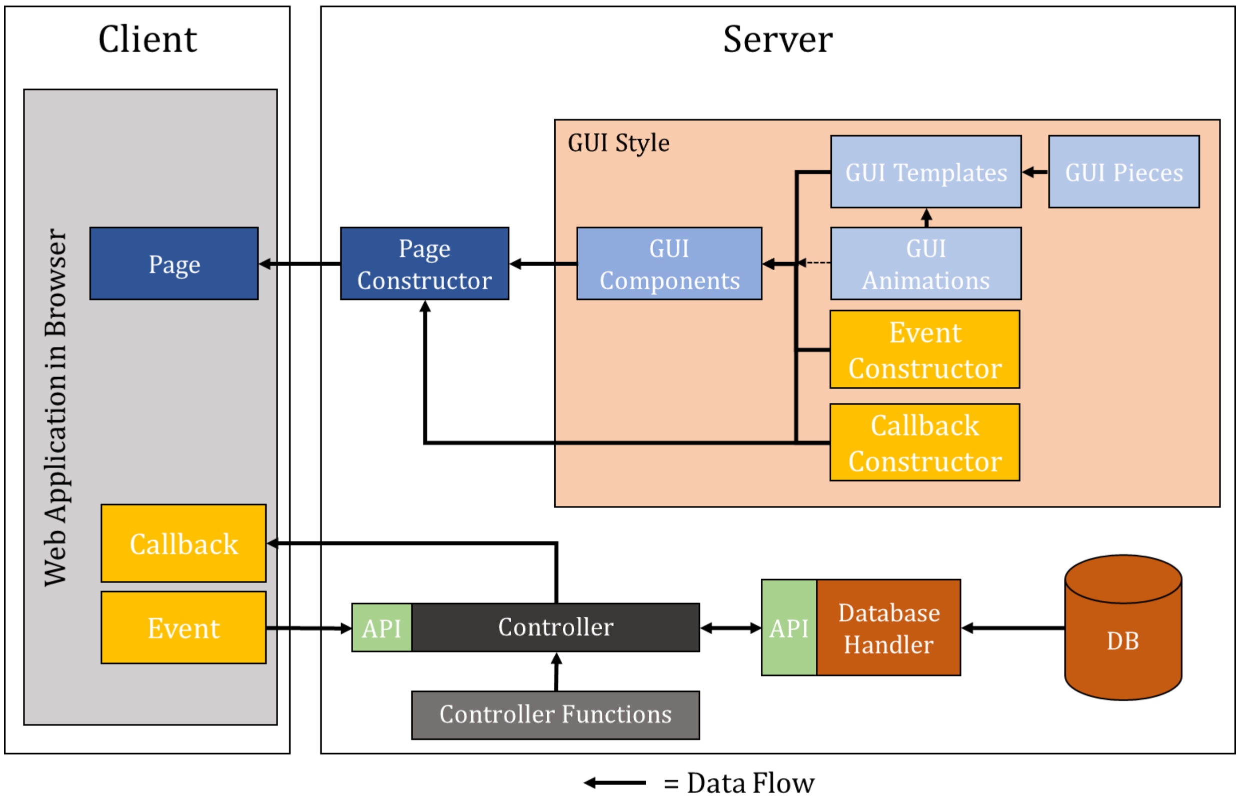


Figure 9 Component Architecture

7.1.3.1 Model

As described in [7.1.2] following components are part of the model.

7.1.3.1.1 Database

The database contains and maintains the data of the system. It must implement a sort of query language which allows to retrieve data and manipulate it.

7.1.3.1.2 Database Handler

The database handler provides an accessible API towards the system which allows to execute pre-defined queries. By defining the interface of the API the database can be interchanged without effecting the rest of the system solely a new database handler would need to be written.

7.1.3.2 Controller

Controls the system response to requests and controls the flow of data.

7.1.3.2.1 Controller

The controller of a particular event or page. It will receive, validate and process data by using the database handler [7.1.3.1.2] in order to response to the Callback.

7.1.3.2.2 Controller Functions

A library containing function which are used by the controllers. The aim is to increase maintainability and reusability by that.

7.1.3.3 View
Assembles all information and GUI elements in order to create a perceivable visualization for the user.

7.1.3.3.1 GUI style
Defines the styling of the GUI elements, such as font, colour and size etc. By separating the styling from the other GUI elements the styles can be changes and replaced easily without affecting the page structure.

7.1.3.3.2 GUI Templates
Contain finished formatted GUI structures using style and animations, but yet do not contain any data or functions. That approach allows to reuse GUI templates with different set of data's which will increase the reusability and consistency of the GUI page.

7.1.3.3.3 GUI Animations
Define animations which can be applied to GUI elements.

7.1.3.3.4 Event Constructor
Constructs events/functions used for requesting services from the controller. By separating them from the rest of the GUI elements the reusability is increased.

7.1.3.3.5 Callback Constructor
Constructs the callback function which is triggered due a controller response after and event triggered the controller. The callbacks change the GUI and its represented data due the controller feedback.

7.1.3.3.6 GUI Components
A GUI component can contain several GUI templates, animations and event constructors. Each GUI component represents a reusable piece of GUI which is populated by data. For example a user registration form could be a GUI element. Note: The code of the GUI components will be executed on the client side.

7.1.3.3.7 Page Constructor
Will construct a client browsable page which consist of one or several GUI components and links to one or several controllers. Basically it assembles all pieces of software into a page which can be used by the user in order to manipulate data over the controller towards the database.

7.2 Technical Research

The chapter describes the research conducted within this project and summaries the results.
Note: The here mentioned research was conducted according to the technical research method defined in [3.2].

7.2.1 Backend Programming Language

Identifies the used backend programming language.

7.2.1.1 Problem Statement

The backend(server) noted in [7.1.3] of the software architecture must use a programming language in order to implemented the defined software architecture. It will be used particularly for implementing the controller and the constructing of GUI elements.

7.2.1.2 Key Parameters

Based on [7.2.1.1] following relevant key parameters were identified.

Parameter	Description	Target
Proficiency	How proficient is the staff with the technology. According the risk assessment [5], the level of proficiency of the development staff towards a particular technology is a thread to the project outcome.	High Proficiency
Dependency	How dependent the technology is on platforms and other technologies.	Low Dependency

Table 19 Backend Programming Language Key Parameters

7.2.1.3 Solution Comparison

Due the mitigation risk strategy defined in the risk assessment[5] and the here defined key parameters. Technologies were the staff has no existing proficiency, were excluded from further steps. That affects: ruby, Lua, CoffeeScript, Erlang, .NET C# etc.

Technology	Proficiency	Dependency
JavaScript	Low	In order to use JavaScript as backend technology one would need to utilize node.js or any stack which includes node.js.
Java	Low	Requires server which can execute Java.
PHP	Mediocre	Requires server which can execute PHP.
Python	Mediocre	Require a framework implementing python, such as Django.

Table 20 Scripting Technologies

7.2.1.4 Conclusion

The backend scripting technology will be PHP since the staff has higher experience in it and it does not require to settle for a particular framework.

7.2.2 Type of Database

The defining the used database type will allow to identify the database technology which shall be used.

7.2.2.1 Problem Statement

The database is used for storing the data of the system and shall be flexible and easy to maintain.

7.2.3 Key Parameters

Parameter	Description	Target
Proficiency	How proficient is the staff with the technology. According the risk assessment the level of proficiency of the development staff towards a particular technology is a thread to the project outcome.	High Proficiency
Scalability	How easy can the data structure / database be scaled and modified. Since the project is executed in agile and thus in continuous development that is an important parameter.	High Scalability
Data Relationship	The relationship of the data is very important for the H^{3}UMANA Database. Since the value is within the meaning of the data.	Easily Possible

Table 21 Type of Databases Parameters

7.2.3.1 Discovered Solutions

Relational Database [SQL], Key-Value Based, Graph, Column, Document.

7.2.3.2 Solution Comparison

Technology	Proficiency	Scalability	Data Relationship
Relational Database [SQL]	Mediocre	Low, SQL databases require a static data structure based on tables.	Mediocre, allows for relationships but can get cumbersome by trying join several tables.
Key-Value Based	Low	High, due not being a relational database and thus just saving plain key-value pairs	None, does not support relationships.
Graph	Low	High, does not require any recursive action after a modification of the structure.	High, relationships is the essence of graph databases.
Column	Low	High, unstructured.	None, does not support relationships.
Document	Low	High, sub category of key-value	None, does not support relationships.

Table 22 Database Technologies

7.2.3.3 Conclusion

The project will utilize a graph database. Mainly due the benefits provided by a graph database:

1. A graph database puts empathizes on the relationships of information. For the implementation of the H^{3}UMANA Database this is not particular necessary but in further versions the database might be used in order to gain information based on the relation of data. Thus choosing such a technology is in the future interest of H^{3}UMANA.
2. The graph database does not require to define a rigid data structure, which allows for modifications being in each iteration without the need of adjusting a data structure.

7.2.4 Database Technology

7.2.4.1 Problem Statement

In order to store data a database technology shall be identified and used the technology must be of the graph type due the type of database research.

7.2.4.2 Key Parameters
Following relevant key parameters were identified.

Parameter	Description	Target
Interface	The DB technology must be able to interface to the backend scripting language	Possible
For Free	The DB technology must be for free of charge (Open Source)	For Free
Support	The DB must have a big community in order to find answers and modifications more easily.	Big
Documentation	The documentation must be comprehensive.	Good

Table 23 Database Technology Key Parameters

7.2.4.3 Discovered Solutions
Neo4j, GraphDB, OrientDB, Arango, TitanDB, Apache TinkerPop

7.2.4.4 Solution Comparison
Due the key parameter "For Free" any database asking for charge is immediately excluded from the comparison. Affected are: GraphDB,

Technology	Interface	Support	Documentation
Neo4j	Rest API, Libraries for PHP (Github) ([21] Neo4j, 2018)	Big, Most Commonly used graph DB ([16] DB-ENGINES, 2018)	Good ([19] Neo4j, 2018) many examples and tutorials were found specifically for PHP.
OrientDB	Rest API, PHP Driver ([18] OrientDB, 2018)	Big, Common used graph DB	Medium
Arango	Rest API, PHP Driver ([22] frankmayer, 2018)	Big, Common used graph DB	Medium
TitanDB	Rest API	Medium, less common DB	Medium ([20] TitanDB, 2018)
Apache TinerPop	Rest AIP, PHP Driver ([23] PommeVerte, 2018)	Medium, less common DB	Low, Not many documents or examples were found for PHP.

Table 24 DB Technology Comparison

7.2.4.5 Conclusion
There are many more graph databases which were not discovered within the given time frame. However the already discovered graph databases fulfilled the requirements and provide a big enough sample for a comparison. ([17] Wikipedia, 2018)

The discovered graph DB do not have clear deviations in regards of the key parameters. Thus differentiation which seemed fit was used and Neo4j was chosen based on following reasons:
- Most accessible documentation and community due huge amount of available information.
- Several different PHP drivers indicate a huge and strong community and allow for options.
- Provides web based developer GUI, for running quires and monitoring the DB.

7.2.5 Style Framework
The style framework is used for easier creation of the front end visualization and in order to enforce a consistent look & feel.

7.2.5.1 Problem Statement
Generate easily the GUI by using a styling framework for CSS, HTML and JavaScript.

7.2.5.2 Key Parameters

Following relevant key parameters were identified.

Parameter	Description	Target
Proficiency	How proficient is the staff with the technology.	High Proficiency
Lean	Is the stack lean or comprehensive. This project targets a lean framework due the tight deadlines as defined in the risk assessment.	Lean
Documentation	How well is the framework documented	Good
Responsive	The framework should be responsive, so that in the future a mobile application could be supported	Responsive

Table 25 Backend Scripting Parameters

7.2.5.3 Discovered Solutions

Materialize, Bootstrap, Ink, HTML KickStart, Kickstrap, Pure, HTML5 Boilerplate, YUI, Zimit, Foundation

7.2.5.4 Solution Comparison

Note: Due the risk mitigation strategy of development resources and the key parameter proficiency all discovered solutions with no proficiency by the staff will be excluded from further consideration.

Technology	Proficiency	Lean	Documentation	Responsive
Materialize	Medium	Middle	Good	Supported
Bootstrap	Low	Lean	Good	Supported
Ink	Low	Lean	Good	Supported
Semantic UI	Low	Lean	Good	Supported

Table 26 DB Technology Comparison

7.2.5.5 Conclusion

The compared solutions are very similar in regards of the key parameters. Thus Materialize was chosen because of the staff proficiency with it. The choice was also motivated by the mitigation strategy of the risk assessment [5].

7.2.6 System Testing

System testing technology should allow to access the graphical user interface and manipulate it. So that systems test over it can be executed.

7.2.6.1 Problem Statement

I want to have a technology which allows to manipulate and read the data of the graphical user interface (HTML, DOM) in order to test the system as a whole.

7.2.6.2 Key Parameters

Parameter	Description	Target
Proficiency	How proficient is the staff with the technology. According the risk assessment the level of proficiency of the development staff towards a particular technology is a thread to the project outcome.	High Proficiency
Dependencies	The technology has a low amount of dependencies	Low Dependencies
Simple	Unit test must be created easily, fast and maintainable	Very Simple
Flexibility	How flexible is the testing framework	High

Table 27 System Testing Key Parameters

7.2.6.3 Discovered Solutions

Selenium, fitness, Cucumber

7.2.6.4 Solution Comparison

Technology	Proficiency	Dependencies	Simple	Flexibility
Selenium	Middle	Low	Very	High
Fitness	Low	Middle	Middle	Low
Cucumber	Low	Low	Middle	High

Table 28 System Testing Solution Comparison

7.2.6.5 Conclusion
Selenium was chosen since it has a higher flexibility than fitness due its implementation as python library. Cucumber is providing many additional features which are not necessary for the given problem statement.

7.2.7 Source Documentation
Following technologies were identified in order to support the defined software documentation principles.

7.2.7.1 Problem Statement
I want to have a technology which allows me to create software documentation automatically based on the source code.

7.2.7.2 Key Parameters

Parameter	Description	Target
Proficiency	How proficient is the staff with the technology. According the risk assessment the level of proficiency of the development staff towards a particular technology is a thread to the project outcome.	High Proficiency
Dependencies	The technology has a low amount of dependencies	Low Dependencies
Simple	writing documentation must be created easily, fast and maintainable	Very Simple

Table 29 Source Code Documentation Key Parameters

7.2.7.3 Discovered Solutions
PHPDocumentor, Doxygen, Sami

7.2.7.4 Solution Comparison

Technology	Proficiency	Dependencies	Simple
PHPDocumentor	Low	Low	Mediocre
Doxygen	Middle	Low	Mediocre
Sami	Low	Low	High

Table 30 Source Code Documentation Solution Comparison

7.2.7.5 Conclusion
The discovered solutions are similar in regards of the key parameters thus other criteria's were considered and Doxygen was chosen as in line documentation tool due following reasons.

- Higher staff proficiency than other tools.
- Supported by other several languages not just PHP.

7.2.8 Assumptions

Following technologies were defined to be used within the project without technical solution research [3.2]. Mainly due a lack of alternatives or dependencies given from other technologies.

Technology	**Comment**
JavaScript, HTML, CSS	Used for creating, structuring, animating and requesting websites.
JQuery, AJAX ([24] JQuery, 2018)	JavaScript library providing general functionalities such as an implementation of AJAX.
Graphaware PHP driver ([25] ikwattro, 2018)	Implements a driver for neo4J database in PHP.
MAMP ([26] MAMP, 2018)	An apache webserver able to run PHP, used for prototyping.

Table 31 Assumed Technologies

7.3 Technology Map

Based on the technical research [7.2] and software architecture [7.1]. following technology map can be created.

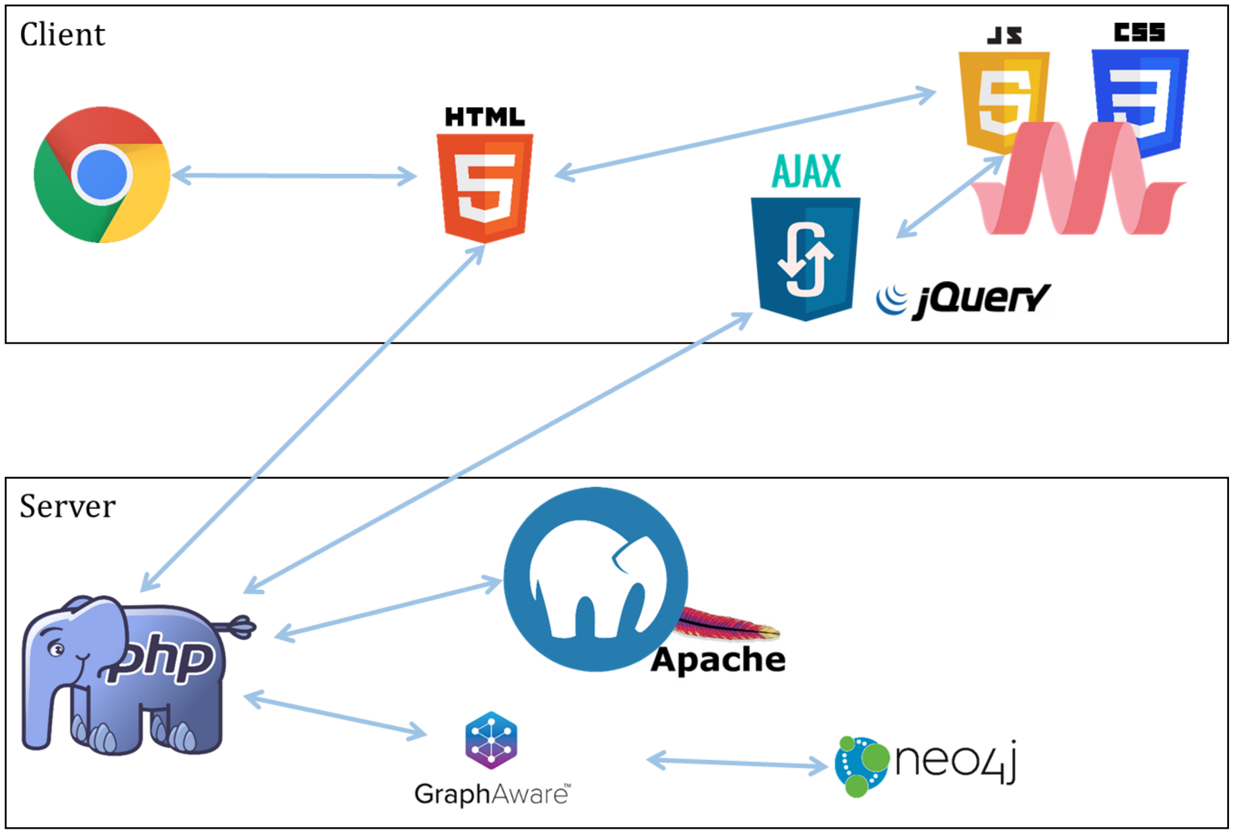


Figure 10 Technology Map

[Figure 10 Technology Map] visualizes the used technologies in their context to each other, in terms of data flow and dependencies.

7.4 Graphical User Interface Design

The graphical user interface was implemented according to following design.

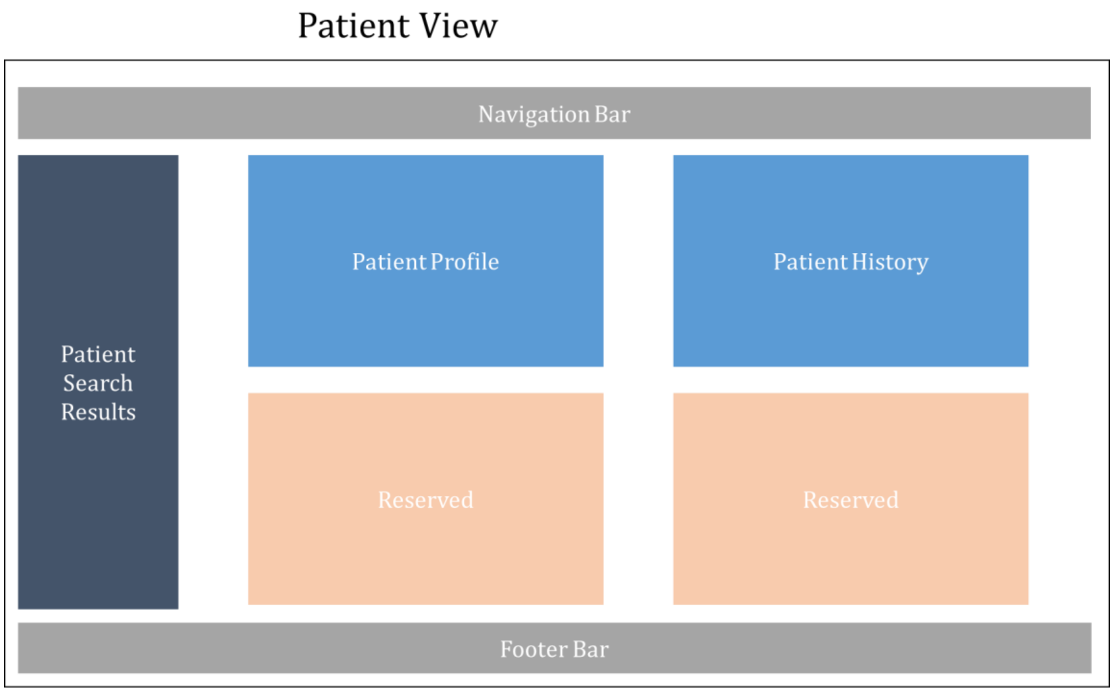


Figure 11 Design Patient View

The design in [Figure 11] was created in collaboration with H^{3}UMANA in brainstorming session. (H^{3}UMANA, 2018). It aims to provide one page where the user would be able to search, select & manipulate patient data.

7.4.1 Patient Search Results

Allows for searching and retrieving of patients.

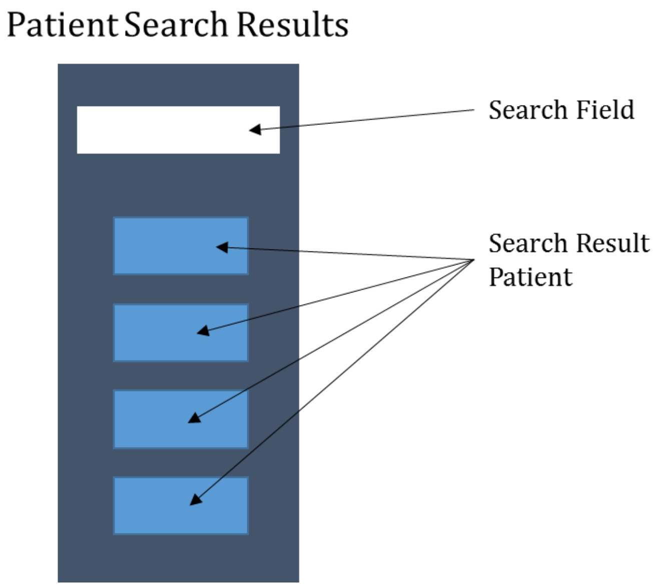


Figure 12 Design Patient Search Window

[Figure 12 Design Patient Search Window] should allow users to search for patients by entering a string into the "Search Field" which can be either a name or social security number. After submitting the query the results will be loaded below as "Search Patient Result" each found patient will be visualised as clickable box containing the names of the patient, additionally if clicked the box will trigger an update event.

7.4.2 Patient Profile

Allows to review patient data and modify it.

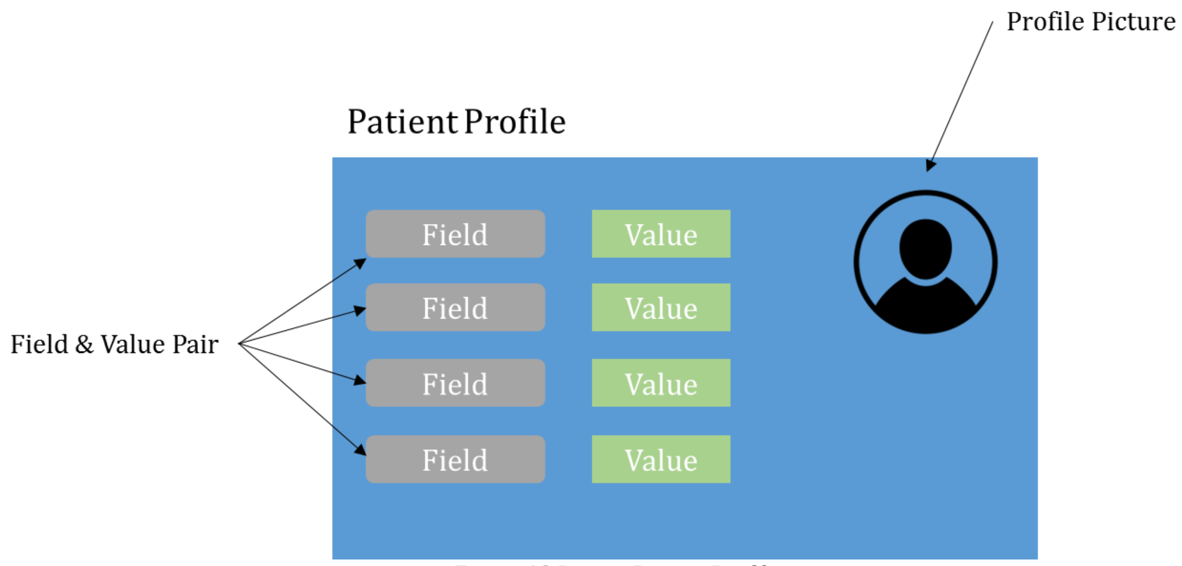


Figure 13 Design Patient Profile

[Figure 13 Design Patient Profile] shall provide an overview of an patient profile. By using the "Field & Value Pairs" the data can be represented and modified where the field indicates the type of data and the value the current value of the particular patient.
Example: First Name: Samuel

7.4.3 Patient History

Allows to review the patient history and add new entries.

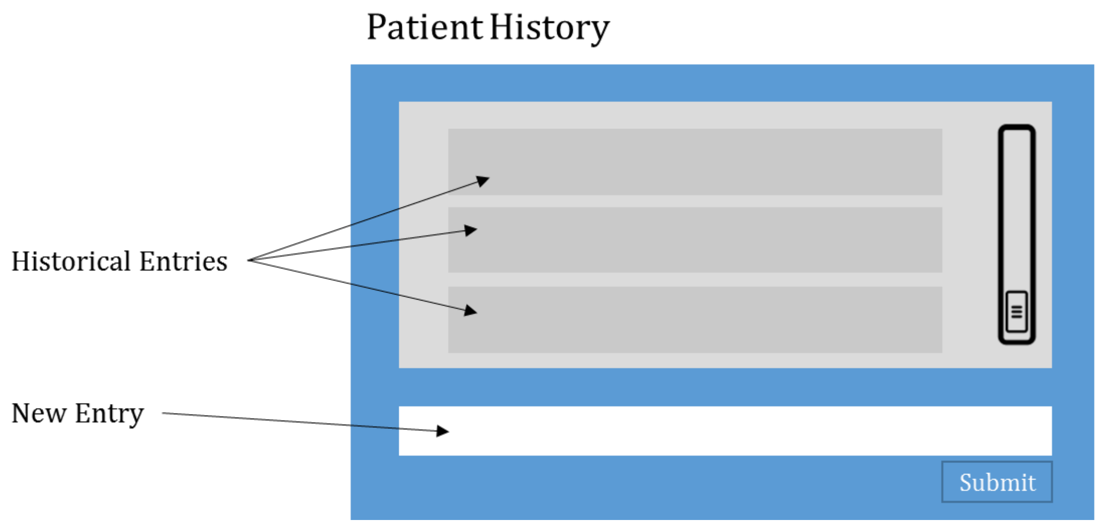


Figure 14 Patient History

[Figure 14 Patient History] provides a scrollable window which contains all "Historical Entries" in a chronological order. A new entry can be added by the "New Entry" field and the submission button. An entry will consist currently only of a undefined string and a timestamp.

7.4.4 Navigation Bar
The navigation bar is with the current implementation solely a placeholder until new pages will be added.

7.4.5 Footer Bar
Will contain actions buttons which will allow for adding or removing of patients.

7.5 Developed Artefacts

This chapter will describe and refer to all artefacts produced within the project timeline.

7.5.1 Web Application Source Code
The source code package is structured within the root folder and contains following:

Directory	Content
/controller	All the implemented controllers as described in the component architecture [7.1.3]
/db	The PHP driver [7.2.8] and implemented database handler as defined in [7.1.3.1.2].
/gui	All the components, templates, styles used within the web application as defined in [7.1.3.3].
/lib	General functions used and implemented
/resources	Resources used by the web application such as pictures etc.

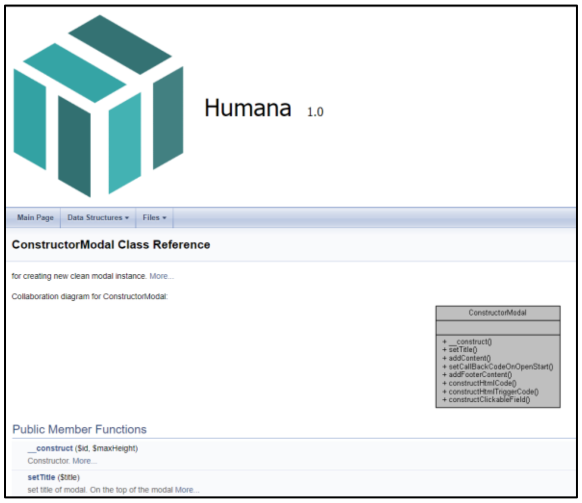
Table 32 Source Code Directories

7.5.1.1 How to Run the Web Application
Following steps need to be executed in order to run the source code of the web application.

1. Start a WebServer which execute PHP on the root directory of the source code.
   Example: MAMP - C:\Users\E9955465\Documents\GitHub\ClientDatabase\root
2. The web application should be reachable on the defined port by the WebServer
   Example: localhost:8080
3. Download install and neo4J https://neo4j.com/download/
4. Run neo4j by following steps:
   a. Start command line
   b. Navigate to /bin folder of neo4J.
   c. Run: "neo4J.bat console" as a command
5. Neo4J web UI should be reachable under localhost:7474

7.5.2 Software Documentation

The software documentation can be opened by running the index.html with a browser. Directory: "/doc/html/index.html". The result could look as following.



ConstructorModal Class Reference

for creating new clean modal instance. More...

Collaboration diagram for ConstructorModal:

```
ConstructorModal
+ __construct()
+ setTitle()
+ addContent()
+ setCallBackCodeOnOpenStart()
+ addFooterContent()
+ constructHtmlCode()
+ constructHtmlTriggerCode()
+ constructClickableField()
```

Public Member Functions

__construct ($id, $maxHeight)
Constructor. More...

setTitle ($title)
set title of modal. On the top of the modal More...

Figure 15 Software Documentation Example

7.5.3 Automatic Tests

Some automatic system tests were implementing using selenium [7.2.6].
They can be run by following steps:

1. Set up environment (WebServer, Neo4J as described in [7.5.1.1].
2. Run the systemTest.py in python 3 from the "/test" directory
   Example:

```
C:\Users\E9955465\Documents\GitHub\ClientDatabase\test>python systemTest.py
```

Figure 16 System Test Example

8 Validation

The validation will be split in several subcategories, in order to reach a higher assessment coverage. It will describe the extend of the implementation, validate the requirements [4.3.1.2] and objectives [2.6].

8.1 Implementation Assessment

This chapter will assess the planned implementation with the de facto state of the current software. The set scope of stories being implement was defined as MVP + End Sprint. Following table will assess each story in regards of its implementation status.

Story ID	**Short Description**	**Status**	**Comment**	**Effort Planned → Real**
IS01	Software Architecture	Done	Can be reviewed at [7.1.3].	6 → 6
IS02	Define Technologies	Done	Can be reviewed at [7.2]. Research was not straight forward and simple as anticipated.	1 → 6
IS03	Software Infrastructure set up	Done	The software infrastructure is it set up and can be found within the source code [7.5]	7 → 7
US01	Add Client to System	Done	Clients can be added.	9 → 9
ISO04	Assess Software Validation	Done	Software validation defined in [7.2.6].	2 → 2
IS05	Implement Software Validation	Partially	The software was validated manually, however automatic testing was just implemented for mandatory requirements.	6 → 6
US02	Modify Client Information	Done	Client Information can be modified. Spend more time on validating.	13 → 15
IS06	Assess Software Documentation	Done	Assessed in [7.2.7].	2 → 1
IS07	Implement Software Documentation	Done	The documentation can be found by following the steps described in [7.5.2].	3 → 4
US03	Delete Client Information	Deferred	The deleting of client information was deferred due missing resources.	4
US04	Retrieve Client Information	Done	Client Information can be retrieved. Learning AJAX needed more time.	8 → 11
IS08	Validate Software	Done	The produced software was validate together with H^{3}UMANA.	4 → 2
IS09	Consistent Documentation and Code	Done	The source code and the documentation was checked for consistency and completeness. Since documentation was included from the beginning less effort was needed.	8 → 5
IS10	User Documentation	Deferred	This story was deferred due resource limitations.	5

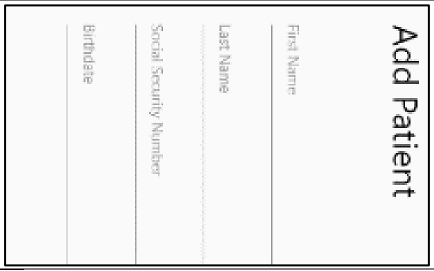
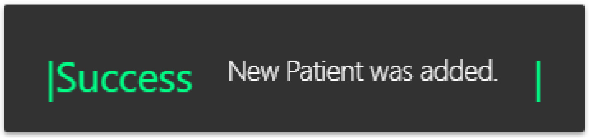
Table 33 Implemented Story Assessment

The bulk of the planned stories were implemented however two stories were deferred due missing resources which was triggered by incorrect estimation of work → Planned amount of SP = 78, Implemented amount of SP = 74 and re estimated SP = 87.

8.2 Requirement Assessment

This chapter will assess the mandatory requirements defined in [4.3.1.2].
The following table assess the requirement for each category and acceptance criteria.

ID	Statement
US01	As a User, I want to add new clients into my system, so that I can store their data.

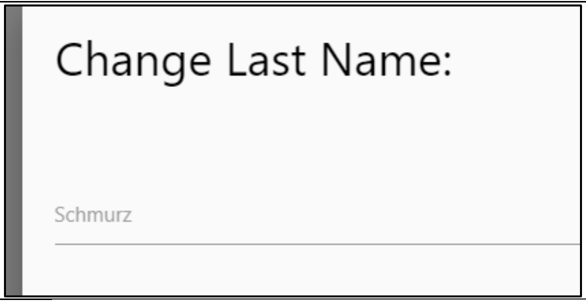
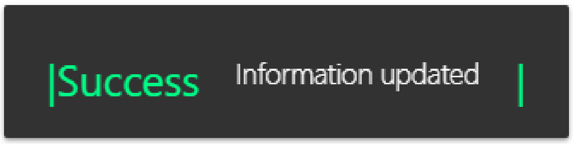
Category	Assessment
Interface	Form provided.  Add Patient First Name Last Name Social Security Number Birthdate
	Feedback provided.  Success New Patient was added.
Logic	Application Checks if client exists.
	Application checks if client is stored correctly.
	The form checks if entered information is valid.
Data Storage	The data storage is able to contain defined data, However the most defined data is not implemented by the web application itself.

Category	Assessment
System	The response time of the system is adequate.

ID	Test Case	Result
US01AC01	1. Open Application 2. Press “PATIENT +” button. 3. Confirm “Add Patient” form is open. 4. Enter patient data: “FistName”, “LastName”, “Birthdate”, “Socialsecruitynumber”. 5. Assess of patient was added or not by feedback notification. 6. Search for patient by using the search field. 7. Assess if newly added patient can be found	OK

Table 34 US01

ID	Statement
US02	As a User, I want to modify client information in the system, so that I can keep their data relevant and up to date.

Category	Functional Requirement
Interface	Form Provided.  Change Last Name: Schmurz
	Feedback provided  Success Information updated
Logic	Is checked partially, missing social security number checking.
	Checked
Data Storage	The data storage allows to update the client information defined in #US01.

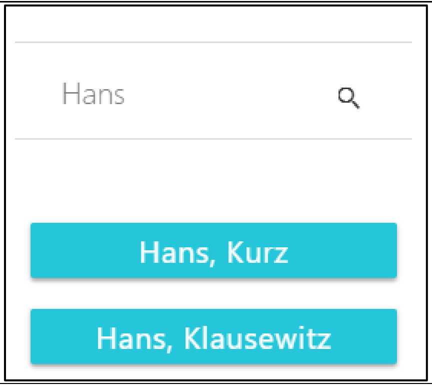
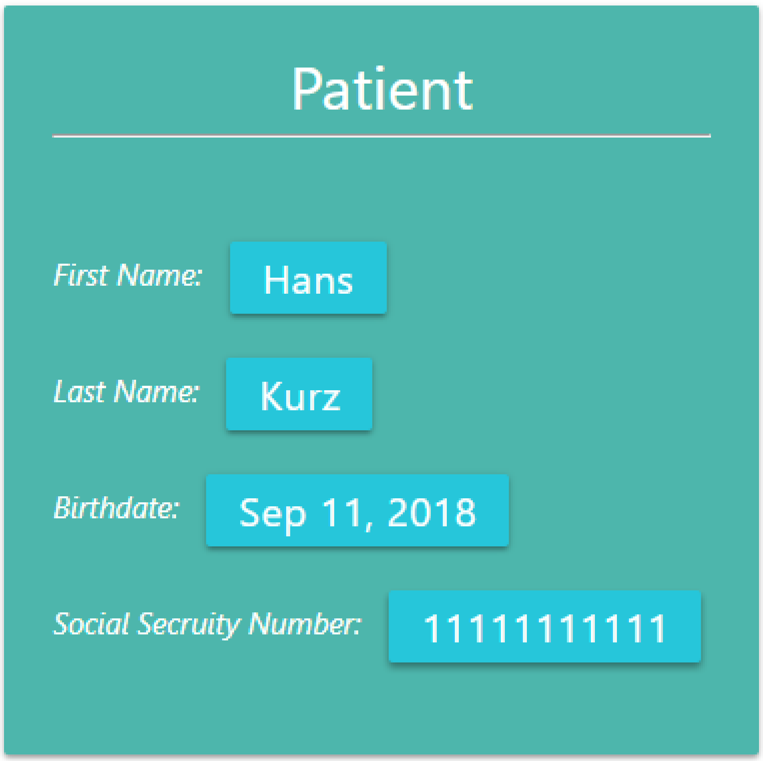
ID	Test Case	Result
US02AC01	1. Open Application 2. Search for existing client 3. Click on target client form the search results 4. Assess if profile window opens. 5. Click on any value to be changed. 6. Assess that change pop us is showing. 7. Change value. 8. Assess that value change was update on page.	OK

Table 35 US02

ID	Statement
US03	As a User, I want to delete client information in the system, so that I can correct mistakes or remove client information.
	Deferred according to [8.1].

Table 36 US03

ID	Statement
US04	As a User, I want to retrieve specific client information from the system, so that I can use it for my work with the client.

Category	Functional Requirement
Interface	Searching provided. 
	Form provided.  Patient First Name: Hans Last Name: Kurz Birthdate: Sep 11, 2018 Social Secruity Number: 11111111111
Data Storage	Client information can be read.

ID	Test Case	Result
US04AC01	1. Open Application 2. Search for existing client 3. Click on target client form the search results 4. Assess if profile window opens.	OK

Table 37 US04

8.3 Automatic System Validation

The automatic system validation designed with selenium [7.2.6] validates the acceptance criteria's of the mandatory requirements [3.3] by following the defined test cases defined in the requirement assessment [8.2]. The implementation of the test is specified in the artefacts [7.5.3].

8.4 Project Objectives Assessment

Following assessment refers to the project objectives stated in [2.6].

Target ID	Assessment	Result
OBJ01	The MVP was defined within requirements [4] and the implementation plan [6].	Done
OBJ02	Clint data can be managed in limited fashion as defined in [8.2] not all defined features were implemented.	Partially
OBJ03	The solution contains an automatic system test. However it covers only best case function testing.	Done
OBJ04	As defined in the architecture [7.1] and developed artefacts [7.5]. the architecture is modular and parts can be replaced.	Done
OBJ05	Based on the requirement assessment [8.2] the MVP was not completely implemented and thus solution is not ready for a pilot project. However it can be used for demonstration purposes.	Partially

Table 38 Objective Assessment

8.5 Project Aim Assessment

This section compares the produced artefacts [7.5] with the project aims stated in [2.5].
The software developed within the project can be used as basic platform for further development and allows for demonstrations towards potential clients, however the implemented software does not include all features necessary for tracking patient as stated in [8.2]. Thus for a pilot project with a client the software is not ready yet as noted in the project objective assessment [8.4].

In order to bring the software into a state where H^{3}UMANA can use it for an pilot project with a client following actions would be executed:

- Gain client feedback on current software and its implementation.
- Add essential features to product backlog.
- Implement remaining stories for MVP. (Delete Patient)

9 Conclusion

The project at this state succeeded to develop a platform with responsive GUI on which demonstrations can be executed. However the aim for the minimal viable product was missed due wrongly estimated effort for the technology research [8.1]. Nevertheless the project succeeded in its major implementations as stated in [8.1].

The next steps of the project should be the completion of the MVP on which following missing parts can be identified:

- Deletion of Patient Data
- Missing Patient Data Fields → Address, Sickness etc.

9.1 Things Done Well

The project was able to assess and implement a variety of prior unknown technologies to the assignee. Such as: Neo4J & Driver, Selenium, AJAX, and JQuery. The state of the GUI is pleasing and it is designed responsive, overall the design of the GUI is in a well demonstrable state.

9.2 Things To Improve

The chosen methodology agile was not helpful for the development within this project. Due the experience gained within this project following statements about agile can be concluded:

- Agile requires to keep a constant pace of work, which is not feasible for projects executed with a setting where the assignee would work on it aside his/her daily duties.
- The rituals of agile become trivial & unnecessary if there is a team of one.

Overall the workload introduced by agile does not justify its de facto none existing benefits for such types of projects. Thus the waterfall or V-Model would have been more adequate choices.

10 References

[1] Samuel A. Marti, 2018. *H3UMANA Database - Project Background,* Prague: Prague College.
[10] Robin Hackshall, 2014. *Product Backlog Refinement.* [Online]
Available at: https://www.scrumalliance.org/community/articles/2014/october/product-backlog-refinement
[Accessed 3 September 2018].
[11] Product Owner , 2018. *What is a Product Owner?.* [Online]
Available at: https://www.scrum.org/resources/what-is-a-product-owner
[Accessed 3 September 2018].
[12] Margaret Rouse, 2015. *Story Point.* [Online]
Available at: https://whatis.techtarget.com/definition/story-point
[Accessed 3 September 2018].
[13] Codecademy, 2018. *MVC: Model, View, Controller.* [Online]
Available at: https://www.codecademy.com/articles/mvc
[Accessed 3 September 2018].
[14], 2018. *Computer Hope.* [Online]
Available at: https://www.computerhope.com/jargon/g/gui.htm
[Accessed 3 September 2018].
[15] Martin, Robter C., 2008. *Clean Code.* 1 ed. s.l.:Pearson Education (US).
[16] DB-ENGINES, 2018. *DB-Engines Ranking of Graph DBMS.* [Online]
Available at: https://db-engines.com/en/ranking/graph+dbms
[Accessed 11 July 2018].
[17] Wikipedia, 2018. *Graph database.* [Online]
Available at: https://en.wikipedia.org/wiki/Graph_database
[Accessed 11 July 2018].
[18] OrientDB, 2018. *OrientDB Manual - version 3.0.3.* [Online]
Available at: https://orientdb.com/docs/last/index.html
[Accessed 11 July 2018].
[19] Neo4j, 2018. *The Neo4j REST API Documentation v3.4.* [Online]
Available at: https://neo4j.com/docs/rest-docs/current/
[Accessed 11 July 2018].
[2] Cordova, Ing. Miguel Antonio, 2018. *Fortalecimiento de la Red de Servicios de Salud y Mejoramiento de la Calidad.* [Online]
Available at: http://www.salud.gob.ec/fortalecimiento-de-la-red-de-servicios-de-salud-y-mejoramiento-de-la-calidad/
[20] TitanDB, 2018. *Titan Graph Database 1.0.0 API.* [Online]
Available at: http://titan.thinkaurelius.com/javadoc/current/
[Accessed 11 July 2018].
[21] Neo4j, 2018. *Using Neo4j from PHP.* [Online]
Available at: https://neo4j.com/developer/php/#_neo4j_for_php_developers
[Accessed 11 July 2018].
[22] frankmayer, 2018. *arangodb-php.* [Online]
Available at: https://github.com/arangodb/arangodb-php
[Accessed 11 July 2018].
[23] PommeVerte, 2018. *gremlin-php.* [Online]
Available at: https://github.com/PommeVerte/gremlin-php
[Accessed 11 July 2018].
[24] JQuery, 2018. *JQuery.* [Online]
Available at: https://jquery.com/download/
[Accessed 4 September 2018].

[25] ikwattro, 2018. *GitHub.* [Online]
Available at: https://github.com/graphaware/neo4j-php-client
[Accessed 3 September 2018].
[26] MAMP, 2018. *MAMP & MAMP PRO.* [Online]
Available at: https://www.mamp.info/en/
[Accessed 3 September 2018].
[3] Mino, D. J. I. S., 2018. *Databases in Ecuador* [Interview] (23 February 2018).
[4] Pico, G. P. S., 2013. *BENEFICIOS DEL USO DE LA HISTORIA CLÍNICA ELECTRÓNICA EN,* Ambato: University of Ambato.
[5] Agile Alliance, 2018. *Minimum Viable Product.* [Online]
Available at:
https://www.agilealliance.org/glossary/mvp/#q=~(filters~(tags~(~'mvp))~searchTerm~'~sort~false~sortDirection~'asc~page~1)
[Accessed 3 September 2018].
[6] Jonathan Rasmusson, -. *What is Agile?.* [Online]
Available at: http://www.agilenutshell.com/
[Accessed 3 September 2018].
[7] Software Testing Help, 2018. *What is SDLC Waterfall Model.* [Online]
Available at: https://www.softwaretestinghelp.com/what-is-sdlc-waterfall-model/
[Accessed 3 September 2018].
[8] Andrew Powell-Morse, 2016. *V-Model: What Is It And How Do You Use It?.* [Online]
Available at: https://airbrake.io/blog/sdlc/v-model
[Accessed 3 September 2018].
[9] Margaret Rouse, -. *Search Software Quality.* [Online]
Available at: https://searchsoftwarequality.techtarget.com/definition/user-story
[Accessed 3 September 2018].
H^{3}UMANA, 2018. *Brainstrom Session, H^{3}UMANA Database Design* [Interview] (2 August 2018).

11 Figures

12 Tables