|  |
| --- |
| REPORT ON MANAGEMENT SYSTEM FOR  “ACADEMY OF THE HANDSOME MEN AND BEAUTIFUL WOMAN(BEA) “WEB APPLICATION  **Advisor:** *Prof. Dr. Olivier Biberstein*  **Author:** *Kristina Shiryagina*  **Version and date** |
| **Bern University of Applied Sciences**  Department  Division |

Contents

[1 Vision and scope 3](#_Toc22513250)

[2 Analysis 5](#_Toc22513251)

[3 Design 1 9](#_Toc22513252)

[4 Implementation 9](#_Toc22513253)

[5 Testing 9](#_Toc22513254)

[6 Conclusion 9](#_Toc22513255)

[7 Glossary 9](#_Toc22513256)

[8 Bibliography 10](#_Toc22513257)

[9 Version control 10](#_Toc22513258)

[10 Project planning 10](#_Toc22513259)

[11 Protocol 11](#_Toc22513260)

# Vision and scope

## Project Description

As the head of information system for Online academy we are tasked with developing a part of new Online Management System. As the idea of online education is getting more popular day by day.

The proposed software product (Online Beauty academy) is an online education system. The system will be used for online-education, to download lectures, conducting online quizzes, course registration, exam reservation, managing results. The system must be right protected.

The online academy will have different online- courses. The are beauty-course , beauty-instructor course and other future courses. Each course will have topics and lessons for each topic. The context of lessons are text, video-tutorials, etc.

A Participant can choose one of the courses and follow the program of this course. If participant want to get a certificate he shall do exams.

Online Exams.

This application will establish a network between the lecturers and participants. Academia enter the questions they want in the exam. These questions are displayed as a test to the eligible participant. The answers enter by the participant are then evaluated and their score is calculated and saved. This score then can be accessed by the Academy to evaluate the performance of participants.

Exam details.

Each topic will have small exam(quizzes). Making exams the participant will collect a points. The sum of all point for all small exams are 30% of final grade. After making all small exams participant will be able to make a final exam that has weight 70% of final grade. The participant must make a reservation for final interactive exam with lecturers. He has to prenote the available Date for this exam. ´

The application has an administrator who keeps an eye on the overall functioning of the system.

## Stakeholder Description

Similar to other technology applications, the success of online-learning is dependent on the extent to which it satisfies the needs and addresses the concerns of its stakeholders.

The **list of stakeholders**

Participant: use the system to register for the course or exam, view information.

Instructors: they could give ideas on the solution for the system’s development and improvement.

Administrator: manage the system after it is built.

Education Institutions

Content Providers

Development team: include all software engineers, business analysts, system analysts, system designers, implementers, testers, QA, and project management. They are tasked to build the system.

Employers

## Vision of Solution

### Problem Statement

Now academies are running various programs as full time courses. The academy timing sometimes make it difficult to study for person who are doing some jobs. The online education would help such person who live far away from education institutes.

The other problem is that the current online management system doesn't have the needed flexibility and is not modern enough. The capabilities are limited.

Online management system is effective, reduce time and cost in courses and exam management process.

### **Project Scope**

We going to build just a part of the academy management system. We have to pay attention to building applications supporting: Participant will be able to register, to enrol and manage their courses, to make a reservation for a final exam.

### Summary of **System Feature**

* Online registration.
* Log in.
* Manage user information.
* Manage Offering Courses.
* Manage Lecturer information.
* Course registration.
* Exam registration.

### **List of features** will not be developed

The features which are currently out of scope and may be added later on as an extension

to the application are:

* Access the system as lecturer
* Manage lecturer Information.
* Manage Financial Activities.
* Uploading course content.
* Course evaluation.
* Downloading course content.
* Video conferencing.
* Info service.
* Information library.

## Software development methodology

The establishment and use of sound engineering principles in order to obtain economically

developed software that is reliable and works efficiently on real machines is called *software*

*engineering.*

***Software engineering*** is the discipline whose aim is:

1. Production of quality software

2. software that is delivered on time

3. cost within the budget

4. satisfies all requirements.

**Software process** is the way in which we produce the software. Apart from hiring smart,

knowledgeable engineers and buying the latest development tools, effective software

development process is also needed, so that engineers can systematically use the best technical

and managerial practices to successfully complete their projects.

A **software life cycle** is the series of identifiable stages that a software product undergoes during

its lifetime .A software lifecycle model is a descriptive and diagrammatic representation of the

software life cycle .A life cycle model represents all the activities required to make a software

product transit through its lifecycle phases .It also captures the order in which these activities are

to be taken .

***Life Cycle Models***

There are various life cycle models to improve the software processes. And we have used the WATERFALL MODEL.



This model contains 6 phases:

o **Feasibility study**

The feasibility study activity involves the analysis of the problem and

collection of the relevant information relating to the product. The main aim

of the feasibility study is to determine whether it would be financially and

technically feasible to develop the product.

o **Requirement analysis and specification**

The goal of this phase is to understand the exact requirements of the

customer and to document them properly.

o **Design**

The goal of this phase is to transform the requirement specification into a

structure that is suitable for implementation in some programming language.

o **Implementation and unit testing**

During this phase the design is implemented. Initially small modules are

tested in isolation from rest of the software product.

o **Integration and system testing**

In this all the modules are integrated and then tested altogether.

o **Operation and maintenance**.

Release of software inaugurates the operation and life cycle phase of the

operation.

## Agile development, Scrum.

Scrum was in the center of developing process.

As scrum projects make progress in a series of “sprints”, we have divided the whole process into

4 sprints. Product was first analysed, designed , then code and tested during the sprint.

Scrum

Roles

* Scrum Master – *Prof.Dr. Olivier Biberstein*
* Developer – Kristina Shiryagina

Artifacts

* Product Backlog
* Sprint Backlog

Meetings have included :

* Product/release planning
* Sprint planning
* Weekly Scrum
* Sprint review
* Sprint retrospective

Scrum Meetings

Each meeting between developer and scrum master was made of several steps, that were repeated each time:

* Attendance: all
* Product Owner presents Product Backlog

with all relevant user stories with their priority

* Discussions and clarifications if needed
* Results:

Prioritized Product Backlog

* Specifies what to build
* Final decision by the Master
* Vision, high level architecture, most important non-functional

requirements

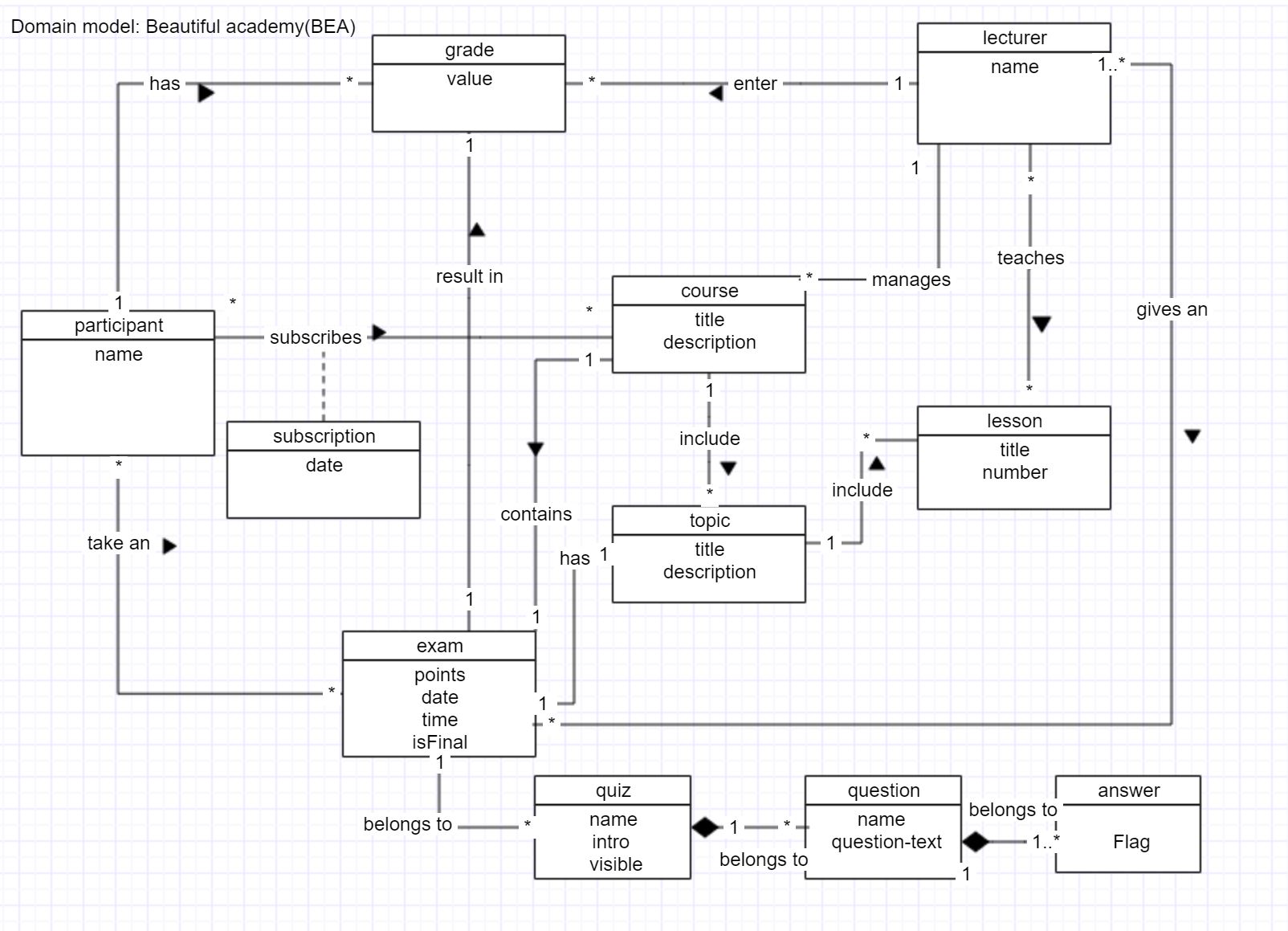
Release planning (if product is to be delivered in releases):

* Select and prioritize items of Product Backlog for the next Release Backlog

# Analysis

## Domain model

This document describes the domain model of the Beautiful Academy BEA. It introduces the most important concepts and the associations among them. It also introduces the respective multiplicities.



## Concept Classes

Concept class Participant models a person who is taking the courses.

Concept class Lecturer models a person who is teaching and give an exam for the participant.

Concept class Course models the main courses that offer application.

Concept class Grade models the grades of a participant.

Concept class Topic models the sub course of the course.

Concept class Lesson models a set of lessons that contains each topic.

Concept class Exam models a exam that can be taken by participant.

Concept class Subscription models a subscription, that has a date when participant is subscribed.

Concept class Quiz models a quiz that belongs to exam.

Concept class Question models a question that belong to quiz.

Concept class Answer models a possible answer to questions.

## Associations

Association take an between **Participant** and **Exam** denotes the fact that Participant can make many exams, hence (\*) multiplicity, and (\*) multiplicity at the Exam side means that the Exam can be done by many Participants.

Association has between **Participant** and **Evaluation** denotes the fact that Participant can get zero or more Evaluations, hence multiplicity (0..\*), and (1 )multiplicity at the Participant side means that each Participant has its own set of Evaluation.

Association subscribes between **Participant** and **Course** denotes the fact that that zero or more Participant can be entered to the Course depending of the number of participant, hence multiplicity(\*), and each Participant can be subscribed to zero more courses, hence multiplicity(\*)

Association result in between **Exam** and **Evaluation** denotes the fact that each exam has it’s own unique evaluation, hence multiplicity 1, and 1 multiplicity at the evaluation side means that each evaluation belongs to unique exam.

Association has between **Exam** and **Topic** denotes the fact that each topic has it’s unique an intermediate exam.

Association contains between **Exam** and **Course** denotes the fact that each exam belongs to it unique course , and each course has it own unique final exam.

Association include between **Course** and **Topic** denotes the fact that course has many topics, hence multiplicity (\*), and each topic belongs to exactly one course, hence multiplicity (\*).

Association include between **Topic** and **Lesson** denotes the fact that topic has many lessons, hence multiplicity (\*), and each lesson belongs to exactly one topic, hence multiplicity (\*).

Association teaches between **Lecturer** and **Lesson** denote the fact that Lecturer can teaches zero or more lessons, hence multiplicity (\*), and each lesson can have zero or more lecturer, hence multiplicity (\*).

Association gives an between **Lecturer** and **Exam** denotes the fact that Lecturer can give zero or more exam, hence multiplicity (\*), and each exam can be done by 1 or more lecturer, hence multiplicity ( 1..\*).

Association enter between the **Lecturer** and **Evaluation** denotes the fact that each lecturer can enter zero or more evaluations, hence multiplicity (\*), and evaluation can be entered by exactly one lecturer, hence multiplicity (1).

Association manages between the **Lecturer** and **Course** denotes the fact that each lecturer can manage zero or more courses, hence multiplicity (\*), and each course can be manage by exactly one lecturer, hence multiplicity (1).

Association belongs to between **Quiz** and **Exam** denotes the fact that each Quiz belongs to exactly one exam, hence multiplicity (1), and each exam can have zero or more quizzes, hence multiplicity(0..\*).

Association belongs to between **Quiz** and **Question** denotes the fact that each Question belongs to exactly one Quiz, hence multiplicity (1), and each Quiz can have zero or more questions, hence multiplicity(0..\*).

Association belongs to between **Question** and **Answers** denotes the fact that each Answer belongs to exactly one Question, hence multiplicity (1), and each exam can have one or more answer, hence multiplicity(1..\*).

## Product Backlog

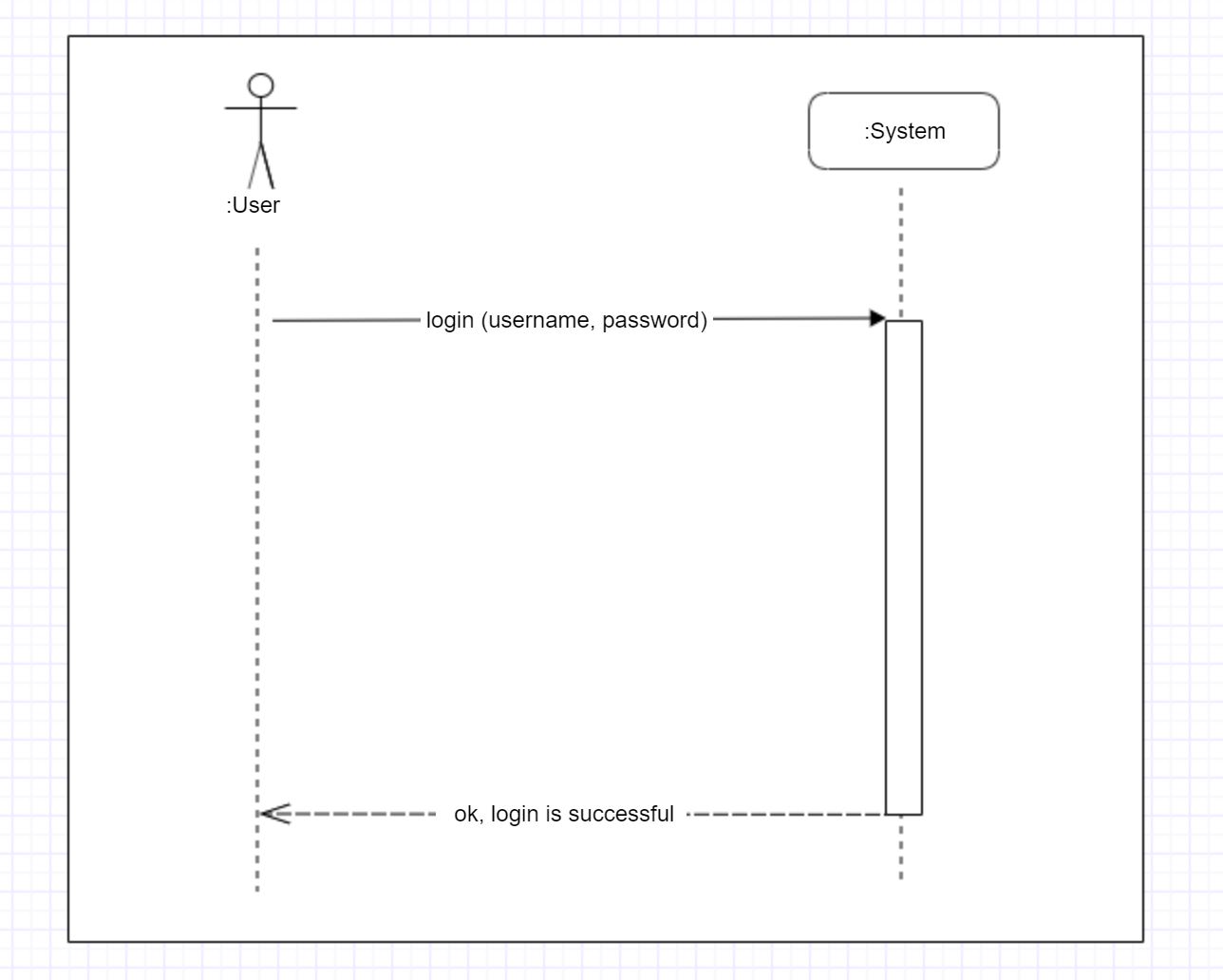
The product backlog is a list of user stories which is used to implement the product vision. It is sorted according to the priority of the user stories according to the product owner. The priority of these stories will be modified during the process of the project.

|  |  |  |
| --- | --- | --- |
| **Id** | **Story name** | **Story description** |
| **01** | Log in | **As a user, I want to be able to login into the system with my credential (username, password).**  Success: The user is logged in and can use the functionality of the system.  Failure: An error message is displayed: “Wrong username or password, please try again!”. |
| **02** | List courses | **As a Participant I want to list courses I'm entitled to subscribe such that I can perform a subscription**. |
| **03** | View information | **As a Participant, I want to see the information about the courses.** |
| **04** | List results | **As a Participant, I want to see my marks of the already finished courses and topics.** |
| **05** | View schedule | **As a Participant or a Lecturer I want to be able to get a schedule showing the time and place of available exams.** |
| **06** | Exam reservation | **As a Participant, I want to be able to select an exam (there are different data on an exam).**  Success: A participant has selected exam.  Failure: An error message is displayed “This date is already reserved, please take another date for your final exam". |
| **07** | Cancelation of exam | **As a Participant, I want to be able to cancel the exam registration.**  Success: A message “You have successfully deleted your exam registration”  Failure: An error message "The period of availability of deleting registration is expired, please take a contact to the administration". |
| **08** | Personal information | **As a Participant or a lecturer I want to see my page with my personal information** .  Success: The participant can see his page with all the information it has.  Failure: We are sorry, this page is on reconstruction , you can access it after 12 hours. |
| **09** | Manage course | **As a System Administrator, I want to be able to change the information on a course .**  Success: An Administrator can manage the data of courses, participants, exams.  Failure: |
| **10** | List participants and corresponding courses | **As a lecturer, I want to see a list of participants with courses that they have .**  Success: A Lecturer can see the list with all the data he needs.  Failure: |

# Design 1

## System Sequence Diagrams

This document describes the (initial) set of the System Sequence Diagrams for BEA.



\* 1. Use case "**Log in**"

Figure 1 – ssd for login

## SSD for login.

# 

# 

Figure 2– sd for spring security » login «

Figure 2 – sd for spring security « login «

# 4.Implementation

## BUILD Backend WITH SPRING BOOT

Before starting with spring boot, it was clear for me about spring boot that spring boot does not provides any extra features on functionality on top of spring framework. Rather, it provides unlimited defaults configurations and useful conventions to create a stand-lone, production grade web applications in no time.

Spring Boot is the starting point for building all Spring-based applications. Spring Boot is designed to get you up and running as quickly as possible, with minimal upfront configuration of Spring.

-Get started in seconds using Spring Initializer

-Build anything: REST API, WebSocket, web, streaming, tasks, and more

-Simplified security

-Rich support for SQL and NoSQL

-Embedded runtime support: Tomcat, Jetty, and Undertow

-Developer productivity tools such as Live Reload and Auto Restart

-Curated dependencies that just work

-Production-ready features such as tracing, metrics, and health status

-Works in your favourite IDE: Spring Tool Suite, IntelliJ IDEA, and NetBeans

Using these features it has really made building a production grade Spring applications very easy and faster for developers. Also, no XML configurations required anymore with spring boot.

## ANGULAR for front-end

The technology for front-end is Angular.

Angular helps build interactive and dynamic single page applications (SPAs) with its compelling features including templating, two-way binding, modularization, RESTful API handling, dependency injection, and AJAX handling. We can use HTML as template language and even extend HTML’ syntax to easily convey the components of the application.

Angular applications are built using TypeScript language, a superscript for JavaScript, which ensures higher security as it supports types (primitives, interfaces, etc.). It helps catch and eliminate errors early when writing the code or performing maintenance tasks.

Angular has a lot of pros

-**simplicity**

**-efficiency**

- Developers find AngularJS very effective especially in creating **dynamic**, single page apps, and supporting MVC (Model View Controller) programming structure.

-**time-saving** Projects that previously used to take many months with other frameworks can now be completed faster with AngularJS. All that AngularJS framework requires is splitting the app into several MVC components. From there, the framework takes over because you do not require additional coding.

-the app is **easy to learn** and get started.

-**data binding** in AngularJS is **very easy**

High Level Architecture

Figure 2 – Architecture

Angular Application(TypeScript

RESTFUL API (Spring boot on Java)

SQL

Front-end application talk to back-end API (or web service). II will use a framework angular with tool angular CLI which is used to build front-end application. Angular is one of the most popular frameworks to build front-end.

Angular has a variety of features like components, modules, forms HTTP communication to talk with web services it makes it easy to develop front-end applications. Angular uses a language called typescript.

To build a RESTful API or the restful web services in the backend we will be using the spring boot framework, we will be using Java as the language. Spring boot is a one of the best Java frameworks to build restful api and micro services.

We will use multiple restful services for authentication as well as to do management as far as authentication is concerned, we will start with basic authentication and we will use JW or JSON web tokens for authentication. We will be using the spring security framework to build authentication.

We will be building separated back-end and front-end.

The most important reason is to create a flexible architecture which can be extended to meet the future needs. For example, if we will develop a native mobile application. We can reuse the back-end API

EXAMPLE

Figure 3 – Possible extension of application, re-use the backend

Angular Application(TypeScript)

RESTFUL API (Spring boot on Java)

SQL

Mobile

IOT

## Spring boot microservices

Microservices it is a different way of building application.

The distinction between what we do on the coding side of things where we can have multiple projects , we can have all kind of modularity but still end up having a monolith deployed on the server. Microservices that thing changes ,

We can follow the same patterns with coding but then what’s happening at runtime is very different .

There are a lot of things to do with microservices. We going to have microservices talk to each other.

We have like a big monolith and we split it into smaller stuff.

There are different aspects to having microservices stock.

We will creating a couple of microservices, that going to work together to form this application.

We will have like a big monolith and we will split it up into smaller stuff, and we will make them communicate with each other.

So we will do:

* Create a microservices
* Make them communicate to each other

Why there is complexity, why do we have so many technologies involved . We have one monolith, one big chunk of code and we breaking it out . We solving scalability, modularity of deployment , we can make changes to one small portion of the application without having to reapply the whole thing .

We will use spring cloud , because it is one of the most robust ways of building micro services.

### Prerequisites

* Java
* Spring boot

Each microservice is going to be a spring boot application.

# Testing

# Conclusion

// In the conclusion of the project, you first summarize the most important results of your project in an understandable way. The main thing here is to filter out the essential content from the previous chapters and get to the point in a compact, clear and concise way.

## Summary

// Summary of the main results of your thesis

## Interpretation

//Interpretation of my results

//. In the theoretical part of your project, you should research and present the current state of research in detail in order to draw the right conclusions from the conclusion of your project. Even if your question could not be answered satisfactorily, you should present it honestly. Check what you could and could not achieve, what you would do differently in retrospect and which methods have proven to be useful.

## Outlook

//Outlook of my project and future research recommendations

# Glossary

**Auinweon**

Et ut aut isti repuditis qui ium 7

**Batnwpe**

Et ut aut isti repuditis qui ium 9

**Cowoll**

Et ut aut isti repuditis qui ium 11

# Bibliography

**Literary Entry**

*Prof.Dr. Olivier Biberstein, Prof.Dr Eric Dubuis , Software Engineering and Design* 7

**Literary Entry**

*Prof.Dr. Olivier Biberstein, Prof.Dr Eric Dubuis, Software Engineering and Design* 9

**Literary Entry** 9

**Literary Entry**

*Spring.io/guides, 2019* 11

# Version control

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Date** | **Description** | **Author** |
| 0.1 | 30.09.2019 |  | Kristina Shiryagina |
| 0.2 | 13.03.2013 | Et ut aut isti | Kristina Shiryagina |
| 1.0 | 21.05.2013 | Et ut aut isti | Kristina Shiryagina |

# Project planning

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Sprint plan | | | | | | |  |  |  |  |  |
|  | | | | | | |  |  |  |  |  |
| Meetings calendar Meetings Date Memo   |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | |  |  |  |  |  |  |  |  |  |  | | **# sprint** | Oct-19 |  |  |  |  |  |  |  |  | |  | Mo | Tu | We | Th | Fr | Sa | Su | time | Notes | |  |  |  | 2 |  |  |  |  | 14:30-15:10 |  | |  |  |  | 9 |  |  |  |  | 14:30-15:10 |  | |  |  |  | 16 |  |  |  |  | 14:30-15:10 |  | |  |  |  | 23 |  |  |  |  | 14:30-15:25 |  | |  |  |  | 30 |  |  |  |  | 14:30-15:10 |  | |  | Nov-19 |  |  |  |  |  |  |  |  | |  |  |  | 6 |  |  |  |  | 14:30-15:10 |  | |  |  |  | 13 |  |  |  |  | 14:30-15:10 |  | |  |  |  | 20 |  |  |  |  |  |  | |  |  |  | 27 |  |  |  |  | 14:30 | Next meeting | |  | Dec-19 |  |  |  |  |  |  |  |  | |  |  |  | 4 |  |  |  |  |  |  | |  |  |  | 11 |  |  |  |  |  |  | |  |  |  | 18 |  |  |  |  |  |  | | | | | | | |  |  |  |  |  |
| Sprint Backlog of Sprint 1    |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | **Id** | **Story Name** | **Story Description** | **Priority** | **Estimated Effort** | **Actual**  **Effort** | **Status** | | **0** | Domain model | Domain model | High |  |  | done | | **1** | Documentation | Initial structure of main document. | High |  |  | done | | **2** | User stories | Write some user stories. | High |  |  | done | | **4** | Vision | Make Vision of project. | High |  |  | done | | **5** | Infrastructure | Set up infrastructure Gitlab | High |  | 0.5h | done | | **6** | User Stories | Define user stories and enter them to the main document. | High |  |  | done | | **7** | User Stories | Extend user stories with "success" and "failure" | High |  |  | in process | | **8** | Infrastructure | Sprint Backlog. Make a document with sprint planning. Divide work in 14 weeks . | High |  |  | done | | **9** | Documentation | Extend documentation with explanation about Angular and Spring. "What is it Spring, Angular …." |  |  |  | Almost  done | | **10** | Documentation | Make clear vision document and put it to the main document | High |  |  | done | |  |  |  |  |  |  |  | |  |  |  |  |  |
|  |  |  |  |  |  |
|  | | | | | | |  |  |  |  |  |
| Sprint Backlog of Sprint 2  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | **Id** | **Story Name** | **Story Description** | **Priority** | **Estimated Effort** | **Actual**  **Effort** | **Status** | | **11** | Postgres SQL | Add postgress SQL to the project, make connection | High |  |  | done | | **12** | diagram | System sequence diagram for login | High |  |  | done | | **13** | User  Story | Implement login | High |  |  | done | |  | diagram | System diagram for spring security | High |  |  | done | | | | | | | |  |  |  |  |  |
| Sprint Backlog of Sprint 3  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | **Id** | **Story Name** | **Story Description** | **Priority** | **Estimated Effort** | **Actual**  **Effort** | **Status** | | **15** | Diagram | Make diagram for register | Medium |  |  |  | | **14** | Domain Classes | Develop sketch of working domain classes | High |  |  |  | | **15** | Unit tests | Develop unit tests for domain model | High |  |  |  | | **16** | Javadoc | Ajust test classes | High |  |  |  | | **17** | Repositories | Add repositories |  |  |  |  | | **18** | Documentations | Change documentation format to latex | medium |  |  |  | |  |  |  |  |  |  |  | |  |  |  |  |  |  |  | | | | | | | |  |  |  |  |  |
| Sprint Backlog of Sprint 4  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | **Id** | **Story Name** | **Story Description** | **Priority** | **Estimated Effort** | **Actual**  **Effort** | **Status** | |  |  |  |  |  |  |  | |  |  |  |  |  |  |  | |  |  |  |  |  |  |  | |  |  |  |  |  |  |  | |  |  |  |  |  |  |  | |  |  |  |  |  |  |  | |  |  |  |  |  |  |  | | | | | | | |  |  |  |  |  |
|  | | | | | | |  |  |  |  |  |
|  | | | | | | |  |  |  |  |  |
|  | | | | | | |  |  |  |  |  |

# Protocol

**Frequency: (weekly)**

**Meeting length: (35-45 minutes)**

Agenda

* **Demo and Discuss Deliverables (Demo)**
* **Planning next Goals (Plan)**
* **Lessons learned (Lessons)**
* **Date, time of the next meeting (next meeting)**

**Report from 09.10.19**

**Plan**

Next goals are:

1. Introduction of a Vision make clear. Write about an application I want to build. I have to write a Vision that can make a good picture about the functionality of this application.
2. Change the problem statement:

* Join 2 Systems in 1 . Rename system in functions. And write that these functions just a part of this system we want to build.
* Write concrete user stories to these functions, group the stories according the function.

1. Analyze domain model, put attributes to each conceptual class, make description for each association. Rebuild domain model according of new clear representation of necessary functions of the system.
2. Sequence diagram of first function we want to implement (probably log in)
3. Try to make class-diagram.
4. Implement of log in function.
5. Write a protocol in the main doc.

**Lessons learned**

To documentation plays decisive role in Software Engineering Project. The first analysing phase have to be done well to make a good start for design and implementation of the IT Product.

**Next Meeting:** 16.10.19, 14:30

Report from 16.10.19

**Plan:**

The next goals are:

1. A domain model – last version. +
2. Vision complete. +
3. Put spring plan in the main document (Time planning is the first preference).
4. After completing 3d point, merge sprint1 and make a del1.
5. Start with Design part: 1st we start with design for login feature.+
6. SSD for login +
7. Write good document for Spring Boot and Angular. Describe the most successful aspects,that has spring boot , angular. Describe why we have chosen it for this project. Describe how can be implement login with spring boot. +
8. Make a Product Backlog( list of user stories) and divide to 4 sprints. +
9. Point 1 and 2 have highest priority. Just when these 2 points successfully completed I will continue my to-dos. +
10. Change the style of document, make headers numerable, and the style more readable. +

**Lesson Learned**

With this practical work I become always clearer the main principles of building software.

**Next meeting:** 23.10.19, 14:30

Report from 23.10.19

**Plan:**

1. Make Vision clear( Describe how people will use this application. Describe what is it exactly the course(content, PDF, Unterricht) The vision needs to be loner ,about 1 A4 page.

Vision-Solution belongs to Vision, here I have to write what we will produce.

1. Domain model. Change word entity to concept by description.

Add two concepts: subscription with a Date attribute .

Add to exam concept boolean , that will show the difference between final and intermediate exam.

1. Change product backlog (failure and success)
2. Make ssd for login with all needed classes( show exactly what’s happened in system ).
3. Implement login function, test it. Have to be able to make a demo.
4. After making all 5 points and if I’ll have more time make ssd for other features.

**Lesson learned:**

More experience about software engineering diagrams

**Next meeting:** 30.10.19, 14:30

Report from 30.10.19

**Plan:**

1. Start frontend with Angular. Make frontend for login and registration.

**Lesson learned:**

**Next meeting:** 06.11.19, 14:30

Report from 06.11.19

**Plan:**

1. Documentation

* Most important thing in a documentation is to write it so that the reader can easily understand the main scope of project, how it will be realised, etc.
* Angular and spring . Make description of angular and spring , reader have to understand for what we use exactly this frameworks. (have to notice in bibliography everything I use from other authors)
* Make a build of architecture of angular and spring( probably layers build)
* Vision more
* User-stories
* Agile and scrum -\/

1. Make registration with via email address. (read about registration via facebook, ect if I have more time)
2. Start to learn about microservices. Try to add first documentation about it.

**Lesson learned:**

Today I’ve learned that I have to plan my work better, means that first I have to realise the most difficult and most important tasks, and then make tasks less difficult.

**Next meeting:** 13.11.19, 14:30

Report from 13.11.19

**Plan:**

1. Documentation:

Describe:

What is it monolith architecture, what is different from monolith and microservices?

What is restful? (in layer diagram)

1. Code

Make registration via email. Start other features. This step I have to do just when I have finished the documentation.

For next time be able to make a demo of what we have.

**Lesson learned:**

Be able to define clear the features I want to realise in this project.

**Next meeting: 27.11.19, 14:30**