**Project Plan**

***CLOTHINGSTORE INC.***

|  |
| --- |
| **Date : 01.03.2023** |
| **Version : 0.1** |
| **State : In progress** |
| **Author : Andrei Pieleanu** |

Contents

[Version history 3](#_Toc128559757)

[Distribution 3](#_Toc128559758)

[Project assignment 3](#_Toc128559759)

[Context 3](#_Toc128559760)

[Goal of the project 3](#_Toc128559761)

[Scope and preconditions 4](#_Toc128559762)

[Types of users 4](#_Toc128559763)

[Functionalities 4](#_Toc128559764)

[Strategy 5](#_Toc128559765)

[Research questions and methodology 5](#_Toc128559766)

[End products 6](#_Toc128559767)

[Project organisation 6](#_Toc128559768)

[Stakeholders and team members 6](#_Toc128559769)

[Communication 6](#_Toc128559770)

[Activities and time plan 7](#_Toc128559771)

[Phases of the project 7](#_Toc128559772)

[Time plan and milestones 8](#_Toc128559773)

[Testing strategy and configuration management 12](#_Toc128559774)

[Testing strategy 13](#_Toc128559775)

[Test environment and required resources. 13](#_Toc128559776)

[Configuration management 13](#_Toc128559777)

[Finances and risk 14](#_Toc128559778)

[Project budget 14](#_Toc128559779)

[Risk and mitigation 14](#_Toc128559780)

# Version history

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Version** | | **Date** | **Author(s)** | **Changes** | **State** |
| 0.1 | 01.03.2023 | | Andrei Pieleanu | Updated project plan & UML | Completed |
| 0.2 | 15.03.2023 | | Andrei Pieleanu | Added product backlog & C4 architecture | Completed |
| 0.3 | 29.05.2023 | | Andrei Pieleanu | Finished working on all proposed features | Completed |

# Distribution

|  |  |  |
| --- | --- | --- |
| **Version** | **Date** | **Receivers** |
|  |  |  |
|  |  |  |

# Project assignment

## Context

*ClothingStore Inc. is a business startup which is specialized in selling clothing objects.*

## Goal of the project

*As a company, ClothingStore Inc. would like a website from where the customers can have access to a wide variety of clothing items to buy from. Also, the workers of the company would like to work with an integrated system for administrating the clothes and the orders made by the clients. This will greatly facilitate the workflow and the employees can easily have access to the managing system.*

## Scope and preconditions

End products

* A fully functional, user-friendly application
* A system-integrated database
* Project plan
* Wireframes
* Test plan
* Test report
* User Requirements Specifications

### Types of users

#### Admins

Admins are able to perform CRUD on the list of clothes in the store (adding clothes must have an initial amount in stock). Products that are out of stock will have the ability to be restocked. They are able to see statistics regarding the purchases for a period of time. They are able to set up certain discounts and/or exclusive offers for products if needed. Admins have access to a variety of filters for clothes.

#### Clients

Clients have access to the shop’s wide variety of clothing and are able to add any clothes as they please. They will only be able to see the available clothes and add products to their cart. They can checkout the cart, choose an address and payment method and make an order. Clients can also see their orders and edit their delivery addresses. They too also have access to a variety of filters.

### Functionalities

Here, I will present the functionalities that the website will have:

#### Website

##### Admins

* Log in
* Log out
* Perform CRUD on the list of clothes
* Add & remove discounts/offers for an item
* Restock any item
* See general statistics
* Filter items
* Search through orders

##### Clients

* Log in
* Log out
* Register
* CRUD products in the shopping cart
* CRUD his own addresses
* See their orders
* Filter items

## Strategy

*For this project, I have opted for the agile-scrum methodology, because it is a bulky project, which constantly requires feedback from the teachers and redesigning. For such a big project, we don’t exactly know all the requirements, so we will have to start from scratch.*

## Research questions and methodology

1. *What do the users want to achieve by using this app?* 
   1. *What are the requirements of the client?*
   2. *What are the requirements of the shop worker?*
2. *What suitable technology can be used for this project?* 
   1. *What kind of software architecture will be used for the project?*
   2. *What technology will be used for this project?*
   3. *What programming languages will be used for this project?*
3. *What are the alternatives for the frontend?* 
   1. *What frontend technologies will be used for this project?*
4. *What are the alternatives for the backend?*
   1. *What backend technologies will be used for this project?*

## End products



# Project organisation

## Stakeholders and team members

*This project is being handled by Andrei Pieleanu. I will have a meeting with my teachers multiple times per week. The teachers are Maja Pesic and Faruk Aydin.*

*The stakeholders for this project will be represented by the employees and the clients of the ClothingStore Inc.*

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Abbreviation** | **Role and functions** | **Availability** |
| *Andrei Pieleanu* | *A.* | *Developer* | *5 days per week. Weekend also included if there’s something critical to be made. 5-6 hours per day should be enough.* |

## Communication

*The meetings with the teachers will be done in person on the 4th floor or on Microsoft teams if in person is not possible. This project will be worked across the span of 18 weeks. This project plan is to show what will be done in those 18 weeks.*

# Activities and time plan

## Phases of the project

### Sprint 1

* *Project plan*
* *Potential wireframes*
* *Initial product backlog with enough tasks and/or user stories refined for the next sprint (including acceptance criteria, estimation and prioritization)*
* *Asking feedback from the teachers*
* *Backend: first setup of RESTful API for your individual track project*
  + *Layering, dependency inversion and injection using Spring*
    - *at least 3 layers: controller --> business --> persistence (data access)*
    - *focus on separation of concerns and testability*
    - *persistence layer classes can't be returned to controller layer (directly or indirectly)*
  + *3 working RESTful services*
  + *One exclusive GIT repo for BE (FE one will come later)*
* *CI/CD environment initialization*
  + *build and test stages (a few tests ok for now)*

### Sprint 2

* *Design document version 1*
  + *Architecture constraints and design decisions (Why are you using Spring Boot? Or React? Or MySQL?)*
  + *C4 Model diagrams with context/explanations for level 1 (system context), 2 (container) and 3 (component).*
    - *Level 3 diagram should be meaningful for the architecture. Suggestion: represent your layers as components of the container.*
    - *Connect explanations with SOLID or other software principles (like KISS, DRY or YAGNI) when applicable.*
* *Backend*
  + *CORS configuration in-place for controllers*
* *Initial Frontend setup*
  + *One exclusive GIT repo for FE*
  + *At least one BE service being consumed from FE app*
  + *Layering: API access should have its own layer*
* *Initial Applied research document*
  + *Topic should be related to your project (individual or group)*
  + *Main/sub questions approved by tech teachers*
  + *Separate document*

### Sprint 3

* *Design document version 2*
  + *Level 4 (Code) diagram with context explanations.*
    - *UML Class diagram*
    - *Should be meaningful for the architecture. For instance, class diagram explaining layering implementation with dependency inversion.*
    - *Connect explanations with SOLID or other software principles (like KISS, DRY or YAGNI) when applicable.*
* *Initial Backend to Database setup*
  + *Replacing fake repositories for real ones*
  + *Unit testing business layer mocking the persistence layer with Mockito (a few tests for now is ok)*
  + *Database versioning with Flyway use is recommended, but optional*
  + *Partial integration tests on the persistence layer are recommended, but optional.*
* *SonarQube*
  + *Installed and running*
  + *Plugged into pipelines in a new stage after tests*
  + *It’s ok allowing this stage to fail in the pipelines. It might take time to get the quality gate passing, there’s no need to force it.*
  + *Take a screenshot of your project in Sonarqube and also for the new GitLab pipeline*
* *Research document*
  + *Problem statement or opportunity*
  + *Main question*
  + *Sub questions*
  + *(Multiple) Chosen methods per sub question and reasoning for these choices*
  + *Results*
  + *Conclusion and recommendation*
  + *References and citations in APA style*

### Sprint 4

* *Design document version 3*
  + *Include a diagram of how your CI is setup with context explanations*
    - *should include all the nodes involved in the pipeline execution (like developer's machine, GitLab server, pipeline runner and Sonarqube instance)*
    - *draw your own diagram (using for instance, app.diagrams.net or PowerPoint)*
  + *Make sure your architecture diagram matches your backend.*
* *Authentication and authorization implementation*
  + *Backend:* 
    - *Login service*
      * *Needs to be JWT based*
      * *Tokens should have expiration time and be signed*
      * *Besides applying it, you should also remember and understand what a JWT is and why it's important that it expires and is digitally signed*
      * *You should remember what happens when a token expires*
      * *You should remember how to re-authenticate a user without forcing him/her to provide credentials again*
      * *Is it safe to send tokens over HTTP? Or do you need HTTPS to protect them in transit? Why? (a HTTPS implementation is not required)*
      * *Access token should have a claim for the user roles*
      * *Refresh tokens is a plus, not required at all*
    - *Role-based authorization on two other services*
      * *for instance, one service only for admin users and another one for "normal" users*
  + *Frontend* 
    - *Login feature implemented and connected to backend*
    - *Storing token in-memory, session storage or local storage*
    - *You should remember the trade-offs involved when deciding where to store the token in the FE*
    - *One feature at least calling a secured service in the backend*
  + *Oauth/OpenID Connect is a plus, not required at all*
* *Continuous Integration and Sonarqube*
  + *All tests passing*
  + *Business layer should be totally or mostly covered by unit tests (> 80%)*
  + *If your quality gate is not passing, you should be aware of the technical debt. Links to an external site. When do you plan to address it? Are other tasks more important right now?*
  + *Take a screenshot of your latest Sonarqube results*

### Sprint 5

* *Final design document*
* *Security report detailing how your application deals (or does not deal) with the OWASP top 10 security risks.*
  + *For the review session, two important cases to self-assess and discuss (of course, your teacher can discuss others too if he/she wants):*
    - *[Broken-access Control] We have two users, A and B, with the same role, CUSTOMER. Can user A get or change data that belongs to user B in your application? Why? Please show in the backend’s code how it works.*
    - *[Injection] If a user types a malicious SQL statement in your application, will that query actually execute in the backend? Why?*
* *Websockets feature*
  + *Simple feature (like chat or notification) related to App*
  + *No security needed*
  + *UX should match the needs of your App*
* *Minimum viable product (MVP) features implemented. We want to see a demo of your system. It is mandatory to submit a "lastName\_firstName\_demo.zip" file (directly on Canvas, not in the repository) which contains:*
  + *A document (1 page) with a list of all planned user stories - show a simple list with user story name and status ("finished", "in progress", or "to do").*
  + *For each user story which is (almost) finished including a video recording which shows a demo of that user story. (\*Note: If you do not submit a video of "User Story X", that means that "User Story X" is not implemented yet.)*
* *Continuous Integration and Sonarqube*
  + *All tests passing*
  + *Business layer should be totally or mostly covered by unit tests (> 80%)*
  + *If your quality gate is not passing, you should be aware of the technical debt. What did you improve since last sprint?*
  + *Take a screenshot of your latest Sonarqube results*

### Sprint 6

* *Final UX feedback report (feedback from 2 different users) and resulting improvements on your UI*
* *Final individual track product with minimum viable product (MVP) features implemented. We want to see a demo of your system. It is mandatory to submit a "lastName\_firstName\_demo.zip" file (directly on Canvas, not in the repository) which contains:*
  + *A document (1 page) with a list of all planned user stories - show a simple list with user story name and status ("finished", "in progress", or "to do").*
  + *For each user story which is (almost) finished include a video recording which shows a demo of that user story. (\*Note: If you do not submit a video of "User Story X", that means that "User Story X" is not implemented in your individual project this semester.)*
* *Continuous Integration and Sonarqube*
  + *All tests passing*
  + *Business layer should be totally or mostly covered by unit tests (> 80%)*
  + *If your quality gate is not passing, you should be aware of the technical debt. What did you improve since last sprint?*
  + *Take a screenshot of your latest Sonarqube results*
  + *(Optional) Integration tests*
  + *(Optional) End-to-end tests*
* *Continuous Delivery*
  + *You remember and understand the importance of Continuous Delivery*
  + *You remember what a container is and why Docker is useful*
  + *You can create a Dockerfile for building images of your backend or frontend*
  + *(Optional) You can run your image against a new environment (test, staging or prod)*
  + *(Optional) You can do Continuous Delivery to an environment (pipelines integration)*
* *(Optional) A web performance review document, using Google Lighthouse, JMeter or a different measurement tool, as well as a reflection on this performance, indicating possible bottlenecks or applied improvements*

## Time plan and milestones

|  |  |  |  |
| --- | --- | --- | --- |
| **Phasing** | **Effort** | **Start date** | **Finish date** |
| 1. Sprint 1 |  | 06.02.2023 | 03.03.2023 |
| 1. Sprint 2 |  | 06.03.2023 | 24.03.2023 |
| 1. Sprint 3 |  | 27.03.2023 | 14.04.2023 |
| 1. Sprint 4 |  | 17.04.2023 | 05.05.2023 |
| 1. Sprint 5 |  | 08.05.2023 | 02.06.2023 |
| 1. Sprint 6 |  | 04.06.2023 | 16.06.2023 |

# Testing strategy and configuration management

## Testing strategy

For the testing strategy, I will opt for a wide variety of testing methods: unit tests, component tests, integration tests, API tests and end-to-end tests.

Unit tests are very important to test the functionality of the business layer. It must have over 80% code coverage.

Components tests will be used to test the functionality of the individual classes. In this type of testing those test objects can be tested independently as a component without integrating with other components e.g., modules, classes, objects, and programs.

Integration tests are used to test multiple components at the same time. This can be particularly useful when a more complex process is being tested. I will test a virtual database by adding multiple things in it and performing multiple operations in it.

Controller tests will ensure the stability and reliability of controller calls and if they return the right thing. For the sake of displaying the test, I will only make a few of them for some classes, where I will also test authentication & authorization features.

End-to-end tests will be handled by my frontend and will check the speed and efficiency of the web application, as well as checking for potential issues that might occur. This type of test is important because I want to make sure my application can handle multiple requests at the same time and can handle multiple users.

## Test environment and required resources.

Tools like Sonarqube, CI/CD pipelines, JUnit and automatic tests will be used to facilitate these testing processes.

## Configuration management

I will be using GIT for version control and a system of branches for each feature. Also, I will be using the Gitlab’s ‘Issues’ page to keep track of my user stories, but also for the teachers to analyze and follow my progress.

# Finances and risk

## Project budget

No real budget will be allocated for this individual project.

## Risk and mitigation

|  |  |  |
| --- | --- | --- |
| **Risk** | **Prevention activities** | **Mitigation activities** |
| 1. Teachers will be unavailable in one day. | Keep the communication opened and announce before being unavailable. | Try to reach them and if it is not possible, try focusing on the current issues and ask other students for help. |
| 1. Technical problems will occur to my hardware. | Be careful with the hardware and don’t navigate on unsafe web pages. | Saving the content and resuming the work on another laptop. |
| 1. Andrei will be unavailable in one day. | Keep the communication opened and announce before being unavailable. | Try to reach the teachers and if it is not possible, try recovering as soon as possible. |
| 1. Software tools might break before a sprint handover. | Be careful with the software and continuously test everything. | Have some backup files saved up and a database generation script. |