Linnéuniversitetet
Kalmar Växjö

Project Design

Författare: Team 1
Examinator: Tobias Andersson
Termin: VT-19
Ämne: Project Course in CS
Kurskod: 1DV005

Index

Introduction

Our client, Tobias Andersson, would like to create a website that serves as e-commerce that sells the services of a personal tutor for a number of subjects. The users will log in in the application and select a tutor and pay a price per hour of study. The orders will be archived and available to the user.

Overview

This document is aimed at the development team. It will give a structure and guidelines of the structure that the team has to follow in order to complete the project.

Goals

1. A user is able to login in the application.
2. A customer is able to select items to his shopping cart.
3. A customer is able to check out and buy products.
4. A customer is able to rate products.
5. An administrator is able to manage products.
6. An administrator is able to manage categories.
7. An administrator is able to mark a product as "deal of the day"

The team defines project success with the following criteria.

- The team is able to deliver a functional application where the basic features are implemented. The use cases and specifications of the features are included in the Requirements Document.
- The application is easy to use for the users of the frontend website and mostly easy for the users of the back end. It is understandable that the complexity of the back end is higher than the front end system.
- The members of the team are able to work together and deploy the application and its deliverables.

Milestones

Deliverable	Date
Setting up Project	10, April
First Iteration. Requirements Documentation	17, April
Second Iteration. Project Plan	24, April
Third Iteration Testing Document	1, May
Fourth Iteration Documentation refinement	8, May
Fifth Iteration	15, May
Sixth Iteration User Guide and System Documentation	22, May
Final Sprint	29, May

Project Organisation

The team has 10 members. The structure is lateral and the team is able to interact with each other without previous requirements. A large number of members has made that many of the roles are shared by several members of the team.

Structure

Scrum master Abdulla Mehdi (Sprint 1-3) Dusting Payne (Sprint 4)	Prioritize user stories. Identify and assign tasks. Coordinates project member to keep the project updated according to the guidelines. Submit deliverables and shows the product to the customer.
Secretary Dustin Payne Einar van de Velde	Monitors progress and provide status reports
Technical writer **Requirements engineer** Helena Tevar Hernandez Einar van de Velde	Coordinates changes and version control of the product. Maintains version control of the documentation and guidelines.
Database builder Maxim Makov	Coordinates the implementation of the database elements of the project.
Tester Andrei Neagu Helena Tevar Hernandez	Verifies at each iteration that the product is meeting the requirements.
Repository managers Abdulla Mehdi Einar van de Velde Dusting Payne Helena Tevar Hernandez	Manage pull requests and merges.
Developers[1] Ahmad Anbarje Abdurahman AlAllaf Domagoj Trupeljak Sarmed Almjamai	Develop the design and implementation of the products.

[1] All the members above mentioned also participate in developing the system

System Design

This project will use a Scrum iterative methodology. During the time given, the team will have the next meetings:

- Spring initial meeting
- Daily meeting
- Spring final meeting
- Customer-deployment meeting

The team will use management tools as Redmine to coordinate tasks and use a Kanban table and Slack as the main communication tool. The team will flavour the methodology by adding group programming as much as possible, to improve the function of this large group.

Environment

Operating System	Windows 10, Ubuntu
Software Languages	Angularjs, Nodejs, Bootstrap
Database	Firebase
Software IDE	Visual Studio Code / WebStorm
Main repository	https://gitlab.com/LenaTevar/508project
Web Browsers	Chrome, Firefox

Versions

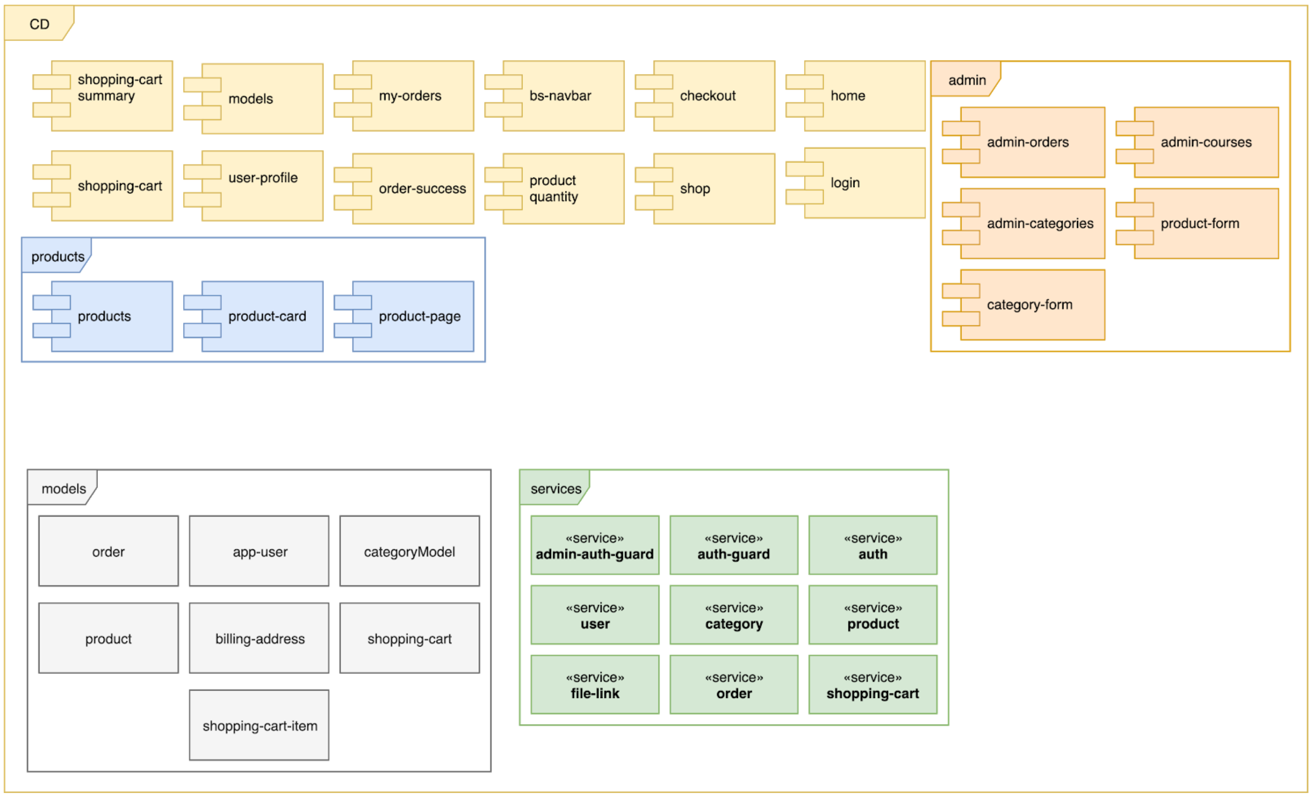
Package	**Version**
Angular CLI	7.3.8
Node	11.13.0
Angular/fire (Newest version from AngularFire2)	5.1.2
rxjs	6.3.3
Bootstrap	4.3.1
npm windows	6.7.0
npm linux	0.34.0

Architecture and Design

This project will follow a **separation by concerns** (SoC) principle. The high risks and time constraints the team face makes SoC the most logical way to approach this project. SoC is a design principle for separating the different components of our applications into sections, so each section is responsible for different concerns, let's say, one functionality of the program. All the components that are related to each other should be put together into the same package. This type of design will be implemented and refined during the process, scaling up with the system. By using a design pattern, we achieve low coupling and high cohesion.
By having a low coupling system, all our components and services have a low connection between them, the interdependence between components is low. For instance, the majority of the connections created in our system are data coupling, where the dependency between the modules is more related to data flow between components.
At the same time, our system as high cohesion because we divided tasks into components grouped together in a logical way, so elements that work together are closer.

Patterns
Online applications that work with a database have to take care of a special situation. When several users manage data from a database at the same time they are working concurrently, especially with firebase, that uses a stream as data flow. For this reason, synchronizing the data is important. The project uses often the **observer pattern**, that is implemented by the package "Reactive extension for JS" or rxjs.
The observer patterns ensure that when a component changes of state or data, a number of related components are also updated automatically. Firebase uses a stream of data, so we use rxjs to create asynchrony, that is to work concurrently without stopping the application to wait for the results.

UML


CD
shopping-cart summary
models
my-orders
bs-navbar
checkout
home
shopping-cart
user-profile
order-success
product quantity
shop
login
admin
admin-orders
admin-courses
admin-categories
product-form
category-form
products
products
product-card
product-page
models
order
app-user
categoryModel
product
billing-address
shopping-cart
shopping-cart-item
services
«service» admin-auth-guard
«service» auth-guard
«service» auth
«service» user
«service» category
«service» product
«service» file-link
«service» order
«service» shopping-cart