<User System>

Requirements Specification and Analysis

<1.0>

<19.17.2019>

<Şuayb Talha Özçelik>

Prepared for

**Finartz GG Department**



**REQUIREMENTS ANALYSIS DOCUMENT**

**1. Introduction**

The purpose of this system is to provide an efficient User Manager platform. The proposed system was designed so as to enable user manager to see all users with ease. In addition, manager can add,delete or update user. The driving motive behind the development of the system was a manage users with one page. The system should be user-friendly and need minimal training/knowledge to be able to use.

**1.1. Purpose of the System**

This system was designed with the aim of providing easy, user-friendly and secure for managers.

**1.2. Current System**

The system has model ,controller ,service ,repository ,exceptions , shared packages. Model has User ,ErrorMessage ,ReCaptchaResponse java classes User has **userId, userName, name, surname, phoneNumber** attributes, these attributes has @NotNull ,@Size annotations.

User class has @Document annotation.

The system has repository package , this package has UserRepository interface with extended MongoRepository.

The system has service package ,this package has interface UserService and java class UserServiceImpl. UserServiceImpl has implements UserService.

The system has controller package, this package has java class UserController. This is class has @Controller annotation. This class’s methods have @RequestMapping annotation. This class control hole system with these methods.

findAllUsers()

deleteUsers()

saveUser()

saveUsers() : method has @RequestParam annotation which is using for ReCaptcha. System check when save actions is clicked with manager to enter this ReCaptcha.

In Exceptions package has AppExceptionsHandler and UserServiceExceptions which are used for system exceptions to understand what is wrong with requests and response.

In user.html we use xmlns:th="http://www.thymeleaf.org" to show model attributes or object in html.

|  |
| --- |
| **Scenario name:** Add User |
| **Participant actor instances:** **Şuayb:**User |
| **Flow of events:**   1. Firstly Şuayb enter the web site on his device 2. Şuayb sees main page. Şuayb sees Add New User button bottom of User List   ( Add New User ) clicks this button.   1. It will be open a Modal Dialog page, Şuayb sees requirement inputs , Şuayb enter all of these requirement inputs. 2. If he didn’t click the Captcha System warns him to enter the Captcha. 3. If these all requirements is done , then he can add the new user with clicking add button. 4. Şuayb clicks Add. 5. New User is created. |

Scenario1

Scenario 2

|  |
| --- |
| **Scenario name:** Edit User |
| **Participant actor instances:** **Şuayb:**User |
| **Flow of events:**   1. Firstly Şuayb enter the web site on his device 2. Şuayb sees main page. Şuayb sees Edit button right-cornet of Users on the User List,   ( Edit ) clicks this button.   1. It will be open a Modal Dialog page, Şuayb sees requirement inputs , Şuayb enter all of these requirement inputs. 2. If he didn’t click the Captcha System warns him to enter the Captcha. 3. If these all requirements is done , then he can add the new user with clicking edit button. 4. Şuayb clicks Edit. 5. User is edited. |

Scenario 3

|  |
| --- |
| **Scenario name:** Delete User |
| **Participant actor instances:** **Şuayb:**User |
| **Flow of events:**   1. Firstly Şuayb enter the web site on his device 2. Şuayb sees main page. Şuayb sees Delete button right-cornet of Users on the User List,   ( Delete ) clicks this button.   1. It will be open a Modal Dialog page, Şuayb sees warn text. 2. If he is sure about deleting the user , he can click delete button. 3. Şuayb clicks Delete. 4. The User is deleted. |

**Manager**: Manager is the person who can manages all the content. Manager can add user, edit and delete user profile.

**Spring-Mvc**: Spring Web MVC is the original web framework built on the Servlet API and has been included in the Spring Framework from the very beginning. The formal name, “Spring Web MVC,” comes from the name of its source module ([spring-webmvc](https://github.com/spring-projects/spring-framework/tree/master/spring-webmvc)), but it is more commonly known as “Spring MVC”.

**Modal Dialog**: A **modal dialog** is a **window** that forces the user to interact with it before they can go back to using the parent application.

**References**

1. <https://spring.io/>
2. <https://www.udemy.com/spring-boot-microservices-and-spring-cloud/learn/lecture/13367504#overview>