|  |
| --- |
| **Website administration** |
| Use Case document for Full Stack development |
| This document covers Software Requirements of Website administration, along with list of Technologies to be used to develop this Software System, and also includes some details on the Architecture |

Table of Contents

[1. Business Requirement (Website Administration) 3](#_Toc47361034)

[2. Technical Spec – Solution Development Environment 6](#_Toc47361035)

[*Front End Layer* 6](#_Toc47361036)

[*Middle Tier Layer* 6](#_Toc47361037)

[*Database & Integration Layer* 6](#_Toc47361038)

[*Ancillary Layer* 6](#_Toc47361039)

[*Security* 6](#_Toc47361040)

[*Deployment & Infrastructure* 6](#_Toc47361041)

[*Editors* 6](#_Toc47361042)

[Coding Standards and Guidelines 7](#_Toc47361043)

[Designing for Testability 7](#_Toc47361044)

[3. UI Components 8](#_Toc47361045)

[3.1. UI Wireframes 8](#_Toc47361046)

[3.2. UX – Angular Components 9](#_Toc47361047)

[4. Mid-Tier NET Components 10](#_Toc47361048)

[4.1. Architecture Diagram 10](#_Toc47361049)

[4.2. Database Constructs, Model Classes and Entity classes 10](#_Toc47361050)

[4.3. Development of Web API 10](#_Toc47361051)

[4.4. JWT Authentication 12](#_Toc47361052)

[5. DevOps Activity 13](#_Toc47361053)

[5.1. Diagram 13](#_Toc47361054)

[5.2. Jenkins CI/CD 13](#_Toc47361055)

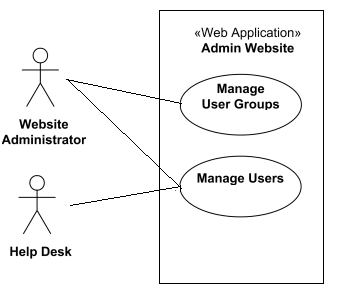
[5.3. Configure Jenkins and Docker for the Project 13](#_Toc47361056)

[5.4. Perform CI/CD 13](#_Toc47361057)

[6. Important Instructions 14](#_Toc47361058)

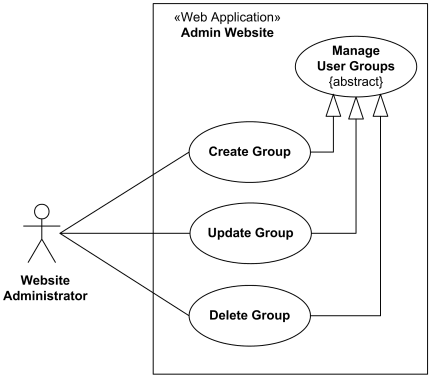
# Business Requirement (Website Administration)

Build a software system which lets website administrator to manage users, user groups, user sessions. Along with administrators, some part of the administrative interfaces should be also available to the Help desk staff, as they need to be able to assist customers having issues while using the customer oriented website.



*Top level use case diagram for the administration website.*

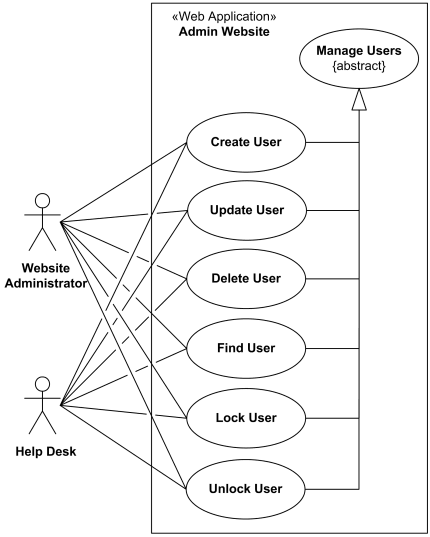
**Manage User Groups** abstract use case is specialized by **Create Group**, **Update Group**, and **Delete Group** use cases. The idea is that website administrator could create different user groups, for example, having different privileges or options, and later some user groups could be modified or even deleted.



*User group management use case diagram for the administration website.*

User management use cases are available both to the **Website Administrator** and to the **Help Desk**. There is standard user CRUD (Create, Retrieve/Find, Update, Delete) functionality set.

Two other use cases, **Lock User** and **Unlock User** are specific to website security. For example, if during some predefined period of time there were several unsuccessful login attempts using wrong user password, user account should be locked for some predefined time to prevent possible brute force password guessing attack. This locking and unlocking is usually done automatically by intrusion detection or website authentication subsystem, but this functionality needs to be available in the manual mode too, just in case. For example, some user might call and ask to lock his or her account.



*User management use case diagram for the administration website.*

# Technical Spec – Solution Development Environment

The technologies included in Full Stack are not limited to following but may consist of:

### Front End Layer

|  |  |
| --- | --- |
| Framework(s)/SDK/Libraries | Version |
| Angular | 8 |
| TypeScript compatible with Angular 8 | 3.4 |
| Bootstrap | 4 |
| CSS | 3 |
| HTML | 5 |
| JavaScript | ECMAScript 2019 / June 2019 |
| JQuery | 3.5.1 / (May 4, 2020) |

### Middle Tier Layer

|  |  |  |
| --- | --- | --- |
| Technology | Framework(s)/SDK/Libraries | Version |
| NET Stack | ASP.NET Core | 2.0 or above |
| .NET Core | 3.0 |

### Database & Integration Layer

|  |  |  |
| --- | --- | --- |
| Technology | Framework(s)/SDK/Libraries | Version |
| NET Stack | Entity Framework | EF 6.X |
| MS-SQL Server | 2020 |
| MongoDB | MongoDB | 3.4 |

### Ancillary Layer

|  |  |  |
| --- | --- | --- |
| Technology | Framework(s)/SDK/Libraries | Version |
| Source Code Management Tool | GIT | 2.25.1 or above |
| Build Tool/.NET Stack | MS Build | 16 |
| Testing Tool/NET Stack | NUnit/XUnit | 3 |
| Controllers can be tested using Postman Tool | | |

### Security

|  |  |
| --- | --- |
| Name | Version |
| ASP.NET Core Security | 2.x |

### Deployment & Infrastructure

|  |  |  |
| --- | --- | --- |
| Technology | Framework(s)/SDK/Libraries | Version |
| Docker | - |  |
| Jenkins(CI/CD) | - | 2.204.5 and above  2.54 is Preferred |
| Node | - | 10.16.0 |

### Editors

|  |  |
| --- | --- |
| Name | Version |
| Visual Studio Professional | 2019 |
| Visual Studio Code | 1.4.2 |

### Coding Standards and Guidelines

Throughout the implementation of use case, standard naming and coding guidelines needs to be followed for GUI and .NET in order to

* produce easily readable,
* consistent code with proper naming conventions
* achieve robustness,
* maintainable code through proper exceptional handling and logging,
* assist software portability,

Refer these links for more details around coding standards.

* <https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/inside-a-program/coding-conventions>
* <http://www.sourceformat.com/pdf/sql-coding-standard-sqlserver.pdf>
* <https://www.c-sharpcorner.com/UploadFile/a8024d/best-practices-coding-standards-folder-structure-for-asp/>
* <https://www.dotnetcurry.com/software-gardening/1333/coding-guidelines-important-for-developers>

### Designing for Testability

Emphasis should be given to Test Driven Development (TDD) and wherever needed, Integration test cases should also be written.

## 3. UI Components

### 3.1. UI Wireframes

In this Phase you will develop, responsive UI of the application using HTLM5, HTML5 API, CSS3 and Bootstrap/Material. (This Phase does not include development of Angular Components)

#### Below are wireframes which need to be developed in this Phase

1. Login for Website administrator and Helpdesk
2. Website administrator Landing Page and Navigation Links
3. Helpdesk Landing Page and Navigation Links
4. Manage user group pages
5. Manage user pages

Based on the requirements, Front End need to be divided into multiple components to accommodate above Wireframes. Angular Routing can be used to create navigation Links. For Authentication, store JWT token in Local or Session Storage. REST APIs are invoked from the corresponding Services,

As known JWT token is generated on the Server side and received by Client on successful authentication. Angular’s HttpInterceptor can be used to automatically send JWT Token thru Header of every HttpRequest.

To complete this case study, you should be comfortable with basic single page web application concepts including REST and CRUD. You may use angular-cli to create your template project. All web pages need to be responsive.

Ref1: <https://cli.angular.io/>

Ref2: <https://github.com/angular/angular-cli>

#### Deliverables of this Phase

1. Wireframes covering menu options and major webpages, in any format
2. Html templates
3. Screen shots of sample Html templates

### 3.2. **UX – Angular Components**

In this Phase you will develop, UX of the SPA application using Angular and Typescript. You will use Angular, Javascript and Jquery where required.

#### This Phase includes development of

1. Angular components using Templates(Templates are created in Phase1)
2. Angular Routing
3. Angular Services, to invoke REST End points.

Angular Routing feature is used to display Navigation links on Web Pages. When User clicks a Navigation link, the corresponding Component is displayed.

As known, Angular Services is used to interact with the REST end points, using Observable or Promises.

Based on the requirements, Front End need to be divided into multiple components to accommodate above Wireframes. Angular Routing can be used to create navigation Links. For Authentication, store JWT token in Local or Session Storage. REST APIs are invoked from the corresponding Services,

As known JWT token is generated on the Server side and received by Client on successful authentication. Angular’s HttpInterceptor can be used to automatically send JWT Token thru Header of every HttpRequest.

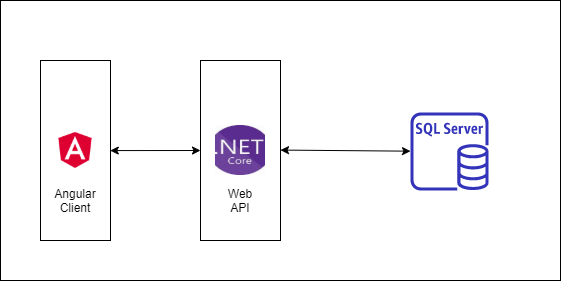
###### Deliverables of this Phase

1. UI screens ready to be integrated with backend REST API endpoints.
2. Demo-able screens with mock data.

## Mid-Tier NET Components

### Architecture Diagram

Architecture of a Single API with REST Controller, Service, Model & Entity Classes and Repository classes



### Database Constructs, Model Classes and Entity classes

This Phase comprises identifying and developing following components

1. Database Constructs like Tables, Views, Integrity Constraints, Indexing, Sequences.
2. Entity Classes and
3. Model Classes

which are required to be used in the next Phases.

Entity classes are the ones which are mapped with underlying Database tables, Entity classes also includes corresponding mappings between them, such as

* One to One Mapping
* One to Many Mapping
* Many to One Mapping
* Many to Many Mapping

### Development of Web API

This specific phase is to design/develop Web API. Analyze the requirement and divide back end functionality into multiple API. Based on the Website administrator requirements

Each of above API Service need to comprise below functionality, which need to be developed

1. REST Controllers
2. Services
3. Entity & Model classes, including appropriate relationship (like One-One, Many-One, etc…) between Entity Classes. (Entity and Model classes have been developed in the Previous Phase)
4. Use Postman to test the API by directly passing requests to each REST end Point, of each service
5. Unit Testing code can be developed using NUnit and perform Unit Testing

### JWT Authentication

Create additional Microservice which takes care of authentication and role activities, and JWT Token validation. Spring Security need to be used for Authentication. On successful authentication or token validation the actual request need to be forwarded to the corresponding Microservice. Invoke authentication REST endpoints from Zuul Gateway. Use PreFilter to perform JWT Token validation by invoking REST endpoint of this Microservice.

Instead of JWT, any other security protocol such as OAuth2 can be used. Authentication data can be stored in MySQL DB or LDAP or any other data source.

## DevOps Activity

This phase includes performing below Activities

**Dockerization:** Dockerize atleast Front End or any one Web API of Mid Tier. Provide Dockerfile and the docker commands used to create image and run Container. Share Screen shots of running Docker. To Setup Docker Client on your VM please refer

**JMeter:** As already known JMeter is used to perform Performance or Load Testing. Create a JMeter Test Case, which invokes a REST End point, with multiple threads. Check in jmx file and share the report generated after Performance Testing. Repeat this for atleast two REST end points.

**Code Coverage:** Code coverage is a Quality Metric to check if sufficient number of Test Cases are created. EclEmma tool can be used as Code Coverage Tool. Code Coverage can be performed on any one Microservice.Ensure that Code Coverage need to be atleast 80%

### Diagram



### Jenkins CI/CD

**Jenkins CI/CD:** As already known Jenkins is popular tool to perform CI/CD. When the code is pushed to GIT, build need to be automatically created and deployed. If possible create a Docker image and run the Container on Docker Host

This Phase also includes completion of Integration of Front end with Mid Tier.

**Deployment on Cloud(optional):** Web API or Front End can be deployed on any Cloud(AWS, Azure, etc…) of your choice.

### Configure Jenkins and Docker for the Project

### Perform CI/CD

#### Assessment Deliverables

1. Jenkinsfile, Jenkins screenshots
2. Dockerfile and Docker commands used and Screen shots of usage of Docker

# Important Instructions

Adhere to the design specifications mentioned in the case study.

1. Please make sure that your code does not have any compilation errors while submitting your case study solution.
2. The final solution should be a zipped code having solution. Solution code will be used to perform Static code evaluation.
3. Implement the code using best design standards/family Design Patterns.
4. Use Internationalization for all the labels and messages in Rest API Development.
5. Do not use System out statements or console.log for logging in Rest API and FrontEnd respectively. Use appropriate logging methods for logging statements/variable/return values.
6. If you are using Spring Restful or Jersey JAX-RS to develop Rest API, then use Maven to build the project and create WAR file.
7. Write web service which takes input and return required details from database.
8. Use JSON format to transfer the results.