

Cognizant Academy

Store & Staff Management

Case Study Specification

Store & Staff Management

Contents

[1 Problem statement 4](#_Toc112751446)

[2 Architecture Diagram for the Problem Statement 4](#_Toc112751447)

[3 Use case details 5](#_Toc112751448)

[4 Functional/Non-Functional Requirement of the Problem Statement 6](#_Toc112751449)

[4.1 Login 6](#_Toc112751450)

[4.1.1 Acceptance Criteria 7](#_Toc112751451)

[4.1.2 UI-Screen View 7](#_Toc112751452)

[4.1.3 REST End Points 7](#_Toc112751453)

[4.1.4 Entity 8](#_Toc112751454)

[4.1.5 Login Table 8](#_Toc112751455)

[4.1.6 Non-Functional Requirements 8](#_Toc112751456)

[4.1.7 Business Validations 8](#_Toc112751457)

[4.2 Load Store Data 9](#_Toc112751458)

[4.2.1 Acceptance criteria 10](#_Toc112751459)

[4.2.2 UI- Screen view 10](#_Toc112751460)

[4.2.3 REST End Points 10](#_Toc112751461)

[4.2.4 Database tables 11](#_Toc112751462)

[4.2.5 Detailed requirement 11](#_Toc112751463)

[4.2.6 Non-Functional Requirements 11](#_Toc112751464)

[4.2.7 Business Validations 11](#_Toc112751465)

[4.3 Load Staff Data 11](#_Toc112751466)

[4.3.1 Acceptance criteria 13](#_Toc112751467)

[4.3.2 UI- Screen view 13](#_Toc112751468)

[4.3.3 REST End Points 13](#_Toc112751469)

[4.3.4 Database tables 14](#_Toc112751470)

[4.3.5 Detailed requirement 14](#_Toc112751471)

[4.3.6 Non-Functional Requirements 14](#_Toc112751472)

[4.3.7 Business Validations 14](#_Toc112751473)

[4.4 Manage Store 14](#_Toc112751474)

[4.4.1 Acceptance criteria 15](#_Toc112751475)

[4.4.2 REST End Points 15](#_Toc112751476)

[4.4.3 UI- Screen view 15](#_Toc112751477)

[4.4.4 Database tables 18](#_Toc112751478)

[4.4.5 Non-Functional Requirements 18](#_Toc112751479)

[4.5 Manage Staff 18](#_Toc112751480)

[4.5.1 Acceptance criteria 19](#_Toc112751481)

[4.5.2 UI- Screen view 20](#_Toc112751482)

[4.5.3 REST End Points 22](#_Toc112751483)

[4.5.4 Database tables 22](#_Toc112751484)

[4.5.5 Non-Functional Requirements 22](#_Toc112751485)

[5 Expected Deliverables 22](#_Toc112751486)

[6 Milestone 23](#_Toc112751487)

[7 Skills to develop the project 23](#_Toc112751488)

[8 Implementation Notes 24](#_Toc112751489)

[9 Evaluation rubrics 24](#_Toc112751490)

# Problem statement

The purpose of the requirements document is to systematically capture requirements for the project and the system “**Employee and Store Registration Portal**” to be developed. The application should be Cloud Native Architecture with Microservices. Both functional and non-functional requirements are captured in this document. It also serves as the input for the project scoping.

**About the System**

The client would like to develop an independent application Employee and store Registration Portal application in order to raise request to change the employee details and store info details.

**Scope of the System**

The scope of the system is explained through its modules as follows

* Store – used by the system to display and raise change request on the store related information like store working hours, store area, region and district, store address, store phone number and other information.
* Employee – Used by the system to display store employee information and able to submit changes in employee information

# Architecture Diagram for the Problem Statement

**Use case Diagram**

**US\_01 Store Information**

View Store Information and m

User

DB

Batch job to update master store info

**US\_02 Employee information**

View Employee info

User

Save request to DB

Batch job to update master employee info

# Use case details

|  |  |  |
| --- | --- | --- |
| **User Story #** | **User Story Name** | **User Story** |
| US\_01 | Master store table | Receive master store information in a csv/txt file on daily basis, schedule a batch job to receive store information in out folder, read the data and update or insert in the db. Once the file is processed move the file to archive folder, in the forma to f store\_info\_MMDDYYYY.txt  File name: store\_info.csv  Schedule time: 5:00 am  Table name: master\_store\_data |
| US\_02 | Employee info table | Receive employee information on daily basis in csv/txt format, schedule a batch job to read the employee\_info.csv file from out folder. Process the file and update and db, after processing move the file to archive folder in the format of employee\_info\_DDMMYYYY.txt  Schedule time: 5:30 am  Table name: master\_employee\_data |
| US\_03 | Raise request for store | As a store manager I can be able to update the store working hours, store area region and district. Store hours, store address. Each request should be saved in db along with the details on which part of the employee details is changed. Refer below user story. |
| US\_04 | UI View for store | Store info form should have a search bar to enter store number which is of 5 characters, throw error if user enters more than 5 characters. And the value should be numeric. Store details should be fetched from master store table.  Ui should show below items as a separate part so that user can change and raise request for each item. Ui should show edit icon for each of the items below to edit only the item. Ex: store address part should show store address along with an edit icon, on clicking edit icon I should be able to update the address and submit for approval. The request id should be shown in place of edit icon on submission. So that I cannot be able to edit and submit again until the existing request is approved or rejected.   1. store address, 2. store phone numbers, 3. store area, region, and code 4. store hours |
| US\_05 | Employee info view | Employee info form should have a search bar to enter employee number. the user should be able to view the employee details which is fetched from master employee table |
| US\_06 | View Store request | All submitted store and employee request in db should be fetched and displayed to user for approval or rejection. |

# Functional/Non-Functional Requirement of the Problem Statement

## Login

|  |  |  |
| --- | --- | --- |
| US-01 | Login | |
| Acceptance Criteria Admin User needs to provide Username and password. | | |
| UI-Screen View Graphical user interface, text, application  Description automatically generated | | |
| REST End Points POST - /login  **Input**   * Username * Password in encrypted format   **Output**   * The service should response 200 ok along with success message. * Error code 400 for any business validation error. * Error code 500 for internal prog. error | |
| Entity User Details   * **Username**   + Variable name as <userName> * **Password**   + Variable name as <password> | | |
| Login Table  |  |  | | --- | --- | | User Name | Password | | ramesh@gamil.com | Nintendo@711 | | arunk@gamil.com | Gameboy@889 | | pixel@yahoo.com | Bharath@334 | | | |
| Non-Functional Requirements  * Authentication should be mandatory & Admin User alone will be able to login with username and password & the role should be ADMIN\_USER. * Application should allow as much as external configuration rather than deploying for any small change. * Proper error message & logging should be in-place. | | |
| Business Validations  * All fields are mandatory. * Email id should be in valid email pattern, containing a single @. * Password contains at least 8 characters and at most 20 characters.   + It contains at least one digit.   + It contains at least one upper case alphabet.   + It contains at least one lower case alphabet.   + It contains at least one special character which includes! @#$%^&\*().   + It doesn’t contain any white space. | | |

## Load Store Data

|  |  |
| --- | --- |
| US\_02 | Load Store Data |
| Acceptance criteria Spring batch Job should be built & it should be able to parse the CSV file and update master store table with the following details,  Capture the details like sternum, address, street1, street2, city, state, zip, time zone, mon\_hours, tue\_hours, wed\_hours, thur\_hours, fri\_hours, sat\_hours, sun\_hours, area, region, district. | |
| UI- Screen view Graphical user interface, text, application, chat or text message  Description automatically generated | |
| REST End Points **POST -** /loadstoredata  Input - Store object with fields in the below table.  **Output**   * The service should response 200 ok along with success message. * Error code 400 for any business validation error. * Error code 500 for internal prog. error | |
| Database tables **MasterStore**   |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | **store\_**  **num** | **address** | **Phone-number** | **area\_region\_ code** | **mon\_**  **hours** | **Tue\_**  **hours** | **Wed\_**  **hours** | **Thur\_**  **hours** | **Fri\_**  **hours** | **Sat\_**  **hours** | **Sun\_**  **hours** | | Az678 | G14-11 th street | +11 2347629308 | Los Angles, California, 12343 | 9am-7pm | 9am-7pm | 9am-7pm | 9am-7pm | 9am-7pm | 9am-7pm |  | | 76bc9 | M18-2 nd street | +11 2228473847 | Los Angles, California, 12343 | 9am-10pm | 9am-10pm | 9am-10pm | 9am-10pm | 9am-10pm |  | 9am-10pm | | 8uh6fk0 | R34-5 th street | +11 3487923987 | Los Angles, California, 12343 | 9am-7pm | 9am-7pm | 9am-7pm | 9am-7pm | 9am-7pm |  |  | | |
| Detailed requirement Receive master store information in a csv/txt file on daily basis, schedule a batch job to receive store information in out folder, read the data and update or insert in the db. Once the file is processed move the file to archive folder, in the forma to f store\_info\_MMDDYYYY.txt  File name - store\_info.csv  Schedule time - 5:00 am | |
| Non-Functional Requirements  * Authentication should be mandatory & Admin User alone will be able to login with username and password & the role should be ADMIN\_USER. * Application should allow as much as external configuration rather than deploying for any small change. * Proper error message & logging should be in-place. * Any failure in batch processing should be notified to the business users through email. * Audit information should be managed for all the record updates | |
| Business Validations  * All fields are mandatory * Story number should be alphanumeric * Hours should be validated like start hrs should be < end hrs & all days should be passed either with hrs or holiday. * Any holiday should be sent as a “HOLIDAY” | |

## Load Staff Data

|  |  |
| --- | --- |
| US\_02 | Load Staff data |
| Acceptance criteria Spring batch Job should be able to parse a csv file & update the master employee table with the following details,  staffId, position, email, office phone no, cellphone, Dept, Area, Region, district. | |
| UI- Screen view Graphical user interface, text, application, chat or text message  Description automatically generated | |
| REST End Points **POST - /**loadstaffdata  **Input**   * Staff Object with the fields in the below table.   **Output**   * The service should response 200 ok along with success message. * Error code 400 for any business validation error. * Error code 500 for internal prog. error | |
| Database tables **MasterStaff**   |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | **Staff\_id** | **position** | **Email** | **Office\_phone\_ no** | **cellphone** | **Dept\_area\_region\_ district** | | 12543 | Manager | Test\_12543@abc.com | +11 3897234872 | 3298729872 | Medic, los Angles, California | | 34329 | Clerk | [Test\_34329@abc.com](mailto:Test_34329@abc.com) | +11 2347238799 | 3298729872 | Medic, los Angles, California | | 92398 | Driver | Test\_92398@abc.com | +11 2347238799 | 3298729872 | Medic, los Angles, California | | |
| Detailed requirement Receive employee information on daily basis in csv/txt format, schedule a batch job to read the employee\_info.csv file from out folder. Process the file and update and db., after processing move the file to archive folder in the format of employee\_info\_DDMMYYYY.txt  Schedule time - 5:30 am | |
| Non-Functional Requirements  * Authentication should be mandatory & Admin User alone will be able to login with username and password & the role should be STORE\_ADMIN. * Application should allow as much as external configuration rather than deploying for any small change. * Proper error message & logging should be in-place. * Any failure in batch processing should be notified to the business users through email. * Audit information should be managed. | |
| Business Validations  * Employee Id should be numeric & min 5 digits. No negative values & all zeros * One of a contact is mandatory & it should follow the standard US phone number format validation. * Position & Dept fields are mandatory. | |

## Manage Store

|  |  |
| --- | --- |
| US\_03 | Manage Store |
| Acceptance criteria As a store manager, he / she can be able to search for a store and submit request  to update the store working hours, store area region and district. Store hours, store address. Each request should be saved in db. Along with the details on which part of the employee details is changed. Refer below user story. | |
| REST End Points **GET - /**retrieveStore/{storeNumber}  **Output**   * The service should response 200 ok along with success message. * Error code 400 for any business validation error.   Error code 500 for internal prog. Error  **POST -** /updatestore  **Input**   * Store Object   **Output**   * The service should response 200 ok along with success message. * Error code 400 for any business validation error. * Error code 500 for internal prog. Error   **Delete -** /deletestore  **Input**   * Store Object   **Output**   * The service should response 200 ok along with success message. * Error code 400 for any business validation error. * Error code 500 for internal prog. error | |
| UI- Screen view UI- Screen view Graphical user interface, text, application  Description automatically generated | |
| Graphical user interface  Description automatically generated with low confidence | |
| Database tables **MasterStore**   |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | **store\_**  **num** | **address** | **Phone-number** | **area\_region\_ code** | **mon\_**  **hours** | **Tue\_**  **hours** | **Wed\_**  **hours** | **Thur\_**  **hours** | **Fri\_**  **hours** | **Sat\_**  **hours** | **Sun\_**  **hours** | | 54321 | G14-11 th street | +11 2347629308 | Los Angles, California, 12343 | 9am-7pm | 9am-7pm |  | 9am-7pm | 9am-7pm | 9am-7pm |  | | 76839 | M18-2 nd street | +11 2228473847 | Los Angles, California, 12343 |  | 9am-10pm | 9am-10pm | 9am-10pm | 9am-10pm |  | 9am-10pm | | 73289 | R34-5 th street | +11 3487923987 | Los Angles, California, 12343 | 9am-7pm | 9am-7pm | 9am-7pm | 9am-7pm | 10am-7pm |  |  | | |
| Non-Functional Requirements  * Authentication should be mandatory & Admin User alone will be able to login with username and password & the role should be STORE\_MANAGER. * Application should allow as much as external configuration rather than deploying for any small change. * Proper error message & logging should be in-place. * Update notification to be sent to all listed business users. | |

## Manage Staff

|  |  |
| --- | --- |
| US \_05 | Manage Staff |
| Acceptance criteria As a store admin, he / she can be able to search for a staff and submit request  to update the details. | |
| UI- Screen view Graphical user interface, text, application  Description automatically generated  Graphical user interface  Description automatically generated with low confidence | |
| REST End Points Define yourself by referring US\_04 | |
| Database tables  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | **Staff\_id** | **position** | **Email** | **Office\_phone\_ no** | **cellphone** | **Dept\_area\_region\_ district** | | 12543 | Manager | Test\_12543@abc.com | +11 3897234872 | 3298729872 | Medic, los Angles, California | | 34329 | Clerk | [Test\_34329@abc.com](mailto:Test_34329@abc.com) | +11 2347238799 | 3298729872 | Medic, los Angles, California | | 92398 | Driver | Test\_92398@abc.com | +11 2347238799 | 3298729872 | Medic, los Angles, California | | |
| Non-Functional Requirements  * Authentication should be mandatory & Admin User alone will be able to login with username and password & the role should be STORE\_MANAGER. * Application should allow as much as external configuration rather than deploying for any small change. * Proper error message & logging should be in-place. | |

# Expected Deliverables

The following deliverables are expected as outcomes:

* Application Code base
* Readme document on the complete application
  + Setup of the application
  + How to run the application
  + Any inference
  + Screenshot of UI results
* Reports:
  + Unit/Functional Test Report

# Milestone

The milestone for the project use is given below

|  |  |  |
| --- | --- | --- |
| **Milestone** | **Duration (in weeks)** | **Topic** |
| Milestone -1 | 1.5 | * Develop the APIs for the application with Java, Spring boot along with Database design, set-up, table creation with test data. |
| Milestone – 2 | 1 | * Design and develop the UI for the application |
| Milestone -3 | 1 | * Integrate service layer with UI component, Containerize the application * (Based on the access, we will define this section, as Cigna is using mostly AWS / OpenShift) |

# Skills to develop the project

List the Technology based on your respective technology stack, that will be used to development the project.

|  |  |
| --- | --- |
| **Layer** | **Tech Stack** |
| Front end | * React * JavaScript * Karma/ Cypress/ Jest |
| Back End | * JDK1.8 * Spring Boot * JPA/Hibernate * Maven * SonarQube * Junit |
| Database | * MongoDB / MySQL / PostgreSQL |
| Deployment Infra | * AWS code commit/<<TBD>> * Amazon ECR repository |

# Implementation Notes

As per the project requirement modification can be done in the below table.

|  |  |
| --- | --- |
| Milestone -1 | **Spring Boot**  Create Spring Boot REST Microservices to perform SAVE Operation using POST method.   * Use Microservices Architecture * Use Spring batch * Follow coding standards * Follow Standard project structure * Message input/output format should be in JSON (Read the values from the property/input files, wherever applicable). Input/output format can be designed as per the discretion of the participant * Database connections and web service URLs should be configurable. * Use browser / POST Man to invoke APIs * Swagger implementation. * Run SonarQube for code quality. * Implement Junit for unit testing. |
| Milestone -2 | * Implement user-stories using any one of the UI frameworks [React] * Implement Forms, databinding, validations * Use appropriate unit test framework. |
| Milestone -3 | * Integrate service layer with UI component. * Containerize the application * Push your Docker images to an Amazon ECR repository |

# Evaluation rubrics

|  |  |
| --- | --- |
| Angular | * Associate must have used Angular Components, Modules, Databinding, data validation, CLI commands. * Associate must have used Forms and Forms validation * Associate must have used Directives * Associate must have developed Reusable Components * Associate must have followed coding standards |
| REACT | * Associate must have used Component, Databinding, data validation, CLI commands. * Associate must have used Forms and Forms validation * Associate must have defined React state * Associate must have followed coding standards |
| Microservices, Java, JPA/Hibernate, Database | * REST controller * Follow controller ->service->Dao model * Entity and Model classes * Appropriate logging statements * Exception handling * Usage of Java 8 features such as streams, lambda, Async |
| Docker | * Containerize the application * Build docker containers * Push your Docker images to an Amazon ECR repository with the docker push command |
| AWS | * Code is committed in AWS * Push your container images to Amazon Elastic Container Registry |