Member Registration Portal

Contents

1.0 Problem statement

The purpose of the requirements document is to systematically capture requirements for the project

and the system "**MySport**" to be developed. The application should be Cloud Native Architecture with Microservice. 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 MySport application in order to automate the process of registering a Player, edit his details and book a sport facility

Scope of the System

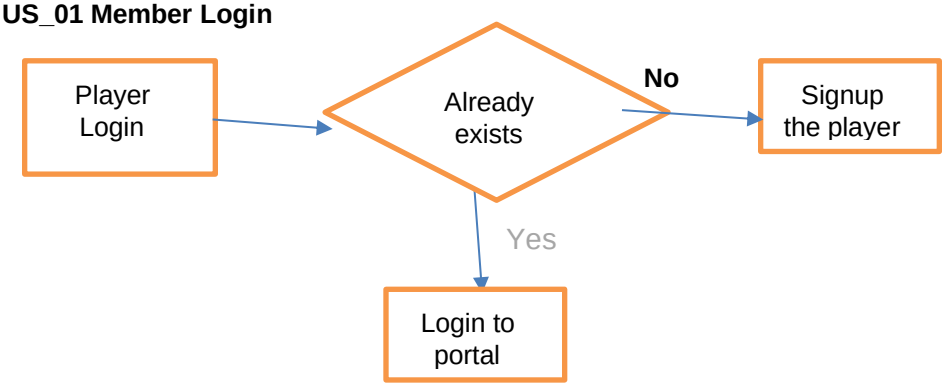
The scope of the system is explained through its modules as follows

- Player Registration – used by players to register the details of self-information into the system. The system stores the details of the member in the system and able to edit it.
- Book Facility – The player should be able to book sport facility like cricket badminton football etc.

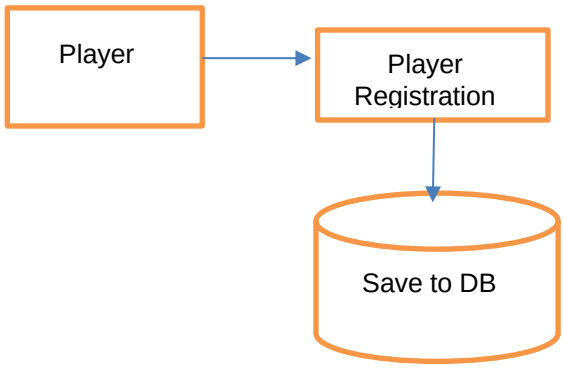
2.0 Architecture Diagram for the Problem Statement

Use case Diagram

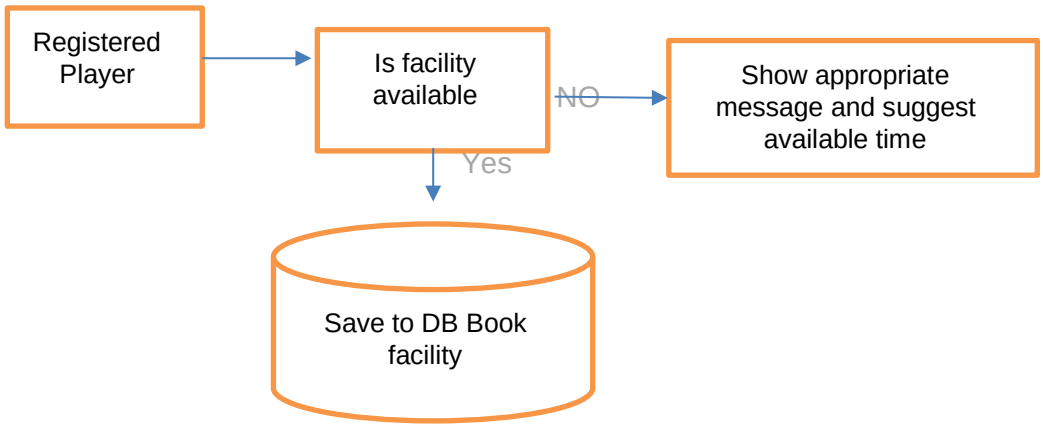
US_01 Member Login



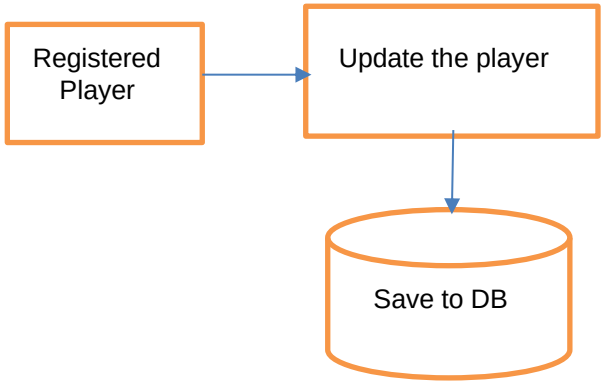
US_02 Member Registration



US_03 Book Facility



US_04 Update Member details



3.0 Use case details

User Story #	User Story Name	User Story
US_01	Login	As a player, I should be able to login to the portal if already registered. If not, I should be able to register as a new player.
US_02	Player Registration	As a Player, I should be able to register my details in the system. Acceptance criteria: Player should be able register the details in the system and it should be saved in the database. Capture the details like Name, Address, State, Country, Email Address, PAN, Contact No, DOB.
US_03	Book Facility	Player should be able to book facility.
US_04	Update the player details	As a Player, I should be able to update the player details

US -01 Login

The player should be able to login to the portal using the registered username and password.

For new users, provide a signup link, which will enable the player to register to the system.

Validations

1. Username should be a valid email Id.
2. Password should be greater than 8 characters long and should contain at least one uppercase, one numeric
   and a special character
3. New player should be able to signup by providing his first name, last name, DOB and valid email id and password. All fields are mandatory.
4. DOB (Date of Birth) should not be less than system date.
5. If the player enters an invalid username/password show error message appropriately.

US -02 Member Registration

Business Rules & Validations

1. Player id should be generated automatically during the time of registration and should be shown in the success message.
2. The player name should contain only alphabets and space.
3. All fields are mandatory.
4. Contact number should be 10 digits.
5. Email id should contain @ and. symbols.
6. Player id should be in the format of 'MS-XXX'.XXX should be random numeric of 3 digits.
7. PAN number must be alpha numeric and no special characters are allowed, no space is allowed. PAN number must be 12.
8. Based on the DOB, age will be calculated.
9. Activation date can be calculated based on the registration date.
10. DOB (Date of Birth) should not be less than system date.
11. Age should be greater than 18.

Validations

- All fields are mandatory.
- Based on the country, state must be populated in the dropdown automatically.
- DOB(Date of Birth) should not be less than system date
- Age should be greater than 18.
- Claim number should be generated automatically and should be a numeric of 10 digits
- Registration Date should not be lesser than system date.

Book facility

US_03	Book Facility
Description	
The player should be able to sports facility. Sports facility available is cricket(2 box cricket), football(1 ground), badminton (3 indoor courts) Timing 6 AM – 10 PM, each slot of 1 hour	

Input Parameters Below are the input parameters. Player id, first name, last name, DOB, date of game, time of game.
Business Rules & Validations • All fields are mandatory. • Player name should contains only characters and space • On entering the playerid , the player details should be automatically populated and it should be a non-editable field. • Date and time of game slot shall be validated.

Member Update

US_04	Member details Update
Description Player should be able to update his details	
Input Parameter ○ Player should be able to update only his mail id, PAN number, state, address, contact number. ○ Email id should be validated. ○ PAN number should not contain any special characters.	
Business Rules & Validations • Email id should contain @ and . Symbols • Contact number should be 10 digit number. • PAN number should not contain any special characters. • On entering the player name , the player details should be auto populated	

Service Requirements

US_01 Login

When the user logins by providing username, password the details should be sent to the POST method and checked in the db if the user exists. If the user does not exist, send exception response. If user found return 200 with success message.
When the user enters the details in signup page, they should be sent to the POST method and saved in db.
Mandatory validation should be done for firstname, lastname, dob, email id and password

US_02 Player Registration

Once the user enters the details, they should be sent to the POST method and saved in the db.
Mandatory fields should be validated as mentioned in the rules above and 400 exception response should be sent with the missing field details.
When the details are saved successfully, the service should response 200 ok along with success message.
If there are any exceptions while connecting/saving to db, the service should throw corresponding error with error status as 500.

US_03 Book Facility
To book facility, application shall show available slots and allow player to choose their preferred sport and slot, a GET method should be implemented to fetch the details.

Once Player enters the details, they should be sent to the POST method and saved in the db.
Mandatory fields should be validated as mentioned in the rules above and 400 exception response should be sent with the missing field details.
Booking id should be generated automatically when the user books a facility
When the details are saved successfully, the service should response 200 ok along with success message.
If there are any exceptions while connecting/saving to db , the service should throw corresponding error with error status as 500.
If facility not available at preferred time by player, the service shall throw business exception.

US_04 Player Update

To retrieve the Player details on entering the name field, a GET method should be implemented to fetch the details.

Once the user enters the details, they should be sent to the PUT method and saved in the db.
Mandatory fields should be validated as mentioned in the rules above and 400 exception response should be sent with the missing field details. When the details are saved successfully, the service should response 200 ok along with success message.
If there are any exceptions while connecting/saving to db , the service should throw corresponding error with error status as 500.

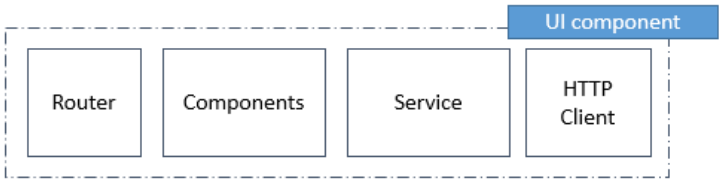
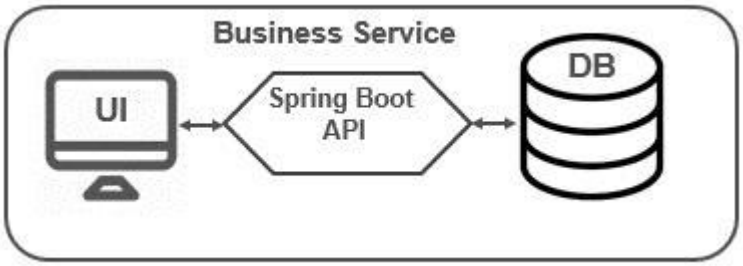
Expected Deliverables

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

4.0 Milestones

The milestone for the project use is given below

Milestone	Duration (in weeks)	Topic
Milestone - 1		Design and develop the UI for the application 
Milestone -2		Develop the required APIs for the application  Or Develop all the APIs with NodeJS.
Milestone -3		Integrate service layer with UI component, Dockerize the application and Push your Docker images to an Amazon ECR repository

5.0 Skills to develop the project

List the Technology based on your respective technology stack, that will be used to development the project.
Associate will choose any one of the technology stack and develop the application.

Skill Stack	Java 8+
Front end	Angular / Bootstrap/ CSS JavaScript/ JQuery Typescript Karma/ Cypress/ Jest
Service End	Spring Boot, Spring MVC, JDK, Maven/ SonarQube/ Junit
Database	MySQL 5+
AWS	AWS CodeCommit, CodeBuild Amazon ECR repository

6.0 Implementation Notes and Rubrics

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

Milestone -1	**SpringBoot:** • Create Spring Boot REST Microservice to perform SAVE Operation using POST method. • Use Microservice Architecture (Refer Evaluation Rubrics) • Use Spring Cloud • Follow coding standards • Follow Standard project structure • Use log4j for logging • 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 • Run SonarQube for code quality. • Implement Junit for unit testing. • Use Mockito framework wherever appropriate. • Enable spring actuator • Integrate with Swagger only for Dev profile • Dockerize the application
Milestone -2	• Implement user-stories using any one of the UI frameworks [Angular] • Design application with Minimum Backend or Mock backend as the main focus in on frontend skills • Implement Forms, databinding, validations • Use Appropriate unit test framework. • Integrate service layer with UI component.
Milestone -3	• Checking source code using AWS CodeCommit. • Configure build with AWS CodeBuild. • Push your Docker images to an Amazon ECR repository

8.0 Evaluation rubrics

Angular	• Build clean and robust application • Dependency Injection • SOLID Principles and Design Patterns • Naming Conventions and standards • Exception Handling • Logging • Performance Considerations • Use Angular CLI • Break down into small reusable components • Maintain proper folder structure • Follow consistent Angular coding styles • Utilize ES6 Features • Use Lazy Loading • Implement the lifecycle hook interface • Cache API calls • State Management • Always Document • Aliases for imports
Typescript/JavaScript	• Apply the best practices • Avoid common pitfalls and mistakes other JavaScript developers make • Write solid JavaScript code

Java	• Lambda Expressions • Functional Interfaces • Default methods in Interface • Static Methods in Interfaces. • Predicate • Function • Consumer • Supplier • Method Reference & Constructor Reference by Double Colon(::) Operator • Stream API • Date & Time API (Joda API) • **Demonstrate 2 Debugging and Troubleshooting** i. Debug a Memory Leak ii. Understand the OutOfMemoryError Exception iii. Troubleshoot a Crash Instead of OutOfMemoryError iv. Diagnose Leaks in Java Language Code v. Diagnose Leaks in Native Code
Spring Boot	• Perform CRUD operations against a in memory db using Spring Data JPA • Perform CRUD operations against Database • Expose out REST APIs using Spring Web • Spring boot auto configures a spring project • Enable health metrics for the application • Customize health metrics endpoint with your own information Use Spring Boot Profiles • **Demonstrate 2 Debugging and Troubleshooting** (Create two scenarios of troubleshooting your Spring Boot application in your project demo)
Microservices	• Keep all code in a Microservices at a similar level of maturity and stability • Do a Separate Build for Each Microservices • Deploy in Containers • Servers as Stateless • Use Fault Tolerance techniques • Use the API gateway • Use Domain-Driven Design • Focus on data and API security • Use Distributed Configuration • Monitor microservices, Use Monitoring Tools (use actuator) • App metrics and health checks • Structure Log Data • Add Context to Every Request • Include a Unique ID in the Response • Use OAuth for user identity and access control

Docker	• Dockerize the application • Build docker containers • Push your Docker images to an Amazon ECR repository with the docker push command
AWS	• Checking source code using AWS CodeCommit • Configure build with AWS CodeBuild • Push your container images to Amazon Elastic Container Registry • Send Logs to a Centralized Location **AWS CodeCommit (Best practices):** • Authentication and access control for AWS CodeCommit, , AWS KMS and encryption, Using rotating credentials • Security in AWS CodeCommit - Data protection in AWS CodeCommit • Identity and Access Management for AWS CodeCommit • Resilience in AWS CodeCommit • Infrastructure security in AWS CodeCommit • Monitoring AWS CodeCommit **AWS CodeBuild (Best practices):** • Security best practices for CodePipeline resources • Monitoring and logging best practices for CodePipeline resources • Jenkins plugin Usage best practices • Security in AWS CodePipeline - Data protection in AWS CodePipeline • Identity and access management for AWS CodePipeline • Logging and monitoring in CodePipeline • Compliance validation for AWS CodePipeline • Resilience in AWS CodePipeline • Infrastructure security in AWS CodePipeline • Security best practices **Debugging and Troubleshooting:** • Exhibit Debugging and Troubleshooting scenarios during project Demo for both AWS CodeCommit, AWS CodeBuild