Customer Session Portal

Duration	:	5 Weeks
Wizard	:	Binuraj E K, Shashikumar S Koti, Gopinath B
Source	:	CITI - TWA

Main Heading

Problem Statement:

To build a portal which be the landing page for the customers who will be advised with the portfolio related activities from the Relationship Manager. RM should be able to manage the sessions based on the customer's availability and requests.

Objective:

To build a portal to manage the customer's sessions which can be utilized as the landing page for the customer – RM interactions.

Technology Stack:

Programming Language: Angular, Java

Framework: Spring Boot with RESTful microservices, any DB, JUNIT5 <Add Frontend framework>

Libraries: RESTful, Lombok <Add Front end libraries>

Activity:

Stage 1: Requirement Gathering

Associates will be interacting with the business stakeholders to understand the requirements with business impact and come up with a rough estimate plan on the deliverables with timeline.

Stage 2: Pre-requisites and Grooming

As a part of this Grooming process,

- Associates will interact with the wizards to understand the functionality, biz logic, design.
- Come-up with the design plan and get the sign off
- Various test cases to be created and it should cover E2E functionality of the application.
- <UI specific content>

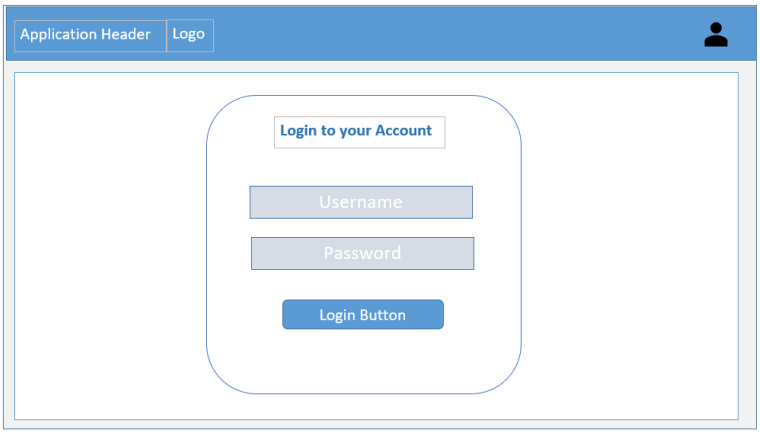
As part of pre-requisites,

- Create a common account for the application or individual account, repository in the github.
- Install visual studio code, Intellij IDEs.
- Preferred DB's IDE and install the DB server.
- Install Postman
- Design table structures and create tables.
- Install SonarQube plugin and configure the rules
- <prerequisites for UI>

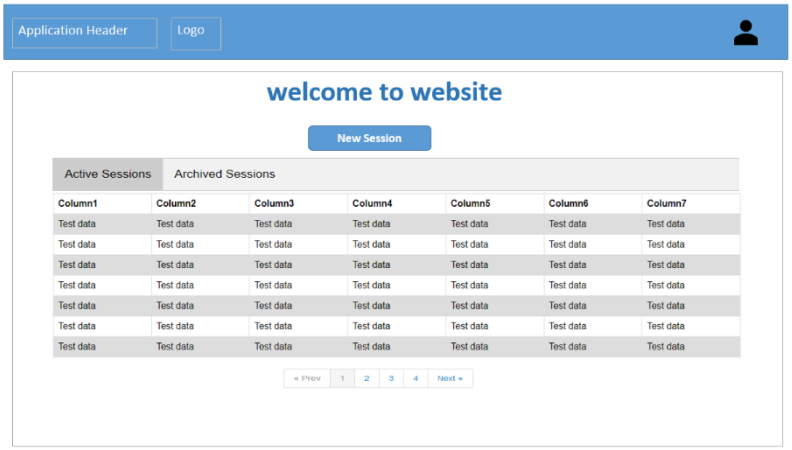
Stage 3: Developing the application

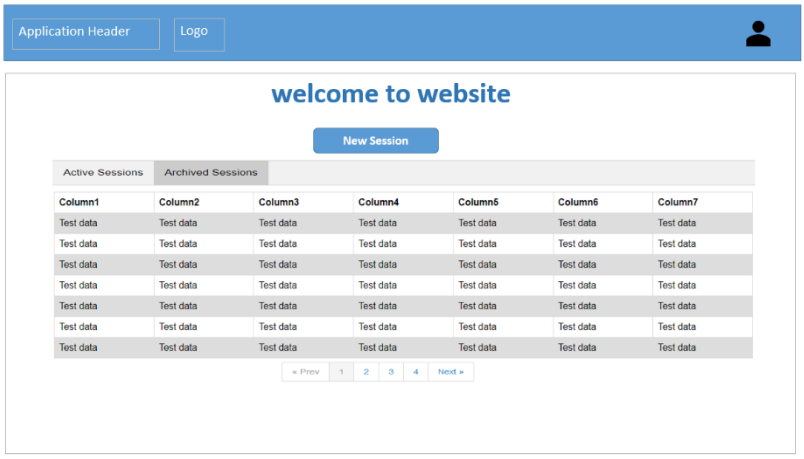
UI Screens goes here.

1. Login

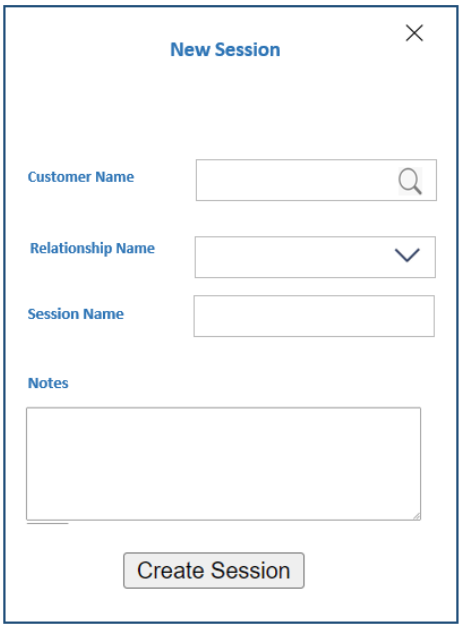


2. Dashboard

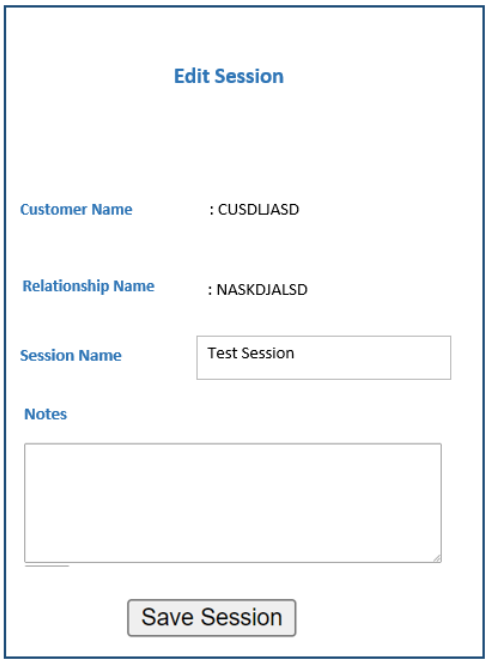




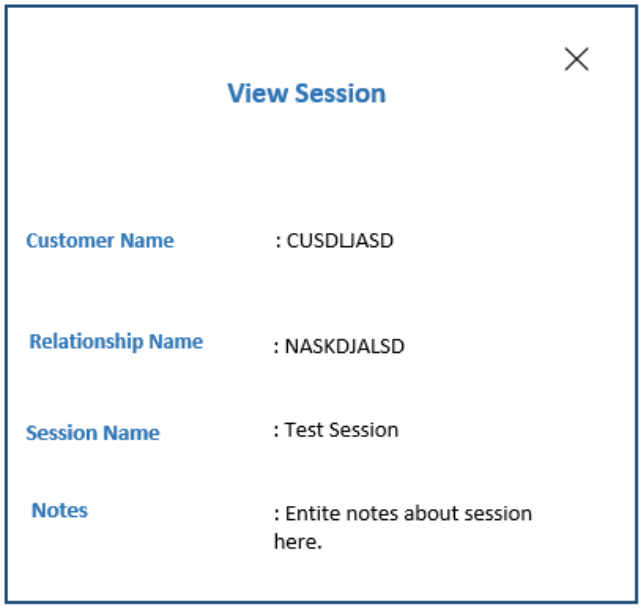
3. Create Session



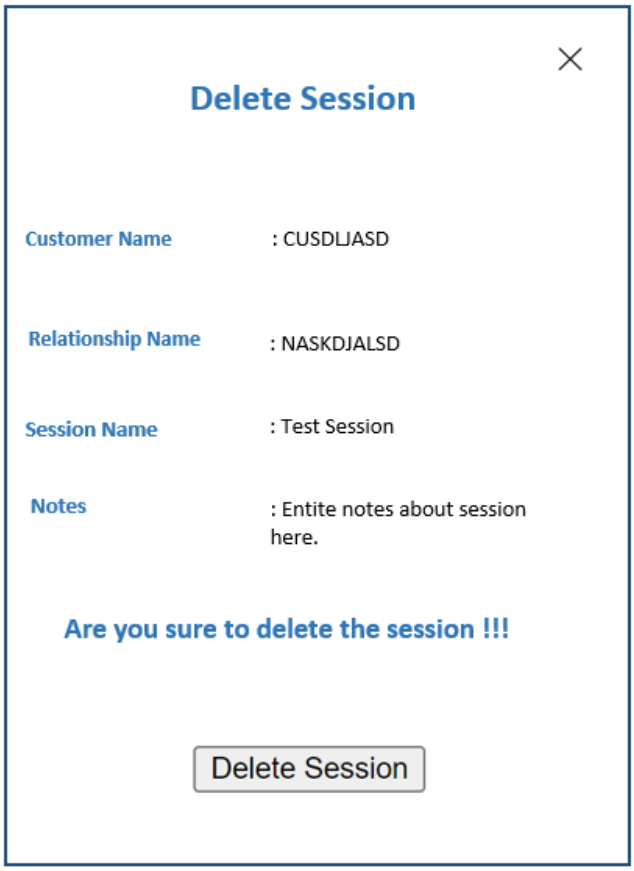
4. Edit Session



5. View Session



6. Delete Session



APIs:

- Create a Session
- Edit/Update Session
- View all active sessions (include view single session)
- Delete a session
- Archive a session
- View all archived sessions (include view single session)
- Park a session
- View all parked sessions (include view single session)

<List down tables>

Stage 4: Coding Standards and benchmarks

- Proper exception handling
- No critical SonarQube issues/blocker
- Proper branch naming conventions and should work with different branches. Pull Requests to be raised and proper merge should happen.
- Unit test coverage should be > 90%
- Every API should have corresponding Postman collection, and this also needs to be pushed to the branch along with the code.
- Design API Gateway and try accessing the APIs via API Gateway rather than exposing the APIs to public. Implement JWT tokenization concept. – *Optional step*
- Logging should be taken care. Proper logging should be done in any ELK with correlation ID.
- Pagination should be handled.
- APIs should be documented using swagger with standards. API naming conventions should be taken care of
- API response time should not take more than 500ms.
- Naming conventions of the objects, classes, methods should be of high priority.
- Don’t keep any strings in the application rather move them to a separate constant file. Project structure should be crisp and optimal.
- No code repetitions. Group the common code together into a generic usage
- Utilize Java 8 features.
- Make the code configurable wherever possible. If utilizing the sql queries, make them configurable to avoid repetitive code changes.