IIHT

Online Shopping App

Contents

[1 Problem Statement 2](#_Toc102731155)

[2 Proposed Online Shopping App Wireframe 2](#_Toc102731156)

[3 Application Architecture 2](#_Toc102731157)

[4 Cloud Architecture 3](#_Toc102731158)

[5 Tool Chain 4](#_Toc102731159)

[6 Development flow 6](#_Toc102731160)

[7 Business-Requirement: 6](#_Toc102731161)

[8 Rubrics/Expected Deliverables 7](#_Toc102731162)

[8.1 Rest API (Products & Frameworks -> Compute & Integration): 7](#_Toc102731163)

[8.2 Database (Products & Frameworks -> Database & Storage): 8](#_Toc102731164)

[8.3 Maven (Tooling): 8](#_Toc102731165)

[8.4 Messaging (Products & Frameworks -> Compute & Integration): 8](#_Toc102731166)

[8.5 Log/ Monitoring (Products & Frameworks -> Governance & Tooling): 9](#_Toc102731167)

[8.6 Debugging & Troubleshooting 9](#_Toc102731168)

[9 Frontend 9](#_Toc102731169)

[10 Platform 9](#_Toc102731170)

[10.1 Compute 9](#_Toc102731171)

[10.2 Compute, Identity & Compliance, Security& Content Delivery 10](#_Toc102731172)

[10.3 FrontEnd Deployment 10](#_Toc102731173)

[11 Methodology 10](#_Toc102731174)

[11.1 Agile 10](#_Toc102731175)

# Problem Statement

Online Shopping Application allows the customers to register and login and search products. Admin shall view the orders placed and add/delete any products to the online system and update the product availability status

Guest users cannot place any orders.

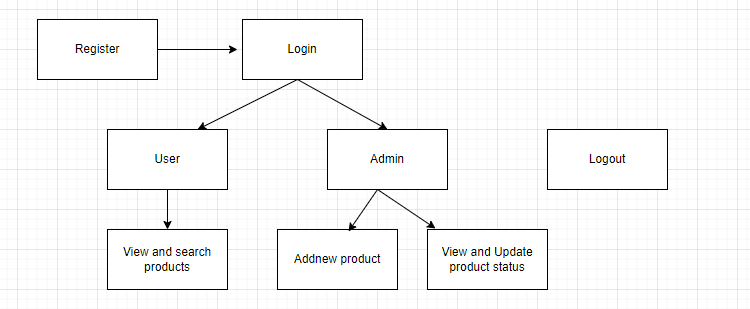
The core modules of online shopping app are:

1. Customer registration and login
2. Search the products
3. Admin can view the product and update the product status
4. Admin to add new products to the system

The scope includes developing the application using toolchain mentioned below.

# Proposed Online Shopping App Wireframe

* + - 1. UI needs improvisation and modification as per given use case.



# Application Architecture

Sonarcube

index.html

App Module (Root Module)

Root Component

Online Shopping App Component

Code Quality (Engineering & Quality)

Online App Services

Presentation (Products & Frameworks)

GIT

Docker

Spring Boot Rest API

Kafka Producer

Junit

Kafka Consumer

Logstash

Compute & Integration (Products & Frameworks)

OpenAPI/Swagger

Actuator

Compute & Integration (Products & Frameworks)

Prometheus/Grafana

DB

Governance & Tooling   
(Products & Frameworks)

Database & Storage (Products & Frameworks)

# Cloud Architecture

Below Diagram shows how Online Application App can be deployed on AWS Cloud

Diagram

Description automatically generated with low confidence

# Tool Chain

|  |  |  |
| --- | --- | --- |
| Competency | Skill | Skill Detail |
| Engineering Mindset | Networking and Content Delivery |  |
|  | Ways of Working |  |
|  | Consulting Mindset |  |
|  | DevOps |  |
| Programming Languages | Application Language | Java |
| Products & Frameworks | Presentation | Angular |
|  |  | Karma & Jasmine |
|  | Compute & Integration | Spring Boot |
|  |  | Kafka |
|  |  | Docker |
|  | Database & Storage | MongoDB |
|  | Governance & Tooling | Git |
|  |  | Maven |
|  |  | Junit |
|  |  | Mockito |
|  |  | Logstash |
|  |  | Prometheus & Grafana |
| Engineering Quality | Code Quality | Sonar Cube |
| Platform | Cloud Tools | AzureACI (Container Services) |
|  |  | Azure CosmosDB/SQL DB |
|  |  | Azure Automation |
|  |  | Azure Redis Cache/Storage |
|  |  | Azure DevOps/Pipeline |
|  |  | Azure API Gateway |
|  |  | Azure Load Balancer |
|  |  | Azure Notification Hubs |
|  |  | Azure Functions |

# Development flow

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No** | **MC** | **Competency** | **Section** | **Indicative**  **Mechanism for Evaluation ( Passing score of 60% in each MC)** | **Points to Note** |
| [**Business Requirement**](#_Business-Requirement:) | | | | |  |
| **1** | **Backend** | Rest API, Database, Messaging, Log/Monitoring, Debugging & Troubleshooting | [Click here](#_Rubrics/Expected_Deliverables) | **Code Submission and Evaluation, Panel Presentation** | It is mandatory to complete this MC with 60% to proceed with the next Frontend MC |
| **2** | **Front End** | Angular/React | [Click here](#_Frontend) | **Code Submission and Evaluation, Panel Presentation** | It is mandatory to complete each MC with 60% to proceed with the next Cloud MC |
| **3** | **Cloud** | Compute, Identity, Compliance, Security and Content Delivery | [Click here](#_Platform) | **Code Submission and Evaluation, Panel Presentation** | It is mandatory to complete this Cloud MC with 60% |

# Business-Requirement:

As an application developer, develop frontend, middleware and deploy the Online shopping App with below guidelines:

|  |  |  |
| --- | --- | --- |
| User  Story # | User Story Name | User Story |
| US\_01 | Registration and Login | As a customer I should be able to login/Register in the online shopping application  Acceptance criteria:   1. A logged-in customer can reset their password so they can login, even if they forget their password. 2. A logged-in user:    1. Cannot change their username.    2. Can logout from their account. 3. As a customer I should be able to furnish following details at the time of registration    1. First Name    2. Last Name    3. Email    4. Login Id    5. Password    6. Confirm Password    7. Contact Number 4. All details fields must be mandatory 5. Login Id and Email must be unique 6. Password and Confirm Password must be same 7. If any constraint is not satisfied, validation message must be shown |
| US\_02 | View &Search Products | As a customer I should be able to view all the products and search any particular products as well  Acceptance criteria:   * 1. Customer can view all the existing products.   2. Customer can search any particular products based on the product names |
| US\_03 | Add new Products to the system | As an admin I should be able to add products to the system.  Acceptance criteria:   * 1. Add new products to the system   2. Below are the details to be added * Product Name * Product Description * Price * Features * Product status |
| US\_04 | View order details and Update product status | As an admin I should be able to view products and update the product status  Acceptance criteria:   * 1. View number of orders placed for a particular product   2. Check the product quantity available   3. If the quantity is 0, update the product status as ‘OUT OF STOCK’, else update as ‘HURRY UP TO PURCHASE’ |

# Rubrics/Expected Deliverables

## Rest API (Products & Frameworks -> Compute & Integration):

* 1. Use Spring Boot to version and implement the REST endpoints.
  2. Implement HTTP methods like GET, POST, PUT, DELETE, PATCH to implement RESTful resources:

|  |  |  |
| --- | --- | --- |
| **POST** | **/api/v1.0/shopping/register** | **Register as new customer** |
| **GET** | **/api/v1.0/ shopping /login** | **Login** |
| **GET** | **/api/v1.0/ shopping /<customername>/forgot** | **Forgot password** |
| **GET** | **/api/v1.0/ shopping /all** | **View all products** |
| **GET** | **/api/v/1.0/shopping/products/search/productname\*** | **Search by product name** |
| **POST** | **/api/v1.0/ shopping /<productname>/add** | **Add any new product** |
| **PUT** | **/api/v1.0/shopping /<productname>/update/<id>** | **Update product status** |
| **DELETE** | **/api/v1.0/ shopping /<productname>/delete/<id>** | **Delete product** |

* 1. **\*productname may be partial or complete username**
  2. Use necessary configuration in place for REST API in application.properties or bootstrap.properties or application.yml; whichever is applicable.
  3. Package Structure for Spring Boot Project will be like com. shoppingapp. \* With proper naming conventions for package and beans.
  4. Use configuration class annotated with @Configuration and @Service for business layer.
  5. Use constructor-based dependency injection in few classes and setter-based dependency injection in few classes.
  6. Follow Spring Bean Naming Conventions

## Database (Products & Frameworks -> Database & Storage):

1. As an application developer:
   1. Implement ORM with Spring Data MongoRepositoryand MongoDB. For complex and custom queries, create custom methods and use @Query, Aggregations (AggregationOperation, MatchOperation, AggregationResults), implementation of MongoTemplateetc as necessary.
   2. Have necessary configuration in place for REST API in application.properties or bootstrap.properties or application.yml OR Java based configuration; whichever is applicable.

## Maven (Tooling):

* + - 1. As an application developer:
         1. Create the spring boot project using Maven CLI
         2. Generate Surefire test reports and share it as a part of deliverables
         3. Using Maven CLI generate the project documentation, and share it as a part of deliverables

## Messaging (Products & Frameworks -> Compute & Integration):

1. As an application developer:
   1. Have a centralized logging system
   2. Be able to communicate using a messaging infrastructure.
   3. Use KafkaTemplate for communication with Springboot and topics in kafka.
   4. Use kafka for messaging infrastructure and implement admin to read the number orders placed for products and available quantity from system and admin to write the product availability status as provided in the user story
   5. Configure Springboot app to log all logging messages to kafka.
   6. Configure all kafka related configuration needed for Spring Boot in \*.properties or \*.yml file.

## Log/ Monitoring (Products & Frameworks -> Governance & Tooling):

1. As an application developer:
   1. Containerize the complete application, which includes front-end, middleware and kafka (consumers and producers) using docker and Dockerfile.
   2. Use .dockerignore as necessary to avoid containerizing un-necessary packages.
   3. Integrate Spring Boot Actuator with Prometheus and Grafana to monitor middleware.
   4. Implement logs with logstash.
   5. Open the preconfigured Logstash in Kibana and check if it successfully connect to Elasticsearch Server.
   6. Perform unit and integration testing of your application and do proper CI/CD

## Debugging & Troubleshooting

1. Generate bug report & error logs - Report must be linkedwith final deliverableswhich should also suggest the resolutionfor the encountered bugs and errors.

# Frontend

* + - 1. Develop the front end for all user stories.
      2. Implement using either Angular or React
      3. Implement all the Front-End validation rules
      4. Proper naming conventions and folder structures
      5. Implement using proper SOLID design principles
      6. Perform unit and integration testing for the front end application

# 11. Platform

## Compute

* + - 1. Use Azure CLIfor container management and deployment of Asp.Net Core application. You should be able to explain and demonstrate the same in interview.
      2. Use NoSQL instance of Azure CosmosDB as a database for the Tweet Application

## 11.2. COMPUTE, Identity & Compliance, Security& Content Delivery

1. Use Azure Automation and AzureCosmos to build a backend process for handling requests for Tweet App.
2. Use Serverless C# Container using Azure ACI and run the tweet app created with Asp.NetCore inside AzureAutomation.
3. Use Azure API Gateway to expose the Azure functions built in the previous step to be accessible on public internet.
4. Use AzureLB to configure the auto-scaling container instances.
5. Configure Azure Notification Hubs to issue messages whenever a Azure LB scales-up and scale-down container instances

Note – Minimum two rest endpoints should be hosted in cloud

# 12Methodology

## 1 Agile

1. As an application developer, use project management tool along to update progress as you start implementing solution.
2. Participants should have used estimation techniques, such as Function points, story points, T-Shirt sizing for User Story point estimation
3. As an application developer, the scope of discussion with mentor is limited to:
   1. Q/A
   2. New Ideas, New feature implementations and estimation.
   3. Any development related challenges
   4. Skill Gaps
   5. Any other pointers key to UI/UX and Middleware Development