IIHT

Time To Complete: 10 to 12 hr

E-Auction Application

Contents

[1 Problem Statement 2](#_Toc74924873)

[2 PROPOSED E-AUCTIONAPPLICATON WIREFRAME 3](#_Toc74924874)

[3 Tool Chain 4](#_Toc74924875)

[4 Business Requirements: 5](#_Toc74924876)

[5 Proposed Rest Endpoints to be exposed 8](#_Toc74924877)

[5.1 Rest APIs: 8](#_Toc74924878)

[6 Rubrics/Expected Deliverables 8](#_Toc74924879)

[7 Implementation/Functional Requirements 9](#_Toc74924880)

[7.1 Product and Frameworks: 9](#_Toc74924881)

[7.2 Governance and Tooling: 10](#_Toc74924882)

[7.3 Code Quality/Optimizations 10](#_Toc74924883)

[8 Platform 11](#_Toc74924884)

[8.1 Cloud Specific Design 11](#_Toc74924885)

[8.2 Design Specification - 1 11](#_Toc74924886)

[8.3 Design specification - 2 11](#_Toc74924887)

[8.4 Design specification – 3 12](#_Toc74924888)

[9 Methodology 12](#_Toc74924889)

[9.1 Agile 12](#_Toc74924890)

# Problem Statement

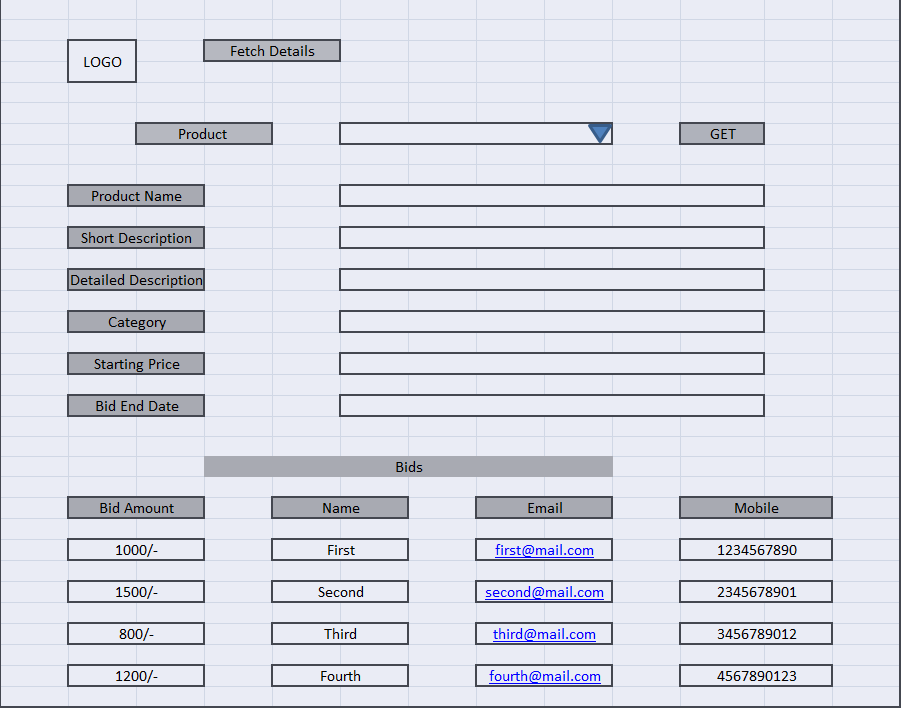
**E-Auction Application** is microservice based Cloud Native Application. The Main purpose of **E-Auction** is to provide the ability to initiate sale, purchase and bid for a product by the customer. The core modules of E-Auction app are:

* Put a new product on auction with starting bid price.
* Bid for the product.
* Update the bid price.
* Enlisting of bids placed on any product.

The scope includes developing the application using tool chain mentioned below.

# PROPOSED E-AUCTIONAPPLICATON WIREFRAME

Below is the wireframe for reference.



# Tool Chain

|  |  |  |
| --- | --- | --- |
| Competency | Skill | Skill Detail |
| Engineering Mindset | Networking and Content Delivery |  |
|  | Ways of Working |  |
|  | Consulting Mindset |  |
|  | DevOps |  |
|  | Secure Coding |  |
|  | Code Quality |  |
| Programming Languages | Application Language | C# |
| Products & Frameworks | Presentation | Angular/React |
|  |  | Javascript/Typescript |
|  |  | Bootstrap |
|  | Compute & Integration | Asp.net Core |
|  |  | ELK Stack |
|  |  | Kafka/Rabbit MQ/Active MQ |
|  |  |  |
|  | Governance & Tooling | Git |
|  |  | Mockito |
|  |  | Nunit |
|  |  | Protractor/Cypress |
| Engineering Quality |  |  |
| Platform | Cloud Tools | Azure ACI |
|  |  | Azure VM |
|  |  | Azure Container Registry |
|  |  | Azure Log Analytics |
|  |  | Azure API Gateway |
|  |  | Azure App service |
|  |  | Azure Notification Hubs |
|  |  | Azure Service Bus |
|  |  | Azure Load Balancer |
|  |  | Azure Web Apps |
|  |  | Azure AD |

# 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, Non-functional considerations | [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 Requirements:

Below are the user stories for the given problem statement

|  |  |  |  |
| --- | --- | --- | --- |
| **User**  **Story #** | **User Story Name** | **User Story** | **Development** |
| US\_01 | Seller Functionalities | As a seller I can add a new product for auction  *While adding a product, following information is required.*   * Product Name * Short Description * Detailed Description * Category * Starting Price * Bid end date   It also requires seller information as follows:  Personal Information   * First Name * Last Name * Address * City * State * Pin * Phone * Email   Constraints:   1. Product Category should be the one from the existing. (Predefined Hard Coded)    1. Painting    2. Sculptor    3. Ornament 2. Starting price should be number 3. Bid end date should be future date, else throw a custom exception in such case   Validations:   1. Product Name is not null, min 5 and max 30 characters. 2. firstName is not null, min 5 and max 30 characters. 3. lastName is not null, min 3 and max 25 characters. 4. email is not null, and it should be valid email pattern, containing a single @. 5. mobile is not null, min 10 and max 10 character and all must be numeric. | Only API to be developed |
| US\_02 | Seller Functionalities | As a seller I can delete a product added for auction  Constraints:   1. If product is tried to be deleted after the bid end date, it must throw a custom exception. 2. When product is tried to be deleted and if at least one bid is placed on that product, throw a custom exception. | Only API to be developed |
| US\_03 | Buyer Functionalities | As a buyer I am able to bid for a product  *While bidding for a product, following information is required from buyer.*   * First Name * Last Name * Address * City * State * Pin * Phone * Email * Product Id * Bid Amount   Precondition:   1. ProductId must be of an existing product available for auction (Can get from Database)   Constraints:   1. If bid is placed after the bid end date, it must throw a custom exception. 2. More than one bid on a product by same user (based on email ID) is not allowed, must throw a custom exception in such case.   Validations:   1. firstName is not null, min 5 and max 30 characters. 2. lastName is not null, min 3 and max 25 characters. 3. email is not null, and it should be valid email pattern, containing a single @. 4. mobile is not null, min 10 and max 10 character and all must be numeric. | Only API to be developed |
| US\_04 | Seller functionality | As a seller I can list all bids received on product put for auction  Considerations:   1. While fetching the above details the API must return the product details (Short Description, Detailed Description, Category, starting price, Bid End Date) along with all bids placed on it. 2. Bid details must be in descending order of bid amount. | API and Frontend to be developed |
| US\_05 | Buyer functionality | As a buyer I am able to update the Bid Amount of my bidding  Constraints:   1. If bid Amount is updated after the bid end date, it must throw a custom exception | Only API to be developed |

# Proposed Rest Endpoints to be exposed

## Rest APIs:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | **URL Exposed** | **Purpose** | | /e-auction/api/v1/seller/add-product | Adds a new product | | / e-auction/api/v1/seller/show-bids/{productId} | Fetches details of all bids on product with product details | | / e-auction/api/v1/seller/delete/{productId} | Deletes the product | |
| |  |  | | --- | --- | | **URL Exposed** | **Purpose** | | /e-auction/api/v1/buyer/place-bid | Places a bid for a product | | /e-auction/api/v1/buyer/update-bid/{productId}/{buyerEmailld}/{newBidAmount} | Updates the bid amount | |

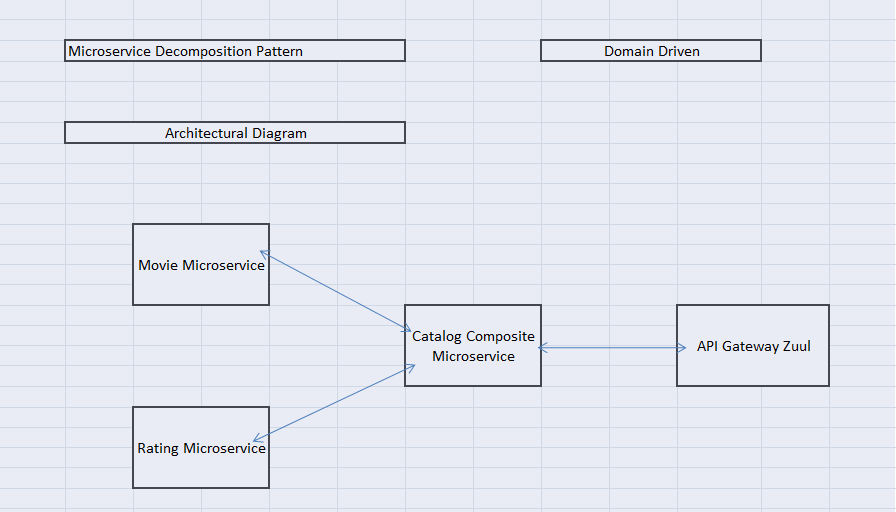
# Rubrics/Expected Deliverables

**Architecture definition**

You are expected to share a design document describing the below. It may contain diagram, flowcharts etc. You may use a presentation tool like PowerPoint or Word Document or simple text file.

* The Design approach should follow SOLID principles.
* Identify the design patterns and application of the same should be clearly stated in the document.
* Architectural assumptions, dependencies, and risks should be clearly stated with mitigation for the same.
* Capture how the Non-Functional requirements like Security, Performance, Availability, Scalability and Resilient are achieved.

**A Reference representation can be as follows:**

****

# Implementation/Functional Requirements

## Product and Frameworks:

1. **Compute and Integration**

Develop the backend application as a microservice architecture. (Implementation as follows)

* + 1. Identify the best decomposition pattern and create microservice based on that (mention the architecture of same in design document)
    2. Identify the best Database Deployment pattern for use case and implement it (mention the same in design document)
    3. Integrate a message broker in your microservice (Kafka, RabbitMQ or ActiveMQ) to implement CQRS pattern. Implement it for posting a bid (User-Story 3: Command Part) and getting the bid Listing (User-Story-4: Query part)
    4. Use any one of the Creational Design patterns for composing the model object to be sent back as response on following endpoint:

/ e-auction/api/v1/report/show-bids/{productId}

(Fetches details of all bids on product with product details. Mention the patterns used in design document and also specify the reason for selecting the one)

* + 1. Optimize you REST endpoints to allow filtering, sorting, and pagination.
    2. Document REST endpoints with OpenAPI or Swagger
    3. Expose all rest Endpoints using a common API Gateway.
    4. Implement service discovery and circuit breaker pattern in microservice architecture.
    5. Implement at least 2 Security OWASP recommendations in your Asp.net Core applications.
    6. Use Spring batch to load initial data to the DB and update the reference digaram also

## Governance and Tooling:

* + - 1. Follow the practise of Creating Testable Component
      2. Configure your frontend application to implement End-To-End Testing using either Protractor or Cypress.
      3. Test suites must contain exception situation testing.
      4. Generate the Code Coverage report of the same.
      5. Implement logging in your end-to-end testing and report the same using either ELK stack.
      6. Perform Regression testing and develop performance test suite
      7. Perform unit and integration testing of your application and do proper CI/CD

## Code Quality/Optimizations

1. Associates should have written clean code that is readable.
2. Associates need to follow SOLID programming principles.

# Frontend

Develop the frontend for User Story 4. (Implementation as follows)

1. Implement using either Angular or React.
2. Implement any one of the Gang of four Patterns to compose data using typescript before presenting the same on UI.
3. Implement at least one approach for UI performance consideration.
4. Identify and Implement client-side Optimization Techniques for Bootstrap.
5. Implement the prevention of XSS cross-site security threats for frontend application.
6. Implement using proper SOLID design principles
7. Perform unit and integration testing for the front end application

# Platform

## Cloud Specific Design

* + - 1. Update the messaging service (Kafka/RabbitMQ/ActiveMq/KubeMQ) with Cloud based service viz. Azure Service Bus.
      2. The backend service for the API should be implemented using any of the NoSQL service supported on Azure.
      3. As per the below specification create an elaborated architecture diagram of Cloud Deployment in cloud section of design document.

## Design Specification - 1

Use User Story-2 from the Business Requirements to implement the below.

1. Implement this user stories as serverless using Azure Functions.
2. Use Azure API Gateway to expose the Azure functions built in the previous step to be accessible on public internet.
3. The process should be passed to Azure Service Bus Queue and Notification (Email Service) should be triggered.
4. Configure the concurrency of your serverless implementation to run 100 instances of your functions in parallel.
5. Configure your application to be able to handle Azure function service exceptions.
6. Add a provision to encrypt the data in transit in your Azure functions implementation.
7. Configure Azure Notification hub to issue messages whenever Azure function is triggered.

## Design specification - 2

Use User Story-3 from the Business Requirements to implement the below.

1. Deploy user Story-3 using App service and configure security group of the Container instance created by Azure App service.
2. Configure the process of recovering instances in the situation of failover.
3. Configure Azure AD to secure the Rest Endpoints exposed by the above implementation.

## Design specification – 3

Use all other user-stories/microservice and frontend application from the Business Requirements to implement the below.

* + - 1. Use Azure API Gateway to publish the API.
      2. Containerize this requirement on ACI.
      3. Chalk out a plan to set up recovery and backup of the VM.
      4. Configure ACR to register logging actions with Azure Log analytics.
      5. Configure your ACI repository to prevent over writing of image.
      6. Configure Azure AD to secure the Rest Endpoints.
      7. Use Azure LB to configure the auto-scaling container instances.

# Methodology

## Agile

1. As an application developer, use project management tool along to update progress as you start implementing solution.
2. 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