System Requirements Specification Index

For

Online Auction

System

Version 1.0

IIHT Pvt. Ltd.

fullstack@iiht.com

Table of Contents

[1 Project Abstract 3](#_Toc77850439)

[2 Assumptions, Dependencies, Risks / Constraints 4](#_Toc77850440)

[2.1 Political Party Constraints: 4](#_Toc77850441)

[2.2 Political Leader Constraints 4](#_Toc77850442)

[3 Business Validations 5](#_Toc77850445)

[4 Rest Endpoints 6](#_Toc77850446)

[4.1 PoliticalPartyController 6](#_Toc77850447)

[4.2 PoliticalLeaderController 6](#_Toc77850448)

[4.3 DevelopmentController 7](#_Toc77850449)

[5 Template Code Structure 8](#_Toc77850451)

[5.1 Package: com.iiht.training.auction 8](#_Toc77850452)

[5.2 Package: com.iiht.training.auction.entity 8](#_Toc77850453)

[5.3 Package: com.iiht.training.auction.dto 9](#_Toc77850454)

[5.4 Package: com.iiht.training.auction.model.exception 10](#_Toc77850456)

[5.5 Package: com.iiht.training.auction.repository 10](#_Toc77850457)

[5.6 Package: com.iiht.training.auction.service 11](#_Toc77850458)

[5.7 Package: com.iiht.training.auction.service.impl 12](#_Toc77850459)

[5.8 Package: com.iiht.training.auction.exception 13](#_Toc77850460)

[5.9 Package: com.iiht.training.auction.controller 15](#_Toc77850460)

[6 Considerations 16](#_Toc77850461)

[7 Execution Steps to Follow 16](#_Toc77850462)

**Political Parties APPLICATION**

**System Requirements Specification**

# Project Abstract

**Online Auction System** Application is Spring boot RESTful application with MySQL, where it allows the sellers to Manage Products, Customers can place a bid on the products before the last date of the bidding.

**Following is the requirement specifications**:

|  |  |
| --- | --- |
|  | Online Auction System |
|  |  |
| Modules |  |
| 1 | Seller |
| 2 | Customer |
|  |  |
|  |  |
| Seller Module Functionalities |  |
|  |  |
| 1 | Register Itself |
| 2 | Can add a new product based on predefined categories |
| 3 | Can delete a product |
| 4 | Get Seller by id |
| 5 | Fetch all registered sellers |
| 6 | Delete an existing Seller |
| 7 | Can View details of bids placed on a particular product |
| 8 | Can view list of all products added for selling |
|  |  |
| Customer Module Functionalities |  |
| 1 | Customer can register itself |
| 2 | Customer can update its information |
| 3 | Get customer by Id |
| 4 | Fetch all registered customers |
| 5 | Get All the Products |
| 6 | Get the product by id |
| 7 | Can view all product placed for bidding based on category |
| 8 | Customer can Place a bid on specific product |
| 9 | Customer can view the all bids placed on a product (only after last date) |
|  |  |
|  |  |

# Assumptions, Dependencies, Risks / Constraints

## Seller Constraints:

* While deleting the seller details, if sellerId does not exist then operation should throw custom exception.
* While fetching the Seller details by id, if sellerId does not exist then operation should throw custom exception.
* While fetching the Product details by id, if productId does not exist then operation should throw custom exception.
* While deleting the Product details, if productId does not exist then operation should throw custom exception

## Customer Constraints

* While deleting a customer, if id does not exist then operation should throw custom exception.
* While fetching the customer details by id, if id does not exist then operation should throw custom exception.
* While placing a bid if customer , if id does not exists then operation should throw custom exception.

**Common Constraints**

* For all rest endpoints receiving @RequestBody, validation check must be done and must throw custom exception if data is invalid
* All the business validations must be implemented in dto classes only.
* All the database operations must be implemented on entity object only
* Do not change, add, remove any existing methods in service layer
* In Repository interfaces, custom methods can be added as per requirements.
* All RestEndpoint methods and Exception Handlers must return data wrapped in **ResponseEntity**

# Business Validations

* Seller name is not null, min 3 and max 100 characters.
* Seller email is not null, min 3, max 100 characters and should be email format
* Seller address is not null, min 3 and max 100 characters.
* Seller phone number is not null, min 10 and max 10 digits only
* Product name is not null, min 3 and max 100 characters.
* Product description is not null, min 3 and max 100 characters.
* Product quantity is not null.
* Product start bidding amount is not null.
* Product price is not null
* Product last date of bidding is not null, it should be in ‘yyyy-mm-dd’ format and future date
* Product category is not null, min 3 and max 100 characters
* Product predefined categories should be [Mobiles, Electronics, Clothing, Home]
* Customer username is not null, min 3 and max 100 characters
* Customer password is not null, min 3 and max 100 characters
* Customer email is not null, min 3, max 100 characters and should be email format
* Customer phone number is not null, min 10 and max 10 digits only
* Customer address is not null, min 3 and max 100 characters

# Rest Endpoints

Rest End-points to be exposed in the controller along with method details for the same to be created

## SellerController

|  |  |
| --- | --- |
| **URL Exposed** | **Purpose** |
| 1. /sellers/register  |  |  | | --- | --- | | Http Method | POST | | Parameter 1 | SellerDto | | Return | SellerDto | | Register a seller |
| /sellers/update   |  |  | | --- | --- | | Http Method | PUT | | Parameter 1 | SellerDto | | Return | SellerDto | | Update a Seller |
| /sellers/get/all   |  |  | | --- | --- | | Http Method | GET | | Parameter 1 | - | | Return | List<SellerDto> | | Fetches the list of all registered Sellers |
| /sellers/get/{sellerId}   |  |  | | --- | --- | | Http Method | GET | | Parameter 1 | Long (sellerId) | | Return | SellerDto | | Fetches the details of a Seller |
| /sellers/delete/{sellerId}   |  |  | | --- | --- | | Http Method | DELETE | | Parameter 1 | Long (sellerId) | | Return | Boolean | | Delete a seller |
| /sellers/get/bids-on-product/{produtId}   |  |  | | --- | --- | | Http Method | GET | | Parameter 1 | Long (productId) | | Return | List<BidsDto> | | Get Bids on a Products |

## ProductController

|  |  |
| --- | --- |
| **URL Exposed** | **Purpose** |
| /products/register   |  |  | | --- | --- | | Http Method | POST | | Parameter 1 | ProductDto | | Return | ProductDto | | Register a Product |
| /products/update   |  |  | | --- | --- | | Http Method | PUT | | Parameter 1 | ProductDto | | Return | ProductDto | | Update the Product |
| /products/get/all   |  |  | | --- | --- | | Http Method | GET | | Parameter 1 | - | | Return | List<ProductDto > | | Fetches all saved Products |
| /products/get/{productId}   |  |  | | --- | --- | | Http Method | GET | | Parameter 1 | Long (productId) | | Return | ProductDto | | Fetch the details of a Product |
| /products/get/by-seller/{sellerId}   |  |  | | --- | --- | | Http Method | GET | | Parameter 1 | Long (sellerId) | | Return | List<ProductDto > | | Fetches the details of all the Products registered by a seller |
| /products/delete/{productId}   |  |  | | --- | --- | | Http Method | DELETE | | Parameter 1 | Long (productId) | | Return | Boolean | | Delete a Product |
| /products/get/by-category/{categoryId}   |  |  | | --- | --- | | Http Method | GET | | Parameter 1 | Long (categoryId) | | Return | List<ProductDto > | | Fetch the details of all the products registered under a category |

## CustomerController

|  |  |
| --- | --- |
| **URL Exposed** | **Purpose** |
| /customers/register   |  |  | | --- | --- | | Http Method | POST | | Parameter 1 | CustomerDto | | Return | CustomerDto | | Register a Customer |
| /customers/update   |  |  | | --- | --- | | Http Method | PUT | | Parameter 1 | CustomerDto | | Return | CustomerDto | | Update an existing Customer |
| /customers/get/all   |  |  | | --- | --- | | Http Method | GET | | Parameter 1 | - | | Return | List<CustomerDto > | | Fetches all the registered customers |
| /customers/get/{id}   |  |  | | --- | --- | | Http Method | GET | | Parameter 1 | Long(id) | | Return | List<CustomerDto> | | Fetch the details of a Customer |
| /customers/delete/{id }   |  |  | | --- | --- | | Http Method | DELETE | | Parameter 1 | Long(id) | | Return | Boolean | | Deletes an existing customer |
| /customers/place-bid   |  |  | | --- | --- | | Http Method | POST | | Parameter 1 | BidsDto | | Return | BidsDto | | Places a bid on the product by the customer |
| /customers/get/all-bids-on-product/{productId}   |  |  | | --- | --- | | Http Method | GET | | Parameter 1 | Long(productid) | | Return | List<BidsDto> | | Customer can get all the bids on a product after the bid ends. |

# Template Code Structure

## Package: com.iiht.training.auction

**Resources**

|  |  |  |
| --- | --- | --- |
| **OnlineAuctionSystemApplication (Class)** | This is the Spring Boot starter class of the application. | Already Implemented |

## Package: com.iiht.training.auction.entity

**Resources**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Class/Interface** | | **Description** | | **Status** |
| **SellerEntity (class)** | * + Annotate this class with proper annotation to declare it as an entity class with **sellerId** as primary key.   + Map this class with **sellers** table.   + Generate the **sellerId** using **IDENTITY** strategy | | Partially implemented. | |
| **ProductEntity(class)** | * + This is class is partially implemented.   + Annotate this class with proper annotation to declare it as an entity class with **productId** as primary key.   + Map this class with **products** table.   + Generate the **productId** using the **IDENTITY** strategy | | Partially implemented. | |
| **CustomerEntity(class)** | * + This is class is partially implemented.   + Annotate this class with proper annotation to declare it as an entity class with **id** as primary key.   + Map this class with **customers** table.   + Generate the **id** using the **IDENTITY** strategy | | Partially implemented. | |
| **BidsEntity(class)** | * + This is class is partially implemented.   + Annotate this class with proper annotation to declare it as an entity class with **id** as primary key.   + Map this class with **bids** table.   + Generate the **id** using the **IDENTITY** strategy | | Partially implemented. | |

## Package: com.iiht.training.auction.dto

**Resources**

|  |  |  |
| --- | --- | --- |
| **Class/Interface** | **Description** | **Status** |
| **SellerDto (class)** | Use appropriate annotations from the **Java Bean Validation API** for validating attribute of this class. (Refer **Business Validation** section for validation rules). | Partially implemented. |
| **ProductDto (class)** | Use appropriate annotations from the **Java Bean Validation API** for validating attribute of this class. (Refer **Business Validation** section for validation rules). | Partially implemented. |
| **CustomerDto (class)** | Use appropriate annotations from the **Java Bean Validation API** for validating attribute of this class. (Refer **Business Validation** section for validation rules). | Partially implemented. |
| **BidsDto (class)** | Use appropriate annotations from the **Java Bean Validation API** for validating attribute of this class. (Refer **Business Validation** section for validation rules). | Partially implemented. |

## Package: com.iiht.training.auction.model.exception

**Resources**

|  |  |  |
| --- | --- | --- |
| **Class/Interface** | **Description** | **Status** |
| **ExceptionResponse (class)** | Object of this class is supposed to be returned in case of exception through exception handlers | Already implemented. |

## Package: com.iiht.training.auction.repository

**Resources**

|  |  |  |
| --- | --- | --- |
| **Class/Interface** | **Description** | **Status** |
| **SellerRepository (interface)** | 1. Repository interface exposing CRUD functionality for **SellerEntity** Entity. 2. You can go ahead and add any custom methods as per requirements | Partially implemented |
| **ProductRepository (interface)** | 1. Repository interface exposing CRUD functionality for **ProductEntity** Entity. 2. You can go ahead and add any custom methods as per requirements | Partially implemented |
| **CustomerRepository (interface)** | 1. Repository interface exposing CRUD functionality for **Customer** Entity. 2. You can go ahead and add any custom methods as per requirements | Partially implemented |
| **BidsRepository (interface)** | 1. Repository interface exposing Bids functionality for **Bids** Entity. 2. You can go ahead and add any custom methods as per requirements | Partially implemented |

## Package: com.iiht.training.auction.service

**Resources**

|  |  |  |
| --- | --- | --- |
| **Class/Interface** | **Description** | **Status** |
| **SellerService (interface)** | Interface to expose method signatures for political party related functionality.  Do not modify, add or delete any method | Already implemented. |
| **ProductService (interface)** | Interface to expose method signatures for political leader related functionality.  Do not modify, add or delete any method | Already implemented. |
| **CustomerService (interface)** | Interface to expose method signatures for Developments related functionality.  Do not modify, add or delete any method | Already implemented. |
| **BidsService (interface)** | Interface to expose method signatures for Developments related functionality.  Do not modify, add or delete any method | Already implemented. |

## Package: com.iiht.training.auction.service.impl

**Resources**

|  |  |  |
| --- | --- | --- |
| **Class/Interface** | **Description** | **Status** |
| **SellerServiceImpl (class)** | * Implements **SellerService**. Contains template method implementation. * Need to provide implementation for seller related functionalities * Add required repository dependency * Do not modify, add or delete any method signature | To be implemented. |
| **ProductServiceImpl (class)** | * Implements **ProductService**. Contains template method implementation. * Need to provide implementation for product related functionalities * Add required repository dependency * Do not modify, add or delete any method signature | To be implemented. |
| **CustomerServiceImpl (class)** | * Implements **CustomerService**. Contains template method implementation. * Need to provide implementation for Customer related functionalities * Add required repository dependency * Do not modify, add or delete any method signature | To be implemented. |
| **BidsServiceImpl (class)** | * Implements **BidsService**. Contains template method implementation. * Need to provide implementation for **Bids** related functionalities * Add required repository dependency * Do not modify, add or delete any method signature | To be implemented. |

## Package: com.iiht.training.auction.exception

**Resources**

|  |  |  |
| --- | --- | --- |
| **Class/Interface** | **Description** | **Status** |
| **GlobalHandler (class)** | * RestControllerAdvice Class for defining global exception handlers. * Contains Exception Handler for **InvalidDataException** class. * Use this as a reference for creating exception handler for other custom exception classes | Partially implemented. |

|  |  |  |
| --- | --- | --- |
| **Class/Interface** | **Description** | **Status** |
| **SellerNotFoundException (Class)** | * Custom Exception to be thrown when trying to fetch or delete the seller info which does not exists. * Need to create Exception Handler for same wherever needed (local or global) | Already created. |
| **ProductNotFoundException (Class)** | * Custom Exception to be thrown when trying fetch or delete Product info which does not exists. * Need to create Exception Handler for same wherever needed (local or global) | Already created. |
| **CustomerNotFoundException (Class)** | * Custom Exception to be thrown when trying to fetch or delete a Customer info which does not exists. * Need to create Exception Handler for same wherever needed (local or global) | Already created. |

## Package: com.iiht.training.auction.controller

**Resources**

|  |  |  |
| --- | --- | --- |
| **Class/Interface** | **Description** | **Status** |
| **SellerController (Class)** | * Controller class to expose all rest-endpoints for Seller related activities. * May also contain local exception handler methods | To be implemented |
| **ProductController (Class)** | * Controller class to expose all rest-endpoints for Product related activities. * May also contain local exception handler methods | To be implemented |
| **CustomerController (Class)** | * Controller class to expose all rest-endpoints for Customers related activities. * May also contain local exception handler methods | To be implemented |

# Considerations

1. There are 2 roles in this application

|  |
| --- |
| Seller |
| Customer |

1. You can perform the following 4 possible actions

|  |
| --- |
| Seller Actions |
| Product Actions |
| Customer Actions |
| Bids on Products |

# Execution Steps to Follow

1. **Open the tools terminal from Terminal option**
2. **On command prompt, cd into your project folder (cd <Your-name-hashkey>)**
3. **To build your project and run test cases use command:**

**mvn clean package**

1. **To launch your application, move into target folder (cd target). Run the following command to run the application:**

**java -jar online-auction-system.jar**

1. **Default credentials for MySQL:**
   1. **Username: petclinic**
   2. **Password: password**
   3. **Default Database: petclinic**
2. **To login to mysql instance: Open the mysql terminal from Terminal option and use following command:**

**mysql -upetclinic -ppassword**

1. **Mandatory: Before final submission run the following command:**

**mvn test**