
System Requirements Specification Index

For

Online Auction System

Version 1.0

IIHT Pvt. Ltd.
fullstack@iiht.com

TABLE OF CONTENTS

1	Project Abstract	3
2	Assumptions, Dependencies, Risks / Constraints	4
2.1	Seller Constraints:	4
2.2	Customer Constraints	4
3	Business Validations	5
4	Rest Endpoints	6
4.1	SellerController	6
4.2	ProductController	6
4.3	CustomerController	7
5	Template Code Structure	8
5.1	Package: com.iiht.training.auction	8
5.2	Package: com.iiht.training.auction.entity	8
5.3	Package: com.iiht.training.auction.dto	9
5.4	Package: com.iiht.training.auction.model.exception	10
5.5	Package: com.iiht.training.auction.repository	10
5.6	Package: com.iiht.training.auction.service	11
5.7	Package: com.iiht.training.auction.service.impl	12
5.8	Package: com.iiht.training.auction.exception	13
5.9	Package: com.iiht.training.auction.controller	15
6	Considerations	16
7	Execution Steps to Follow	16

Online Auction APPLICATION

System Requirements Specification

1 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)

2 ASSUMPTIONS, DEPENDENCIES, RISKS / CONSTRAINTS

2.1 SELLER CONSTRAINTS:

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

2.2 CUSTOMER CONSTRAINTS

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

2.3 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**

3 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

4 REST ENDPOINTS

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

4.1 SELLERCONTROLLER

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

4.2 PRODUCTCONTROLLER

URL Exposed		Purpose
/products/register		Register a Product
Http Method	POST	
Parameter 1	ProductDto	
Return	ProductDto	

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

4.3 CUSTOMERCONTROLLER

URL Exposed		Purpose
/customers/register		Register a Customer
Http Method	POST	
Parameter 1	CustomerDto	
Return	CustomerDto	
/customers/update		Update an existing Customer
Http Method	PUT	
Parameter 1	CustomerDto	
Return	CustomerDto	
/customers/get/all		Fetches all the registered customers

Http Method	GET		
Parameter 1	-		
Return	List<CustomerDto >		
/customers/get/{id}			Fetch the details of a Customer
Http Method	GET		
Parameter 1	Long(id)		
Return	List<CustomerDto>		
/customers/delete/{id }			Deletes an existing customer
Http Method	DELETE		
Parameter 1	Long(id)		
Return	Boolean		
/customers/place-bid			Places a bid on the product by the customer
Http Method	POST		
Parameter 1	BidsDto		
Return	BidsDto		
/customers/get/all-bids-on-product/{productId}			Customer can get all the bids on a product after the bid ends.
Http Method	GET		
Parameter 1	Long(productId)		
Return	List<BidsDto>		

5 TEMPLATE CODE STRUCTURE

5.1 PACKAGE: COM.IIHT.TRAINING.AUCTION

Resources

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

5.2 PACKAGE: COM.IIHT.TRAINING.AUCTION.ENTITY

Resources

Class/Interface	Description	Status
SellerEntity (class)	<ul style="list-style-type: none"> Annotate this class with proper annotation to declare it as an entity class with sellerId as primary key. 	Partially implemented.

	<ul style="list-style-type: none"> o Map this class with sellers table. o Generate the sellerId using IDENTITY strategy 	
ProductEntity(class)	<ul style="list-style-type: none"> o This class is partially implemented. o Annotate this class with proper annotation to declare it as an entity class with productId as primary key. o Map this class with products table. o Generate the productId using the IDENTITY strategy 	Partially implemented.
CustomerEntity(class)	<ul style="list-style-type: none"> o This class is partially implemented. o Annotate this class with proper annotation to declare it as an entity class with id as primary key. o Map this class with customers table. o Generate the id using the IDENTITY strategy 	Partially implemented.
BidsEntity(class)	<ul style="list-style-type: none"> o This class is partially implemented. o Annotate this class with proper annotation to declare it as an entity class with id as primary key. o Map this class with bids table. o Generate the id using the IDENTITY strategy o 	Partially implemented.

5.3 PACKAGE: COM.IIHT.TRAINING.AUCTION.DTO

Resources

Class/Interface	Description	Status
SellerDto (class)	Use appropriate annotations from the Java Bean Validation API for validating attributes 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 attributes 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 attributes 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 attributes of this class. (Refer Business Validation section for validation rules).	Partially implemented.

5.4 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.

5.5 PACKAGE: COM.IIHT.TRAINING.AUCTION.REPOSITORY

Resources

Class/Interface	Description	Status
SellerRepository (interface)	<ol style="list-style-type: none">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)	<ol style="list-style-type: none">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)	<ol style="list-style-type: none">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)	<ol style="list-style-type: none">1. Repository interface exposing Bids functionality for Bids Entity.	Partially implemented

	2. You can go ahead and add any custom methods as per requirements	
--	--	--

5.6 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.

5.7 PACKAGE: COM.IIHT.TRAINING.AUCTION.SERVICE.IMPL

Resources

Class/Interface	Description	Status
SellerServiceImpl (class)	<ul style="list-style-type: none">● 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)	<ul style="list-style-type: none">● 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)	<ul style="list-style-type: none">● 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)	<ul style="list-style-type: none"> 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.
--------------------------------	--	--------------------

5.8 PACKAGE: COM.IIHT.TRAINING.AUCTION.EXCEPTION

Resources

Class/Interface	Description	Status
GlobalHandler (class)	<ul style="list-style-type: none"> 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)	<ul style="list-style-type: none"> Custom Exception to be thrown when trying to fetch or delete the seller info which does not exist. 	Already created.

	<ul style="list-style-type: none"> • Need to create Exception Handler for same wherever needed (local or global) 	
ProductNotFoundException (Class)	<ul style="list-style-type: none"> • Custom Exception to be thrown when trying to fetch or delete Product info which does not exist. • Need to create Exception Handler for same wherever needed (local or global) 	Already created.
CustomerNotFoundException (Class)	<ul style="list-style-type: none"> • Custom Exception to be thrown when trying to fetch or delete a Customer info which does not exist. • Need to create Exception Handler for same wherever needed (local or global) 	Already created.

5.9 PACKAGE: COM.IIHT.TRAINING.AUCTION.CONTROLLER

Resources

Class/Interface	Description	Status
SellerController (Class)	<ul style="list-style-type: none">• Controller class to expose all rest-endpoints for Seller related activities.• May also contain local exception handler methods	To be implemented
ProductController (Class)	<ul style="list-style-type: none">• Controller class to expose all rest-endpoints for Product related activities.• May also contain local exception handler methods	To be implemented
CustomerController (Class)	<ul style="list-style-type: none">• Controller class to expose all rest-endpoints for Customers related activities.• May also contain local exception handler methods	To be implemented

6 CONSIDERATIONS

A. There are 2 roles in this application

Seller
Customer

B. You can perform the following 4 possible actions

Seller Actions
Product Actions
Customer Actions
Bids on Products

7 EXECUTION STEPS TO FOLLOW

1. All actions like build, compile, running application, running test cases will be through Command Terminal.
2. To open the command terminal the test takers, need to go to Application menu (Three horizontal lines at left top) -> Terminal -> New Terminal.
3. To build your project use command:
mvn clean package -Dmaven.test.skip
4. To launch your application, move into the target folder (**cd target**). Run the following command to run the application:
java -jar online-auction-system-0.0.1-SNAPSHOT.jar
5. This editor Auto Saves the code.
6. If you want to exit(logout) and continue the coding later anytime (using Save & Exit option on Assessment Landing Page) then you need to use **CTRL+Shift+B**-command compulsorily on code IDE. This will push or save the updated contents in the internal git/repository. Else the code will not be available in the next login.
7. These are time bound assessments the timer would stop if you logout and while logging in back using the same credentials the timer would resume from the same time it was stopped from the previous logout.
8. To test any Restful application, the last option on the left panel of IDE, you can find ThunderClient, which is the lightweight equivalent of POSTMAN.
9. This is a web-based application, to run the application on a browser, use the internal browser in the workspace. Click on the second last option on the left panel of IDE, you can find Browser Preview, where you can launch the application.

Note: The application will not run in the local browser

10. Default credentials for MySQL:
 - a. Username: **root**
 - b. Password: **pass@word1**
11. To login to mysql instance: Open new terminal and use following command:
 - a. **sudo systemctl enable mysql**
 - b. **sudo systemctl start mysql**
 - c. **mysql -u root -p**

The last command will ask for password which is 'pass@word1'

12. Mandatory: Before final submission run the following command:

mvn test

13. You need to use **CTRL+Shift+B** - command compulsorily on code IDE, before final submission as well. This will push or save the updated contents in the internal git/repository, and will be used to evaluate the code quality.

HEALTH CLUB

IIHT

Time To Complete: 10 to 12 hr

CONTENTS

- 1 Problem Statement
- 2 Proposed Health Club Wireframe
 - 2.1 Welcome page
 - 2.2 View Appointments
 - 2.3 Place Appointment
 - 2.4 QUERY
- 3 Business-Requirement:
- 4 Constraints
- 5 Mandatory Assessment Guidelines

1 PROBLEM STATEMENT

Health Club is SPA (Single Page Application) for placing a request for appointment with personal physical trainer, manage appointments and dropping a query

The core modules of Health Club app are:

1. Welcome Page
2. Apply for an appointment
3. View Appointment
4. Query

2 PROPOSED HEALTH CLUB WIREFRAME

UI needs improvisation and modification as per given use case and to make test cases passed.

2.1 WELCOME PAGE

Logo				
	Home	View Appointments	Place Appointment	Contact Us
Introduction Text (any)				
	Useful Links		Contact	
	Home		Company address	
	View Appointments		Company email	
	Place Appointment		Company phone	
	Contact Us		Company fax	
©2021 Copyright GET-FIT Health Club				

2.2 VIEW APPOINTMENTS

Logo									
	Home	View Appointments	Place Appointment	Contact Us					
<u>S.No.</u>	Name	Phone	email	Age	Complete Address	Trainer Preference	Physio Required	Package	Total Amount
		Useful Links				Contact			
		Home				Company address			
		View Appointments				Company email			
		Place Appointment				Company phone			
		Contact Us				Company fax			
©2021 Copyright									

2.3 PLACE APPOINTMENT

Logo	Home View Appointments Place Appointment Contact Us			
Name			Age	
Email			Mobile No.	
Address Line 1				
Address Line 2				
City			State	
Country			Pin Code	
Trainer Preference	<input type="radio"/> Male Trainer <input type="radio"/> Female Trainer <input type="radio"/> No Preference			
Do you need Physiotherapist	<input type="radio"/> Yes <input type="radio"/> No			
Select a package	<input type="radio"/> One time appointment (Rs. 500/-) <input type="radio"/> 4 sessions per week (Rs. 400/- per session) <input type="radio"/> 5 sessions per week (Rs. 300/- per session)			
Weeks	2			
Amount(Rs)				
			Submit	
Useful Links			Contact	
Home			Company address	
View Appointments			Company email	
Place Appointment			Company phone	
Contact Us			Company fax	
©2021 Copyright				

2.4 QUERY

Logo				
	Home	View Appointments	Place Appointment	Contact Us
Drop us a message				
	Your Name *	Your Message *		
	Your Email *			
	Your Phone *			
		Send		
	Useful Links		Contact	
	Home		Company address	
	View Appointments		Company email	
	Place Appointment		Company phone	
	Contact Us		Company fax	
©2021 Copyright				

3 BUSINESS-REQUIREMENT:

As an application developer, develop the Health Club App (Single Page App) with below guidelines:

User Story #	User Story Name	User Story
US_01	Welcome Page	<p>As a user I should be able to visit the welcome page as default page.</p> <p>Acceptance criteria:</p> <ol style="list-style-type: none">1. User can click any button given in the menu bar.
US_02	Post Appointment	<p>As a user I should be able to post an appointment</p> <p>Acceptance criteria:</p> <ol style="list-style-type: none">1. As a user I should be able to furnish following details at the time of placing an appointment<ol style="list-style-type: none">1.1 Name1.2 Age1.3 Email1.4 Mobile No1.5 Address Line 11.6 Address Line 21.7 City1.8 State1.9 Country1.10 Pin Code1.11 Trainer Preference1.12 Physiotherapist requirement (Yes or No)1.13 Select a package1.14 Weeks1.15 Amount2. Weeks number type input box should be visible when 2nd or 3rd package option is selected.3. If physiotherapist is required add additional 200/- in final amount4. Amount should be disabled and should be calculated automatically based on the selected package.2. All details fields must be mandatory.3. Address line 2 may contain the same address as address line 1.4. Email& Mobile must be unique.5. If any constraint is not satisfied, a validation message must be shown.6. A success or failure message should be visible after the submit button is clicked.
US_03	Manage Appointment	<ol style="list-style-type: none">1. As a user I should be able to view all appointment requests, and after selecting any appointment <p>Acceptance criteria:</p> <ol style="list-style-type: none">1. Message should be visible if no appointment is available to show.
US_04	Query	<p>As a user I should be able to post a feedback/query/message</p> <p>Acceptance criteria:</p>

		<ol style="list-style-type: none"> 1. As a user I should be able to furnish following details at the time of filling contact us form <ol style="list-style-type: none"> a. Name b. Email c. Phone d. Message 2. Message should not go beyond 200 characters. 3. All four fields must be mandatory. 4. A success or failure message should be visible after the submit button is clicked.
--	--	---

4 CONSTRAINTS

1. On the page load, input focus must come to the first name input field.
1. You should be able to press the "TAB" key and "SHIFT + TAB" to navigate from top field to bottom field and vice-versa.
1. On click of "Submit" button, appointment details must be saved via fake-rest API in health-club.json.
2. Fake rest api is implemented with json-server.

Example JSON for reference of fields to be used for placing appointment:

```
{
  "firstname": "test",
  "lastname": "test",
  "age": 24,
  "phonenumber": 9988776655,
  "email": "test@test.com",
  "streetaddress": "test",
  "city": "test",
  "state": "test",
  "country": "india",
  "pincode": 560058,
  "trainerpreference": "Male Trainer",
  "physiotherapist": "Yes",
  "packages": "500",
```

```
"inr": 1000,  
"paisa": 10,  
"id": 1  
}
```

5 MANDATORY ASSESSMENT GUIDELINES

1. All actions like build, compile, running application, running test cases will be through Command Terminal.
2. To open the command terminal the test takers, need to go to Application menu (Three horizontal lines at left top) -> Terminal ->New Terminal.
3. This editor Auto Saves the code.
4. If you want to exit(login) and continue the coding later anytime (using Save & Exit option on Assessment Landing Page) then you need to use CTRL+Shift+B-command compulsorily on code IDE. This will push or save the updated contents in the internal git/repository. Else the code will not be available in the next login.
5. These are time bound assessments the timer would stop if you logout and while logging in back using the same credentials the timer would resume from the same time it was stopped from the previous logout.
6. To test any Restful application, the last option on the left panel of IDE, you can find ThunderClient, which is the lightweight equivalent of POSTMAN.
7. This is a web-based application, to run the application on a browser, use the internal browser in the workspace. Click on the second last option on the left panel of IDE, you can find Browser Preview, where you can launch the application.
Note: The application will not run in the local browser
8. You can follow series of command to setup React environment once you are in your project-name folder:
 - a. npm install -> Will install all dependencies -> takes 10 to 15 min

- b. `npm run start` -> To compile and deploy the project in browser. You can press `<Ctrl>` key while clicking on `localhost:8080/8081` to open project in browser -> takes 2 to 3 min
 - c. `npm run json-server` -> to deploy fake rest api created with json-server -> takes 10 to 15 seconds
 - d. `npm run test` -> to run all test cases. **It is mandatory to run this command before submission of workspace** -> takes 5 to 6 min
9. You may also run “`npm run test`” while developing the solution to re-factor the code to pass the test-cases.
10. You might need to write class names for passing test cases.
11. Once you are done with development and ready with submission, you may navigate to the previous tab and submit the workspace. It is mandatory to click on **“Submit Assessment”** after you are done with code.
12. You need to use `CTRL+Shift+B` - command compulsorily on code IDE, before final submission as well. This will push or save the updated contents in the internal git/repository, and will be used to evaluate the code quality.