System Requirements Specification Index

For

Django Rest-API
Online Auction System
(Collaborative)

Version 1.0

IIHT Pvt. Ltd.
fullstack@iiht.com

TABLE OF CONTENTS

1. Project Abstract

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

Following are the requirement specifications:

	Online Auction System
Modules	
1	Seller
2	Customer
3	Product
4	Bids
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 i.e products based on seller id
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 by date

2. Assumptions, Dependencies, Risks / Constraints

2.1 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

2.2 Customer Constraints

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

3. Business Validations

- Seller name is max 100 characters.
- Seller email is max 100 characters and should be email format.
- Seller address is max 100 characters.
- Seller phone number max 10 digits only.
- Product name is max 100 characters.
- Product description is max 100 characters.
- Product last date should be in 'yyyy-mm-dd' format and future date.
- Product category is max 100 characters.
- Product predefined categories should be [Mobiles, Electronics, Clothing, Home]
- Customer username is max 100 characters.
- Customer password is max 100 characters.
- Customer email is max 100 characters and should be email format.
- Customer phone number is max 10 digits only.
- Customer address 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

Class Name	Method Name	Purpose Of Method
SellerView	get(self,request,pk=None,format=None)	Get Seller by id and Fetch all registered sellers
	post(self, request,format=None)	Seller Register Itself
	delete(self,request,pk,format=None)	Delete an existing Seller
ProductView	get(self,request,pk=None,format=None)	Get All the Products and Get the product by id
	post(self, request,format=None)	Can add a new product based on predefined categories
	delete(self,request,pk,format=None)	Can delete a product
GetProductView	get(self,request,pk=None,format=None)	To view list of all products added for selling i.e products based on seller id
ListProductsByCategoryView	get(self,request,pk=None,format=None)	To view all product placed for bidding based on category
CustomerView	get(self,request,pk=None,format=None)	Get customer by Id and Fetch all registered customers
	post(self, request,format=None)	Customer can register itself
	put(self,request,pk,format=None)	Customer can update its information (full update)
	patch(self,request,pk,format=None)	Customer can update its information (partial update)
BidsView	get(self,request,pk=None,format=None)	Customer can view the all bids placed on a product
	post(self, request,format=None)	Customer can Place a bid on specific product
BidsByDateView	get(self,request,pk=None,format=None)	Customer can view the all bids

		placed on a product by date

5. Template Code Structure

Resources (Models)

Class	Description	Status
SellerModel	o A model class for Seller. o It will map to the SellerModel table.	Already implemented.
ProductModel	o A model class for Product o It will map to the ProductModeltable.	Already implemented.
CustomerModel	o A model class for Customer. o It will map to the CustomerModel table.	Already implemented.
BidsModel	o A model class for Bids. o It will map to the BidsModeltable.	Already implemented.

Resources (Serializers)

Class	Description	Status
SellerSerializer	A serializer for SellerModel	Already implemented
ProductSerializer	A serializer for ProductModel	Already implemented
CustomerSerializer	A serializer for CustomerModel	Already implemented
BidsSerializer	A serializer for BidsModel	Already implemented

Resources (Views)

Class	Description	Status
SellerView	A class for the get, post, delete functionalitieson**SellerModel**model**.**	To be implemented

ProductView	A class for the get, post, delete functionalities on**ProductModel**model**.**	To be implemented
GetProductView	A class for the get functionalityon **ProductModel**model**.**	To be implemented
ListProductsByCategoryView	A class for the get functionality to list products by category**.**	To be implemented
CustomerView	A class for the get, post, put, patch functionalities on **CustomerModel**model**.**	To be implemented
BidsView	A class for the get, post functionalities on **BidsModel**model**.**	To be implemented
BidsByDateView	A class for the get functionality on **BidsModel**model to get bids details by date**.**	To be implemented

Resources (Exceptions)

Class	Description	Status
IdNotAvailable	Object of this exception class is supposed to be returned in case specified id is not available.	Already implemented.
InvalidData	Object of this exception class is supposed to be returned in case data is invalid.	Already implemented.

IdOrDateNotAvailable	Object of this exception class is supposed to be returned in case No Bids are available with specified product Id or Date.	Already implemented.
ProductNotAvailable	Object of this exception class is supposed to be returned in case Specified product is not available.	Already 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. The editor Auto Saves the code.
4. If you want to exit(logout) and to 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. To test any UI based application the second last option on the left panel of IDE, you can find Browser Preview, where you can launch the application.
8. Install 'djangorestframework' module before running the code. For this use the following command.

   pip install djangorestframework

9. Use the following command to run the server

   python3 manage.py runserver

10. Mandatory: Before final submission run the following commands to execute testcases

    python3 manage.py test auctionapp.test.test_functional

    python3 manage.py test auctionapp.test.test_exceptional

    python3 manage.py test auctionapp.test.test_boundary

11. To test rest end points

    Click on 'Thunder Client' or use Ctrl+Shift+R->Click on 'New Request' (at left side of IDE)

12. 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.

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.

------- * --------