System Requirements Specification Index

For

Django Rest-API
Home Insurance-App

Version 1.0

IIHT Pvt. Ltd.
fullstack@iiht.com

TABLE OF CONTENTS

HOME INSURANCEAPPLICATION
System Requirements Specification

1 Project Abstract

Home InsuranceApp is Django Rest API application with SQLite Database, where it allows any unregistered users (visitors) toregister themselves, Create Quote, retrieve an existing Quote, buy a policy, View policy and whereas admin can perform renew the policy and cancel the policyactivities.

Following is the requirement specifications:

	Home Insurance Application
Modules	
1	UserModel
2	QuoteModel
3	PolicyModel
UserModelModule Functionalities	
1	User Registration
QuoteModule Functionalities	
1	Create Quote if user exist
2	Retrieve an existing Quote by user id and quote id
Policy Module Functionalities	
1	Buy a Policy by quote id
2	View Policy by policykey
3	Renew Policyby policykey
4	Cancel Policyby policykey

2 Assumptions, Dependencies, Risks / Constraints

- While retrieving an existing Quote with quote_id and user_id if any one of these does not exist,then should throw custom exception**IdDoesNotExist**.
- While creating quote if user id does not exist, then should throw custom exception**UserDoesNotExist**.
- While buying policy if quote id does not exist, then should throw custom exception**QuoteIdDoesNotExist**.
- While showing policy if policy key does not exist, then should throw custom exception**PolicyKeyDoesNotExist**.
- While renew policy if policy key does not exist, then should throw custom exception**PolicyKeyDoesNotExist**.
- While cancel policy if policy key does not exist, then should throw custom exception**PolicyKeyDoesNotExist**.

3 Business Validations

1. Policy status can be Renewed or Cancelled, by default It can be None.
2. Date format should be 'YYYY-MM-DD'.

4 Rest Endpoints

Class Name	Method Name	Purpose Of Method
UserRegisterView	post(self, request,format=None)	To register the user.
QuoteView	get(self,request,pk=None,format=None)	To retrieve an existing quote.
	post(self, request,format=None)	To create the quote.
BuyPolicyView	post(self, request,format=None)	To buy a policy.
ShowPolicyView	get(self,request,pk=None,format=None)	To view a policy with policy key.
RenewPolicyView	patch(self,request,pk,format=None)	To renew a policy with policy key.
CancelPolicyView	patch(self,request,pk,format=None)	To cancel a policy with policy key.

Resources (Models)

Class	Description	Status
UserModel	o A model class for User. o It will map to the Usertable.	Already implemented.
QuoteModel	o A model class for Quote. o It will map to the Quotetable.	Already implemented.
PolicyModel	o A model class for Policy. o It will map to the Policytable.	Already implemented.

Resources (Serializers)

Class	Description	Status
UserSerializer	A serializer for UserModel	Already implemented
QuoteSerializer	A serializer for QuoteModel	Already implemented
PolicySerializer	A serializer for PolicyModel	Already implemented

Resources (Views)

Class	Description	Status
UserRegisterView	A class for post functionality on **UserModel**model**.**	To be implemented
QuoteView	A class for, get and post functionalitieson **QuoteModel**model**.**	To be implemented
BuyPolicyView	A class for post functionality on **PolicyModel**model**.**	To be implemented
ShowPolicyView	A class for get functionality on **PolicyModel**model**.**	To be implemented

RenewPolicyView	A class for patch functionality on **PolicyModel**model**.**	To be implemented
CancelPolicyView	A class for patch functionality on **PolicyModel**model**.**	To be implemented

Resources (Services)

Class	Description	Status
PolicyService	A service class for cancel the policy by policy key.	To be implemented.

Resources (Exceptions)

Class	Description	Status
IdDoesNotExist	Custom exception can be raised in case specified quoteid or user id does not exist.	Already implemented.
PolicyKeyDoesNotExist	Custom exception can be raised in case specified policy keydoes not exist.	Already implemented.
UserDoesNotExist	Custom exception can be raised in case specified user id does not exist.	Already implemented.
QuoteIdDoesNotExist	Custom exception can be raised in case specified quote id does not exist.	Already implemented.

6 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 insuranceapp.test.test_functional

    python3 manage.py test insuranceapp.test.test_exceptional

    python3 manage.py test insuranceapp.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.

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