

---

# System Requirements Specification Index

For

## Django Rest-API Home Insurance-App

Version 1.0

IIHT Pvt. Ltd.  
fullstack@iiht.com

## TABLE OF CONTENTS

|   |   |   |
|---|---|---|
| 1 | Project Abstract.....                               | 3 |
| 2 | Assumptions, Dependencies, Risks / Constraints..... | 4 |
| 3 | Business Validations.....                           | 4 |
| 4 | Rest Endpoints.....                                 | 4 |
| 5 | Template Code Structure.....                        | 5 |
| 6 | Execution Steps to Follow.....                      | 7 |

# 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) to register themselves, Create Quote, retrieve an existing Quote, buy a policy, View policy and whereas admin can perform renew the policy and cancel the policy activities.

**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               |
|--------------------|---|----------------------|
| <b>UserModel</b>   | <ul style="list-style-type: none"><li>o A model class for User.</li><li>o It will map to the Usertable.</li></ul>     | Already implemented. |
| <b>QuoteModel</b>  | <ul style="list-style-type: none"><li>o A model class for Quote.</li><li>o It will map to the Quotetable.</li></ul>   | Already implemented. |
| <b>PolicyModel</b> | <ul style="list-style-type: none"><li>o A model class for Policy.</li><li>o It will map to the Policytable.</li></ul> | Already implemented. |

**Resources (Serializers)**

| Class                   | Description                  | Status              |
|-------------------------|------------------------------|---------------------|
| <b>UserSerializer</b>   | A serializer for UserModel   | Already implemented |
| <b>QuoteSerializer</b>  | A serializer for QuoteModel  | Already implemented |
| <b>PolicySerializer</b> | A serializer for PolicyModel | Already implemented |

**Resources (Views)**

| Class                   | Description   | Status            |
|-------------------------|---|-------------------|
| <b>UserRegisterView</b> | A class for post functionality on <b>UserModel</b> model.             | To be implemented |
| <b>QuoteView</b>        | A class for, get and post functionalities on <b>QuoteModel</b> model. | To be implemented |
| <b>BuyPolicyView</b>    | A class for post functionality on <b>PolicyModel</b> model.           | To be implemented |
| <b>ShowPolicyView</b>   | A class for get functionality on <b>PolicyModel</b> model.            | To be implemented |

|                         |  |                   |
|-------------------------|--|-------------------|
| <b>RenewPolicyView</b>  | A class for patch functionality on <b>PolicyModel</b> model. | To be implemented |
| <b>CancelPolicyView</b> | A class for patch functionality on <b>PolicyModel</b> model. | To be implemented |

#### Resources (Services)

| Class                | Description  | Status             |
|----------------------|--|--------------------|
| <b>PolicyService</b> | A service class for cancel the policy by policy key. | To be implemented. |

#### Resources (Exceptions)

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

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. 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.
5. 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.
6. 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.
7. Install 'djangorestframework' module before running the code. For this use the following command.

`pip install djangorestframework`

8. Use the following command to run the server

`python3 manage.py runserver`

9. 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`

10. To test rest end points

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

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.

-----\*-----