**Real Estate Sharks**   
**Testing Documentation**

Date: 3/26/2022

Version: 1.0.0



Source (<https://www.dreamstime.com/shark-house-vector-logo-design-home-icon-image158864365>)

Table of Contents

1. **Project Description**.........................................................................................

**1.1 Project....................................................................................................................**

1. **Over-all testing Plan**........................................................................................

**2.1 Approach.................................................................................................................**

**2.2 Assignments............................................................................................................**

**2.3 Front-end................................................................................................................**

**2.4 Back-End.................................................................................................................**

**2.5 Django-REST-API.....................................................................................................**

1. **Static Testing**...................................................................................................
2. **Units Testing**...................................................................................................
3. **Integration Tests**.............................................................................................

**5.1 Integration Testing**

**5.1.1 GET**

**5.1.2 POST**

**5.1.3 DELETE**

**5.1.4 UPDATE**

1. **Validation Tests**...............................................................................................
2. **System Tests**....................................................................................................

**Section 1) Project description:**

**1.1 Project:** The overall Real Estate Sharks project will consist of two applications and an API for integration. The backend database will be built on DJango and SQLite. The API will be built on Django’s Restful-API Framework. The front end will be built on React.js and supporting libraries. The goal of the project is to create a client web application that small property managers can use to keep track of their infrastructure and tenants. This includes current and future tenants. The Client app will be using the Django app database to store and update information pertaining to the property managers' needs as described in the SDD documentation. The result will be a full-stack application running on a webserver host such as azure cloud or amazon AWS.

**Section 2) Over-all testing plan:**

**2.1 Approach:** Since the front and back ends are considered two separate applications, one being the React website, and the other being the Django and SQLite backend database, each one will be tested separately to ensure they both work independently. Thirdly we will be testing the Django-Restful-API using a web-based HTTP validator. For this, we will be using the thorough tests-driven approach, where we will be going through the following process.

1) Determine the specific functionality of the object/model, visualize it, then write the test case to verify the desired functionality.

2) Write enough code of the object/model to fail the designed test then make the minimal changes required to pass the test

3) Refactor the code to check for any redundancies or enhancements, then make sure the re-factoring still passes the test.

**2.2 Assignments:** As the backend and frontend work independently from each other, we will be testing our applications separately. Given that they are test-driven, all testing will be complete before the integration and deployment to a web host server.

**2.3 Front-end:** The react.js front-end website will be utilizing the testing framework Jest. Jest is a JavaScript test runner that lets you access the DOM via jsdom. The DOM is the Document Object Model, and it is the data representation of the objects that make up the content on the web pages. It is represented in HTML with embedded JSX. With this known, using Jest we can get a good approximation of how the browser works, it is not precise but often good enough for testing React components. Jest uses the document model tree to iterate through the specified components and their children.

**2.4 Back-End:** Python has a testing library for testing models, views, and serializers. Python’s “unittest” framework was originally inspired by JUnit and has a similar approach. It supports test automation, setup sharing, and independent testing. The testing is achieved by subclassing the python object and running test functions with the “assert” keyword (along with many others) on the object. For this project, we will be mostly using “assertEquals” to ensure the correct functionality. The SQLite database will be tested using the built-in python tests. For this, we will simply do a visual print test of our database to ensure we can add, get, update, and remove data from the database.

**2.5 Django-REST-API:** The Django Restful-API Framework will be test-driven using the built-in test-driven framework and a third-party tool such as Postman or hopscotch. Postman and Hopscotch are applications specifically used for the API testing. They are HTTP clients that test HTTP requests using a graphical user interface to “Stringify” the JSON object. They work directly with the back-end API in your browser to verify API requests and responses.

Django REST Framework automatically creates Browsable API documentation. Since we are using a one-to-one relationship between our API endpoints and models, we can directly test our project using the built-in testing framework. For this to work, we will be applying unit tests imported from the “TestCase” module.

**3. Static Testing:**

*Static tests include things like convention, code quality, code structure, etc.*

*Describe how you will conduct static testing on your finished code base. What will you test for? How will these tests be conducted and by whom? Justify your plan.*

*Code quality and convention*

*Review auto checker*

***3.1Static Testing****: The React Front side will have will have ill have separate functions that will simulate JSON responses from the Django rest framework API. Each of the different Sudo responses will have predetermined JSON objects that our class-based functions will determine whether the objects match the prearranged criteria and naming conventions as well as data types of the expected data. An example of this will be with the Get Unit endpoint that will send back a JSON object containing data pertaining to the Unit object Model. I will check if there is an address, living area, etc., and check the data types of the received data that it matches up and if not have an error example for each of the pieces if incorrect. I will do that for each subsequent endpoint and show examples of the error messages that the react system will produce.*

**4. Unit Tests:**

*Unit tests provide testing of individual modules, classes, or functions. This testing is best done as you develop.*

*Describe how you will break your tests into unit tests and plan out some of your unit tests.*

**Unit Testing:** The unit testing will be done by separate python scripts that will simulate front-end responses to said endpoints. We will be using the requests module from python to simulate an HTTP request, and for each of our different endpoints whether they be post, get, put, or delete we will have the respective data that, that endpoint needs if any expected in the python scripts each labeled based on the type of endpoint it is interacting with and separated into 3 different folders for the 3 separate sections of the project. We will also be utilizing the API Factory inside of the Django framework to simulate the responses for more advanced use cases such as user token authentication as well as a quick way to run all the suites at once. The Django internal testing will be separate from the Python requests tests that we will have but as all will interact with the entire system, they are considered unit tests.

5= checklist for later

**5. Integration Tests:**

**5.1 Integration Testing:** The project integration testing will be performed as top-down tests. Integration testing is testing where multiple modules or applications are tested together. For this, we will be combining the Django+SQLite App, Django Restful-API, and React Web App. With the top-down test strategy, we will be testing starting with the React web app to make API POST, GET, DELETE, and UPDATE HTTP requests. The API will then unpack the requests based on the sterilizers and views. Then the unpacked request will be handled by the backend App to manipulate the SQLite database. Some of the tests will be based on the HTTP requests and responses from a top-down perspective. These are as follows with respect to the UnitLists model. The UnitLists is the model for structuring the apartments data.

**5.1.1 GET:** When there is a desire for a user on the client app (React frontend) to see the apartment units' data (displayed as cards) either by looking at the whole database or using the search criteria as displayed on the “Apartments” page, A HTTP GET request will be sent to the Django API. The GET request (utilizing axios react.js module) will be received by the API and interpreted. Once read by the API, the API will get the requested data from the backend and package it into a JSON object based on the pre-defined serialization of the data. Once packaged the JSON object will be sent to the react front end and unpacked into properties (props in react), then displayed in a structured manner.

**5.1.2 POST:** The POST request is usually used in conjunction with an HTML form submission to add new data objects to the database. When an admin user wants to add a new apartment to the database, they will fill out the HTML from on the “Apartments” page and click submit. Then the form will be packaged into a JSON object and sent to the API. The API will unpack the JSON object and add it to the SQLite database.

**5.1.3 DELETE:** The DELETE request is usually used in conjunction with the unit's object model id. When an admin user wants to remove an apartment from the database, they will fill make a delete request on the “Apartments” page. That request will get sent to the API along with the id of the unit object. The API will have the backend delete the object from the database and return if the request was successful or not via a promise.

**5.1.4 UPDATE** When there is a desire for a user on the client app (React frontend) to update the apartment units' data, they will fill out an update form whose data entry fields will only be available to an admin and be used to package the info into a JSON object, then an HTTP UPDATE request will be sent to the Django API. The UPDATE request (utilizing axios react.js module) will be received by the API and interpreted. Once read by the API, the API will send the requested data change function call to the backend to be executed. Once executed and confirmed the API will let the react front end know via a POST call. The POST request will need to be sent to get the UnitList data again to be displayed as it has changed.

**6. Validation Tests: (CAVAN)**

Table of requirements and why met and why didn’t

Incl client review

*Describe your validation tests. At the least you’ll need a requirements review and client evaluation.*

**7. System Tests:**

*Describe your system tests if applicable. Provide at least 3.*

*Organize and present each of your test cases along with the results in a clear and logical manner. Where possible each test case should follow this template:*

* *Test ID*
* *Time/Date*
* *Requirements ID*
* *Tester*
* *Environment*
* *Goal*
* *Testing Procedure*
* *Expected results*
* *Actual results*
* *Status*
* *Comments*