**Test Plan**

Testing our application is one of the most important parts of the software development process. There are several different strategies that we plan to use, including unit testing, integration testing, and end-to-end testing.

Unit testing involves testing individual components and services in isolation without requiring a full application setup and helps ensure specific features work as expected. Integration testing involves testing how the different components and services operate together within the application. Lastly, end-to-end testing involves testing the entire application similar to how a user would interact with it. All in all, we plan to test each individual Angular component and service.

**Test Cases**

AC1.1 **App Component Test**

AC1.2 This test will ensure that the app component loads properly (includes TypeScript, HTML, and CSS code)

AC1.3 Deploy the application and ensure app component loads properly (along with index.html).

AC1.4 No inputs needed

AC1.5 Expect to see header and footer load properly with no errors

AC1.6 Normal

AC1.7 Whitebox

AC1.8 Functional

AC1.9 Unit

IC1.1 **Input Bar Component Test**

IC1.2 This test will ensure that the input bar component loads properly (includes TypeScript, HTML, and CSS code)

IC1.3 Deploy the application with the input bar component displayed in the main content div in the app component.

IC1.4 No inputs needed

IC1.5 Page loads with elements from test case(s) above with four select boxes, two buttons, and a mat-icon button.

IC1.6 Normal

IC1.7 Whitebox

IC1.8 Functional

IC1.9 Unit

TC1.1 **Table Component Test**

TC1.2 This test will ensure that the table component loads properly (includes TypeScript, HTML, and CSS code)

TC1.3 Deploy the application with the table component displayed in the input bar component below the other elements

TC1.4 No inputs needed (besides possibly test data to see if table works properly)

TC1.5 Table component displays a mat-table with the desired columns and test data (if included)

TC1.6 Normal

TC1.7 Whitebox

TC1.8 Functional

TC1.9 Integration

CP1.1 **CORS Proxy Test**

CP1.2 Ensure that the CORS proxy works and sets the necessary headers to allow CORS for a specified website

CP1.3 Run the CORS proxy server locally on any port. Using Postman or any other API platform, test API calls to the exoplanet archive using the CORS proxy. More specifically, try returning the ‘pscomppars’ table

CP1.4 Inputs: website URL (exoplanet archive), CORS proxy port

CP1.5 CORS proxy should add the necessary headers to the request and return the ‘pscomppars’ table with no errors. Request should not be blocked.

CP1.6 Normal

CP1.7 Blackbox

CP1.8 Functional

CP1.9 Integration

QC1.1 **Query Construction Test**

QC1.2 The purpose of this test is to ensure that a valid query and URL are generated by the site when the search button is clicked.

QC1.3 After choosing four random values for each of the select boxes, click the search button.

QC1.4 Inputs: four random values for the select boxes

QC1.5 Full URL + query should be outputted to the console in the correct format

QC1.6 Normal

QC1.7 Whitebox

QC1.8 Functional

QC1.9 Integration

DT1.1 **Data Transfer Test**

DT1.2 The purpose of this test is to check if exoplanet data is transferred properly from the data service to the other components.

DT1.3 User should enter their query in the select boxes and click search. After clicking search, the exoplanet data should be retrieved from the exoplanet archive through the CORS proxy by the data service. The data is then mapped to an exoplanet interface and is sent to the inputbar component to be displayed in the table component.

DT1.4 Inputs: example query

DT1.5 The exoplanet data should be retrieved from the archive and stored on the site. Output to console to check

DT1.6 Normal

DT1.7 Whitebox

DT1.8 Functional

DT1.9 Integration

DD1.1 **Data Display Test**

DD1.2 The purpose of this test is to check if the exoplanet data is properly displayed in the mat-table in the Table component.

DD1.3 After making sure the exoplanet data transfers between the components, add the data to a mat-table and ensure that all the data is loaded with one exoplanet per row.

DD1.4 Inputs: example query

DD1.5 A mat-table should generate after clicking search with the exoplanet data (one exoplanet per row)

DD1.6 Normal

DD1.7 Blackbox

DD1.8 Functional

DD1.9 Integration

ER1.1 **Expand Row Test**

ER1.2 The purpose of this test is to ensure that an exoplanet’s row is expanded after clicking on it.

ER1.3 Deploy the application and do a test search to display the table with data. After the data is shown, click on an exoplanet’s row and check if row is expanded.

ER1.4 Inputs: query for test search

ER1.5 After clicking on a row, it should expand with an animation and show additional details about that specific exoplanet.

ER1.6 Normal

ER1.7 Whitebox

ER1.8 Functional

ER1.9 Unit

ED1.1 **Exo-Detail Test**

ED1.2 The purpose of this test is to ensure that the exoplanet detail component is properly loaded with the appropriate data inside of an expanded row.

ED1.3 After deploying the application, initiate a search using a test query and expand the row of an individual exoplanet.

ED1.4 Inputs: query for test search

ED1.5 After expanding the row, the exoplanet detail component should load with an image of the discovery facility, a button to launch the NASA Eyes page for that exoplanet, and other additional details included in the exoplanet detail component.

ED1.6 Normal

ED1.7 Whitebox

ED1.8 Functional

ED1.9 Integration

WR1.1 **Window Resize Test**

WR1.2 The primary purpose of this test is to ensure the website displays properly on all types of screens and displays.

WR1.3 To test this, we will attempt to resize the browser window to make sure all the HTML elements continue to display properly and are responsive enough to be able to still function.

WR1.4 No inputs needed

WR1.5 When resizing the window, nothing should look especially glitchy or out of place. It is okay if some elements are off screen, however the user needs to be able to navigate to, access, and interact with those elements. The end goal is to have the site display properly on all types of displays (sizes, resolutions, etc.)

WR1.6 Boundary

WR1.7 Blackbox

WR1.8 Performance

WR1.9 Integration

TT1.1 **Time to Load Test**

TT1.2 The purpose of this test is to see how long it takes for the site to load after deployment.

TT1.3 Deploy the application and record the time it takes to load the page using the network and performance tabs in the Chrome dev tools.

TT1.4 No inputs are necessarily needed for this test

TT1.5 The expected result is a fast and responsive website that doesn’t take forever to load

TT1.6 Boundary

TT1.7 Blackbox

TT1.8 Performance

TT1.9 Integration

**Test Case Matrix**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Code | Normal/Abnormal | Blackbox/Whitebox | Functional/Performance | Unit/Integration |
| AC | Normal | Whitebox | Functional | Unit |
| IC | Normal | Whitebox | Functional | Unit |
| TC | Normal | Whitebox | Functional | Integration |
| CP | Normal | Blackbox | Functional | Integration |
| QC | Normal | Whitebox | Functional | Integration |
| DT | Normal | Whitebox | Functional | Integration |
| DD | Normal | Blackbox | Functional | Integration |
| ER | Normal | Whitebox | Functional | Unit |
| ED | Normal | Whitebox | Functional | Integration |
| WR | Boundary | Blackbox | Performance | Unit |
| TT | Boundary | Blackbox | Performance | Integration |