Testing Report

Ontario Academic Atlas

Introduction:

The Ontario Academic Atlas project aims to provide a comprehensive educational resource for students and educators in Ontario. As part of its development process, both unit testing and black box testing have been conducted to ensure the reliability, functionality, and usability of the system. This report presents an overview of the testing methodologies, results, issues identified, and recommendations for the Ontario Academic Atlas project.

Scope of Testing:

Unit Testing:

- Focuses on evaluating individual components, such as views, URLs, and database connections.
- Verifies the correctness and functionality of each component in isolation.

Black Box Testing:

- Evaluates the system's functionality from an end-user perspective.
- Tests various aspects of the application, including navigation, search functionality, data visualization, and user interaction.

Testing Methodology:

Unit Testing:

- Test cases designed based on requirements and specifications for each component.
- Systematic approach followed for setup, execution, and result verification.
- Positive and negative scenarios considered for comprehensive coverage.

Black Box Testing:

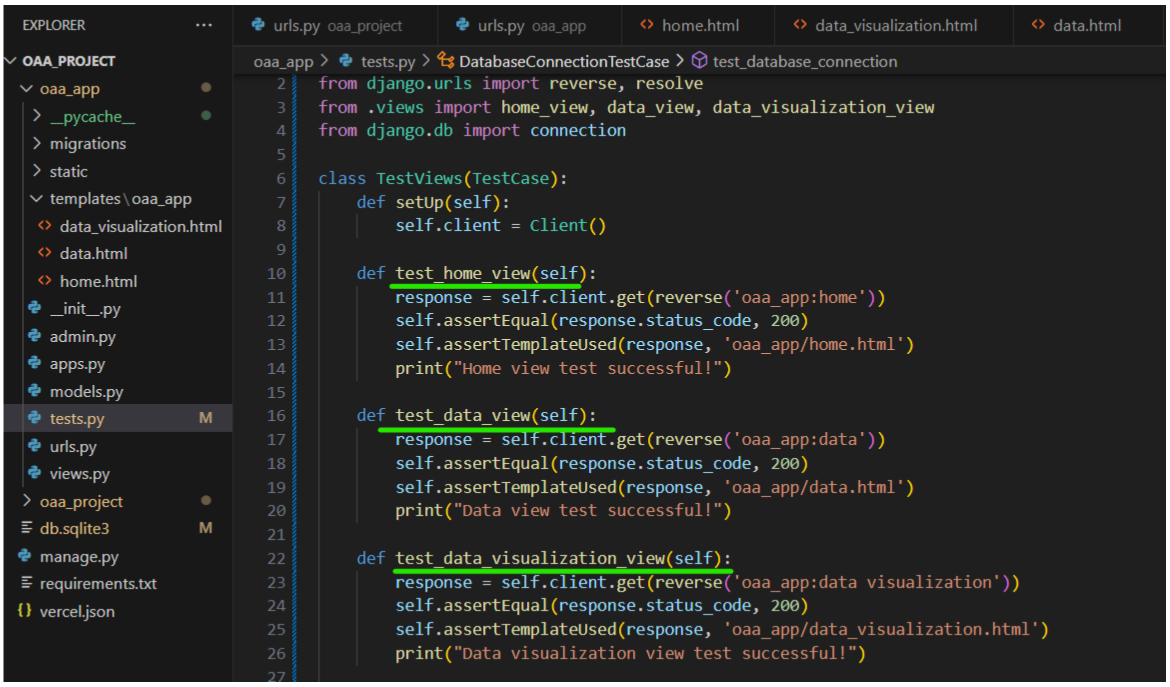
- Test scenarios derived from user stories and system requirements.
- Testing process includes test planning, execution, and reporting.
- Manual and automated testing techniques employed to maximize coverage and efficiency.

Test Results:

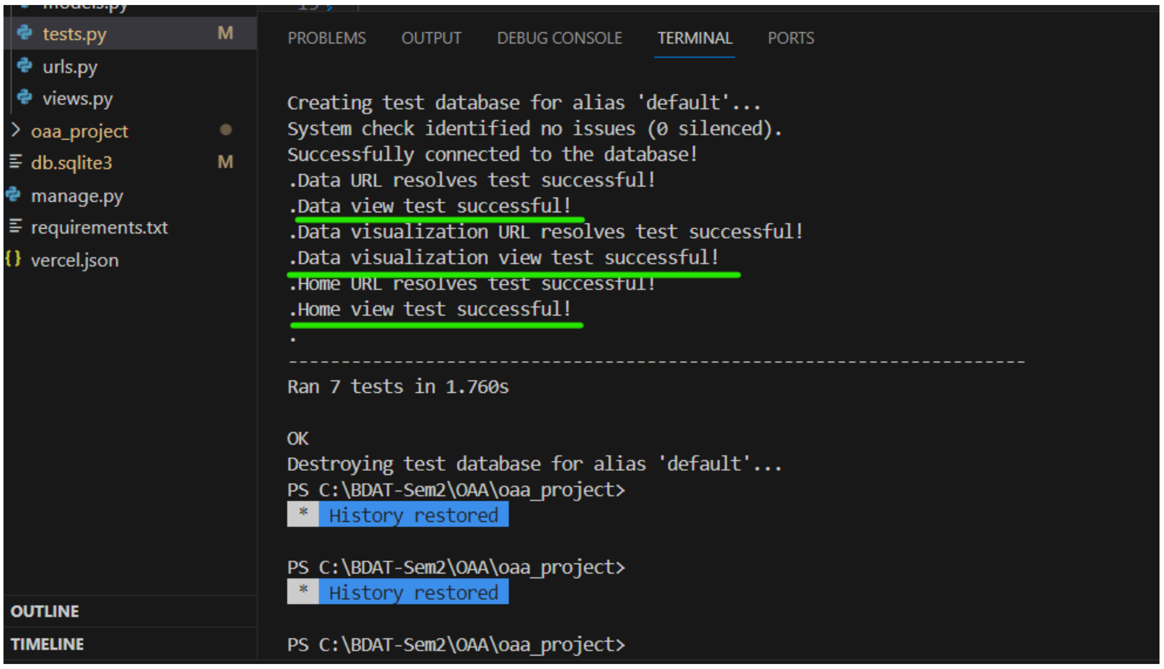
<u>Unit Testing:</u>

Views Testing:

- Home View: Renders successfully, returns status code 200, and utilizes the correct template ('home.html').
- Data View: Renders successfully, returns status code 200, and utilizes the correct template ('data.html').
- Data Visualization View: Renders successfully, returns status code 200, and utilizes the correct template ('data_visualization.html').



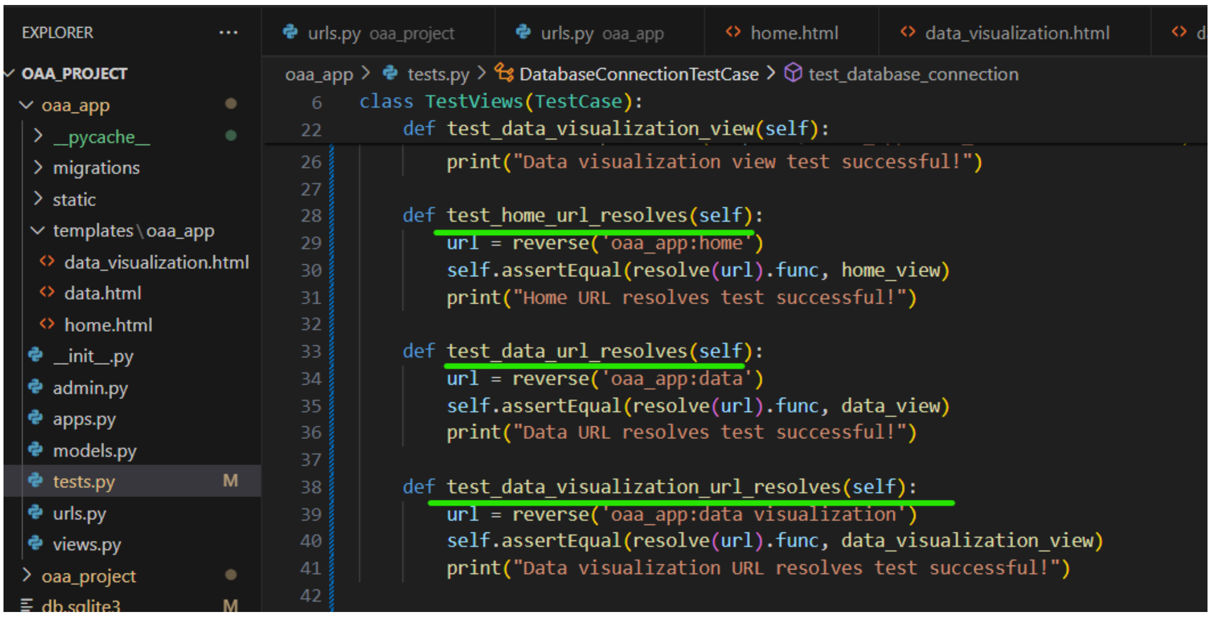
```python
from django.urls import reverse, resolve
from .views import home_view, data_view, data_visualization_view
from django.db import connection

class TestViews(TestCase):
    def setUp(self):
        self.client = Client()

    def test_home_view(self):
        response = self.client.get(reverse('oaa_app:home'))
        self.assertEqual(response.status_code, 200)
        self.assertTemplateUsed(response, 'oaa_app/home.html')
        print("Home view test successful!")

    def test_data_view(self):
        response = self.client.get(reverse('oaa_app:data'))
        self.assertEqual(response.status_code, 200)
        self.assertTemplateUsed(response, 'oaa_app/data.html')
        print("Data view test successful!")

    def test_data_visualization_view(self):
        response = self.client.get(reverse('oaa_app:data visualization'))
        self.assertEqual(response.status_code, 200)
        self.assertTemplateUsed(response, 'oaa_app/data_visualization.html')
        print("Data visualization view test successful!")
```



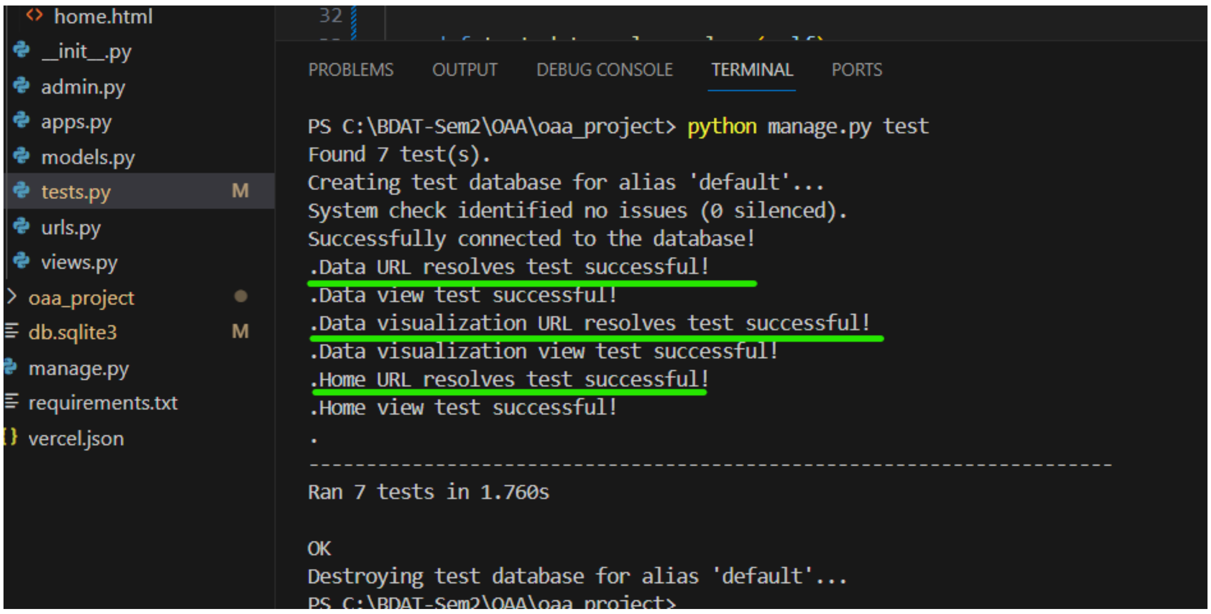
```
Creating test database for alias 'default'...
System check identified no issues (0 silenced).
Successfully connected to the database!
.Data URL resolves test successful!
.Data view test successful!
.Data visualization URL resolves test successful!
.Data visualization view test successful!
.Home URL resolves test successful!
.Home view test successful!
.
----------------------------------------------------------------------
Ran 7 tests in 1.760s

OK
Destroying test database for alias 'default'...
PS C:\BDAT-Sem2\OAA\oaa_project>
 *  History restored

PS C:\BDAT-Sem2\OAA\oaa_project>
 *  History restored

PS C:\BDAT-Sem2\OAA\oaa_project>
```

URLs Testing:

- Home URL Resolves: Resolves to the expected view function, 'home_view'.
- Data URL Resolves: Resolves to the expected view function, 'data_view'.
- Data Visualization URL Resolves: Resolves to the expected view function, 'data_visualization_view'.



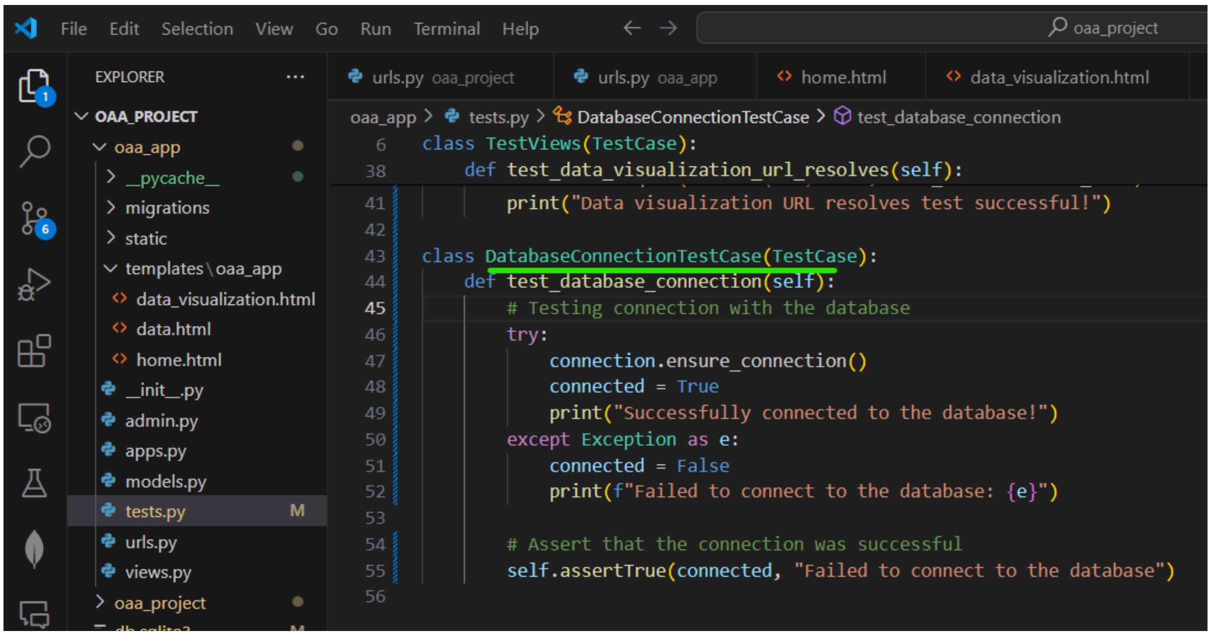
```
class TestViews(TestCase):
    def test_data_visualization_view(self):
        print("Data visualization view test successful!")

    def test_home_url_resolves(self):
        url = reverse('oaa_app:home')
        self.assertEqual(resolve(url).func, home_view)
        print("Home URL resolves test successful!")

    def test_data_url_resolves(self):
        url = reverse('oaa_app:data')
        self.assertEqual(resolve(url).func, data_view)
        print("Data URL resolves test successful!")

    def test_data_visualization_url_resolves(self):
        url = reverse('oaa_app:data_visualization')
        self.assertEqual(resolve(url).func, data_visualization_view)
        print("Data visualization URL resolves test successful!")
```



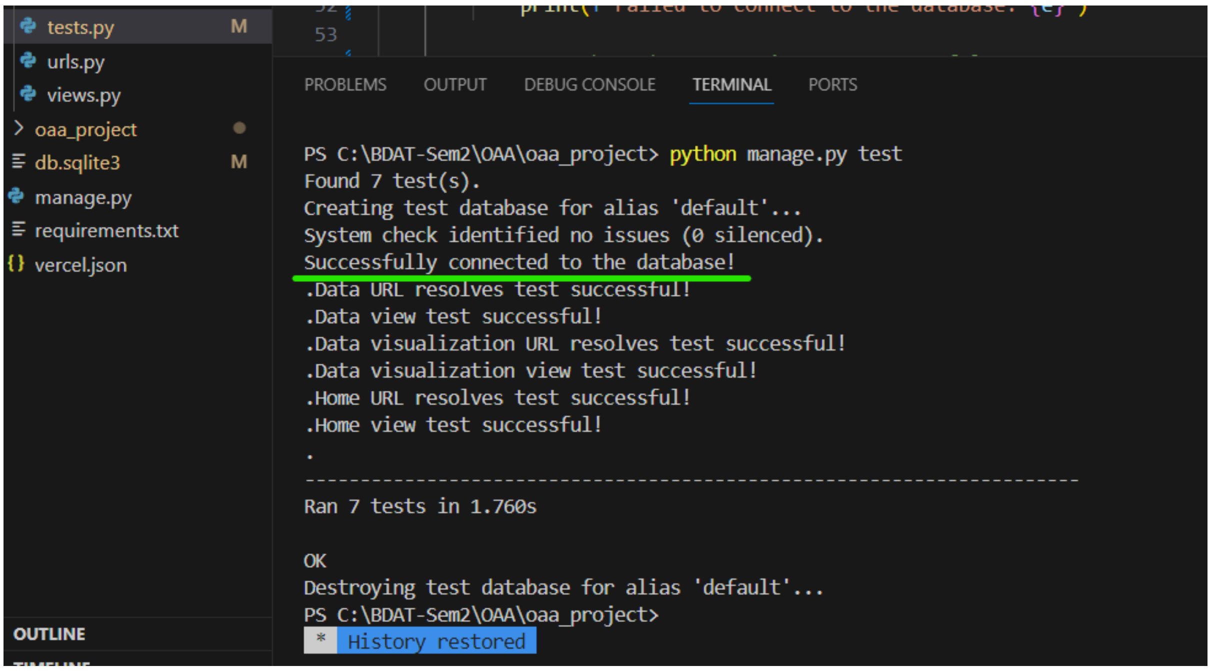
```
PS C:\BDAT-Sem2\OAA\oaa_project> python manage.py test
Found 7 test(s).
Creating test database for alias 'default'...
System check identified no issues (0 silenced).
Successfully connected to the database!
.Data URL resolves test successful!
.Data view test successful!
.Data visualization URL resolves test successful!
.Data visualization view test successful!
.Home URL resolves test successful!
.Home view test successful!
.
----------------------------------------------------------------------
Ran 7 tests in 1.760s

OK
Destroying test database for alias 'default'...
PS C:\BDAT-Sem2\OAA\oaa_project>
```

Database Connection Testing:

- Successfully establishes a connection with the database, ensuring application interaction with the underlying data storage.



```
class TestViews(TestCase):
    def test_data_visualization_url_resolves(self):
        print("Data visualization URL resolves test successful!")

class DatabaseConnectionTestCase(TestCase):
    def test_database_connection(self):
        # Testing connection with the database
        try:
            connection.ensure_connection()
            connected = True
            print("Successfully connected to the database!")
        except Exception as e:
            connected = False
            print(f"Failed to connect to the database: {e}")

        # Assert that the connection was successful
        self.assertTrue(connected, "Failed to connect to the database")
```



```
PS C:\BDAT-Sem2\OAA\oaa_project> python manage.py test
Found 7 test(s).
Creating test database for alias 'default'...
System check identified no issues (0 silenced).
Successfully connected to the database!
.Data URL resolves test successful!
.Data view test successful!
.Data visualization URL resolves test successful!
.Data visualization view test successful!
.Home URL resolves test successful!
.Home view test successful!
.
----------------------------------------------------------------------
Ran 7 tests in 1.760s

OK
Destroying test database for alias 'default'...
PS C:\BDAT-Sem2\OAA\oaa_project>
 *  History restored
```

Black Box Testing:

Navigation Testing:

- Intuitive navigation observed, enabling users to access desired information easily.

Search Functionality Testing:

- Search functionality performs effectively, returning accurate results based on user queries.

Data Visualization Testing:

- Data visualization components provide clear and informative representations of academic data.

User Interaction Testing:

- User interaction features, such as filtering and sorting options, are responsive and user-friendly.
- Refine search algorithms to enhance result relevance and accuracy for all types of queries.

Conclusion:

Unit testing and black box testing of the Ontario Academic Atlas project have provided valuable insights into the system's functionality, reliability, and usability. While the project performed well overall, the identified issues and recommendations will be addressed to further enhance the user experience and ensure project success. Moving forward, continued testing and refinement will be integral to achieving the project's objectives and delivering a high-quality educational resource for the Ontario academic community.