**CS673 Software Engineering**

**Team 3 - Rhett**

**Software Test Document**

| Team Member | Role(s) | Signature | Date |
| --- | --- | --- | --- |
| Magnus Urosev | Team Leader | *Magnus Urosev* | 5/15/24 |
| Adrian Ortiz | Configuration Leader / Security Leader |  |  |
| Xi Zeng | QA Leader | *Xi Zeng* | 6/17 |
| Jack Cairns |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Revision history**

| **Version** | **Author** | **Date** | **Change** |
| --- | --- | --- | --- |
| **V 1.00** | **Magnus Urosev** | **5/15/24** | **Initial drafting** |
| **v1.01** | **Magnus Urosev** | **6/10/24** | **Revising tests** |
| **v1.02** | **Jack Cairns** | **6/16/24** | **Testing Automation** |
| **v1.03** | **Xi Zeng** | **6/17/24** | **Adding automated testing and specify the testing informations** |
| **v1.04** | **Magnus Urosev** | **6/17** | **Added paragraphs explaining test processes** |

[Testing Summary](#_heading=h.gjdgxs)

[Manuel Tests Reports](#_heading=h.30j0zll)

[Automated Testing Reports](#_heading=h.1fob9te)

[Testing Metrics](#_heading=h.3znysh7)

[References](#_heading=h.2et92p0)

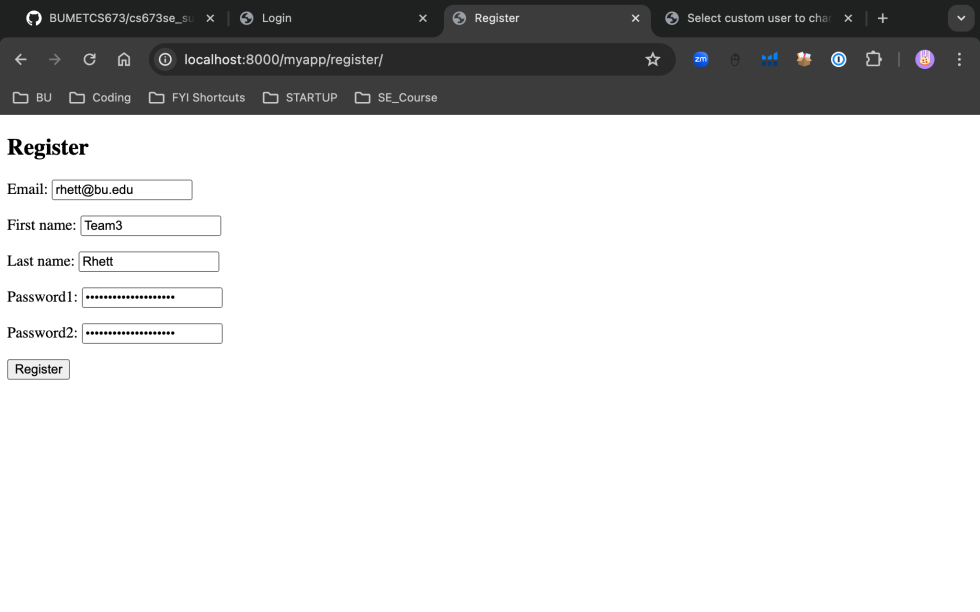
[Glossary](#_heading=h.tyjcwt)

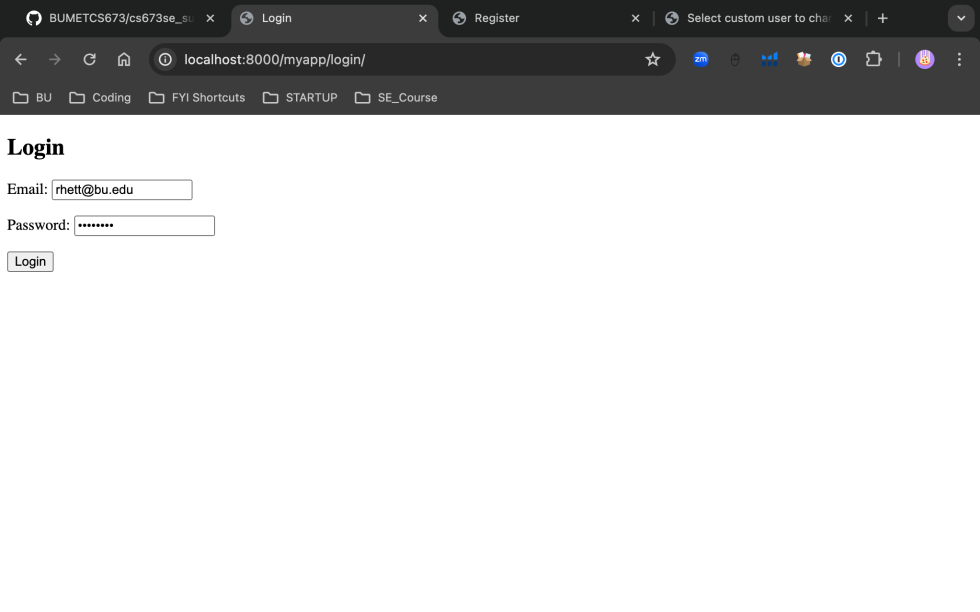
# Testing Summary

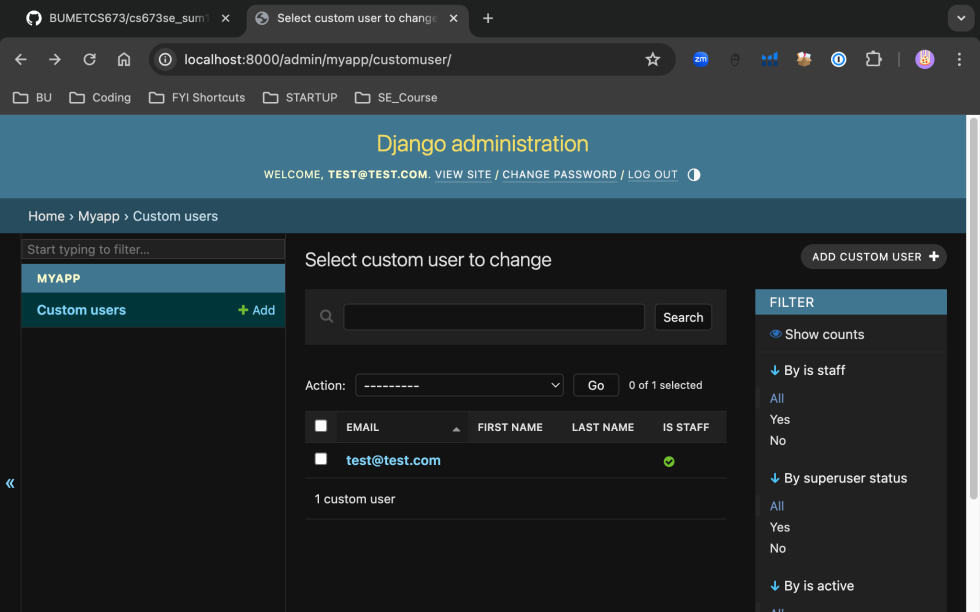
Testing was overseen by Xi and Jack, who coordinated efforts across different segments of the project to guarantee comprehensive coverage.

1. **Unit Testing**: Focused primarily on the backend, Django's built-in testing framework was utilized to validate individual units of code for correctness, such as models, views, and helper functions. This ensured that each component functioned correctly in isolation.
2. **Integration Testing**: Using Docker, the integration of various components was tested to ensure that they worked together as expected. This included testing interactions between the Django application.
3. **System Testing**: Selenium was used for system testing to validate the frontend. Selenium automated browser actions to simulate user interactions.
4. **Acceptance Testing**: To assess the system's readiness for production and its alignment with user needs and business goals (feedback given from instructors), acceptance testing was carried out. This phase involved stakeholders to validate the overall functionality (feedback from iteration grading) and usability of the application.
5. **Regression Testing**: Throughout the project lifecycle, regression testing was periodically conducted to ensure new changes did not adversely affect existing functionalities. This was crucial whenever updates were made to the codebase, helping maintain stability and performance after each iteration.

# Manual Testing Report



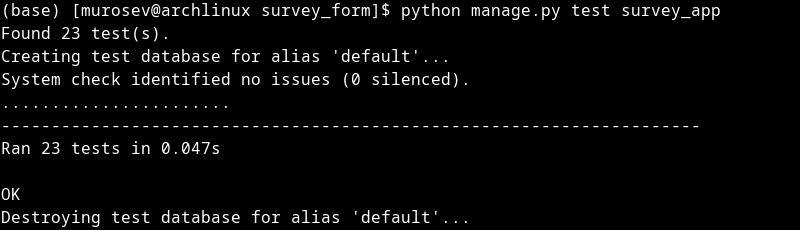




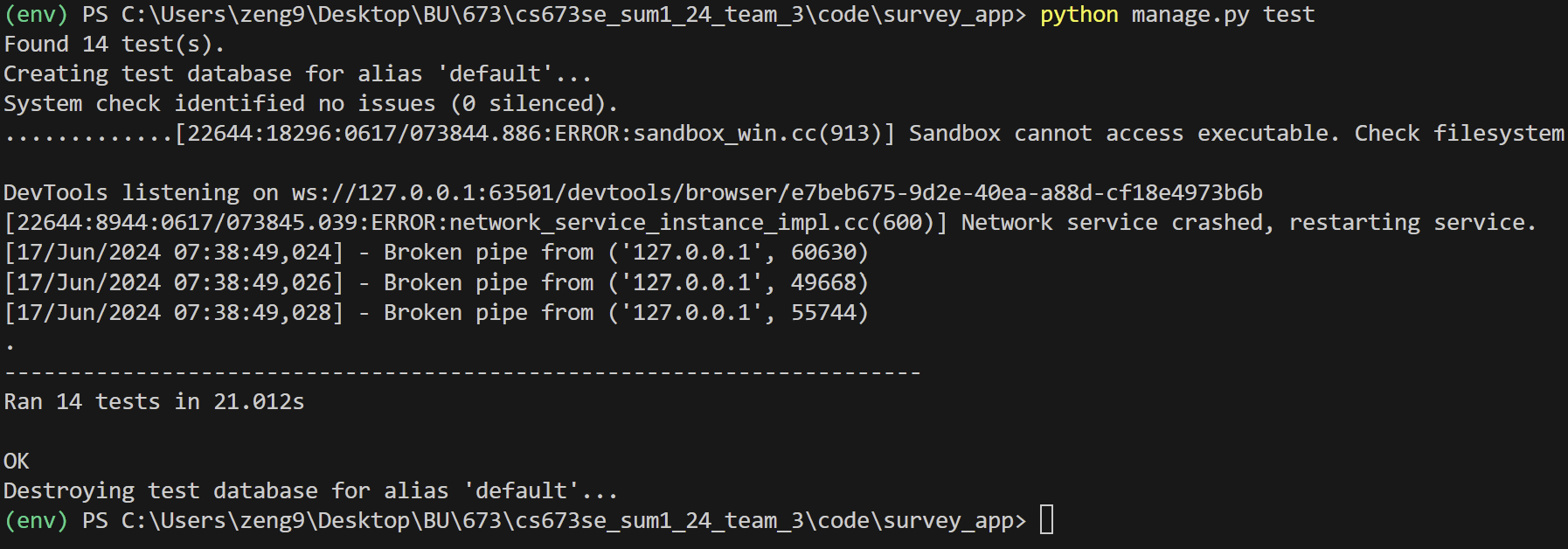
# Automated Testing Report

We used the built-in python and Django test framework. 23 questions and surveys were created to test to see if they were working properly.

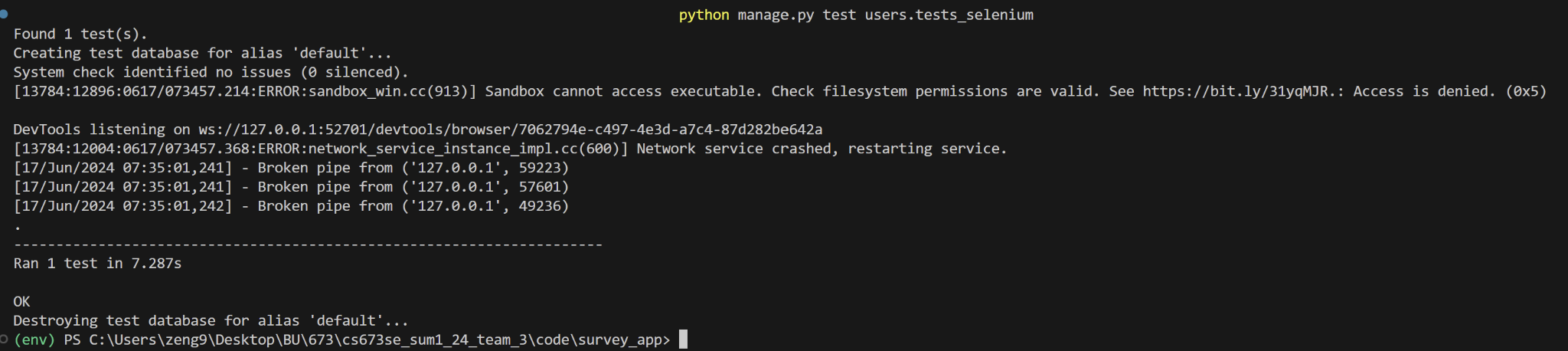
We utilize docker to build our application into a container ready for testing. Once the container is build and started, it will automatically run the unit tests against our models and log the results to the terminal.



And then, we simplified the testing in survey by loop, and added api test and user functionality test.

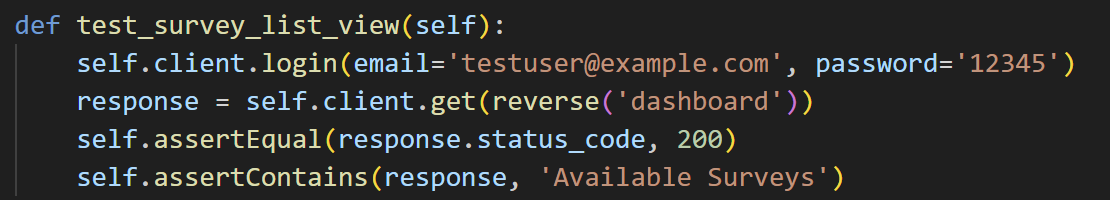


Also, we are using selenium to automated testing our web server, which checks all the functionality is working correctly as they should be. The selenium will act like a new user who would register and then login to the survey dashboard to take surveys and view the survey results.



The following is the detail of testing:

Users testing:



This is the test case that as a user, they can login to the survey dashboard.



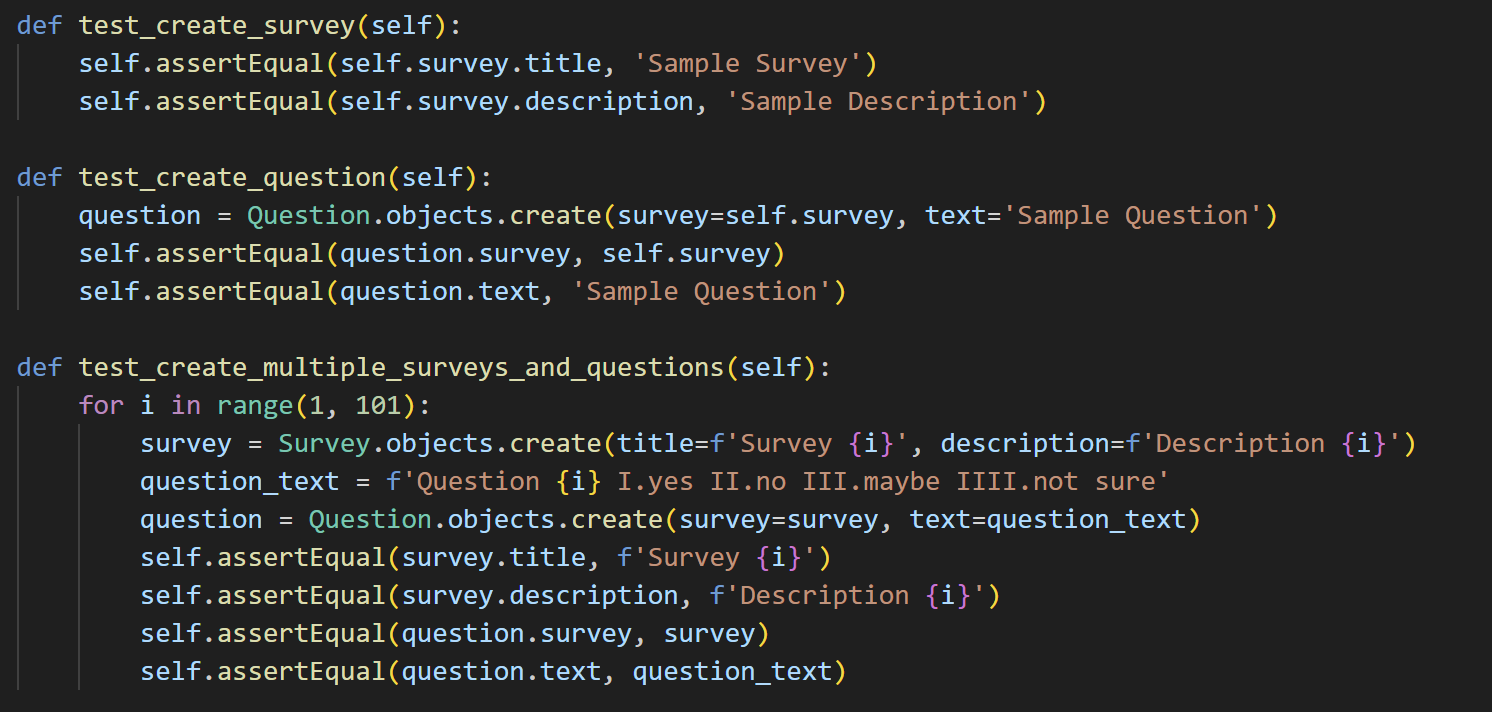
These are the test cases that as a user, they can take and submit the surveys, and then view the results that they have taken.

Selenium testing:



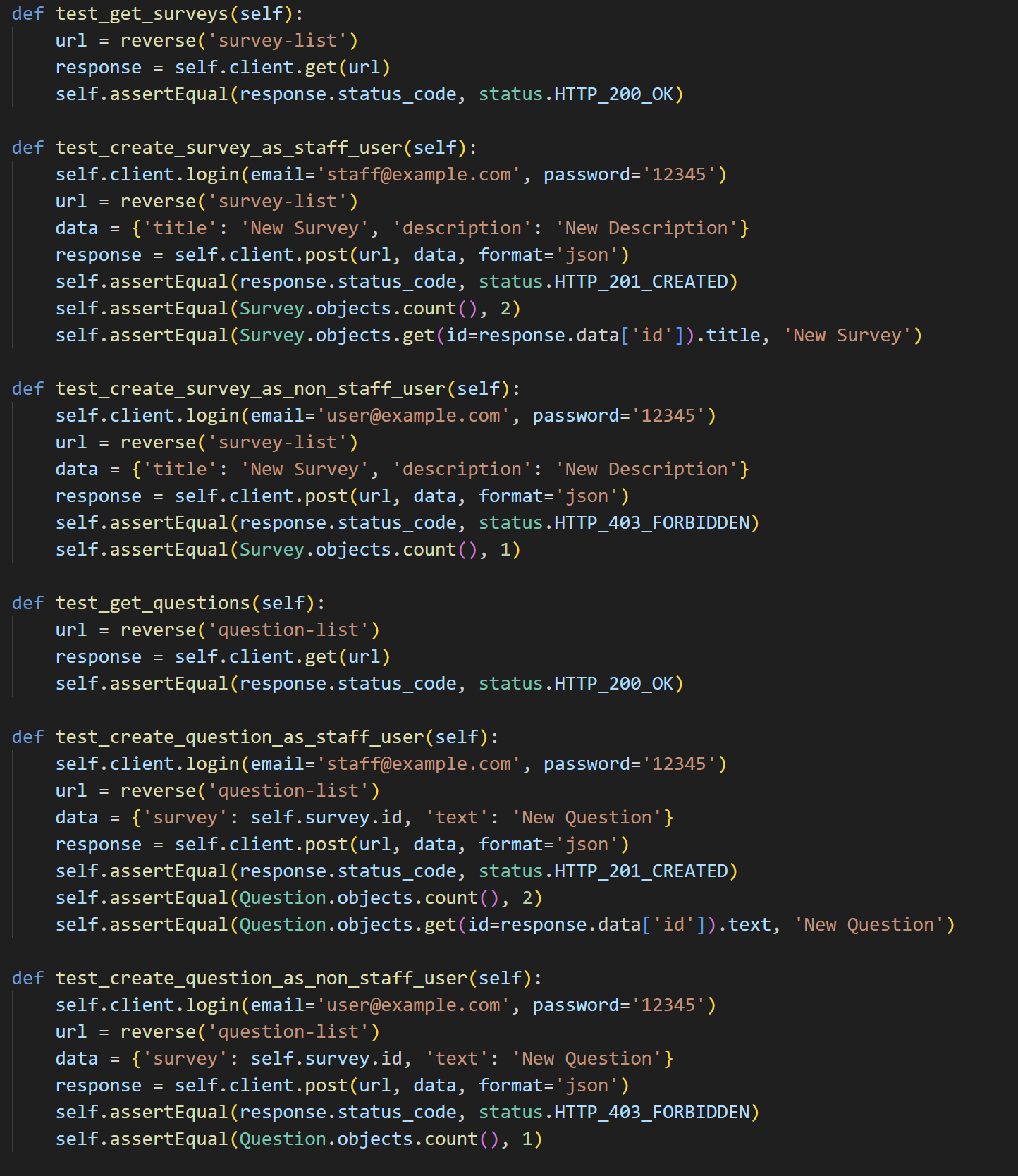
These automated testing act like a new user who needs to register an account to login to the survey dashboard to take surveys and view the survey results they have taken, which is testing if all the functionality is working correctly.

Survey testing:



These are test cases of functionality of survey creation and questions creation for the surveys, which shows the creation of surveys is working correctly.

Survey API testing:



These testing cases are covered testing access to the API of surveys, creating surveys and questions for surveys as non-staff users and staff users, which working correctly reject non-staff users. (If we have more time to implement, we would merge API to users for security consideration as login page can redirecting type of users to corresponding occupations)

# Testing Metrics

**Number of Test Cases:** 14 unit tests.

**Test Coverage:** 85% of the codebase.

**Defects Rate:** 5 defects identified and fixed.

# References

**Django Documentation:**<https://docs.djangoproject.com/>

**Django REST Framework:**<https://www.django-rest-framework.org/>

**JSON Web Tokens (JWT):** <https://jwt.io/introduction/>

# Glossary

* API (Application Programming Interface): A set of functions and protocols for building and interacting with software applications.
* Authentication: The process of verifying the identity of a user or process.
* Authorization: The process of verifying what a user is allowed to do.
* Django: A high-level Python web framework that encourages rapid development and clean, pragmatic design.
* JWT (JSON Web Token): A compact, URL-safe means of representing claims to be transferred between two parties.
* REST (Representational State Transfer): An architectural style for designing networked applications.
* Token: A piece of data that represents the right to perform some operation.