School of Computational Sciences and Artificial Intelligence
CSAI 203: Introduction to Software Engineering
Final Documentation

Final Documentation for Shenron

Prepared by

Amr Khalid 202201502

Abdalrahman Ashraf 202202066

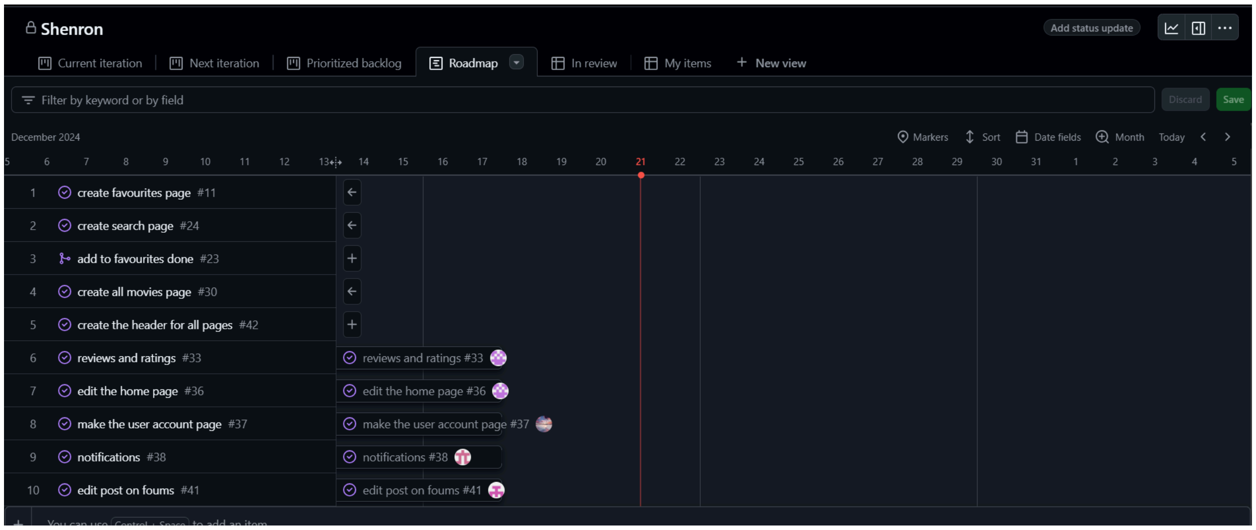
Omar Yasser 202201589

Yusuf Tamer 202201929

School of Computational Sciences and Artificial Intelligence
CSAI 203: Introduction to Software Engineering
Final Documentation

Development Process

During the first stages of this project, many possible features and functionalities were proposed to be included in the project scope. Although a clear vision and purpose were present, the project scope was susceptible to change. For these reasons, an agile development process was adopted throughout the development cycle of the *shenron* project. The agile development process of choice was Scrum. Scrum is an agile method that provides a framework for agile project organization and planning. It does not mandate any specific technical practices. The development process followed the scrum framework of dividing the project into smaller tasks. Tasks were prioritized. Then, the list of prioritized tasks were used to create the product backlog. Each set of related tasks were grouped together to be delivered in an increment. The team worked together on each increment. GitHub was used to share progress and update the product and increment backlogs and resolve issues. Regular meetings were held during each increment. Finally, after each increment, a meeting was held to discuss how the process of delivering an increment can be improved. See Image. 1 for a snapshot of the GitHub product backlog management.



1	create favourites page #11
2	create search page #24
3	add to favourites done #23
4	create all movies page #30
5	create the header for all pages #42
6	reviews and ratings #33
7	edit the home page #36
8	make the user account page #37
9	notifications #38
10	edit post on foums #41

Image 1: *product backlog management*

Object Oriented Design

To ensure the system promotes modularity, scalability, and reliability, an Object Oriented Design (OOD) approach was adopted in designing the system. OOD was implemented by using basic OOP concepts, such as inheritance, polymorphism, abstraction, and encapsulation. Moreover, more advanced OOD principles were applied. The advanced principles included applying design patterns and SOLID principles. Design patterns are programming methodologies that are proven to be useful in designing software systems to avoid and/or solve common problems encountered during software development. SOLID principles are five essential principles that enhance software design process by making code more maintainable, scalable, and adaptable.

A) SOLID principles

The two main SOLID principles used during development are Single Responsibility Principle (SRP) and Open/Closed Principle (O/CP). SRP ensures that each class should have only one reason to change. In return, this ensures that each class has only one purpose. This SOLID principle was applied using the fact that in django a class reflects the representation of an entity in the database. So, a class would only need to be changed if and only if a functionality or attribute to that class would have to be changed. Second, O/CP states that your class should be open for extension but closed for modification. O/CP is applied using the MVT hierarchy of django-based web applications. In MVT, the views, which are functionalities related to some model class, are separated from the definition of the class itself. Hence, more functionalities can be added without having to change the class definition of main functionalities.

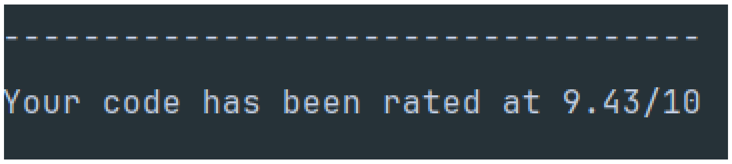
B) Design Patterns

All three types of design patterns were used to ensure the software system follows an OOD. For the creational design pattern, singleton design pattern was used as each model class had only one instance to which all other software components had access to. The singleton design pattern was ideal in our case because each class model will be used to control the corresponding entity in the database. For the structural design pattern, decorator design pattern was used. Decorator design pattern is used to add basic regular functionalities, login requirement functionality, for example, without having to write the logic for it in each functionality that will use it. Resulting in loose decoupling of functionalities and easier testing and modification. Finally, for the behavioural design pattern, command and template design pattern were used. Command design pattern is used as django encapsulates web requests in a request object. The request object contains all the information needed for processing that request. Moreover, the request can be passed to other objects and/or stored for later processing. More information about HttpRequests in Django can be found here. Template design pattern is inherently a design pattern in django framework because all model classes are subclasses of one class, which is the models class found in the django.db module.

Code Quality Analysis

During the code writing process, the team tried to follow the guidelines of the used frameworks as much as possible. All class names were upper case. All function names were written in snake case naming convention. All names of classes, functions, and variables were chosen carefully so that they are meaningful and representative of the process they are included in. Reverse url paths logic was applied according to the guidelines provided by Django documentation. As well, Django documentation guidelines were used for writing django templating code. Finally, The code structure followed the apps structure recommended by Django documentations.

The following figure represents the result of running *pylint* command on the best structured app in the project:



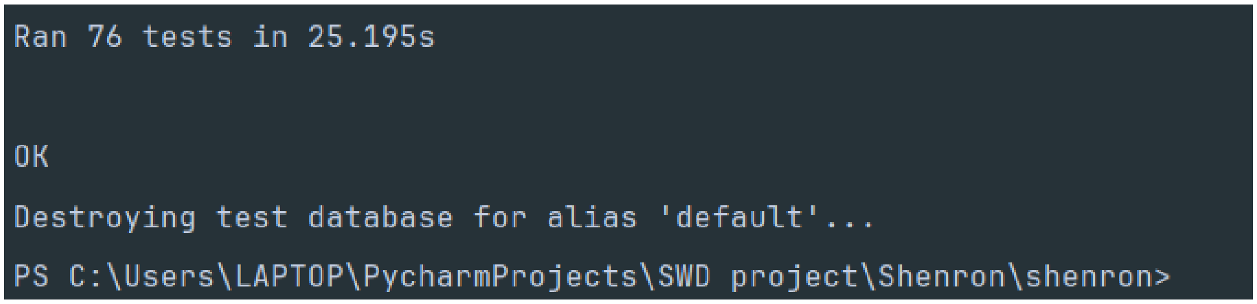
```
-----------------------------------

Your code has been rated at 9.43/10
```

Testing

To rigorously test the code and evaluate its performance, both unit testing and automated testing were used.

A) Unit Testing

In unit testing, testing consisted of testing each app's models and views. All testing related to an app is included in the tests.py file inside of that app. The testing of the basic functionalities of each model were conducted using a class while another class was implemented for testing views and verifying that the system returns the right response. Both classes are subclasses of the TestCase class found in the django.test module. A total of 76 tests were conducted. All of them were successful. Some test cases contained multiple assertions to account for edge cases. Test cases including functionalities that depend on external APIs were handled by replacing the API's response with an appropriate value. See Fig. 1 for the test results.

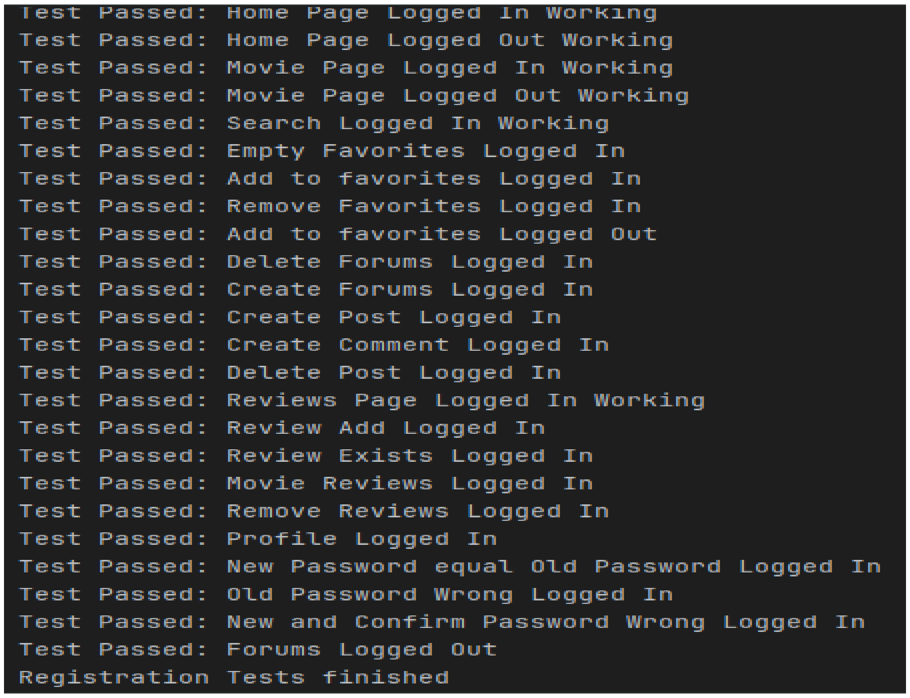


```
Ran 76 tests in 25.195s


OK
Destroying test database for alias 'default'...
PS C:\Users\LAPTOP\PycharmProjects\SWD project\Shenron\shenron>
```

Fig. 1: *Unit Testing Results*

B) Automated Testing

In automated testing, we went through each page and checked all its functionalities and accessibility using the Selenium Python Library. Multiple Test cases were also done in the case of user misinput. See Fig. 2 for the test results.



```
Test Passed: Home Page Logged In Working
Test Passed: Home Page Logged Out Working
Test Passed: Movie Page Logged In Working
Test Passed: Movie Page Logged Out Working
Test Passed: Search Logged In Working
Test Passed: Empty Favorites Logged In
Test Passed: Add to favorites Logged In
Test Passed: Remove Favorites Logged In
Test Passed: Add to favorites Logged Out
Test Passed: Delete Forums Logged In
Test Passed: Create Forums Logged In
Test Passed: Create Post Logged In
Test Passed: Create Comment Logged In
Test Passed: Delete Post Logged In
Test Passed: Reviews Page Logged In Working
Test Passed: Review Add Logged In
Test Passed: Review Exists Logged In
Test Passed: Movie Reviews Logged In
Test Passed: Remove Reviews Logged In
Test Passed: Profile Logged In
Test Passed: New Password equal Old Password Logged In
Test Passed: Old Password Wrong Logged In
Test Passed: New and Confirm Password Wrong Logged In
Test Passed: Forums Logged Out
Registration Tests finished
```

Fig. 2: *Automated Testing Results*

CI/CD Workflow and Deployment Model Documentation

The CI/CD automates the process of testing, building, and deploying the website to an Azure web app using Docker. The process is done by pushing any code to the main branch, CI/CD starts working and doing the steps like building the image, container, doing the test, installing the dependencies, and also connecting to Azure registry container which is connected to the Azure web app so when anything changes in the main branch the web app is being deployed again to be sure any change will take an effect.

Deployment model using Docker image

The application has been collected into a docker container whose image includes the application code and the dependencies. The docker file defines the image, copies all the files inside the shenron project, installs dependencies, and sets runtime commands as well . To connect the image to the cloud we used Azure container registry which we push the image to after being done on the docker. then we configure the web app with the image that has been pushed

Reflections, Challenges, and Future Work

While working on developing this software system, we discovered new ways of thinking about processes. We acquired the ability to think of systems and entities on an abstract level, utilize this abstract view to design our system, and then be able to use that design in implementing the system. Interacting with different software components and being able to use various software packages was a big part of building this software system. For instance, understanding what an API is and how to use it. Choosing the right tools is helpful, as well. Applying design patterns and SOLID principles allowed for a better understanding of OOD paradigm. Using tools like Git and GitHub was helpful in easily sharing and accessing progress allowing for a seamless team collaboration. Being able to experience how real world applications are made and going through the numerous stages of application development like: requirement specification, design, implementation, and deployment is quite eye opening.

Even though developing a web application provided us with experience, we faced many challenges during the development. The main challenges were related to the software engineering part more than that of the development part. Software engineering tasks are hard to understand and implement, especially when it is the first time, because they require knowledge and experience. To be of knowledge and experience requires investing a lot of time. Hence, resulting in another challenge, which is the time limitations. Having so little time associated with the amount of new things that we had to learn and deal with from the start greatly increased the difficulty of developing this application and completing the project as intended.

In the future we hope to not only exploit the methods and processes we learned to be able to effectively design, implement, and deliver a software product, but also learn from the mistakes we made and the infinite number of bugs and errors that we had to resolve and deal with.