**Installing Django**

Install python and pip

Then install Django using : *pip install Django*

Other useful commands:

*\* python .\manage.py makemigrations*

*\* python .\manage.py migrate*

*\* python manage.py runserver*

*\* python .\manage.py createsuperuser*

**1. Satisfying API Specification (Automated Test)**

How it was accomplished:

* I ensured that the API meets the specification by writing automated tests for all API endpoints based on the provided OpenAPI specification.
* The API tests were created using Django’s built-in testing framework, which allowed me to simulate requests to the server and verify that the responses matched the expected outputs (e.g., creating flashcards, submitting ratings, retrieving data).
* I did not have access to the automated tests (according to the instructions), so I based my implementation on the API requirements and wrote tests that would cover all possible scenarios, such as successful data creation and invalid data submission.
* When I pushed my code, automated tests were run via GitHub Actions to ensure compliance with the API specification.

**2. Usage of a Test Framework to Test Code (Coverage & Quality)**

How it was accomplished:

* I utilized Django’s TestCase class to write tests for all major features of the application, including API endpoints and form submissions.
* The coverage was achieved by testing not only the success cases (e.g., when a flashcard set is successfully created) but also the edge cases (e.g., handling invalid inputs).
* Django’s testing framework allowed me to cover end-to-end testing for all features, ensuring that the application works as intended from start to finish.

**Test Quality (10 marks)**

How it was accomplished:

* Unit Testing: I wrote unit tests for critical parts of the code, such as:
  + Validating the rating calculation for flashcard sets.
  + Ensuring that user authentication works as expected (e.g., restricted access for unauthorized users).
  + Checking form validation to make sure invalid data isn't accepted.
* Fuzz Testing: I simulated unpredictable inputs (e.g., random strings or malformed data) to ensure that the application handles unexpected data gracefully.
* Accessibility Testing: Ensured the application is usable with screen readers and that no critical elements are missing or improperly labeled.
* Security Testing: Included tests for common vulnerabilities, like Cross-Site Request Forgery (CSRF), ensuring that the API does not accept unauthorized actions or malicious data.

**3. Automatic Testing**

How it was accomplished:

* I integrated automated testing into the development process using GitHub Actions, which allowed the tests to be run automatically on every push to the repository.
* This ensured that no new changes would break the application and that the application would always meet the required specifications.
* The test results from GitHub Actions were continuously monitored to maintain code quality.

**4. Visual Appearance and User Experience**

How it was accomplished:

* I focused on creating a clean, responsive design using flexbox and grid layouts for various components like the flashcard sets and reviews.
* The design is mobile-friendly and works across different screen sizes.
* Buttons are styled with hover effects, and all forms are user-friendly with clear labels and placeholders.
* I ensured that the back buttons and navigation elements are placed in easy-to-reach/see locations for an intuitive user experience.
* The overall color scheme and typography were chosen to be clean and legible, ensuring users can easily interact with the application.

**5. Changes to OpenAPI Specification**

How it was accomplished:

* I updated the OpenAPI specification to reflect the actual behavior of the API.
* I added additional fields in the API responses, such as average rating for flashcard sets, so the data returned by the API was more complete.
* I updated the POST endpoint for reviews to ensure users can only submit one review per flashcard set. If a user tries to submit a second review, the API returns an appropriate error.

**6. Implementation of Future Features**

How it was accomplished:

* Future Enhancements: The following features were considered for future implementation:
  1. User Profiles: Adding functionality for users to create and manage their profiles, where they can view their flashcard sets and reviews.
  2. Search and Filtering: Implementing a search bar and filters to allow users to easily find flashcard sets based on title, ratings, or creation date.
  3. Real-Time Collaboration: Allowing multiple users to edit a flashcard set in real-time, which could make the platform more collaborative.
  4. Leaderboard: Implementing a leaderboard that ranks flashcard sets based on user ratings and engagement.