SeetGeek Design Doc Assignment 2

Authors: Aarti Singh, Amaar Jivanji, Buchi Maduekwe, Lia Mason

Overall Structure:

Files: Frontend.py, backend.py, _main_.py

The structure contains four layers. There are the frontend, backend, the server and the database. The user interacts with the frontend, triggering actions that will then trigger backend classes and functions to build and run which requires information from the database.

Class and Method Diagrams:

Name of class/function	Description
Classes	
User (db.Model)	A user model that defines the SQL table. Stores ID, email, password, and name
Functions	
profile (user)	Retrieves the list of tickets from backend and stores it in variable 'tickets' Handles the user home page request ('/'). Renders template index.html
register_get()	Renders the register.html template
register_post()	Ensures entries in registration form meet all constraints for email, password and username
login_get()	Renders the login.html template
login_post()	Stores email, password, and user
logout()	Ends session and redirects user to login page.
sell_get()	Redirects to user home page after user submits a "sell ticket" form
sell_post()	Stores data (name, quantity, price, expiration) after user submits a "sell ticket" form

authenticate (inner function)	Check the current session to see if the user has logged in. If login, it will call the inner_function with the logged in user object.
register_user()	Performs password hashing and registers new user to the database
get_all_tickets()	Returns list of tickets
login_user()	Logs user in, given email and password
get_user()	Returns a user that has the matched email address
sell_form()	Update the database with the new ticket
buy_form	Remove the ticket from the tickets to be sold database and add the ticket to the users bought tickets.
buy_submit	Submits data from the buy form.

Test Plan:

Levels (Frontend, Backend, Integration):

Test folder for each component: Frontend, Backend, Integration

Requirements partitioned into separate testable features and different components and features are tested in different python files as shown below.

Frontend:

- Test_fnconnection.py
- Test_fnregistration.py
- Test_fnlogin.py
- Test_fnlogout.py
- Test_fnbuy.py
- Test_fnsell.py
- Test_fnuserprofilepage.py
- Test_fnaccountbalance.py

Backend:

- Test_bnlogin.py
- Test_bnlogout.py
- Test_bbuy.py

- Test_bnsell.py

Order:

1. Frontend (patch backend to return specific instances without running backend program)
2. Backend
3. Integration: Individual software modules are combined and tested as a group

Techniques and tools:

- Mocking -patching backend to return specific values
- Input partitioning
- GitHub Actions
- Selenium
- Unit, partial and complete regression testing when new features added or existing features modified
- Boundary testing for input values
- Assert statements
- Test all types of user inputs
    - Utilising TestingSummary.md to ensure all possible inputs and outputs are tested

Environments:

- Text Editor
- Github
- Visual Studio Code
- Google Chrome
- Mac OS and Windows

Responsibility:

- R1: Buchi
- R2: Amaar
- R3: Lia & Aarti
- R7: Buchi
- R8: Aarti

Budget Management:

- Getting the owner to manually check and monitor the CI action minutes
- In order to keep track of CI action minutes, the group will log the remaining minutes after each group meeting in a table, that will inform members when minutes are running low
- Skip running CI when just editing markdown files
- This will ensure the group avoids all unnecessary costs