Gal Peled 201581712

Web Application Development Coursework 2 Document

Purpose and Features

My website is called “Pokémon Market” and it is an imaginational e-commerce market of Pokémon. Users can register to the site as Pokemon trainers and login.
When a new trainer sign up, he starts with 1,000 PokeBalls, which is actually his budget for “catching” Pokemon. Each Pokemon on the market has a price in PokeBalls and the trainer can catch Pokemon from the available on the market if he has enough PokeBalls for it, and also sell his own Pokemon to earn PokeBalls.
Trainers can also like and dislike Pokemon by clicking their thumbs up and thumbs down buttons.
Trainers can see on the left side of the market page the current available Pokemon with their name, icon, price, type, number of owners, as well as likes, dislikes by other trainers. They can enter the Pokedex to see some additional information about the Pokemon with a bigger picture of him.
On the right side of the market page, trainers can see their owned Pokemon which they already caught from the market with their name and price. They can sell them for their price and earn PokeBalls in return. The “Pokedex”, “Catch” and “Sell this Pokemon” will open a little window with the appropriate details and interactions, implemented with Bootstrap’s modal component.
The market page is available only when a trainer is signed in, if that’s not the case, he would get a message saying that he needs to sign in to view this page and redirected to the login page.
The home page appears at start is welcoming the trainer. A click on “Catch ‘Em All” will redirect him to the market page if logged in, or to the the login page otherwise.
If signed in, be can also enter the market from a link on the navigation bar, there he can also see a note with his name, his current amount of PokeBalls, and other links to change his password or logout.
For changing his password, he would have to type his old password, the new one and also confirm the new password. If the latter is done successfully, he will get a message that the password has changed and will be redirected to the login page to login with his new password. If there are problems with changing the password, corresponding messages will appear on the website.
When a trainer clicks on “Logout”, he gets a message that he is logged out and he’s redirected to the home page when now he is not logged in anymore.
When a user is not logged in, he will see in the navigation bar links for “Login” and “Sign Up” instead of “Change Password” and “Logout”.
Login will ask the trainer for a Trainer Name and password, then he will need to press “Continue Your Journey” to login and if entered valid credentials, that is if there is such Trainer in the database and the password is matched, he will get a successful message and will be redirected to the market.

If the name and password are incorrect, a corresponding message will appear. The login page has also a link to Register under the written “Not a trainer yet?” which redirects the user to the sign-up form.
In sign up page, the user will requested to write a Trainer Name, Email Address, Password and Password confirmation. There is also a link to login if he is already a trainer.
If the user entered invalid inputs (for example, password is too short, confirmation password doesn’t match, invalid email address or too short trainer name) he gets corresponding messages when he clicks “Go Catch ‘Em All” in the bottom of the sign up page.
If entered valid inputs and clicked the button, a new trainer account is created and the user is redirected to the market page to start catching and selling Pokemon.

Link to the Deployed Website

My web application has been deployed with Python Anywhere for hosting and domain name by pushing the code to Gitlab and then moving it to Python Anywhere.

https://gpeled.pythonanywhere.com/

There is no need for a username or password to access.

Analaysis

Web Forms in my web application are implemented in Python using Flask WTForms with fields and validators on the server side in the forms.py file and are manipulated in views.py for their suitable purpose with Flask Form and request objects. On the client side, forms are implemented with HTML, CSS and Bootstrap in the files login.html, sign_up.html, change_password.html as well as in pokemon_modals.html and owned_pokemon_modals.html which are both included in market.html. The existing forms in the website are Sign Up form for register a new trainer account, Login form to login an existing trainer account, Catch Pokemon form for a trainer to catch an available Pokemon on the market, Sell Pokemon Form to sell Pokemon owned by the current trainer, Change Password form for the user to change his password. Catch Pokemon and sell Pokemon forms are implemented by sending hidden input tags on the server side with the Pokemon name value to catch/sell (in pokemon_modals.html and owned_pokemon_modals.html). All forms are applied with validation on both client side and server side and informative feedback is provided to the user who send incorrect input as well of correct input.

Gal Peled 201581712

Database is implemented with Flask and SQL Alchemy in models.py with python object oriented programming and are manipulated in view.py and in the html files with Jinja for different purposes. They are migrated using Flask Migrate which uses Alembic for reflect changes in the models.
The models in my web application are Trainer and Pokemon. **Trainer** has id column as primary key, trainername string column, email string column, password_hash string column (not the real password, for security reasons), pokeballs integer column which holds his amount of pokeballs (new trainer starts with 1000 by default).
Trainer has __repr__ method, pokeballWithComma method for adding a comma to the pokeballs number if it is big enough. password setter method for setting his password_hash column to hash code of the original password for security with Flask bcrypt. checkPassword method for checking if the password received as parameter is matched to the trainer by unhashing the real hash_password and compare.
Pokemon model has id column as primary key, name string column, price integer column for representing his price in PokeBalls, type string column, description column for the Pokedex information, likes and dislikes integer columns for representing the reviews by different trainers on the Pokemon. Pokemon has __repr__ method, catch method for handling Pokémon catching from the market by trainers, sell method for handling Pokemon selling by trainers who own them, and numOfTrainers for outputting the number of trainers own this Pokemon instance.
They have many to many relationship with each other implemented with the Trainer's pokemon property of db.relationsship and the trainersPokemon db.table instance which connects Pokemon and Trainers by their id. Trainer can access his owned Pokemon with the pokemon property while Pokemon can access their trainers with owned_trainers backref.

Authentication, Sessions and Cookies are implemented using Flask Login. @login_required decorator is used to make sure user is logged in for the needed pages. current_user object is used to identify the user. login_user and logout_user methods are used to login and logout the user, respectively with login_manager and UserMixin for applying sessions and cookies and manipulating the database in accordance. flask_bcrypt is used for the database security by hashing passwords. A user privacy warning is displayed when entering the website for the user to accept the cookies or deny them. Users are staying logged in until they log out and a new user can register with the sign up page. Login and register are handled by forms, validations and database. User can also change his password in the change password page when logged in.

Styling: I used a consistent styling on all pages for a smooth user experience, the styling is mostly red and white as suitable for the Pokemon world the website is based on (PokeBall is red and white). The website is responsive for any device. Pokeball icons and logo are included as well as different icons and images for the Pokemon. Feedbacks to the user are colored after their purpose. Like button is green while dislike button is red for distinguishing them and mark them "good" or "bad" respectively.

Gal Peled 201581712

Unit tests are implemented with python unit test framework in various different methods included in the files test_market.py, test_trainer.py, test_view.py
test_trainer.py includes test methods to for testing login, sign-up, logout, password change.
test_view.py includes test methods for testing routes.
test_market.py includes test methods for testing Pokemon, market, catching and selling manipulations.

Logging is implemented with python logging module using FileHandler objects and different loggers with different severity levels and formats. Debug and Info logs are logged in the console with Flask default logger while more serious logs including Warning, Error and Critical are logged in log.log file with custom made logger object, both containing useful messages for easy tracking.

AJAX and **jQuery** are used with **JSON** for review system where Trainers can review different Pokemon on the market by like or dislike them and the number of likes and dislikes of each Pokemon is visible. This is implemented with the review method in view.py for manipulating the database on the server side. While on the client side it's implemented in review.js jQuery file for manipulating the page accordingly without refreshing it each time a new like/dislike record is received.

Gal Peled 201581712

Security Issues and Handling

Cross-site scripting could be used by attacker users in the forms of my website when they can send inputs containing unsensitized JavaScript for malicious purposes.
I am handling it in my web app by converting all HTML entities into escaped sequences such that an unsensitized input could not return to the user. This is done by Flask templating engine.

Exploited session management could be used by attackers to hijack authenticated sessions. I use Flask-Login to avoid it by managing let it manage my web app session management and authentication system.

Sensitive data exposure and **Insecure Direct object reference** could be used in my website to retrieve passwords of users by hackers or stealing Pokemon without catching them with PokeBalls. I handle it with users password hashing used with Flask-Bcrypt and proper authorization done with Flask-Login.

Security configurations issues can cause intrusions to my website. I mitigate it by using Python Anywhere which has effective and up-to-date security configurations.

References

https://pokemondb.net/

https://docs.python.org/3/library/logging.html

https://flask.palletsprojects.com/en/2.0.x/logging/

https://docs.python.org/3/library/unittest.html

https://flask.palletsprojects.com/en/2.0.x/testing/

https://flask.palletsprojects.com/en/2.0.x/patterns/jquery/

https://api.jquery.com/jquery.ajax/

https://flask-login.readthedocs.io/en/latest/

https://flask.palletsprojects.com/en/2.0.x/reqcontext/

https://stackoverflow.com/questions/7478366/create-dynamic-urls-in-flask-with-url-for

https://docs.sqlalchemy.org/en/14/orm/basic_relationships.html

https://michaelcho.me/article/many-to-many-relationships-in-sqlalchemy-models-flask

https://flask-bcrypt.readthedocs.io/en/latest/

https://getbootstrap.com/docs/4.0/components/modal/

https://getbootstrap.com/docs/4.1/utilities/sizing/

https://help.pythonanywhere.com/pages/Flask/

https://pythonbasics.org/flask-login/

https://flask.palletsprojects.com/en/2.0.x/patterns/wtforms/

https://getbootstrap.com/docs/4.0/components/card/

https://www.osano.com/cookieconsent

https://pythonhosted.org/Flask-Security/