COP290
Semester 2 2022-23
Lab 3

Learning Objectives: Reading code, designing and writing clean code, backend, frontend, CI/CD, SQL, testing.

Problem statement:
This lab will be done in groups of four. The main objective is to design and develop a website.

- You need to design and develop APIs required for a website and set up a database on MySQL. Data for your websites can be obtained online from websites like Kaggle. Then, you will design the frontend for the website and get it running. You will also implement unit and integration tests and run them continuously.

- The next step is to add a login page while the website is up and running while checking that the other components are not getting affected using the tests implemented in the previous part. The login page should take basic information (with a picture optional) from a user and then add the user to the database. It is advised to use an encryption algorithm to store passwords.

- The last step is to add an existing Machine Learning API as a feature of your website. You can choose an API hosted by google listed here. This API can be used to make an existing feature better i.e. add recommendations to a selling website or have personalised feeds on a Twitter-type platform. Or you can plan and add a new feature altogether.

Some website examples are listed below. You can choose either one of these or any other website of your choice. The tasks should include building APIs, a database, and a login page along with multiple web pages.

1. Twitter / Stack Overflow / Reddit
    a. Basic website with functionality to post opinions/questions, reply to other posts (comments/answers) and upvote or downvote the posts and comments/answers.
    b. A post can be a combination of text and pictures.
    c. Feature to follow other users to check out their posts and a feed for each user checking posts from the other users that they follow.
    d. All the searched posts and the comments/answers for each post can be sorted in different orders: the number of likes and time of posting.
    e. The user's profile page should display basic information about any registered user and their posts.

2. Bazar portal
    a. Visitors should be able to see the items listed for sale
    b. Registered users of the portal should have access to its primary features, which are listed below.
        i. Buy and sell things
        ii. The seller should be able to upload at least the product's name, price and 1 photograph.
        iii. User's profile page that displays basic information about any registered user and the status of all the products pertaining to them.
    c. You can build a buying system like Amazon or a bidding system like eBay.

Milestones and their end dates:

Week 1, 1st March

Problem statement and design idea documentation
The document should include the following steps:

- **Overview**: Website name and its basic description
- **Dataset**: Link to the dataset and description of how it is relevant to your website. (For example, if you are building a railway system website, then you would require a dataset with information on the train system of some place with timings, stations and everything)
  Note: You can make custom datasets according to your needs, use APIs that call datasets or merge datasets. It's all very flexible!
- **Content Structure**: A site map or hierarchy of web pages. (what all web-pages will it include and which all are connected)
- **Functionality**: A detailed description of the features of each webpage.

Week 2, 8th March

Click-through prototype on Figma
A front-end prototype will show different pages of your website and will show transitions after clicking different buttons.
This prototype will help you decide on the page layout and different APIs you might need.
Using Figma plugins, this prototype can give you a rough front-end starter code for all your web pages.

Database ER Diagram
Design the database in the form of an ER Diagram.
Have a look at https://github.com/tmibvishal/healTrip for an example ER Diagram.

Week 3, 15th March

APIs Design and Development

Design a structure of APIs in the form of a YAML file and visualise it with Swagger.

You can use Swagger Codegen to generate starter code for a Python server and a Javascript client.

Connecting the backend (swagger-generated code) with your database.

Week 4, 22nd March

Write unit and integration tests for your code and use GitHub to run all the tests whenever you commit new code. Coverage (calculated by Python Library) should be greater than 60%.

Week 5, 1st April

Building the front end for all the basic features, integrating the back end with the front end and getting the website up and running. Also, host the website on a Baadal VM.

Week 6, 8th April

Adding a login page feature and related web pages. The task is to rely on unit and integration tests so that the website doesn't fall apart.

Week 7, 15th April

Adding an existing ML Google API to the webpage without disturbing other components used in integration testing.

Week 8, 23rd April

Final touch-ups and presentation!

Note that these are only recommended time allocations. Depending on the complexity and comfort with the language, different parts may take more or less time. It may be wiser to start on the next week's task as soon as you are done with the current week's tasks.

Submission instructions:

To track the project's progress, we have some checkpoint submissions. No late extensions are provided.

Week 1, 1st March

Initial website design plan with the name of the website, team members, its overview, dataset, content structure and functionality.
Deliverables: Google docs link pasted to the sheet here

Week 3, 15th March

Frontend prototype designed on Figma. ER diagram for the database. API design and documentation on swagger.
Deliverables: Google docs link submitted to Moodle. Figma project link, a video of interacting with the frontend prototype, a pdf for ER diagram, and a YAML file for API design.

Week 6, 8th April

Up and running website
Deliverables: A short video of website features, GitHub link to the code, test coverage report, and a link to the website hosted on Baadal VM. *Append all these to the Google docs link submitted on week 3.*

Week 8, 23rd April

Deliverables: You will be required to give a presentation and a demo of your website followed by Q & A. Also submit a final report with links to all the updated design and documentation (website documentation, Figma link, ER diagram, swagger YAML file). Distribute 40 tokens among your team members specifying the workload split. Also, specify who worked on what. *Final submission to moodle. Keep the week-1 and week-2 and 3 Google docs links.*

Grading:

Completeness: 10
Code cleanliness: 5
API design: 5
Frontend design: 5
Database design: 5
Login and ML API integration: 5
Final presentation and report (counted towards exam total): 15
Total: 50

Every missed checkpoint submission will result in a deduction of 10% from the final score of every team member. For example, if a team does not submit week 1 and week 3 checkpoints on time, 20% of marks will be deducted from each team member's final score.

References:

Web-development in Python

1. REST APIs: https://realpython.com/api-integration-in-python/
2. Flask & Frontend: https://www.digitalocean.com/community/tutorials/how-to-make-a-web-application-using-flask-in-python-3

MySQL:

1. https://dev.mysql.com/doc/mysql-getting-started/en/
2. https://www.guru99.com/er-diagram-tutorial-dbms.html

HTML/CSS/JavaScript:
https://www.w3schools.com/

Swagger for server and client starter code generation (https://editor.swagger.io/):

1. https://www.youtube.com/watch?v=TybZad1zwes

Figma (www.figma.com/):

1. https://designcourse.com/blog/post/figma-tutorial---an-introduction-to-a-very-impressive-ui-design--prototyping-tool
2. https://www.animaapp.com/blog/design-to-code/how-to-export-figma-to-html/

Continuous Integration / Continuous Delivery / Testing:

1. https://www.infoworld.com/article/3271126/what-is-cicd-continuous-integration-and-continuous-delivery-explained.html
2. https://realpython.com/python-continuous-integration/
3. https://docs.github.com/en/actions/automating-builds-and-tests/building-and-testing-python
4. Coverage: https://www.pythontutorial.net/python-unit-testing/python-unittest-coverage/

API testing using Postman:
https://learning.postman.com/docs/getting-started/introduction/

UX design:
https://www.nngroup.com/articles/#videos
http://baddesigns.com/examples.html

Recommended books:
1. For designing user interfaces: The Design of Everyday Things by Don Norman.
2. For object-oriented programming: Head First Design Patterns by Eric Freeman, Elisabeth Robson, Bert Bates, Kathy Sierra