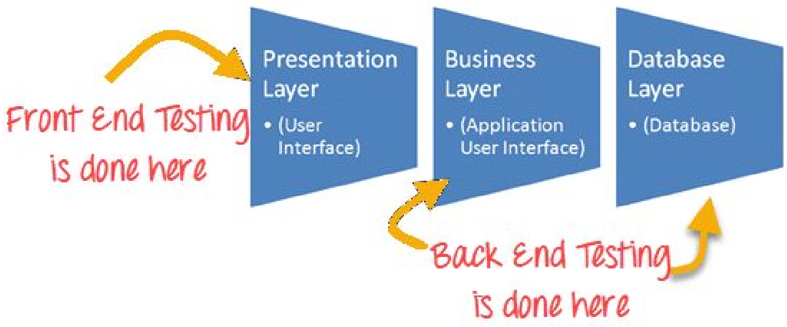
Flask and Web Apps

Draw the nodes of a web application.


Front End Testing is done here
Presentation Layer
• (User Interface)
Business Layer
• (Application User Interface)
Database Layer
• (Database)
Back End Testing is done here

Installing Flask

Why do people use Flask? What does it all you to do?

Flask is microframework for python. The “micro” in microframework means Flask aims to keep the core simple but extensible. You could install flask with the following command:

```
pip install Flask
```

The Backend

What is an API?

Create a folder dedicated to this project. Within it, create a file called `app.py.`
Write the code below into `app.py.`

```
from flask import Flask

app = Flask(__name__)

@app.route("/")
def hello():
   return "Hello World!"


if __name__ == '__main__':
   app.run(debug=True)
```

You may now run the program.
Open your browser and access localhost:5000. And voila now you have your first flask application :)

Behind the Code

```
from flask import Flask
```

This line asks the application to import the Flask module from flask package. Flask is used to create instances of a web application.

```
app = Flask(__name__)
```

This line creates an instance of your web application. `__name__` is a special variable in python, it will equal to "`__main__`" if the module(python file) being executed is the main program.

```
@app.route("/")
```

This line is called a function decorator. Decorators define the routes that will cause our function to execute. For example if we set route to `"/"` like above, the code will be executed if we access `localhost:5000/`. You could set the route to `"/hello"` and our `"hello world"` will be shown if we access `localhost:5000/hello.`

```
def hello():
   return "Hello World!"
```

This line defines the function that will be executed if we access the route.

```
if __name__ == '__main__':
   app.run(debug=True)
```

This line means that your flask app will run if app.py is run. Notice also that we are setting the debug parameter to true. That will print out possible Python errors on the web page helping us trace the errors.

Adding a Frontend

We will make a super simple frontend where the HTML, CSS, and Javascript are all in the same file.
Create a file called index.html and enter the following

```
<!doctype html>
<html>
<head>
  <meta charset="utf-8">

  <title></title>

  <script type="text/javascript">
  </script>

  <style type="text/css">
  </style>

</head>

<body>
</body>

</html>
```

Add a button to page that will execute a javascript function when it is clicked by writing...

```
<button onclick="myfunction()">Click Me!</button>
```

The Messenger
Define a javascript function between the `<script></script>` tags.

```
function myFunction(){

}
```

In the function, place the following code

```
jQuery.ajax({
                           url: 'your_route',
                           type: 'POST',
                           cache: false,
                           //data: JSON.stringify(request),
                           contentType: 'application/json',
                           //processData: false,
                           //success: on_request_success,
                           //error: on_request_error
              });
```

Modify the decorator in the API to execute the function only when a POST request is received

```
@app.route('your_route', methods = ['POST'])
@cross_origin(origin='*',headers=['Content- Type','Authorization'])
```

Assignment:

- Create an API that will turn a light on when a POST request is received.
- Add an additional button to turn on a colored light when it is clicked.