TOPIC
WEB DESIGN PROJECT LEARN IT

TUTOR
Mir Arash Ghias Zadeh

STUDENT
Amir Hosein Shygun

Techs Used
HTML - CSS - JAVASCRIPT - PYTHON

File Structure

FRONTEND

- **HTML**
- **Base.html**
    - A singular main Base.html file
    - Has the Base.css and Base.js links added
    - Takes two variables for the specific page from Flask and adds the two links
    - Base CSS and base JS files will fill the header, body, and footer divs and set them up for the remaining of the pages.

- **CSS / JS**
- A pair of CSS and JAVASCRIPT for each page
- Each page will only fill the in the middle #body div.
- Each div has classes or IDs for better addressing from their CSS counterparts

- **PAGES**
- **Base.js / .css**
    - **createHeaderDiv()** - creates the header for the entire site and also adds the responsive fields for narrower widths of screens
    - **createBodyDiv()** - sets a div with #body id in between that all other js files will add their divs to
    - **createFooterDiv()** - creates the footer for the entire site
    - **addIconLink()** - adds the link to the cdn of remixicon
    - **setTheme(theme)** - adjusts the theme of the page regarding to the previously set theme in API (if none, it defaults to dark)

- **Authorization.js / .css**
    - **createAuthFormDiv()** - Fill the login form
    - **checkLog(status, event)** - Checks the data and proceeds with API calls

- **Contact.js / .css**
    - **createLeftDiv()** - Adds the phone hero and the informations regarding contacts

- **createRightDiv()** - Has a field for the user to send their opinions to the admins

- Course.js / .css

- **createStudyDiv(dataSuccess, dataMessage)** - Adds the two columns of informations regarding the study course name in the URL

- Exam.js / .css

- **createTestsDiv(course_name, data)** - Adds the two columns of informations regarding the study course name in the URL

- Home.js / .css

- **createHeroDiv()** - Creates the top text for the home page
- **createTopicsDiv()** - Creates a row of topics that get updated based on the topics added by admins

- Profile.js / .css

- **initializeAllowedDivs()** - gets the user status from the backend and decides the allowed divs for the future divs
- **addBtnsDiv() -** adds the side btns that switch between divs. It changes based on the status of the user (admin teacher or student)

- profileAdmin.js

- **addPersonalInfoDiv(dataDetails)** - adds the personal info and allows the user to change and update them
- **addManageUsersDiv()** - adds a list of all users for the admin to see and remove if see fit
- **addManageOpinionsDiv()** - allows the admin to see all the opinions created by users
- **addManageAllTransactionsDiv()** - allows the admin to view all the transactions (charging wallet and/or joining a course)
- **addManageTopicsDiv()** - allows admin to delete topics and/or add a new topic with name and url for image

- profileTeacher.js

- **addPersonalInfoDiv(dataDetails)** - adds the personal info and allows the user to change and update them

- **addManageCoursesDiv(dataDetails)** - allows the teacher to delete course and/or add a new course with name (i didn't manage to write the part to add body, price, dependencies etc because of time constraints. If needed be, i'll have em added later)
- **addManageTestsDiv()** - allows teacher to add tests (again this is unfinished and a work in progress because of time constraints)

- profileStudent.js

- **addPersonalInfoDiv(dataDetails)** - adds the personal info and allows the user to change and update them
- **addCoursesDiv(dataDetails, type, btnText)** - adds two divs to the students page, one is for finished courses, and the other is for in progress courses. It allows the student to either study the course or take an exam of the selected course
- **addWalletDiv(dataDetails)** - allows student to see their history of transactions (charging the wallet and/or joining a course) and also allows the student to charge their wallet

- shop.js / .css

- **createTopicsDiv(topics)** - adds a top row of all the topics which allows the user to select a topic and only see the courses belonging to that topic
- **createCoursesDiv(topics)** - creates a table of courses with the option to join the course, and depending to whether the user is a student, already has the course, or has finished it, it answers to the student
- **buyCourse(course)** - handles the process of calling the API with the provided info

BACKEND

- The routing and API calls are handled via the single app.py Python Flask file

- app.py

- **open_file(address):** - Reads and returns JSON data from the specified file.

- **save_file(address, update):** - Writes the provided data to the specified JSON file.

- **def update_session():** - Updates the session user data from users.json.

- **def set_theme(theme):** - Stores the selected theme in the session.
- **def get_theme():** - Returns the currently selected theme, defaults to "dark".
- **def logout():** - Logs out the user by clearing the session.
- **def home_page():** - Renders the home page.
- **def about_page():** - Renders the about page.
- **def contact_page():** - Renders the contact page.
- **def contact_submit():** - Saves submitted contact messages to messages.json.
- **def shop(topic)**: - Renders the shop page for a specific topic.
- **def courses_specific(chosen_course):** - Handles course purchase and updates user session and transactions.
- **def authorization_page():** - Renders the login/register page.
- **def authorization_api(status):** - Processes login or registration, updates user session and users.json.
- **def update_user():** - Updates user profile info and refreshes session.
- **def get(name):** - Returns content of a specified JSON file or session data.
- **def course_page(course):** - Renders the course details page.
- **def get_course_details(course_name):** - Returns course data if the user is logged in.
- **def exam_page(exam):** - Renders the exam page for a specific course.
- **def get_exam_details(exam_name):** - Returns 5 random questions from the exam.

- **def load_profile(profile_div):** - Loads specific sections of the profile (wallet, info, etc.).

- **def update_json(action, json_file):** - Adds or removes data in a JSON file based on the given action.

JSON FILES

- Data is handled using various JSON files

- **Messages.json**
  - Contains the opinions and messages left from students which the admin has the option to read
- **Tests.json**
  - A library of tests for each of the courses made by teachers. The exam will take 5 random tests from the database and shows them to the student
- **Topics.json**
  - A library of main topics and their respective courses and their details such as price, dependencies, title, explanation, and examples which look like tests
- **Transactions.json**
  - Contains all the transaction histories of students charging their wallets and/or joining a course
- **Users.json**
  - Contains a list of all users, admins, and teachers, and their respective details such as username, firstname, last name, age, city, courses created, topics created, courses joined, and etc.