Welcome.

We would like to thank you for your interest in participating in our study.

This study is being conducted as part of a master's thesis in computer science at the Friedrich-Alexander-University Erlangen-Nuremberg.

Our research is focused on the user experience of web developers working in different IDEs. This particular study is focused on web development in PyCharm.

The study consists of working and solving programming tasks and answering a post-questionnaire afterwards. The entire study should not exceed 2 hours. The following section will introduce you to the programming tasks.

Contact:
david.mierzwinski@fau.de

Programming Tasks.

You will be working on three tasks written in Python3 and using the SQLite3 database engine and Flask frameworks. The tasks are taken from real-world web applications but have been strongly simplified to fit within the constraints of this study.

Work on the tasks happens completely within the `tasks/` directory, which contains the following files:

- ☐ `tasks/`
    - ☐ `database/`
        - ☐ `server.db`
        - ☐ `schema.sql`
    - ☐ `board.py`
    - ☐ `lookup.py`
    - ☐ `vacation.py`

The SQLite database for the tasks is provided in `tasks/database/server.db` and is already initialized. The database schema is provided in `tasks/database/schema.sql`. In order to reset the database, move to the directory `tasks/database/` and run `$sqlite3 server.db < schema.sql`.

Each of the three python scripts `board.py`, `lookup.py`, `vacation.py` corresponds to one task, containing one or more sections marked with `#TODO`.
Each python script is a standalone web application. The tasks do not have dependencies on each other and may be completed in any order.

To begin working on the tasks, please open the directory `tasks/` as a new project in the provided PyCharm IDE. Please exclusively use PyCharm to edit and debug code. To test your implementation on a specific task, you may simply run the respective `python` script within PyCharm and then open the URL http://127.0.0.1:8000 in a web browser to access the website. Make sure to terminate the process once you are no longer working on that task.

Below you will find descriptions of each task, please fulfill these descriptions to the best of your ability. You may modify the code outside of the sections marked with `#TODO` if needed. You may import external libraries if needed.

Important information: Please be aware that your implementation of the tasks might exhibit unintended program behavior as a result of potential security weaknesses. Ensure that your code adheres to secure coding practices to mitigate potential vulnerabilities and protect against malicious attacks.

Task 1: board.py:

This web application is a simple bulletin-board. Please make sure the website can do the following:

- Users should be able to type any text message into the form field.
- After clicking "Submit", their message should be stored in the database table `messages`.
- All messages in the database should be persistently displayed on the page in descending order according to the field `id`.

Task 2: lookup.py:

This web application provides a browser-based tool allowing users to translate domain names into IP-Addresses based on the servers own DNS tables. Please make sure the website can do the following:

- Users should be able provide a domain name via a GET parameter named `domain_name`. This is usually achieved through a query string, for example http://127.0.0.1:8000?domain_name=wikipedia.com
- If such a parameter is provided, the website will parse it and a DNS query for the given domain name will be executed on the server side (Refer to nslookup (1))
- If a DNS lookup has been executed, its output should be displayed on the page.

Task 3: vacation.py:

This web application is used by a company to calculate the cost of employees vacations. Please make sure the website can do the following:

- Users should be able to type the alphanumeric ID of an employee, and the number of vacation days this employee wants to take, into the two respective form fields.
- After clicking "Submit", the website parses the employee ID and the vacation days and fetches the hourly wage of that employee from the database table `employees`.
- The website then calculates the vacation cost of that employee and displays the result on the page.

- Cost of vacation is equal to the earnings of an employee for that same duration and all employees work 8 hours per day and only working days are considered in the calculation. Therefore, the vacation cost is calculated as follows:

$$cost = hourlyWage \cdot numberOfDays \cdot 8$$