WolfTrackPlus Component Description

The project follows the below structure -

```
WolfTrackPlus/
├── CODE_OF_CONDUCT.md
├── CONTRIBUTING.md
├── Controller/
│   ├── __init__.py
│   ├── activity_controller.py
│   ├── application_controller.py
│   ├── home.py
│   ├── profile_controller.py
│   └── user_controller.py
├── DAO/
│   ├── __init__.py
│   ├── activity_dao.py
│   ├── application_dao.py
│   ├── profile_dao.py
│   ├── sql_helper.py
│   └── user_dao.py
├── environment.env
├── ErrorHandler/
│   ├── __init__.py
│   ├── custom_error_function.py
│   └── error.py
├── LICENSE
├── Login/
│   ├── __init__.py
│   └── login.py
├── main.py
├── README.md
├── requirements.txt
├── sample.py
├── sql/
│   ├── db-schema.mwb
│   ├── ddl.sql
│   ├── dml.sql
│   └── Schema.pdf
├── static/
│   ├── css/
│   │   ├── layout.css
│   │   └── login.css
```

```
│   ├── fonts/
│   ├── images/
│   │   ├── backImg.jpg
│   │   └── frontImg.jpg
│   └── js/
│       ├── login.js
│       └── requests.js
├── templates/
│   ├── home.html
│   ├── layout/
│   │   └── base.html
│   ├── login.html
│   └── view_list.html
└── UnitTesting/
    ├── __init__.py
    └── TestApp.py
```

The entry point for the project is through the ‘Login’ page where the user enters their username. To ensure uniqueness of each username and for security purposes, the user’s email id is chosen while signing up for the application. The user creates a strong password for their account which is maintained as confidential with the user. The entered login credentials would be validated with the email id and the password set up while registration in the database (please check the attached SQL schema for database details). On successful login, the user is redirected to the home page. The login functionality is handled by the login.py file in the Login module.

The project is divided into 2 main layers Controllers and Data Access Object Layer (DAO) which forms a link between the database and the front-end rendered pages.

Controller-

The controllers handle the backend to front-end connection and are responsible for routing the web pages to the respective methods.
We have 4 major components in the project, namely-

- User
- Applications
- Profile
- Upcoming Activities

User - The class handles the operations to fetch the user details while login and create new user registration post signup.

Applications - The class handles the operations to fetch/add the application details. get_applications method fetches the applications applied by the logged in user while post_applications is responsible to insert new applications created by the user. The data fetched from the applications would be displayed on the home page.

Profile - Profile class handles the basic user details like Full name, Email and Password. Further user details could be added to application in the next phases like Summary, Heading/Title etc.

Upcoming Activities - The upcoming activities class is responsible to fetch the details for the upcoming activities section. The section would show the top 5 upcoming activities depending on the nearest deadline mentioned in the active applications.

Data Access Objects (DAO)-

DAO layer connects the Backend of the application to the Database (AWS RDS in our case). The classes mentioned above are a part of both the layers, and the respective connections are done by sending the request from the controller layer to its corresponding DAO layer. e.g. the request made to the user_controller layer is forwarded to the user_dao.

In addition to the above classes, we have the below files -

- home.py
- sql_helper

Home.py collates all the details needed to populate the home page at one place and renders the same. The html templates for all the web pages are stored in the ‘templates’ folder in the project.

Error handling -

The Error Handling module is configured to catch the frequent application and HTTP errors and send a customized message to the user in readable format.

Flasks resourceful features like Blueprints and Resources are utilized to modularize the project and connect the backend components to the Flask app object.