The project follows the below code structure -

```
├── CITATION.md
├── CODE_OF_CONDUCT.md
├── CONTRIBUTING.md
├── LICENSE
├── README.md
├── Simplii.mwb
├── Simplii.sql
├── UnitTests
│   ├── config.json
│   ├── database_connection_test.py
│   ├── database_crud_test.py
│   ├── database_tables_test.py
│   ├── login_page_test.py
│   ├── sample_test.py
│   └── sql_helper.py
├── app.py
├── docs
│   ├── Phase\ 2
│   │   ├── DB_Schema.jpeg
│   │   ├── ImprovementsDoc.md
│   │   ├── Phase2_Video.mp4
│   │   ├── Screenshots
│   │   │   ├── WhatsApp\ Image\ 2021-11-04\ at\ 1.46.27\ AM\ (1).jpeg
│   │   │   ├── WhatsApp\ Image\ 2021-11-04\ at\ 1.46.27\ AM\ (2).jpeg
│   │   │   ├── WhatsApp\ Image\ 2021-11-04\ at\ 1.46.27\ AM\ (3).jpeg
│   │   │   ├── WhatsApp\ Image\ 2021-11-04\ at\ 1.46.27\ AM\ (4).jpeg
│   │   │   ├── WhatsApp\ Image\ 2021-11-04\ at\ 1.46.27\ AM.jpeg
│   │   │   ├── WhatsApp\ Image\ 2021-11-04\ at\ 1.48.18\ AM\ (1).jpeg
│   │   │   └── WhatsApp\ Image\ 2021-11-04\ at\ 1.48.18\ AM.jpeg
│   │   └── proj2rubric.md
│   └── Phase1
│       ├── Component\ Description.pdf
│       ├── Footer\ Screenshot.PNG
│       ├── Screenshot_Header.PNG
│       ├── Task\ list\ Screenshot.PNG
│       ├── VideoPresentation.mp4
│       ├── architectureImages
│       │   ├── SE_ERDiagram.png
│       │   └── webappFlowchart.png
│       ├── lintingEvidence
│       │   └── app.pyLinting.png
│       ├── proj1rubric.md
│       └── proj1rubricComments
├── issue_templates
│   ├── Bug_Report.md
│   └── FeatureRequest.md
├── pyproject.toml
├── requirements.txt
├── sql
│   └── ddl.sql
├── src
│   ├── __init__.py
│   ├── __pycache__
│   │   └── __init__.cpython-38.pyc
│   ├── controller
│   │   ├── __init__.py
│   │   ├── __pycache__
│   │   │   ├── __init__.cpython-38.pyc
│   │   │   └── task_controller.cpython-38.pyc
```

```
│   │   ├── category_controller.py
│   │   ├── home_controller.py
│   │   ├── task_controller.py
│   │   └── user_controller.py
│   ├── error_handler
│   │   ├── __init__.py
│   │   └── error.py
│   ├── login
│   │   ├── __init__.py
│   │   ├── __pycache__
│   │   │   ├── __init__.cpython-38.pyc
│   │   │   └── login.cpython-38.pyc
│   │   └── login.py
│   ├── models
│   │   ├── __init__.py
│   │   ├── __pycache__
│   │   │   ├── __init__.cpython-38.pyc
│   │   │   ├── sql_helper.cpython-38.pyc
│   │   │   └── task_model.cpython-38.pyc
│   │   ├── category_model.py
│   │   ├── sql_helper.py
│   │   ├── task_model.py
│   │   └── user_model.py
│   └── routing.py
├── static
│   ├── __init__.py
│   ├── css
│   │   ├── layout.css
│   │   └── login.css
│   ├── images
│   │   ├── backImg.jpg
│   │   ├── frontImg.jpg
│   │   ├── planner.webp
│   │   └── wolfpack.jpeg
│   ├── quotes.csv
│   ├── scripts
│   │   ├── name-block-scripts.js
│   │   └── task-block-scripts.js
│   ├── styles.css
│   ├── tasks
│   │   ├── completed
│   │   └── todo
│   └── user_information.json
├── templates
│   ├── edit_task.html
│   ├── home.html
│   ├── index.html
│   ├── layout
│   │   ├── base.html
│   │   ├── nav.html
│   │   └── user_details.html
│   ├── login.html
│   ├── tasks
│   │   ├── all_tasks.html
│   │   ├── backlog.html
│   │   ├── future_tasks.html
│   │   └── this_week.html
│   ├── view_all_tasks.html
│   └── view_user_details.html
└── test.py
```

The entry point for the project is through the ‘Login’ page where the user enters their username.
To ensure the 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 Models 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 modal methods.
We have 3 major controllers in the project, namely-
- User
- Tasks
- Category

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

Tasks - The class handles the operations to fetch/add the details of the tasks.
get_tasks() method fetches all the tasks created for the user.
create_task() is responsible to create new tasks for the user.
delete_tasks() method deletes the tasks based on the parameter values provided by the user.
update_task() is responsible to update specific tasks by the user.

Category - The category class handles the categories/tags for each of the tasks.
get_category() method fetches all the categories defined to tag the tasks.
create_category() is responsible to create new categories for the user.
update_category() is responsible to update specific categories by the user.

Models -
Models get the request from the controller and then interact with the database to fetch the request.
We have 3 major models in the project, namely-
- User
- Tasks
- Category

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

Tasks - The class handles the operations to fetch/add the details of the tasks.
get_all_tasks() method fetches all the tasks from the database and returns the result to the controller.
get_backlog_tasks() method fetches all the backlog tasks from the database and returns the result to the controller.
get_thisweek_tasks() method fetches all this week tasks from the database and returns the result to the controller.
get_future_tasks() method fetches all the future tasks from the database and returns the result to the controller.
create_task() is responsible to create new tasks and update the database with the new values.
delete_tasks() method deletes the tasks from the database based on the parameter values provided by the controller method.
update_task() is responsible to update specific tasks based on the perimeter by the controller.

Category - The category class handles the categories/tags for each of the tasks.
get_category() method fetches all the categories from the database and returns the result to the controller.
create_category() is responsible to create new categories and update the database with the new values.
update_category() is responsible to update specific categories based on the perimeter by the controller.

Templates -
The templates folder contains all the HTML pages designed to create the user interface of the app.

Static -
The static folder contains all the static files for the project. It mainly consists of all the images in the static/images folder and the CSS required for the HTML styling in the static/css folder

UnitTests -
All our test files are present in the Unittests folder. It contains the testcases to check the Database and UI connections and responses.

Docs/phase2 -
All the document related to phase2 of the project is present in Docs/phase2 folder.