## **Name- Rajarshi Mazumder**

#### \*\*\*The database operations might be a little slow because of the hosting platform. Please excuse any inconvenience caused by it\*\*\*

## **PVP Project Documentation-**

The project has been made with **Python**, using the **Django** framework.

It uses Python, along with **HTML/CSS/JS** for the frontend, and **SQLite** database to store the data.

## TLDR:

#### **Category** class- Creates categories like **SCIENCE, HUMANITIES, ARTS** etc.

Each category has a **passing mark,** and students in a category must get a total of this **passing mark** or more in 2 subjects of that category.

By default it has 2 categories - SCIENCE and HUMANITIES

#### **Subject** class- Creates and deletes **subjects** that each belong to a **category**.

By default it has 5 subjects- Maths, Biology, Japanese, English, History

#### **Student** class- Creates and deletes **students** that each belong to a **category**.

#### **TestConditions** class- Creates **test\_condition** that evaluates if a student passes or fails

#### **Evaluation\_Strategy** class- Creates **evaluation\_strategy** that creates logic for determining if a student passes or fails.

**Project Directory structure-**

**PVP\_students\_test\_app**, which contains all the relevant Python files for this project.

**templates** directory- contains the HTML files

**Static** directory- contains all the CSS/ JS files

### **Modules/ Classes**

**urls.py**- Contains routing information for the different urls, that handle creation of students, subjects, categories, test\_conditions etc

**views.py** - The starting point for the execution of the app, which contains the method to handle the route to the home page( **index**() ).

-The **index**() function handles creating and sending the context data to the HTML frontend.

-The **default\_data\_check**() function checks whether the app has some default data in it as soon as the home page loads

-The **get\_default\_colors**() function gets the default color schemes for the app

**Views\_container-**

Contains the other views files that handle the functionalities such as student creation, subject creation, category creation, test\_condition etc

**category\_views.py**- Handles creation of new categories

**student\_views.py**- Handles creation and deletion of students

**subject\_views.py**- Handles creation and deletion of subjects

**test\_condition\_views.py**- Handles updation of test conditions

**models.py-**  This file contains the classes for **Student**, **Subject**, **Category** and **Test\_Conditions**. All of these classes inherit from the **Django models**.**Model** class, which allows them to be created as individual tables in the database,

**forms.py**- This file creates individual forms from the **Student**, **Subject** and **Category** models, which in turn makes it very easy to handle inputs for each of these classes to create data in the database.

**apps.py**- configures the app to run

**admin.py**- registers the different models to the Django admin console

**evaluations-**

**Evaluate\_Results**- This class handles evaluation of each student’s result.

**evaluation\_strategies.py**-

**-Evaluation\_Strategy-** This is an abstract class that contains the definition for the evaluate method. New evaluation strategies are created from this class

**-Evaluate\_By\_Total\_Score-** This class inherits the **Evaluation\_Strategy** class, to evaluate the student result based on total\_score

**-Evaluate\_By\_Marks\_In\_Category-** This class also inherits the **Evaluation\_Strategy** class, to evaluate the student result marks in his category

**data\_state\_handle-**

**default\_data\_creation.py-** contains functions to create the default data for the app

**default\_data-** contains some default data like enums for **Student\_Evaluation\_Results**, colors for the different categories( **Default\_Category\_Colors**) **,** etc

**Migrations-**

Contains django generated files created during data migration