A red text on a white background

Description automatically generated

|  |  |
| --- | --- |
| Project Title | TaxMaster: A Web-based Portal for Tax Calculation |
| Project Estimated Start Date | 26.09.2024 |
| Project Estimated End Date | 17.10.2024 |
| Candidate Name | Sanchari Ray |
| Mentor Name | Samarth Shrivastava |

**GEBS GET Training Project Document**

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| Doc Version | Submitted Date | Reviewed By and Date | Comment/Remarks |
|  |  |  |  |

Reviewer(s)

|  |  |
| --- | --- |
| Name | Title |
|  |  |

Approver(s)

|  |  |
| --- | --- |
| Name | Title |
|  |  |

Document Reference(s)

|  |
| --- |
| Releated Documents : |
|  |

1.Introduction

1.1 Scope

* **Project Name:**  TaxMaster: A Web-based Portal for Tax Calculation
* **Objective:** The objective of TaxMaster is to create an user-friendly platform where individuals can securely upload their financial details. Utilizing predefined tax rules and regulations, the system will automatically calculate taxes and provide personalized suggestions to optimize tax returns. This platform aims to simplify the tax planning process, enhance user experience, and maximize tax savings by offering accurate, reliable, and compliant tax optimization strategies. The portal also ensures data security, user privacy, and continuous updates to reflect the latest tax laws, making tax management efficient and accurate.
* **Goal:** The goal of TaxMaster is increase user enhancement which will reduce the complexity of tax calculations to any users. It will help maximum to maximum users to maximize their tax savings through optimized suggestions and the platform will have the ability to give security to the users uploading their confidential data.

1.2 Technology Stack

**Software Requirements:**

* *VSCODE(Visual Studio Code)*  for performing the Frontend and Backend functionalities.
* Install and Setup *SQL Server Management Studio (SSMS)* or any other server which supports MS SQL.

**Technology Used:**

* Programming Lnaguage : **Python**
* Technology for Frontend : **Django Template(HTML,CSS)**
* Technology for Backend : **Django**
* Database **: Microsoft SQL Server**

**Prerequisites to be installed:**

* Python Installation of version 3.9 or later from Python Official Website
* Installation of Django:

*‘pip install Django’*

* Installation of Django API:

*‘pip install djangorestframework’*

1.3 Glossary or Terminology

* **Old Tax Regime :** The tax regime that allows to claim deductions like HRA,LTA,Section 80c etc.
* **New Tax Regime :** The tax regime with lower tax rates but without deductions.
* **Annual Income :** The total income earned by the individual in a financial year before any deductions.
* **Taxable income :** Income on which tax is computed after deductions.
* **Tax Rates :** Percentage of tax applicable on income based on slabs.
* **Standard Deduction :** A flat deduction available to all salarised taxpayers.
* **Section 80C Deduction :** Deduction including investments in PPF, NSC, ELSS etc.
* **Section 80D Deduction :** Deductions related to medical insurance premiums.
* **Surcharge Rates :** Additional tax on individuals with income exceeding specific thresholds.
* **Cess Rate :** An additional tax calculated as 4% of the total tax liability.
* **Tax Advisor :** An financial expert who helps individuals to plan and optimize their tax situation.
* **JWT(Json Web Tokens) :** JWT is used for transmitting secure, verifiable information between two parties in web applications.

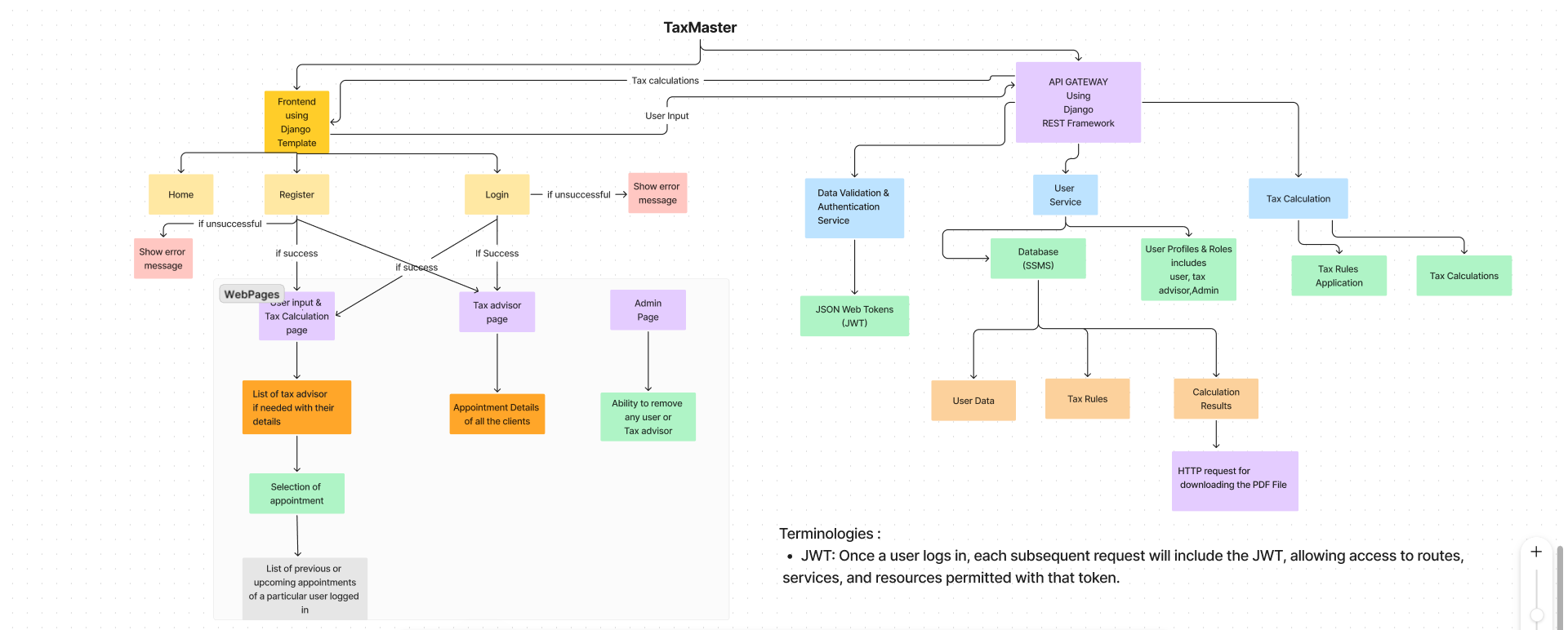
1.4 High Level Design

**Detailed Design of UserWebPage:**

**A diagram of a company

Description automatically generated with medium confidence**

**High Level Design:**

****

[*The link for figma*](https://www.figma.com/board/DjV4dPlbyqHElUMZG0y5wA/Tax-Calculation-Web-Design?node-id=40-1427&t=QRLACmeM8D050WjK-1)*:* [*https://www.figma.com/board/DjV4dPlbyqHElUMZG0y5wA/Tax-Calculation-Web-Design?node-id=40-1427&t=QRLACmeM8D050WjK-1*](https://www.figma.com/board/DjV4dPlbyqHElUMZG0y5wA/Tax-Calculation-Web-Design?node-id=40-1427&t=QRLACmeM8D050WjK-1)

1.5 Programming Standards

1. **Separation of Concerns(SOC):**
   1. *HTML ,CSS Separation in Frontend:*

Used HTML,CSS code separately for code cleanliness.

*1.2 Views and Forms Separation in Backend:*

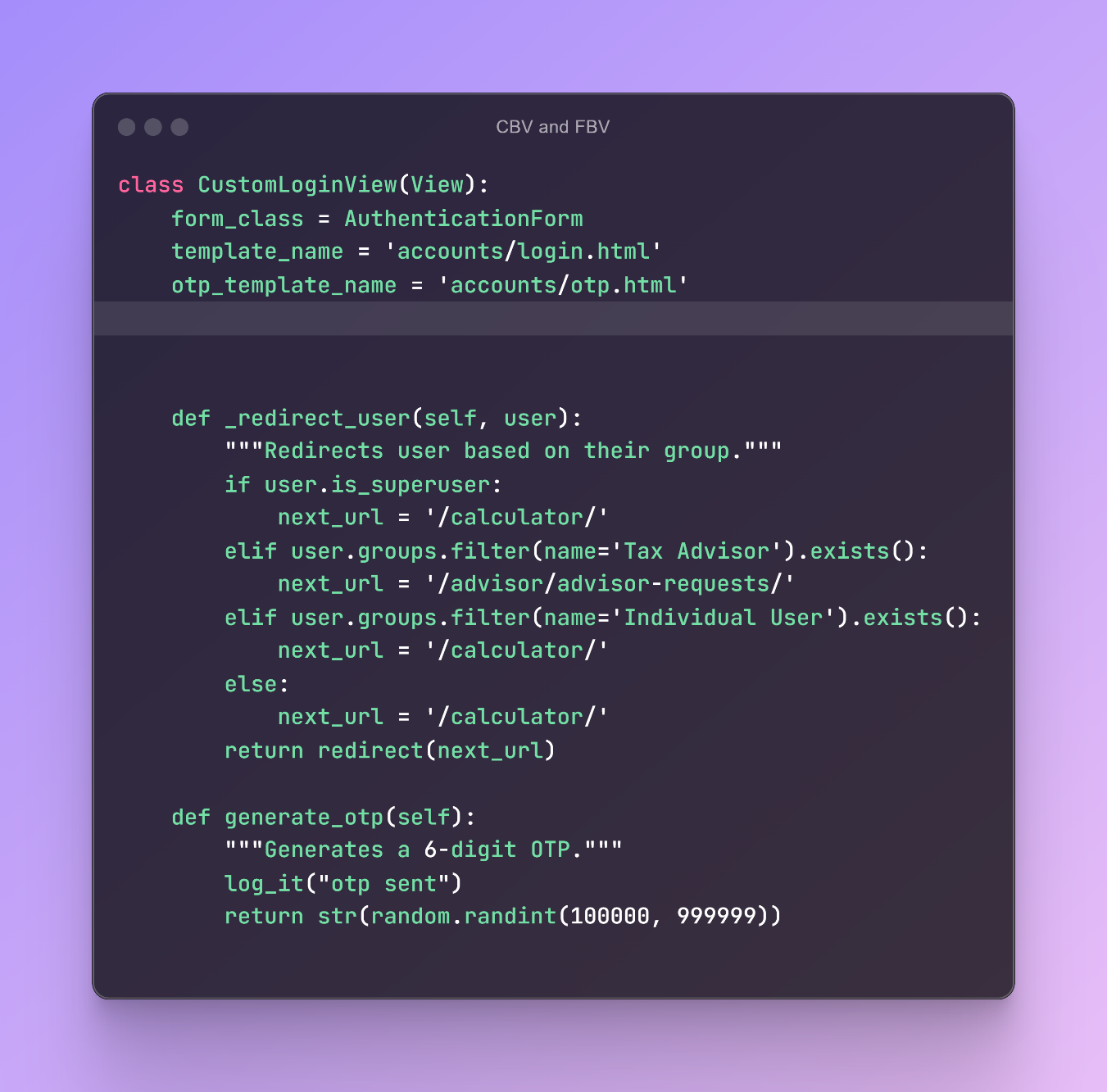
Business logic such as authentication, OTP handling etc. is separated into dedicated views and the error rules and messages are defined inside their respective forms.

1. **Django Best Practices:**

*2.1 Usage of Class-Based Views and Function-Based Views:*

Used Class-based views like **‘CustomLoginView’** is made which uses a class-based view structure which helps to keep the login logic organized. Simpler tasks such as checking the OTP or processing the login form are handled by function-based views.

***BELOW IS THE CODE SNIPPET FOR Class-Based Views and Function-Based Views:***



*2.2 Django Decorators:*

Decorators like **‘@login\_required’** and **‘@user\_passes\_test’** are used to manage access control, ensuring that specific views are only accessible to logged-in users.

***BELOW IS THE CODE SNIPPET FOR DJANGO DECORATORS LIKE ‘@login\_required’:***

A screenshot of a computer program

Description automatically generated

1. **Validation Logic:**

*3.1 Custom validation methods* (e.g., **clean\_time** and **clean\_requested\_date**) are used to enforce business rules (e.g., ensuring selected times are within the specified slot range and that the requested date is not in the past).

3.2 The code raises **forms.ValidationError** when validation fails, providing feedback to users, and helps the users to rectify the error.

*3.3 Usage of Try and Except block:*

Implemented to handle exceptions-errors that occur during the exception of code. These will help to catch the errors and handle them gracefully.

***BELOW IS THE CODE SNIPPET FOR Try and Except Block:***

A screenshot of a computer screen

Description automatically generated

1. **Other Standards:**

4.1 Maintained a Github Repository where a main branch is created and regularly merge the branches (like working branches) into the main branch only after ensuring that the code is error-free.

4.2 Maintained logs as well where all the error logs as well as success logs with timestamps are visible.

1.6 Components

In my project, I am maintaining mainly 5 components which are:

* **accounts** : In this component, I have maintained the functionalities like Registration, Login, OTP Validations , login for forget password and setting up a new password.
* **advisor**: In advisor component, all the functionalities related to tax-advisor is maintained like, maintaining the list of available tax-advisors, maintaining the tax-advisor profile and all the requests been sent to a tax-advisor for an appointment are maintained.
* **appointment:** Herethe logic behind the appointment related functionalities are coded where requesting an appointment, managing an appointment(accepting or declining), sending the appointment requests to the respective tax-advisor is being maintained.
* **base:** The landing page, the dashboard for user regarding their appointment and the user’s details are maintained in the base component**.**
* **calculator :** The main component of the project is the calculator where all the logic related to tax calculations , printing a pdf , showing the tax-advisor list to an user, the tax slabs are mentioned in this component.

1.7 Pre-requisite

* Taxation Rules and Policies of Indian Government.
* Aware of the salary structure which includes basic pay, allowances, bonuses, deductions etc.
* Understanding of Django’s project structure, views, models and templates.
* Knowledge on Python as Django is built in it.
* Basic frontend knowledge for user interfaces like HTML, CSS.
* Basic understanding of database designs, SQL queries and how to manage salary and tax data in relational database(Microsoft SQL Server).

1.8 User Guide

//User manual to go through the project

2.Testing

2.1 Test Scenario(s)

* **Test Case 1: Registration Page:**
* Validations of all the user-details:
* For all the fields in the registration page, users-details are validated like the first name and last name of an user cannot be in small letters.
* If an user with an username is registered , then another user cannot use that username.
* Password validators like password must contain at least one uppercase letter, one lowercase letter, one digit, and one special character is tested.
* If in the ‘Confirm Password’ field, different password id given then it will throw an error message. An user should give same password and then only the registration will be successful.
* **Test Case 2 : OTP Validations :**
* An OTP will be sent to the registered mail-id and the following testing are done:
* The OTP must be of 6-digits otherwise the OTP will throw an error.
* If an user inputs wrong 6 digits OTP, it will throw an error.
* An user cannot use previously sent correct OTP, it will throw an error.
* **Test Case 3 : Login Page:**
* An user should give their username and password correctly .
* If an user is giving wrong details, the user cannot login.
* **Test Case 4 : Requesting an Appointment to a Tax-Advisor:**
* An user can only request appointment from the present date to next 1 year.
* If an user tries to book appointment in the previous , it will throw an error message.
* An user needs to book appointment within the specified time in the slots, otherwise it will throw an error message and the appointment will not succeed.