

INTERNSHIP REPORT
BACHELOR OF TECHNOLOGY
in
COMPUTER SCIENCE AND ENGINEERING
BY
BABASAHEB BHIMRAO AMBEDKAR UNIVERSITY
Under

Supervision of
Mr Singhasan Prasad
On **PROJECT**- WEB DEV BASED ON DJANGO WITH SERVER POSTGRESQL



R.D.S.O{Lucknow}
(Duration:3rd JUNE, 2024 to 3rd JULY, 2024)



RESEARCH DESIGN & STANDARDS ORGANISATION
MANAKNAGAR LUCKNOW-226011

SUBMITTED TO- MR. ANUBHAV AGARWAL
SUBMITTED BY- ABHISHEK KR TRIPATHI

ACKNOWLEDGEMENT

First I would like to thank **Mr.Singhasan Prasad**, of **RDSO, LUCKNOW** for giving me the opportunity to do an internship within the organization.

I also would like all the people that worked along with me **RDSO, LUCKNOW** with their patience and openness they created an enjoyable working environment.

It is indeed with a great sense of pleasure and immense sense of gratitude that I acknowledge the help of these individuals.

I am highly indebted to **Director NMP Verma** for the facilities provided to accomplish this internship.

I would like to thank my Head of the Department **Dr. RA Khan** for his constructive criticism throughout my internship.

I would like to thank **Dr.HR Khan**, College internship coordinator **MRs Neha Gupta** internship coordinator Department of CSE for their support and advices to get and complete internship in above said organization.

I am extremely great full to my department staff members and friends who helped me in successful completion of this internship.

ABSTRACT

Django Introduction

Django is a high-level Python web framework that enables rapid development of secure and maintainable websites. [Built by experienced developers, Django takes care of much of the hassle of web development, allowing you to focus on writing your app without reinventing the wheel¹](#). Here are some key points:

1. **Complete:** Django follows the “Batteries included” philosophy, providing almost everything developers need out of the box. It seamlessly integrates various components, follows consistent design principles, and offers extensive documentation.
2. **Versatile:** Django can be used to build various types of websites, from content management systems to social networks. It works with any client-side framework and delivers content in multiple formats (HTML, RSS feeds, JSON, XML).
3. **Secure:** Django helps developers avoid common security mistakes. It manages user accounts securely, stores passwords as hashes, and protects against vulnerabilities.

Organization Information:

Research Designs and Standards Organisation (RDSO) is an ISO 9001 research and development organization under the Ministry of Railways, India. Established in **1957**, RDSO serves as the technical advisor and advisor for the Railway Board and Zonal Railways. [Its headquarters are located in Manak Nagar, Lucknow¹²](#).

RDSO plays a crucial role in identifying and assimilating technology suitable for Indian operating and climatic conditions. It ensures the rightful adaptation of necessary technologies, standards, and specifications. [As an applied research facilitator, RDSO contributes significantly to the development and advancement of railway rolling stock, permanent way infrastructure, and other critical components³](#).

Programs and opportunities:

Web development is a multifaceted domain that encompasses various technologies, methodologies, and paradigms. It goes beyond mere coding; it involves designing, building, and maintaining websites and web applications. Key aspects of web development include:

1. **Front-End Development:** This focuses on creating the user interface (UI) that users interact with directly. It involves HTML, CSS, and JavaScript to build responsive, visually appealing web pages.
2. **Back-End Development:** The backbone of web applications, back-end development deals with server-side logic, databases, APIs, and business logic. Popular back-end frameworks include Django, Ruby on Rails, and Node.js.
3. **Full Stack Development:** Full stack developers work on both front-end and back-end components. They understand the entire web development stack, from databases to user interfaces.

Methodologies:

We follow a structured methodology for our projects which starts from designing the solution to the implementation phase. Well planned Project reduces the time to deliver the project and any additional ad-hoc costs to our clients, hence we dedicate majority of our time understanding our clients business and gather requirements. This ground up approach helps us deliver not only the solution to our clients but also add value to your investments.

Key parts of the report:

Under each division we further provide specific industry solutions on focused domains with cutting edge technologies.

Benefits of the Company/Institution through our report:

Under each division we further provide specific industry solution on focused domains with cutting edge technologies. We emphasize on building relationships with our clients by delivering projects on time and within budget.

INDEX

S.no	CONTENTS	Page no
1.	Introduction	1
1.1	Modules.....	2
2.	Analysis	3
3.	Software requirements specifications	4
4.	Technology.....	5
4.1	HTML	5
4.2	DJANGO.....	6
4.3	POSTGRES	7
4.4	PYTHON.....	7
4.5	POSTGRES Data Base	8
5.	Coding.....	10
6.	Screenshots.....	11
7.	Conclusion	16
8.	Bibilography.....	24

Learning Objectives/Internship Objectives

- Internships are generally thought of to be reserved for college students looking to gain experience in a particular field. However, a wide array of people can benefit from Training Internships in order to receive real world experience and develop their skills.
- An objective for this position should emphasize the skills you already possess in the area and your interest in learning more
- Internships are utilized in a number of different career fields, including architecture, engineering, healthcare, economics, advertising and many more.
- Some internship is used to allow individuals to perform scientific research while others are specifically designed to allow people to gain first-hand experience working.
- Utilizing internships is a great way to build your resume and develop skills that can be emphasized in your resume for future jobs. When you are applying for a Training Internship, make sure to highlight any special skills or talents that can make you stand apart from the rest of the applicants so that you have an improved chance of landing the position.

WEEKLY OVERVIEW OF INTERNSHIP ACTIVITIES

1st WEEK	DATE	NAME OF THE TOPIC/MODULE COMPLETED
	1/05/24	Introduction of DJANGO
	9/05/24	Features of DJANGO
	10/05/24	Introduction of Python
	11/05/24	Introduction of POSTGRES
	12/05/24	Continuing POSTGRES
	13/05/24	Understanding different types of class and collection

2nd WEEK	DATE	NAME OF THE TOPIC/MODULE COMPLETED
	15/05/24	Introduction to web development
	16/05/24	Understanding web development tools
	24/05/24	Continuing understanding tools
	18/05/24	Introduction to Django project
	19/05/24	Continuing with project
	20/05/24	Introduction to app creation

3rd WEEK	DATE	NAME OF THE TOPIC/MODULE COMPLETED
	22/05/24	Introduction to HTML
	23/05/24	Introduction to web programming
	24/05/24	Understanding what is Internet website, web request and web response
	25/05/24	Understanding client side web technologies VS server side web technologies
	26/05/24	Introduction to CSS
	27/05/24	HTML continued

4th WEEK	DATE	NAME OF THE TOPIC/MODULE COMPLETED
	29/05/24	Understanding UI creation
	30/05/24	Introduction to web server controls
	31/05/24	Working with different web server controls
	01/06/24	Standard controls , validation controls
	02/06/24	Creating users on POSTGRES server
	03/06/24	Project session

1. INTRODUCTION

Certainly! Creating an **authentication project using Django** involves building a robust user management system for web applications. [Django, a powerful Python web framework, provides a comprehensive authentication system that handles user registration, login, password management, and permissions¹](#). Here are the key steps:

1. **User Model:** Django's authentication revolves around the `User` model. It represents users interacting with your site and includes attributes like username, email, and password. [You can create users programmatically using the `create_user\(\)` function or interactively via the Django admin interface¹](#).
2. **Superusers:** Superusers (admin staff) have special privileges. You can create them using the `createsuperuser` command, which prompts you for a password. [Superusers can manage other users and access administrative features¹](#).
3. **Password Management:** Django securely hashes passwords and does not store them in plain text. To change a user's password, use the `set_password()` method. [Avoid manipulating the password attribute directly¹](#).
4. **Customization:** If your authentication needs differ from the default, Django allows extensive customization. [You can tailor authentication to fit your project's requirements¹](#).

1.1Module Description:

Shopper:

=====

- 1) User registers the site.
- 2) Products will be showed
- 3) If user selected the product and then save
- 4) User selected product is send to the Order.
- 5) If user wants to buy the product they can also buy.

Supplier

=====

- 1) send product details
- 2) send payment verification
- 3) Store buying detail
- 4) Store line items using join product and order

Order

=====

- 1) Store all the user details.
- 2) Order detail

2. SYSTEM ANALYSIS

2.1 Requirement Analysis

Existing System:

In an Existing we address these challenges and present an approach to efficient, incremental consolidation of data-intensive flows. Following common practice, our method iterates over information requirements to create the final design. we show how to efficiently accommodate a new information requirement to an existing design and also, how to update a design in lieu of an evolving information requirement. The final design satisfying all requirements comprises a multi-flow. As ‘coal’ is formed after the process and extreme compaction of layers of partially decomposed materials¹, Co AI processes individual data flows and incrementally consolidates them into a unified multi-flow.

Proposed System

Following the previously proposed set of flow transformations in the context of ETL processes in Co AI we extend this set considering also the associative property of n-array operations (e.g., Join) and thus rely on the following four flow transformations used for reordering the operations. Swap Applied to a pair of adjacent unary operations, it interchanges the order of these operations. Distribute/Factorize. Applied on a unary operation over an adjacent n-array operation, it respectively distributes the unary operation over the adjacent nary operation or factorizes several unary operations over the adjacent n-array operation. Merge/Split. Applied on a set of adjacent unary operations, it respectively merges several operations into a single unary operation or splits a unary operation into several unary operations. Re-associate. Applied on a pair of mutually associative n-array operations, it interchanges the order in which these operations are executed.

3. SOFTWARE REQUIREMENTS SPECIFICATIONS

3.1 System configurations

The software requirement specification can produce at the culmination of the analysis task. The function and performance allocated to software as part of system engineering are refined by established a complete information description, a detailed functional description, a representation of system behavior, and indication of performance and design constrain, appropriate validate criteria, and other information pertinent to requirements.

Software Requirements:

- Operating system : Windows 11 Ultimate.
- Coding Language : H T M L & P Y T H O N
- Front-End : Visual Studio 2024 Professional.
- Data Base : POSTGRES Server 20024.

Hardware Requirement:

- System : Intel I5 12th GEN 2.5 GHz.
- Hard Disk : 1TB.
- Ram : 16GB.

4. TECHNOLOGY

4.1 HTML:

1. Introduction:

- HTML is the standard markup language used to create web pages. It combines two essential concepts:
 - **Hypertext**: Defines links between web pages, allowing users to navigate from one page to another.
 - **Markup language**: Structures text content using tags, making it machine-readable.
- [HTML annotations \(tags\) define how text should be manipulated and displayed on web pages¹.](#)

2. Features of HTML:

- **Ease of Learning**: HTML is straightforward and easy to learn.
- **Platform-Independent**: Works across different operating systems and browsers.
- **Multimedia Support**: Allows embedding images, videos, and audio.
- **Hypertext**: Enables linking between pages.
- [Markup Language: Uses tags to structure content¹.](#)

3. HTML Elements and Tags:

- **Elements**: Consist of an opening tag, content, and a closing tag. Always include closing tags to avoid unexpected behavior.
- **Basic Structure**:
 - `<!DOCTYPE html>`
 - `<html>`
 - `<head>`
 - `<title>First HTML Code</title>`
 - `</head>`
 - `<body>`
 - `<h2>Welcome To GFG</h2>`
 - `<p>Hello Geeks</p>`
 - `</body>`
 - `</html>`
 - `<html>`: Root element containing all other elements.
 - `<head>`: Contains behind-the-scenes elements (e.g., metadata, styles).
 - `<title>`: Displays the webpage title in the browser.
 - [<body>: Holds visible content¹.](#)

4. HTML Page Structure:

- Essential building blocks:
 - `<!DOCTYPE html>`: Declares the document as an HTML file.
 - `<html>`: Root element.
 - `<head>`: Includes metadata (e.g., styles, title).

- `<body>`: Displays visible content.
- [Additional elements can be added within `<head>` \(e.g., `<style>`, `<base>`\)¹.](#)

5. Why Learn HTML?:

- Foundation for web development.
- Understand how web pages work.

4.2 DJANGO:

Django is a high-level Python web framework that empowers rapid development of secure and maintainable websites. [Created by experienced developers, Django streamlines web development, allowing you to focus on writing your app without reinventing the wheel¹](#). Here are the key points:

1. Features:

- **Fast Development:** Django's built-in features (like authentication, ORM, and templating) accelerate development.
- **Security:** Django handles common security concerns (e.g., SQL injection, cross-site scripting) out of the box.
- **Scalability:** It scales seamlessly, making it suitable for both small projects and large applications.
- **Modularity:** Django encourages reusable components (apps) for better organization and maintainability.

2. Components:

- **Models:** Define data models using Python classes. Django's ORM maps these to database tables.
- **Views:** Handle user requests, query the database, and render templates.
- **Templates:** Use Django's template language to create dynamic HTML pages.
- **URLs:** Map URLs to views using URL patterns.
- **Admin Interface:** Django provides an automatic admin interface for managing data.

3. Community and Ecosystem:

- Django has an active community, extensive documentation, and a rich ecosystem of third-party packages.
- It's widely used in various domains, including news websites, e-commerce platforms, and social networks.

4.3 POSTGRES:

PostgreSQL, commonly pronounced as "Post-GRES," is an open-source object-relational database system with a rich history spanning over 35 years. Let's explore what makes PostgreSQL a powerful choice for developers and organizations:

1. Foundation and Origins:

- PostgreSQL emerged from the **POSTGRES project** at the **University of California, Berkeley** in **1986**.
- It has undergone **active development** ever since, earning a strong reputation for its reliability, data integrity, and robust feature set.

2. Key Features:

- **SQL Language:** PostgreSQL extends the SQL language, making it suitable for complex data workloads.
- **Scalability:** It safely handles large datasets and scales seamlessly.

- **Extensibility:** Developers can define custom data types, functions, and even write code in different programming languages without recompiling the database.
 - **ACID Compliance:** PostgreSQL ensures atomicity, consistency, isolation, and durability for transactions.
 - **Powerful Add-ons:** Notably, **PostGIS** provides geospatial capabilities.
3. **Data Types and Integrity:**
 - **Primitives:** Integer, Numeric, String, Boolean.
 - **Structured:** Date/Time, Array, Range, UUID.
 - **Document:** JSON/JSONB, XML, Key-value (Hstore).
 - **Geometry:** Point, Line, Circle, Polygon.
 - **Custom Types:** Developers can create their own data types.
 - **Data Integrity:** Supports UNIQUE, NOT NULL, Primary Keys, and Foreign Keys.
 4. **Performance and Concurrency:**
 - **Sophisticated Query Planner:** Optimizes queries and supports index-only scans.
 - **Multi-Version Concurrency Control (MVCC):** Ensures consistent reads and writes.
 - **Parallelization:** Read queries and B-tree index building benefit from parallel execution.
 5. **Community and Adoption:**
 - PostgreSQL boasts an **active community** and extensive documentation.
 - It's used across various domains, including news websites, e-commerce platforms, and social networks.

4.4 PYTHON:

Certainly! **Python** is a high-level, interpreted, and general-purpose dynamic programming language that focuses on code readability. Here are some key points about Python:

1. **Readability and Simplicity:**
 - Python's design philosophy emphasizes code readability.
 - Its syntax resembles the English language, making it intuitive for developers.
 - Fewer steps are needed compared to languages like Java and C.
2. **Features:**
 - **Interpreted:** Python runs on an interpreter system, allowing code execution as soon as it's written. This enables rapid prototyping.
 - **Versatile:** Python supports multiple programming paradigms, including procedural, object-oriented and functional programming.
 - **Comprehensive Standard Library:** Often described as a "batteries included" language, Python provides extensive built-in modules for various tasks.
3. **Use Cases:**
 - **Web Development:** Python is used for server-side web applications.
 - **Software Development:** It's suitable for creating production-ready software.
 - **Mathematics and Data Science:** Python handles big data and complex mathematics.
 - **System Scripting:** Python can read and modify files, connect to databases, and automate tasks.
4. **Popularity:**
 - Python ranks among the most popular and fastest-growing languages globally.
 - Its simplicity, versatility, and active community contribute to its widespread adoption.

4.5 POSTGRES DATABASE:

PostgreSQL is a powerful, open-source object-relational database system that combines the SQL language with features designed to safely store and scale complex data workloads. Here are some key points about PostgreSQL:

1. **Origins and Development:**

- PostgreSQL traces its roots back to the **POSTGRES project** at the **University of California, Berkeley** in **1986**.
- Over **35 years** of active development have shaped PostgreSQL into a reliable, feature-rich database system.

2. **Features:**

- **SQL and Beyond:** PostgreSQL supports standard SQL features along with extensions like foreign keys, subqueries, and triggers.
- **Extensibility:** Developers can define custom data types, functions, and even write code in different programming languages without recompiling the database.
- **Data Integrity:** Features include primary keys, foreign keys, and exclusion constraints.
- **Concurrency and Performance:** Sophisticated query planning, parallelization, and multi-version concurrency control (MVCC) enhance performance.

3. **Data Types:**

- PostgreSQL offers a wide range of data types:
 - Primitives: Integer, Numeric, String, Boolean.
 - Structured: Date/Time, Array, Range, UUID.
 - Document: JSON/JSONB, XML, Key-value (Hstore).
 - Geometry: Point, Line, Circle, Polygon.
 - Custom Types: Developers can create their own data types.

4. **Community and Adoption:**

- PostgreSQL has a dedicated open-source community.
- It runs on all major operating systems and has powerful add-ons like **PostGIS** for geospatial data.

5. CODING:

SIGNUP FORM:

```
{% load static %}
<link rel="stylesheet" href={%static 'style.css'%}>
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Signup Form</title>
  <style>
    body {
      font-family: Arial, sans-serif;
      {% load static %}
      
      background-image: url('rr.png');
      background-repeat: no-repeat;
      background-attachment: fixed;
      background-size: cover;
      background-position: center;
      height: 100vh;
      margin: 0;
      background-color: #E5E5E5;
      display: flex;
      justify-content: center;
      align-items: center;
      height: 100vh;
      margin: 0;
    }
    .signup-container {
      background-color: #F0F0F0; /* Set background color to transparent */
      filter: contrast(120%) saturate(150%);
      padding: 40px;
      border: 1px solid #ddd; /* Add a border */
      border-radius: 30px; /* Make corners round */
      box-shadow: 0 5px 10px rgba(0,0,0,0.1);
      width: 400px;
    }
    .signup-container h2 {
      color: #e74c3c;
      margin-bottom: 10px;
    }
    .signup-container p {
      margin-bottom: 20px;
    }
    .signup-container input {
      margin: 0 auto; /* Add horizontal margin to center the input fields */
      display: block; /* Make the input fields block-level elements */
      width: 100%; /* Set the width to 80% to create some space around the input
fields */
      margin-bottom: 10px;
      border: 5px solid #ddd;
      border-radius: 30px;
      padding: 10px;
    }
```

```

    }
    .signup-container button {
        background-color: #e74c3c;
        color: white;
        border: none;
        border-radius: 30px;
        padding: 10px;
        width: 100%;
        cursor: pointer;
    }
    .signup-container a {
        display: block;
        text-align: center;
        margin-top: 10px;
        color: #e74c3c;
        text-decoration: none;
    }
</style>
</head>
<body>
    <div class="signup-container">
        <h2>Signup Form</h2>
        <p>Create a new account.</p>
        <form method="POST" action="signupForm">
            {% csrf_token %}
            <label for="username">Username:</label>
            <input type="text" id="username" name="username"><br><br>
            <label for="email">email:</label>
            <input type="email" id="email" name="email"><br><br>
            <label for="password">Password:</label>
            <input type="password" id="password" name="password"><br><br>
            <input type="submit" value="Sign up">
        </form>
        <a href="#">Already have an account? Login here"</a>

    </div>

    <script>
        document.getElementById('signupForm').addEventListener('submit', function(event){
            event.preventDefault();
            // Add your JavaScript code here to handle the signup event
            alert('Signup button clicked!');
        });
    </script>
</body>
</html>

```

SIGNIN PAGE:

```
{% load static %}
<link rel="stylesheet" href="{% static 'style.css' %}">
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Login Form</title>
  <style>
    body {
      font-family: Arial, sans-serif;
      {% load static %}
      
      background-image:url('rr.png');
      background-size: cover;
      background-position: center;
      height:100vh;
      margin:0;
      background-color: #E5E5E5;
      display: flex;
      justify-content: center;
      align-items: center;
      height: 100vh;
      margin: 0;
    }
    .login-container {
      background-color: #F0F0F0;
      filter: contrast(120%) saturate(150%);
      padding: 40px;
      border: 1px solid #ddd; /* Add a border */
      border-radius: 30px;
      box-shadow: 0 5px 10px rgba(0,0,0,0.1);
      width: 400px;
    }
    .login-container h2 {
      background-color: #F0F0F0;
      margin-bottom: 10px;
      font-weight: bold;
    }
    .login-container p {
      margin-bottom: 20px;
      font-size: 14px;
      color: #666;
    }
    .login-container input {
      margin-bottom: 10px;
      border: 5px solid #ddd;
      border-radius: 30px;
```

```

padding: 10px;
width: 100%;
font-size: 16px;
}
.login-container button[type="submit"] {
background-color: #e74c3c;
color: #fff;
border: none;
border-radius: 30px;
padding: 10px;
width: 100%;
font-size: 16px;
cursor: pointer;
}
.login-container button[type="submit"]:hover {
background-color: #e67e73;
}
.login-container a {
display: block;
text-align: center;
margin-top: 10px;
color: #e74c3c;
text-decoration: none;
font-size: 14px;
}
.login-container a:hover {
color: #e67e73;
}
</style>
</head>
<body>
<div class="login-container">
<h2>Login Form</h2>
<p>Enter your login credentials.</p>
<form method="POST" action="loginForm">
{ %csrf_token% }
<label for="username">username</label>
<input type="text" placeholder="username" id="username" name="username" required>
<label for="password">password</label>
<input type="password" placeholder="Password" id="password" name="password" required>
<input type="submit" value="LOGIN">
</form>
<a href="#">Don't have an account? Sign up here</a>
</div>

<script>
document.getElementById('loginForm').addEventListener('submit', function(event){
event.preventDefault();
// Add your JavaScript code here to handle the login event
alert('Login button clicked!');
});

```

```
</script>
</body>
</html>
```

HOME PAGE:

```
{% load static %}
<link rel="stylesheet" href="{% static 'style.css' %}">
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Homepage</title>
  <style>
    body {
      font-family: Arial, sans-serif;
      background-color: #f0f0f0;
      background-image: url('{% static 'rr.png' %}');
      background-repeat: no-repeat;
      background-attachment: fixed;
      background-size: cover;
      background-position: center;
      height: 100vh;
      margin: 0;
    }
    .homepage {
      max-width: 800px;
      margin: 40px auto;
      padding: 20px;
      background-color: #fff;
      border: 1px solid #ddd;
      border-radius: 30px;
      box-shadow: 0 0 10px rgba(0,0,0,0.1);
    }
    .homepage h2 {
      margin-top: 0;
      color: #333;
    }
    .highlighted-button {
      background-color: #337ab7;
      color: #fff;
      padding: 10px 20px;
      border: none;
      border-radius: 5px;
      cursor: pointer;
    }
    .highlighted-button:hover {
```

```

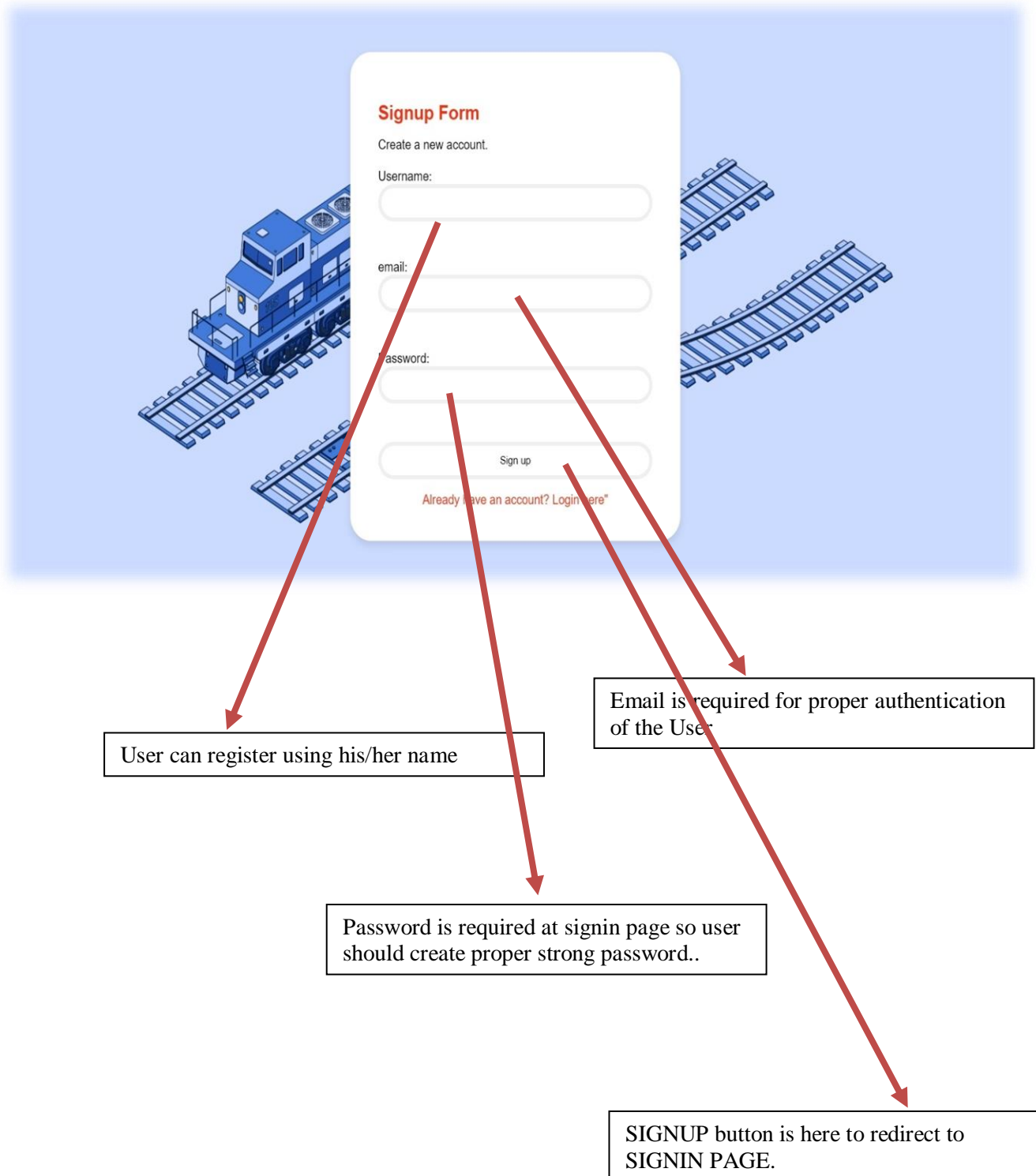
        background-color: #23527c;
    }
    .floating-button {
        position: fixed;
        bottom: 20px;
        right: 20px;
        z-index: 1;
    }
</style>
</head>
<body>
    <div class="homepage">
        <h2>Welcome to your account, {{ request.user.username }}!</h2>
        <button class="highlighted-button" onclick="editProfile()">Edit Profile</button>
        <button class="highlighted-button" onclick="changePassword()">Change Password</button>
        <button class="highlighted-button floating-button" onclick="location.href='{ % url 'logout'
% }'">Logout</button>
    </div>

    <script>
        function editProfile() {
            // add code here to edit user profile
            alert("Edit Profile feature coming soon!");
        }
        function changePassword() {
            // add code here to change user password
            alert("Change Password feature coming soon!");
        }
        function logout() {
            // add code here to logout the user
            alert("You have been logged out!");
        }
    </script>
</body>
</html>

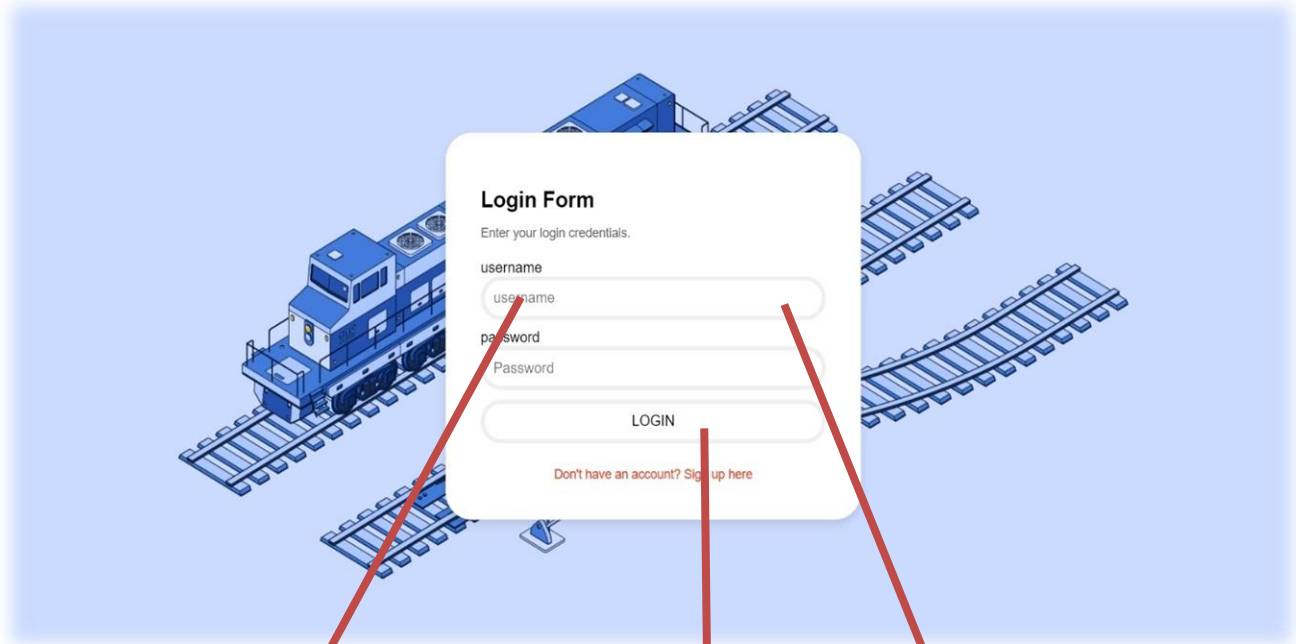
```

5. SCREENSHOTS

Signup page:



Signin page:

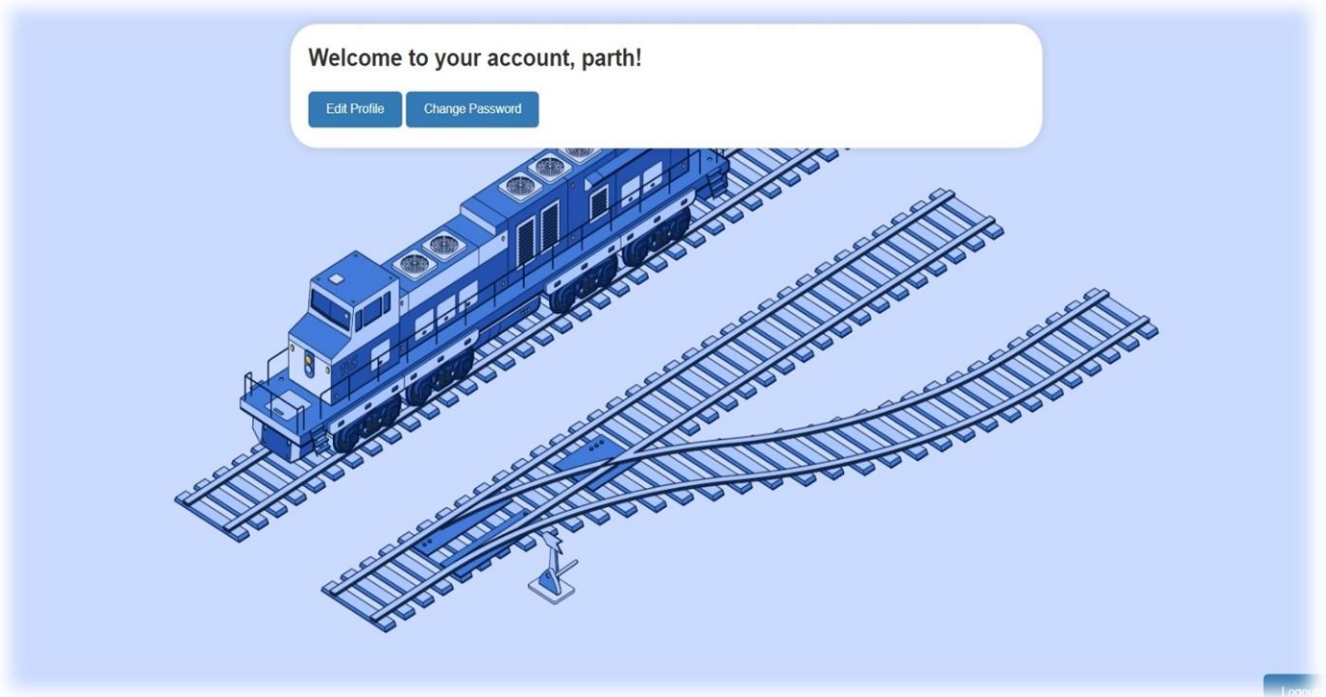


Username should be valid and should be enter correctly because it is a case sensitive.

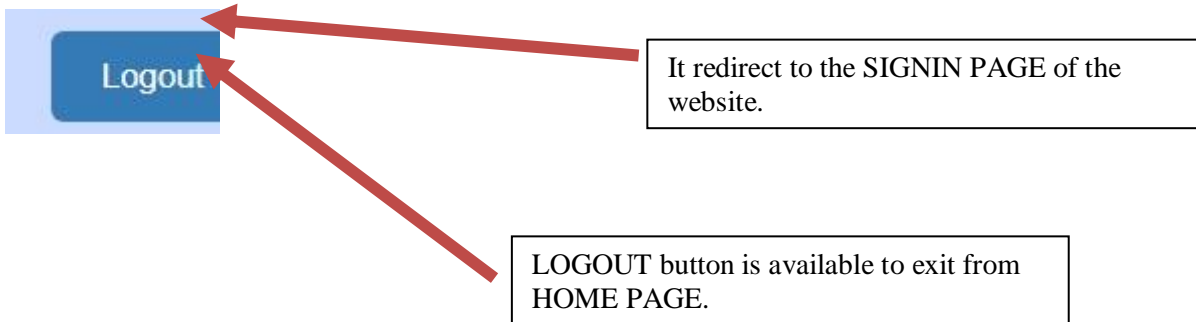
Password should be fill for moving to HOME PAGE.

LOGIN button here, it redirect to the HOME PAGE..

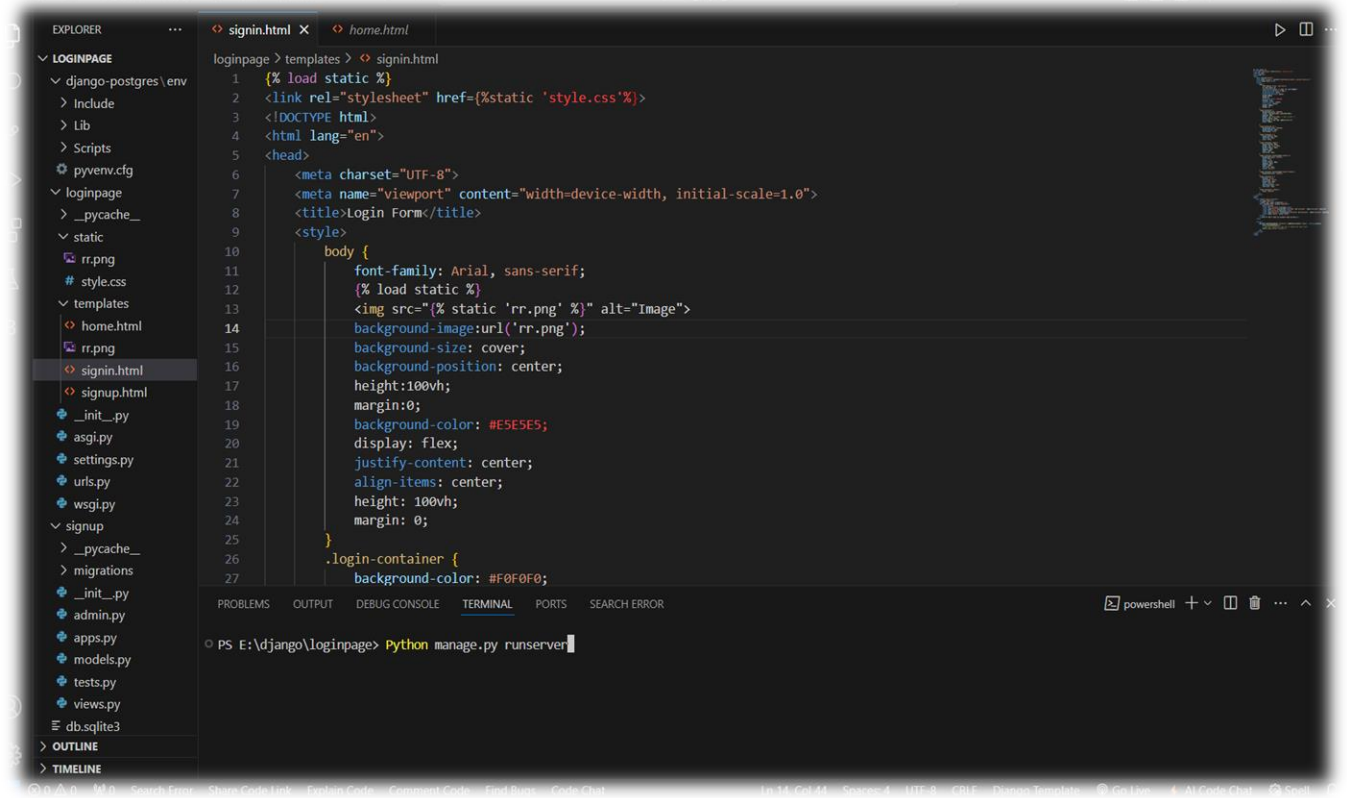
Home Page:



LOGOUT BUTTON IS ALSO AVAILABLE THAT IS LINKED TO THE AIGNIN PAGE:



Starting server:



```
loginpage > templates > sign_in.html
1  {% load static %}
2  <link rel="stylesheet" href="{% static 'style.css' %}">
3  <!DOCTYPE html>
4  <html lang="en">
5  <head>
6      <meta charset="UTF-8">
7      <meta name="viewport" content="width=device-width, initial-scale=1.0">
8      <title>Login Form</title>
9      <style>
10         body {
11             font-family: Arial, sans-serif;
12             {% load static %}
13             
14             background-image: url('rr.png');
15             background-size: cover;
16             background-position: center;
17             height: 100vh;
18             margin: 0;
19             background-color: #E5E5E5;
20             display: flex;
21             justify-content: center;
22             align-items: center;
23             height: 100vh;
24             margin: 0;
25         }
26         .login-container {
27             background-color: #F0F0F0;
```

```
PS E:\django\loginpage> Python manage.py runserver
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS SEARCH ERROR

```
PS E:\django\loginpage> Python manage.py runserver
```

This command should be entered in the
TERMINAL to start the server.

```
System check identified 1 issue (0 silenced).
July 01, 2024 - 11:44:52
Django version 5.0.6, using settings 'loginpage.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```

This link should be click using
CTRL+CLICK to redirect to the signup
page..

Signup Form

Create a new account.

Username:

email:

Password:

[Already have an account? Login here*](#)