A PROJECT REPORT

On

"Study and Partial Development of e-Permit System"

Submitted To
IIS DEPARTMENT AND TRAINING AND DEVELOPMENT

In the partial fulfillment for the award of the degree of
"B.TECH COMPUTER SCIENCE & ENGINEERING"
SESSION 2021-2025

NRL, Golaghat

Submitted By:

Rishika Hazarika (Roll No: 210710007037)
Shreyoshi Ghosh (Roll No: 210710007046)
Shruti Sarma (Roll No: 210710007047)

B.Tech CSE 7th Semester
Jorhat Engineering College, Jorhat, Assam- India

Guided by:

Usha Kakati, Chief Manager, IIS Dept., NRL, Golaghat
Mridul Das, Software Developer, IIS Dept., NRL, Golaghat

JORHAT ENGINEERING COLLEGE,
GARMUR, JORHAT - 785001

Acknowledgement

We extend our deepest appreciation to our industry mentor, Usha Kakati, Chief Manager (IIS, NRL) for her invaluable guidance, constant encouragement and insightful suggestions. Her expertise and commitment have been instrumental in shaping the direction and quality of this project.

We would like to thank Mrs. Gayotri Sarma, Senior Manager (T&D) and Mr. Sangam Panchanan, Assistant Manager (T&D) for giving us the opportunity to learn during this internship in NRL.

We would also like to thank all the officers and staff for their valuable inputs, which were instrumental in completing the study and development work successfully. Special thanks to Saranga Pani Nath, Officer (IIS) and Mridul Das, Software Developer for their immense help in enhancing our understanding of the technologies we learned during this internship at NRL.

Lastly, we would like to express our heartfelt gratitude to our family and co-internees for their constant encouragement, love, and understanding. Their unwavering support has been a driving force behind our pursuit of knowledge and personal growth.

Rishika Hazarika

Shreyoshi Ghosh

Shruti Sarma

Certificate

To whom it may concern

This is to certify that **Rishika Hazarika**, a student of Jorhat Engineering College, has undergone summer internship programme at Numaligarh Refinery Limited from 18th of June, 2024 to 17th of July, 2024.

During the tenure of the project, she was found to be sincere, hardworking and dedicated. I wish her all the success in life.

Mr. Sangam Panchanan
Assistant Manager (T&D)
Numaligarh Refinery Limited

Certificate

To whom it may concern

This is to certify that **Shruti Sarma**, a student of Jorhat Engineering College, has undergone summer internship programme at Numaligarh Refinery Limited from 18th of June, 2024 to 17th of July, 2024.

During the tenure of the project, she was found to be sincere, hardworking and dedicated. I wish her all the success in life.

Mr. Sangam Panchanan
Assistant Manager (T&D)
Numaligarh Refinery Limited

Certificate

To whom it may concern

This is to certify that **Shreyoshi Ghosh**, a student of Jorhat Engineering College, has undergone summer internship programme at Numaligarh Refinery Limited from 18th of June, 2024 to 17th of July, 2024.

During the tenure of the project, she was found to be sincere, hardworking and dedicated. I wish her all the success in life.

Mr. Sangam Panchanan
Assistant Manager (T&D)
Numaligarh Refinery Limited

Executive Summary

This project was conducted at Numaligarh Refinery Limited (NRL), a distinguished Mini Ratna Public Sector Undertaking (PSU) of India. NRL operates under the aegis of its parent company, OIL India Limited (OIL). The focus of this summer internship project is to study the e-Permit system in NRL and develop a web-based Permit Issue and Permit Receive Module of the aforementioned system.

During this study, we delved into the intricate processes involved in the e-Permit system implementation at NRL. The project encompasses the design and development of two of the e-Permit system's web-based modules for issue and receipt of permits, which are in compliance with safety regulations and operational standards. We studied the current workflows and investigated the various technological frameworks utilized in the development of the e-Permit system. We also explored the web technologies and architecture employed, particularly the Model-View-Controller (MVC) framework. The study further examined the work safety protocols in place under the OISD guidelines and analyzed the system's potential for further development to meet NRL's evolving needs

Contents

1 Abstract **8**

2 Company Profile **9**

2.1 Numaligarh Refinery Limited 9

2.2 Information Technology 9

3 Introduction to the Project **10**

3.1 Overview of the Project: 10

3.2 Background of the Project: 10

3.3 Objective of the Project: 11

4 e-permit System **12**

4.1 Overview of e-permit System: 12

4.2 Types of Permit: 12

4.3 Work Permit Fields: 13

4.4 System Flow of e-permit system 15

4.5 System Modules 16

5 Project Flow Overview: **19**

6 Brief Overview of Project Components **20**

7 Comprehensive Description of Project components: **21**

7.1 Front End: 21

7.1.1 Layout Design 21

7.1.2 Login Page: 29

7.1.3 Permit Issue Page: 37

7.1.4 Permit Receive Page: 41
7.1.5 All Permit Issued Information Page (Show All Permit Page) 47
7.2 Database Design 47
7.2.1 Microsoft SQL Server Management Studio 48
7.2.2 Relational Database Management System (RDBMS) 48
7.2.3 Stored Procedures 49
7.3 Back End 51
7.3.1 C# programming language 51
7.3.2 .NET Framework 52
7.3.3 Model-View-Controller (MVC) Architecture 53

8 Conclusion and Future Scope **58**

1. Abstract

This report "e-permit system" presents a comprehensive analysis of the e-permit system implemented at Numaligarh Refinery Limited (NRL). The e-permit system is designed to streamline and enhance the efficiency of the work permit process within the refinery. By leveraging digital tools and automation, the system aims to include safety, reduce paperwork and ensure compliance with regulatory protocols. The report provides a brief overview of all the features of the e-permit system, but our project is limited to the implementation of only two functionalities of the system. These two implemented modules, namely Permit Issue Module and Permit Receiver Module are described in detail, including their design and development. This system has been developed using technologies HTML, CSS, JavaScript, Bootstrap5, jQuery, C#, ASP.NET Framework and MVC Architecture. The current system can demonstrate significant improvements in reducing permit issuance time, minimizing human errors, and enhancing safety standards. There is considerable scope for future development to make the system fully operational for refinery use. This report concludes with recommendations for further enhancements and additional functionalities to achieve full operational capability by hosting on a server.

Keywords: e-permit system, Permit Issue Module, Permit Receiver Module, HTML, CSS, Javascript, Bootstrap5, jQuery, C#, ASP.NET framework, MVC Architecture

2. Company Profile

2.1. Numaligarh Refinery Limited

Numaligarh Refinery Limited (NRL), a division of Oil India Limited, is a Public Sector Undertaking (PSU) in Oil and Gas Sector. The refinery was set up at Numaligarh in the district of Golaghat, Assam on 22nd April 1993 in accordance with the provisions made in the historic Assam Accord signed on 15th August, 1985. It has been conceived as a vehicle for speedy industrial and economic development of the region.

NRL is a Schedule- A, Category-I Mini Ratna Central Public Sector Enterprises (CPSE) under the Ministry of Petroleum and Natural Gas, Govt. of India. The 3 MMTPA Refinery was dedicated to the nation by former Prime Minister of India Shri Atal Bihari Vajpayee on 9th July, 1999. With its concern, commitment and contribution to socio-economic development of the state of Assam and North East India, it is considered as one of the glowing manifestations of successful business enterprise in the region, with its footprints across the globe.

NRL is embarking on a major integrated Refinery Expansion Project to treble its capacity from 3 MMTPA to 9 MMTPA at an estimated investment of more than Rs. 28,000 Crore, one of the highest in the North East region. Aligning itself with the Govt. of India's target of achieving net zero by the year 2070, NRL has carved a roadmap to attain its net zero goals by the year 2038.

2.2. Information Technology

The company has made significant progress in the use of Information Technology for managing its resources. Since its inception, the Company has been guided by the principle of making use of Information Technology as a strategic tool of management and to increase transparency in corporate governance. To integrate all the business functions of the Company, the Company implemented an Enterprise Resource Planning (ERP) solution of M/s Ramco Systems which was further upgraded to cater new challenges emerging due to new business processes.
Furthermore, the company had implemented different SAP ERP suite including SAP R/3 4.7 ERP (August 2005), SAP ECC 6.0 ERP (March 2011) and with the digital transformation, the SAP version was migrated to S/4 HANA (February 2022). Along with it, it has successfully adopted IBM BAW (Business Automation Workflow) to automate digital workflows across the business functions of finance, HR, procurement, operations, IT and e-Office Approvals.

3. Introduction to the Project

3.1. Overview of the Project:

The e-permit system is designed to ensure safe working practices at Numaligarh Refinery Limited by following the guidelines of OISD-STD-105. This proposed software system should be able to efficiently process, store and manage important information and data at Numaligarh Refinery Limited.

This management system handles a core part of the work execution process which starts form requesting the e-permit to validating it with all the required factors(including availability of active e-permits, job notification from SAP system against equipment, validate equipment and contractor work order, contract workmen gate pass, e-permit user authentication etc).

The e-permit system is an extensive software with different functionalities. Our project focuses on developing the website component of the e-permit system, which aims to enhance the accessibility and usability of this tool.

3.2. Background of the Project:

The oil and gas sector is among the eight core industries in India and plays a major role in influencing the decision making for all the other important sections of the economy. The oil and gas industries, particularly Numaligarh Refinery Limited, involves the refining, transportation and marketing of crude oil, petroleum, natural gas, etc.

The Oil Industry of India which is over 100 years old, has adopted diverse practices due to collaborations with various foreign companies and governments. National standardisation in design philosophies and operating and maintenance practices was hardly in existence. This coupled with feedback from significant accidents that occurred in India in the recent past and abroad, emphasised the need for the industry to review the existing state-of-art practices.

With respect to this, the Ministry of Petroleum and Natural Gas in 1986, supported by the Oil Industry Safety Directorate (OISD), established a safety council. This council formulated and implemented a series of self-regulatory measures aimed at removing obsolescence, standardising and upgrading the existing safety protocols. The first document on "Work Permit System" was published by OISD in February 1988 and was amended in July 1998.

In industries and refineries like Numaligarh Refinery Limited (NRL), hydrocarbon processing and handling installation presents unique risks, necessitating stringent safety measures. To provide safe working conditions and ensure the secure execution of tasks, a comprehensive work permit system must be implemented. The primary objective of this system is to guarantee that work is conducted in the safest manner possible, preventing injuries to personnel, protect property from damage, avoiding fires and mitigating other potential hazards.

3.3. Objective of the Project:

Numaligarh Refinery Limited (NRL) is actively developing an advanced e-permit system based on the guidelines of the OISD work permit system. We have successfully implemented the Permit Issue and Permit Receive modules, which are integral to the overall functionality of the e-permit system. This development would mark a significant milestone to enhance operational efficiency and safety protocols within the refinery. By prioritizing the implementation of these two essential components, a robust foundation for the entire e-Permit system can be laid.

The Permit Issue module ensures a streamlined and industry-compliant process for issuing permits, while the Permit Receive module facilitates the accurate, prompt and timely receipt of permits, maintaining all essential checks and balances. This system simplifies the permit management process and also improves accountability and traceability, thus providing a more efficient and reliable framework for permit handling.

In addition to these core modules, the system can eventually incorporate several other features designed to optimize workflows and support comprehensive permit management. However, the integration of additional features and modules is beyond the scope of the current project.

The successful deployment of the Permit Issue and Permit Receive modules would act as a foundation for further development and an advanced e-permit system, contributing significantly to work safety and operational excellence.

4. e-permit System

4.1. Overview of e-permit System:

The e-permit system, designed under OISD-STD-105 guidelines to ensure safe working practices, functions as a management tool centered around both developers and users of the system. It manages the critical elements of the work execution process, beginning with the request and validation of e-permits, which includes checking for other active permits and relevant job notifications from the SAP system concerning equipment. Moreover, the system also verifies equipment and contractor work orders, manages contract workmen gate passes and ensures e-permit user authentication through an online training and evaluation process. By automating all the necessary steps, the e-permit system ensures that all safety protocols are followed, making the work environment safer and efficient.

The software system is able to process, store and manage various important information, data related to Numaligarh Refinery Limited. The information may be entered either manually or fetched from various existing software systems of NRL, which are imperative for the running business of NRL.

4.2. Types of Permit:

Work in oil and gas industries is classified into two types: cold work and hot work.

- **Cold Work:** An activity which does not produce sufficient heat to ignite a flammable air-hydrocarbon mixture or a flammable substance.
- **Hot Work:** An activity that can produce a spark or flame or other source of ignition having sufficient energy to cause ignition, where the potential for flammable vapours, gases, or dust exists.

Based on the nature of the work to be performed, the following two types of work permits are used:

- **Cold Work Permit**: Work falling under the category of cold work such as opening process machinery, blinding and deblinding, tightening of flanges, hot bolting, inspection, painting etc. shall be performed through cold work permit.
- **Hot Work Permit:** All hot work such as welding, grinding, gas cutting, burning, shot blasting, soldering, chipping, excavation, open fire, use of certain non-explosion proof equipment etc. shall be carried out through hot work permit. Entry and operation of petrol and diesel driven vehicles or equipment in hazardous areas also fall in the category of hot work, and shall be performed under the hot work permit

4.3. Work Permit Fields:

Based on OISD guidelines, the mandatory work permit fields and checklisted items are elaborated below:

1. **Exact Location (Area / Unit / Equipment no):** Exact location of the area or the unit in which the work is to be carried out shall be written against it.
2. **Description of work:** Precise description of the work to be performed shall be written.
3. **Equipment / Area inspected:** Equipment or area where work is to be conducted, should be inspected to ensure that it is safe to carry out the work and assess other safety requirements.
4. **Surrounding area checked / cleaned:** Unsafe conditions for performance of work may arise from surrounding area. It should be cleaned-up to remove flammable material such as oil, rags, grass etc.
5. **Sewers, Manholes, Closed Blow Down (CBD) etc. and Hot Surfaces covered:** Flammable gases may be released from nearby sewers. Hot un-insulated surfaces (equipment / pipelines) may provide a source of ignition and therefore, these are to be properly covered to prevent fires.
6. **Hazard from other routine / non-routine operations considered and persons alerted:** Other activities (routine / non-routine) being carried out near-by, which can create conditions unsafe for performance of the permit work, should be taken into consideration and the concerned persons should be alerted accordingly.
7. **Equipment electrically isolated and tagged:** Before issuing a permit, it shall be ensured that electrical isolation has been done, switches are locked-out and cautionary tags duly signed with date and time are attached.
8. **Running water hose / Portable extinguisher provided / Fire water system available:** Running water hose and portable fire extinguisher are required to flush / dilute in case of release of any hazardous chemical or to quench sparks and to put out small fires immediately. In order to meet any contingency, it should be ensured that the fire water system including firewater pumps, storage, network etc. is checked and kept ready for immediate use.
9. **Equipment blinded / disconnected / closed / isolated / wedge opened:** Equipment for which the work permit is being issued, should be isolated from the rest of the plant in order to ensure that there is no change in the work environment with respect to presence of toxic / flammable gases, liquids, hazardous chemicals etc. during the course of the work. Blinding is one of the most effective ways of isolation. Blinds should be installed as close to the equipment as possible. If lines cannot be blinded, these should be disconnected and the open ends should be made safe by installing pipe caps / plugs, blind flanges etc.

10. **Equipment properly drained / depressurised:** Equipment under pressure should be depressurised after isolation. This should be followed by draining / purging / water flushing etc. as the case may be.

11. **Equipment properly steamed / purged:** Purging of equipment (tanks, vessels, pipelines etc.) is done to make them free of flammable hydrocarbon and toxic gases. Steam / Inert Gas is used for gas freeing of vessels and pipes in refineries and other locations.

12. **Equipment water flushed:** Water flushing is an effective means of cooling, cleaning and even gas freeing of equipment. It is also employed to remove traces of acids / chemicals.

13. **Gas / Oxygen deficiency test done:** Gas test includes measurement of:

    (a) Hydrocarbons by Explosive meter

    (b) Oxygen Deficiency by Oxygen Meter

    (c) Toxic gases like Hydrogen Sulphide, Carbon Monoxide, Nickel Carbonyl, and Chlorine etc. by techniques like Indicator Tube method, Lead Acetate Paper etc.

14. **Shield against sparks provided:** Hot works like welding, grinding etc generates sparks which can provide source of ignition to the surroundings. In order to protect operating area from the hazards of sparks generated, shields are to be provided to contain the sparks generated. The shield material should be non-flammable and should be kept wet with water.

15. **Proper ventilation and lighting provided:** Where natural ventilation is not available, fans / air eductors are provided. These are also required for speedy dispersal of fumes generated by welding job. Only approved reduced voltage extension lights (not exceeding 24 volts) are to be allowed for work inside vessels from consideration of personal safety.

16. **Proper means of exit provided:** Proper means of exit is required in case of emergencies developed on account of the work or otherwise. Availability of an alternate route of escape should be considered.

17. **Area cordoned off and caution boards provided:** To prevent any unwarranted entry in the work area and also to caution other personnel taking actions which may endanger people working on the permit job, precautionary tags / boards are to be provided to display.

18. **Portable equipment / Hose nozzles properly grounded:** As a precaution against static electricity generation, portable equipment / hose nozzles e.g. nozzle of a shot blasting gun, are to be grounded.

19. **Standby person provided for entry to confined space:** Whenever a person is entering in a confined space, minimum two designated persons shall be kept at the

manhole or entry point. The designated person shall be in constant communication with the persons inside the confined space.

20. **Standby personnel provided for fire watch from Process / Maintenance / Contractor / Fire Department:** Depending on the criticality of the job, work permit issuer shall decide the type of standby to be provided i.e. from which department, of which level, how many and also additional fire fighting support facilities etc.

21. **Iron Sulphide removed / kept wet:** Pyrophoric substances may be present in operating area / equipment handling hydrocarbon. Iron Sulphide scale is the most common pyrophoric substance encountered. These should be either removed to safe locations or kept wet all the time to prevent their auto-ignition.

22. **Clearance obtained for excavation / Road cutting / dyke cutting etc:** For any excavation work which may affect underground sewers / telephone lines / cables / pipelines etc., clearance shall be obtained from all the concerned sections. Road cutting can hamper the movement of the fire vehicles; initial clearance should be obtained from Fire Department and final approval from the higher designated authorities. A higher level authority should be designated for authorizing dyke cutting.

23. **Checked spark arrestor on mobile equipment:** No vehicle / engine without spark arrestor shall be permitted in operating areas.

24. **Checked for oil / gas trapped behind lining in equipment:** Before undertaking hot jobs, a check should be done for oil / gas trapped behind lining in the equipment.

4.4. System Flow of e-permit system

The e-permit system operates as designed, beginning with the request for a permit. Upon request, the system checks for existing permits in the selected area and verifies checklist items before processing the request. Once the request is submitted, it is forwarded to the permit issuer for that area. The issuer then verifies the request and proceeds to issue the permit. If approval is required, the system follows the necessary approval path; otherwise, the permit is directly visible to the recipient.

Once the recipient receives the permit, they can commence the job. Upon job completion, they can request to close the permit. If the job extends beyond the permitted time, a renewal request can be submitted. For permits older than 7 days, a new permit request is made instead of a renewal, and the current permit moves to the closure request list. The system then checks for related permits to be closed and notifies the issuer accordingly.

The e-permit system also integrates additional safety features such as Lockout/Tagout (LOTO) and Job Safety Analysis (JSA). The LOTO feature is linked with electrical permits, and the system performs background checks on the relevant instrument/equipment when issuing or closing electrical permits. Alerts are raised as needed. The JSA check is incorporated into the system to ensure safety before requesting or issuing any permit.

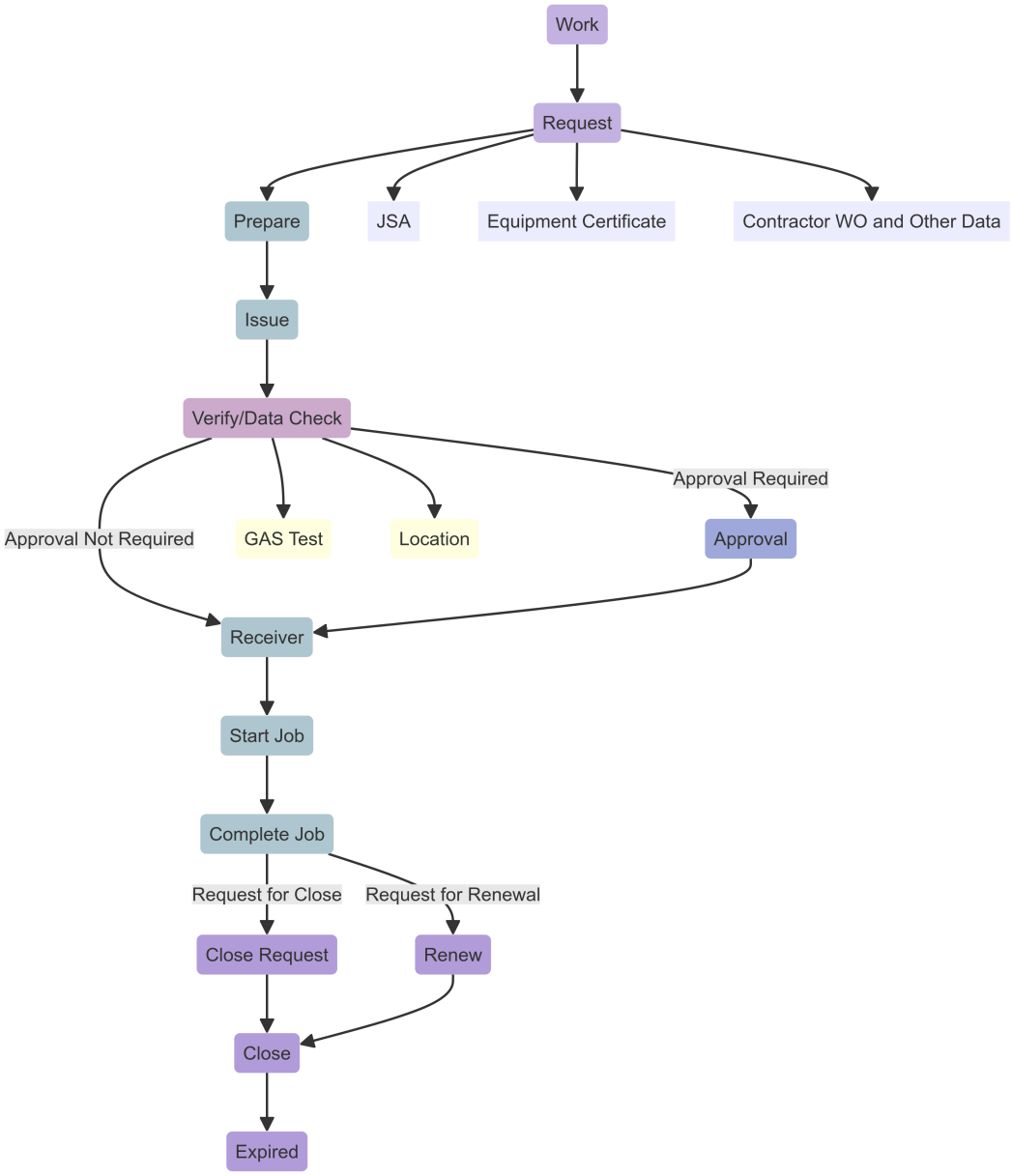

Figure 1: System Flow Diagram of the e-permit System

4.5. System Modules

The e-permit System is divided into following main modules:

1. Permit Request Module
2. Permit Issue Module
3. Permit Approval Module
4. Permit Receiver Module

5. Permit Close Request Module
6. Permit Close Module

For this project, our focus is on implementing the **Permit Issue** and **Permit Receiver** Modules from the aforementioned list. The detailed description of these specific modules are provided below:

1. **Permit Issue Module:** The Permit Issue Module operates based on area and time, displaying requests for a specific area and the current shift time to the issuer assigned to that area. Before issuing the permit, the issuer verifies the request and updates the checklist, if necessary. Once the issuer is satisfied with the verification, they can issue the permit to the appropriate function as requested.

   The issuer has two main functions:

   (a) Issue Permit from Existing Request: If the request has already been made, the issuer verifies the request and updates the checklist as needed. Once all data and the checklist are satisfactory, the permit can be issued.

   (b) Issue Permit without Request: If there is no related request and the issuer needs to issue a new permit independently, they can do so. The same template as in the request permit will be displayed. Once the issuer completes all requirements of the request template, they can directly issue the permit for their area and the current shift only.

   During the permit issuance process, the system checks the work area location in the background. The issuer can issue the permit only if they are present at that specific location.

   It is important to note that, although the Permit Issue module typically issues permits based on requests, the module we have implemented will simply issue the permit. Our module bypasses the request process entirely, and upon login or signup, it directly opens to the issue permit page.

2. **Permit Receiver Module:** The Permit Receiver Module displays a list of issued permits based on the logged-in user. Within this module, the receiver can mark a permit as received, at which point the e-permit system considers it a valid active permit. If the permit is not marked as received within 7 days from the date of issue, it remains invalid and is subsequently removed from the active system flow, being stored in the invalid permit list.

   Upon the completion of the job associated with a permit, the receiver will mark the permit as closed, which then moves it to the Permit Close Request Module. Additionally, from the Permit Receiver Module, the receiver can request an extension (renewal) of the permit from the issuer. This extension request is limited to no more than 7 days or 7 renewals (whichever comes first) from the original permit issue date.

However, in the Permit Receiver Module that we have implemented, it will only display the list of all issued permits, which can then be confirmed as received or left as is. There is no functionality for an invalid permit list, the 7 day validation period, the Permit Close Request Module, or permit renewal, as we have not implemented these features.

5. Project Flow Overview:

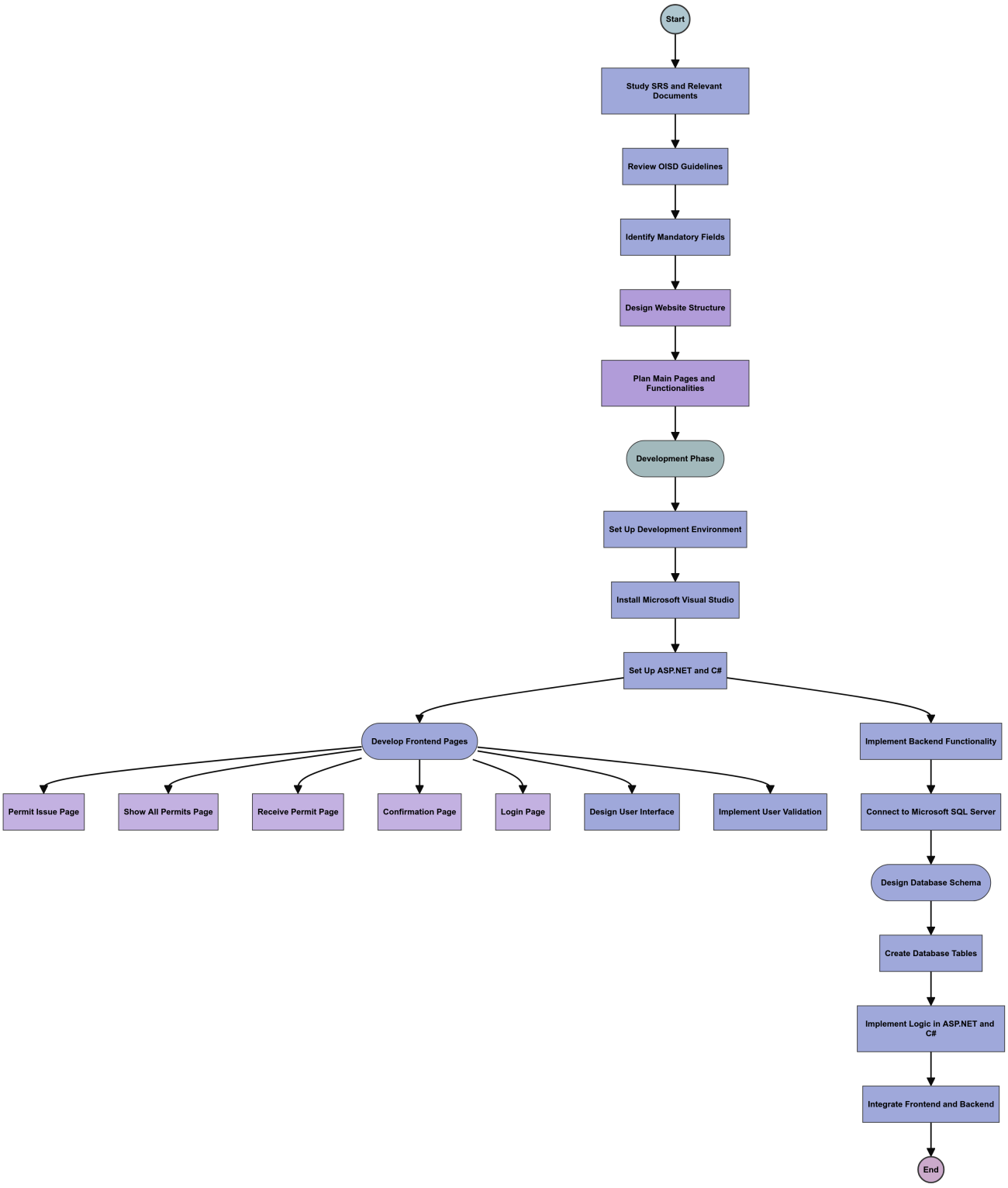

Figure 2: Project Flow Diagram

6. Brief Overview of Project Components

The project comprises mainly of three fundamental components, each serving a critical role in its architecture and functionality.

1. **Front End:** The front end of our project includes the user-interface, designed for a optimal and seamless experience. It encompasses the page layout, user interactions and presentation logic to enhance the usability and accessibility.
    - Technologies used: HTML, CSS, JavaScript, Bootstrap5, JQuery.
2. **Back End:** The back end includes server-side scripting and databases to facilitate communication between the front end and the database. It is responsible for processing requests from the front end, executing operations, retrieving and storing data in the database, and sending responses back to the user interface.
    - Technologies used: C#, ASP.NET Framework, Model-View-Controller (MVC) Architecture.
3. **Database Design:** Database Design is a set of procedures or collection of tasks involving various steps taken to implement the database. It involves structuring and organizing data to ensure efficient storage, retrieval and management. It provides a blueprint of how the data is going to be stored in the system.
    - Technologies used: Microsoft SQL Server Management Studio

Comprehensive details of the above three components are provided in the subsequent sections of this report.

7. Comprehensive Description of Project components:

7.1. Front End:

The frontend of our project is designed to create a more efficient workflow thereby ensuring usability and improved user experience. It includes the following key pages & components:

7.1.1 Layout Design

Layout design for a website is the arrangement and organization of visual elements on a web page, which ensures both functionality and a positive user experience. For our website, the Layout Design includes:

- **Header:** It contains a welcome message and a dropdown submenu that contains fields like edit profile, settings and log out.
- **Navigation:** This section facilitates the users to find their way around the site. The navigation of our website consists of a dashboard sidebar that contains all the required pages- Issue Page, Receive Page and Show All Permit Page.

Dashboard Sidebar:

HTML of the Sidebar:

```
    <!---------- Boxicons CSS ===== -->
     <link href='https://unpkg.com/boxicons@2.1.1/css/boxicons.min.
        css' rel='stylesheet'>

     <!--<title>Dashboard Sidebar Menu</title>-->
</head>

<body>
     <nav class="sidebar close">
          <header>
               <div class="image-text">
                    <span class="image">
                         <img src="nrl-og-logo.png" alt="">
                    </span>

                    <div class="text logo-text">
                         <span class="name">Fire and Safety <br>
                            Department</span>
                    </div>
               </div>

```

```
20              <i class='bx bx-chevron-right toggle'></i>
21          </header>
22
23          <div class="menu-bar">
24              <div class="menu">
25
26                  <ul class="menu-links">
27                      <li class="nav-link">
28                          <a href="#">
29                              <i class='bx bx-notepad icon'></i>
30                              <span class="text nav-text">Issue Permit
                                    </span>
31                          </a>
32                      </li>
33
34                      <li class="nav-link">
35                          <a href="#">
36                              <i class='bx bxs-left-arrow-square icon
                                    '></i>
37                              <span class="text nav-text">Show All
                                    Permit</span>
38                          </a>
39                      </li>
40
41                      <li class="nav-link">
42                          <a href="#">
43                              <i class='bx bxs-cog icon'></i>
44                              <span class="text nav-text">Receive</
                                    span>
45
46                      </li>
47
48                  </ul>
49              </div>
50
51              <div class="bottom-content">
52                  <li class="">
53                      <a href="#">
54                          <i class='bx bx-log-out icon'></i>
55                          <span class="text nav-text">Logout</span>
56                      </a>
57                  </li>
58
59                  <li class="mode">
60                      <div class="sun-moon">
61                          <i class='bx bx-moon icon moon'></i>
62                          <i class='bx bx-sun icon sun'></i>
```

```
                    </div>
                    <span class="mode-text text">Dark mode</span>

                    <div class="toggle-switch">
                        <span class="switch"></span>
                    </div>
                </li>

            </div>
        </div>

    </nav>
```

Explanation:

- For easy navigation of users, a dynamic dashboard is created. The dashboard consists of boxicons for interactive user experience. This is integrated with the help of an external CSS file from an icon library.
- The navigation is initiated through the `<nav>` tag that will help to define a set of navigation links to all the required pages.
- **Header:**
  1. This section consists of a `<div>` class with the image and a class text logo-text containing the Fire and Safety Department.
  2. An icon is placed for easy toggling of the sidebar to open/close state.
- **Menu:**
  1. The div class of 'menu-bar' contains 'menu-links' contains a list of items representing the navigation link. It contains links for the required items in the menu (eg: Issue Permit, Receive Permit, Show All Permit)
- **Bottom:**
  1. A list is created wherein a "Logout" element is included.
  2. Another list for the class 'mode' is created for toggling to dark mode. The 'div' class with icons is included for indicating dark and light modes.A 'div' with class 'toggle-switch', contains an element representing the switch.

CSS for the Sidebar:

```
/* ===== Sidebar ===== */
 .sidebar{
    position: fixed;
```

```
    top: 0;
    left: 0;
    height: 100%;
    width: 250px;
    padding: 10px 14px;
    background: var(--sidebar-color);
    transition: var(--tran-05);
    z-index: 100;
}
.sidebar.close{
    width: 88px;
}

/* ===== Reusable code - Here ===== */
.sidebar li{
    height: 50px;
    list-style: none;
    display: flex;
    align-items: center;
    margin-top: 10px;
}
.sidebar header .image,
.sidebar .icon{
    min-width: 60px;
    border-radius: 6px;
}
.sidebar .icon{
    min-width: 60px;
    border-radius: 6px;
    height: 100%;
    display: flex;
    align-items: center;
    justify-content: center;
    font-size: 20px;
}
.sidebar .text,
.sidebar .icon{
    color: var(--text-color);
    transition: var(--tran-03);
}
.sidebar .text{
    font-size: 17px;
    font-weight: 500;
    white-space: nowrap;
    opacity: 1;
}
.sidebar.close .text{
```

```
    opacity: 0;
}
/* ========================== */
.sidebar header{
    position: relative;
}
.sidebar header .image-text{
    display: flex;
    align-items: center;
}
.sidebar header .logo-text{
    display: flex;
    flex-direction: column;
}
header .image-text .name {
    margin-top: 2px;
    font-size: 18px;
    font-weight: 600;
}
header .image-text .profession{
    font-size: 16px;
    margin-top: -2px;
    display: block;
}
.sidebar header .image{
    display: flex;
    align-items: center;
    justify-content: center;
}
.sidebar header .image img{
    width: 40px;
    border-radius: 6px;
}
.sidebar header .toggle{
    position: absolute;
    top: 50%;
    right: -25px;
    transform: translateY(-50%) rotate(180deg);
    height: 25px;
    width: 25px;
    background-color: var(--primary-color);
    color: var(--sidebar-color);
    border-radius: 50%;
    display: flex;
    align-items: center;
    justify-content: center;
    font-size: 22px;
```

```
    cursor: pointer;
    transition: var(--tran-05);
}
body.dark .sidebar header .toggle{
    color: var(--text-color);
}
.sidebar.close .toggle{
    transform: translateY(-50%) rotate(0deg);
}
.sidebar .menu{
    margin-top: 40px;
}
.sidebar li.search-box{
    border-radius: 6px;
    background-color: var(--primary-color-light);
    cursor: pointer;
    transition: var(--tran-05);
}
.sidebar li a{
    list-style: none;
    height: 100%;
    background-color: transparent;
    display: flex;
    align-items: center;
    height: 100%;
    width: 100%;
    border-radius: 6px;
    text-decoration: none;
    transition: var(--tran-03);
}
.sidebar li a:hover{
    background-color: var(--primary-color);
}
.sidebar li a:hover .icon,
.sidebar li a:hover .text{
    color: var(--sidebar-color);
}
body.dark .sidebar li a:hover .icon,
body.dark .sidebar li a:hover .text{
    color: var(--text-color);
}
.sidebar .menu-bar{
    height: calc(100% - 55px);
    display: flex;
    flex-direction: column;
    justify-content: space-between;
    overflow-y: scroll;
```

```
}
.menu-bar::-webkit-scrollbar{
    display: none;
}
.sidebar .menu-bar .mode{
    border-radius: 6px;
    background-color: var(--primary-color-light);
    position: relative;
    transition: var(--tran-05);
}
.menu-bar .mode .sun-moon{
    height: 50px;
    width: 60px;
}
.mode .sun-moon i{
    position: absolute;
}
.mode .sun-moon i.sun{
    opacity: 0;
}
body.dark .mode .sun-moon i.sun{
    opacity: 1;
}
body.dark .mode .sun-moon i.moon{
    opacity: 0;
}
.menu-bar .bottom-content .toggle-switch{
    position: absolute;
    right: 0;
    height: 100%;
    min-width: 60px;
    display: flex;
    align-items: center;
    justify-content: center;
    border-radius: 6px;
    cursor: pointer;
}
.toggle-switch .switch{
    position: relative;
    height: 22px;
    width: 40px;
    border-radius: 25px;
    background-color: var(--toggle-color);
    transition: var(--tran-05);
}
.switch::before{
    content: '';
```

```
    position: absolute;
    height: 15px;
    width: 15px;
    border-radius: 50%;
    top: 50%;
    left: 5px;
    transform: translateY(-50%);
    background-color: var(--sidebar-color);
    transition: var(--tran-04);
}

body.dark .switch::before{
    left: 20px;
}

.home{
    position: absolute;
    top: 0;
    top: 0;
    left: 250px;
    height: 100vh;
    width: calc(100% - 250px);
    background-color: var(--body-color);
    transition: var(--tran-05);
}
.home .text{
    font-size: 20px;
    font-weight: 500;
    color: var(--text-color);
    padding: 12px 60px;
}

.sidebar.close ~ .home{
    left: 78px;
    height: 100vh;
    width: calc(100% - 78px);
}
body.dark .home .text{
    color: var(--text-color);
}
```

Explanation:

- **Sidebar styles:** The underlying CSS explains the structure of the sidebar. It is fixed to the left side of the viewport with a width of 250px that changes when the 'close' class is added. A sidebar colour is set and a transition of `transition: var(--tran-05)`

is added to change the sidebar's width.

- Reusable Styles for Sidebar Items:
  1. The list items within the sidebar are displayed as flex containers with a centered alignment.
  2. An image and an icon is added inside the header within the sidebar. The sidebar icon is customised with a specified custom CSS variable `(--text color)` with a transition effect.
  3. When the sidebar has the 'close' class the elements inside the sidebar will become fully transparent with an opacity of 0, hiding the text.
- **Sidebar Header Styles:** The sidebar element is displayed as a flex container with the logo and the text displayed with flexibility.
- **Sidebar Toggle Button Styles:** The toggle button is positioned to the right of the sidebar with a background color. On applying the 'dark' class to the body, the toggle button's colour changes.
- **Sidebar Dark Mode styles:** The switch's background colour changes smoothly as the button moves based on the dark mode state.
- **Home Content Styles:** The home content is designed in a way that its content can be adjusted accordingly when the sidebar is closed and the text colour gets updated in dark mode.

7.1.2 Login Page:

This page serves as the entry point for users, facilitating both sign in and sign up processes. Users can either create a new account or access their existing account by providing their credentials (username, email and password).

Step wise implementation of the login page is shown below:

login.html

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale
       =1.0">
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/
       libs/font-awesome/6.4.2/css/all.min.css">
    <link rel="stylesheet" href="login.css">
```

```
</head>

<body>

    <div class="container" id="container">
        <div class="form-container sign-in">
            <form>

                <h1>Create Account</h1><br>

                <span>Register with your personal details</span><br>

                <input type="text" placeholder="Name">
                <input type="email" placeholder="Email">
                <input type="password" placeholder="Password">
                <button>Sign Up</button>
            </form>
        </div>
        <div class="form-container sign-up">
            <form>
                <h1>Sign In</h1>

                <span>or use your email password</span>
                <input type="email" placeholder="Email">
                <input type="password" placeholder="Password">
                <a href="#">Forget Your Password?</a>
                <button>Sign In</button>
            </form>
        </div>
        <div class="toggle-container">
            <div class="toggle">
                <div class="toggle-panel toggle-left">
                    <h1>Hello User!</h1><br>
                    <button class="hidden" id="login">Go Back to
                       Sign Up</button>
                </div>
                <div class="toggle-panel toggle-right">
                    <img src="nrl_logo.jpeg" width="100">
                    <h1>Welcome to Fire and Safety Department</h1>
                    <p>Numaligarh Refinery Limited</p>
                    <button class="hidden" id="register">Sign In</
                       button>
                </div>
            </div>
        </div>
    </div>

```

```
    <script src="login.js"></script>
</body>

</html>
```

Explanation:

- Document Type Declaration (`<!DOCTYPE html>`): Specifies the document type and version of HTML used (HTML5).
- HTML Structure (`<html lang="en"`): Defines the root element of the document with the language set to English ('en').
- Head Section (`<head>`):
  - It contains metadata and links to external resources used by the document.
  - Meta Tags:
    - character set definition (`<meta charset="UTF-8">`)
    - compatibility settings (`<meta http-equiv="X-UA-Compatible" content="IE=edge">`)
    - viewport settings (`<meta name="viewport" content="width=device-width, initial-scale=1.0">`)
- Body Section (`<body>`):
  - The body section constitutes the visible content and structure of the web page. To define a division or a section in the HTML document, the `<div>` tag is used which acts as a container for HTML elements that can be styled with CSS.
  - `<div class="container" id="container">` is the main container that holds all the contents of the page.
  - `<div class="form-container sign-in">` is used that contains all the sign-in form elements. The form includes input fields for the name, email, password and a sign-up button. A `<h1>` header is created and for description the `<span>` tag is used.
  - Similarly, a container for sign-up is also created with the required fields.
  - `<div class="toggle-container">` is created that holds the toggle panels.
  - `<div class="toggle-panel toggle-left">` contains the left panel content. The left toggle panel will consist of a sign-up button to toggle back to the sign-up form and 'Hello user' message.
  - `<div class="toggle-panel toggle-right">` contains the right panel content. The right toggle panel will have the sign-in button to toggle to the sign-in form, the logo and welcome message.

– The JavaScript file is linked externally to provide all the functionalities.

login.css

```
@import url('https://fonts.googleapis.com/css2?family=Montserrat:wght@300;400;500;600;700&display=swap');

*{
    margin: 0;
    padding: 0;
    box-sizing: border-box;
    font-family: 'Montserrat', sans-serif;
}

body{
    background: rgb(19,42,84);
    background: linear-gradient(90deg, rgba(19,42,84,1) 32%, rgba(235,235,240,1) 100%);
    display: flex;
    align-items: center;
    justify-content: center;
    flex-direction: column;
    height: 100vh;
}

.container{
    background-color: #fff;
    border-radius: 30px;
    box-shadow: 0 5px 15px rgba(0, 0, 0, 0.35);
    position: relative;
    overflow: hidden;
    width: 768px;
    max-width: 100%;
    min-height: 480px;
}

.container p{
    font-size: 14px;
    line-height: 20px;
    letter-spacing: 0.3px;
    margin: 20px 0;
}

.container span{
    font-size: 12px;
}

.container a{
```

```
    color: #333;
    font-size: 13px;
    text-decoration: none;
    margin: 15px 0 10px;
}

.container button{
    background-color: #132A54;
    color: #fff;
    font-size: 12px;
    padding: 10px 45px;
    border: 1px solid transparent;
    border-radius: 8px;
    font-weight: 600;
    letter-spacing: 0.5px;
    text-transform: uppercase;
    margin-top: 10px;
    cursor: pointer;
}

.container button.hidden{
    background-color: transparent;
    border-color: #fff;
}

.container form{
    background-color: #fff;
    display: flex;
    align-items: center;
    justify-content: center;
    flex-direction: column;
    padding: 0 40px;
    height: 100%;
}

.container input{
    background-color: #eee;
    border: none;
    margin: 8px 0;
    padding: 10px 15px;
    font-size: 13px;
    border-radius: 8px;
    width: 100%;
    outline: none;
}

.form-container{
```

```
90      position: absolute;
91      top: 0;
92      height: 100%;
93      transition: all 0.6s ease-in-out;
94  }
95
96  .sign-in{
97      left: 0;
98      width: 50%;
99      z-index: 2;
100 }
101
102 .container.active .sign-in{
103     transform: translateX(100%);
104 }
105
106 .sign-up{
107     left: 0;
108     width: 50%;
109     opacity: 0;
110     z-index: 1;
111 }
112
113 .container.active .sign-up{
114     transform: translateX(100%);
115     opacity: 1;
116     z-index: 5;
117     animation: move 0.6s;
118 }
119
120 @keyframes move{
121     0%, 49.99%{
122         opacity: 0;
123         z-index: 1;
124     }
125     50%, 100%{
126         opacity: 1;
127         z-index: 5;
128     }
129 }
130
131
132 .toggle-container{
133     position: absolute;
134     top: 0;
135     left: 50%;
136     width: 50%;
```

```
    height: 100%;
    overflow: hidden;
    transition: all 0.6s ease-in-out;
    border-radius: 150px 0 0 100px;
    z-index: 1000;
}

.container.active .toggle-container{
    transform: translateX(-100%);
    border-radius: 0 150px 100px 0;
}

.toggle{
    background-color: #132A54;
    height: 100%;
    background: linear-gradient(to right, #132A54);
    color: #fff;
    position: relative;
    left: -100%;
    height: 100%;
    width: 200%;
    transform: translateX(0);
    transition: all 0.6s ease-in-out;
}

.container.active .toggle{
    transform: translateX(50%);
}

.toggle-panel{
    position: absolute;
    width: 50%;
    height: 100%;
    display: flex;
    align-items: center;
    justify-content: center;
    flex-direction: column;
    padding: 0 30px;
    text-align: center;
    top: 0;
    transform: translateX(0);
    transition: all 0.6s ease-in-out;
}

.toggle-left{
    transform: translateX(-200%);
}
```

```
184
185 .container.active .toggle-left{
186     transform: translateX(0);
187 }
188
189 .toggle-right{
190     right: 0;
191     transform: translateX(0);
192 }
193
194 .container.active .toggle-right{
195     transform: translateX(200%);
196 }
```

Explanation: The CSS properties are used to define the layout, colors, fonts and animations for different elements on the page.

- The Montserrat font from Google Fonts is imported, which is used throughout the webpage.
- The block inside (*) applies to all the elements which sets the box size to border-box that removes the default margin and padding to set a certain padding and border in the element's total width and height.
- The body includes a background gradient, flexbox layout for a flexible responsive layout structure to center content vertically and horizontally with a viewport height of 100 percent.
- The `.container` has a white background with rounded corners and a shadow for a 3D effect. To handle internal positioning, we have used relative positioning and overflow is set to hidden.
- For styling the text paragraphs ('p'), spans ('span') and links ('a') are given specific font sizes and margins. Similarly, the buttons are given background color, padding, rounded corners and an uppercase style.
- The form container is given absolute positioning and smooth transitions.
- `.sign-in` and `.sign-up` are provided with positioning and visibility.
- .container-active is a class for animation and transition.
- The `.toggle-container` is positioned absolutely to the top left corner and has rounded borders. The transition effect is applied to all CSS properties for smooth animation.
- `.container.active.toggle-container`: When the parent container has the class "active" this style moves the toggle-container to the left by 100 percent, therby changing the border-radius for better apperance.

- `.toggle`: It represents the toggle element itself that is styled with a background color, a linear gradient and text colour. It also has a transition effect for smooth movement.
- `.container.active.toggle`: When the parent container is active the toggle element moves 50 percent to the right to become visible within the container.
- `.toggle-panel` defines the panel inside the toggle container that is positioned absolutely and has flex properties to center its content.
- `.toggle-left` and `.toggle-right` : Here, the classes are used to position content to the left and right of the toggle panel.

7.1.3 Permit Issue Page:

The permit issue page is the direct implementation of the Permit Issue Module. which is a drop down menu. The header has already been discussed above. The **"e-permit Issue Form"** includes fields **"Issued By Department", "Issued By Area"**, **"Issued to Department"**, **"Issued To Area"**, **"Permit Request Date"**, **"Select Shift"**, **"Location of Work"**, and **"Job Description"**. Each field is accompanied by suitable input elements such as text boxes, dropdowns and date pickers. Additional sections include **"SAP"** and **"Work Order"** with radio buttons and text areas for input. The buttons **"Issue Cold Permit"**, **"Issue Hot Permit"** and **"Reset"** provide clear actions for the user to complete the form submission process. The issue permit page also contains a sidebar, which has been described above.

For the given fields and sections, below are the respective code:

- **'Issued By Department':**

```
<div class="input-field">
    <label>Issued By Department:</label>
    <input type="text" placeholder="Department Name" required>
</div>
```

- **'Issued By Area':**

```
<div class="input-field">
    <label>Issued By Area:</label>
    <select required>
      <option value="" disabled selected>Select Area</option>
      <option value="area1" style="color: black;">Area 1</
         option>
      <option value="area2" style="color: black;">Area 2</
         option>
      <option value="area3" style="color: black;">Area 3</
         option>
```

```
      <option value="area4" style="color: black;">Area 4</
         option>
      <option value="area5" style="color: black;">Area 5</
         option>
    </select>
</div>
```

- **'Issued To Department':**

```
<div class="input-field">
   <label>Issued To Department:</label>
   <select required>
     <option value="" disabled selected>Select Department</
        option>
     <option value="Department1" style="color: black;">
        Department 1</option>
     <option value="Department2" style="color: black;">
        Department 2</option>
     <option value="Department3" style="color: black;">
        Department 3</option>
     <option value="Department4" style="color: black;">
        Department 4</option>
     <option value="Department5" style="color: black;">
        Department 5</option>
   </select>
</div>
```

- **'Issued To Area':**

```
<div class="input-field">
   <label>Issued To Area:</label>
   <select required>
     <option value="" disabled selected>Select Area</option>
     <option value="area1" style="color: black;">Area 1</option
        >
     <option value="area2" style="color: black;">Area 2</option
        >
     <option value="area3" style="color: black;">Area 3</option
        >
     <option value="area4" style="color: black;">Area 4</option
        >
     <option value="area5" style="color: black;">Area 5</option
        >
   </select>
</div>
```

- **'Permit Request Date':**

```
<div class="input-field">
   <label>Permit Request Date:</label>
   <input type="date" placeholder="enter birth date" required>
</div>
```

- **'Select Shift':**

```
<div class="input-field">
   <label>Select Shift:</label>
   <select required>
     <option value="" disabled selected>Select Shift</option>
     <option value="6AM-2PM" style="color: black;">6 AM - 2 PM
        </option>
     <option value="2PM-10PM" style="color: black;">2 PM - 10
        PM</option>
     <option value="10PM-6AM" style="color: black;">10 PM - 6
        AM</option>
   </select>
</div>
```

- **'Location of Work':**

```
<div class="input-field">
   <label>Location of Work:</label>
   <input type="text" placeholder="Work Location" required>
</div>
```

- **'Job Description':**

```
<div class="input-field">
   <label>Job Description:</label>
   <input type="text" placeholder="Job Description" required>
</div>
```

- **Section 'SAP':**

```
<div class="sap-container">
  <h2>SAP</h2>
    <form>
      <div class="sa<p-fields">
         <div class="sap-input-field">
            <div class="sap-radio-group">
              <input type="radio" id="With_SAP_notification"
                 name="sap" value="With SAP notification">
              <label for="With_SAP_notification">With SAP
                 notification</label>
              <input type="radio" id="With_SAP_PMO_Notification
                 " name="sap"
```

```
                                value="With SAP PMO
                                   Notification">
            <label for="With_SAP_PMO_Notification">With SAP
               PMO Notification</label>
            <input type="radio" id="With_SAP_PM_sub_order"
               name="sap" value="With SAP PM sub order">
            <label for="With_SAP_PM_sub_order">With SAP PM
               sub order</label>
            <input type=">radio" id="Without_SAP_notification
               " name="sap"
                                value="Without SAP notification
                                   ">
            <label for="Without_SAP_notification">Without SAP
                notification</label>
          </div>
        </div>
        <br>
        <div class="sap-input-field">
          <label>Reason for without SAP notification:</label>
          <textarea id="comments" name="comments" rows="4"
             cols="50" placeholder="Enter your comments here
             ..." </textarea>
        </div>
      </div>
    </form>
</div>
```

- **Section 'Work Order':**

```
<div class="work-order-container">
  <h2>Work Order</h2>
    <form>
      <div class="work-order-fields">
        <div class="work-order-input-field">
          <div class="work-order-radio-group">
            <input type="radio" id="WWON" name="work-order"
               value="With Work Order Number">
            <label for="WWON">With Work Order Number</label>
            <input type="radio" id="WION" name="work-order"
               value="Without Work Order Number">
            <label for="WION">Without Work Order Number</label>
          </div>
        </div>
        <br>
        <div class="work-order-input-field">
          <label>Reason for without Work Order Number</label>
          <textarea id="comments" name="comments" rows="4" cols
             ="50" placeholder="Enter your comments here..."></
```

```
                textarea>
17          </div>
18        </div>
19      </form>
20  </div>
```

The fields and sections of the permit issue page are designed to collect user input necessary for issuing permits. The 'Issue Cold Permit' and 'Issue Hot Permit' buttons lead to respective permit checklists, which the user must complete based on specific job description and requirements. These checklists contain essential regulations and conditions that must be followed during the execution of the permitted job. This ensures compliance with safety standards and proper job execution protocols.

7.1.4 Permit Receive Page:

This page is the direct implementation of the Permit Receiver Module, designed to allow users to receive permits. Upon receiving the permit, it opens to an information page which serves as a comprehensive display of all pertinent details regarding the permit received. It functions to provide users with a consolidated view of permit-related information, encompassing crucial details such as permit numbers, issuance dates, type of permit issued, associated permissions, etc.

Step wise implementation of the permit receive page is shown below:

receive.html

```
1  <!DOCTYPE html>
2  <html lang="en">
3
4  <head>
5      <meta charset="UTF-8">
6      <meta http-equiv="X-UA-Compatible" content="IE=edge">
7      <meta name="viewport" content="width=device-width, initial-scale
          =1.0">
8
9      <link href="https://stackpath.bootstrapcdn.com/bootstrap/5.0.0-
          alpha1/css/bootstrap.min.css" rel="stylesheet">
10     <link rel="stylesheet" type="text/css" href="https://cdn.
          datatables.net/1.11.3/css/dataTables.bootstrap5.min.css">
11     <link rel="stylesheet" href="receive.css">
12
13
14 </head>
15
```

```
<body>
    <div class="form-container">
        <h2><b>All Permit Receive Information (Receive / Pending)</b> </h2>
        <div class="divider"></div>
        <div class="table-container">
            <table id="example" class="table table-striped" style="width:100%">
                <thead>
                    <tr>
                        <th>Sl No.</th>
                        <th>Permit No.</th>
                        <th>Job Description</th>
                        <th>Issued By Area</th>
                        <th>Issued Type</th>
                        <th>Receive Status</th>
                        <th>Action</th>
                    </tr>
                </thead>
                <tbody>
                    <tr>
                        <td></td>
                    </tr>


                </tbody>

            </table>
        </div>
    </div>

    <script src="https://code.jquery.com/jquery-3.5.1.js"></script>
    <script src="https://stackpath.bootstrapcdn.com/bootstrap/5.0.0-alpha1/js/bootstrap.bundle.min.js"></script>
    <script type="text/javascript" src="https://cdn.datatables.net/1.11.3/js/jquery.dataTables.min.js"></script>
    <script type="text/javascript" src="https://cdn.datatables.net/1.11.3/js/dataTables.bootstrap5.min.js"></script>
    <script>
        $(document).ready(function () {
            $('#example').DataTable();
        });
    </script>


</body>

```

```
58 </html>
```

Explanation:

- Head Section (`<head>`):
  - **Bootstrap CSS:** `bootstrap.min.css` links to Bootstrap 5 alpha version for styling the webpage. It refers to the CSS file provided by Bootstrap framework version 5 in its alpha stage. Bootstrap is a popular front-end framework used for designing and styling responsive websites and web applications.
  - **DataTables CSS:** `dataTables.bootstrap5.min.css` links to DataTables CSS for enhancing the table styling. DataTables CSS refers to the CSS file provided by the DataTables library. DataTables is a powerful JavaScript library that enhances HTML tables with advanced functionalities such as sorting, searching, pagination, etc., thereby improving data presentation and user interactions.
  - **Custom CSS:** `receive.css` is linked for additional styling specific to this page.
- Body Section (`<body>`):
  - `<div class="form-container">` contains the primary content wrapper within the HTML structure of a webpage. It encapsulates and organizes the main content related to forms.
  - `<div class="divider"></div>` is a visual divider that separates the `h2` header from the rest of the contents within the form container.
  - `<div class="table-container">` holds the DataTables-enhanced table within the HTML structure.
  - `<table id="example" class="table table-striped" style="width:100%">` defines a table that is uniquely identifiable with an ID `(id='example')` for targeting and manipulation with JavaScript. The `class="table table-striped"` attribute applies predefined Bootstrap classes to the table; the `table` class provides basic table styling, while the `table-striped` class enables alternating row colors, enhancing readability and distinguishing between rows. The `style="width: 100%"` attribute enables the table to occupy the full width of its parent container, providing a responsive layout that adapts to different screen sizes.
  - `<thead>` contains the table headers `<th>` defining column names.
  - `<tbody>`, initially empty, will be populated dynamically with rows of required permit information after integration with database.
- Scripts (`<script>`):
  - **JQuery:** The script tag link `'code.jquery.com/jquery-3.5.1.js'` incorporates the jQuery library version 3.5.1 into the HTML document. jQuery is a free

and open-source JavaScript library designed to simplify HTML DOM tree traversal and manipulation, as well as event handling, CSS animations, and AJAX interactions.

- **Bootstrap JS:** It integrates the Boostrap JavaScript library (`'bootstrap.bundle.min.js'`) into the HTML document for interactive components and modal dialogs. It is a minified and bundled version of Boostrap's JavaScript components, which includes both the core Bootstrap scripts and the `Popper.js` library. `Popper.js` is essential for positioning tooltips and popovers. Tooltips and popovers provide additional context or information on hover or click, thereby enhancing user experience without cluttering the interface.
- **DataTables JS:** The http links `'jquery.dataTables.min.js'` and `'dataTables.bootstrap5.min.js'` incorporates the DataTables JavaScript library and its Bootstrap 5 integration into the HTML document. These files provide advanced table functionalities like sorting, searching and pagination, facilitating improved data presentation and user interaction. Integration of DataTables with Bootstrap 5 provides a cohesive and responsive user interface.
- **Initialization of DataTables:** The JavaScript snippet `'$(document).ready(function () {$('example').DataTable();o});'` initializes the DataTable on document load. The DataTable features are applied to the table with `id="example"`, making it interactive with advanced functionalities.

receive.css

```
/* Google Font Import - Poppins */
@import url('https://fonts.googleapis.com/css2?family=Poppins:
   wght@300;400;500;600;700&display=swap');
*{
    margin: 0;
    padding: 0;
    box-sizing: border-box;
    font-family: 'Poppins', sans-serif;
}

:root{
    /* ===== Colors ===== */
    --body-color: #E4E9F7;
    --sidebar-color: #FFF;
    --primary-color: #695CFE;
    --primary-color-light: #F6F5FF;
    --toggle-color: #DDD;
    --text-color: #707070;

    /* ====== Transition ====== */
    --tran-03: all 0.2s ease;
    --tran-03: all 0.3s ease;
```

```
    --tran-04: all 0.3s ease;
    --tran-05: all 0.3s ease;
}

body{
    min-height: 100vh;
    background-color: var(--body-color);
    transition: var(--tran-05);
}


/* Existing CSS */

/* Form Styles */
.form-container {
    position:relative;
    left:10px;
    top:10px;
    padding: 20px;
    background: var(--sidebar-color);
    border-radius: 8px;
    margin: 20px;
    box-shadow: 0 5px 10px rgba(33, 30, 235, 0.1);

}

.form-container h2 {
    margin-bottom: 20px;
    font-size: 24px;
    color: var(--text-color);
}

h2 {
    margin-top: 20px; /* Space above h2 header */
}
th {
    white-space: nowrap;

}
.divider {
    border-bottom: 1px solid rgb(175, 175, 175);
    margin-bottom: 20px;
}

.dataTables_wrapper .row:first-child {
    margin-bottom: 20px; /* Space between Show/Search controls and
       table */
```

```
}

.table-container {
    padding: 25px;
    margin: auto;
}



@media (max-width: 2560px) {
    .table-container {
        overflow-x: auto;
        width: 100%;
        /* Full width on smaller screens */
    }

    .dataTables_length, .dataTables_filter {
        display: block;
        margin-bottom: 10px; /* Space between stacked elements */
    }
    .dataTables_length {
        margin-right: 0; /* Remove right margin when stacked */
    }
}
```

Explanation:

- `'@import url()'` imports the Poppins font from Google Fonts with various weights (300, 400, 500, 600, 700) for use throughout the webpage.
- The universal selector `*` applies the specified styles to all elements within the webpage. `'margin'`, `'padding'`, `'box-sizing'` and `'font-family'` are set.
- `':root'` defines CSS variables globally. Custom properties `'--body-color'`,`'--sidebar-close'`, `'--primary-color'`, etc., and transitions are specified. They can be reused throughout the stylesheet.
- Body styles `'min-height'`, `'background-color'` and `'transition'` are set. `'background-color'` sets the background color to the value of `'--body-color'` defined above. Transition effect is defined to `'--tran-05'`.
- `.form-container` specifies the properties to style the form container element with padding, a background color, rounded corners, a shadow and slight offsets. `'.form-container h2'` styles the `'<h2>'` headers within the container to have appropriate spacing, a consistent font size, and a specific text color.
- `'white-space:nowrap'` prevents the table headers `'th'` from wrapping to the next line.

- `'.divider'` class styles the `'divider'` element to have a bottom border that acts as a visual divider. It adds spacing below the element to separate it from subsequent content.
- `'.dataTables_wrapper .row:first-child'` class sets the `'margin-bottom'` to 20px, adding space below the first child row of the `'dataTables_wrapper'`.
- `'.table_container'` class specifies a padding of 25px inside the container and centers it horizontally through `'margin:auto'`.
- Media queries applies styles for screens with a maximum width of 2560px. It ensures that the `'.table_container'` element becomes scrollable horizontally and occupies the full width of the screen on smaller devices also. It modifies the display of `'.dataTables_length'` and `'.dataTables_filter'` elements to stack vertically with added spacing and removes the right margin from `'.dataTables_length'` when stacked.

7.1.5 All Permit Issued Information Page (Show All Permit Page)

This page provides a comprehensive overview of all the permits that have been issued. It features a searchable and paginated table, including columns for **'Sl No.'**, **'Permit No.'**, **'Job Description'**, **'Issued To Area'**, **'Issued Type'**, **'Receive Status'** and **'Action'**.

Each row of the table represents an individual permit entry, with specific details. The **'Issued Type'** column indicates whether the permit is a cold or hot permit. **'Receive Status'** shows the current status of the permit. The **'Action'** column provides relevant actions that can be taken, such as confirming whether the permit has been received.

This layout ensures that users can efficiently monitor and manage all issued permits, ensuring transparency and easy access to critical information. The design of the page facilitates quick searching and smooth navigation through potentially large lists of permits.

This page has the same code design as the permit receive page.

7.2. Database Design

A well-designed database is crucial for any data-driven projects, ensuring efficient data storage, quick retrieval and a scalable system. The essential components of database design includes:

- **Requirements Analysis:** This phase involves identifying the data requirements, including the types of data to be stored, their relationships and any constraints that need to be applied to the data.

- **Conceptual Design:** Here, an Entity-Relationship (ER) Diagram is created to define the entities, their attributes and relationships between the entities.
- **Logical Design:** The conceptual design is transformed into a logical structure, typically a relational model, detailing the tables, columns, and relationships that can be implemented within a database management system.
- **Physical Design:** The logical design is translated into a physical structure that describes how data will be stored in the database. It includes defining database tables that correspond to entities, indexing columns and specifying data types and constraints to ensure data integrity.

7.2.1 Microsoft SQL Server Management Studio

Microsoft SQL Server Management Studio (SSMS) is an integrated environment for managing any SQL infrastructure. It provides a suite of tools to access, configure, manage, administer, and develop all components of SQL Server. SSMS provides tools for creating, modifying and managing databases, tables and indexes. Users can efficiently handling database objects, while maintaining data integrity. The environment also supports executing and optimizing SQL queries. SSMS supports SQL Server, Azure SQL Database, and Azure SQL Data Warehouse.

7.2.2 Relational Database Management System (RDBMS)

Relational Database Management System is a type of database management system that stores data in a structured format using rows and columns. Users can create, update and manage relational databases where data is organized into tables. These tables are interconnected based on shared data. RDBMS uses Structured Query Language (SQL) for database access and management, enabling complex queries, transactions and data manipulation. RDBMS follows the ACID (Atomicity, Consistency, Isolation, Durability) property that ensures reliable transaction processing. Popular RDBMS examples are MySQL, PostgreSQL, Oracle Database and Microsoft SQL Server.

To facilitate the project's requirements, the "E_permit" relational database was created using Microsoft SQL Server Management Studio. The database serves as a structured repository for managing permit-related information efficiently. The 'Issue_permit' table within this database is designed to store essential details regarding issued permits.

```
1 CREATE TABLE [dbo].[Issue_permit](
2     [ID] [int] IDENTITY(1,1) NOT NULL,
3     [Permit_Issue_ID] [varchar](36) NOT NULL,
4     [Permit_Issue_No] [varchar](30) NULL,
5     [IssuedByDeparment] [varchar](30) NULL,
```

```
    [IssuedByArea] [varchar](20) NULL,
    [IssuedToDeparment] [varchar](30) NULL,
    [IssuedToArea] [varchar](20) NULL,
    [PermitIssueDate] [date] NULL,
    [PermitShift] [varchar](10) NULL,
    [LocationOfWork] [varchar](100) NULL,
    [JobDescription] [varchar](100) NULL,
    [IsGasTestRequired] [bit] NULL,
    [IsPermitToBeRecievedByIndividual] [bit] NULL,
    [WithSAPnotification] [bit] NULL,
    [WithSAPPMONotification] [bit] NULL,
    [WithSAPPMsuborder] [bit] NULL,
    [WithuoutSAPnotification] [bit] NULL,
    [ReasonForWithoutSAPnotification] [varchar](100) NULL,
    [WithWorkOrderNumber] [bit] NULL,
    [WithoutWorkOrderNumber] [bit] NULL,
    [ReasonForWithoutWorkOrderNumber] [varchar](100) NULL,
    [IssueColdPermit] [bit] NULL,
    [IssueHotPermit] [bit] NULL,
    [IsPermitReceived] [bit] NULL,
 CONSTRAINT [PK_Issue_pe_872FA80BE636C285] PRIMARY KEY CLUSTERED
(
    [Permit_Issue_ID] ASC
)
WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE OFF,IGNORE_DUP_KEY =
   OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON,
   OPTIMIZE_FOR_SEQUENTIAL_KEY
= OFF) ON [PRIMARY]
) ON [PRIMARY]
GO
```

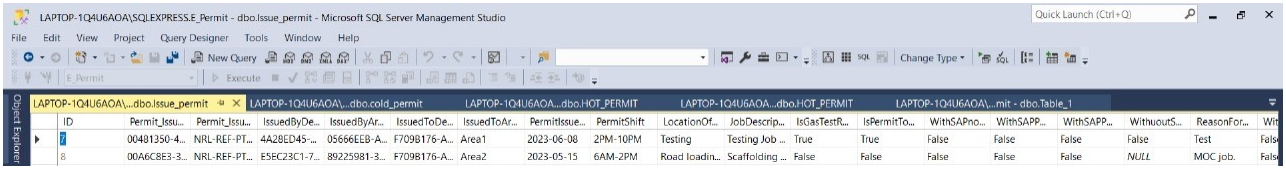

Figure 3: Table 'Issue_permit'

Tables 'registration' 'cold_permit' and 'HOT_PERMIT' have been implemented similarly.

7.2.3 Stored Procedures

Stored Procedures are pre-compiled and stored group of SQL queries within a database management system. They allow developers to execute frequently used or complex SQL queries. This procedures are designed to enhance database performance and security by reducing

network traffic by executing on the server side. Stored procedures can accept input parameters, execute queries, modify data and return results or status indicators.

Our "E_permit" database incorporates several stored procedures to streamline data operations and enhance functionality within the permit system. These stored procedures are:

- LOGINSELECT
- insertregistration
- coldpermitinsert
- hotpermitinsert
- issuepermitinsert
- showallpermit
- showpermitbyID
- UpdateRecievedStatus

The 'coldpermitinsert' stored procedure facilitates the insertion of cold permit checklist information into the 'cold_permit' table.

```
ALTER PROCEDURE
[dbo].[coldpermitinsert]

    @permitId varchar(36),
    @WorkAreaInspected bit,
    @SurroundingARea nvarchar(max),
    @EquipmentElectricallyIsolated bit,
    @RunningWater nvarchar(max),
    @Equipment bit,
    @EquipmentProperty nvarchar(MAX),
    @EquipmentWaterFlushed bit,
    @IronSulphide bit,
    @EquipmentProperlySteamed bit,
    @GasTest nvarchar(MAX),
    @StandbyPerson nvarchar(MAX),
    @VentilationandLighting bit,
    @AreaCordoned bit,
    @RadioIsotopes bit,
    @ppe nvarchar(MAX),
    @Stnt nvarchar(20)= ' '
as
begin
```

```
23
24 insert into cold_permit(permitId,
25 WorkAreaInspected ,
26 SurroundingARea ,
27 EquipmentElectricallyIsolated ,
28 RunningWater ,
29 Equipment ,
30 EquipmentProperty,
31 EquipmentWaterFlushed ,
32 IronSulphide ,
33 EquipmentProperlySteamed ,
34 GasTest ,
35 StandbyPerson ,
36 VentilationandLighting ,
37 AreaCordoned ,
38 RadioIsotopes ,
39 ppe
40 )
41 values (@permitId,@WorkAreaInspected,@SurroundingARea,
      @EquipmentElectricallyIsolated,
42 @RunningWater,@Equipment,@EquipmentProperty,@EquipmentWaterFlushed,
      @IronSulphide,@EquipmentProperlySteamed,@GasTest,@StandbyPerson,
      @VentilationandLighting,@AreaCordoned,@RadioIsotopes,@ppe)
43 end
```

The other stored procedures have been implemented similarly.

7.3. Back End

The backend of our project has been developed using C#, the ASP.NET framework and the Model-View-Controller (MVC) architecture. This combination ensures a robust, scalable and maintainable system to manage permit related operations efficiently.

7.3.1 C# programming language

C# is a modern, innovative, open-source, cross-platform, object-oriented programming language developed by Microsoft. It is part of the .NET ecosystem, designed for building a wide range of applications, from desktop and web applications to mobile and cloud-based solutions. C# enforces strong typing, thereby catching errors at compile time rather than at runtime.

7.3.2 .NET Framework

.NET Framework is a software development framework for building and running applications on Windows. It provides a consistent object-oriented programming environment whether object code is stored and executed locally, executed locally but web-distributed or executed remotely.

Architecture of .NET Framework

The two major components of .NET Framework are the **Common Language Runtime** and the **.NET Framework Class Library.**

- The Common Language Runtime (CLR) is the execution engine for .NET applications that handles running these applications. It provides services like thread-management, garbage collection, type-safety, exception handling and more. .NET applications, written in C#, compiled into Common Intermediate Language (CIL). This compiled code is stored in assemblies-files with a .dll or .exe file extension. When the app runs the CLR takes the assembly and uses a just-in-time compiler (JIT) to turn into machine code that can execute on the specific architecture of the computer it is running on.
- .NET Framework Class Library provides a set of APIs and types for common functionality. It contains a vast array of classes, interfaces, and value types that provide a wide range of functionalities. These functionalities support and simplify many common programming tasks, such as file input/output, data manipulation, and application security.

ASP.NET Framework

ASP.NET is an open-source web framework, created by Microsoft, for building modern web apps and services with .NET. ASP.NET is cross-platform and runs on Windows, Linux, macOS and Docker. ASP.NET adds to the .NET platform the following features:

- Base framework for processing web requests in C#.
- **Webpage templating syntax,** known as Razor for building dynamic webpages using C#.
- **Libraries for common web patterns**, such as Model-View-Controller (MVC).
- **Authentication system** that includes libraries, a database and template pages for handling logins, including multi-factor authentication and external authentication with Google, X and more.
- **Editor extensions** to provide syntax highlighting, code completion and other functionality specifically for developing webpages.

7.3.3 Model-View-Controller (MVC) Architecture

The Model-View-Controller Framework is an architecture/ design pattern that seperates an application into three main logical components:

- Model
- View
- Controller

Each architectural components is built to handle specific development aspects of an application. MVC is one of the most frequently used industry-standard web-development frameworks to create scalable and extensible projects.

1. **Model:** Model serves as the central component responsible for managing the application's data and enforcing business logic. It encapsulates the data structure, performs CRUD operations (Create, Read, Update, Delete) with the database or external data sources, and ensures data integrity by enforcing business rules. The Model communicates with both the View and Controller components, responding to requests from the Controller by retrieving or updating data. It notifies the View and Controller of any changes in state, facilitating seamless data interaction and application functionality.
2. **View:** View is used for all the UI logic of the application. It manages the presentation and user interface of the application and generates and renders the user interface elements based on data provided by the Model through interactions with the Controller.
3. **Controller:** The Controller acts as an intermediary between the Model and the View. It processes all the business logic and incoming requests based on user input, manipulates data using the Model component, and interacts with the View to render the final output.

To cite as example, below are the code implementation of Model, View and Controller for the 'All Permit Issued Information Page'.

Model:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;

namespace Epermit.Models
{
    public class PermitDetails
```

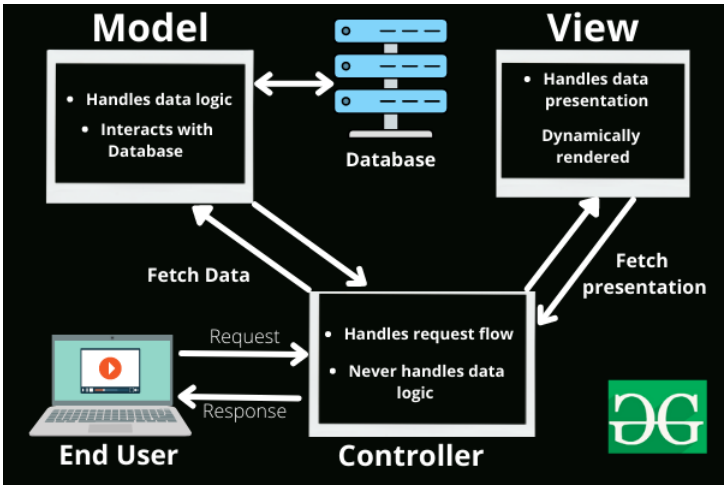

Figure 4: MVC Architecture Design (Courtesy of GeeksForGeeks)

```
    {
        public string Permit_Issue_No { get; set; }
        public string IssuedByDeparment { get; set; }
        public string IssuedByArea { get; set; }
        public string JobDescription { get; set; }
        public string PermitShift { get; set; }
        public bool IsGasTestRequired { get; set; }
        public DateTime PermitIssueDate { get; set; }
        public string IssuedToDeparment { get; set; }
        public string IssuedToArea { get; set; }
        public string LocationOfWork { get; set; }
        public bool IsPermitToBeRecievedByIndividual { get; set; }

        public bool WithSAPnotification { get; set; }
        public bool WithSAPPMONotification { get; set; }
        public bool WithSAPPMsuborder { get; set; }
        public bool WithuoutSAPnotification { get; set; }
        public string ReasonForWithoutSAPnotification { get; set; }
        public bool WithWorkOrderNumber { get; set; }
        public bool WithoutWorkOrderNumber { get; set; }
        public string ReasonForWithoutWorkOrderNumber { get; set; }

    }
}
```

The `'PermitDetails'` class represents the model component defining the structure and data for permits within the application. It includes various properties such as permit issue number, issuing department, job description, shift, and SAP-related notifications.This class ensures comprehensive data capture and facilitates the management and processing of permit information. The View uses this model to display permit details to user, while the controller

interacts with the model to handle user input and update the data accordingly.

Controller:

```
public ActionResult Showallpermit()
{
    List<PermitDetails> listOfPermitDetails = new List<PermitDetails
       >();

    using (SqlConnection con = new SqlConnection(System.
       Configuration.ConfigurationManager.ConnectionStrings["
       permitdatabase"].ConnectionString))
    {
        SqlCommand sqlCommand = con.CreateCommand();
        sqlCommand.Connection = con;
        sqlCommand.Parameters.Clear();
        sqlCommand.CommandText = "[E_Permit].[dbo].[showallpermit]";
        sqlCommand.CommandType = System.Data.CommandType.
           StoredProcedure;
        sqlCommand.Connection.Open();

    using (SqlDataReader reader = sqlCommand.ExecuteReader())
        {
            listOfPermitDetails = DataReaderMapToList<PermitDetails
               >(reader).ToList();
        }
        sqlCommand.Connection.Close();
    }
    return View(listOfPermitDetails);
}
```

The `'Showallpermit'` method in the controller retrieves the list of all permits from the database and displays them in the view. It establishes a connection to the database using a connection string from the configuration file, then executes the stored procedure to fetch the permit data. The results are mapped to a list of objects and this list is then passed to the view for display.

View:

```
@model List<Epermit.Models.PermitDetails>
@{
    ViewBag.Title = "Showallpermit";
    Layout = "~/Views/Shared/_Layout.cshtml";
}
<div class="showall-form-container">
    <h2><b>All Permit Issue Information (Issued / Pending)</b> </h2>
    <div class="divider"></div>
    <div class="table-container">
```

```
        <table id="example" class="table table-striped" style="width
           :100%">
            <thead>
                <tr>
                    <th class="showall">Sl No.</th>
                    <th>Permit No.</th>
                    <th>Job Description</th>
                    <th>Issued To Area</th>
                    <th>Issued Type</th>
                    <th>Receive Status</th>
                    <th>Action</th>
                </tr>
            </thead>
            <tbody>

                @if (Model != null)
                {
                    if (Model.Count > 0)
                    {
                        foreach (var item in Model)
                        {
                            <tr>
                                <td>1.</td>
                                <td>@item.Permit_Issue_No</td>
                                <td>@item.JobDescription</td>
                                <td>Cold Permit</td>
                                <td>>Receive Pending</td>
                                <td>Permit Not Received</td>
                            </tr>
                        }
                    }
                }
            </tbody>
        </table>
    </div>
</div>

<script src="https://code.jquery.com/jquery-3.5.1.js"></script>

<script src="https://stackpath.bootstrapcdn.com/bootstrap/5.0.0-
   alpha1/js/bootstrap.bundle.min.js"></script>
<script type="text/javascript" src="https://cdn.datatables.net
   /1.11.3/js/jquery.dataTables.min.js"></script>
<script type="text/javascript" src="https://cdn.datatables.net
   /1.11.3/js/dataTables.bootstrap5.min.js"></script>
<script>
    $(document).ready(function () {
```

```
53          $('#example').DataTable();
54      });
55  </script>
```

The `Showallpermit` view renders a table displaying a list of all permit issues (issued and pending). It uses a model of type `List<PermitDetails>` and sets the view's title and layout. The table includes columns for serial number, permit number, job description, issued area, issued type, receive status, and actions. If the model contains permit data, each permit is displayed in a row. The view integrates jQuery and DataTables plugins to enhance table functionality, enabling features like sorting and searching. Bootstrap is used for styling the table and ensuring responsive design.

8. Conclusion and Future Scope

The development of the **e-permit system** represents a significant advancement in establishing standard work protocols and ensuring safe working practices within the refinery. By adhering to the guidelines of **OISD-STD-105**, the system efficiently processes, stores and manages critical information. The successful implementation of key modules such as **Permit Issue** and **Permit Receiver** showcases the system's ability to provide a comprehensive solution for managing permits within the refinery environment.

The system's architecture, leveraging the **MVC** framework ensures a robust, scalable and maintainable platform. This architecture, combined with the use of **C#** programming and the **ASP.NET** framework provides a solid foundation for managing permit-related operations efficiently.

However, the e-permit system is not yet fully functional. There is considerable scope and potential for future development and expansion to make the system fully operational for refinery use. Integration in existing systems such as SAP for seamless data exchange and work flow automation can be explored. Developing a mobile-application version of this system can significantly improve accessibility for field workers and supervisors, allowing them to manage permits on the go. The system can be integrated with AI for real-time detection of protocol violations.