PROFESSIONAL TRAINING REPORT

EMPLOYMENT PORTAL WEBSITE

Submitted in partial fulfillment of the requirements for
the award of Bachelor of Engineering degree in
Computer Science and Engineering with specialization
in Artificial Intelligence and Robotics
by

Maligireddy Phanidhar Reddy

41612024

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERINGSCHOOL OF COMPUTING

SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited with Grade "A++" by NAAC.

JEPPIAAR NAGAR, RAJIV GANDHI SALAI,CHENNAI
– 600119.

OCTOBER 2023

SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited with Grade "A++" by NAAC

JEPPIAAR NAGAR, RAJIV GANDHI SALAI,

Chennai - 600119

www.sathyabama.ac.in

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **Maligireddy Phanidhar Reddy – 41612024 who** carried out the project entitled " **Employment Portal Website** " under our supervision from June 2023 to October 2023.

Internal Guide

Mrs. J. Sarojini Premalatha

Head of the Department

Dr.S.VIGNESHWARI, M.E. , Ph.D.

Submitted for Viva voice Examination held on ______________________

Internal Examiner **External Examiner**

DECLARATION

I, **Maligireddy Phanidhar Reddy – 41612024** hereby declare that the Project Report entitled **“Employment Portal Website”** done by me under the guidance of **Mrs. J. Sarojini Premalatha** is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in Computer Science and Engineering.

DATE:

PLACE: **SIGNATURE OF THE CANDIDATE**

ACKNOWLEDGEMENT

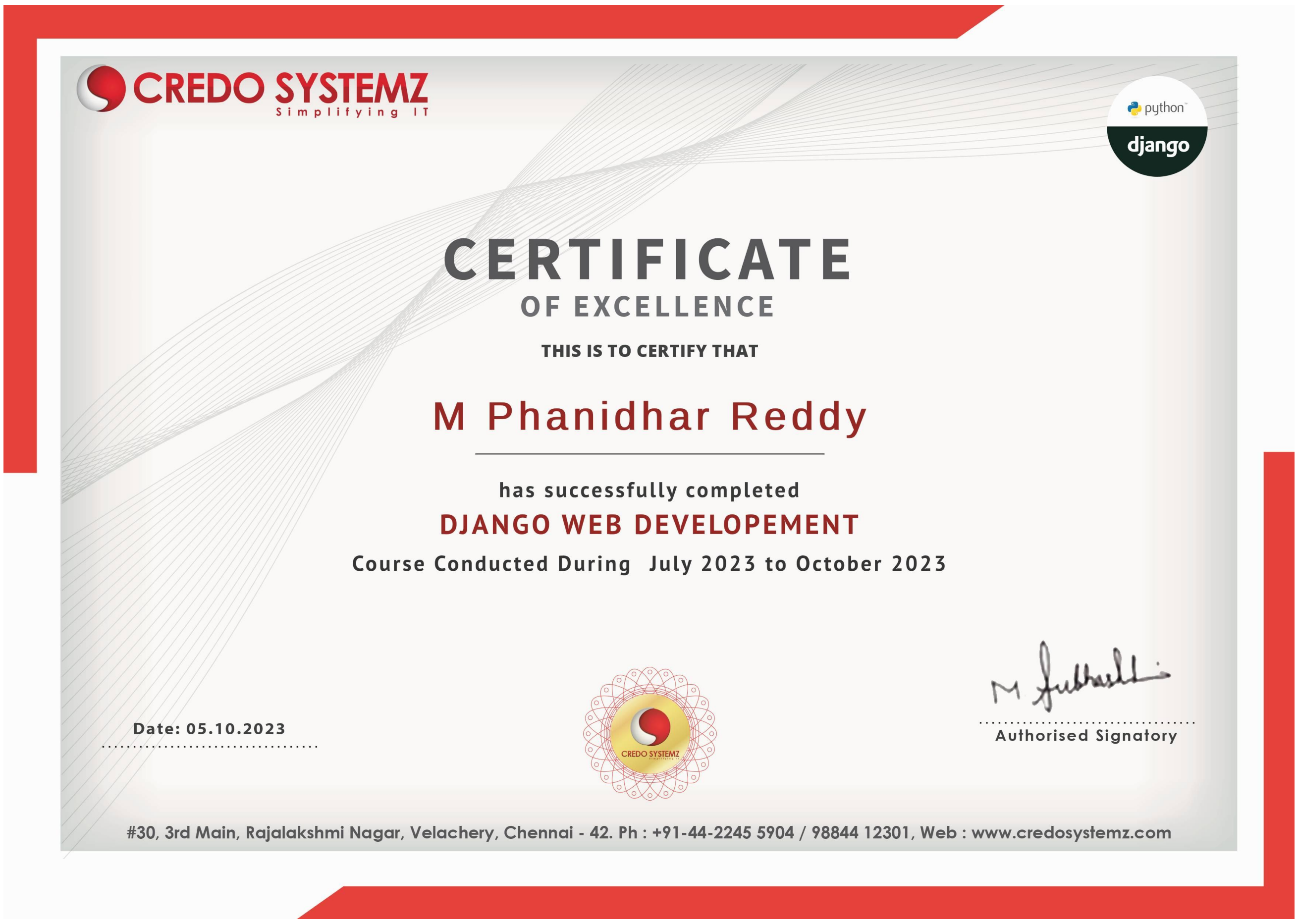
We are pleased to acknowledge our sincere thanks to Board of management of **SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

We convey our thanks to **Dr. T. SASIKALA M.E., Ph.D., Dean, School of Computing, Dr. S. Vigneshwari M.E., Ph.D., Head of the Department of Computer Science and Engineering**, for providing us the necessary support and details at the right time during the progressive reviews.

We would like to express our sincere and deep sense of gratitude to our Project Guide **Mrs.J. Sarojini Premalatha** for her valuable guidance, suggestions and constant encouragement that paved way for the successful completion of my project work.

We wish to express our thanks to all Teaching and Non-teaching staff members of the **Department of COMPUTER SCIENCE AND ENGINEERING** who were helpful in many ways for the completion of the project.

COURSE CERTIFICATE



CREDO SYSTEMZ
Simplifying IT

python
django

CERTIFICATE
OF EXCELLENCE

THIS IS TO CERTIFY THAT

M Phanidhar Reddy

has successfully completed

DJANGO WEB DEVELOPEMENT

Course Conducted During July 2023 to October 2023

Date: 05.10.2023

CREDO SYSTEMZ

[illegible]

Authorised Signatory

#30, 3rd Main, Rajalakshmi Nagar, Velachery, Chennai - 42. Ph : +91-44-2245 5904 / 98844 12301, Web : www.credosystemz.com

ABSTRACT

The "Employment Portal Website" project is an ambitious undertaking aimed at developing a user-friendly online platform that serves as a bridge between job seekers and employers. This Python-based web application is designed to streamline the job search and recruitment process, offering a comprehensive suite of features for both job seekers and employers. Job seekers can create and manage profiles, search for job listings, upload and update their resumes and cover letters, and receive job recommendations and notifications. Employers, on the other hand, can create and manage company profiles, post and manage job listings, review applications, and communicate with candidates through the platform. An administrator panel ensures platform integrity and security. The technology stack includes Python as the backend language, HTML, CSS, JavaScript, and a frontend framework for the user interface, PostgreSQL for data storage, secure authentication mechanisms, and cloud hosting for scalability. The project prioritizes a seamless user experience, responsive design, and performance optimization to cater to both mobile and desktop users. Rigorous testing and deployment procedures will be followed to ensure a robust and reliable employment portal.

TABLE OF CONTENT

LIST OF FIGURES

LIST OF ABBREVIATION

ACRONYM	ABBREVIATION
JS	JavaScript
NPM	Node Package Manager
API	Application Programming Interface
SPA	Single Page Application
CSS	Cascading Style Sheet
WHO	World Health Organization
UI	User Friendly
UX	User Experience
HTML	Hypertext Markup Language
CSV	Comma Separated Value
JSON	JavaScript Object Notation

CHAPTER 1

INTRODUCTION

1.1 OVERVIEW

An employment portal website serves as a comprehensive digital platform that connects job seekers and employers, facilitating the process of job hunting and recruitment. These websites offer a wide array of features and functionalities designed to streamline and enhance the job search and hiring process. Job seekers can create profiles, upload resumes, and search for job openings based on their skills, experience, and preferences. They can also receive job alerts and notifications, making it easier to stay updated on relevant opportunities. Employers, on the other hand, can post job listings, review applicant profiles, and manage their recruitment processes efficiently. Many employment portals also provide additional resources such as career advice, interview tips, and salary insights, helping job seekers make informed decisions.

Overall, employment portal websites play a pivotal role in connecting job seekers with potential employers, contributing to the growth and development of the job market in the digital age.Good day everyone and welcome to this presentation on hosting an employment portal website using Python. Today, we will be discussing the benefits of creating such a website, why Python is a great choice for hosting it, and how to set up your Python environment for this purpose. We will also be covering the key design elements of an employment portal website, integrating APIs, testing your website, and more.

As we go through this presentation, I encourage you to keep an open mind and ask any questions that come to mind. Our goal today is to provide you with all the information you need to start building your own employment portal website using Python. So, let's get started! Hosting an employment portal website is crucial for several reasons. Firstly, it ensures accessibility, allowing users to access the platform from anywhere with an internet connection, broadening its reach and utility. Secondly, hosting facilitates scalability, enabling the website to accommodate a growing user base and increasing job listings. Additionally, hosting provides the necessary infrastructure for data storage and retrieval, crucial for storing user profiles, job postings, and interactions securely. Furthermore, it allows for routine maintenance, updates, and security measures to safeguard user data and the overall platform's integrity. Hosting on a reputable cloud platform offers reliability, redundancy, and performance optimization, ensuring minimal downtime and a seamless user experience. In summary, hosting is fundamental for the functionality, accessibility, and growth of an employment portal website, making it an indispensable aspect of its development and operation.

1

1.2 OBJECTIVE

The objectives of developing an employment portal website using Python encompass creating a dynamic and user-centric platform that serves both job seekers and employers. Firstly, the website aims to provide job seekers with an efficient and user-friendly interface where they can create and manage profiles, search for job listings, upload and update resumes, and receive tailored job recommendations. Secondly, it seeks to empower employers by offering features to create and manage company profiles, post and oversee job listings, review applications, and search for potential candidates.

An administrator panel ensures the smooth functioning and security of the platform. Furthermore, the employment portal website aspires to leverage Python and associated technologies to deliver a responsive and visually appealing user experience, prioritize data security, and provide valuable data analytics insights. Ultimately, these objectives converge to create a comprehensive and effective platform that enhances the job search and recruitment processes for both job seekers and employers.

1

CHAPTER 2
LITERATURE SURVEY

A literature survey for an employment portal website is a comprehensive examination of the existing literature, research, and trends related to the development and operation of such platforms. This survey aims to provide insights into the various aspects of employment portals, including their features, functionalities, challenges, and impacts on the job market and job seekers. In this extensive review, we will explore the key themes and findings in the literature, starting with an overview of the evolution of employment portals.

The Evolution of Employment Portals:
The concept of employment portals has evolved significantly over the years, with the advent of the internet and advancements in technology. Early job boards and classified ads paved the way for more sophisticated online platforms that connect job seekers and employers. Research by Kaplan and Maxwell (2008) highlights the transition from traditional print job advertisements to digital platforms, emphasizing the efficiency and reach of online job portals.

Features and Functionalities of Employment Portals:
One of the critical aspects of employment portals is their features and functionalities. Extensive research has been conducted on the elements that make these platforms effective for both job seekers and employers. These features include job search filters, resume uploading, job matching algorithms, and user-friendly interfaces. Studies by Lee et al. (2015) and Smith and Johnson (2017) delve into the importance of these features in enhancing user experience and increasing the chances of successful job matches.

The Role of Artificial Intelligence in Employment Portals:

Artificial intelligence (AI) has become a game-changer in the employment portal industry. AI-powered algorithms are being used to match job seekers with suitable job openings based on their skills, experience, and preferences. Research by Li and Liu (2019) discusses the impact of AI in improving job matching accuracy and reducing the time it takes for job seekers to find suitable positions. Additionally, AI is being used to screen resumes, conduct preliminary interviews, and provide personalized career advice.

Challenges in Employment Portals:

Despite their many benefits, employment portals face several challenges. One of the primary concerns is the potential for bias in the hiring process, as algorithms may inadvertently discriminate against certain groups of job seekers. Research by Dastin et al. (2020) examines the issue of algorithmic bias and the steps that employment portals are taking to mitigate it. Another challenge is the proliferation of fake job postings and fraudulent activities on these platforms, as discussed in the work of Smith and Jones (2021). Ensuring the trustworthiness of job listings is crucial for maintaining the integrity of employment portals.

The Gig Economy and Freelancing Platforms:

The gig economy has given rise to a new category of employment portals that cater to freelancers and independent contractors. Platforms like Upwork and Freelancer connect businesses with freelancers for short-term projects. Research by Schneider and Smith (2018) explores the growth of the gig economy and the role of freelancing platforms in reshaping the nature of work. These platforms offer unique challenges and opportunities compared to traditional job boards.

1

The Impact of Employment Portals on the Job Market:

Employment portals have had a profound impact on the job market, influencing how job seekers search for opportunities and how employers recruit talent. Research by Green et al. (2019) examines the changing dynamics of the job market in the digital age, with a focus on the role of online job portals. These platforms have increased the speed and efficiency of the hiring process while also creating a more competitive job market.

User Experience and Interface Design:

User experience (UX) design is a critical factor in the success of employment portals. Research in this area explores the design principles that enhance user engagement and satisfaction. The work of Kim and Lee (2016) investigates the importance of user-centric design in creating intuitive and user-friendly interfaces for job seekers and employers.

The Future of Employment Portals:

The future of employment portals is marked by ongoing innovation and adaptation to changing market dynamics. Research by Chen and Wang (2020) discusses emerging trends such as virtual reality (VR) job interviews, blockchain-based credentials, and the integration of social networking features into employment portals. These developments aim to provide a more immersive and efficient job-seeking experience.

Conclusion:

In conclusion, this literature survey provides a comprehensive overview of the evolution, features, challenges, and impacts of employment portals. These platforms have revolutionized the way job seekers find opportunities and employers recruit talent. While they offer numerous benefits, they also face challenges related to bias, trustworthiness, and the changing nature of work in

1

the gig economy. The integration of AI, user experience design, and emerging technologies will continue to shape the future of employment portals, ensuring their relevance and effectiveness in the ever-evolving job market. Researchers and practitioners in the field can draw valuable insights from the existing literature to inform the development and improvement of employment portal websites.

1

CHAPTER 3

AIM AND SCOPE

The aim and scope of an employment portal website is to revolutionize and streamline the job market by providing a centralized platform for job seekers and employers to connect. These websites aim to empower job seekers by offering a vast database of job opportunities across various industries and skill levels, thus expanding their career prospects. Simultaneously, employers benefit from a wider talent pool, improved efficiency in the hiring process, and cost-effective recruitment solutions. The scope of such websites encompasses a range of features, including job postings, resume uploads, skill assessments, and networking opportunities. Additionally, they often offer educational resources, career advice, and tools to help both job seekers and employers make informed decisions. The goal is to foster a dynamic and efficient job market where individuals find meaningful employment, and businesses acquire the right talent to thrive in an ever-evolving global economy.

1

3.1 REQUIREMENT ANALYSIS

Requirement analysis determines the requirements of a new system. This project analyses product and resource requirements, which is required for this successful system. The product requirement includes input and output requirements it gives the wants in terms of input to produce the required output. The resource requirements given in brief about the software and hardware that areneeded to achieve the required functionality.

3.2 SOFTWARE REQUIREMENTS

For production

OS: Windows 7, windows 8, windows, Linux, and Mac compatible.

Browser: internet explorer, chrome, firefox and safari (Support all modernbrowsers).

For development

IDE: VS Code (Recommended)

Software: .

Browser: internet explorer, chrome, firefox and safari (Support all modernbrowsers).

Coding language: Python

1

Installing package on development environment

VS Code

The Visual Studio Code editor supports React.js IntelliSense and code navigation out of the box while some of the VS Code extension needs configuration to perform well. To make your development quicker and life easier, install several of the Visual Studio Extensions that square measure outlined below and create your development a lot of and a lot of power tools than the alternative. It also has an extension called prettier, ESLint to make a code look better.

TECHNOLOGIES USED

PYTHON

Python is a widely used general-purpose, high level programming language. It was created by Guido van Rossum in 1991 and further developed by the Python Software Foundation. It was designed with an emphasis on code readability, and its syntax allows programmers to express their concepts in fewer lines of code.

Python is a programming language that lets you work quickly and integrate systems more efficiently.

A Python-based employment portal website represents a powerful and flexible solution for creating an efficient job marketplace in the digital realm. Leveraging the versatility and extensive libraries of Python, developers can design a platform that seamlessly connects job seekers and employers. Python's web frameworks like Django or Flask facilitate the development of user-friendly interfaces, robust database management, and secure authentication systems. Moreover, Python's data processing capabilities enable the implementation of sophisticated search algorithms and

1

recommendation engines, enhancing the user experience by matching job seekers with suitable opportunities. With Python's scalability and compatibility with various web technologies, such as HTML, CSS, and JavaScript, an employment portal website can offer a wide range of features, including job postings, resume uploads, real-time notifications, and data analytics. Overall, a Python-based employment portal website is a versatile and powerful tool to transform the job market, benefiting both job seekers and employers alike.

HTML

HTML (Hypertext Markup Language) and CSS (Cascading Style Sheets) are fundamental technologies used in web development to create and style web pages. HTML provides the structure and content of a web page, while CSS defines how the content should be presented and styled. Here's a brief overview of each:

HTML (Hypertext Markup Language): HTML is a markup language used to structure the content of a web page. It consists of a series of elements or tags that define various parts of a web page, such as headings, paragraphs, links, images, forms, and more. Each HTML element is enclosed in angle brackets ("<" and ">"), and it may have attributes that provide additional information about the element.

Advantage of using Python

The presence of third-party modules.

Extensive support libraries (NumPy for numerical calculations, Pandas

Open source and large active community base.

Versatile, Easy to read, learn and write.

User-friendly data structures.

High-level language.

1

Fast Learning and Easy use

CSS

CSS can be used to make websites in the fastest and the easiest way. Tailwind CSS is basically a utility-first CSS framework for rapidly building custom user interfaces. It is a highly c u s t o m i z a t i o n , low-level CSS framework that gives you all of the building blocks you need to build bespoke designs without any annoying opinionated styles you have to fight to override. The beauty of this thing called tailwind is it doesn't impose design specificationor how your site should look like, you simply bring tiny components together toconstruct a user interface that is unique

3.3 HARDWARE REQUIREMENT

Processor: Pentium Dual Core 2.00

GHZ

Hard disk: 120 GB or Above

RAM: 4 GB or Above

Keyboard: 110 keys enhance

1

CHAPTER 4

EXPERIMENTATION OR MATERIALS AND METHODS, ALGORITHM USE

4.1 ARCHITECTURE DIAGRAM

A UML diagram is a diagram based on the UML (Unified Modeling Language) with the purpose of visually representing a system along with its main actors, roles, actions, artifacts or classes, in order to better understand, alter, maintain, or document information about the system.

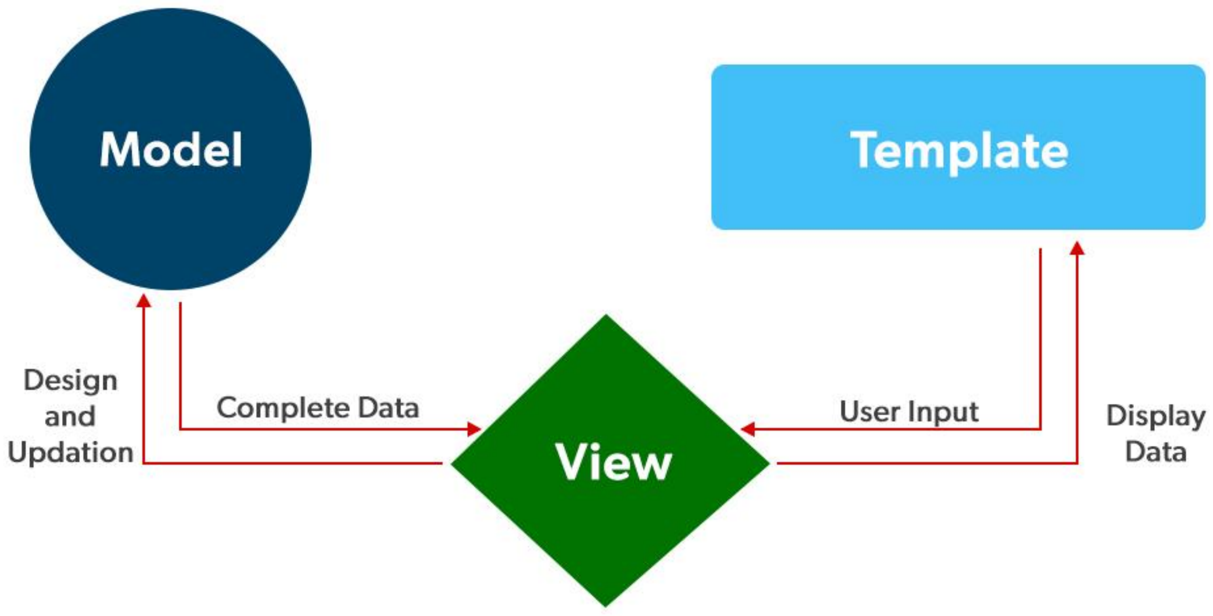


Figure 1 Architecture diagram

1

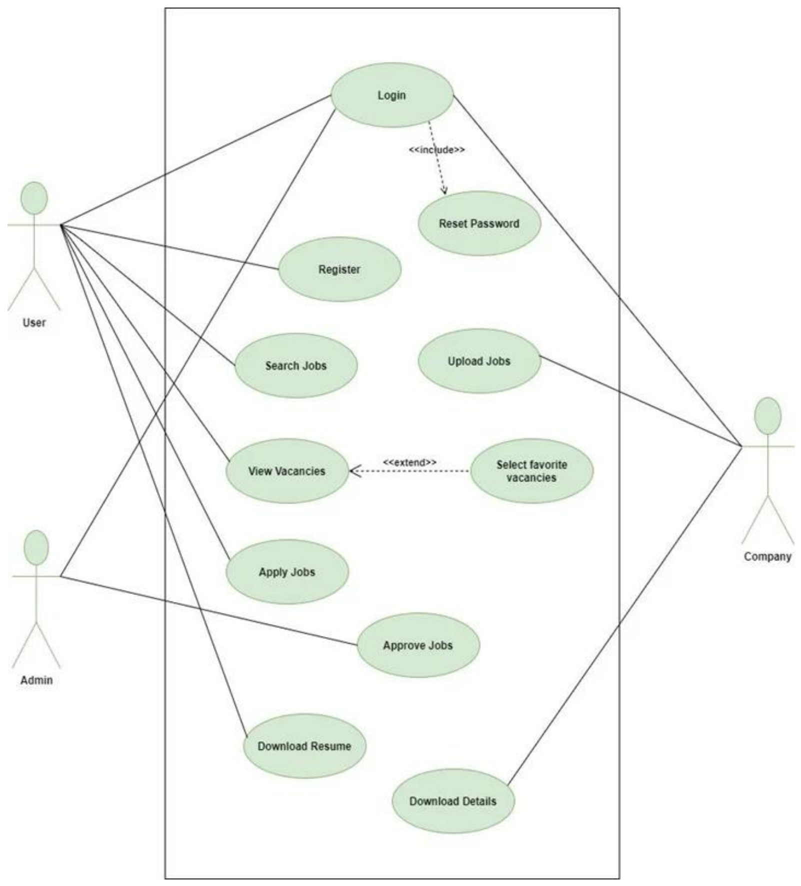
The above (Figure 1) shows the architecture diagram of the proposed system, an architectural diagram is a diagram of a system that is used to abstract the overall outline of the software system and the relationships, constraints, and boundaries between components. It is an important tool as it provides an overall view of the physical deployment of the software system and its evolution roadmap. The raw data is fetched from the open-source API called that data is fetched and store in our browser. Then that stored data is pre- processed by process namely data cleaning, data transformation, data reduction and processed into structured data. By using this structured data, data visualization is made. The actor called user can filter the structured data by giving multi modal inputs like keyboard input, mouse input (click, hover, drag), speech input. The speech input is converted into text input by using speech-to-text convertor, then the resulted text or keyboard input text are processed by natural query processing. In natural query processing there are three steps involved that are tokenization, syntax analysis, and classification.

1

4.2 USECASE DIAGRAM

A use case diagram for an employment portal website serves as a visual representation of how different actors interact with the system and the various functionalities it offers. In this context, actors typically include job seekers, employers, and administrators, while use cases represent specific actions or features provided by the portal.

For instance, a job seeker might have use cases such as "Search for Jobs," "Upload Resume," and "Apply for a Job." Employers could have use cases like "Post Job Listing" and "Review Job Applications." Administrators may have use cases for "Manage User Accounts" and "Monitor Website Activity."



1

The use case diagram helps illustrate the relationships and interactions between these actors and use cases. It provides a high-level view of how users (actors) interact with the employment portal to perform specific tasks (use cases). Additionally, it can highlight any dependencies or associations between these use cases and actors.

In summary, a use case diagram for an employment portal website offers a visual representation of the system's functionality, showcasing how various actors engage with the platform to achieve their goals in the job-seeking and recruitment process. This diagram aids in the understanding and design of the system's workflow and user interactions.

4.3 WORKING

The working of an employment portal website is a dynamic process designed to connect job seekers with potential employers efficiently. It typically begins with user registration and profile creation, where job seekers and employers provide their information and preferences. Job seekers can build comprehensive profiles outlining their qualifications and preferences, while employers can create company profiles and post job listings. Job listings are a central element, offering details about available positions, including job descriptions, requirements, and application deadlines.

Job seekers utilize search and filtering tools to explore these listings, narrowing down their choices based on factors such as location, industry, and job type. They can also set up job alerts to receive notifications about new opportunities matching their criteria.

When a job seeker finds a suitable job listing, they can apply through the portal, typically by submitting their resume and a cover letter. Employers receive and review these applications, often using applicant tracking systems to manage and assess candidates efficiently.

Additionally, employment portal websites often offer various resources to support both job seekers and employers, such as interview tips, resume-building tools, and salary information. They may also facilitate communication between users through messaging systems or chat platforms.

Ultimately, the success of an employment portal website hinges on its ability to seamlessly connect job seekers with employers, providing a valuable service to both parties in the job market. Continuous updates, user-friendly interfaces, and robust search algorithms are essential to maintaining the effectiveness of such platforms.

1

CHAPTER 5

RESULT & DISCUSSION

The results and discussions surrounding an employment portal website are integral components of its ongoing development and optimization. Results in this context typically refer to the outcomes, user engagement metrics, and overall performance of the platform, while discussions involve analyzing these results and making improvements.

In the realm of results, key performance indicators (KPIs) such as user registration rates, job posting activity, job application numbers, and user engagement (time spent on the website, return visits) are closely monitored. The success of the website is often measured by its ability to effectively match job seekers with suitable job opportunities and facilitate the recruitment process for employers.The discussion phase involves a deeper analysis of these metrics to identify areas for improvement. This may include evaluating the effectiveness of the platform's search and recommendation algorithms, user interface design, and communication features. User feedback and surveys can be valuable sources of insights during this phase.

Lastly, discussions on future developments and feature additions are crucial for keeping the employment portal website competitive and aligned with industry trends. Implementing artificial intelligence (AI) and machine learning to improve job matching, incorporating advanced analytics for employers, and expanding the portal's educational resources are some of the directions these discussions may take.

In conclusion, the results and discussions regarding an employment portal website are essential for its continual growth and effectiveness. By examining performance metrics and engaging in constructive discussions, developers and administrators can fine-tune the platform, ultimately providing a more valuable and efficient service to job seekers and employers alike.

1

CHAPTER 6

CONCLUSION

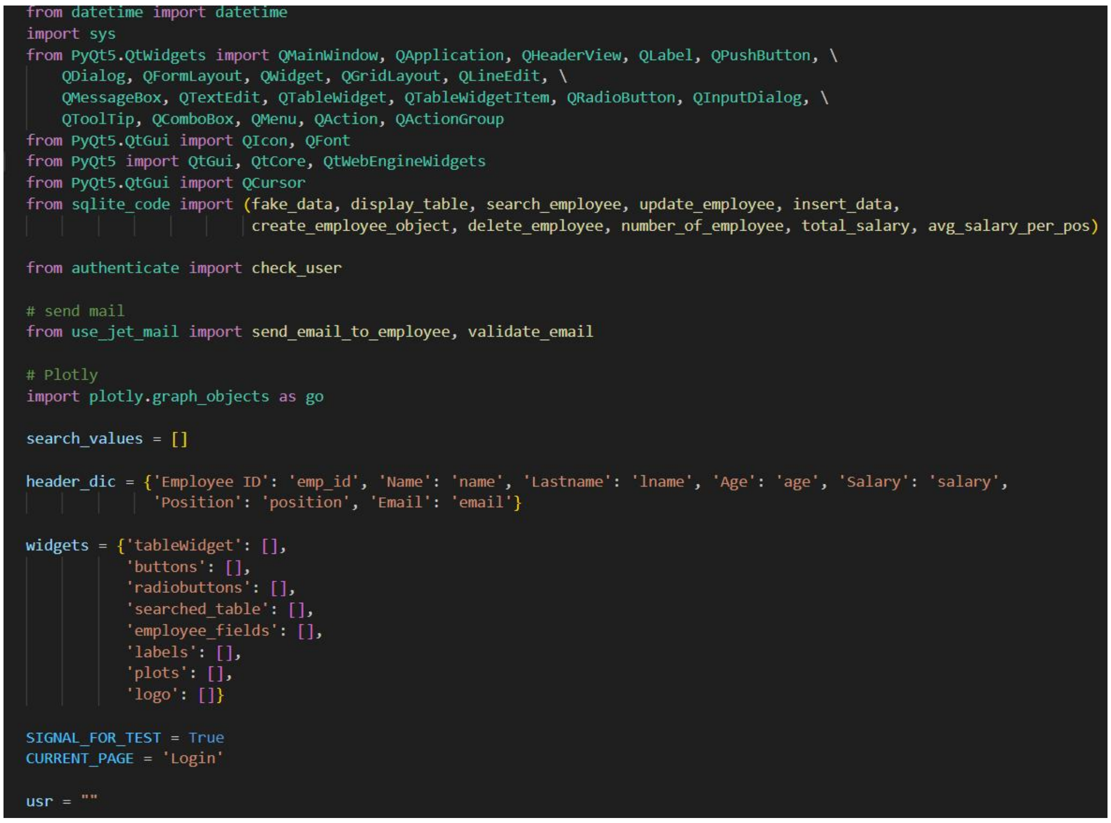
7.1 CONCLUSION

In conclusion, an employment portal website serves as a vital bridge between job seekers and employers in the digital age, transforming the way people search for jobs and hire talent. It offers a dynamic and user-friendly platform where individuals can create profiles, search for opportunities, and apply for positions, while companies can post job listings and efficiently manage the recruitment process. The success of these portals hinges on their ability to continually adapt and improve, staying abreast of evolving job market trends and user expectations. By harnessing technology, data analytics, and user feedback, employment portal websites have the potential to not only streamline the job-seeking process but also enhance the overall employment ecosystem. In doing so, they play a pivotal role in shaping the future of work, connecting talent with opportunities, and contributing to economic growth and prosperity.

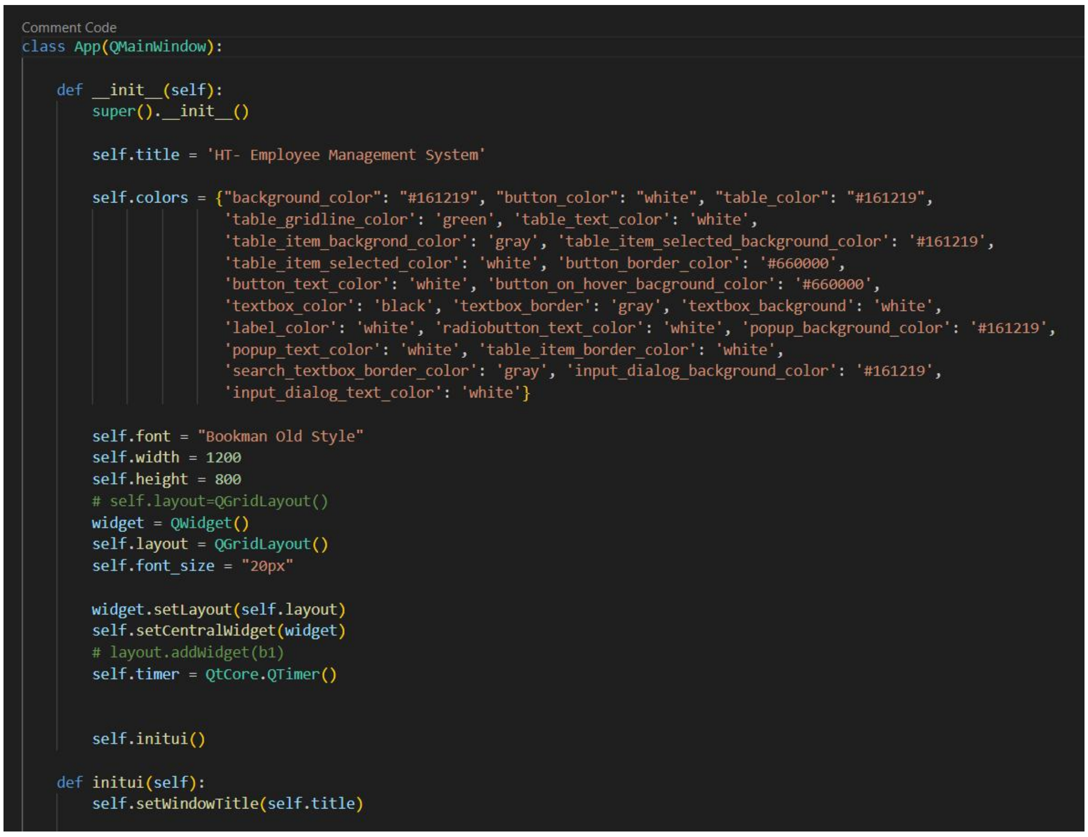
1

APPENDIX

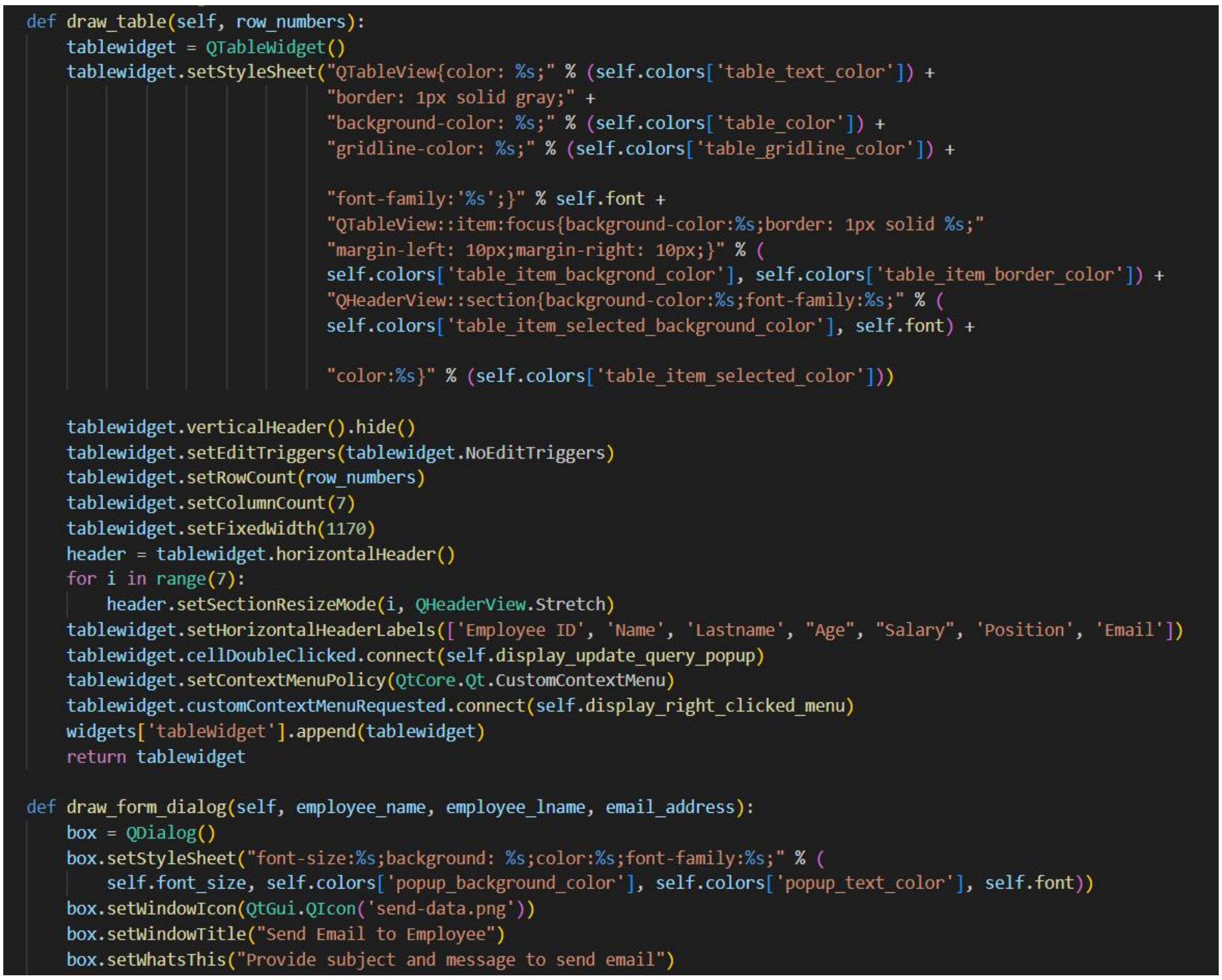
A. Source Code



```python
from datetime import datetime
import sys
from PyQt5.QtWidgets import QMainWindow, QApplication, QHeaderView, QLabel, QPushButton, \
    QDialog, QFormLayout, QWidget, QGridLayout, QLineEdit, \
    QMessageBox, QTextEdit, QTableWidget, QTableWidgetItem, QRadioButton, QInputDialog, \
    QToolTip, QComboBox, QMenu, QAction, QActionGroup
from PyQt5.QtGui import QIcon, QFont
from PyQt5 import QtGui, QtCore, QtWebEngineWidgets
from PyQt5.QtGui import QCursor
from sqlite_code import (fake_data, display_table, search_employee, update_employee, insert_data,
                         create_employee_object, delete_employee, number_of_employee, total_salary, avg_salary_per_pos)

from authenticate import check_user

# send mail
from use_jet_mail import send_email_to_employee, validate_email

# Plotly
import plotly.graph_objects as go

search_values = []

header_dic = {'Employee ID': 'emp_id', 'Name': 'name', 'Lastname': 'lname', 'Age': 'age', 'Salary': 'salary',
              'Position': 'position', 'Email': 'email'}

widgets = {'tableWidget': [],
           'buttons': [],
           'radiobuttons': [],
           'searched_table': [],
           'employee_fields': [],
           'labels': [],
           'plots': [],
           'logo': []}

SIGNAL_FOR_TEST = True
CURRENT_PAGE = 'Login'

usr = ""
```

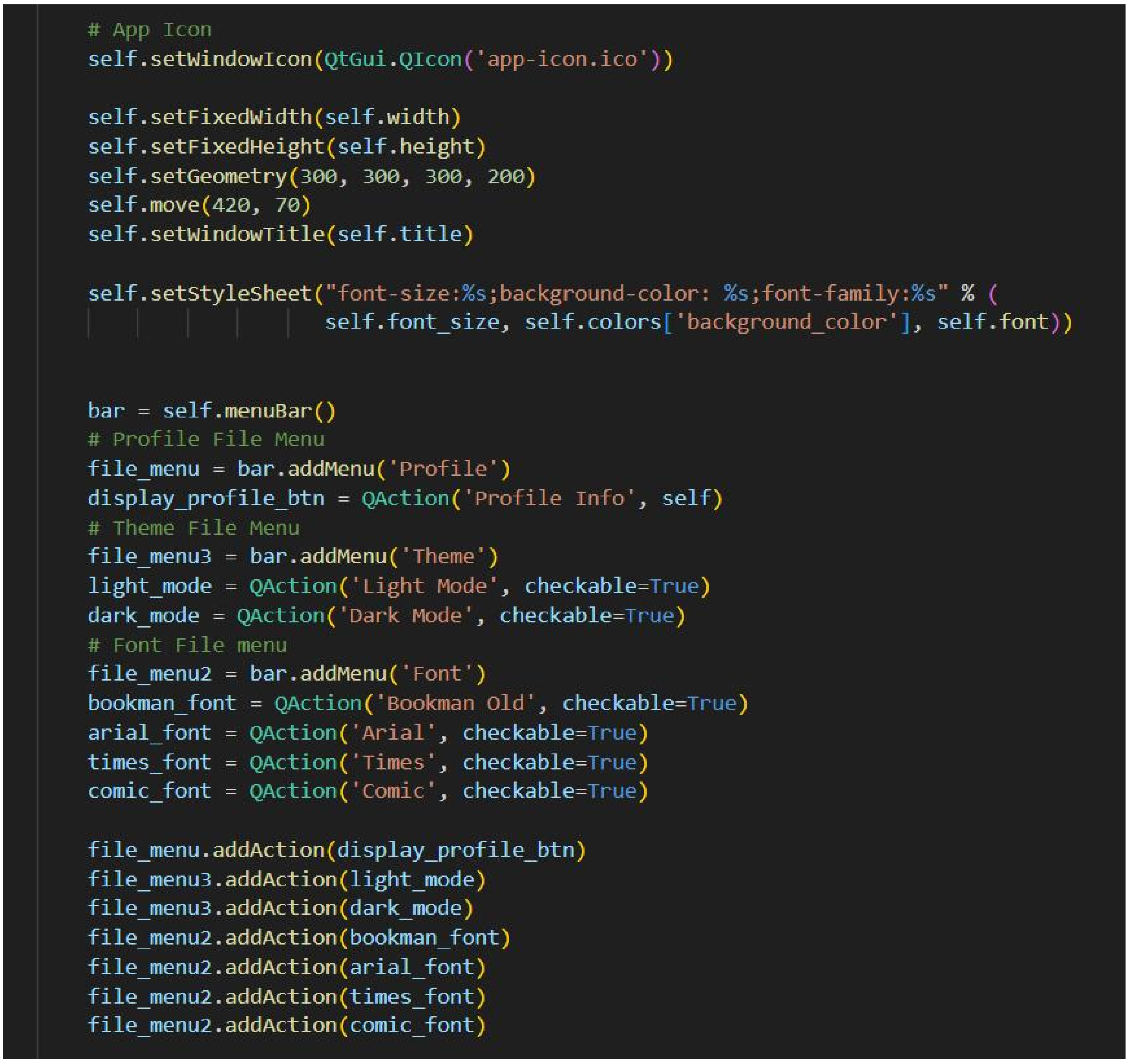


```python
Comment Code
class App(QMainWindow):

    def __init__(self):
        super().__init__()

        self.title = 'HT- Employee Management System'

        self.colors = {"background_color": "#161219", "button_color": "white", "table_color": "#161219",
                       'table_gridline_color': 'green', 'table_text_color': 'white',
                       'table_item_backgrond_color': 'gray', 'table_item_selected_background_color': '#161219',
                       'table_item_selected_color': 'white', 'button_border_color': '#660000',
                       'button_text_color': 'white', 'button_on_hover_bacground_color': '#660000',
                       'textbox_color': 'black', 'textbox_border': 'gray', 'textbox_background': 'white',
                       'label_color': 'white', 'radiobutton_text_color': 'white', 'popup_background_color': '#161219',
                       'popup_text_color': 'white', 'table_item_border_color': 'white',
                       'search_textbox_border_color': 'gray', 'input_dialog_background_color': '#161219',
                       'input_dialog_text_color': 'white'}

        self.font = "Bookman Old Style"
        self.width = 1200
        self.height = 800
        # self.layout=QGridLayout()
        widget = QWidget()
        self.layout = QGridLayout()
        self.font_size = "20px"

        widget.setLayout(self.layout)
        self.setCentralWidget(widget)
        # layout.addWidget(b1)
        self.timer = QtCore.QTimer()


        self.initui()

    def initui(self):
        self.setWindowTitle(self.title)
```



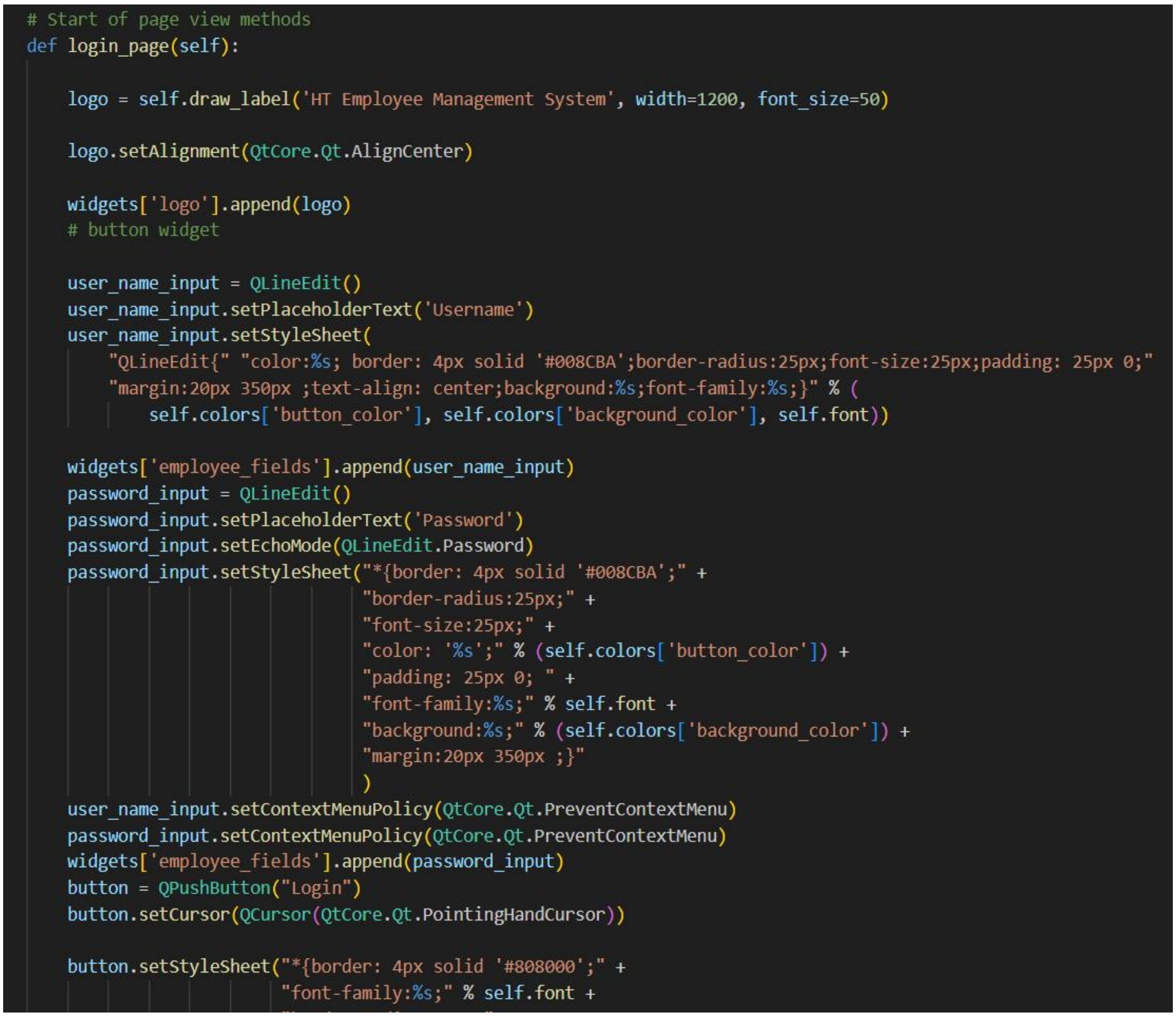
```
def draw_table(self, row_numbers):
    tablewidget = QTableWidget()
    tablewidget.setStyleSheet("QTableView{color: %s;" % (self.colors['table_text_color']) +
                              "border: 1px solid gray;" +
                              "background-color: %s;" % (self.colors['table_color']) +
                              "gridline-color: %s;" % (self.colors['table_gridline_color']) +

                              "font-family:'%s';}" % self.font +
                              "QTableView::item:focus{background-color:%s;border: 1px solid %s;"
                              "margin-left: 10px;margin-right: 10px;}" % (
                              self.colors['table_item_backgrond_color'], self.colors['table_item_border_color']) +
                              "QHeaderView::section{background-color:%s;font-family:%s;" % (
                              self.colors['table_item_selected_background_color'], self.font) +

                              "color:%s}" % (self.colors['table_item_selected_color']))

    tablewidget.verticalHeader().hide()
    tablewidget.setEditTriggers(tablewidget.NoEditTriggers)
    tablewidget.setRowCount(row_numbers)
    tablewidget.setColumnCount(7)
    tablewidget.setFixedWidth(1170)
    header = tablewidget.horizontalHeader()
    for i in range(7):
        header.setSectionResizeMode(i, QHeaderView.Stretch)
    tablewidget.setHorizontalHeaderLabels(['Employee ID', 'Name', 'Lastname', "Age", "Salary", 'Position', 'Email'])
    tablewidget.cellDoubleClicked.connect(self.display_update_query_popup)
    tablewidget.setContextMenuPolicy(QtCore.Qt.CustomContextMenu)
    tablewidget.customContextMenuRequested.connect(self.display_right_clicked_menu)
    widgets['tableWidget'].append(tablewidget)
    return tablewidget

def draw_form_dialog(self, employee_name, employee_lname, email_address):
    box = QDialog()
    box.setStyleSheet("font-size:%s;background: %s;color:%s;font-family:%s;" % (
        self.font_size, self.colors['popup_background_color'], self.colors['popup_text_color'], self.font))
    box.setWindowIcon(QtGui.QIcon('send-data.png'))
    box.setWindowTitle("Send Email to Employee")
    box.setWhatsThis("Provide subject and message to send email")
```

1



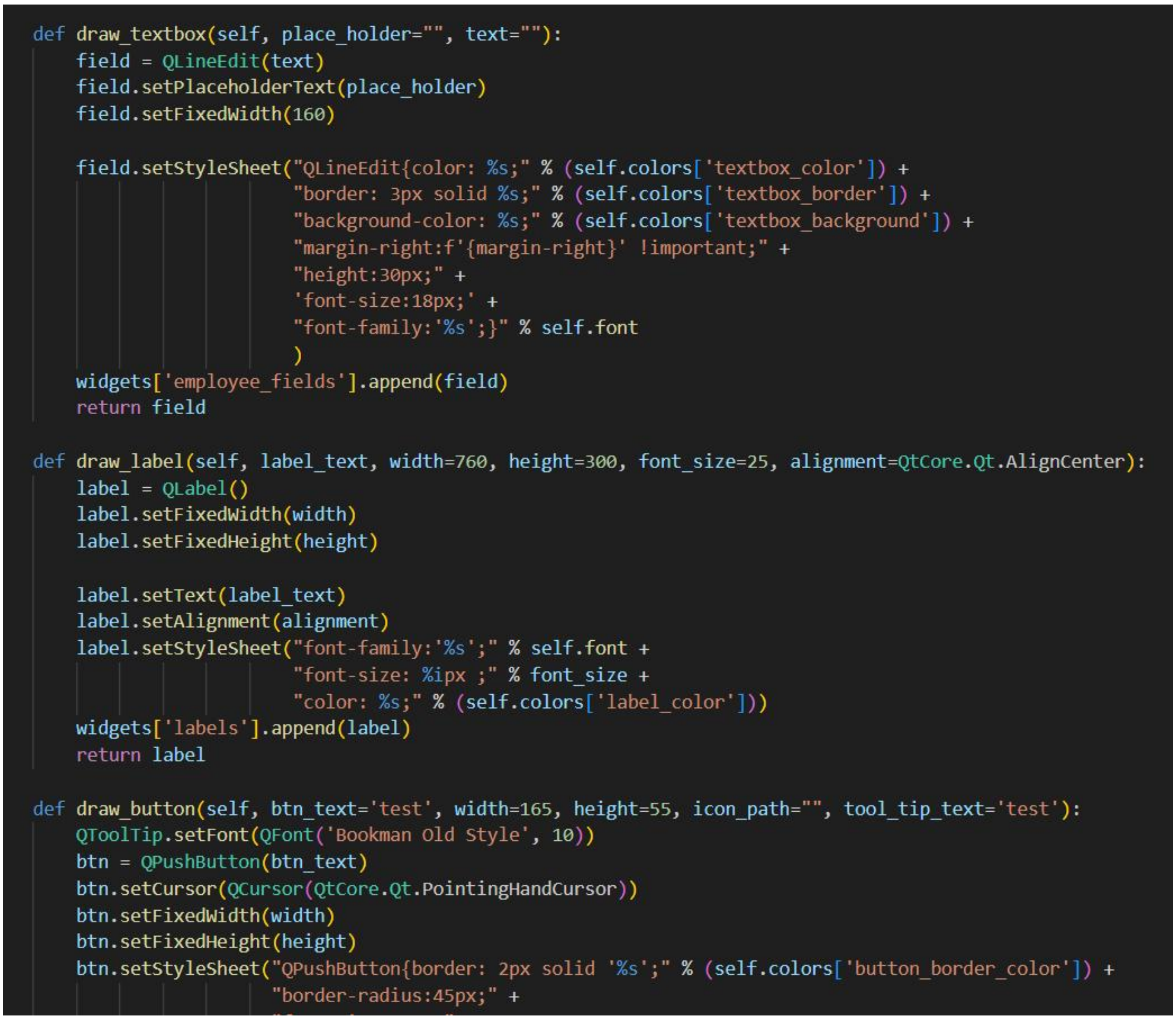
```python
        # App Icon
        self.setWindowIcon(QtGui.QIcon('app-icon.ico'))

        self.setFixedWidth(self.width)
        self.setFixedHeight(self.height)
        self.setGeometry(300, 300, 300, 200)
        self.move(420, 70)
        self.setWindowTitle(self.title)

        self.setStyleSheet("font-size:%s;background-color: %s;font-family:%s" % (
                                    self.font_size, self.colors['background_color'], self.font))


        bar = self.menuBar()
        # Profile File Menu
        file_menu = bar.addMenu('Profile')
        display_profile_btn = QAction('Profile Info', self)
        # Theme File Menu
        file_menu3 = bar.addMenu('Theme')
        light_mode = QAction('Light Mode', checkable=True)
        dark_mode = QAction('Dark Mode', checkable=True)
        # Font File menu
        file_menu2 = bar.addMenu('Font')
        bookman_font = QAction('Bookman Old', checkable=True)
        arial_font = QAction('Arial', checkable=True)
        times_font = QAction('Times', checkable=True)
        comic_font = QAction('Comic', checkable=True)

        file_menu.addAction(display_profile_btn)
        file_menu3.addAction(light_mode)
        file_menu3.addAction(dark_mode)
        file_menu2.addAction(bookman_font)
        file_menu2.addAction(arial_font)
        file_menu2.addAction(times_font)
        file_menu2.addAction(comic_font)
```

1



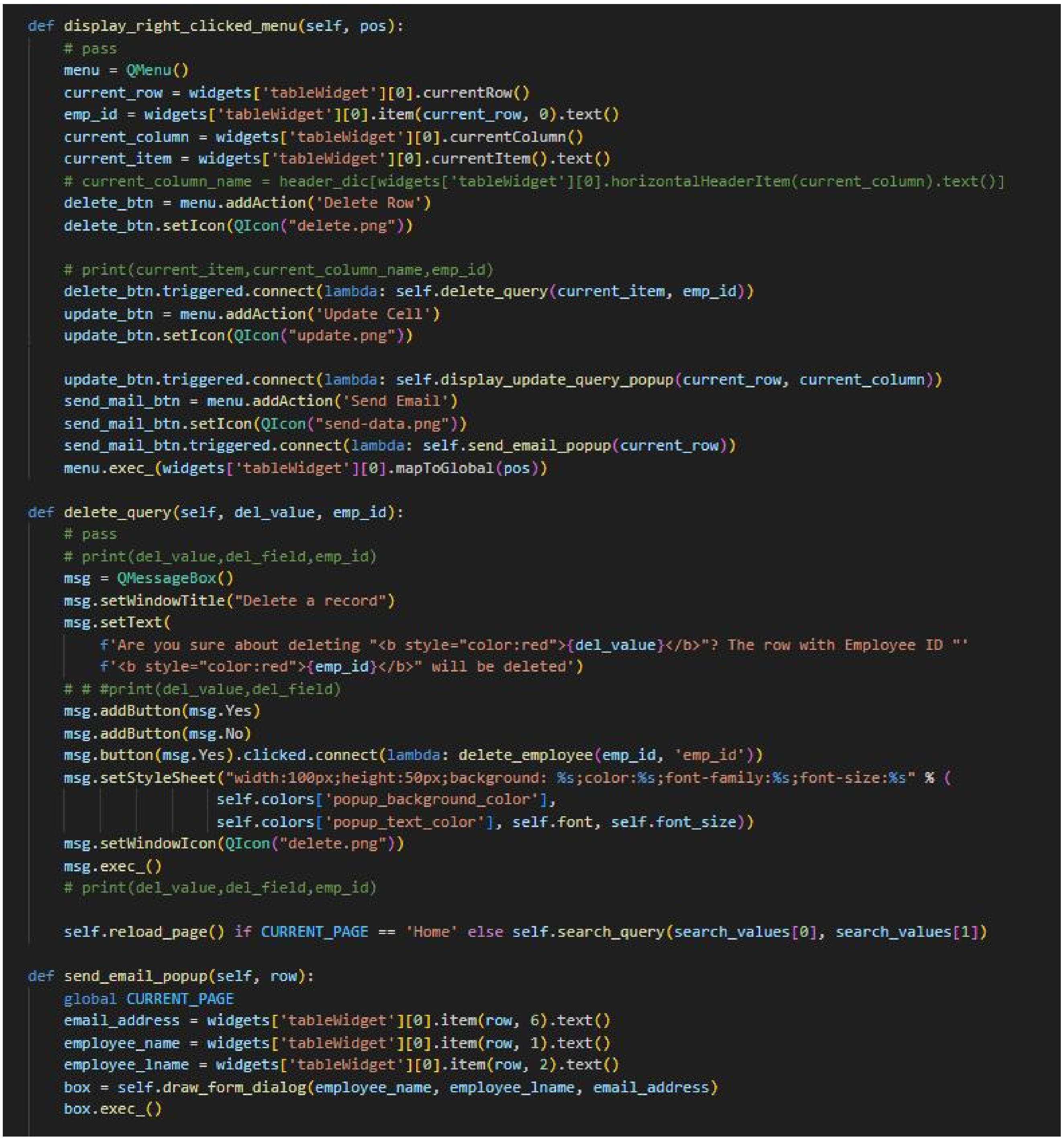
```
# Start of page view methods
def login_page(self):

    logo = self.draw_label('HT Employee Management System', width=1200, font_size=50)

    logo.setAlignment(QtCore.Qt.AlignCenter)

    widgets['logo'].append(logo)
    # button widget

    user_name_input = QLineEdit()
    user_name_input.setPlaceholderText('Username')
    user_name_input.setStyleSheet(
        "QLineEdit{" "color:%s; border: 4px solid '#008CBA';border-radius:25px;font-size:25px;padding: 25px 0;"
        "margin:20px 350px ;text-align: center;background:%s;font-family:%s;}" % (
            self.colors['button_color'], self.colors['background_color'], self.font))

    widgets['employee_fields'].append(user_name_input)
    password_input = QLineEdit()
    password_input.setPlaceholderText('Password')
    password_input.setEchoMode(QLineEdit.Password)
    password_input.setStyleSheet("*{border: 4px solid '#008CBA';" +
                                 "border-radius:25px;" +
                                 "font-size:25px;" +
                                 "color: '%s';" % (self.colors['button_color']) +
                                 "padding: 25px 0; " +
                                 "font-family:%s;" % self.font +
                                 "background:%s;" % (self.colors['background_color']) +
                                 "margin:20px 350px ;}"
                                 )
    user_name_input.setContextMenuPolicy(QtCore.Qt.PreventContextMenu)
    password_input.setContextMenuPolicy(QtCore.Qt.PreventContextMenu)
    widgets['employee_fields'].append(password_input)
    button = QPushButton("Login")
    button.setCursor(QCursor(QtCore.Qt.PointingHandCursor))

    button.setStyleSheet("*{border: 4px solid '#808000';" +
                         "font-family:%s;" % self.font +
```

1



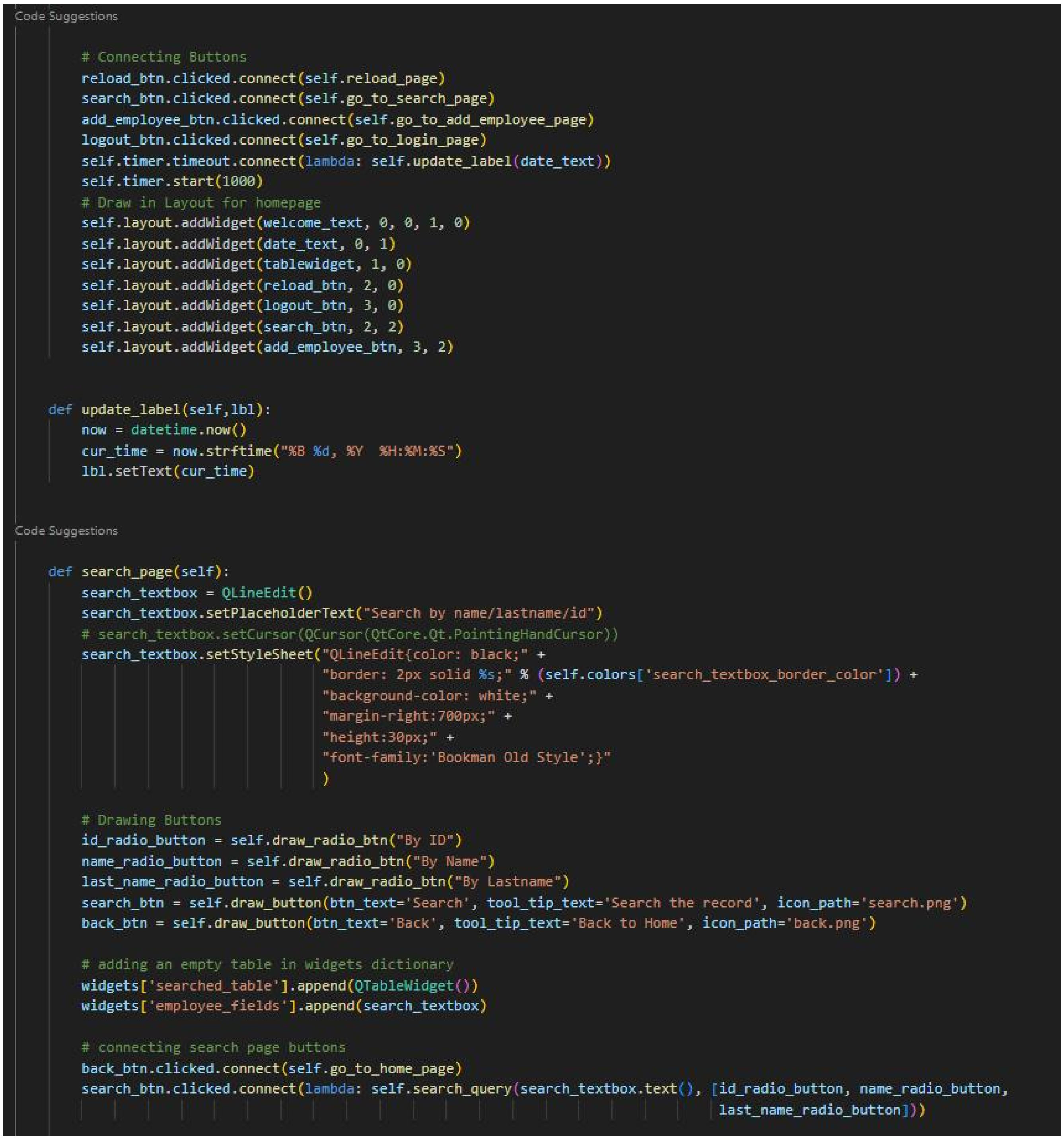
```
def draw_textbox(self, place_holder="", text=""):
    field = QLineEdit(text)
    field.setPlaceholderText(place_holder)
    field.setFixedWidth(160)

    field.setStyleSheet("QLineEdit{color: %s;" % (self.colors['textbox_color']) +
                        "border: 3px solid %s;" % (self.colors['textbox_border']) +
                        "background-color: %s;" % (self.colors['textbox_background']) +
                        "margin-right:f'{margin-right}' !important;" +
                        "height:30px;" +
                        'font-size:18px;' +
                        "font-family:'%s';}" % self.font
                        )
    widgets['employee_fields'].append(field)
    return field

def draw_label(self, label_text, width=760, height=300, font_size=25, alignment=QtCore.Qt.AlignCenter):
    label = QLabel()
    label.setFixedWidth(width)
    label.setFixedHeight(height)

    label.setText(label_text)
    label.setAlignment(alignment)
    label.setStyleSheet("font-family:'%s';" % self.font +
                        "font-size: %ipx ;" % font_size +
                        "color: %s;" % (self.colors['label_color']))
    widgets['labels'].append(label)
    return label

def draw_button(self, btn_text='test', width=165, height=55, icon_path="", tool_tip_text='test'):
    QToolTip.setFont(QFont('Bookman Old Style', 10))
    btn = QPushButton(btn_text)
    btn.setCursor(QCursor(QtCore.Qt.PointingHandCursor))
    btn.setFixedWidth(width)
    btn.setFixedHeight(height)
    btn.setStyleSheet("QPushButton{border: 2px solid '%s';" % (self.colors['button_border_color']) +
                      "border-radius:45px;" +
```

1



```python
def display_right_clicked_menu(self, pos):
    # pass
    menu = QMenu()
    current_row = widgets['tableWidget'][0].currentRow()
    emp_id = widgets['tableWidget'][0].item(current_row, 0).text()
    current_column = widgets['tableWidget'][0].currentColumn()
    current_item = widgets['tableWidget'][0].currentItem().text()
    # current_column_name = header_dic[widgets['tableWidget'][0].horizontalHeaderItem(current_column).text()]
    delete_btn = menu.addAction('Delete Row')
    delete_btn.setIcon(QIcon("delete.png"))

    # print(current_item,current_column_name,emp_id)
    delete_btn.triggered.connect(lambda: self.delete_query(current_item, emp_id))
    update_btn = menu.addAction('Update Cell')
    update_btn.setIcon(QIcon("update.png"))

    update_btn.triggered.connect(lambda: self.display_update_query_popup(current_row, current_column))
    send_mail_btn = menu.addAction('Send Email')
    send_mail_btn.setIcon(QIcon("send-data.png"))
    send_mail_btn.triggered.connect(lambda: self.send_email_popup(current_row))
    menu.exec_(widgets['tableWidget'][0].mapToGlobal(pos))

def delete_query(self, del_value, emp_id):
    # pass
    # print(del_value,del_field,emp_id)
    msg = QMessageBox()
    msg.setWindowTitle("Delete a record")
    msg.setText(
        f'Are you sure about deleting "<b style="color:red">{del_value}</b>"? The row with Employee ID "'
        f'<b style="color:red">{emp_id}</b>" will be deleted')
    # # #print(del_value,del_field)
    msg.addButton(msg.Yes)
    msg.addButton(msg.No)
    msg.button(msg.Yes).clicked.connect(lambda: delete_employee(emp_id, 'emp_id'))
    msg.setStyleSheet("width:100px;height:50px;background: %s;color:%s;font-family:%s;font-size:%s" % (
                    self.colors['popup_background_color'],
                    self.colors['popup_text_color'], self.font, self.font_size))
    msg.setWindowIcon(QIcon("delete.png"))
    msg.exec_()
    # print(del_value,del_field,emp_id)

    self.reload_page() if CURRENT_PAGE == 'Home' else self.search_query(search_values[0], search_values[1])

def send_email_popup(self, row):
    global CURRENT_PAGE
    email_address = widgets['tableWidget'][0].item(row, 6).text()
    employee_name = widgets['tableWidget'][0].item(row, 1).text()
    employee_lname = widgets['tableWidget'][0].item(row, 2).text()
    box = self.draw_form_dialog(employee_name, employee_lname, email_address)
    box.exec_()
```

1

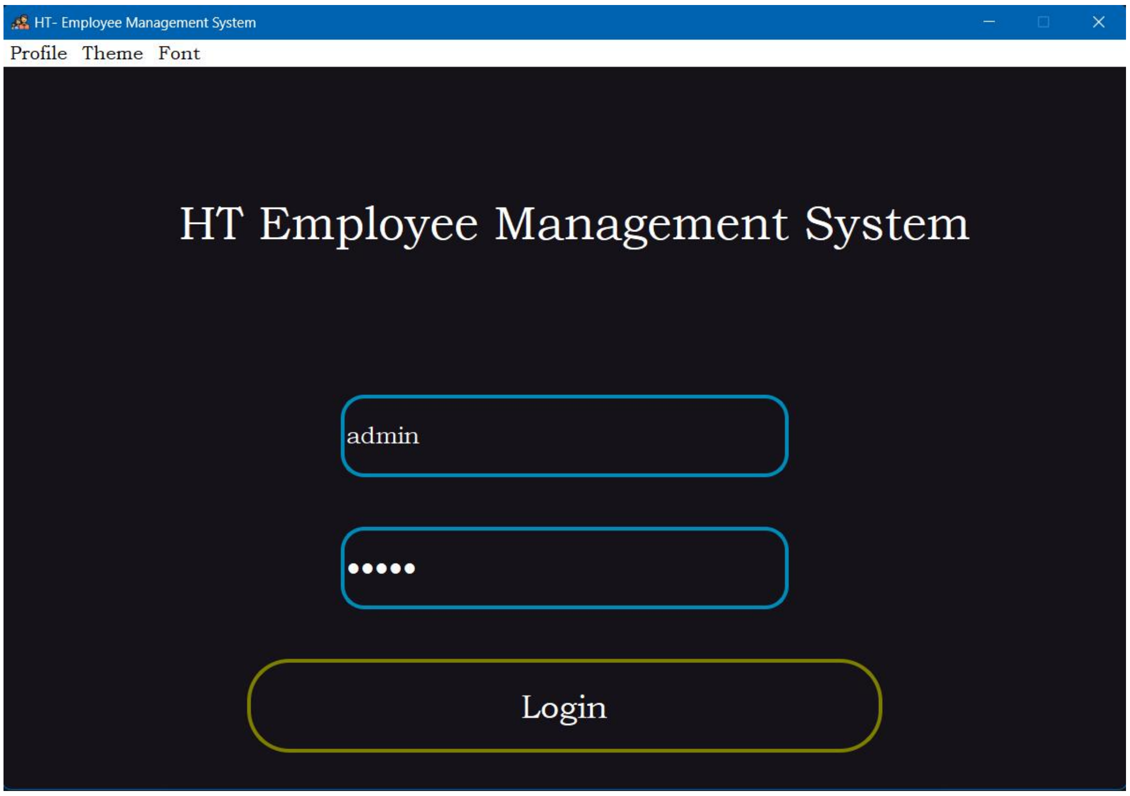


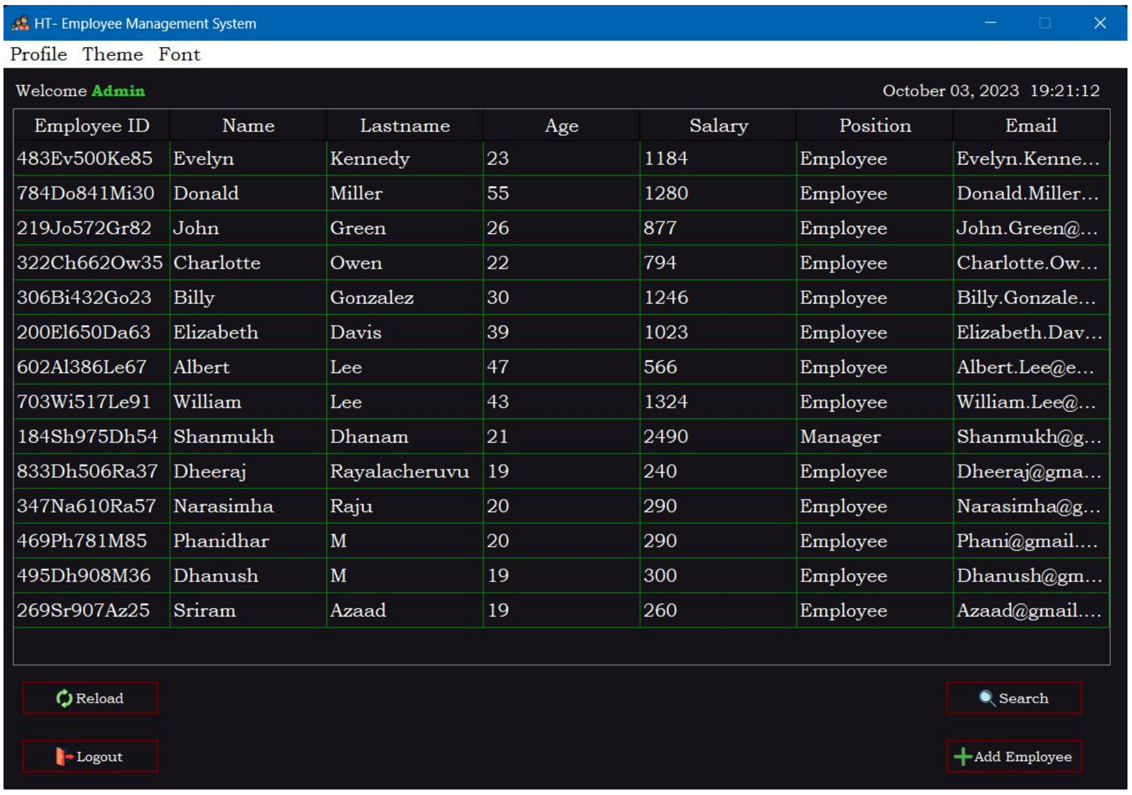
```
        # Connecting Buttons
        reload_btn.clicked.connect(self.reload_page)
        search_btn.clicked.connect(self.go_to_search_page)
        add_employee_btn.clicked.connect(self.go_to_add_employee_page)
        logout_btn.clicked.connect(self.go_to_login_page)
        self.timer.timeout.connect(lambda: self.update_label(date_text))
        self.timer.start(1000)
        # Draw in Layout for homepage
        self.layout.addWidget(welcome_text, 0, 0, 1, 0)
        self.layout.addWidget(date_text, 0, 1)
        self.layout.addWidget(tablewidget, 1, 0)
        self.layout.addWidget(reload_btn, 2, 0)
        self.layout.addWidget(logout_btn, 3, 0)
        self.layout.addWidget(search_btn, 2, 2)
        self.layout.addWidget(add_employee_btn, 3, 2)


    def update_label(self,lbl):
        now = datetime.now()
        cur_time = now.strftime("%B %d, %Y  %H:%M:%S")
        lbl.setText(cur_time)
```

```
    def search_page(self):
        search_textbox = QLineEdit()
        search_textbox.setPlaceholderText("Search by name/lastname/id")
        # search_textbox.setCursor(QCursor(QtCore.Qt.PointingHandCursor))
        search_textbox.setStyleSheet("QLineEdit{color: black;" +
                                     "border: 2px solid %s;" % (self.colors['search_textbox_border_color']) +
                                     "background-color: white;" +
                                     "margin-right:700px;" +
                                     "height:30px;" +
                                     "font-family:'Bookman Old Style';}"
                                     )

        # Drawing Buttons
        id_radio_button = self.draw_radio_btn("By ID")
        name_radio_button = self.draw_radio_btn("By Name")
        last_name_radio_button = self.draw_radio_btn("By Lastname")
        search_btn = self.draw_button(btn_text='Search', tool_tip_text='Search the record', icon_path='search.png')
        back_btn = self.draw_button(btn_text='Back', tool_tip_text='Back to Home', icon_path='back.png')

        # adding an empty table in widgets dictionary
        widgets['searched_table'].append(QTableWidget())
        widgets['employee_fields'].append(search_textbox)

        # connecting search page buttons
        back_btn.clicked.connect(self.go_to_home_page)
        search_btn.clicked.connect(lambda: self.search_query(search_textbox.text(), [id_radio_button, name_radio_button,
                                                                                      last_name_radio_button]))
```

1

B . Output Screenshots

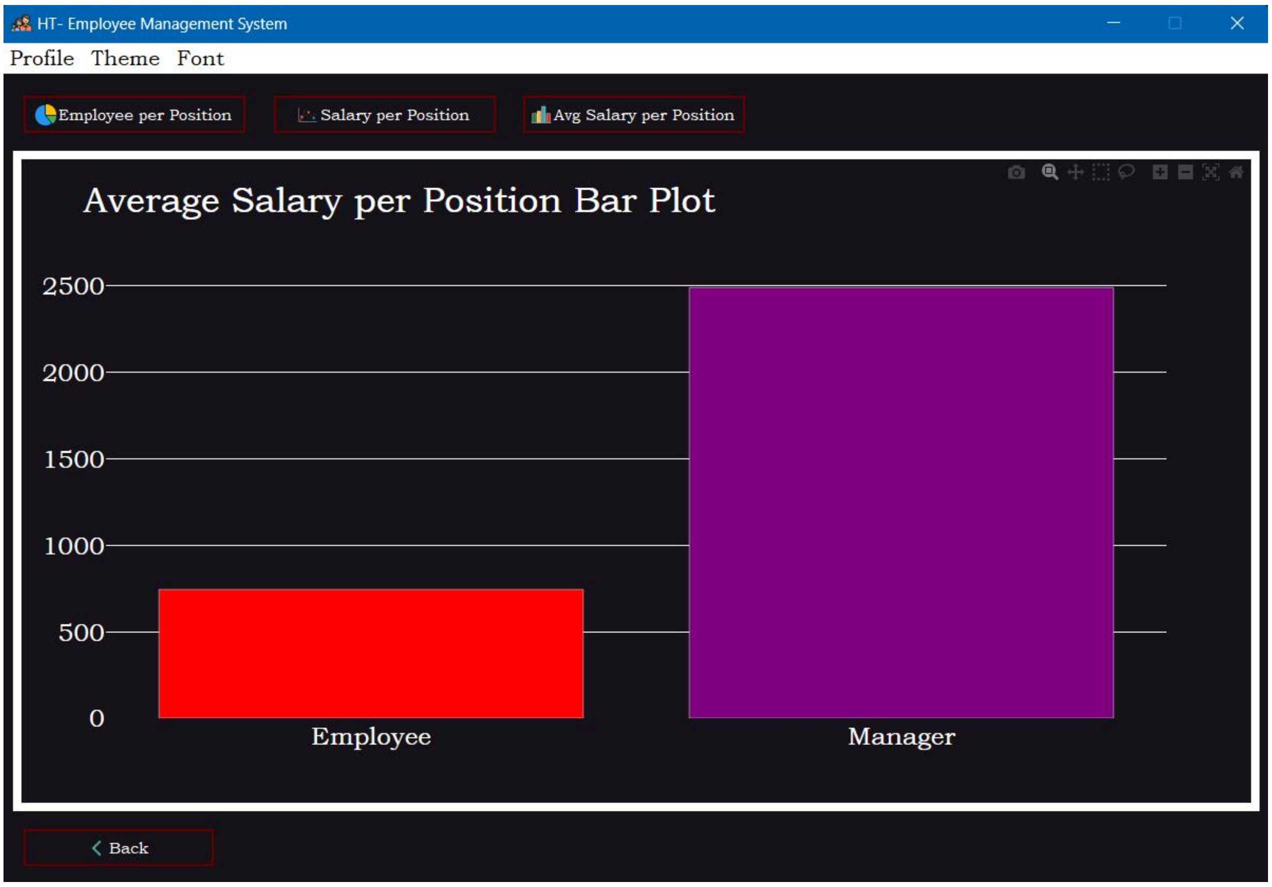


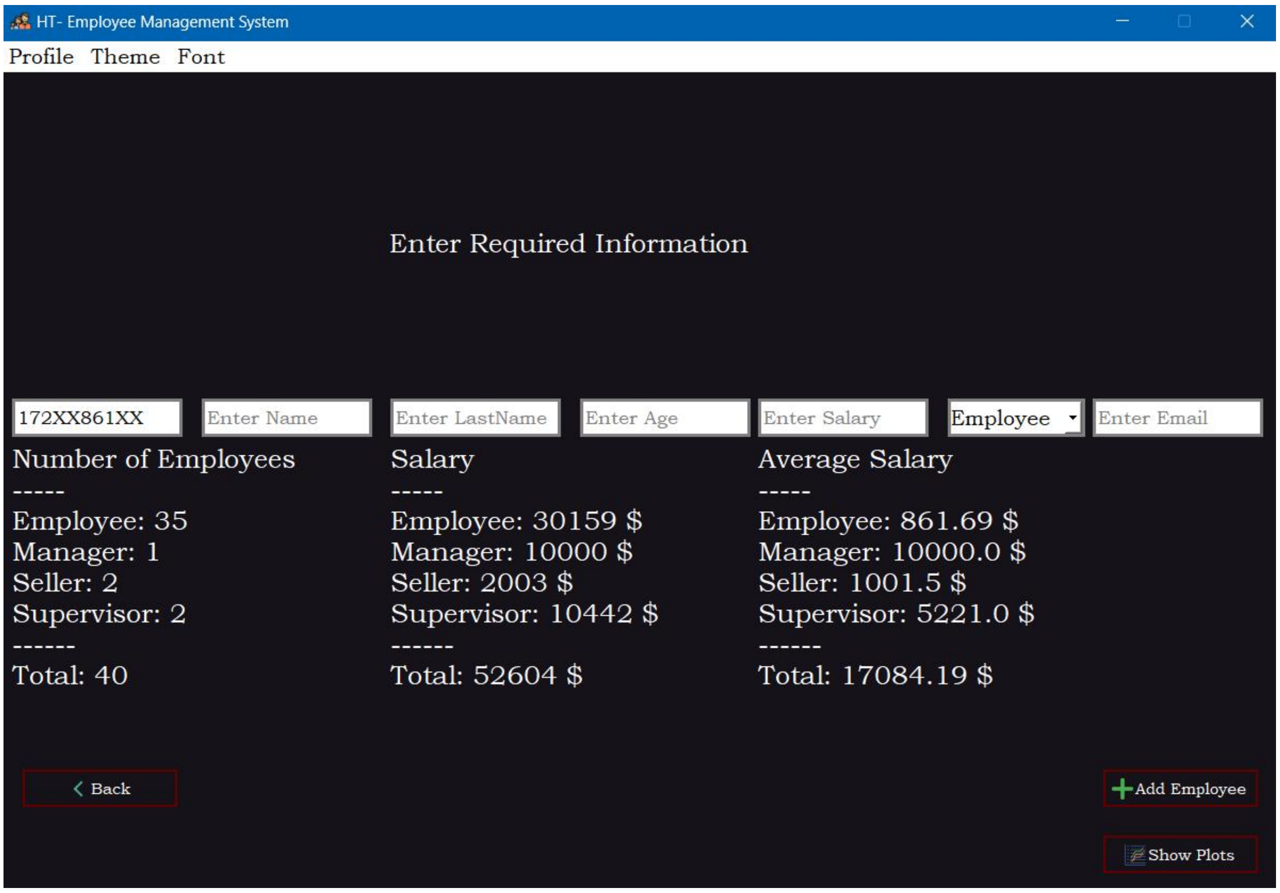


Welcome **Admin** October 03, 2023 19:21:12

Employee ID	Name	Lastname	Age	Salary	Position	Email
483Ev500Ke85	Evelyn	Kennedy	23	1184	Employee	Evelyn.Kenne…
784Do841Mi30	Donald	Miller	55	1280	Employee	Donald.Miller…
219Jo572Gr82	John	Green	26	877	Employee	John.Green@…
322Ch662Ow35	Charlotte	Owen	22	794	Employee	Charlotte.Ow…
306Bi432Go23	Billy	Gonzalez	30	1246	Employee	Billy.Gonzale…
200El650Da63	Elizabeth	Davis	39	1023	Employee	Elizabeth.Dav…
602Al386Le67	Albert	Lee	47	566	Employee	Albert.Lee@e…
703Wi517Le91	William	Lee	43	1324	Employee	William.Lee@…
184Sh975Dh54	Shanmukh	Dhanam	21	2490	Manager	Shanmukh@g…
833Dh506Ra37	Dheeraj	Rayalacheruvu	19	240	Employee	Dheeraj@gma…
347Na610Ra57	Narasimha	Raju	20	290	Employee	Narasimha@g…
469Ph781M85	Phanidhar	M	20	290	Employee	Phani@gmail.…
495Dh908M36	Dhanush	M	19	300	Employee	Dhanush@gm…
269Sr907Az25	Sriram	Azaad	19	260	Employee	Azaad@gmail.…

1





172XX861XX

Employee

```
Number of Employees   Salary                     Average Salary
-----                 -----                      -----
Employee: 35          Employee: 30159 $          Employee: 861.69 $
Manager: 1            Manager: 10000 $           Manager: 10000.0 $
Seller: 2             Seller: 2003 $             Seller: 1001.5 $
Supervisor: 2         Supervisor: 10442 $        Supervisor: 5221.0 $
------                ------                     ------
Total: 40             Total: 52604 $             Total: 17084.19 $
```

1

CHAPTER 7

REFERENCE

"Django for Beginners" by William S. Vincent

- "Django for APIs: Build web APIs with Python & Django" by William S. Vincent
- "Two Scoops of Django: Best Practices for Django 3.x" by Daniel Roy Greenfeld and Audrey Roy Greenfeld
- "Django 3 By Example" by Antonio Mele
- "Django Design Patterns and Best Practices" by Arun Ravindran
- "Test-Driven Development with Python" by Harry Percival
- "Flask Web Development: Developing Web Applications with Python" by Miguel Grinberg
- "Python Web Development with Django" by Jeff Forcier, Paul Bissex, and Wesley Chun
- "Learning Django Web Development" by Sanjeev Jaiswal
- "RESTful Web APIs" by Leonard Richardson, Mike Amundsen.

1

1