

PROJECT REPORT

AI Assist Hub

A Report Submitted to

Jawaharlal Nehru Technological University Kakinada, Kakinada

in partial fulfillment for the award of the degree of

**BACHELOR OF TECHNOLOGY
IN
COMPUTER SCIENCE AND ENGINEERING**

Submitted by

**Name : KANDALA A V N VAMSI KRISHNA SAI
Regd No : 21KN1A0582**



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

NRI INSTITUTE OF TECHNOLOGY

Autonomous

(Approved by AICTE, Permanently Affiliated to JNTUK, Kakinada)

Accredited by NBA (CSE, ECE & EEE), Accredited by NAAC with 'A' Grade

ISO 9001: 2015 Certified Institution

Pothavarappadu (V), (Via) Nunna, Agiripalli (M), Krishna Dist., PIN: 521212, A.P, India.

2022-2023



NRI INSTITUTE OF TECHNOLOGY

(An Autonomous Institution, Approved by AICTE, Permanently Affiliated to JNTUK, Kakinada)

Accredited by NBA (CSE, ECE & EEE), Accredited by NAAC with 'A' Grade
ISO 9001: 2015 Certified Institution

Pothavarappadu (V), (Via) Nunna, Agiripalli (M), Krishna Dist., PIN: 521212, A.P, India.

CERTIFICATE

This is to certify that the “**Project report**” submitted by **KANDALA A V N VAMSI KRISHNA (21KN1A0582)**, is work done by him and submitted during 2022-2023 academic year, in partial fulfillment of the requirements for the award of the degree of **BACHELOR OF TECHNOLOGY in COMPUTER SCIENCE AND ENGINEERING**.

Head of the Department

(Dr. D. SUNEETHA)

ACKNOWLEDGEMENT

We take this opportunity to thank all who have rendered their full support to our work. The pleasure, the achievement, the glory, the satisfaction, the reward, the appreciation and the construction of our project cannot be expressed with a few words for their valuable suggestions.

We are expressing our heartfelt thanks to **Head of the Department, Dr. D. SUNEETHA** garu for her continuous guidance for completion of our Project work.

We are extending our sincere thanks to **Dean of the Department, Dr. K. V. SAMBASIVA RAO** for his continuous guidance and support to complete our project successfully.

We are thankful to the **Principal, Dr. C. NAGA BHASKAR** garu for his encouragement to complete the Project work.

We are extending my sincere and honest thanks to the **Chairman, Dr. R. VENKATA RAO garu & Secretary, Sri K. Sridhar** garu for their continuous support in completing the Project work.

Finally, we thank the Administrative Officer, Staff Members, Faculty of Department of CSE, NRI Institute of Technology and my friends, directly or indirectly helped us in the completion of this project.

Name: KANDALA A V N VAMSI KRISHNA SAI Regd No: 21KN1A0582

INDEX

1. Abstract.....	1
2. Introduction.....	2
2.1 Existing SYSTEM.....	2-3
2.2 Proposed SYSTEM.....	3
2.3 Hardware and Software Requirements.....	3-4
2.4 Feasibility Study.....	4-6
3. Requirements Specification Document	
3.1 Introduction.....	6
3.2 Functional Requirements.....	6-7
3.3 Non-Functional Requirements.....	7
3.3.1 Analysis, Design & Data requirements.....	7
3.3.2 Constraints.....	7-8
3.3.3 Guidelines.....	8
3.3.4 Validation criteria.....	8
4 Architecture & Technologies	
4.1 Introduction.....	9-11
4.2 Project Plan.....	12
4.3 System Analysis.....	12-13
4.4 Client / Presentation tier.....	13
4.5 Technologies.....	13-19
5. Design	
5.1 Data Base Design Phase	
5.1.1 Gmail Registration table.....	20
5.1.2 OTP table.....	20
5.2 Coding Phase.....	20-29
5.3 Frontend Development Phase.....	29-46
5.3 Screen Shots.....	47-56
6. Testing.....	57-58
7. Conclusion.....	58
8. Bibliography.....	58

1. Abstract:

This abstract presents an overview of three Python-based automation solutions for various tasks: an Alarm System, Email Automation, and a Virtual Assistant. These solutions leverage the power of Python programming language to enhance efficiency, productivity, and convenience in daily activities.

Alarm System:

The Alarm System is a Python program designed to automate time-sensitive reminders and notifications. By utilizing the datetime library, the program allows users to set custom alarms for specific dates and times. It also includes a user-friendly interface. The Alarm System provides a reliable and flexible solution for managing tasks and important events effectively.

Email Automation:

The Email Automation system simplifies the process of sending bulk emails and streamlines communication. Built using Python's SMTP (Simple Mail Transfer Protocol) library, this program allows users to automate email campaigns, personalized email blasts, and scheduled email deliveries. With support for variables, attachments, and HTML formatting, the Email Automation system offers a versatile tool for efficient and targeted communication.

Virtual Assistant:

The Virtual Assistant is an interactive Python program that acts as a digital assistant, providing assistance with a wide range of tasks. The Virtual Assistant can understand and respond to user queries, perform web searches, retrieve information, schedule appointments, set reminders, and more. With its extensible architecture, the Virtual Assistant can be further customized to integrate with various APIs and services, making it a powerful tool for automation and personal productivity.

Main Objective:

The main objective of these Python-based automation solutions is to enhance efficiency, productivity, and convenience in daily activities. By leveraging Python's capabilities, each solution—an Alarm System, Email Automation, and a Virtual Assistant—addresses specific needs. The Alarm System provides a reliable way to manage reminders and notifications, keeping users organized and on schedule. Email Automation streamlines communication by automating bulk emails, personalized messages, and scheduled deliveries, making it easier to execute targeted campaigns. The Virtual Assistant uses natural language processing and machine learning to perform tasks such as answering queries and boosting personal productivity. Together, these solutions showcase Python's power in automating routine tasks, enhancing user convenience, and optimizing productivity in everyday life.

Using this software, we can:

- **Manage Reminders:** Set custom alarms for tasks and events using the Alarm System to ensure important deadlines are met.
- **Automate Emails:** Use the Email Automation system to streamline bulk email sending, personalize messages, and schedule deliveries efficiently.
- **Customize Integrations:** Extend the Virtual Assistant's capabilities by integrating it with various APIs and services for personalized automation.

2. Introduction:

The rapid advancement of technology has paved the way for innovative solutions to enhance productivity and convenience in our daily lives. Among these advancements is the AI Assist Hub, a collection of Python-based automation solutions that have revolutionized how we manage tasks and communication. This report introduces three core components of the AI Assist Hub: The Alarm System, Email Automation, and Virtual Assistant, each designed to leverage the power of Python programming to streamline routine activities. The Alarm System offers a user-friendly approach to managing time-sensitive reminders and notifications, ensuring that users remain organized and punctual. Email Automation simplifies the communication process by enabling the efficient sending of bulk and personalized emails, thus optimizing targeted communication strategies. Meanwhile, the Virtual Assistant acts as an intelligent digital helper, utilizing natural language processing and machine learning to assist with a wide range of tasks, from answering queries to scheduling appointments. Collectively, these solutions illustrate the versatility of Python in automating everyday tasks, enhancing user convenience, and boosting productivity.

CRUD OPERATIONS:

CREATE Operation: Performs the INSERT statement to create a new record.

READ Operation: Reads table records based on the input parameter.

UPDATE Operation: Executes an update statement on the table. It is based on the input parameter.

DELETE Operation: Deletes a specified row in the table. It is also based on the input parameter.

2.1 EXISTING SYSTEM

Alarm System:

- Relies on manual reminders via physical planners or basic calendar apps.
- Limited customization and integration options.

Email Automation:

- Involves manual composition and sending of emails.
- Lacks advanced features like scheduling and personalization without additional tools.

Virtual Assistant:

- Offers basic task management tied to specific platforms.
- Limited intelligence and adaptability for complex tasks.

Disadvantages:

Alarm System:

- Limited customization and integration.
- Manual input required.
- Basic notification features.

Email Automation:

- Manual composition and sending.
- Minimal automation and scheduling.
- Integration challenges.

Virtual Assistant:

- Platform-specific and inflexible.
- Basic intelligence for complex tasks.
- Limited customization and API integration.

2.2 PROPOSED SYSTEM:

The Proposed System: AI Assist Hub:

1. **Alarm System:**
 - Provides flexible, customizable alarms with Python, integrating with other systems for better task management.
2. **Email Automation:**
 - Automates bulk and personalized emails with scheduling, variables, and HTML support, reducing manual effort.
3. **Virtual Assistant:**
 - Uses natural language processing to perform a wide range of tasks, integrating with APIs for enhanced functionality.

The AI Assist Hub aims to streamline tasks, enhance user convenience, and boost productivity beyond traditional methods.

2.3 Hardware and Software Requirements

Hardware and software specifications

HARDWARE AND SOFTWARE INTERFACES:

HARDWARE REQUIREMENTS

Client Site:

Processor	: Intel Pentium IV
Speed	: 2.00GHZ
RAM	: 1GB
Hard Disk	: 150 GB
Key Board (104 keys)	: Standard
Screen Resolution	: 1024 x 764 Pixels

Server Site:

Processor	: Intel Pentium IV
Speed	: 2.00GHZ
RAM	: 1GB
Hard Disk	: 150 GB
Key Board (104 keys)	: Standard
Screen Resolution	: 1024 x 764 Pixels

SOFTWARE REQUIREMENTS

OPERATING SYSTEM	: WINDOWS XP AND ABOVE
DATA BASE	: MySQL - mysql-connector-java-8.0.13.jar
PROGRAMMING LANGUAGE	: Python 3
IDE	: Visual Studio Code

2.4 FEASIBILITY STUDY:

Feasibility study is an important phase in the software development process. It enables the developer to have an assessment of the product being developed. It refers to the feasibility study of the product in terms of outcomes of the product, operational use and technical support required for implementing it.

Feasibility study should be performed on the basis of various criteria and parameters. The various feasibility studies are:

- Economic Feasibility
- Operational Feasibility
- Technical Feasibility

Economic Feasibility: It refers to the benefits or outcomes we are deriving from the product compared to the total cost we are spending for developing the product. If the benefits are more or less the same as the older system, then it is not feasible to develop the product.

In the present system, the development of the new product greatly enhances the accuracy of the system and cuts short the delay in the processing of application.

The errors can be greatly reduced and at the same time providing a great level of security. Here we don't need any additional equipment except memory of required capacity. No need for spending money on client for maintenance because the database used is web enabled database.

Operational Feasibility: It refers to the feasibility of the product to be operational. Some products may work very well at design and implementation but may fail in the real time environment. It includes the study of additional human resource required and their technical expertise.

In the present system, all the operations can be performed easily compared to existing system and supports for the backlog data. Hence there is need for additional analysis. It was found that the additional modules added are isolated modules as far as the operational is concerned, so the Developed system is operationally feasible.

Technical Feasibility: It refers to whether the software that is available in the market fully supports the present application. It studies the pros and cons of using particular software for the development and its feasibility. It also studies the additional training needed to be given to the people to make the application work.

In the present system, the user interface is user friendly and does not require much expertise and training. It just needs a mouse click to do any sort of application. The software that is used for developing is server pages fully is highly suitable for the present application since the users require fast

access to the web pages and with a high degree of security. This is achieved through integration of web server and database server in the same environment.

Implementation plan:

The main plan for the system developed is to upgrading existing system to the proposed system. There are mainly 4 methods of upgrading the existing system to proposed

- Parallel Run System
- Direct Cut-Over System
- Pilot System
- Phase-in Method

Parallel Run System: It is the most secure method of converting from an existing to new system. In this approach both the systems run in parallel for a specific period of time. During that period if any serious problems were identified while using the new system, the new system is dropped and the older system is taken at the start point again.

Direct Cut -Over Method: In this approach a working version of the system is implemented in one part of the organization such as single work area or department. When the system is deemed complete, it is installed throughout the organization either all at once (direct cut-over) or gradually (phase-in).

Phase-in Method: In this method a part of the system is first implemented and over time other remaining parts are implemented.

Implementation plan used: The workflow Management system is developed on the basis of “Parallel Run Method” because we upgraded the system, which is already in use to fulfill the requirements of the client. The system already in use is treated as the old system and the new system is developed on the basis of the old system and maintained the standards processed by the older system. The upgraded system is working well and is implemented on the client successfully. of the candidate recruitment.

Project Plan

It was decided to use good Software engineering principals in the development of the system since the company had quite a big client network & was aiming to provide staffing for the clients or to develop the internal projects of the companies& expand their operations in the near future. So the following Project Plan was drawn up:

1. The Analysts will interact with the current manual system users to get the Requirements. As a part of this the Requirements Specification Document will be created.
2. The requirements Specifications document will contain the Analysis & Design of the system & ML will be used as the modeling language to express the Analysis & Design of the System. According to Grady Booch et al, in The Unified Modeling Language User Guide [UML-1998], “The Unified

Modeling Language (UML) is a graphical language for visualizing, specifying, constructing, and documenting the artifacts of a software intensive system. The UML gives you a standard way to write a system's blueprints, covering conceptual things, such as business processes and system functions, as well as concrete things, such as classes written in a specific programming language, database schemas, and reusable software components”.

3. The Analysis, Design, Implementation & testing of the System itself will be broadly based on the Rational Unified Software Development process. According to Ivar Jacobson et al, in The Unified Software Development Process (The Addison-Wesley Object Technology Series) [USDP-2000], the Unified Software Development Process contains Inception, Elaboration, Construction & Transition as the main Phases, which contain further cycles & iterations. This process will be followed to produce an incremental cycle, which will deliver milestones like the Requirements Specification Document etc., at the end of each of the iterations, Phases or cycles.

4. The Architecture & Technologies will be decided as a part of the Analysis of the requirements.

5. Once the Design is ready the Implementation & Testing strategy of the system will commence. Each will be independent of the other. The implementation of the system itself will be broken down into sub-systems following the Software Engineering principles for the development of robust software.

6. Once the implementation is ready, the System testing will take place. If the system is judged to be stable then Acceptance testing by the Users will take place & once the Users are satisfied the System will be rolled out to the Users & they will be trained on how to use it for an initial period.

The following chapters contain an account of how the Technology & architecture for the system were chosen.

3.Requirements Specification Document

3.1 Introduction

According to Roger Pressman in Software Engineering: A Practitioner's Approach (McGraw-Hill Publications) [SEPA-1997], the requirement specification document is produced at the end of Analysis of the system. This document is a very comprehensive document & contains all the User requirements & Analysis diagrams. The Requirements are broadly divided into two groups:

1. Functional requirements
2. Non-functional requirements

3.2 Functional Requirements

The main purpose of functional requirements within the requirement specification document is to define all the activities or operations that take place in the system. These are derived through interactions with the users of the system. Since the Requirements Specification is a comprehensive document & contains a lot of data, it has been broken down into different Chapters in this report. The depiction of the Design of the System in UML is presented in a separate chapter. The Data Dictionary is presented in the Appendix of the system.

1. The System should allows the administrator to manage different levels of tests and their sequence.

2. It allows the administrator to manage the questions in each category.
3. It allows the administrator to manage the list of questions in each category.
4. Candidate can register himself for writing the test.
5. The system then takes the candidate the first level after logging in.
6. The system generates the test by generating questions randomly pickup the questions from the list
7. It allows the candidate to select answers of questions
8. This system finally evaluates the test, display the result and store it.
9. This system can then takes the candidate to the next level.
10. It allows the administrator to generate the report bases on some cut off marks.
11. It allows the candidate the feedback

3.3 Non-Functional Requirements

The non-functional requirements consist of

1. Analysis, Design & Data requirements (Use-case diagrams, textual analysis, sequence diagrams, data dictionary etc.)
2. Constraints.
3. Guidelines.
4. Validation Criteria.

3.3.1 Analysis, Design & Data requirements

The use case diagrams, textual analysis and sequence diagrams & data dictionary fall into this category. Since each category above is of considerable importance, they have been dealt in separate chapters. An outline is only included here.

The Analysis & Design phases of the system yield Use Case diagrams, textual analysis, Sequence Diagrams, Class diagrams & Data Dictionary. Data dictionary consists of process statements showing how data is flowing from starting point to end point.

3.3.2 Constraints

These are the requirements that are not directly related to the functionality of the system.

These should be considered as mandatory when the system is developed. The following Constraints were arrived at for the system:

1. The system should be available over the intranet so that the Users like the candidates can use the system from their system which was assigned to him.
2. For gaining entry into the system the users should be registered and should be able use login & passwords for gaining access to the system.
3. The users should be able to change their passwords for increased security.
4. The system should conform to the requirement specified and final deliverables of the project before some date.
5. The system should be easy to understand and organized in a structured way. The users should also receive feedback about any errors that occur.
6. There should be no limitation about the hardware platform that is to be used to run the system.
7. Data integrity should be maintained if an error occurs or the whole system comes down.
8. A user should to be registered in the system once in 6 months only.
9. A user can take-up the next level test once he clears previous level.

3.3.3 Guidelines

We have discussed mandatory requirements in the previous section. The requirements in this section should be taken as suggestions & they should be thought of as recommendations to further enhance the usability of the system.

1. The system should display a menu for users to choose from.
2. The system should display users' requests in a reasonable time.
3. Services of the system should be available 24 hours a day.
4. The system should be designed in such a way that it is easy to enhance it with more functionality. It should be scalable & easily maintainable.

3.3.4 Validation Criteria

The Validation Criteria are dealt separately in the Chapter dealing with the Test Strategy & Test cases.

4. Architecture & Technologies

4.1. Introduction

General Methodology in Developing Software Project

The general methodology in developing a system is involved in different phases, which describe the system's life cycle model for developing software project. The concept includes not only forward motion but also have the possibility to return that is cycle back to an activity previously completed. This cycle back or feedback may occur as a result of the failure with the system to meet a performance objective or as a result of changes in redefinition of system activities. Like most systems that life cycle of the computer-based system also exhibits distinct phases.

Those are,

Requirement Analysis Phase

Design Phase

Development Phase

Coding Phase

Testing Phase

1. Requirement Analysis Phase:

This phase includes the identification of the problem, in order to identify the problem; we have to know information about the problem, the purpose of the evaluation for problem to be known. We have to clearly know about the client's requirements and the objectives of the project.

2. Design Phase:

Software design is a process through which the requirements are translated into a representation of software. One of the software requirements have been analyzed and specified, the software design involves three technical activities: design, coding generation and testing. The design of the system is in modular form i.e. the software is logically partitioned into components that perform specific functions and sub functions. The design phase leads to modules that exhibit independent functional characteristics. It even leads to interfaces that reduce the complexity of the connections between modules and with the external environment. The design phase is of main importance because in this activity, decisions ultimately affect the success of software implementation and maintenance.

3. Development Phase:

The development phase includes choosing of suitable software to solve the particular problem given. The various facilities and the sophistication in the selected software give a better development of the problem.

4. Coding Phase:

The coding phase is for translating the design of the system-produced during the design phase into code in a given programming language, which can be executed by a computer and which performs the computation specified by the design.

5. Testing Phase:

Testing is done in various ways such as testing the algorithm, programming code; sample data debugging is also one of following the above testing.

Requirement Specification:

Here, the focus is on specifying what has been found giving analysis such as representation, specification languages and tools, and checking the specification are addressed during this activity.

The Requirement phase terminates with the production of the validate SRS document. Producing the SRS document is the basic goal of this phase.

The purpose of the Software Requirement Specification is to reduce the communication gap between the clients and the developers. Software Requirement Specification is the medium through which the client and user needs are accurately specified. It forms the basis of software development. A good SRS should satisfy all the parties involved in the system.

Purpose:

The purpose of this document is to describe all external requirements or client provisioning. It also describes the interfaces for the system.

Scope:

This document is the only one that describes the requirements of the system. It is meant for the use by the developers, and will also use by the basis for validating the final delivered system. Any changes made to the requirements in the future will have to go through a formal change approval process. The developer is responsible for asking for clarifications, where necessary, and will not make any alternations without the permission of the client.

System Design:

Design:

Design of software involves conceiving, planning out and specifying the externally observable characteristics of the software product. We have data design, architectural design and user interface design in the design process. These are explained in the following section. The goal of design process is to provide a blue print for implementation, testing and maintenance activities.

The primary activity during data design is to select logical representations of data objects identified during requirement analysis and software analysis. A data dictionary explicitly represents the relationships among data objects and constraints on the elements of the data structure. A data dictionary should be established and used to define both data and program design.

Design process is in between the analysis and implementation process. The following design diagrams (Data Flow Diagrams and E-R Diagrams) make it easy to understand and implement .

The design process for software system has two levels.

1. System Design or Top-Level Design.
2. Detailed Design or Logical Design.

System Design or Top-Level Design:

In the system design the focus is on deciding which modules are needed for the system, the specification of these modules and how these modules should be interconnected.

Detailed Design or Logical Design:

In detailed design the interconnection of the modules or how the specifications of the modules can be satisfied is decided. Some properties for a software system design are

- Verifiability.
- Completeness.
- Consistency.
- Trace ability.
- Simplicity/Understandability.

The Requirements provided by the users are converted into Users Requirement Specifications described above. The URS documents are then revised, validated, authorized and approved by the users. The development commences after the approval phase i.e. after the signing off of the URS documents. Thus, the URS is concerned to be the most important document from user and developer prospective. The Developer will try to adhere to the requirements specified in the URS documents in order to develop the required application. We have used Agile models a development model.

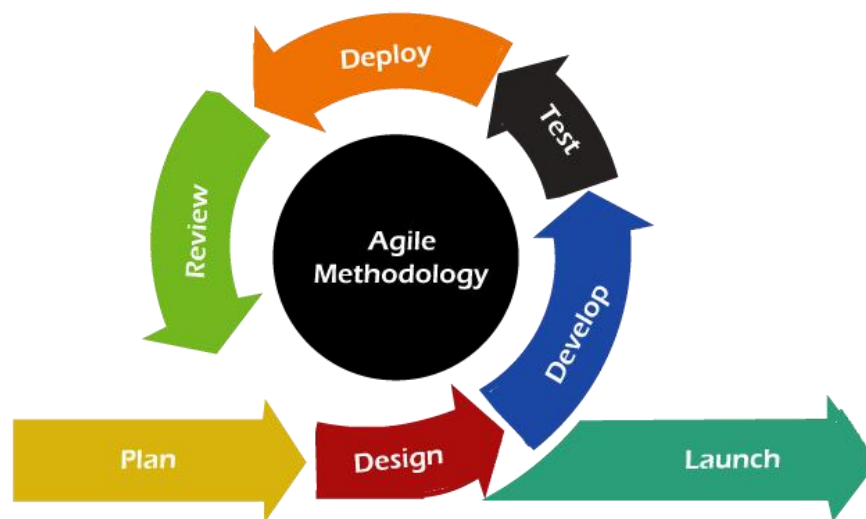


Fig. 2.1 Agile Methodology

4.2. Project Plan

Areas, which would be considered during the planning and analysis, would be:

- Ensure that the information flow is process driven.
- Reduce the manual effort so the maximum extent for all activities.
- Ensure validation at each and every level.
- Act as an effective tool in decision support.
- Provide user friendly system.

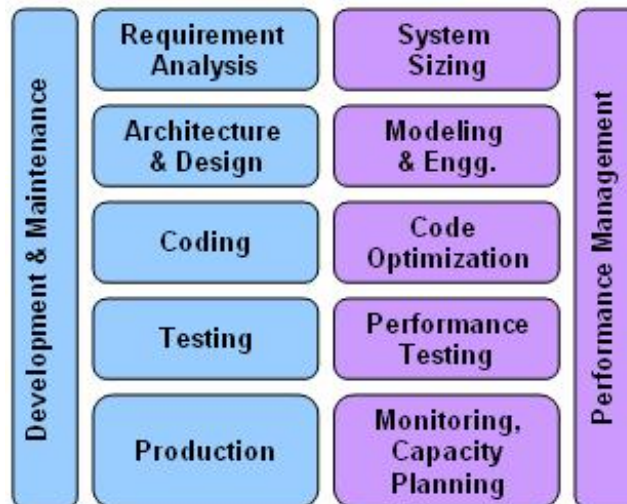


Fig. 2.2 Project Planning and Management Approach

4.3 SYSTEM ANALYSIS:

Analysis is the detailed study of the various operations performed by the system and their relationships within and outside of the system. A key question is: what must be done to solve the problem? One aspect of analysis is defining the boundaries of the system and determining whether or not candidate system should consider other related systems. During analysis, data are collected on the available files, decision points, and transactions handled by the present system.

System Requirement Specification

What is SRS?

Software Requirement Specification (SRS) is the starting point of the software developing activity. As system grew more complex it became evident that the goal of the entire system cannot be easily comprehended. Hence the need for requirement phase arose. The software project is initiated by the client needs. The SRS is the means of translating the ideas of the minds of the clients (the input) into a formal document (the output of the requirement phase).

The SRS phase consists of two basic activities:

- 1) Problem/Requirement Analysis: The process is order and more nebulous of the two, deals with understanding the problem, the goal and constraints.

- 2) Requirement Specification: Here, the focus is on specifying what has been found giving analysis such as representation, specification languages and tools, and checking the specifications are addressed during this activity.

The Requirement phase terminates with the production of the validate SRS document. Producing the SRS document is the basic goal of this phase.

Role of SRS:

The purpose of the Software Requirement Specification is to reduce the communication gap between the clients and the developers. Software Requirement Specification is the medium through which the client and the user needs are accurately specified. It forms the basis of the software development. A good SRS should satisfy all the parties involved in the system.

4.4. CLIENT SERVER MODEL

When an architect designs a building, he has a vision of the finished product and produces a result based on that vision. Client – server, on the other hand, is more like Darwinian model of evolution of a living species. No one has a vision of the finished products; rather, day-today events and gradual changes affect it over time in reaction to those events.

In the beginning, application was fairly simple, reading input transaction in a ‘batch’, processing them against a data store, and the output was paper. Record retrieval was usually a set of subroutines embedded in the updating program.

Common functions gradually migrated from the application to the operating system. Database processing was one of the first major functions to be removed from application control. Much of the time database functions in the application included retrieval, replacement and insertion. Since it was function had to be introduced database administration. This new function was separated from the application code and involved defining the structure of the database, value ranges backup, rollback, and so forth.

Advantages of Client – Server Model:

- The hardware and software can be placed where it will do the best.
- In Client – Server model PCs, the power can be spread across the client and the server.
- On client side, an Active X object is used to present data
- By having the client side, it can do more work
- The client software supplies the interface (Such as windowed program) and the knowledge of how to pass the request to the server and the format of the data for the user when it's returned from the server. The server's job is to manipulate the data according to the user's request.

4.5 TECHNOLOGIES

- IDE - Visual Studio Code
- Python 3
- Oracle-SQL

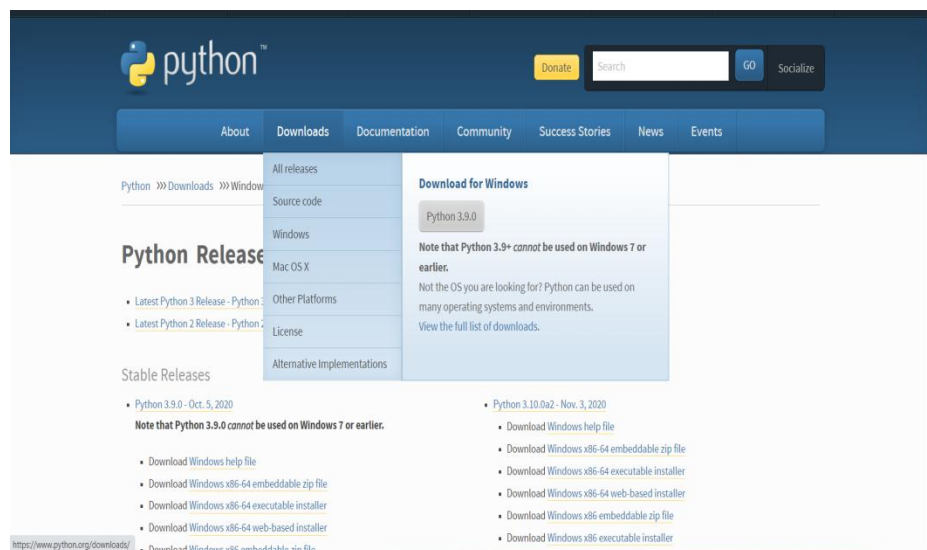
Python:

Python is a computer programming language often used to build websites and software, automate tasks, and conduct data analysis. Python is a general-purpose language, meaning it can be used to create a variety of different programs and isn't specialized for any specific problems.

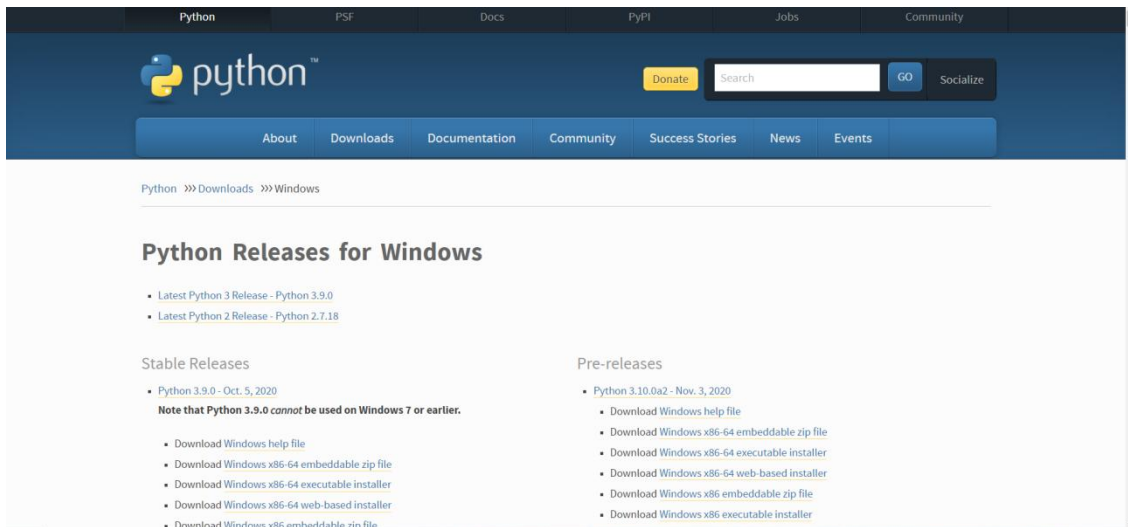


Python Installation and Setting Up Environment:

- Go to the www.python.org and click on the Downloads.
- Select the OS according to your Pc.



- On the next page you will see all the version of python that are available for download.
- Python latest Version is 3.9.0 . Click on the link, It will start the downloading process.



- After downloading, click on run. It will open the installation window.
- Check the box Add Python 3.9 to PATH and then click on install.



Python Flask:

Flask is a lightweight Python web framework that provides useful tools and features for creating web applications in the Python Language. It gives developers flexibility and is an accessible framework for new developers because you can build a web application quickly using only a single Python file.

Oracle Database 11g Express Edition

Free to develop, deploy, and distribute

Oracle Database 11g Express Edition (Oracle Database XE) is an entry-level, small-footprint database based on the Oracle Database 11g Release 2 code base. It's free to develop, deploy, and distribute; fast to download; and simple to administer.

Oracle Database XE is a great starter database for:

- **Developers** working on Node.js, Python, PHP, Java, .NET, XML, and Open Source applications.
- **DBAs** who need a free, starter database for training and deployment.
- **Independent Software Vendors (ISVs) and hardware vendors** who want a starter database to distribute free of charge.
- **Educational institutions and students** who need a free database for their curriculum With Oracle Database XE, you can now develop and deploy applications with a powerful, proven, industry-leading infrastructure, and then upgrade when necessary without costly and complex migrations. Oracle Database XE can be installed on any size host machine with any number of CPUs (one database per machine), but XE will store up to 11GB of user data, use up to 1GB of memory, and use one CPU on the host machine.

Oracle Database 11g Express Edition Quick Tour

The new Express Edition reflects essential updates to the Oracle Database code base since 10.2.0.1, and thus contains an avalanche of value for developers and DBAs.

Published September 2011

In this article you will learn about top new features introduced in Oracle 11G Express Edition (XE). This latest version of Oracle's free database offering packages the essential updates to Enterprise Edition from 10.2.0.1 and through 11.2.0.2. (That tells a lot about the magnitude of changes: four release cycles of 10g and four of 11g.) From relatively simple improvements like the PIVOT operator, deferred segment creation, and virtual columns to the ground breaking adaptive cursor sharing, database resident connection pooling, and edition-based redefinition features, this new release sets a new standard for "express" RDBMSs.

With respect to growing needs for storage, Oracle Database 11g XE now offers a full 11GB for user data alone, which is almost threefold increase since previous 4GB limit in Oracle Database 10g XE. Other hardware restrictions remain the same with CPU usage capped at one physical core and memory at 1GB. Oracle Database 11g XE is available immediately for Windows and Linux platforms.

Below find a quick tour of the Oracle Database 11g Release 2 functionality that is now made available gratis via Express Edition.

Install and Upgrade

Upgrading to this new version of XE is very simple compared to traditional methods like Database Upgrade Assistant (DBUA) or manual upgrade: The entire process comprises getting a dump from your existing database, uninstalling the previous release, installing the new one, and importing the dump. (Note: Oracle Application Express applications go through a separate path and are excluded from the full dump; the provided *gen_inst.sql* script takes care of that.)

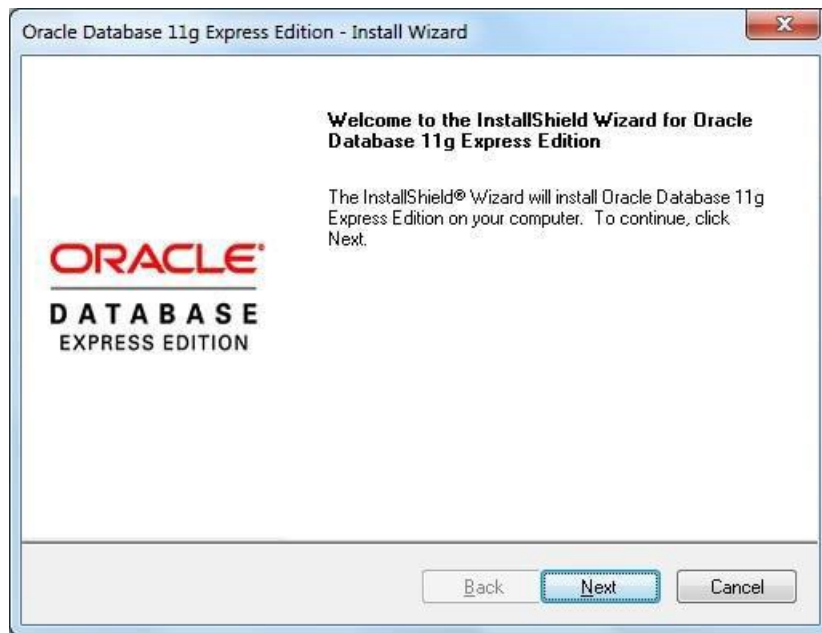


Figure 1 Installation screen

The installation procedure allows for either regular deployments or silent installs so the process can be easily scripted for mass deployment. For more information, refer to Oracle 11g express edition installation guide which explains the exact steps required to perform the upgrade.



Application Development

Oracle Application Express (APEX) remains one of the most actively developed Oracle Database features, with two important milestones reached since the 2.1 release that was included with Oracle Database 10g XE. First, Oracle APEX 3.0 brought PDF printing, Access migration, page caching, and a number of builder enhancements. Then, release 4.0, which is bundled with 11g XE in the form of Oracle APEX 4.0.2, introduced Web sheets - a unique technology for effortless content management where users take control of both content and structure of exposed data. Other improvements include native dynamic actions, a plug-ins framework, team features for streamlining the application development process, RESTful Web Services, a J2EE APEX listener, and a revamped application builder. (The most current release at the time of this writing, version 4.1, delivers improved error handling, ability to use ROWIDs instead of primary keys in forms, spreadsheet upload capabilities, and even more refinements to the builder.)

Check out the Oracle APEX change logs 2.2, 3.0, 3.2, 4.0 and 4.1 to learn more details about these releases.



HTML:

- HTML stands for Hyper Text Markup Language
- HTML is the standard markup language for creating Web pages
- HTML describes the structure of a Web page
- HTML consists of a series of elements
- HTML elements tell the browser how to display the content
- HTML elements label pieces of content such as "this is a heading", "this is a paragraph", "this is a link", etc.

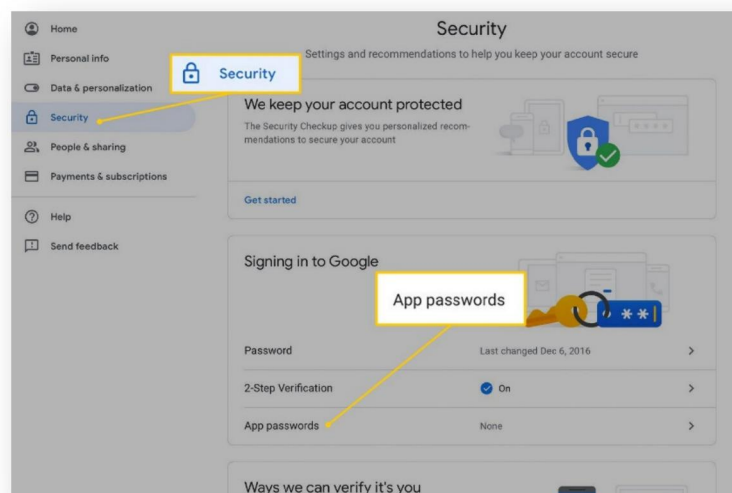
What is a Gmail App Password?

A Gmail App Password is a unique, 16-character password generated specifically for applications or devices that need to access your Gmail account. This password is used instead of your regular Gmail password to enhance security, especially for third-party applications that do not support two-factor authentication.

Creating a Gmail App Password:

To use email automation and send personalized emails, follow these steps to create an App Password:

1. **Login to Your Gmail Account:**
 - Access your Gmail account and go to the "Manage your Google Account" tab.
2. **Navigate to App Passwords:**
 - In the account settings, select "Security," then find and click on "App Passwords."



3. **Generate the App Password:**

- Choose the device or application for which you are creating the password and click "Generate."

Google Account

← App passwords

App passwords let you sign in to your Google Account from apps on devices that don't support 2-Step Verification. You'll only need to enter it once so you don't need to remember it. [Learn more](#)

You don't have any app passwords.

Select the app and device you want to generate the app password for.

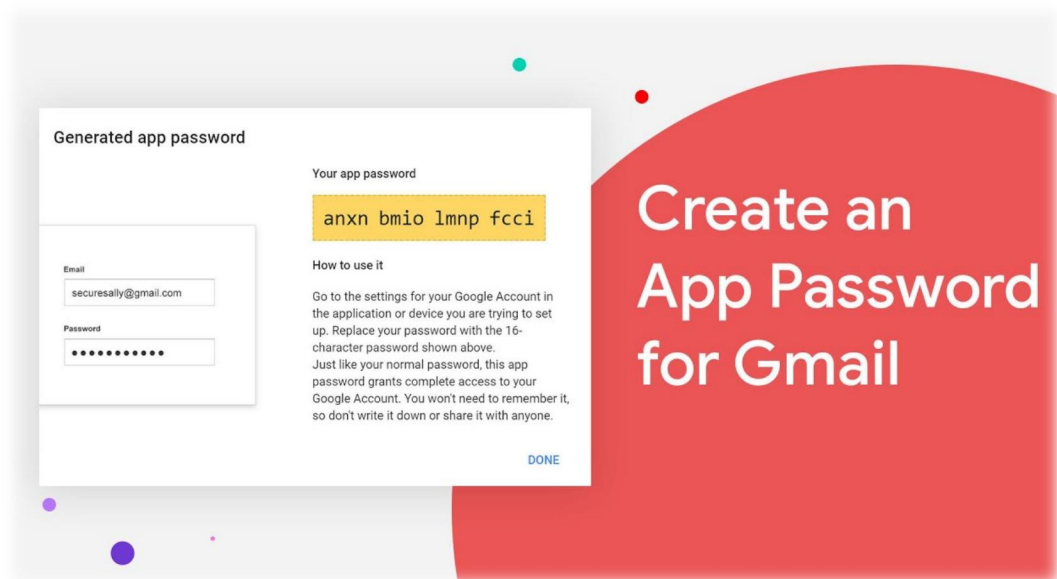
Mail Windows Computer

GENERATE

Privacy Terms Help About

4. **Store the Password Securely:**

- A 16-character App Password will be generated. Save it securely, as it will be used to access your Gmail account for email automation purposes.



5.Designing:

5.1: Data Base Design

The database consists of 2 main tables namely Gmail Registration, Otp. These 2 tables are used in the email automation and each of their structure is given below:

5.1.1 Gmail Registration table:

It consists of all the details that the user should provide.

```
SQL> desc greg;
Name                                         Null?    Type
-----
USERNAME                                     VARCHAR2(50)
APPCODE                                     VARCHAR2(30)
PASSWORD                                     VARCHAR2(30)
```

5.1.1 Gmail Registration table

5.1.2 Otp table:

It consists of all the details that the Otp while doing Otp Generation and Otp Verification in the Email Automation.

```
SQL> desc otp;
Name                                         Null?    Type
-----
OTP                                          VARCHAR2(10)
SNO                                          VARCHAR2(3)
```

5.1.2 Otp table

5.2: Coding:

5.2.1: Importing the required modules and establishing connection with database.

```
from flask import Flask, request, redirect,render_template,url_for,flash,session
from playsound import playsound
import datetime,time
import cx_Oracle
import smtplib
import random
import math
from email.mime.multipart import MIMEMultipart
from email.mime.text import MIMEText
from email.mime.base import MIMEBase
from email import encoders
from gtts import gTTS
import speech_recognition as sr
```



```

from time import ctime
import uuid
import webbrowser
import os,re #to read files from your local system
import secret_Key

con = cx_Oracle.connect('root/password@localhost:1521/xs')
cursor=con.cursor()

app = Flask(__name__)
app.secret_key = secret_Key

```

5.2.2: Code for the routing of Home page.

```

@app.route('/')
def home():
    return render_template('Home.html')

@app.route('/topbar')
def topbar():
    return render_template('topbar.html')

@app.route('/leftmenu')
def leftmenu():
    return render_template('leftmenu.html')

@app.route('/centerbox')
def centerbox():
    return render_template('centerbox.html')

@app.route('/rightmenu')
def rightmenu():
    return render_template('rightmenu.html')

@app.route('/sendemail')
def sendemail():
    return render_template('sendemail.html')

```

5.2.3: Code for the Alarm.

```

@app.route('/alarm',methods=['GET','POST'])
def alarm():
    if request.method == 'POST':
        alarm_time = request.form['alarm_time']
        alarm_message = request.form['alarm_message']
        # convert alarm time to datetime object
        alarm_datetime = datetime.datetime.strptime(alarm_time, '%Y-%m-%dT%H:%M')

```

```

# calculate time difference between now and alarm time
time_diff = (alarm_datetime - datetime.datetime.now()).total_seconds()
# play sound after time difference elapses
while True:
    #print(time_diff)
    time.sleep(time_diff)
    playsound('Alarm Clock Alarm.mp3')
    return render_template('index.html', success_message='Alarm set for {}
with message " {}".format(alarm_time, alarm_message))

```

5.2.4: Code for the Gmail Registration.

```

@app.route('/gmailregistration',methods=['GET','POST'])
def gmailregistration():
    if request.method == 'POST':
        uname=request.form['username']
        uappcode=request.form['AppPasscode']
        upass=request.form['Password']
        result=cursor.execute("insert into greg
values(:s,:s,:s)",(uname,uappcode,upass))
        con.commit()
        return redirect(url_for('gmaillogin'))
    return render_template('gmailformregistration.html')

```

5.2.5: Code for the Gmail Login.

```

@app.route('/gmaillogin',methods=['GET','POST'])
def gmaillogin():
    if request.method == 'POST':
        un=request.form['username']
        up=request.form['Password']
        result=cursor.execute("select * from greg where username=:s and
password=:s",(un,up))
        record=result.fetchone()
        if record:
            session['loggedin']=True
            session['username']=record[0]
            return redirect(url_for('sendemail'))
        con.commit()
    return render_template('gmaillogin.html')

```

5.2.6: Code for the Mail with OTP Generation.

```

@app.route('/mailwithotpgeneration',methods=['GET','POST'])
def mailwithotpgeneration():
    if request.method == 'POST':
        Sender=request.form['sender']

```

```

password=request.form['password']
Receiver=request.form['receiver']
result=cursor.execute("select * from greg where username=:s and
password=:s" ,(Sender,password))
record=result.fetchone()
appcode=record[1]
print=(appcode)
if record:
    sender = Sender #give your email id
    receiver = Receiver #give receiver gmailid or take user input
    subject="Otp Generation"
    msg = MIMEMultipart()
    msg['From'] = sender
    msg['To'] = receiver
    msg['Subject'] = subject
    text = msg.as_string()
    #Email Automation using Python -->smtpplib,email packages
    #smtpplib -->Simple Mail Transfer Protocol

    #connect to the server
    server = smtpplib.SMTP('smtp.gmail.com',587) #give gmail outgoing smtp
address
    #along with TLS port number
    #start the server
    server.starttls()
    #login to gmail
    server.login(Sender,appcode) #give your gmail address and
    #gmail app passcode

    #we can include random number to it
    msg = "OTP is "+str(random.randint(1000,9999))
    receiver_mail = Receiver
    #send the mail
    server.sendmail("gmailid",receiver_mail,msg)
    return render_template('sendemail.html',message='Mail Sent!!!!')

```

5.2.7: Code For the Mail with OTP Generation and Validation.

```

@app.route('/mailwithotpverification',methods=['GET','POST'])
def mailwithotpverification():
    if request.method == 'POST':
        Sender=request.form['sender']
        Password=request.form['password']
        Receiver=request.form['receiver']
        result=cursor.execute("select * from greg where username=:s and
password=:s" ,(Sender>Password))
        record=result.fetchone()
        appcode=record[1]
        if record:

```

```

sender = Sender #give your email id
receiver = Receiver #give receiver gmailid or take user input
msg = MIMEMultipart()
msg['From'] = sender
msg['To'] = receiver
msg['Subject'] = 'Otp Generation and Verification'
#OTP Verification to Email
#OTP Creation -->math module and random module
#we will generate 6digit otp using base digits
digits = "0123456789"
OTP=''
#now we will use math module along with module to generate a customized
#6 digit otp
for i in range(6):
    OTP = OTP + digits[math.floor(random.random()*10)]
#print=(OTP)
message = OTP + " is your OTP"
#print=(msg)

#link the above otp to our email and then we will go for verification
s = smtplib.SMTP('smtp.gmail.com', 587)
s.starttls()
s.login(Sender,appcode) #gmail and app passcode
user=Sender
receiver = Receiver
s.sendmail(user,receiver,message)
aa=1
otdt=cursor.execute("update oTp set oTp=:s where sno=:s",(OTP,aa))
con.commit()

if s.sendmail:
    return redirect(url_for('otpverification'))
return render_template('mailwithotpverification.html')

```

```

@app.route('/otpverification',methods=['GET','POST'])
def otpverification():
    if request.method == 'POST':
        oTp=request.form['otp']
        result=cursor.execute("select * from oTp ")
        record=result.fetchone()
        ab=record[0]
        con.commit()
        if (oTp==ab):
            return render_template('sendemail.html',message='Valid OTP')
        else:
            return render_template('otpverification.html',message='Invalid OTP')
    return render_template('otpverification.html')

```

5.2.8: Code for the Mail with Subject.

```
@app.route('/mailwithsubject',methods=['GET','POST'])
def mailwithsubject():
    if request.method == 'POST':
        Sender=request.form['sender']
        Password=request.form['password']
        Receiver=request.form['receiver']
        Subject=request.form['subject']
        Body=request.form['body']
        result=cursor.execute("select * from greg where username=:s and
password=:s" ,(Sender>Password))
        record=result.fetchone()
        appcode=record[1]
        if record:
            sender = Sender #give your email id
            receiver = Receiver #give receiver gmailid or take user input
            subject = Subject #give your own subject
            msg = MIMEMultipart()
            msg['From'] = sender
            msg['To'] = receiver
            msg['Subject'] = subject

            body = Body
            msg.attach(MIMEText(body,'plain'))
            text = msg.as_string() #include this as final msg in sendmail

            #same include your base email code -->smtpplib
            server= smtplib.SMTP('smtp.gmail.com',587)
            #login to the gmail
            server.starttls()
            server.login(sender,appcode) #give your app passcode
            server.sendmail(sender,receiver,text)
            if server.sendmail:
                return render_template('sendemail.html',message='Mail Sent !!')
        return render_template('mailwithsubject.html')
```

5.2.9: Code for the Mail with Attachment.

```
@app.route('/mailwithattach',methods=['GET','POST'])
def mailwithattach():
    if request.method == 'POST':
        Sender=request.form['sender']
        Password=request.form['password']
        Receiver=request.form['receiver']
        Subject=request.form['subject']
        Body=request.form['body']
        attach=request.files['myfile']
        result=cursor.execute("select * from greg where username=:s and
password=:s" ,(Sender>Password))
```

```

record=result.fetchone()
appcode=record[1]
print=(attach)
if record:
    sender = Sender #give your email id
    receiver = Receiver #give receiver gmailid or take user input
    subject = Subject #give your own subject
    msg = MIMEMultipart()
    msg['From'] = sender
    msg['To'] = receiver
    msg['Subject'] = subject

    body = Body
    filename=attach.filename
    attach.save(filename)
    msg.attach(MIMEText(body))

    #we will make our attachment to be encoded and added to our mail
    part = MIMEBase('application', 'octet-stream')
    part.set_payload(open(filename,'rb').read())
    encoders.encode_base64(part)
    part.add_header('Content-Disposition', f'attachment; filename= {filename}')
    msg.attach(part)
    text = msg.as_string() #include this as final msg in sendmail

    #same include your base email code -->smtplib
    server= smtplib.SMTP('smtp.gmail.com',587)
    server.starttls()
    #login to the gmail
    server.login(sender,appcode) #give your app passcode
    server.sendmail(sender,receiver,text)
    if server.sendmail:
        return render_template('sendemail.html',message="Mail Sent !!")

return render_template('mailwithattach.html')

```

5.2.10: Code for the Virtual Assistant.

```

@app.route('/virtualassistant')
def virtualassistant():
    while True:
        global print
        def listen():
            """Listening function to respond what we speak"""
            r = sr.Recognizer()
            with sr.Microphone() as source:
                print("Start talking now...") #own statement
                audio = r.listen(source,phrase_time_limit=10) #time limit is user
            interest
                data="" #whatever we speak it will be stored in it

```

```

try:
    data = r.recognize_google(audio,language='en-US')
    print("You said:"+data)
except sr.UnknownValueError:
    print("I cannot hear you speak louder")
except sr.RequestError as e:
    print("Microphone is not working")
return data
#tts = gTTS(text=data)
#tts.save("audio.mp3")
#playsound.playsound("audio.mp3")
#listen()

#another function to respond back
def respond(speech):
    """Function for responding back"""
    tts = gTTS(text=speech)
    tts.save("Speech.mp3")
    filename = "Speech%s.mp3"%str(uuid.uuid4())
    tts.save(filename)
    playsound(filename)
    os.remove(filename)

#we will give actionsto our virtual assistant

def virtual_asstnt(data):
    if "how are you" in data:
        listening = True
        respond("I am good hope you are doing well")
        data = listen()
        listening = virtual_asstnt(data)
    elif "name" in data:
        listening = True
        respond("My name is Tessa. I am developed by Mr.Vamsi Krishna")
        data = listen()
        listening = virtual_asstnt(data)
    elif "time" in data:
        listening = True
        respond(ctime())
        data = listen()
        listening = virtual_asstnt(data)
    elif "i love you" in data:
        listening = True
        respond("Awww !! So Sweet to listen that you love mee!! I love You
Too!!")
        data = listen()
        listening = virtual_asstnt(data)
    elif "open google" in data.lower():
        listening = True
        url = "https://www.google.com/"

```

```

        webbrowser.open(url)
        respond("Success")
        data = listen()
        listening = virtual_asstnt(data)
    elif "locate" in data:
        listening = True
        webbrowser.open('https://www.google.com/maps/search/'+
                        data.replace("locate",""))
        result = "Located"
        respond("Located {}".format(data.replace(
            "locate","")))
        data = listen()
        listening = virtual_asstnt(data)
    elif "play music" in data:
        listening = True
        respond("Playing")
        playsound("Apna Bana Le Piya - Bhediya - Lofi ! Hindi.mp3")
        while playsound:
            data = listen()
            listening = virtual_asstnt(data)
    elif "word" in data.casefold(): #same you can link for other shortcuts
        listening = True
        os.startfile(r'C:/Program Files/Microsoft
Office/root/Office16/WINWORD.EXE')
        respond("Success")
        data = listen()
        listening = virtual_asstnt(data)
    elif "stop talking" in data:
        listening = False
        respond("Okay bye take care")
    try:
        return listening
    except:
        print("Timed out")
    respond("Hey!!! Good to talk with again. How can I Help you") #frst greeting
    listening=True
    while listening==True:
        data = listen()
        listening = virtual_asstnt(data)
    return render_template('virtualassistant.html')

```

5.2.11: Code for to Reset the App Password.

```

@app.route('/resetappcode',methods=['GET','POST'])
def resetappcode():
    if request.method == 'POST':
        uname=request.form['username']
        uappcode=request.form['AppPasscode']
        upass=request.form['Password']

```



```

        result=cursor.execute("update greg set appcode=:s where username=:s and
password=:s",(uappcode,uname,upass))
        con.commit()
        return redirect(url_for('gmaillogin'))

    return render_template('resetappcode.html')

```

5.3: Frontend Development:

5.3.1: Code for Home Page

HTML:

```

<html>
<head>
<frameset rows=20,80 frameborder="no">
<frameset cols=*>
<frame src="{{url_for('topbar')}}" name=a noresize>
</frameset>
<frameset cols=20,60,20>
<frame src="{{url_for('leftmenu')}}" name=1 noresize>
<frame src="{{url_for('centerbox')}}" name=2 noresize>
<frame src="{{url_for('rightmenu')}}" name=3 noresize>
</frameset>

</head>

</html>

```

5.3.2: Code for the Top Bar

CSS

```

body {
    background-image: linear-gradient(to right, #8360c3, #2ebf91);
}

h1{
    text-align: center;
    margin-top: 50px;
}

```

HTML

```

<html>
<head>
    <link rel="stylesheet" href="{{url_for('static',filename='css/topbar.css')}}">
</head>
<body>

```

```
<h1>The 3 Projects of AI using Python</h1>
</body>
</html>
```

5.3.3: Code for the Left Menu

CSS

```
.leftmenu{
    margin:20px 20px;
    padding:100px;
    background-image: radial-gradient(circle farthest-side, #fceabb, #f8b500);
    width:50px;
    height:auto;
    border-radius:10px;
    right:20px;
    justify-content: center;
}

button{
left:25px;
border-radius:5px;
width:150px;
height:50px;
background-color:rgb(69, 54, 169);
color:white;
margin:10px -45px;
}

body {
    background-image: linear-gradient(to right, #8360c3, #2ebf91);
}

button:hover{
background-color:lightpink;
}

.admin_login {
    background-color : red;
    color: white;
    padding: 10px 25px;
    border-radius: 4px;
    border-color: red;
    font-size: 20px;
    font-family: cursive;
    width:200px;
    height:50px;
    margin:10px 35px;
}
```

```
#admin_login_position {
    position: fixed;
    bottom: 10px;
    left: 20px;
}
.admin_login:hover {
    background-color: brown;
    color: white;
}
```

HTML

```
<html>
<head>
    <link rel="stylesheet" href="{{url_for('static',filename='css/leftmenu.css')}}">

</head>
<body>
<div class="leftmenu" >
    <a href="{{url_for('alarm')}}>target="2"><button id="alarm" class="button" > Alarm</button></a>

</div>

</body>
</html>
```

5.3.4: Code for the Right Menu

CSS

```
.rightmenu{
    margin:20px 20px;
    padding:100px;
    background-image: radial-gradient(circle farthest-side, #fceabb, #f8b500);
    width:50px;
    height:auto;
    border-radius:10px;
    right:20px;
    justify-content: center;
}

button{
```

```

left:25px;
border-radius:5px;
width:170px;
height:50px;
background-color:rgb(69, 54, 169);
color:white;
margin:10px -55px;
}
body {
    background-image: linear-gradient(to right, #8360c3, #2ebf91);
}

button:hover{
background-color:lightpink;
}

```

HTML

```

<html>
<head>
    <link rel="stylesheet" href="{{url_for('static',filename='css/rightmenu.css')}}">
</head>
<body>
<div class="rightmenu" >
    <a href="{{url_for('gmaillogin')}}" target="2"><button id="SendEmail"
class="button" >Send Email</button></a>
    <a href="{{url_for('virtualassistant')}}" target="2"><button id="Virtaulassistant"
class="button" >Virtual Assistant</button></a>

</div>

</body>
</html>

```

5.3.5: Code for the Centre Box

HTML

```

<html>
<head>
<style>
body{
    background-image:linear-gradient(to right,#0f0c29, #302b63, #24243e);
}
</style>

```

```
</head>
<body>
</body>
</html>
```

5.3.6: Code for the Alarm

CSS

```
.submitbutton {
    background-color :navy;
    color: white;
    padding: 10px 25px;
    border-radius: 4px;
    border-color:lightskyblue;
    font-size: 20px;
    font-family: cursive;
}
.submitbutton:hover {
    background-color:lightgreen;
    color: white;
}
.addstudent{
    visibility:visible;
    background-image: linear-gradient( 109.6deg, rgba(156,252,248,1) 11.2%,
    rgba(110,123,251,1) 91.1% );
    margin:30px auto;
    padding: 50px;
    color:black;
    font-size: 15px;
    width: 300px;
    height:auto;
    border-radius: 20px;
}
body{
    background-image:linear-gradient(to right,#0f0c29, #302b63, #24243e);
    margin:160px 190px;
}

.audio_div{
    display:none;
}
```

HTML

```
<html>
<head>
    <link rel="stylesheet" href="{url_for('static',filename='css/alarm.css')}">
</head>
```

```

<body>

  <div class="addstudent" id="formdiv">
    {% if success_message %}
    <div>{{ success_message }}</div>
    {% endif %}

    <form method="post">
      <label for="alarm_time">Alarm Time:</label>
      <p><input type="datetime-local" name="alarm_time"></p>
      <label for="alarm_message">Alarm Message:</label>
      <p><input type="text" name="alarm_message"></p>
      <p><input type="submit" value="Set Alarm"></p>
    </form>

  </div>
  <script>
    {% if success_message %}
    alert('{{ success_message }}');
    {% endif %}

  </script>
</body>

</html>

```

5.3.7: Code for the Send Email

HTML

```

<html>
  <head>
    <style>
      body{
        background-image:linear-gradient(to right,#0f0c29, #302b63, #24243e);

      }
      .lo{
        background-image: linear-gradient( 109.6deg, rgba(156,252,248,1) 11.2%,
        rgba(110,123,251,1) 91.1% );
        margin:30px auto;
        padding: 50px;
        color:white;
        font-size: 15px;
        width: 300px;
        height:300px;
      }
    </style>
  </head>
  <body>
    <div class="sendemail" id="formdiv">
      <div>{{ success_message }}</div>
      <form method="post">
        <label for="email">Email:</label>
        <p><input type="text" name="email"></p>
        <label for="password">Password:</label>
        <p><input type="password" name="password"></p>
        <p><input type="submit" value="Send Email"></p>
      </form>
    </div>
  </body>
</html>

```

```

        border-radius: 20px;
        padding:90px;
    }
    button{
        background-color:burlywood;
        color:orangered;
        width:200px;
        height:60px;
        font-family: cursive;
    }

</style>
</head>
<body>
    <div class="lo" id="divbox">
        <a href="{{url_for('mailwithotpgeneration')}}" target="2"><button>Mail
with OTP Generation</button></a><br><br>
        <a href="{{url_for('mailwithotpverification')}}" target="2"><button>Mail
with OTP Verification</button></a><br><br>
        <a href="{{url_for('mailwithattach')}}" target="2"><button>Mail with
Attach</button></a><br><br>
        <a href="{{url_for('mailwithsubject')}}" target="2"><button>Mail with
Subject</button></a><br><br>

    </div>
    <script>
        {% if message %}
        alert('{{ message }}');
        {% endif %}
        {% if msg %}
        alert('{{ msg }}');
        {% endif %}
    </script>
</body>
</html>

```

5.3.8: Code for the Gmail Registration

HTML

```

<html>
    <head>
        <title>
            LOGIN
        </title>
        <style>
            body{
                background-repeat: no-repeat;
                background-attachment: fixed;
                background-size: cover;
            }
        </style>
    </head>
    <body>

```

```

        background-image:linear-gradient(to right,#0f0c29, #302b63, #24243e);
        width:890;
        height:500;
    }
    .lo{
        background-image: linear-gradient( 109.6deg, rgba(156,252,248,1) 11.2%,
        rgba(110,123,251,1) 91.1% );
        margin:30px auto;
        padding: 50px;
        color:black;
        font-size: 15px;
        width: 300px;
        height:auto;
        border-radius: 20px;
    }
    #center{
        margin: 20px auto;
    }
    #green{
        color:green;
        font-size: 10px;
    }
    #blue{
        color:blue;
        font-size: 10px;
    }
    h1{
        color: green;
        text-align: center;
    }
</style>
</head>
<body>
    <div class="lo" id="divbox">
        <h1>Registration</h1>
        <form id="center" method="post" action="/gmailregistration">
            <label>Email</label>
            <p><input type="text" placeholder="Enter the gmail" id=Emailid
name="username"></p>
            <label>AppPasscode</label>
            <p><input type="password" placeholder="Enter the AppPasscode"
id="Passcode" name="AppPasscode"></p>
            <label>Set Password</label>
            <p><input type="password" placeholder="Enter the password"
id="Password" name="Password"></p>

            <p><input type="submit" id="SubmitButton"></p>
        </form>
    </div>
</body>
</html>

```


5.3.9: Code for the Gmail Login.

HTML

```
<html>  
  <head>  
    <title>  
      LOGIN  
    </title>  
    <style>  
      body{  
        background-image:url('coverpage.jpg');  
        background-repeat: no-repeat;  
        background-attachment: fixed;  
        background-size: cover;  
        background-image:linear-gradient(to right,#0f0c29, #302b63, #24243e);  
        width:900;  
        height:500;  
      }  
      .lo{  
        background-image: linear-gradient( 109.6deg, rgba(156,252,248,1) 11.2%  
rgba(110,123,251,1) 91.1% );  
        margin:30px auto;  
        padding: 50px;  
        color:black;  
        font-size: 15px;  
        width: 300px;  
        height:auto;  
        border-radius: 20px;  
      }  
      #center{  
        margin: 20px auto;  
      }  
      #green{  
        color:green;  
        font-size: 10px;  
      }  
      #blue{  
        color:blue;  
        font-size: 10px;  
      }  
    </style>  
  </head>  
</html>
```

```

        h1{
            color: green;
            text-align: center;
        }
    </style>
</head>
<body>
    <div class="lo" id="divbox">
        <h1>Login</h1>
        <form id="center" method="post" action="/gmaillogin">
            <label>Email</label>
            <p><input type="text" placeholder="Enter the gmail" id=Emailid
name="username"></p>
            <label>Password</label>
            <p><input type="password" placeholder="Enter the password"
id="Password" name="Password"></p>

            <p><input type="submit" id="SubmitButton"
value="Submit">&nbsp;&nbsp;&nbsp;<a href="{{url_for('resetappcode')}}" target="2">Reset
AppCode!</a></p><br>
            Account not registered ?<a href="{{url_for('gmailregistration')}}"
target="2">Register!!</a>

        </form>
    </div>

</body>
</html>

```

5.3.10: Code for the Mail with Otp Generation.

HTML

```

<html>
    <head>
        <style>
            body{
                background-image:linear-gradient(to right,#0f0c29, #302b63, #24243e);
            }
            .lo{
                background-image: linear-gradient( 109.6deg, rgba(156,252,248,1) 11.2%,
                rgba(110,123,251,1) 91.1% );
                margin:30px auto;
                padding: 50px;
                color:black;
                font-size: 15px;
                width: 200px;
                height:auto;
            }
        </style>
    </head>
    <body>
        <div class="lo" id="divbox">
            <h1>Login</h1>
            <form id="center" method="post" action="/gmaillogin">
                <label>Email</label>
                <p><input type="text" placeholder="Enter the gmail" id=Emailid
name="username"></p>
                <label>Password</label>
                <p><input type="password" placeholder="Enter the password"
id="Password" name="Password"></p>

                <p><input type="submit" id="SubmitButton"
value="Submit">&nbsp;&nbsp;&nbsp;<a href="{{url_for('resetappcode')}}" target="2">Reset
AppCode!</a></p><br>
                Account not registered ?<a href="{{url_for('gmailregistration')}}"
target="2">Register!!</a>

            </form>
        </div>

    </body>
</html>

```

```

        border-radius: 20px;
        padding:40px;
    }
    button{
        background-color:palegreen;
        color:orangered;
        width:130px;
        height:50px;
        font-family: cursive;
    }
    h1{
        color: green;
        text-align: center;
    }

</style>
</head>
<body>
    <div class="lo">
        <h1>Mail With OTP Generation</h1>
        <form action="/mailwithotpgeneration" method="POST">
            <label>From</label>
            <p><input type="text" width="200px" height="50px" placeholder="Enter
sender Gmail" name="sender"></p>
            <label>Sender Password</label>
            <p><input type="password" width="200px" height="50px"
placeholder="Enter sender Password" name="password"></p>
            <label>To</label>
            <p><input type="text" width="200px" height="50px" placeholder="Enter
Receiver Gmail" name="receiver"></p>
            <button type="submit">Generate OTP</button>
        </form>
    </div>
    <script>
        {% if message %}
        alert('{{ message }}');
        {% endif %}
        {% if msg %}
        alert('{{ msg }}');
        {% endif %}

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

```

5.3.11: Code for the Mail With OTP Generation and OTP Verification.

HTML

```
<html>
  <head>
    <style>
      body{
        background-image:linear-gradient(to right,#0f0c29, #302b63, #24243e);

      }
      .lo{
        background-image: linear-gradient( 109.6deg, rgba(156,252,248,1) 11.2%,
        rgba(110,123,251,1) 91.1% );
        margin:30px auto;
        padding: 50px;
        color:black;
        font-size: 15px;
        width: 250px;
        height:auto;
        border-radius: 20px;
        padding:80px;
      }
      .bu{
        background-color:palegreen;
        color:orangered;
        width:130px;
        height:50px;
        font-family: cursive;
        position:fixed;
        left:590px;

      }
      .bul{
        background-color:palegreen;
        color:orangered;
        width:130px;
        height:50px;
        font-family: cursive;
        right:590px;
        position:fixed;
      }
      h1{
        color: green;
        text-align: center;
      }

    </style>
  </head>
  <body>
```

```

<div class="lo">
  <h1>Mail With OTP Verification</h1>
  <form action="/mailwithotpverification" method="POST">
    <label>From</label>
    <p><input type="text" width="200px" height="50px" placeholder="Enter
sender Gmail" name="sender"></p>
    <label>Sender Password</label>
    <p><input type="password" width="200px" height="50px"
placeholder="Enter sender Password" name="password"></p>
    <label>To</label>
    <p><input type="text" width="200px" height="50px" placeholder="Enter
Receiver Gmail" name="receiver"></p>
    <button class="bu" type="submit">Generate OTP</button>

  </form>
</div>
<script>
  {% if message %}
  alert('{{ message }}');
  {% endif %}
  {% if msg %}
  alert('{{ msg }}');
  {% endif %}

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

```

Otpverification.html

```

<html>
  <head>
    <style>
      body{
        background-image:linear-gradient(to right,#0f0c29, #302b63, #24243e);

      }
      .lo{
        background-image: linear-gradient( 109.6deg, rgba(156,252,248,1) 11.2%,
        rgba(110,123,251,1) 91.1% );
        margin:30px auto;
        padding: 50px;
        color:black;
        font-size: 15px;
        width: 250px;
        height: 80px;
        border-radius: 20px;
        padding:80px;

      }
      button{

```

```

        background-color:palegreen;
        color:orangered;
        width:130px;
        height:50px;
        font-family: cursive;

    }
</style>
</head>
<body>
    <div class="lo">
        <form action="/otpverification" method="POST">
            <label>Enter the OTP</label>
            <p><input type="text" width="200px" height="50px" placeholder="Enter
The OTP" name="otp"></p>
            <button type="submit">Verify!!!</button>
        </form>
    </div>

    <script>
        {% if message %}
        alert('{{ message }}');
        {% endif %}
    </script>
</body>
</html>

```

5.3.12: Code for the Mail with Subject.

HTML

```

<html>
    <head>
        <style>
            body{
                background-image:linear-gradient(to right,#0f0c29, #302b63, #24243e);

            }
            .lo{
                background-image: linear-gradient( 109.6deg, rgba(156,252,248,1) 11.2%,
                rgba(110,123,251,1) 91.1% );
                margin:30px auto;
                color:black;
                font-size: 15px;
                width:auto;
                height:auto;
                border-radius: 20px;
                padding:25px;

            }

```

```

        button{
            background-color:palegreen;
            color:orangered;
            width:130px;
            height:50px;
            font-family: cursive;
            position:absolute;
            right:300px;
            bottom:20px;
        }
        h1{
            color: green;
            text-align: center;
        }

    </style>
</head>
<body>
    <div class="lo">
        <h1>Mail with Subject</h1>
        <form action="/mailwithsubject" method="POST">
            <label>From</label>
            <p><input type="text" width="200px" height="50px" placeholder="Enter
sender Gmail" name="sender"></p>
            <label>Sender Password</label>
            <p><input type="password" width="200px" height="50px"
placeholder="Enter sender Password" name="password"></p>
            <label>To</label>
            <p><input type="text" width="200px" height="50px" placeholder="Enter
Receiver Gmail" name="receiver"></p>
            <label>Subject</label>
            <p><textarea rows="2" cols="50" placeholder="Enter The Subject of
Mail" name="subject"></textarea></p>
            <label>Body</label>
            <p><textarea rows="15" cols="50" placeholder="Enter The Text"
name="body"></textarea></p>

            <button type="submit">Send Mail !!</button>
        </form>
    </div>
    <script>
        {% if message %}
        alert('{{ message }}');
        {% endif %}
        {% if msg %}
        alert('{{ msg }}');
        {% endif %}

    </script>
</body>

```

```
</html>
```

5.3.13: Code for the Mail with Attachment.

HTML

```
<html>
  <head>
    <style>
      body{
        background-image:linear-gradient(to right,#0f0c29, #302b63, #24243e);

      }
      .lo{
        background-image: linear-gradient( 109.6deg, rgba(156,252,248,1) 11.2%,
        rgba(110,123,251,1) 91.1% );
        margin:30px auto;
        color:black;
        font-size: 15px;
        width: auto;
        height:auto;
        border-radius: 20px;
        padding:35px;
      }
      button{
        background-color:palegreen;
        color:orangered;
        width:130px;
        height:50px;
        font-family: cursive;
        position:absolute;
        right:300px;
        bottom:20px;
      }
      .o{
        width:130px;
        height:50px;

        left:300px;
        bottom:25px;
      }
      h1{
        color: green;
        text-align: center;
      }

    </style>
  </head>
  <body>
```



```

<div class="lo">
    <h1>Mail with Attachment</h1>
    <form action="/mailwithattach" method="POST" enctype="multipart/form-
data">
        <label>From</label>
        <p><input type="text" width="200px" height="50px" placeholder="Enter
sender Gmail" name="sender"></p>
        <label>Sender Password</label>
        <p><input type="password" width="200px" height="50px"
placeholder="Enter sender Password" name="password"></p>
        <label>To</label>
        <p><input type="text" width="200px" height="50px" placeholder="Enter
Receiver Gmail" name="receiver"></p>
        <label>Subject</label>
        <p><textarea rows="2" cols="50" placeholder="Enter The Subject of
Mail" name="subject"></textarea></p>
        <label>Body</label>
        <p><textarea rows="15" cols="50" placeholder="Enter The Text"
name="body"></textarea></p>
        <label>Choose file to attach</label>
        <p><input class="o" type="file" id="myfile" name="myfile"
accept="file_extension|audio/*|video/*|image/*|media_type"> multiple></p>

        <button type="submit">Send Mail !!</button>
    </form>
</div>
<script>
    {% if message %}
    alert('{{ message }}');
    {% endif %}
    {% if msg %}
    alert('{{ msg }}');
    {% endif %}

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

```

5.3.14: Code for the Virtual Assistant.

HTML

```

<html>
    <head>
        <style>
            body{
                background-image:linear-gradient(to right,#0f0c29, #302b63, #24243e);
            }
            .lo{

```

```

        background-color:darkgoldenrod;
        width: 800px;
        height: 500px;
        color:black;
        border-radius: 5px;
        top:100px;
        left:50px;
        margin: 30px auto;

    }
    .a{
        color:black;
        border-radius: 5px;
        margin:50px auto;
        background-color: aqua;
        text-align: center;
        width: 700px;
        height:300px;
    }
</style>

</head>
<body>
    <form>
        <div class="lo">
             Virtual Assistant

            <div class="a" id="textdiv">Hii

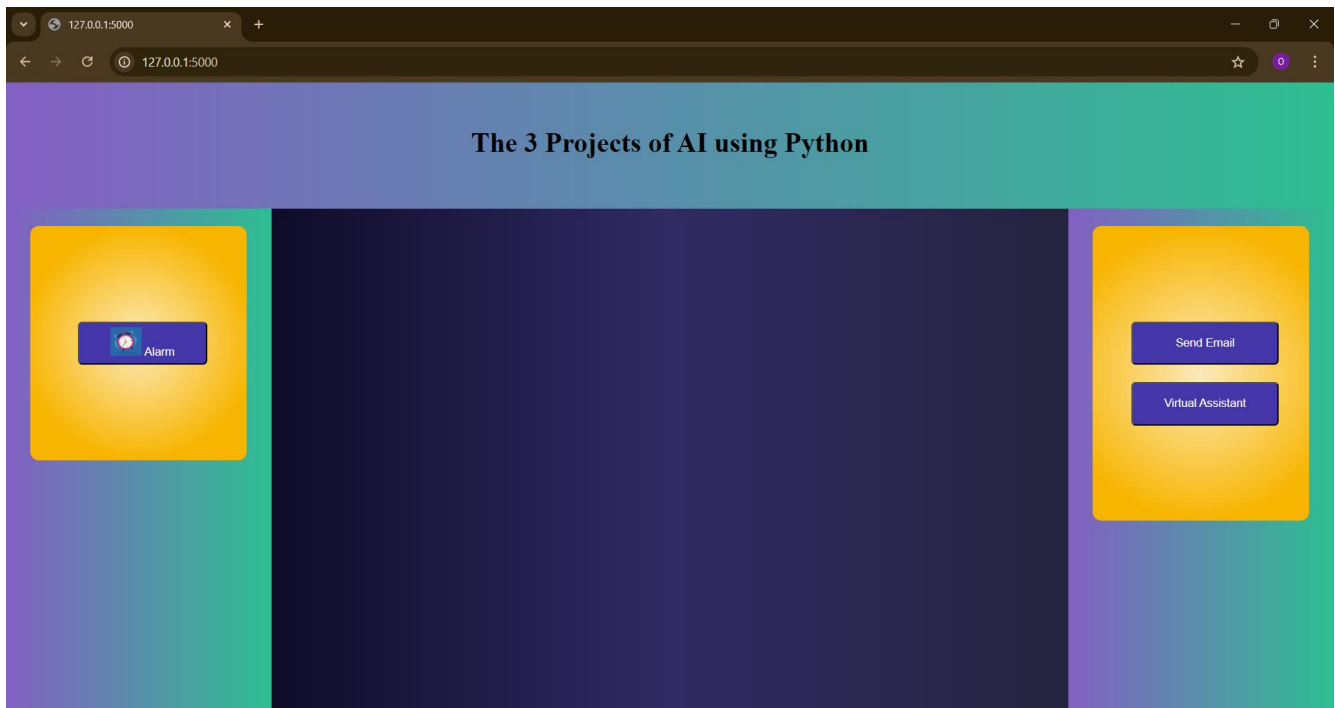
            </div>
        </div>
    </form>
</body>
<script>
    {% if message %}
    document.getElementById('textdiv').innerHTML('{{ message }}');
    {% endif %}
    {% if msg %}
    alert('{{ msg }}');
    {% endif %}
</script>
</html>

```

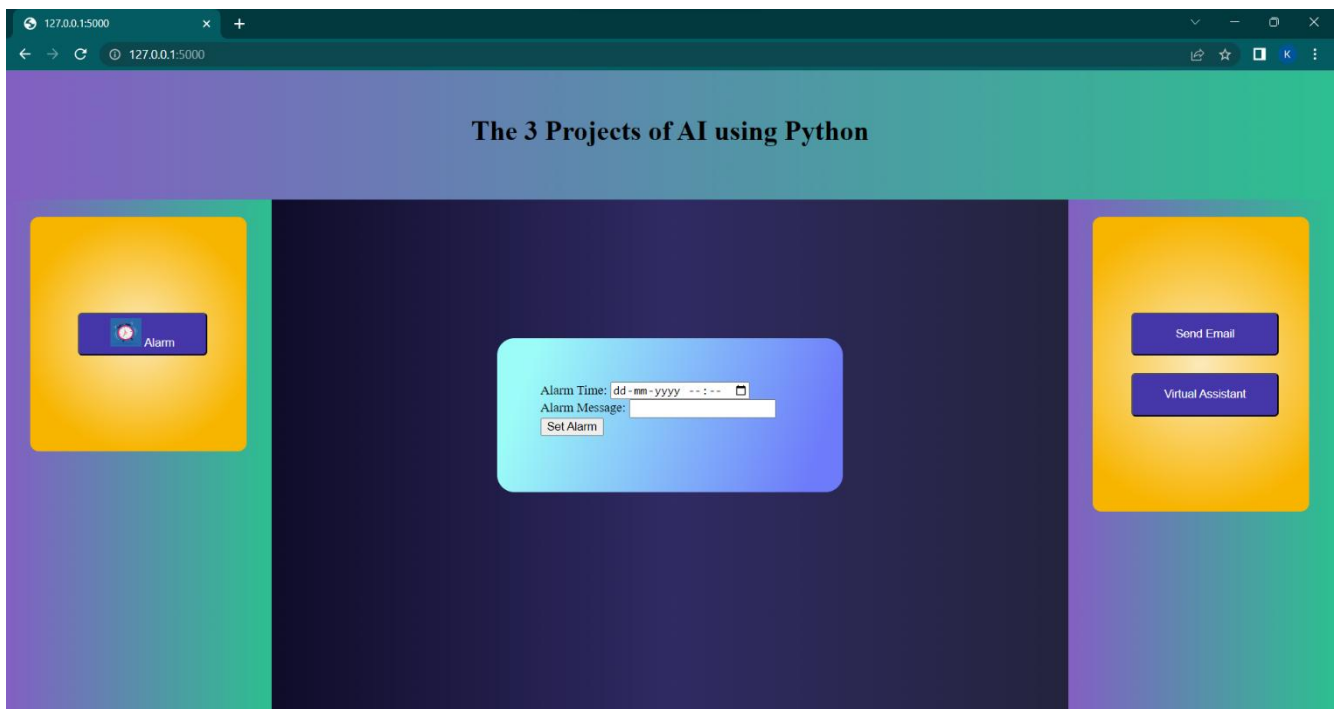
5.4: Screen Shots:

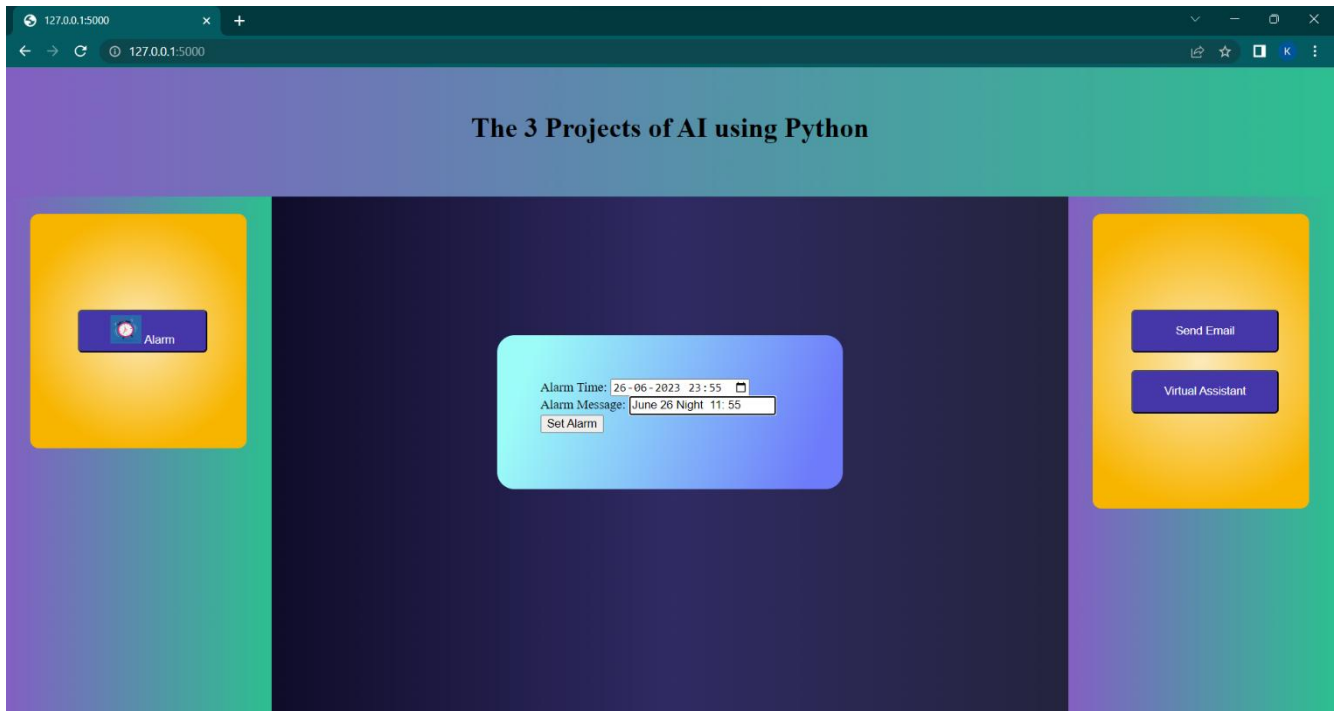
User Side:

HOME



Alarm





Gmail Registration

The 3 Projects of AI using Python

Registration

Email

AppPasscode

Set Password

Gmail Login

The 3 Projects of AI using Python

Login

Email

Password

[Reset AppCode!?](#)

Account not registered ?[Register!!](#)

Reset App Password

The 3 Projects of AI using Python

Reset Appcode

Email
Enter the gmail

Password
Enter the password

AppPasscode
Enter the AppPasscode

Submit

Alarm

Send Email

Virtual Assistant

Email Automation Interface

The 3 Projects of AI using Python

Mail with OTP Generation

Mail with OTP Verification

Mail with Attach

Mail with Subject

Alarm

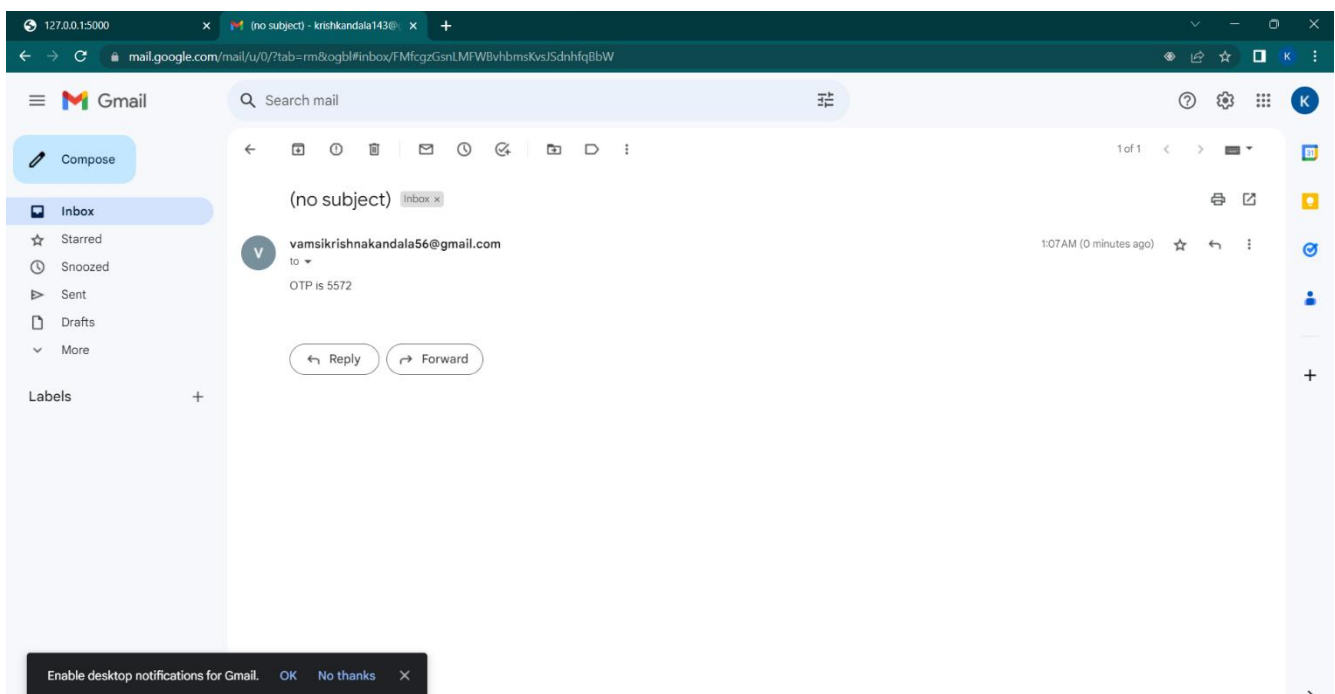
Send Email

Virtual Assistant

Mail with OTP Generation

The screenshot shows a web browser window with the address bar displaying '127.0.0.1:5000'. The page title is 'The 3 Projects of AI using Python'. The main content area has a dark blue background with a central light blue box titled 'Mail With OTP Generation'. This box contains a form with the following fields: 'From' (with a sub-label 'Enter sender Gmail'), 'Sender Password' (with a sub-label 'Enter sender Password'), and 'To' (with a sub-label 'Enter Receiver Gmail'). A green 'Generate OTP' button is at the bottom of the form. To the left of the central box is a yellow box with an 'Alarm' button. To the right is a yellow box with 'Send Email' and 'Virtual Assistant' buttons.

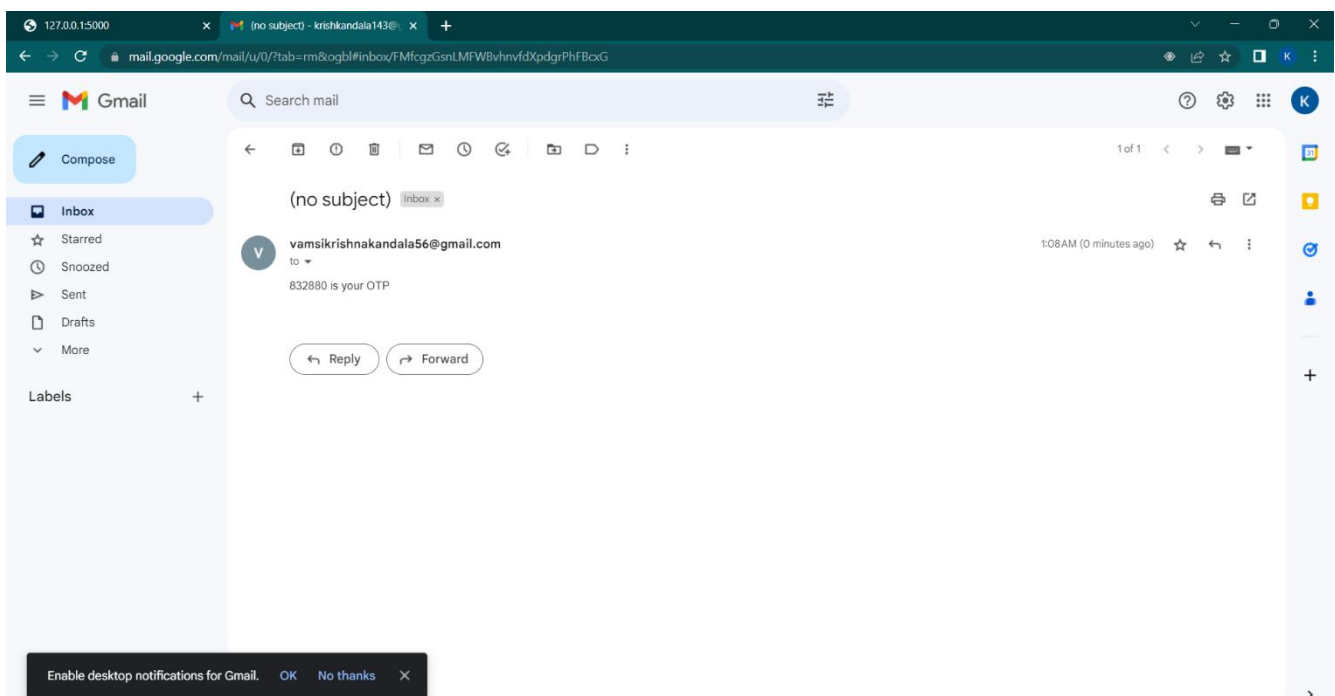
Receiver Receives OTP Mail



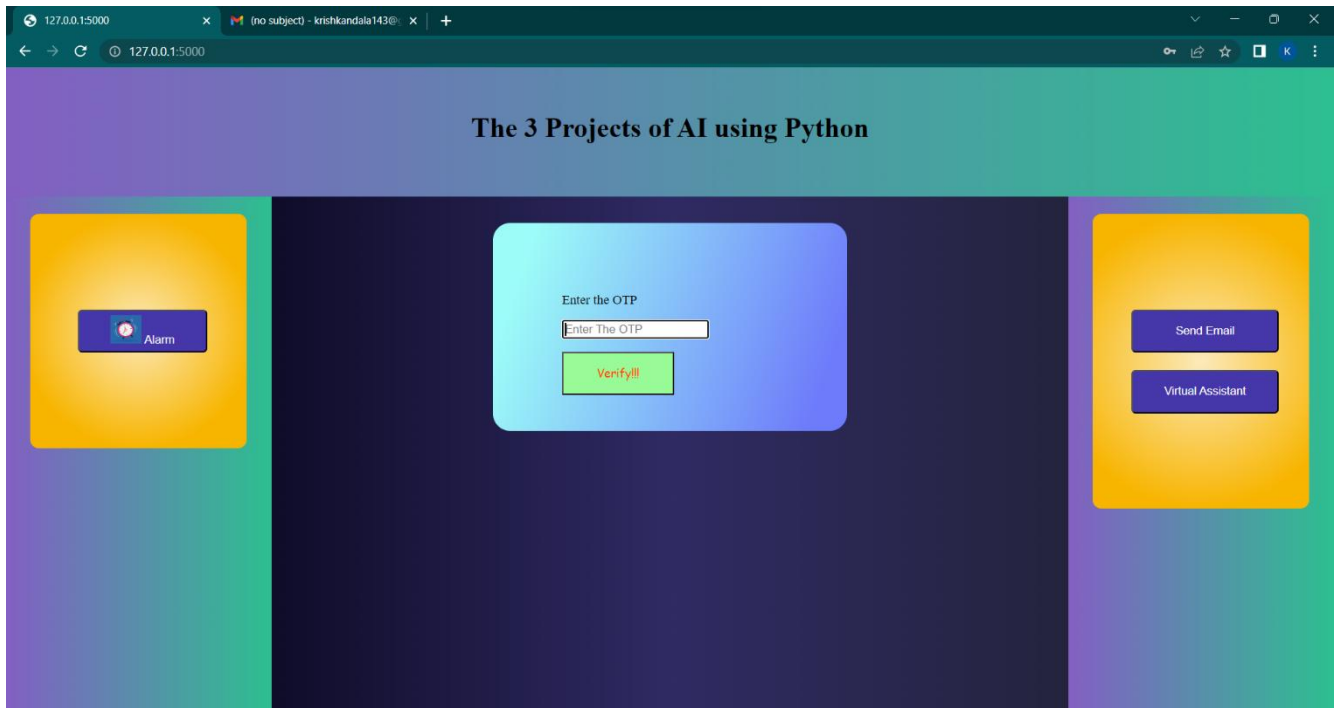
Mail with OTP Generation and Verification



Receiver Receives Mail with OTP

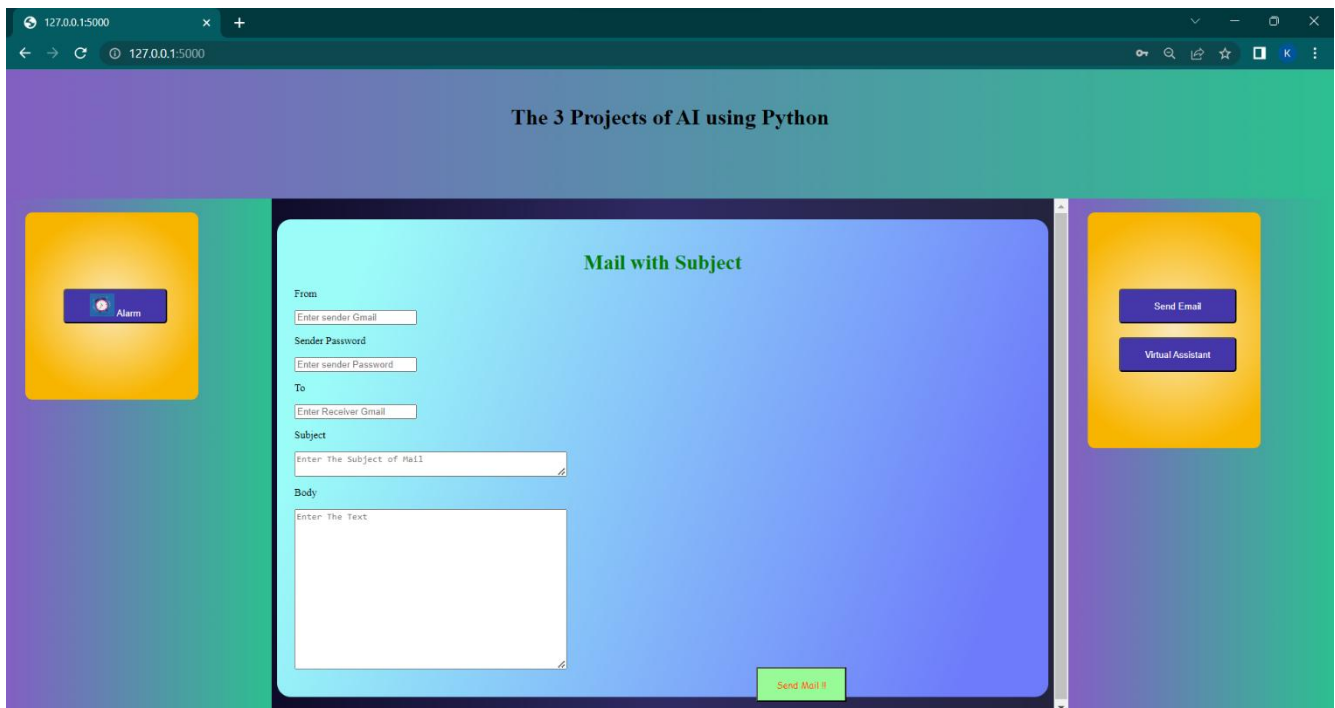


OTP Verification Interface

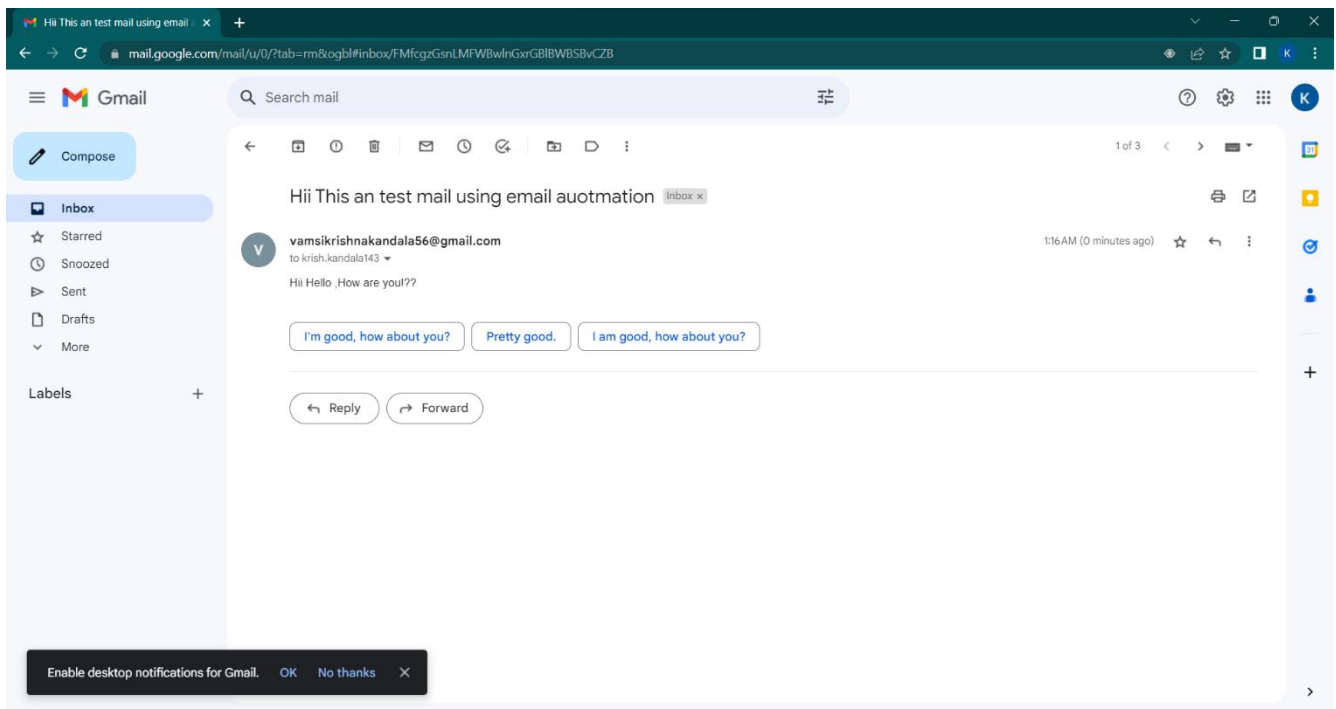
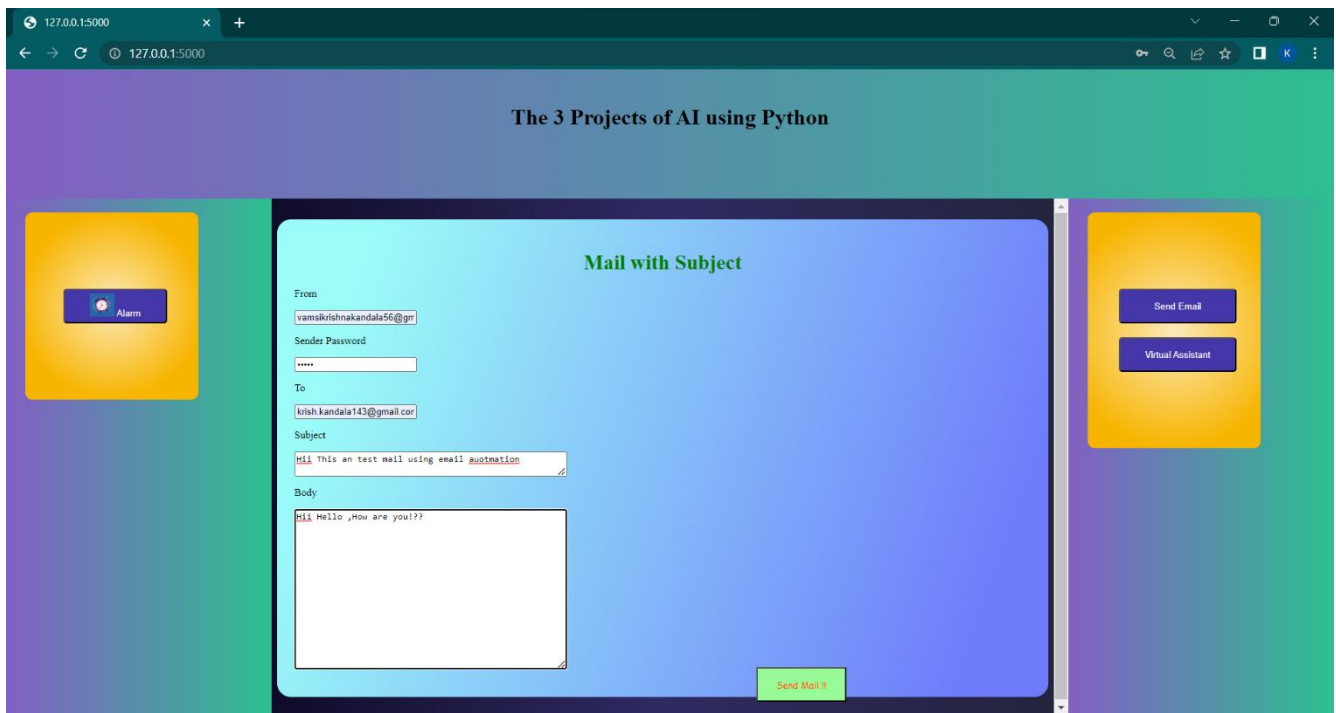


The screenshot shows a web browser window with the address bar displaying "127.0.0.1:5000". The page title is "The 3 Projects of AI using Python". The interface features a central blue box with the text "Enter the OTP" and a text input field labeled "Enter The OTP". Below the input field is a green button labeled "Verify!!!". To the left of the central box is a yellow box with a blue button labeled "Alarm". To the right of the central box is a yellow box with two blue buttons: "Send Email" and "Virtual Assistant".

Mail with Subject



The screenshot shows a web browser window with the address bar displaying "127.0.0.1:5000". The page title is "The 3 Projects of AI using Python". The interface features a central blue box with the text "Mail with Subject". Below the text are several text input fields: "From" (labeled "Enter sender Gmail"), "Sender Password" (labeled "Enter sender Password"), "To" (labeled "Enter Receiver Gmail"), "Subject" (labeled "Enter The Subject of Mail"), and "Body" (labeled "Enter The Text"). A green button labeled "Send Mail" is located at the bottom right of the central box. To the left of the central box is a yellow box with a blue button labeled "Alarm". To the right of the central box is a yellow box with two blue buttons: "Send Email" and "Virtual Assistant".



Mail With Attachment

127.0.0.1:5000

The 3 Projects of AI using Python

Mail with Attachment

From:

Sender Password:

To:

Subject:

Body:

Choose file to attach: N...en multiple>

127.0.0.1:5000

The 3 Projects of AI using Python

Mail with Attachment

From:

Sender Password:

To:

Subject:

Body:

Choose file to attach: p...jpg multiple>

6.TESTING

Software testing is a critical element of the software quality assurance and represents the ultimate review of specification, design and coding. Testing is the exposure of the system to trial input to see whether it produces correct output.

Testing Phases: Software testing phases include the following:

- Test activities are determined and test data selected.
- The test is conducted and test results are compared with the expected results.

There are various types of testing:

Unit Testing: Unit testing is essentially for the verification of the code produced during the coding phase and the goal is test the internal logic of the module/program. In the Generic code project, the unit testing is done during coding phase of data entry forms whether the functions are working properly or not. In this phase all the drivers are tested they are rightly connected or not.

Integration Testing: All the tested modules are combined into subsystems, which are then tested. The goal is to see if the modules are properly integrated, and the emphasis being on the testing interfaces between the modules. The generic code integration testing is done mainly on table creation module and insertion module.

System Testing: It is mainly used if the software meets its requirements. The reference document for this process is the requirement document. **Acceptance Testing:** It is performed with realistic data of the client to demonstrate that the software is working satisfactorily.

Testing Methods: Testing is a process of executing a program to find out errors. If testing is conducted successfully, it will uncover all the errors in the software. Any testing can be done basing on two ways:

White Box Testing: It is a test case design method that uses the control structures of the procedural design to derive the test cases. Using this testing a software engineer can derive the following test cases: Exercise all the logical decisions on either true or false sides. Execute all loops at their boundaries and within their operational boundaries. Exercise the internal data structures to assure their validity.

Black Box Testing: It is a test case design method used on the functional requirements of the software. It will help a software engineer to derive sets of input conditions that will exercise all the functional requirements of the program.

Black box Testing attempts to find errors in the following categories:

- Incorrect or missing functions
- Interface errors
- Errors in data structures
- Performance errors
- Initialization and termination errors

By Black box testing we derive a set of test cases that satisfy the following criteria:

- Test cases that reduce by a count that is greater than one, the number of additional test cases that must be designed to achieve reasonable testing.
- Test cases that tell us something about the presence or absence of classes of errors rather than errors associated only with a specific test at hand.

TEST APPROACH:

Testing can be done in two ways:

- Bottom-up approach
- Top-down approach

Bottom-up approach: Testing can be performed starting from smallest and lowest level modules and proceeding one at a time. For each module in bottom up testing a short program executes the module and provides the needed data so that the module is asked to perform the way it will when embedded within the larger system. When bottom level modules are tested attention turns to those on the next level that use the lower-level ones they are tested individually and then linked with the previously examined lower-level modules.

Top-down approach: This type of testing starts from upper-level modules, since the detailed activities usually performed in the lower-level routines are not provided stubs are written. A stub is a module shell called by upper-level module and that when reached properly will return a message to the calling module indicating that proper interaction occurred. No attempt is made to verify the correctness of the lower-level module.

7. Conclusion:

Python enhances PowerPoint presentations by automating tasks such as alarms, email notifications, and virtual assistants. With Python scripts, you can set alarms to trigger slide transitions or highlight key points at scheduled times, ensuring seamless presentations. Additionally, Python can manage email notifications, allowing for automated updates and reminders to be sent to attendees before, during, or after the presentation. By implementing voice-activated virtual assistants, presenters can control slides and access real-time data, improving audience engagement. These features streamline the presentation process and save valuable time and effort for presenters.

8. BIBLIOGRAPHY

WEBSITES REFERRED:

<https://www.python.org/>
[https://en.wikipedia.org/wiki/Python_\(programming_language\)](https://en.wikipedia.org/wiki/Python_(programming_language))
<https://www.learnpython.org/>
<https://realpython.com/>
<https://flask.palletsprojects.com/en/2.3.x/>
<https://www.tutorialspoint.com/flask/index.htm>
<https://www.fullstackpython.com/flask.html>