**CHAPTER 4**

**DEVELOPMENT PROCESS**

* 1. **REQUIREMENT ANALYSIS**

Requirements are a feature of a system or description of something that the system is capable of doing in order to fulfil the system’s purpose. It provides the appropriate mechanism for understanding what the customer wants, analyzing the needs assessing feasibility, negotiating a reasonable solution, specifying the solution unambiguously, validating the specification and managing the requirements as they are translated into an operational system.

* 1. **PYTHON:**

Python is a dynamic, high level, free open source and interpreted programming language. It supports object-oriented programming as well as procedural oriented programming. In Python, we don’t need to declare the type of variable because it is a dynamically typed language.

For example, x=10. Here, x can be anything such as String, int, etc.

Python is an interpreted, object-oriented programming language similar to PERL, that has gained popularity because of its clear [syntax](https://whatis.techtarget.com/definition/syntax) and readability. Python is said to be relatively easy to learn and portable, meaning its statements can be interpreted in a number of [operating system](https://whatis.techtarget.com/definition/operating-system-OS)s, including UNIX-based systems, Mac OS, MS-DOS, OS/2, and various versions of Microsoft Windows 98. Python was created by Guido van Rossum, a former resident of the Netherlands, whose favourite comedy group at the time was Monty Python's Flying Circus. The source code is freely available and open for modification and reuse. Python has a significant number of users.

**Features in Python**

There are many features in Python, some of which are discussed below

* Easy to code
* Free and Open Source
* Object-Oriented Language
* GUI Programming Support
* High-Level Language
* Extensible feature
* Python is Portable language
* Python is Integrated language
* Interpreted Language
  1. **ANACONDA**

Anaconda distribution comes with over 250 packages automatically installed, and over 7,500 additional open-source packages can be installed from [PyPI](https://en.wikipedia.org/wiki/Python_Package_Index" \o "Python Package Index) as well as the [conda](https://en.wikipedia.org/wiki/Conda_(package_manager)" \o "Conda (package manager)) package and virtual environment manager. It also includes a GUI, Anaconda Navigator, as a graphical alternative to the command line interface (CLI).

The big difference between conda and the [pip package manager](https://en.wikipedia.org/wiki/Pip_(package_manager)" \o "Pip (package manager)) is in how package dependencies are managed, which is a significant challenge for Python data science and the reason conda exists.

When pip installs a package, it automatically installs any dependent Python packages without checking if these conflict with previously installed packages. It will install a package and any of its dependencies regardless of the state of the existing installation. Because of this, a user with a working installation of, for example, Google Tensorflow, can find that it stops working having used pip to install a different package that requires a different version of the dependent numpy library than the one used by Tensorflow. In some cases, the package may appear to work but produce different results in detail.

In contrast, conda analyses the current environment including everything currently installed, and, together with any version limitations specified (e.g., the user may wish to have Tensorflow version 2,0 or higher), works out how to install a compatible set of dependencies, and shows a warning if this cannot be done.

Opensource packages can be individually installed from the Anaconda repository, Anaconda Cloud (anaconda.org), or the user's own private repository or mirror, using the conda install command. Anaconda, Inc. compiles and builds the packages available in the Anaconda repository itself, and provides binaries for Windows 32/64 bit, Linux 64 bit and MacOS 64-bit. Anything available on [PyPI](https://en.wikipedia.org/wiki/Python_Package_Index" \o "Python Package Index) may be installed into a conda environment using pip, and conda will keep track of what it has installed itself and what pip has installed.

Custom packages can be made using the conda build command, and can be shared with others by uploading them to Anaconda Cloud, [PyPI](https://en.wikipedia.org/wiki/Python_Package_Index" \o "Python Package Index) or other repositories.

The default installation of Anaconda2 includes Python 2.7 and Anaconda3 includes Python 3.7. However, it is possible to create new environments that include any version of Python packaged with conda.

### Anaconda Navigator

Anaconda Navigator is a desktop [graphical user interface (GUI)](https://en.wikipedia.org/wiki/Graphical_user_interface" \o "Graphical user interface) included in Anaconda distribution that allows users to launch applications and manage conda packages, environments and channels without using [command-line commands](https://en.wikipedia.org/wiki/Command-line_interface" \o "Command-line interface). Navigator can search for packages on Anaconda Cloud or in a local Anaconda Repository, install them in an environment, run the packages and update them. It is available for [Windows](https://en.wikipedia.org/wiki/Windows" \o "Windows), [macOS](https://en.wikipedia.org/wiki/MacOS" \o "MacOS) and [Linux](https://en.wikipedia.org/wiki/Linux" \o "Linux).

The following applications are available by default in Navigator:

* [JupyterLab](https://en.wikipedia.org/wiki/Project_Jupyter" \l "JupyterLab" \o "Project Jupyter)
* [Jupyter Notebook](https://en.wikipedia.org/wiki/Project_Jupyter" \l "Jupyter_Notebook" \o "Project Jupyter)
* QtConsole
* [Spyder](https://en.wikipedia.org/wiki/Spyder_(software)" \o "Spyder (software))
* [Glue](https://en.wikipedia.org/wiki/Glue_(software)" \o "Glue (software))
* [Orange](https://en.wikipedia.org/wiki/Orange_(software)" \o "Orange (software))
* [RStudio](https://en.wikipedia.org/wiki/RStudio" \o "RStudio)
* [Visual Studio Code](https://en.wikipedia.org/wiki/Visual_Studio_Code" \o "Visual Studio Code)
  + 1. **JUPYTER NOTEBOOK**

Jupyter [Notebook](https://en.wikipedia.org/wiki/Notebook_interface" \o "Notebook interface) (formerly IPython Notebooks) is a [web-based interactive](https://en.wikipedia.org/wiki/Rich_Internet_application" \o "Rich Internet application) computational environment for creating Jupyter notebook documents. The "notebook" term can colloquially make reference to many different entities, mainly the Jupyter [web application](https://en.wikipedia.org/wiki/Web_application" \o "Web application), Jupyter Python web server, or Jupyter document format depending on context. A Jupyter Notebook document is a [JSON](https://en.wikipedia.org/wiki/JSON" \o "JSON) document, following a versioned schema, containing an ordered list of input/output cells which can contain code, text (using [Markdown](https://en.wikipedia.org/wiki/Markdown" \o "Markdown)), mathematics, plots and rich media, usually ending with the ".ipynb" extension.

Jupyter Notebook can connect to many kernels to allow programming in different languages. By default, Jupyter Notebook ships with the IPython kernel. As of the 2.3 release[[11]](https://en.wikipedia.org/wiki/Project_Jupyter" \l "cite_note-releasenote23-11)[[12]](https://en.wikipedia.org/wiki/Project_Jupyter" \l "cite_note-releasenote20-12) (October 2014), there are currently 49 Jupyter-compatible kernels for many programming languages, including [Python](https://en.wikipedia.org/wiki/Python_(programming_language)" \o "Python (programming language)), [R](https://en.wikipedia.org/wiki/R_(programming_language)" \o "R (programming language)), [Julia](https://en.wikipedia.org/wiki/Julia_(programming_language)" \o "Julia (programming language)) and [Haskell](https://en.wikipedia.org/wiki/Haskell_(programming_language)" \o "Haskell (programming language)).

The Notebook interface was added to IPython in the 0.12 release[[14]](https://en.wikipedia.org/wiki/Project_Jupyter" \l "cite_note-releasenote012-14) (December 2011), renamed to Jupyter notebook in 2015 (IPython 4.0 – Jupyter 1.0). Jupyter Notebook is similar to the notebook interface of other programs such as [Maple](https://en.wikipedia.org/wiki/Maple_(software)" \o "Maple (software)), [Mathematica](https://en.wikipedia.org/wiki/Mathematica" \o "Mathematica), and [SageMath](https://en.wikipedia.org/wiki/SageMath" \o "SageMath), a computational interface style that originated with Mathematica in the 1980s. According to [The Atlantic](https://en.wikipedia.org/wiki/The_Atlantic" \o "The Atlantic), Jupyter interest overtook the popularity of the Mathematica notebook interface in early 2018.

**4.3.3 Streamlit**

**Streamlit** is an open-source Python framework designed for the rapid development of interactive, data-driven web applications. It simplifies the process of turning data scripts and machine learning models into web applications, making it particularly well-suited for developers who want to create applications without needing extensive front-end development knowledge. Streamlit has gained popularity for building quick prototypes, dashboards, and applications that rely on real-time data processing and visualization.

In this section, we will delve into the key features, advantages, and capabilities of Streamlit, and how it enhances the development of AI-based systems, especially for applications like trip planners that require seamless data interactions and visualizations.

### ****Overview of Streamlit****

Streamlit was created with the goal of enabling data scientists and machine learning engineers to build and deploy web applications with minimal effort. The primary objective of the framework is to allow Python developers to convert their data analysis scripts and models into interactive applications using simple and intuitive code. Unlike traditional web frameworks such as Flask or Django, Streamlit focuses more on providing an easy interface for developing rich user interfaces without requiring advanced knowledge of HTML, CSS, or JavaScript.

Streamlit applications can be developed using just a few lines of Python code. The framework automatically handles the web server, frontend interface, and updates in real-time, making it ideal for rapid prototyping and showcasing data visualizations or AI models.

### ****Key Features of Streamlit****

**Simple Pythonic API:**

Streamlit allows developers to write applications with Python, using simple commands like st.write(), st.slider(), and st.plotly\_chart(). There is no need for HTML, CSS, or JavaScript knowledge, which simplifies the process and makes it accessible to data scientists.

**Real-Time Interactivity:**

Streamlit apps are interactive, meaning that changes to input fields, sliders, or selections immediately update the visualizations and outputs. This makes it ideal for data exploration, model testing, and generating personalized results dynamically, which is especially useful in AI-based applications like a trip planner.

**Ease of Deployment:**

Once an app is created, it can be deployed easily to various platforms using simple commands. Streamlit also supports integration with cloud platforms like AWS, Heroku, or Google Cloud, enabling scalable applications with minimal effort.

**Built-in Data Visualization:**

Streamlit supports a wide variety of plotting libraries, such as Matplotlib, Plotly, Altair, and others. These libraries allow developers to add rich, interactive visualizations to their applications, providing a more engaging and informative user experience.

* 1. **RESOURCE REQUIREMENTS:**

**SOFTWARE REQUIREMENTS**:

|  |  |
| --- | --- |
| Operating System | Windows 7or later |
| Simulation Tool | Anaconda (Jupyter notebook) |
| Documentation | Ms – Office |

**HARDWARE REQUIREMENTS:**

|  |  |
| --- | --- |
| CPU type | I5 |
| Ram size | 4GB |
| Hard disk capacity | 80 GB |
| Keyboard type | Internet keyboard |
| Monitor type | 15 Inch colour monitor |
| CD -drive type | 52xmax |

* 1. **PROPOSED SYSTEM**

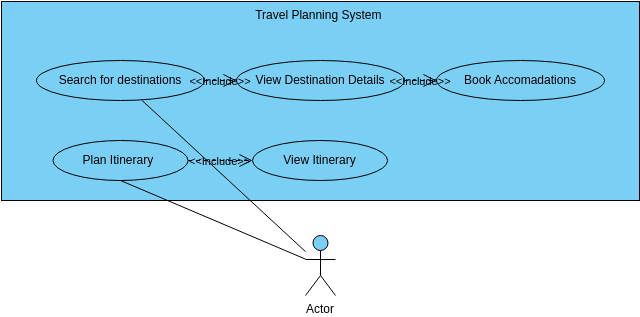
The **Proposed System** aims to provide a comprehensive solution for trip planning by integrating a **budget plan**, **day planner**, and **personalized travel recommendations**. Built using the **Streamlit framework**, the system allows users to easily plan their trip by entering their travel details such as budget, travel dates, and preferred destinations. The system is designed to fetch worldwide data for destinations and hotels using **API keys**, providing users with up-to-date information on various travel options.

Users can input their trip preferences into the system, such as the location, budget constraints, and desired activities. Based on the provided input, the system then searches through various APIs to gather information about potential places to visit and hotels for accommodation. This ensures that users receive a range of suggestions within their budget. The system not only provides a budget breakdown but also offers a **day-wise trip plan**, allowing users to organize their activities based on the time they have available and their personal interests.

The system fetches data through third-party **APIs** that provide real-time details on travel destinations, available accommodations, and activities. This data is processed and presented in a user-friendly interface where users can explore different travel options and select their preferred choices. The integration of **API keys** ensures that the data fetched is accurate, up-to-date, and sourced from reliable platforms, enhancing the overall user experience.

In summary, the **Proposed System** allows users to plan their trips efficiently by considering both their budget and preferences, making it easier for them to make informed decisions about their travel destinations and accommodations.

* 1. **ADVANTAGES**
* High Accuracy and performance
* Less complication
* Highest prediction rate
  1. **SYSTEM ARCHITECTURE**



* 1. **MODULES DESCRIPTION:**

### ****Home Screen Module****

The **Home Screen Module** serves as the entry point for users to interact with the AI-based trip planner. This module is designed with simplicity and ease of use in mind, ensuring that users can seamlessly access all the features the system offers.

#### ****1. User Data Input****

The home screen will display fields for users to input their travel details. These details are essential for generating personalized trip plans. The user data input section typically includes:

**Destination Preferences**: A text field or dropdown menu where users can enter or select their preferred travel destination.

**Budget**: An input field to specify their budget for the entire trip.

**Activity Interests**: A checkbox or dropdown to specify what type of activities they are interested in (e.g., adventure, relaxation, sightseeing).

**Number of Travelers**: A field to specify the number of travelers in the group.

**Accommodation Preferences**: Options for users to specify their accommodation preferences (e.g., hotels, resorts, hostels).

Once the user inputs all the required details, they can proceed to the next step to search for travel recommendations.

#### ****2. Search for Trip Planner****

The **Search for Trip Planner** functionality allows the system to process the user’s input and search for travel options that match their preferences. Upon clicking the search button, the system will:

**Query APIs**: The system will use API keys to connect to third-party travel APIs (e.g., for destinations, accommodations, activities) to fetch relevant data.

**Process Results**: The system will process the data received from the APIs and filter results based on user preferences such as budget, destination, and activity interests.

**Display Options**: The system will display a list of recommended destinations, accommodations, and activities in a visually appealing format, with essential information such as prices, descriptions, and availability.

The user can refine their search based on the results and select specific options to add to their trip plan.

#### ****3. Search Results****

The **Result Section** will display the search outcomes based on the input provided. This section will include:

**Destination Suggestions**: A list of recommended travel destinations with a brief description, including key attractions, activities, and nearby hotels.

**Accommodation Options**: A curated list of available hotels or other accommodation types that match the user's budget and preferences, showing prices and ratings.

**Activities and Attractions**: A list of suggested activities (e.g., tours, adventure sports, local attractions) that users can choose to include in their itinerary.

**Cost Estimates**: For each destination and accommodation option, an estimated total cost of the trip will be displayed, including accommodation, activities, and transportation if applicable.

The results will be displayed in an interactive format, allowing users to click on each option for more detailed information. The system will also display the total estimated cost for each itinerary, helping users make informed decisions about their trip.

### ****Key Features of the Home Screen Module:****

**User-Friendly Interface**: Simple and intuitive design for easy navigation.

**Dynamic Data Input**: Users can input their travel preferences easily.

**Real-Time Search**: The system fetches real-time data from APIs for accurate travel recommendations.

**Result Display**: The results are clearly displayed with all necessary information like prices, descriptions, and availability.

This module is designed to provide a smooth, efficient experience for users, making the process of planning a trip as easy and enjoyable as possible.