**Rasa College Chat Bot**

**ABSTRACT**

The growth of technologies like Artificial Intelligence (AI), Big Data & Internet of Things (IoT), etc. has marked many advancements in the technological world since the last decade. These technologies have a wide range of applications. One such application is “Chatterbot or “Chatbot”. Chatbots are conversational AIs, which mimics the human while conversing & eliminates the need of human by automating mundane tasks. In the study undertaken, we have created a chatbot in education domain & it is named as “College Chatbot”, This chatbot is a web-based application that analyses and understands user's queries and provides an instant and accurate response. Rasa technology is used to construct this chatbot. It's an open-source technology, which uses its two main packages i.e., Rasa Core & Rasa Natural Language Understanding (NLU) in order to build a Contextual AI Chatbot.

NLU is used to infer the intent and to extract the necessary entities from user input & the Rasa Core provides the output by building a probabilistic model with the help of Recurrent Neural Network (RNN). Evaluation of the model is done by getting a confusion matrix and performance measures like Precision, Accuracy & F1 Score which come out to be 0.628, 0.725 and 0.669 respectively on average basis. This chatbot's accuracy, lack of dependability on human resources, 24 x 7 accessibility and low maintenance creates various opportunities for its implementation. This conversational agent can not only be used in educational institutions but also in places where enquiry becomes a tedious task.

**Keywords:** Intelligent chatbot, Virtual assistants, College tasks,

**INTRODUCTION**

These days, we see the talk bots wherever Chatbots are the wellspring of answers to the client's inquiries in a specific space where it is working. Visit bots are the wellspring of answers to the client's inquiries. The requirement for school request framework emerges because of different reasons which include: the moderate idea of school site, an outcast would not realize where to look for a specific snippet of data, hard for the individual external school's area to remove data. The school enquiry framework will give the reaction by summing up the question and afterward yield answers, it additionally gives particular data what the client needs. Rasa is an open-source system for building AI bots which comprise of two parts: Rasa NLU and Rasa center.

Rasa center is the segment which handles the exchange motor for the system and helps in making more unpredictable chatbots with customization. Rasa's NLU assists the engineers with the innovation and the apparatuses essential for catching and understanding client input, deciding the purpose and substances.

**LITERATURE SURVEY**

**[1] S. Pawar, O. Rane, O. Wankhade and P. Mehta, "A Web Based College Enquiry Chatbot with Results", International Journal of Innovative Research in Science Engineering and Technology*,* vol. 7, no. 4, pp. 3874-3880, April 2018.**

A chatbot (also known as a talkbot, chatterbot, Bot, IM bot, interactive agent, or Artificial Conversational Entity) is a computer program which conducts a conversation via auditory or textual methods. The chatbot has information stored in its database to identify the sentences and making a decision itself as response to answer a given question. The college enquiry chat bot will be built using algorithms that analyzes queries and understand user’s message. This System will be a web application which provides answer to the query of the student very effectively. Students just have to put their query to the chat-bot which is used for chatting. The system will use bigram and sentence similarity algorithms to give appropriate answers to the user. If the answer is found invalid, then there is a system to declare the answer as invalid. These invalid answers can be deleted or modified by the administrator of the system.

**[2] A. Jiao, "An Intelligent Chatbot System Based on Entity Extraction Using RASA NLU and Neural Network", IOP Conf. Series: Journal of Physics: Conf. Series*,* 2020.**

Intelligent chatbot systems are popular issues in the application fields of robot system and natural language processing. As the development of natural language processing and neural network algorithms, the application of artificial intelligence is increasing in Chatbot systems, which are typically used in dialog systems for various practical purposes including customer service or information acquisition. This paper designs the functional framework and introduces the principle of RASA NLU for the Chatbot system, then it integrates RASA NLU and neural network (NN) methods and implements the system based on entity extraction after intent recognition. With the experimental comparison and validation, our developed system can realize automatic learning and answering the collected questions about finance. The system analysis of two methods also validate that RASA NLU outperforms NN in accuracy for a single experiment, but NN has better integrity to classify entities from segmented words.

**[3] K. Deepika, V. Tilekya, J. Mamatha and T. Subetha, "Jollity Chatbot-A contextual AI Assistant",2020 Third International Conference on Smart Systems and Inventive Technology (ICSSIT)*,* pp. 1196-1200, 2020.**

Chatbot is a software application that can stimulate a conversation via text, instead of direct contact with a live human through messaging applications, websites and mobile applications. Chatbot applications help to make interactions between people and services by enhancing the customer experience. Chatbot is widely used in the areas of food ordering, ecommerce and transportation, etc. Practically it is not possible to find a permanent companion to make us happy all the time. Hence, this paper has planned to design a jollity chatbot to talk with the human users and make sure that it entertains and give suggestion and motivation in tough times. The jollity chatbot is implemented in Rasa, an open -source conversational AI framework and it is easy to customize. The proposed method has added 12 intents with each more than 8 text examples constituting a total of 100 input samples in nlu.md and their response in domain.yml. The flow of interactions is given in stories.md. The jollity chatbot is deployed in Telegram using ngrok and the server URL details and the access token are given in the credentials.yml. The system is experimented with various evaluation measures like accuracy of the intents, accuracy of the stories and the confusion matrix to shows that the proposed jollity chatbot system is more robust and can identify the user intents appropriately.

**[4] H. Gawade, V. Patil, P. Vishe and S. Kolpe, "College Enquiry Chat-Bot System", International Journal of Engineering Research & Technology, vol. 9, no. 9, pp. 741-742, September 2020.**

Nowadays, many people are using smartphone with many new applications i.e. technology is growing day by day. Today Artificial Intelligence is playing a major role in a variety of fields ranging from industries in product manufacturing, to customer care in public relations. As there are many online Artificial Intelligence (AI) systems or chat bots which are in existence that help people solve their problems. So, we are going to implement a virtual assistant based on AI that can solve any college related query. This will work as a College Oriented Intelligence machine. This virtual machine will respond the queries of students on college related issues. A chat bot has information stored in its database to identify the sentences and making a decision itself as response to answer a given question. The college enquiry chat bot will be built using algorithm that analyses queries and understand user’s message.

**[5]** **M. Joshi and R. K. Sharma, "An Analytical Study and Review of Open Source Chatbot Framework*",*International Journal of Engineering Research & Technology (IJERT), vol. 09, no. 06, June 2020.**

In the era of chatbots, besides imitating humans they can also perform complex tasks like booking tickets for movie etc. Out of various implementations, RASA is open source implementation for NLU and DIET model. It can interact with database, api, conversational flow, interactive learning with reinforcement Neural network. In this study, various features of rasa core are studied and upto much extent it \can perform complex tasks. Implementation details are studied like interaction with database, API. Tracker Store has been examined with modifying the socket.io core file adding metadata to the user message data, so that user ip and port can be captured. Furthermore, the action, interactive learning and implementation details are tested on windows PyCharm IDE.

**EXISTING SYSTEM**

In the existing system, if any student wants any information like College admissions, How many branches in college, Hostel fee structure, College fee structure, How many seats available for specific branch, About college environment, Available Specializations in a department like Data Science, Machine Learning in CSE , Infrastructure of college like labs, library, College placements, Internships and college professors, Special events that occur in our college i.e. college fests, College extra circular activities updates like annual day, sports events, technical events, Achievements of college, NIRF Ranking, Naas rating to college etc.., they have to go to the collage and meet the collage admin to know the all information about collage and fecilities after that they have to go to the admissions. This is the time taking process also performance decreases.

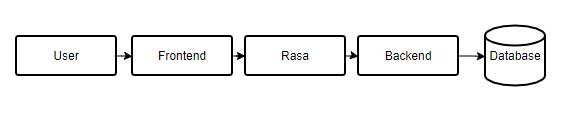
**Drawbacks**

* Complex user interface. Making user difficult to recognize the operations
* Late response to user due to highly dumped data set
* More processing time to process huge data

**PROPOSED SYSTEM**

Existing Systems were based on either rule based or neural networks but rasa brings best of both worlds. It uses both rule based engines and neural networks based models to deliver output and produce user-like conversations. In proposed system, the students no need to go to collage to get the all information about collage and facilities. It takes less time to train as we are using pre-trained neural network and using transfer learning on them.

**Architecture**



**Advantages**

* More visually pleasing
* Less response time to user
* Simple data set
* Less processing time

**APPLICATIONS:**

* Used to provide psychology related advice to patients.
* Government agencies can use them in health care field especially for poor people as it is easy to scale up and is inexpensive.
* Hospitals and Health care professionals can use them to save time and money

**SYSTEM REQUIREMENTS**

**HARDWARE CONFIGURATION:**

* Processor - I5/Intel Processor
* RAM - 8GB (min)
* Hard Disk - 500 GB

**SOFTWARE CONFIGURATION:**

* Operating System : Windows 10
* Server-side Script : Python 3.7.9
* IDE : VS Code
* Libraries Used : rasa.
* Framework : Django

**MODULES/ System Studies**

**1. System**

**1.1 Take Data:**

System will receive data from the user.

**1.2 Preprocessing:**

The system will undergo for preprocessing.

**1.3 Training:**

The System will get trained.

**1.4 Model:**

The system will work based on model.

**1.5 Results:**

The system will deliver the output to the user.

**2. User**

**Testing**

**2.1 Send Query:**

User will send Query to the system.

**2.2 View Query Result:**

User will view his query result.

**ALGORITHMS:**

**NATURAL LANGUAGE PROCESSING:**

Natural language processing (NLP) is a subfield of linguistics, computer science, and artificial intelligence concerned with the interactions between computers and human language, in particular how to program computers to process and analyse large amounts of natural language data. The result is a computer capable of "understanding" the contents of documents, including the contextual nuances of the language within them. The technology can then accurately extract information and insights contained in the documents as well as categorize and organize the documents themselves.

A chatbot is an NLP software that can simulate a conversation (or a chat) with a user in natural language through messaging applications, websites, mobile apps or through the telephone.

Why are chatbots important? A chatbot is often described as one of the most advanced and promising expressions of interaction between humans and machines. However, from a technological point of view, a chatbot only represents the natural evolution of a Question Answering system leveraging Natural Language Processing (NLP). Formulating responses to questions in natural language is one of the most typical Examples of Natural Language Processing applied in various enterprises’ end-use applications.

Turn human language into structured data

Rasa Open Source provides open source natural language processing to turn messages from your users into intents and entities that chatbots understand. Based on lower-level machine learning libraries like Tensorflow and spaCy, Rasa Open Source provides natural language processing software that’s approachable and as customizable as you need. Get up and running fast with easy to use default configurations, or swap out custom components and fine-tune hyperparameters to get the best possible performance for your dataset.

Rasa Open Source is the most flexible and transparent solution for conversational AI—and open source means you have complete control over building an NLP chatbot that really helps your users.

What is natural language processing?

Natural language processing is a category of machine learning that analyzes freeform text and turns it into structured data. Natural language understanding is a subset of NLP that classifies the intent, or meaning, of text based on the context and content of the message. The difference between NLP and NLU is that natural language understanding goes beyond converting text to its semantic parts and interprets the significance of what the user has said.

Rasa Open source is a robust platform that includes natural language understanding and open source natural language processing. It’s a full toolset for extracting the important keywords, or entities, from user messages, as well as the meaning or intent behind those messages. The output is a standardized, machine-readable version of the user’s message, which is used to determine the chatbot’s next action.

Natural language processing is used anywhere an application needs to take raw user text as input: whether it’s a voice assistant receiving input from speech-to-text software, or a chatbot asking a user to type in their question. Natural language processing is the essential step that turns a string of words into a form that can be interpreted and acted upon by other systems in the application.

Want to see what other developers have built with Rasa? Visit the Rasa Community Showcase.

**Why open source NLP?**

Rasa Open Source is licensed under the Apache 2.0 license, and the full code for the project is hosted on GitHub. Rasa Open Source is actively maintained by a team of Rasa engineers and machine learning researchers, as well as open source contributors from around the world. This collaboration fosters rapid innovation and software stability through the collective efforts and talents of the community.

Unlike NLP solutions that simply provide an API, Rasa Open Source gives you complete visibility into the underlying systems and machine learning algorithms. NLP APIs can be an unpredictable black box—you can’t be sure why the system returned a certain prediction, and you can’t troubleshoot or adjust the system parameters. Rasa Open Source is completely transparent. You can see the source code, modify the components, and understand why your models behave the way they do.

Open source NLP also offers the most flexible solution for teams building chatbots and AI assistants. The modular architecture and open code base mean you can plug in your own pre-trained models and word embeddings, build custom components, and tune models with precision for your unique data set. Rasa Open Source works out-of-the box with pre-trained models like BERT, HuggingFace Transformers, GPT, spaCy, and more, and you can incorporate custom modules like spell checkers and sentiment analysis.

**Leverage the latest state-of-art NLP research**

Rasa’s dedicated machine learning Research team brings the latest advancements in natural language processing and conversational AI directly into Rasa Open Source. Working closely with the Rasa product and engineering teams, as well as the community, in-house researchers ensure ideas become product features within months, not years.

The Rasa Research team brings together some of the leading minds in the field of NLP, actively publishing work to academic journals and conferences. The latest areas of research include transformer architectures for intent classification and entity extraction, transfer learning across dialogue tasks, and compressing large language models like BERT and GPT-2. As an open source NLP tool, this work is highly visible and vetted, tested, and improved by the Rasa Community. Open source NLP for any spoken language, any domain Rasa Open Source provides natural language processing that’s trained entirely on your data. This enables you to build models for any language and any domain, and your model can learn to recognize terms that are specific to your industry, like insurance, financial services, or healthcare.

In the insurance industry, a word like “premium” can have a unique meaning that a generic, multi-purpose NLP tool might miss. Rasa Open Source allows you to train your model on your data, to create an assistant that understands the language behind your business. This flexibility also means that you can apply Rasa Open Source to multiple use cases within your organization. You can use the same NLP engine to build an assistant for internal HR tasks and for customer-facing use cases, like consumer banking.

Regional dialects and language support can also present challenges for some off-the-shelf NLP solutions. Rasa’s NLU architecture is completely language-agostic, and has been used to train models in Hindi, Thai, Portuguese, Spanish, Chinese, French, Arabic, and many more. You can build AI chatbots and virtual assistants in any language, or even multiple languages, using a single framework.

**Support multiple intents and hierarchical entities**

In the real world, user messages can be unpredictable and complex—and a user message can’t always be mapped to a single intent. Rasa Open Source is equipped to handle multiple intents in a single message, reflecting the way users really talk. Consider an example like “Yes, place my order. When will it arrive?” Rasa’s NLU engine can tease apart multiple user goals, so your virtual assistant responds naturally and appropriately, even to complex input.

Rasa’s open source NLP engine also enables developers to define hierarchical entities, via entity roles and groups. This unlocks the ability to model complex transactional conversation flows, like booking a flight or hotel, or transferring money between accounts. Entity roles and groups make it possible to distinguish whether a city is the origin or destination, or whether an account is savings or checking.

**Open source NLP tools for complete control of data privacy**

Protecting the security and privacy of training data and user messages is one of the most important aspects of building chatbots and voice assistants. Organizations face a web of industry regulations and data requirements, like GDPR and HIPAA, as well as protecting intellectual property and preventing data breaches.

Rasa Open Source deploys on premises or on your own private cloud, and none of your data is ever sent to Rasa. All user messages, especially those that contain sensitive data, remain safe and secure on your own infrastructure. That’s especially important in regulated industries like healthcare, banking and insurance, making Rasa’s open source NLP software the go-to choice for enterprise IT environments.

**Built-in NLU model performance testing and training data version control**

Rasa’s open source NLP engine comes equipped with model testing capabilities out-of-the-box, so you can be sure that your models are getting more accurate over time, before you deploy to production.

Measure F1 score, model confidence, and compare the performance of different NLU pipeline configurations, to keep your assistant running at peak performance. All NLU tests support integration with industry-standard CI/CD and DevOps tools, to make testing an automated deployment step, consistent with engineering best practices.

The Rasa stack also connects with Git for version control.Treat your training data like code and maintain a record of every update. Easily roll back changes and implement review and testing workflows, for predictable, stable updates to your chatbot or voice assistant.

**A conversation-driven approach to natural language processing**

Even the best NLP systems are only as good as the training data you feed them. Compared to other tools used for language processing, Rasa emphasises a conversation-driven approach, using insights from user messages to train and teach your model how to improve over time. Rasa’s open source NLP works seamlessly with Rasa X to capture and make sense of conversation data, turn it into training examples, and track improvements to your chatbot’s success rate.

**Contextual Conversations**

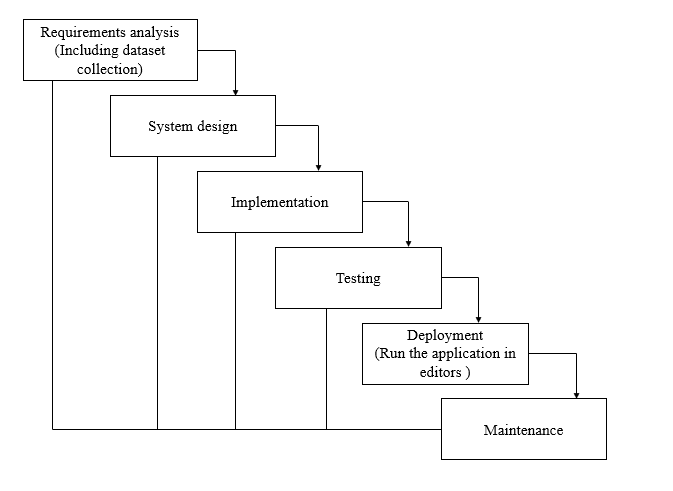
Taking context into account is often key to providing a good user experience. This page is a guide to creating contextual conversation patterns.

**APPLICATIONS:**

* Chatbots make life even easier for consumers. With chatbots, there’s no more long waits on hold to talk to a person on the phone or going through multiple steps to research and complete a purchase on websites.
* Product Suggestions
* Customer Support

**SOFTWARE DEVELOPMENT LIFE CYCLE – SDLC:**

In our project we use waterfall model as our software development cycle because of its step-by-step procedure while implementing.



**Fig1**: Waterfall Model

* **Requirement Gathering and analysis** − All possible requirements of the system to be developed are captured in this phase and documented in a requirement specification document.
* **System Design** − The requirement specifications from first phase are studied in this phase and the system design is prepared. This system design helps in specifying hardware and system requirements and helps in defining the overall system architecture.
* **Implementation** − With inputs from the system design, the system is first developed in small programs called units, which are integrated in the next phase. Each unit is developed and tested for its functionality, which is referred to as Unit Testing.
* **Integration and Testing** − All the units developed in the implementation phase are integrated into a system after testing of each unit. Post integration the entire system is tested for any faults and failures.
* **Deployment of system** − Once the functional and non-functional testing is done; the product is deployed in the customer environment or released into the market.

**Maintenance** − There are some issues which come up in the client environment. To fix those issues, patches are released. Also, to enhance the product some better versions are released. Maintenance is done to deliver these changes in the customer environment.

**SYSTEM DESIGN**

**UML DIAGRAMS**

UML stands for Unified Modeling Language. UML is a standardized general-purpose modeling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group.

The goal is for UML to become a common language for creating models of object-oriented computer software. In its current form UML is comprised of two major components: A Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with, UML.

The Unified Modeling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business modeling and other non-software systems.

The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems.

The UML is a very important part of developing objects-oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

**GOALS:**

The Primary goals in the design of the UML are as follows:

1. Provide users a ready-to-use, expressive visual modeling Language so that they can develop and exchange meaningful models.
2. Provide extendibility and specialization mechanisms to extend the core concepts.
3. Be independent of particular programming languages and development process.
4. Provide a formal basis for understanding the modeling language.
5. Encourage the growth of OO tools market.
6. Support higher level development concepts such as collaborations, frameworks, patterns and components.
7. Integrate best practices.

**USE CASE DIAGRAM:**

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.



**CLASS DIAGRAM:**

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class contains information.



**SEQUENCE DIAGRAM:**

A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.



**COLLABORATION DIAGRAM:**

In collaboration diagram the method call sequence is indicated by some numbering technique as shown below. The number indicates how the methods are called one after another. We have taken the same order management system to describe the collaboration diagram. The method calls are similar to that of a sequence diagram. But the difference is that the sequence diagram does not describe the object organization whereas the collaboration diagram shows the object organization.



**DEPLOYMENT DIAGRAM**

Deployment diagram represents the deployment view of a system. It is related to the component diagram. Because the components are deployed using the deployment diagrams. A deployment diagram consists of nodes. Nodes are nothing but physical hardware used to deploy the application.



**ACTIVITY DIAGRAM:**

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.



**COMPONENT DIAGRAM**

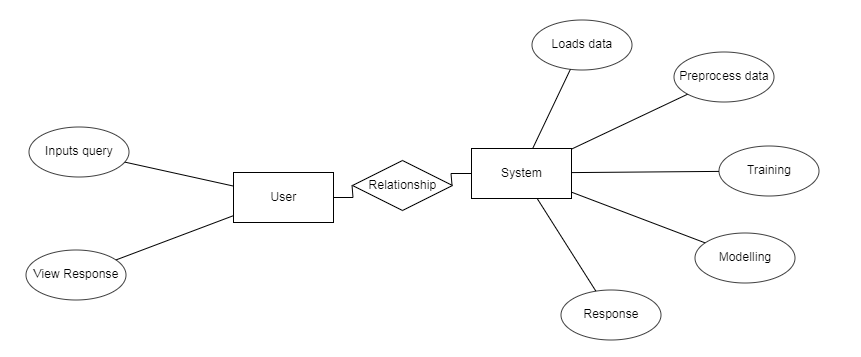
A component diagram, also known as a UML component diagram, describes the organization and wiring of the physical **c**omponents in a system. Component diagrams are often drawn to help model implementation details and double-check that every aspect of the system's required function is covered by planned development.



**ER DIAGRAM:**

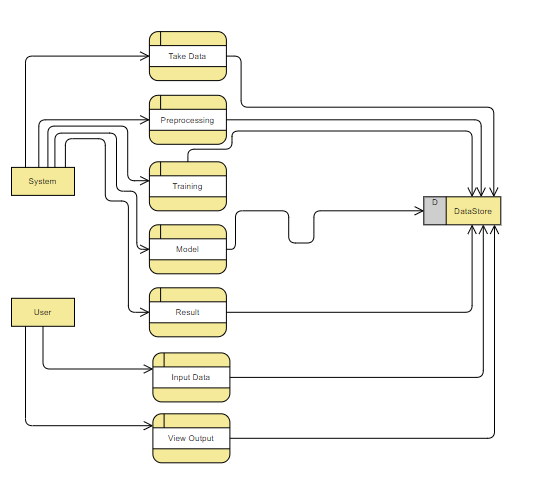
An Entity–relationship model (ER model) describes the structure of a database with the help of a diagram, which is known as Entity Relationship Diagram (ER Diagram). An ER model is a design or blueprint of a database that can later be implemented as a database. The main components of E-R model are: entity set and relationship set.

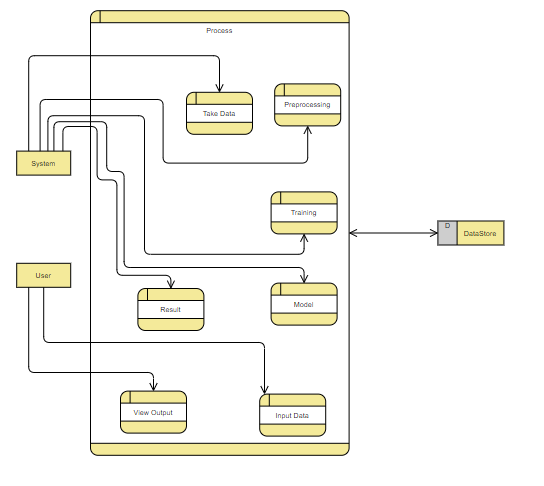
An ER diagram shows the relationship among entity sets. An entity set is a group of similar entities and these entities can have attributes. In terms of DBMS, an entity is a table or attribute of a table in database, so by showing relationship among tables and their attributes, ER diagram shows the complete logical structure of a database. Let’s have a look at a simple ER diagram to understand this concept.



**DFD DIAGRAM:**

A Data Flow Diagram (DFD) is a traditional way to visualize the information flows within a system. A neat and clear DFD can depict a good amount of the system requirements graphically. It can be manual, automated, or a combination of both. It shows how information enters and leaves the system, what changes the information and where information is stored. The purpose of a DFD is to show the scope and boundaries of a system as a whole. It may be used as a communications tool between a systems analyst and any person who plays a part in the system that acts as the starting point for redesigning a system.





# INTRODUCTION TO PYTHON

**Python**

### What Is A Script?

Up to this point, I have concentrated on the interactive programming capability of Python.  This is a very useful capability that allows you to type in a program and to have it executed immediately in an interactive mode

**Scripts are reusable**

Basically, a script is a text file containing the statements that comprise a Python program.  Once you have created the script, you can execute it over and over without having to retype it each time.

**Scripts are editable**

Perhaps, more importantly, you can make different versions of the script by modifying the statements from one file to the next using a text editor.  Then you can execute each of the individual versions.  In this way, it is easy to create different programs with a minimum amount of typing.

**You will need a text editor**

Just about any text editor will suffice for creating Python script files.

You can use *Microsoft Notepad, Microsoft WordPad, Microsoft Word,*or just about any word processor if you want to.

**Difference between a script and a program**

**Script:**

Scripts are distinct from the core code of the application, which is usually written in a different language, and are often created or at least modified by the end-user. Scripts are often interpreted from source code or byte code, whereas the applications they control are traditionally compiled to native machine code.

**Program:**

The program has an executable form that the computer can use directly to execute the instructions.

The same program in its human-readable source code form, from which executable programs are derived (e.g., compiled)

**Python**

what is Python? Chances you are asking yourself this. You may have found this book because you want to learn to program but don’t know anything about programming languages. Or you may have heard of programming languages like C, C++, C#, or Java and want to know what Python is and how it compares to “big name” languages. Hopefully I can explain it for you.

**Python concepts**

If you're not interested in the how and whys of Python, feel free to skip to the next chapter. In this chapter I will try to explain to the reader why I think Python is one of the best languages available and why it’s a great one to start programming with.

• Open-source general-purpose language.

• Object Oriented, Procedural, Functional

• Easy to interface with C/ObjC/Java/Fortran

• Easy-ish to interface with C++ (via SWIG)

• Great interactive environment

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages.

* **Python is Interpreted** − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
* **Python is Interactive** − You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.
* **Python is Object-Oriented** − Python supports Object-Oriented style or technique of programming that encapsulates code within objects.
* **Python is a Beginner's Language** − Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

**History of Python**

Python was developed by Guido van Rossum in the late eighties and early nineties at the National Research Institute for Mathematics and Computer Science in the Netherlands.

Python is derived from many other languages, including ABC, Modula-3, C, C++, Algol-68, SmallTalk, and Unix shell and other scripting languages.

Python is copyrighted. Like Perl, Python source code is now available under the GNU General Public License (GPL).

Python is now maintained by a core development team at the institute, although Guido van Rossum still holds a vital role in directing its progress.

**Python Features**

Python's features include −

* **Easy-to-learn** − Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.
* **Easy-to-read** − Python code is more clearly defined and visible to the eyes.
* **Easy-to-maintain** − Python's source code is fairly easy-to-maintain.
* **A broad standard library** − Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.
* **Interactive Mode** − Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.
* **Portable** − Python can run on a wide variety of hardware platforms and has the same interface on all platforms.
* **Extendable** − You can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.
* **Databases** − Python provides interfaces to all major commercial databases.
* **GUI Programming** − Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.
* **Scalable** − Python provides a better structure and support for large programs than shell scripting.

Apart from the above-mentioned features, Python has a big list of good features, few are listed below −

* It supports functional and structured programming methods as well as OOP.
* It can be used as a scripting language or can be compiled to byte-code for building large applications.
* It provides very high-level dynamic data types and supports dynamic type checking.
* IT supports automatic garbage collection.
* It can be easily integrated with C, C++, COM, ActiveX, CORBA, and Java.

**Dynamic vs Static**

Types Python is a dynamic-typed language. Many other languages are static typed, such as C/C++ and Java. A static typed language requires the programmer to explicitly tell the computer what type of “thing” each data value is.

For example, in C if you had a variable that was to contain the price of something, you would have to declare the variable as a “float” type.

This tells the compiler that the only data that can be used for that variable must be a floating-point number, i.e., a number with a decimal point.

If any other data value was assigned to that variable, the compiler would give an error when trying to compile the program.

Python, however, doesn’t require this. You simply give your variables names and assign values to them. The interpreter takes care of keeping track of what kinds of objects your program is using. This also means that you can change the size of the values as you develop the program. Say you have another decimal number (a.k.a. a floating-point number) you need in your program.

With a static typed language, you have to decide the memory size the variable can take when you first initialize that variable. A double is a floating-point value that can handle a much larger number than a normal float (the actual memory sizes depend on the operating environment).

If you declare a variable to be a float but later on assign a value that is too big to it, your program will fail; you will have to go back and change that variable to be a double.

With Python, it doesn’t matter. You simply give it whatever number you want and Python will take care of manipulating it as needed. It even works for derived values.

For example, say you are dividing two numbers. One is a floating-point number and one is an integer. Python realizes that it’s more accurate to keep track of decimals so it automatically calculates the result as a floating-point number.

**Variables**

Variables are nothing but reserved memory locations to store values. This means that when you create a variable you reserve some space in memory.

Based on the data type of a variable, the interpreter allocates memory and decides what can be stored in the reserved memory. Therefore, by assigning different data types to variables, you can store integers, decimals or characters in these variables.

**Standard Data Types**

The data stored in memory can be of many types. For example, a person's age is stored as a numeric value and his or her address is stored as alphanumeric characters. Python has various standard data types that are used to define the operations possible on them and the storage method for each of them.

Python has five standard data types −

* Numbers
* String
* List
* Tuple
* Dictionary

## Python Numbers

Number data types store numeric values. Number objects are created when you assign a value to them

## Python Strings

Strings in Python are identified as a contiguous set of characters represented in the quotation marks. Python allows for either pairs of single or double quotes. Subsets of strings can be taken using the slice operator ([ ] and [:] ) with indexes starting at 0 in the beginning of the string and working their way from -1 at the end.

## Python Lists

Lists are the most versatile of Python's compound data types. A list contains items separated by commas and enclosed within square brackets ([]). To some extent, lists are similar to arrays in C. One difference between them is that all the items belonging to a list can be of different data type.

The values stored in a list can be accessed using the slice operator ([ ] and [:]) with indexes starting at 0 in the beginning of the list and working their way to end -1. The plus (+) sign is the list concatenation operator, and the asterisk (\*) is the repetition operator.

## Python Tuples

A tuple is another sequence data type that is similar to the list. A tuple consists of a number of values separated by commas. Unlike lists, however, tuples are enclosed within parentheses.

The main differences between lists and tuples are: Lists are enclosed in brackets ( [ ] ) and their elements and size can be changed, while tuples are enclosed in parentheses ( ( ) ) and cannot be updated. Tuples can be thought of as **read-only** lists.

## Python Dictionary

Python's dictionaries are kind of hash table type. They work like associative arrays or hashes found in Perl and consist of key-value pairs. A dictionary key can be almost any Python type, but are usually numbers or strings. Values, on the other hand, can be any arbitrary Python object.

Dictionaries are enclosed by curly braces ({ }) and values can be assigned and accessed using square braces ([]).

**Different modes in python**

Python has two basic modes: normal and interactive.

The normal mode is the mode where the scripted and finished .py files are run in the Python interpreter.

Interactive mode is a command line shell which gives immediate feedback for each statement, while running previously fed statements in active memory. As new lines are fed into the interpreter, the fed program is evaluated both in part and in whole

**Some Python Libraries:**

1. Pandas
2. Numpy
3. Pymysql
4. Scikit-learn

**Pandas:**

* Pandas provide us with many Series and Data Frames. It allows you to easily organize, explore, represent, and manipulate data.
* Smart alignment and indexing featured in Pandas offer you a perfect organization and data labeling.
* Pandas has some special features that allow you to handle missing data or value with a proper measure.
* This package offers you such a clean code that even people with no or basic knowledge of programming can easily work with it.
* It provides a collection of built-in tools that allows you to both read and write data in different web services, data-structure, and databases as well.
* Pandas can support JSON, Excel, CSV, HDF5, and many other formats. In fact, you can merge different databases at a time with Pandas.

**Numpy:**

* Arrays of Numpy offer modern mathematical implementations on huge amount of data. Numpy makes the execution of these projects much easier and hassle-free.
* Numpy provides masked arrays along with general array objects. It also comes with functionalities such as manipulation of logical shapes, discrete Fourier transform, general linear algebra, and many more.
* While you change the shape of any N-dimensional arrays, Numpy will create new arrays for that and delete the old ones.
* This python package provides useful tools for integration. You can easily integrate Numpy with programming languages such as C, C++, and Fortran code.
* Numpy provides such functionalities that are comparable to MATLAB. They both allow users to get faster with operations.

**Pymysql:**

* PyMySQL is a database connector for Python, libraries to enable Python programs to talk to a MySQL server.
* Access to the port settings through Python properties.
* PyMySQL is a pure Python MySQL driver, first written as a rough port of the MySQL-Python driver.
* PyMySQL meets all of criterion for a driver.
* It is fully open source, hosted on Github, released on Pypi, is actively maintained.
* It is written in pure Python so is eventlet-monkeypatch compatible, and is fully Python 3 compatible.

**Scikit-Learn:**

* The random module is a simple and efficient tool for predictive data analysis
* Accessible to everybody, and reusable in various contexts
* The library is focused on modeling data. It is not focused on loading, manipulating and summarizing data. For these features, refer to NumPy and Pandas.
* The functionality that scikit-learn provides include:
* Regression, including Linear and Logistic Regression
* Classification, including K-Nearest Neighbors
* Clustering, including K-Means and K-Means++
* Model selection
* Preprocessing, including Min-Max Normalization

**Python class and objects**

These are the building blocks of OOP. class creates a new object. This object can be anything, whether an abstract data concept or a model of a physical object, e.g., a chair. Each class has individual characteristics unique to that class, including variables and methods. Classes are very powerful and currently “the big thing” in most programming languages. Hence, there are several chapters dedicated to OOP later in the book.

The class is the most basic component of object-oriented programming. Previously, you learned how to use functions to make your program do something.

Now will move into the big, scary world of Object-Oriented Programming (OOP). To be honest, it took me several months to get a handle on objects.

When I first learned C and C++, I did great; functions just made sense for me.

Having messed around with BASIC in the early ’90s, I realized functions were just like subroutines so there wasn’t much new to learn.

However, when my C++ course started talking about objects, classes, and all the new features of OOP, my grades definitely suffered.

Once you learn OOP, you’ll realize that it’s actually a pretty powerful tool. Plus many Python libraries and APIs use classes, so you should at least be able to understand what the code is doing.

One thing to note about Python and OOP: it’s not mandatory to use objects in your code in a way that works best; maybe you don’t need to have a full-blown class with initialization code and methods to just return a calculation. With Python, you can get as technical as you want.

As you’ve already seen, Python can do just fine with functions. Unlike languages such as Java, you aren’t tied down to a single way of doing things; you can mix functions and classes as necessary in the same program. This lets you build the code

Objects are an encapsulation of variables and functions into a single entity. Objects get their variables and functions from classes. Classes are essentially a template to create your objects.

Here’s a brief list of Python OOP ideas:

• The class statement creates a class object and gives it a name. This creates a new namespace.

• Assignments within the class create class attributes. These attributes are accessed by qualifying the name using dot syntax: ClassName.Attribute.

• Class attributes export the state of an object and its associated behavior. These attributes are shared by all instances of a class.

• Calling a class (just like a function) creates a new instance of the class.

This is where the multiple copies part comes in.

• Each instance gets ("inherits") the default class attributes and gets its own namespace. This prevents instance objects from overlapping and confusing the program.

• Using the term self identifies a particular instance, allowing for per-instance attributes. This allows items such as variables to be associated with a particular instance.

**Inheritance**

First off, classes allow you to modify a program without really making changes to it.

To elaborate, by subclassing a class, you can change the behavior of the program by simply adding new components to it rather than rewriting the existing components.

As we’ve seen, an instance of a class inherits the attributes of that class.

However, classes can also inherit attributes from other classes. Hence, a subclass inherits from a superclass allowing you to make a generic superclass that is specialized via subclasses.

The subclasses can override the logic in a superclass, allowing you to change the behavior of your classes without changing the superclass at all.

Operator Overloads

Operator overloading simply means that objects that you create from classes can respond to actions (operations) that are already defined within Python, such as addition, slicing, printing, etc.

Even though these actions can be implemented via class methods, using overloading ties the behavior closer to Python’s object model and the object interfaces are more consistent to Python’s built-in objects, hence overloading is easier to learn and use.

User-made classes can override nearly all of Python’s built-in operation methods

**Exceptions**

I’ve talked about exceptions before but now I will talk about them in depth. Essentially, exceptions are events that modify program’s flow, either intentionally or due to errors.

They are special events that can occur due to an error, e.g. trying to open a file that doesn’t exist, or when the program reaches a marker, such as the completion of a loop.

Exceptions, by definition, don’t occur very often; hence, they are the "exception to the rule" and a special class has been created for them. Exceptions are everywhere in Python.

Virtually every module in the standard Python library uses them, and Python itself will raise them in a lot of different circumstances.

Here are just a few examples:

• Accessing a non−existent dictionary key will raise a KeyError exception.

• Searching a list for a non−existent value will raise a ValueError exception

. • Calling a non−existent method will raise an AttributeError exception.

• Referencing a non−existent variable will raise a NameError exception.

• Mixing datatypes without coercion will raise a TypeError exception.

One use of exceptions is to catch a fault and allow the program to continue working; we have seen this before when we talked about files.

This is the most common way to use exceptions. When programming with the Python command line interpreter, you don’t need to worry about catching exceptions.

Your program is usually short enough to not be hurt too much if an exception occurs.

Plus, having the exception occur at the command line is a quick and easy way to tell if your code logic has a problem.

However, if the same error occurred in your real program, it will fail and stop working. Exceptions can be created manually in the code by raising an exception.

It operates exactly as a system-caused exceptions, except that the programmer is doing it on purpose. This can be for a number of reasons. One of the benefits of using exceptions is that, by their nature, they don’t put any overhead on the code processing.

Because exceptions aren’t supposed to happen very often, they aren’t processed until they occur.

Exceptions can be thought of as a special form of the if/elif statements. You can realistically do the same thing with if blocks as you can with exceptions.

However, as already mentioned, exceptions aren’t processed until they occur; if blocks are processed all the time.

Proper use of exceptions can help the performance of your program.

The more infrequent the error might occur, the better off you are to use exceptions; using if blocks requires Python to always test extra conditions before continuing.

Exceptions also make code management easier: if your programming logic is mixed in with error-handling if statements, it can be difficult to read, modify, and debug your program.

User-Defined Exceptions

I won’t spend too much time talking about this, but Python does allow for a programmer to create his own exceptions.

You probably won’t have to do this very often but it’s nice to have the option when necessary.

However, before making your own exceptions, make sure there isn’t one of the built-in exceptions that will work for you.

They have been "tested by fire" over the years and not only work effectively, they have been optimized for performance and are bug-free.

Making your own exceptions involves object-oriented programming, which will be covered in the next chapter

. To make a custom exception, the programmer determines which base exception to use as the class to inherit from, e.g., making an exception for negative numbers or one for imaginary numbers would probably fall under the Arithmetic Error exception class.

To make a custom exception, simply inherit the base exception and define what it will do.

**Python modules**

Python allows us to store our code in files (also called modules). This is very useful for more serious programming, where we do not want to retype a long function definition from the very beginning just to change one mistake. In doing this, we are essentially defining our own modules, just like the modules defined already in the Python library.

To support this, Python has a way to put definitions in a file and use them in a script or in an interactive instance of the interpreter. Such a file is called a module; definitions from a module can be imported into other modules or into the main module.

**Testing code**

As indicated above, code is usually developed in a file using an editor.

To test the code, import it into a Python session and try to run it.

Usually there is an error, so you go back to the file, make a correction, and test again.

This process is repeated until you are satisfied that the code works. T

he entire process is known as the development cycle.

There are two types of errors that you will encounter. Syntax errors occur when the form of some command is invalid.

This happens when you make typing errors such as misspellings, or call something by the wrong name, and for many other reasons. Python will always give an error message for a syntax error.

**Functions in Python**

It is possible, and very useful, to define our own functions in Python. Generally speaking, if you need to do a calculation only once, then use the interpreter. But when you or others have need to perform a certain type of calculation many times, then define a function.

You use functions in programming to bundle a set of instructions that you want to use repeatedly or that, because of their complexity, are better self-contained in a sub-program and called when needed. That means that a function is a piece of code written to carry out a specified task.

## To carry out that specific task, the function might or might not need multiple inputs. When the task is carried out, the function can or cannot return one or more values.

## There are three types of functions in python:

## help(),min(),print().

## Python Namespace

Generally speaking, a **namespace** (sometimes also called a context) is a naming system for making names unique to avoid ambiguity. Everybody knows a namespacing system from daily life, i.e., the naming of people in first name and family name (surname).

An example is a network: each network device (workstation, server, printer, ...) needs a unique name and address. Yet another example is the directory structure of file systems.

The same file name can be used in different directories, the files can be uniquely accessed via the pathnames. Many programming languages use namespaces or contexts for identifiers. An identifier defined in a namespace is associated with that namespace.

This way, the same identifier can be independently defined in multiple namespaces. (Like the same file names in different directories) Programming languages, which support namespaces, may have different rules that determine to which namespace an identifier belongs.

Namespaces in Python are implemented as Python dictionaries, this means it is a mapping from names (keys) to objects (values). The user doesn't have to know this to write a Python program and when using namespaces.

Some namespaces in Python:

* **global names** of a module
* **local names** in a function or method invocation
* **built-in names**: this namespace contains built-in functions (e.g. abs(), cmp(), ...) and built-in exception names

**Garbage Collection**

Garbage Collector exposes the underlying memory management mechanism of Python, the automatic garbage collector. The module includes functions for controlling how the collector operates and to examine the objects known to the system, either pending collection or stuck in reference cycles and unable to be freed.

**Python XML Parser**

XML is a portable, open-source language that allows programmers to develop applications that can be read by other applications, regardless of operating system and/or developmental language.

What is XML? The Extensible Markup Language XML is a markup language much like HTML or SGML.

This is recommended by the World Wide Web Consortium and available as an open standard.

XML is extremely useful for keeping track of small to medium amounts of data without requiring a SQL-based backbone.

XML Parser Architectures and APIs The Python standard library provides a minimal but useful set of interfaces to work with XML.

The two most basic and broadly used APIs to XML data are the SAX and DOM interfaces.

Simple API for XML SAX: Here, you register callbacks for events of interest and then let the parser proceed through the document.

This is useful when your documents are large or you have memory limitations, it parses the file as it reads it from disk and the entire file is never stored in memory.

Document Object Model DOM API: This is a World Wide Web Consortium recommendation wherein the entire file is read into memory and stored in a hierarchical tree − based form to represent all the features of an XML document.

SAX obviously cannot process information as fast as DOM can when working with large files. On the other hand, using DOM exclusively can really kill your resources, especially if used on a lot of small files.

SAX is read-only, while DOM allows changes to the XML file. Since these two different APIs literally complement each other, there is no reason why you cannot use them both for large projects.

**Python Web Frameworks**

A web framework is a code library that makes a developer's life easier when building reliable, scalable and maintainable web applications.

## Why are web frameworks useful?

Web frameworks encapsulate what developers have learned over the past twenty years while programming sites and applications for the web. Frameworks make it easier to reuse code for common HTTP operations and to structure projects so other developers with knowledge of the framework can quickly build and maintain the application.

Common web framework functionality

Frameworks provide functionality in their code or through extensions to perform common operations required to run web applications. These common operations include:

1. URL routing
2. HTML, XML, JSON, and other output format templating
3. Database manipulation
4. Security against Cross-site request forgery (CSRF) and other attacks
5. Session storage and retrieval

Not all web frameworks include code for all of the above functionality. Frameworks fall on the spectrum from executing a single use case to providing every known web framework feature to every developer. Some frameworks take the "batteries-included" approach where everything possible comes bundled with the framework while others have a minimal core package that is amenable to extensions provided by other packages.

## Comparing web frameworks

There is also a repository called [compare-python-web-frameworks](https://github.com/mattmakai/compare-python-web-frameworks) where the same web application is being coded with varying Python web frameworks, templating engines and object.

## Web framework resources

* When you are learning how to use one or more web frameworks it's helpful to have an idea of what the code under the covers is doing.
* Frameworks is a really well-done short video that explains how to choose between web frameworks. The author has some particular opinions about what should be in a framework. For the most part I agree although I've found sessions and database ORMs to be a helpful part of a framework when done well.
* what is a web framework? is an in-depth explanation of what web frameworks are and their relation to web servers.
* Django vs Flash vs Pyramid: Choosing a Python web framework contains background information and code comparisons for similar web applications built in these three big Python frameworks.
* This fascinating blog post takes a look at the code complexity of several Python web frameworks by providing visualizations based on their code bases.
* Python’s web frameworks benchmarks is a test of the responsiveness of a framework with encoding an object to JSON and returning it as a response as well as retrieving data from the database and rendering it in a template. There were no conclusive results but the output is fun to read about nonetheless.
* What web frameworks do you use and why are they awesome? is a language agnostic Reddit discussion on web frameworks. It's interesting to see what programmers in other languages like and dislike about their suite of web frameworks compared to the main Python frameworks.
* This user-voted question & answer site asked "What are the best general purpose Python web frameworks usable in production?". The votes aren't as important as the list of the many frameworks that are available to Python developers.

## Web frameworks learning checklist

1. Choose a major Python web framework (Django or Flask are recommended) and stick with it. When you're just starting it's best to learn one framework first instead of bouncing around trying to understand every framework.
2. Work through a detailed tutorial found within the resources links on the framework's page.
3. Study open-source examples built with your framework of choice so you can take parts of those projects and reuse the code in your application.
4. Build the first simple iteration of your web application then go to the [deployment](https://www.fullstackpython.com/deployment.html) section to make it accessible on the web.

**2. SYSTEM STUDY**

**2.1 FEASIBILITY STUDY**

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

Three key considerations involved in the feasibility analysis are

* ECONOMICAL FEASIBILITY
* TECHNICAL FEASIBILITY
* SOCIAL FEASIBILITY

**ECONOMICAL FEASIBILITY**

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

### TECHNICAL FEASIBILITY

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

**SOCIAL FEASIBILITY**

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

### SYSTEM TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub-assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the

Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

**TYPES OF TESTS**

**Unit testing**

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

**Integration testing**

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

**Functional test**

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input : identified classes of valid input must be accepted.

Invalid Input : identified classes of invalid input must be rejected.

Functions : identified functions must be exercised.

Output : identified classes of application outputs must be exercised.

Systems/Procedures: interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

**SYSTEM TEST**

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration-oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

**White Box Testing**

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

**Black Box Testing**

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box .you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

**6.1 Unit Testing:**

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

**Test strategy and approach**

Field testing will be performed manually and functional tests will be written in detail.

**Test objectives**

* All field entries must work properly.
* Pages must be activated from the identified link.
* The entry screen, messages and responses must not be delayed.

**Features to be tested**

* Verify that the entries are of the correct format
* No duplicate entries should be allowed
* All links should take the user to the correct page.

# 6.2 Integration Testing

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

**Test Results:** All the test cases mentioned above passed successfully. No defects encountered.

**6.3 Acceptance Testing**

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

**Test Results:** All the test cases mentioned above passed successfully. No defects encountered

**CONCLUSION**

Chat bots are a thing of the future which is yet to uncover its potential but with its rising popularity and craze among companies, they are bound to stay here for long types of chat bots being introduced, it is of great excitement to witness the growth of a new domain in technology while surpassing the previous threshold. We are inventing the system because of the need of the increasing population of our country. As we know if we want to join in a college we need to go to colleges and from the college admins, college staff we need to get the all information about college structure in the sense how would be the faculty lecture, How many branches in college, Hostel fee structure, College fee structure, How many seats available for specific branch, About college environment, Available Specializations in a department like Data Science, Machine Learning in CSE , Infrastructure of college like labs, library, College placements, Internships and college professors, Special events that occur in our college i.e. college fests, College extra circular activities updates like annual day, sports events, technical events, Achievements of college, NIRF Ranking, Naas rating to college etc.., Thus, the college chatbot will give the assistance to the students even the students no need to visit the colleges.

**REFERENCES**

[1]A. Allen, ‘‘Morphing telemedicine-telecare-telehealth-ehealth,’’ Telemed Today, Special, no. 2000, 2000.

[2]S. L. Murphy, J. Xu, K. D. Kochanek, and E. Arias, ‘‘Mortality in the United States, 2017,’’ *NCHS Data Brief*, no. 328, pp. 1–8, Nov. 2018.

[3]K. Myers, P. Berry, J. Blythe, K. Conley, M. Gervasio, D. L. McGuinness, D. Morley, A. Pfeffer, M. Pollack, and M. Tambe, ‘‘An intelligent personal assistant for task and time management,’’ *AI Mag.*, vol. 28, no. 2, p. 47, 2007.

[4] Flora Amato, Stefano Marrone, “Chatbots meet eHealth: automat zing healthcare”, proceeding of diet, May-2018.

[5] BenildaEleonor V. Comendador, “Pharmabot: A pediatric generic Medicine consultant Chatbot”, proceeding of the JACE, April 2015.

[6] Divya, Indumathi, Ishwarya, Priyasankari, “A SelfDiagnosis Medical Chatbot Using Artificial Intelligence”, proceeding MAT Journal, October-2017.

[7] Tobias Kowatsch,” Text-based Healthcare Chatbots Supporting Patient and Health”, 01 October 2017.

[8] Chin-Yuan Huang, Ming-Chin Yang, Chin-Yu Huang, “A Chatbot-supported Smart Wireless Interactive Healthcare System for Weight Control and Health Promotion”, proceeding of the IEEE, April-2018.

[9] Boukricha, H., Wachsmuth, I.: Modeling Empathy for a Virtual Human: How, When and to What Extent. The 10th International Conference on Autonomous Agents and Multiagent Systems-Volume 3. International Foundation for Autonomous Agents and Multiagent Systems, 2011., pp. 1135–1136

[10] Agarwal, R., Gao, G., DesRoches, C., et al.: The Digital Transformation of Healthcare: Current Status and the Road Ahead. Information Systems Research 21, 796-809 (2010).

[11] Aron, A., Aron, E.N., Smollan, D.: Inclusion of Other in the Self Scale and the structure of interpersonal closeness. Journal of Personality and Social Psychology 63, 596-612 (1992).

[12] Bickmore, T., Cassell, J.: Social Dialogue with Embodied Conversational Agents. In: Kuppevelt, J.C.J., Bernsen, N.O., Dybkjær, L. (eds.) Advances in Natural Multimodal Dialogue Systems, vol. 30, pp. 23–54. Springer, Dordrecht (2005).

[13] Bickmore, T., Gruber, A., Picard, R.: Establishing the computer–patient working alliance in automated health behavior change interventions. Patient Education and Counseling 59, 21-30 (2005).