**A PROJECT REPORT ON**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Title of the Project\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**Submitted to \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* University for the partial fulfillment of the Requirement for the**

**Award of Degree for**

**\*\*\*\*\*\*\*\*\*\*\*Course Name\*\*\*\*\*\*\*\*\*\*\*\*\***

**Done By**

**Mr. /Miss \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*Institute of Management and Computer Science**

**CERTIFICATE**

**This is to certify that Mr., /Miss \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* bearing Roll**

**No. \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* have developed software project**

**Titled \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* For \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Software**

**Solutions as Partial fulfillment for the award of the Degree of**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**Head of the Department Principal**

**\*\*\*\*\*\*College Name\*\*\*\*\*\***

**External**

**ACKNOWLEDGEMENT**

At every outset I express my gratitude to almighty lord for showering his grace and blessings upon me to complete this project.

Although our name appears on the cover of this book, many people had contributed in some form or the other form to this project Development. We could not done this project without the assistance or support of each of the following we thank you all.

I wish to place on my record my deep sense of gratitude to my project guide, **Mr. \*\*\*\*\*\*, \*\*\*\*\* Software Solutions,** for his constant motivation and valuable help through the project work. Express my gratitude to **Mr. \*\*\*\*\*\***, Director of \*\*\*\*\*\*\* **Institute of Management & Computer Sciences** for his valuable suggestions and advices throughout the \*\*\*\*\* course. I also extend my thanks to other Faculties for their Cooperation during my Course.

Finally I would like to thank my friends for their cooperation to complete this project.

\*\*\*\*\*\*\*Your Name\*\*\*\*\*\*\*\*\*

**ABSTRACT:**

Brain tumor segmentation is an important task in medical image processing. Early diagnosis of brain tumors plays an important role in improving treatment possibilities and increases the survival rate of the patients. Manual segmentation of the brain tumors for cancer diagnosis, from large amount of MRI images generated in clinical routine, is a difficult and time consuming task. There is a need for automatic brain tumor image segmentation. The purpose of this paper is to provide a review of MRI-based brain tumor segmentation methods. Recently, automatic

segmentation using deep learning methods proved popular since these methods achieve the state-of-the-art results and can address this problem better than other methods. Deep learning methods can also enable efficient processing and objective evaluation of the large amounts of MRI-based image data. There are number of existing review papers, focusing on traditional methods for MRI-based brain tumor image segmentation. Different than others, in this paper, we focus on the recent trend of deep learning methods in this field. First, an introduction to brain tumors and methods for brain tumor segmentation is given. Then, the state-of-the-art algorithms with a focus on recent trend of deep learning methods are discussed. Finally, an assessment of the current state is presented and future developments to standardize MRI-based brain tumor segmentation methods into daily clinical routine are addressed.

**Index**

1. **INTRODUCTION**
   1. INTRODUCTION TO PROJECT
   2. ORGANIZATION PROFILE
   3. PURPOSE OF THE SYSTEM
   4. PROBLEMS IN EXISTING SYSTEM
2. **SYSTEM ANALYSIS**
   1. INTRODUCTION
   2. STUDY OF THE SYSTEM
   3. HARDWARE AND SOFTWARE REQUIRMENTS
   4. PROPOSED SYSTEM
   5. INPUT AND OUTPUT
   6. PROCESS MODULES USED WITH JUSTIFICATION
3. **FEASIBILITY REPORT**
   1. TECHNICAL FEASIBILITY
   2. OPERATIONAL FEASIBILITY
   3. ECONOMICAL FEASIBILTY
4. **SOFTWARE REQUIREMENT SPECIFICATIONS**
   1. FUNCTIONAL REQUIREMENTS
   2. PERFORMANCE REQUIREMENTS

**5. SELECTED SOFTWARE**

5.1INTRODUCTION TO PYTHON

5.2PYTHON

**6. SYSTEM DESIGN**

6.1 INTRODUCTION

6.2 SYSTEM WORK FLOW

* 1. UML DIAGRAMS

**7. OUTPUT SCREENS**

**8. SYSTEM TESTING AND IMPLEMENTATION**

8.1INTRODUCTION

* 1. IMPLEMENTATION

**9. SYSTEM SECURITY**

9.1INTRODUCTION

9.2SECURITY IN SOFTWARE

**10. CONCLUSION**

**11. FUTURE ENHANCEMENTS**

**INTRODUCTION:**

Cancer can be defined as the uncontrolled, unnatural growth and division of the cells in the body. Occurrence, as a mass, of these unnatural cell growth and division in the brain tissue is called a brain tumor. While brain tumors are not very common, they are one of the most lethal cancers1.

Depending on their initial origin, brain tumors can be considered as either primary brain tumors or metastatic brain tumors. In primary ones, the origin of the cells are brain tissue cells, where in metastatic ones cells become cancerous at any other part of the body and spread into the brain. Gliomas are type of brain tumors that originate from glial cells.

They are the main type of brain tumors that current brain tumor segmentation research focuses on. The term glioma is a general term that is used to describe different types of gliomas ranging from low-grade gliomas like astrocytomas and oligodendrogliomas to the high grade (grade IV) glioblastoma multiform (GBM), which is the most aggressive and the most common primary malignant brain tumor2. Surgery, chemotherapy and radiotherapy are the techniques used, usually in combination, to treat gliomas3.

Early diagnosis of gliomas plays an important role in improving treatment possibilities. Medical Imaging techniques such as Computed Tomography (CT), Single-Photon Emission Computed Tomography (SPECT), Positron Emission Tomography (PET), Magnetic Resonance Spectroscopy (MRS) and Magnetic Resonance Imaging (MRI) are all used to provide valuable information about shape, size, location and metabolism of brain tumors assisting in diagnosis. While these modalities are used in combination to provide the highest detailed information about the brain tumors, due to its good soft tissue contrast and widely availability MRI is considered as the standard technique. MRI is a non-invasive in vivo imaging technique that uses radio frequency signals to excite target tissues to produce their internal images under the influence of a very powerful magnetic field.

Images of different MRI sequences are generated by altering excitation and repetition times during image acquisition. These different MRI modalities produce different types of tissue contrast images, thus providing valuable structural information and enabling diagnosis and segmentation of tumors along with their subregions4. Four standard MRI modalities used for glioma diagnosis include T1-weighted MRI (T1), T2-weighted MRI (T2), T1-weighted MRI with gadolinium contrast enhancement (T1-Gd) and Fluid Attenuated Inversion Recovery (FLAIR) (see Fig. 1). During MRI acquisition, although can vary from device to device, around one hundred and fifty slices of 2D images are produced to represent the 3D brain volume. Furthermore, when the slices from the required standard modalities are combined for diagnosis the data becomes very populated and complicated. Generally, T1 images are used for distinguishing healthy tissues, whereas T2 images are used to delineate the edema region which produces bright signal on the image. In T1-Gd images, the tumor border can easily be distinguished by the bright signal of the accumulated contrast agent (gadolinium ions) in the active cell region of the tumor tissue.

Since necrotic cells do not interact with the contrast agent, they can be observed by hypo intense part of the tumor core making it possible to easily segment them from the active cell region on the same sequence. In FLAIR images, signal of water molecules are suppressed which helps in distinguishing edema region from the Cerebrospinal Fluid (CSF). Before applying any therapy, it is crucial to segment the tumor in order to protect healthy tissues while damaging and destroying tumor cells during the therapy. Brain tumor segmentation involves diagnosing, delineating and separating tumor tissues, such as active cells, necrotic core and edema (Fig. 2) from normal brain tissues including Gray Matter (GM), White Matter (WM) and CSF. In current clinical routine, this task involves manual annotation and segmentation of large amount of multimodal MRI images. However, since manual segmentation is a very time consuming procedure, development of robust automatic segmentation methods, to provide efficient and objective segmentation, became an interesting and popular research area in recent years5. Current high segmentation performances obtained by deep learning methods make them good candidates for achieving this task. The rest of the paper is organized as follows: First we briefly review methods for brain tumor image segmentation in section 2. Then, in section 3, we especially focus on methods based on deep learning algorithms, which provide the state-of-the-art results in recent years. Inparticular, we compare designs of different deep learning methods and their performances. Finally, in conclusions, we assess the current state-of-the-art and provide future directions for development.

**ORGANIZATION PROFILE**

Software Solutions is an IT solution provider for a dynamic environment where business and technology strategies converge. Their approach focuses on new ways of business combining IT innovation and adoption while also leveraging an organization’s current IT assets. Their work with large global corporations and new products or services and to implement prudent business and technology strategies in today’s environment.

**RANGE OF EXPERTISE INCLUDES:**

* Software Development Services
* Engineering Services
* Systems Integration
* Customer Relationship Management
* Product Development
* Electronic Commerce
* Consulting
* IT Outsourcing

We apply technology with innovation and responsibility to achieve two broad objectives:

* Effectively address the business issues our customers face today.
* Generate new opportunities that will help them stay ahead in the future.

**THIS APPROACH RESTS ON:**

* A strategy where we architect, integrate and manage technology services and solutions - we call it AIM for success.
* A robust offshore development methodology and reduced demand on customer resources.
* A focus on the use of reusable frameworks to provide cost and times benefits.

They combine the best people, processes and technology to achieve excellent results - consistency. We offer customers the advantages of:

**SPEED:**

They understand the importance of timing, of getting there before the competition. A rich portfolio of reusable, modular frameworks helps jump-start projects. Tried and tested methodology ensures that we follow a predictable, low - risk path to achieve results. Our track record is testimony to complex projects delivered within and evens before schedule.

**EXPERTISE:**

Our teams combine cutting edge technology skills with rich domain expertise. What’s equally important - they share a strong customer orientation that means they actually start by listening to the customer. They’re focused on coming up with solutions that serve customer requirements today and anticipate future needs.

**A FULL SERVICE PORTFOLIO:**

They offer customers the advantage of being able to Architect, integrate and manage technology services. This means that they can rely on one, fully accountable source instead of trying to integrate disparate multi vendor solutions.

**SERVICES:**

Xxx is providing its services to companies which are in the field of production, quality control etc with their rich expertise and experience and information technology they are in best position to provide software solutions to distinct business requirements

**Software Requirements:**

* OS: Windows
* Python IDE : python 2.7.x
* Pycharm IDE
* Browser: Firefox 28
* Selenium: Selenium 2.4 Above

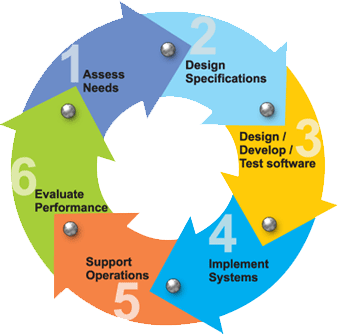
**Hardware Requirements:**

* RAM: 4GB
* Processor:Intel i3
* Hard Disk: 500GB

**2.1 INTRODUCTION**

**Software Development Life Cycle:-**

There is various software development approaches defined and designed which are used/employed during development process of software, these approaches are also referred as "Software Development Process Models". Each process model follows a particular life cycle in order to ensure success in process of software development.



**Requirements**

Business requirements are gathered in this phase.  This phase is the main focus of the project managers and stake holders.  Meetings with managers, stake holders and users are held in order to determine the requirements.  Who is going to use the system?  How will they use the system?  What data should be input into the system?  What data should be output by the system?  These are general questions that get answered during a requirements gathering phase.  This produces a nice big list of functionality that the system should provide, which describes functions the system should perform, business logic that processes data, what data is stored and used by the system, and how the user interface should work.  The overall result is the system as a whole and how it performs, not how it is actually going to do it.

**Design**

The software system design is produced from the results of the requirements phase.  Architects have the ball in their court during this phase and this is the phase in which their focus lies.  This is where the details on how the system will work is produced.  Architecture, including hardware and software, communication, software design (UML is produced here) are all part of the deliverables of a design phase.

**Implementation**

Code is produced from the deliverables of the design phase during implementation, and this is the longest phase of the software development life cycle.  For a developer, this is the main focus of the life cycle because this is where the code is produced.  Implementation my overlap with both the design and testing phases.  Many tools exists (CASE tools) to actually automate the production of code using information gathered and produced during the design phase.

**Testing**

During testing, the implementation is tested against the requirements to make sure that the product is actually solving the needs addressed and gathered during the requirements phase.  Unit tests and system/acceptance tests are done during this phase.  Unit tests act on a specific component of the system, while system tests act on the system as a whole.

So in a nutshell, that is a very basic overview of the general software development life cycle model.  Now let’s delve into some of the traditional and widely used variations.

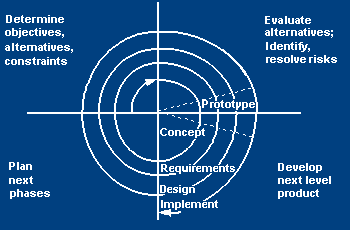
**SDLC METHDOLOGIES**

This document play a vital role in the development of life cycle (SDLC) as it describes the complete requirement of the system. It means for use by developers and will be the basic during testing phase. Any changes made to the requirements in the future will have to go through formal change approval process.

**SPIRAL MODEL** was defined by Barry Boehm in his 1988 article, “A spiral Model of Software Development and Enhancement. This model was not the first model to discuss iterative development, but it was the first model to explain why the iteration models.

As originally envisioned, the iterations were typically 6 months to 2 years long. Each phase starts with a design goal and ends with a client reviewing the progress thus far. Analysis and engineering efforts are applied at each phase of the project, with an eye toward the end goal of the project.

**The following diagram shows how a spiral model acts like:**



The steps for Spiral Model can be generalized as follows:

* The new system requirements are defined in as much details as possible. This usually involves interviewing a number of users representing all the external or internal users and other aspects of the existing system.
* A preliminary design is created for the new system.
* A first prototype of the new system is constructed from the preliminary design. This is usually a scaled-down system, and represents an approximation of the characteristics of the final product.
* A second prototype is evolved by a fourfold procedure:

1. Evaluating the first prototype in terms of its strengths, weakness, and risks.
2. Defining the requirements of the second prototype.
3. Planning an designing the second prototype.
4. Constructing and testing the second prototype.

* At the customer option, the entire project can be aborted if the risk is deemed too great. Risk factors might involved development cost overruns, operating-cost miscalculation, or any other factor that could, in the customer’s judgment, result in a less-than-satisfactory final product.
* The existing prototype is evaluated in the same manner as was the previous prototype, and if necessary, another prototype is developed from it according to the fourfold procedure outlined above.
* The preceding steps are iterated until the customer is satisfied that the refined prototype represents the final product desired.
* The final system is constructed, based on the refined prototype.
* The final system is thoroughly evaluated and tested. Routine maintenance is carried on a continuing basis to prevent large scale failures and to minimize down time.

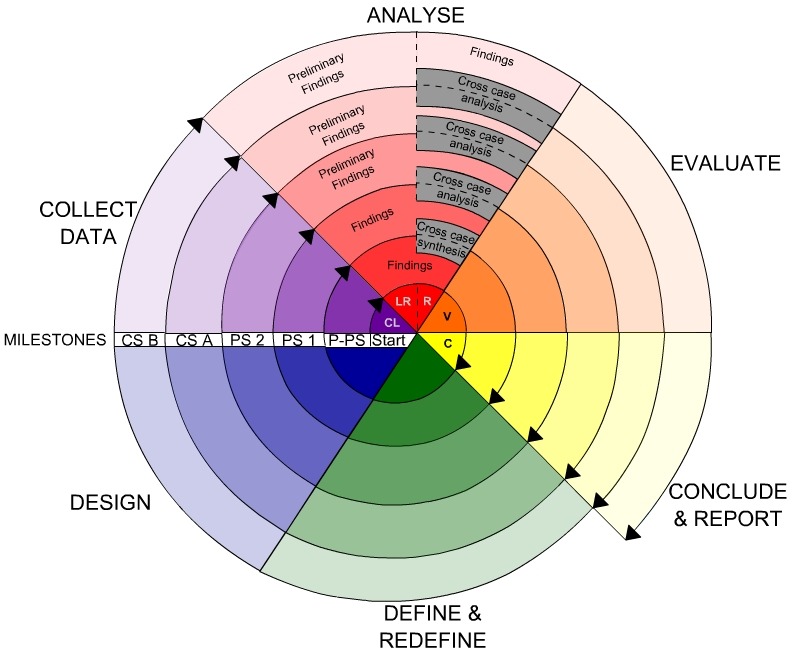
**2.2 STUDY OF THE SYSTEM**

In the flexibility of uses the interface has been developed a graphics concepts in mind, associated through a browser interface. The GUI’s at the top level has been categorized as follows

1. Administrative User Interface Design
2. The Operational and Generic User Interface Design

The administrative user interface concentrates on the consistent information that is practically, part of the organizational activities and which needs proper authentication for the data collection. The Interface helps the administration with all the transactional states like data insertion, data deletion, and data updating along with executive data search capabilities.

The operational and generic user interface helps the users upon the system in transactions through the existing data and required services. The operational user interface also helps the ordinary users in managing their own information helps the ordinary users in managing their own information in a customized manner as per the assisted flexibilities.



**3. Fundamental Concepts on (Domain)**

What is Cloudera?

Cloudera is revolutionizing enterprise data management by offering the first unified Platform for Big Data: The Enterprise Data Hub. Cloudera offers enterprises one place to store, process, and analyze all their data, empowering them to extend the value of existing investments while enabling fundamental new ways to derive value from their data.

Why do customers choose Cloudera?

Cloudera was the first commercial provider of python-related software and services and has the most customers with enterprise requirements, and the most experience supporting them, in the industry. Cloudera’s combined offering of differentiated software (open and closed source), support, training, professional services, and indemnity brings customers the greatest business value, in the shortest amount of time, at the lowest TCO.

**Cloud Computing**

Overview Cloud Computing provides us a means by which we can access the applications as utilities, over the Internet. It allows us to create, configure, and customize applications online.

What is Cloud? The term Cloud refers to a Network or Internet. In other words, we can say that Cloud is something, which is present at remote location. Cloud can provide services over network, i.e., on public networks or on private networks, i.e., WAN, LAN or VPN.

Applications such as e-mail, web conferencing, customer relationship management (CRM),all run in cloud. What is Cloud Computing? Cloud Computing refers to manipulating, configuring, and accessing the applications online. It offers online data storage, infrastructure and application



We need not to install a piece of software on our local PC and this is how the cloud computing overcomes platform dependency issues. Hence, the Cloud Computing is making our business application mobile and collaborative. Basic Concepts There are certain services and models working behind the scene making the cloud computing feasible and accessible to end users.

Following are the working models for cloud computing:

Deployment Models

Service Models

**DEPLOYMENT MODELS**

Deployment models define the type of access to the cloud, i.e., how the cloud is located? Cloud can have any of the four types of access: Public, Private, Hybrid and Community.

**PUBLIC CLOUD**

The Public Cloud allows systems and services to be easily accessible to the general public. Public cloud may be less secure because of its openness, e.g., e-mail.

**PRIVATE CLOUD**

The Private Cloud allows systems and services to be accessible within an organization. It offers increased security because of its private nature.

COMMUNITY CLOUD

The Community Cloud allows systems and services to be accessible by group of organizations.

HYBRID CLOUD

The Hybrid Cloud is mixture of public and private cloud. However, the critical activities are performed using private cloud while the non-critical activities are performed using public cloud

**SERVICE MODELS**

Service Models are the reference models on which the Cloud Computing is based. These can be categorized into three basic service models as listed below:

1. Infrastructure as a Service (IaaS)

2. Platform as a Service (PaaS)

3. Software as a Service (SaaS)

There are many other service models all of which can take the form like XaaS, i.e., Anything as a Service. This can be Network as a Service, Business as a Service, Identity as a Service, Database as a Service or Strategy as a Service.

INFRASTRUCTURE AS A SERVICE (IAAS) IaaS provides access to fundamental resources such as physical machines, virtual machines, virtual storage, etc.

PLATFORM AS A SERVICE (PAAS) PaaS provides the runtime environment for applications, development & deployment tools, etc.

OFTWARE AS A SERVICE (SAAS) SaaS model allows to use software applications as a service to end users.

Benefits Cloud Computing has numerous advantages. Some of them are listed below:

One can access applications as utilities, over the Internet.

Manipulate and configure the application online at any time.

It does not require to install a specific piece of software to access or manipulate cloud application.

Cloud Computing offers online development and deployment tools, programming runtime environment through Platform as a Service model.

Cloud resources are available over the network in a manner that provides platform independent access to any type of clients.

Cloud Computing offers on-demand self-service. The resources can be used without interaction with cloud

service provider. Cloud Computing is highly cost effective because it operates at higher efficiencies with greater utilization.

It just requires an Internet connection. Cloud Computing offers load balancing that makes it more reliable.

Risks Although Cloud Computing is a great innovation in the world of computing, there also exist downsides of cloud computing.

Some of them are discussed below: SECURITY & PRIVACY It is the biggest concern about cloud computing. S

ince data management and infrastructure management in cloud is provided by third-party, it is always a risk to handover the sensitive information to such providers.

Although the cloud computing vendors ensure more secure password protected accounts, any sign of security breach would result in loss of clients and businesses.

LOCK-IN It is very difficult for the customers to switch from one Cloud Service Provider (CSP) to another. It results in dependency on a particular CSP for service.

ISOLATION FAILURE This risk involves the failure of isolation mechanism that separates storage, memory, routing between the different tenants.

MANAGEMENT INTERFACE COMPROMISE In case of public cloud provider, the customer management interfaces are accessible through the Internet. I

NSECURE OR INCOMPLETE DATA DELETION It is possible that the data requested for deletion may not get deleted. It happens either because extra copies of data are stored but are not available or disk destroyed also stores data from other tenants.

Characteristics There are four key characteristics of cloud computing. They are shown in the following diagram:

ON DEMAND SELF-SERVICE Cloud Computing allows the users to use web services and resources on demand. One can logon to a website at any time and use them. BROAD NETWORK ACCESS Since Cloud Computing is completely web based, it can be accessed from anywhere and at any time.

RESOURCE POOLING Cloud Computing allows multiple tenants to share a pool of resources. One can share single physical instance of hardware, database and basic infrastructure.

RAPID ELASTICITY It is very easy to scale up or down the resources at any time. Resources used by the customers or currently assigned to customers are automatically monitored and resources. It make it possible

**Data Mining**

There is a huge amount of data available in the Information Industry. This data is of no use until it is converted into useful information. It is necessary to analyze this huge amount of data and extract useful information from it. Extraction of information is not the only process we need to perform; data mining also involves other processes such as Data Cleaning, Data Integration, Data Transformation, Data Mining, Pattern Evaluation and Data Presentation. Once all these processes are over, we would be able to use this information in many applications such as Fraud Detection, Market Analysis, Production Control, Science Exploration, etc.

What is Data Mining?

Data Mining is defined as extracting information from huge sets of data. In other words, we can say that data mining is the procedure of mining knowledge from data. The information or knowledge extracted so can be used for any of the following applications:

Market Analysis

Fraud Detection

Customer Retention

Production Control

Science Exploration

Data Mining Applications

Data mining is highly useful in the following domains: Market Analysis and Management

Corporate Analysis

& Risk Management Fraud Detection

Apart from these, data mining can also be used in the areas of production control, customer retention, science exploration, sports, astrology, and Internet Web Surf-Aid

Market Analysis and Management

Listed below are the various fields of market where data mining is used: Customer Profiling - Data mining helps determine what kind of people buy what kind of products.

Identifying Customer Requirements - Data mining helps in identifying thebest products for different customers. It uses prediction to find the factors that may attract new customers.

Cross Market Analysis - Data mining performs Association/correlations between product sales.

Target Marketing - Data mining helps to find clusters of model customers who share the same characteristics such as interests, spending habits, income, etc.

Determining Customer purchasing pattern - Data mining helps in determining customer purchasing pattern.

Providing Summary Information - Data mining provides us various multidimensional summary reports.

Corporate Analysis andRisk Management

Data mining is used in the following fields of the Corporate Sector: Finance Planning and Asset Evaluation - It involves cash flow analysis and prediction, contingent claim analysis to evaluate assets.

Resource Planning - It involves summarizing and comparing the resources and spending. Competition - It involves monitoring competitors and market directions.

Fraud Detection

Data mining is also used in the fields of credit card services and telecommunication to detect frauds. In fraud telephone calls, it helps to find the destination of the call, duration of the call, time of the day or week, etc. It also analyzes the patterns that deviate from expected norms.

Data mining essential step in the process of knowledge discovery

1.Data cleaning (to remove noise and inconsistent data)

2. Data integration (where multiple data sources may be combined)

3. Data selection (where data relevant to the analysis task are retrieved from the database)

4. Data transformation (where data are transformed or consolidated into forms appropriate for mining by performing summary or aggregation operations, for instance)

5. Data mining (an essential process where intelligent methods are applied in order to extract data patterns)

6. Pattern evaluation (to identify the truly interesting patterns representing knowledge based on some interestingness measures)

7. Knowledge presentation (where visualization and knowledge representation techniques are used to present the mined knowledge to the user)

Based on this view, the architecture of a typical data mining system may have the following major components

Database, data warehouse, World Wide Web, or other information repository: This is one or a set of databases, data warehouses, spreadsheets, or other kinds of information repositories.

Data cleaning and data integration techniques may be performed on the data. Database or data warehouse server: The database or data warehouse server is responsible for fetching the relevant data, based on the user’s data mining request.

Knowledge base: This is the domain knowledge that is used to guide the search or evaluate the interestingness of resulting patterns. Such knowledge can include concept hierarchies, used to organize attributes or attribute values into different levels of abstraction. Knowledge such as user beliefs, which can be used to assess a pattern’s interestingness based on its unexpectedness, may also be included. Other examples of domain knowledge are additional interestingness constraints or thresholds, and metadata (e.g., describing data from multiple heterogeneous sources).

Data mining engine: This is essential to the data mining system and ideally consists of a set of functional modules for tasks such as characterization, association and correlation analysis, classification, prediction, cluster analysis, outlier analysis, and evolution analysis.

Pattern evaluation module: This component typically employs interestingness measures and interacts with the data mining modules so as to focus the search toward interesting patterns. It may use interestingness thresholds to filter out discovered patterns. Alternatively, the pattern evaluation module may be integrated with the mining module, depending on the implementation of the data mining method used.

For efficient data mining, it is highly recommended to push the evaluation of pattern interestingness as deep as possible into the mining process so as to confine the search to only the interesting patterns.

User interface: This module communicates between users and the data mining system, allowing the user to interact with the system by specifying a data mining query or task, providing information to help focus the search, and performing exploratory data mining based on the intermediate data mining results. In addition, this component allows the user to browse database and data warehouse schemas or data structures, evaluate mined patterns, and visualize the patterns in different forms.

**DATA MINING ON WHAT KIND OF DATA?**

**RELATIONAL DATABASES**

A database system, also called a database management system (DBMS), consists of a collection of interrelated data, known as a database, and a set of software programs to manage and access the data A relational database is a collection of tables, each of which is assigned a unique name. Each table consists of a set of attributes (columns or fields) and usually stores a large set of tuples (records or rows)

Each tuple in a relational table represents an object identified by a unique key and described by a set of attribute values A semantic data model, such as an entity-relationship (ER) data model, is often constructed for relational databases. An ER data model represents the database as a set of entities and their relationships Relational data can be accessed by database queries written in a relational query language, such as SQL

**DATA WAREHOUSES**

A data warehouse is a repository of information collected from multiple sources, stored under a unified schema, and that usually resides at a single site Data warehouses are constructed via a process of data cleaning, data integration, data transformation, data loading, and periodic data refreshing

**OBJECT-RELATIONAL DATABASES**

Based on an object-relational data model Extends the relational model by providing a rich data type for handling complex objects and object orientation Objects that share a common set of properties can be grouped into an object class. Each object is an instance of its class. Object classes can be organized into class/subclass hierarchies

**ADVANCED DATA AND INFORMATION SYSTEMS**

With the progress of database technology, various kinds of advanced data and information systems have emerged and are undergoing development to address the requirements of new applications handling spatial/temporal data (such as maps) engineering design data (such as the design of buildings, system components, or integrated circuits) hypertext and multimedia data (including text, image, video, and audio data) time-related data (such as historical records or stock exchange data) stream data (such as video surveillance and sensor data, where data flow in and out like streams) the World Wide Web (a huge, widely distributed information repository made available by the Internet)

**THE WORLD WIDE WEB**

The World Wide Web and its associated distributed information services, such as Yahoo! and Google provide rich, worldwide, on-line information services, where data objects are linked together to facilitate interactive access Capturing user access patterns in such distributed information environments is called Web usage mining (or Weblog mining)

Database or data warehouse server responsible for fetching the relevant data, based on the user’s data mining request can be decouples/loose coupled/tightly coupled with the database layer

Knowledge base the domain knowledge that is used to guide the search or evaluate the interestingness of resulting patterns interestingness constraints or thresholds, metadata, concept hierarchies, etc.

Data mining engine this is essential to the data mining system and ideally consists of a set of functional modules for tasks such as characterization, association and correlation analysis, classification, prediction, cluster analysis, outlier analysis, and evolution analysis query languages (DMQL) based on mining primitives to access the data

Pattern evaluation module interacts with the data mining modules so as to focus the search toward interesting patterns may use interestingness thresholds to filter out discovered patterns may be integrated with the mining module

User interface communicates between users and the data mining system allows the user to interact with the system by specifying a data mining query or task, providing information to help focus the search, and performing exploratory data mining based on the intermediate data mining results allows the user to browse database and data warehouse schemas or data structures, evaluate mined patterns, and visualize the patterns in different forms

**4. System Analysis**

The **Systems Development Life Cycle (SDLC)**, or *Software Development Life Cycle* in [systems engineering](http://en.wikipedia.org/wiki/Systems_engineering), [information systems](http://en.wikipedia.org/wiki/Information_systems) and [software engineering](http://en.wikipedia.org/wiki/Software_engineering), is the process of creating or altering systems, and the models and [methodologies](http://en.wikipedia.org/wiki/Methodologies) that people use to develop these systems.

In software engineering the SDLC concept underpins many kinds of [software development methodologies](http://en.wikipedia.org/wiki/Software_development_methodologies). These methodologies form the framework for planning and controlling the creation of an information system the [software development process](http://en.wikipedia.org/wiki/Software_development_process).

**SOFTWARE MODEL OR ARCHITECTURE ANALYSIS:**

Structured project management techniques (such as an SDLC) enhance management’s control over projects by dividing complex tasks into manageable sections. A software life cycle model is either a descriptive or prescriptive characterization of how software is or should be developed. But none of the SDLC models discuss the key issues like Change management, Incident management and Release management processes within the SDLC process, but, it is addressed in the overall project management. In the proposed hypothetical model, the concept of user-developer interaction in the conventional SDLC model has been converted into a three dimensional model which comprises of the user, owner and the developer. In the proposed hypothetical model, the concept of user-developer interaction in the conventional SDLC model has been converted into a three dimensional model which comprises of the user, owner and the developer. The ―one size fits all‖ approach to applying SDLC methodologies is no longer appropriate. We have made an attempt to address the above mentioned defects by using a new hypothetical model for SDLC described elsewhere. The drawback of addressing these management processes under the overall project management is missing of key technical issues pertaining to software development process that is, these issues are talked in the project management at the surface level but not at the ground level.

**Proposed System:**

Unit Test case instances are grouped together according to the features they test. PyUnit provides a mechanism for this: the 'test suite', represented by the class Test Suite in the unit testmodule. We use the Test Suite class for defining and running the test suite. And we can add multiple test cases into it. Also, apart from the Test Suite class, we need to use Test Loader and TextTestRunner classes to create and run a test suite. TheTest Loader class reads all the test methods from the specified test files that contain the definition of the test suite. Then, the Test Runner class take control of the test suite and run all the tests specified By default, the Python Unit test library emits the test output on the terminal console. If you want to share the results with management and stakeholders, then sending console logs isn’t the appropriate way. So you need to generate a report which looks presentable and professional. A summary report formatted nicely, with a drill-down access to the details is what required. Since the unit test library doesn’t have the ability to produce such report, so you should use the HTMLTestRunner extension. After executing the test suite, we’ll see that HTMLTestRunner runs all the tests as does the test runner of the unit test library.

**2.5Functional requirements**

Outputs from computer systems are required primarily to communicate the results of processing to users. They are also used to provide a permanent copy of the results for later consultation. The various types of outputs in general are:

* External Outputs, whose destination is outside the organization,.
* Internal Outputs whose destination is within organization and they are the
* User’s main interface with the computer.
* Operational outputs whose use is purely within the computer department.
* Interface outputs, which involve the user in communicating directly.
* Understanding user’s preferences, expertise level and his business requirements through a friendly questionnaire.
* Input data can be in four different forms - Relational DB, text files, .xls and xml files. For testing and demo you can choose data from any domain. User-B can provide business data as input.

**Non-Functional Requirements**

1. Secure access of confidential data (user’s details). SSL can be used.
2. 24 X 7 availability.
3. Better component design to get better performance at peak time
4. Flexible service based architecture will be highly desirable for future extension

Preliminary investigation examine project feasibility, the likelihood the system will be useful to the organization. The main objective of the feasibility study is to test the Technical, Operational and Economical feasibility for adding new modules and debugging old running system. All system is feasible if they are unlimited resources and infinite time. There are aspects in the feasibility study portion of the preliminary investigation:

* Technical Feasibility
* Operational Feasibility
* Economical Feasibility

**3.1. TECHNICAL FEASIBILITY**

The technical issue usually raised during the feasibility stage of the investigation includes the following:

* Does the necessary technology exist to do what is suggested?
* Do the proposed equipments have the technical capacity to hold the data required to use the new system?
* Will the proposed system provide adequate response to inquiries, regardless of the number or location of users?
* Can the system be upgraded if developed?
* Are there technical guarantees of accuracy, reliability, ease of access and data security?

Earlier no system existed to cater to the needs of ‘Secure Infrastructure Implementation System’. The current system developed is technically feasible. It is a web based user interface for audit workflow at NIC-CSD. Thus it provides an easy access to the users. The database’s purpose is to create, establish and maintain a workflow among various entities in order to facilitate all concerned users in their various capacities or roles. Permission to the users would be granted based on the roles specified. Therefore, it provides the technical guarantee of accuracy, reliability and security. The software and hard requirements for the development of this project are not many and are already available in-house at NIC or are available as free as open source. The work for the project is done with the current equipment and existing software technology. Necessary bandwidth exists for providing a fast feedback to the users irrespective of the number of users using the system.

**3.2. OPERATIONAL FEASIBILITY**

Proposed projects are beneficial only if they can be turned out into information system. That will meet the organization’s operating requirements. Operational feasibility aspects of the project are to be taken as an important part of the project implementation. Some of the important issues raised are to test the operational feasibility of a project includes the following: -

* Is there sufficient support for the management from the users?
* Will the system be used and work properly if it is being developed and implemented?
* Will there be any resistance from the user that will undermine the possible application benefits?

This system is targeted to be in accordance with the above-mentioned issues. Beforehand, the management issues and user requirements have been taken into consideration. So there is no question of resistance from the users that can undermine the possible application benefits.

The well-planned design would ensure the optimal utilization of the computer resources and would help in the improvement of performance status.

**3.3. ECONOMICAL FEASIBILITY**

A system can be developed technically and that will be used if installed must still be a good investment for the organization. In the economical feasibility, the development cost in creating the system is evaluated against the ultimate benefit derived from the new systems. Financial benefits must equal or exceed the costs.

The system is economically feasible. It does not require any addition hardware or software. Since the interface for this system is developed using the existing resources and technologies available at NIC, There is nominal expenditure and economical feasibility for certain.

**ABSTRACT:**

Now a days, testing has gained much importance in every field. Without testing, no application can be shipped. If tried to do so, there may be lots and lots of complaints from the users of the application which may lead to lose their confidence in our application. Trying to fix one bug without testing may create other. Thus, testing is very important when it comes to making a polished product. An automated test is basically the same thing you'd do via manual testing - but it is done by the computer so it allows you to go do other things, as well as easily repeat your tests. The test code is going to run a series of instructions to interact with a web browser - mimicking how an actual end user would interact with an application. The script is going to navigate the browser, click a button, enter some text input, click a radio button, select a drop down, drag and drop, etc. In short, the code will test the functionality of the web application. Despite it's widespread industry adoption, the Selenium project is far from done. Selenium's goal is to drive any browser the same way an end user would (e.g. opening pages, clicking buttons, entering text, etc.) with any programming language on any OS platform Selenium Python API is however gaining importance in terms of functional/acceptance testing for the websites. With Selenium, we can automate the web based applications and perform various actions like clicking, hovering, filling the form data, etc easily.

**Proposed System:**

Our implementation is based on the Pylearn2 library [17]. Pylearn2 is an open-source machine learning library specializing in deep learning algorithms. It also supports the use of GPUs, which can greatly accelerate the execution of deep learning algorithms. Since CNN’s are able to learn useful features from scratch, we applied only minimal pre-processing. We employed the same pre-processing as Tustison et al., the winner of the 2013 BRATS challenge [32]. The pre-processing follows three steps. First, the 1% highest and lowest intensities are removed. Then, we apply an N4ITK bias correction [3] to T1 and T1C modalities. The data is then normalized within each input channel by subtracting the channel’s mean and dividing by the channel’s standard deviation. As for post-processing, a simple method based on connected components was implemented to remove flat blobs which might appear in the predictions due to bright corners of the brains close to the skull. The hyper-parameters of the different architectures (kernel and max pooling size for each layer and the number of layers) can be seen in Figure 3. Hyper-parameters were tuned using grid search and cross-validation on a validation set (see Bengio [6]). The chosen hyper-parameters were the ones for which the model performed best on the validation set. For max pooling, we always use a stride of 1. This is to keep per-pixel accuracy during full image prediction. We observed in practice that max pooling in the global path does not improve accuracy. We also found that adding additional layers to the architectures or increasing the capacity of the model by adding additional feature maps to the convolutional blocks do not provide any meaningful performance improvement. Biases are initialized to zero except for the softmax layer for which we initialized them to the log of the label frequencies. The kernels are randomly initialized from U (−0.005, 0.005). Training takes about 3 minutes per epoch for the TwoPathCNN model on an NVIDIA Titan black card. At test time, we run our code on a GPU in order to exploit its computational speed. Moreover, the convolutional nature of the output layer allows us to further accelerate computations at test time. This is done by feeding as input a full image and not individual patches. Therefore, convolutions at all layers can be extended to obtain all label probabilities p(Yi j|X) for the entire image. With this implementation, we are able to produce a segmentation in 25 seconds per brain on the Titan black card with the TwoPathCNN model. This turns out to be 45 times faster than when we extracted a patch at each pixel and processed them individually for the entire brain. Predictions for the MFCascadeCNN model, the LocalCascadeCNN model, and InputCascadeCNN model take on average 1.5 minutes, 1.7 minutes and 3 minutes respectively

**Software Requirements:**

* OS: Windows
* Python IDE : python 2.7.x
* Pycharm IDE
* Browser: Firefox 28
* Selenium: Selenium 2.4 Above

**Hardware Requirements:**

* RAM: 4GB
* Processor:Intel i3
* Hard Disk: 500GB

**FEASIBILITY STUDY:**

Preliminary investigation examine project feasibility, the likelihood the system will be useful to the organization. The main objective of the feasibility study is to test the Technical, Operational and Economical feasibility for adding new modules and debugging old running system. All system is feasible if they are unlimited resources and infinite time. There are aspects in the feasibility study portion of the preliminary investigation:

* Technical Feasibility
* Economical Feasibility
* Operation Feasibility

**Technical Feasibility:**

In the feasibility study first step is that the organization or company has to decide that what technologies are suitable to develop by considering existing system.

Here in this application used the technologies like Visual Studio 2012 and SqlServer 2014. These are free software that would be downloaded from web.

Visual Studio 2013 –it is tool or technology.

**Functional requirement:**

Outputs from computer systems are required primarily to communicate the results of processing to users. They are also used to provide a permanent copy of the results for later consultation. The various types of outputs in general are:

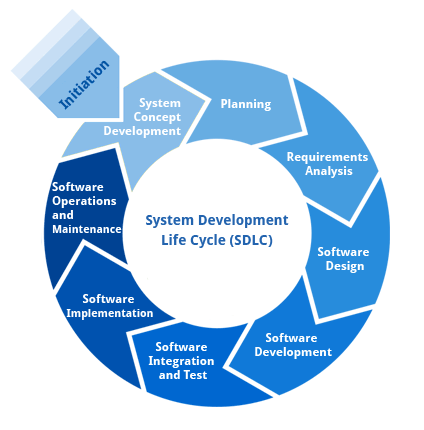
* External Outputs, whose destination is outside the organization,.
* Internal Outputs whose destination is within organization and they are the
* User’s main interface with the computer.
* Operational outputs whose use is purely within the computer department.
* Interface outputs, which involve the user in communicating directly.
* Understanding user’s preferences, expertise level and his business requirements through a friendly questionnaire.
* Input data can be in four different forms - Relational DB, text files, .xls and xml files. For testing and demo you can choose data from any domain. User-B can provide business data as input.

**Non functional requirements:**

* Secure access of confidential data (user’s details). SSL can be used.
* 24 X 7 availability.
* Better component design to get better performance at peak time
* Flexible service based architecture will be highly desirable for future extension

**Software Development Life Cycle**

The **Systems Development Life Cycle (SDLC)**, or Software Development Life Cycle in systems engineering, information systems and software engineering, is the process of creating or altering systems, and the models and methodologies use to develop these systems.



**Requirement Analysis and Design**

Analysis gathers the requirements for the system. This stage includes a detailed study of the business needs of the organization. Options for changing the business process may be considered. Design focuses on high level design like, what programs are needed and how are they going to interact, low-level design (how the individual programs are going to work), interface design (what are the interfaces going to look like) and data design (what data will be required). During these phases, the software's overall structure is defined. Analysis and Design are very crucial in the whole development cycle. Any glitch in the design phase could be very expensive to solve in the later stage of the software development. Much care is taken during this phase. The logical system of the product is developed in this phase.

**Implementation**

In this phase the designs are translated into code. Computer programs are written using a conventional programming language or an application generator. Programming tools like Compilers, Interpreters, and Debuggers are used to generate the code. Different high level programming languages like C, C++, Pascal, .Net are used for coding. With respect to the type of application, the right programming language is chosen.

**Testing**

In this phase the system is tested. Normally programs are written as a series of individual modules, this subject to separate and detailed test. The system is then tested as a whole. The separate modules are brought together and tested as a complete system. The system is tested to ensure that interfaces between modules work (integration testing), the system works on the intended platform and with the expected volume of data (volume testing) and that the system does what the user requires (acceptance/beta testing).

**Maintenance**

Inevitably the system will need maintenance. Software will definitely undergo change once it is delivered to the customer. There are many reasons for the change. Change could happen because of some unexpected input values into the system. In addition, the changes in the system could directly affect the software operations. The software should be developed to accommodate changes that could happen during the post implementation period.

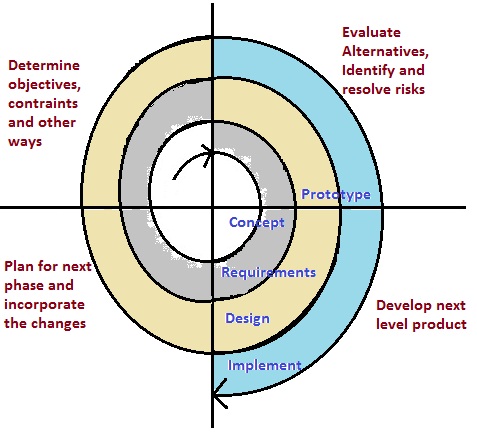
**SDLC METHDOLOGIES**

This document play a vital role in the development of life cycle (SDLC) as it describes the complete requirement of the system. It means for use by developers and will be the basic during testing phase. Any changes made to the requirements in the future will have to go through formal change approval process.

**SPIRAL MODEL** was defined by Barry Boehm in his 1988 article, “A spiral Model of Software Development and Enhancement. This model was not the first model to discuss iterative development, but it was the first model to explain why the iteration models.

As originally envisioned, the iterations were typically 6 months to 2 years long. Each phase starts with a design goal and ends with a client reviewing the progress thus far. Analysis and engineering efforts are applied at each phase of the project, with an eye toward the end goal of the project.

**The following diagram shows how a spiral model acts like:**



The steps for Spiral Model can be generalized as follows:

* The new system requirements are defined in as much details as possible. This usually involves interviewing a number of users representing all the external or internal users and other aspects of the existing system.
* A preliminary design is created for the new system.
* A first prototype of the new system is constructed from the preliminary design. This is usually a scaled-down system, and represents an approximation of the characteristics of the final product.
* A second prototype is evolved by a fourfold procedure:

1. Evaluating the first prototype in terms of its strengths, weakness, and risks.
2. Defining the requirements of the second prototype.
3. Planning an designing the second prototype.
4. Constructing and testing the second prototype.

* At the customer option, the entire project can be aborted if the risk is deemed too great. Risk factors might involved development cost overruns, operating-cost miscalculation, or any other factor that could, in the customer’s judgment, result in a less-than-satisfactory final product.
* The existing prototype is evaluated in the same manner as was the previous prototype, and if necessary, another prototype is developed from it according to the fourfold procedure outlined above.
* The preceding steps are iterated until the customer is satisfied that the refined prototype represents the final product desired.
* The final system is constructed, based on the refined prototype.
* The final system is thoroughly evaluated and tested. Routine maintenance is carried on a continuing basis to prevent large scale failures and to minimize down time.

## 4.1. FUNCTIONAL REQUIREMENTS

**OUTPUT DESIGN**

Outputs from computer systems are required primarily to communicate the results of processing to users. They are also used to provides a permanent copy of the results for later consultation. The various types of outputs in general are:

* External Outputs, whose destination is outside the organization
* Internal Outputs whose destination is within organization and they are the
* User’s main interface with the computer.
* Operational outputs whose use is purely within the computer department.
* Interface outputs, which involve the user in communicating directly.

**OUTPUT DEFINITION**

# The outputs should be defined in terms of the following points:

* + - Type of the output
    - Content of the output
    - Format of the output
    - Location of the output
    - Frequency of the output
    - Volume of the output
    - Sequence of the output

It is not always desirable to print or display data as it is held on a computer. It should be decided as which form of the output is the most suitable.

**INPUT DESIGN**

Input design is a part of overall system design. The main objective during the input design is as given below:

* To produce a cost-effective method of input.
* To achieve the highest possible level of accuracy.
* To ensure that the input is acceptable and understood by the user.

**INPUT STAGES:**

The main input stages can be listed as below:

* Data recording
* Data transcription
* Data conversion
* Data verification
* Data control
* Data transmission
* Data validation
* Data correction

**INPUT TYPES:**

It is necessary to determine the various types of inputs. Inputs can be categorized as follows:

* External inputs, which are prime inputs for the system.
* Internal inputs, which are user communications with the system.
* Operational, which are computer department’s communications to the system?
* Interactive, which are inputs entered during a dialogue.

**INPUT MEDIA:**

At this stage choice has to be made about the input media. To conclude about the input media consideration has to be given to;

* Type of input
* Flexibility of format
* Speed
* Accuracy
* Verification methods
* Rejection rates
* Ease of correction
* Storage and handling requirements
* Security
* Easy to use
* Portability

Keeping in view the above description of the input types and input media, it can be said that most of the inputs are of the form of internal and interactive. As

Input data is to be the directly keyed in by the user, the keyboard can be considered to be the most suitable input device.

**ERROR AVOIDANCE**

At this stage care is to be taken to ensure that input data remains accurate form the stage at which it is recorded up to the stage in which the data is accepted by the system. This can be achieved only by means of careful control each time the data is handled.

**ERROR DETECTION**

Even though every effort is make to avoid the occurrence of errors, still a small proportion of errors is always likely to occur, these types of errors can be discovered by using validations to check the input data.

**DATA VALIDATION**

Procedures are designed to detect errors in data at a lower level of detail. Data validations have been included in the system in almost every area where there is a possibility for the user to commit errors. The system will not accept invalid data. Whenever an invalid data is keyed in, the system immediately prompts the user and the user has to again key in the data and the system will accept the data only if the data is correct. Validations have been included where necessary.

The system is designed to be a user friendly one. In other words the system has been designed to communicate effectively with the user. The system has been designed with popup menus.

**USER INTERFACE DESIGN**

It is essential to consult the system users and discuss their needs while designing the user interface:

**USER INTERFACE SYSTEMS CAN BE BROADLY CLASIFIED AS:**

1. User initiated interface the user is in charge, controlling the progress of the user/computer dialogue. In the computer-initiated interface, the computer selects the next stage in the interaction.
2. Computer initiated interfaces

In the computer initiated interfaces the computer guides the progress of the user/computer dialogue. Information is displayed and the user response of the computer takes action or displays further information.

**USER\_INITIATED INTERGFACES**

User initiated interfaces fall into tow approximate classes:

1. Command driven interfaces: In this type of interface the user inputs commands or queries which are interpreted by the computer.
2. Forms oriented interface: The user calls up an image of the form to his/her screen and fills in the form. The forms oriented interface is chosen because it is the best choice.

**COMPUTER-INITIATED INTERFACES**

The following computer – initiated interfaces were used:

1. The menu system for the user is presented with a list of alternatives and the user chooses one; of alternatives.
2. Questions – answer type dialog system where the computer asks question and takes action based on the basis of the users reply.

Right from the start the system is going to be menu driven, the opening menu displays the available options. Choosing one option gives another popup menu with more options. In this way every option leads the users to data entry form where the user can key in the data.

**ERROR MESSAGE DESIGN:**

The design of error messages is an important part of the user interface design. As user is bound to commit some errors or other while designing a system the system should be designed to be helpful by providing the user with information regarding the error he/she has committed.

This application must be able to produce output at different modules for different inputs.

**4.2. PERFORMANCE REQUIREMENTS**

Performance is measured in terms of the output provided by the application.

Requirement specification plays an important part in the analysis of a system. Only when the requirement specifications are properly given, it is possible to design a system, which will fit into required environment. It rests largely in the part of the users of the existing system to give the requirement specifications because they are the people who finally use the system. This is because the requirements have to be known during the initial stages so that the system can be designed according to those requirements. It is very difficult to change the system once it has been designed and on the other hand designing a system, which does not cater to the requirements of the user, is of no use.

The requirement specification for any system can be broadly stated as given below:

* The system should be able to interface with the existing system
* The system should be accurate
* The system should be better than the existing system

The existing system is completely dependent on the user to perform all the duties.

**IMPLIMENTATION ON (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, where as 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 your not interested in the the hows 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

# 20 Python libraries

**1.** Requests. The most famous http library written by kenneth reitz. It’s a must have for every python developer.

**2.** Scrapy. If you are involved in webscraping then this is a must have library for you. After using this library you won’t use any other.

**3.** wxPython. A gui toolkit for python. I have primarily used it in place of tkinter. You will really love it.

**4.** Pillow. A friendly fork of PIL (Python Imaging Library). It is more user friendly than PIL and is a must have for anyone who works with images.

**5.** SQLAlchemy. A database library. Many love it and many hate it. The choice is yours.

**6.** BeautifulSoup. I know it’s slow but this xml and html parsing library is very useful for beginners.

**7.** Twisted. The most important tool for any network application developer. It has a very beautiful api and is used by a lot of famous python developers.

**8.** NumPy. How can we leave this very important library ? It provides some advance math functionalities to python.

**9.** SciPy. When we talk about NumPy then we have to talk about scipy. It is a library of algorithms and mathematical tools for python and has caused many scientists to switch from ruby to python.

**10.** matplotlib. A numerical plotting library. It is very useful for any data scientist or any data analyzer.

**11.** Pygame. Which developer does not like to play games and develop them ? This library will help you achieve your goal of 2d game development.

**12.** Pyglet. A 3d animation and game creation engine. This is the engine in which the famous [python port](https://github.com/fogleman/Minecraft) of minecraft was made

**13.** pyQT. A GUI toolkit for python. It is my second choice after wxpython for developing GUI’s for my python scripts.

**14.** pyGtk. Another python GUI library. It is the same library in which the famous Bittorrent client is created.

**15.** Scapy. A packet sniffer and analyzer for python made in python.

**16.** pywin32. A python library which provides some useful methods and classes for interacting with windows.

**17.** nltk. Natural Language Toolkit – I realize most people won’t be using this one, but it’s generic enough. It is a very useful library if you want to manipulate strings. But it’s capacity is beyond that. Do check it out.

**18.** nose. A testing framework for python. It is used by millions of python developers. It is a must have if you do test driven development.

**19.** SymPy. SymPy can do algebraic evaluation, differentiation, expansion, complex numbers, etc. It is contained in a pure Python distribution.

**20.** IPython. I just can’t stress enough how useful this tool is. It is a python prompt on steroids. It has completion, history, shell capabilities, and a lot more. Make sure that you take a look at it.

**Numpy**

NumPy’s main object is the homogeneous multidimensional array. It is a table of elements (usually numbers), all of the same type, indexed by a tuple of positive integers. In NumPy dimensions are called axes. The number of axes is rank.

• Offers Matlab-ish capabilities within Python

• Fast array operations

• 2D arrays, multi-D arrays, linear algebra etc.

**matplotlib**

• High quality plotting library.

**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 carred out, the function can or can not 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 firstname and familiy 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.

**Python-Data Base Communication**

Connector/Python provides a connect() call used to establish connections to the MySQL server. The following sections describe the permitted arguments for connect() and describe how to use option files that supply additional arguments.

A database is an organized collection of data. The data are typically organized to model aspects of reality in a way that supports processes requiring this information.

The term "database" can both refer to the data themselves or to the database management system. The Database management system is a software application for the interaction between users database itself.

Databases are popular for many applications, especially for use with web applications or customer-oriented programs

Users don't have to be human users. They can be other programs and applications as well. We will learn how Python or better a Python program can interact as a user of an SQLdatabase.   
  
This is an introduction into using SQLite and MySQL from Python.

The Python standard for database interfaces is the Python DB-API, which is used by Python's database interfaces.

The DB-API has been defined as a common interface, which can be used to access relational databases.

In other words, the code in Python for communicating with a database should be the same, regardless of the database and the database module used. Even though we use lots of SQL examples, this is not an introduction into SQL but a tutorial on the Python interface.

SQLite is a simple relational database system, which saves its data in regular data files or even in the internal memory of the computer, i.e. the RAM.

It was developped for embedded applications, like Mozilla-Firefox (Bookmarks), Symbian OS or Android.

SQLITE is "quite" fast, even though it uses a simple file. It can be used for large databases as well.

If you want to use SQLite, you have to import the module sqlite3. To use a database, you have to create first a Connection object.

The connection object will represent the database. The argument of connection - in the following example "companys.db" - functions both as the name of the file, where the data will be stored, and as the name of the database. If a file with this name exists, it will be opened.

It has to be a SQLite database file of course! In the following example, we will open a database called company.

MySQL Connector/Python enables Python programs to access MySQL databases, using an API that is compliant with the Python Database API Specification v2.0 (PEP 249). It is written in pure Python and does not have any dependencies except for the Python Standard Library.

For notes detailing the changes in each release of Connector/Python, see MySQL Connector/Python Release Notes.

MySQL Connector/Python includes support for:

* Almost all features provided by MySQL Server up to and including MySQL Server version 5.7.
* Converting parameter values back and forth between Python and MySQL data types, for example Python datetime and MySQL DATETIME. You can turn automatic conversion on for convenience, or off for optimal performance.
* All MySQL extensions to standard SQL syntax.
* Protocol compression, which enables compressing the data stream between the client and server.
* Connections using TCP/IP sockets and on Unix using Unix sockets.
* Secure TCP/IP connections using SSL.
* Self-contained driver. Connector/Python does not require the MySQL client library or any Python modules outside the standard library

**6.1. INTRODUCTION**

Software design sits at the technical kernel of the software engineering process and is applied regardless of the development paradigm and area of application. Design is the first step in the development phase for any engineered product or system. The designer’s goal is to produce a model or representation of an entity that will later be built. Beginning, once system requirement have been specified and analyzed, system design is the first of the three technical activities -design, code and test that is required to build and verify software.

The importance can be stated with a single word “Quality”. Design is the place where quality is fostered in software development. Design provides us with representations of software that can assess for quality. Design is the only way that we can accurately translate a customer’s view into a finished software product or system. Software design serves as a foundation for all the software engineering steps that follow. Without a strong design we risk building an unstable system – one that will be difficult to test, one whose quality cannot be assessed until the last stage. The purpose of the design phase is to plan a solution of the problem specified by the requirement document. This phase is the first step in moving from the problem domain to the solution domain. In other words, starting with what is needed, design takes us toward how to satisfy the needs. The design of a system is perhaps the most critical factor affection the quality of the software; it has a major impact on the later phase, particularly testing, maintenance. The output of this phase is the design document. This document is similar to a blueprint for the solution and is used later during implementation, testing and maintenance. The design activity is often divided into two separate phases System Design and Detailed Design.

System Design also called top-level design aims to identify the modules that should be in the system, the specifications of these modules, and how they interact with each other to produce the desired results. At the end of the system design all the major data structures, file formats, output formats, and the major modules in the system and their specifications are decided.

During, Detailed Design, the internal logic of each of the modules specified in system design is decided. During this phase, the details of the data of a module is usually specified in a high-level design description language, which is independent of the target language in which the software will eventually be implemented.

In system design the focus is on identifying the modules, whereas during detailed design the focus is on designing the logic for each of the modules. In other works, in system design the attention is on what components are needed, while in detailed design how the components can be implemented in software is the issue.

Design is concerned with identifying software components specifying relationships among components. Specifying software structure and providing blue print for the document phase. Modularity is one of the desirable properties of large systems. It implies that the system is divided into several parts. In such a manner, the interaction between parts is minimal clearly specified.

During the system design activities, Developers bridge the gap between the requirements specification, produced during requirements elicitation and analysis, and the system that is delivered to the user.

Design is the place where the quality is fostered in development. Software design is a process through which requirements are translated into a representation of software.

**Data Flow Diagrams:**

A graphical tool used to describe and analyze the moment of data through a system manual or automated including the process, stores of data, and delays in the system. Data Flow Diagrams are the central tool and the basis from which other components are developed. The transformation of data from input to output, through processes, may be described logically and independently of the physical components associated with the system. The DFD is also know as a data flow graph or a bubble chart.

DFDs are the model of the proposed system. They clearly should show the requirements on which the new system should be built. Later during design activity this is taken as the basis for drawing the system’s structure charts. The Basic Notation used to create a DFD’s are as follows:

**1. Dataflow:** Data move in a specific direction from an origin to a destination.

**2. Process:** People, procedures, or devices that use or produce (Transform) Data. The physical component is not identified.

**3. Source:** External sources or destination of data, which may be People, programs, organizations or other entities.

**4. Data Store:** Here data are stored or referenced by a process in the System.

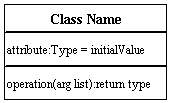
**What is a UML Class Diagram?**

Class diagrams are the backbone of almost every object-oriented method including UML. They describe the static structure of a system.

**Basic Class Diagram Symbols and Notations**

Classes represent an abstraction of entities with common characteristics. Associations represent the relationships between classes.

Illustrate classes with rectangles divided into compartments. Place the name of the class in the first partition (centered, bolded, and capitalized), list the attributes in the second partition, and write operations into the third.



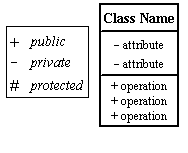
**Active Class**

Active classes initiate and control the flow of activity, while passive classes store data and serve other classes. Illustrate active classes with a thicker border.



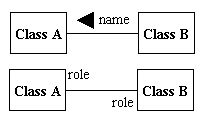
**Visibility**

Use visibility markers to signify who can access the information contained within a class. Private visibility hides information from anything outside the class partition. Public visibility allows all other classes to view the marked information. Protected visibility allows child classes to access information they inherited from a parent class. [.](http://www.smartdraw.com/resources/tutorials/Text-and-Tables)



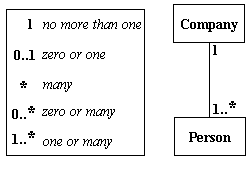
**Associations**

Associations represent static relationships between classes. Place association names above, on, or below the association line. Use a filled arrow to indicate the direction of the relationship. Place roles near the end of an association. Roles represent the way the two classes see each other.  
***Note:*** It's uncommon to name both the association and the class roles.



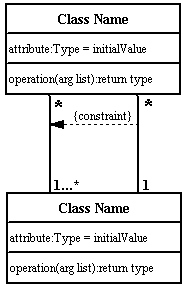
**Multiplicity (Cardinality)**

Place multiplicity notations near the ends of an association. These symbols indicate the number of instances of one class linked to one instance of the other class. For example, one company will have one or more employees, but each employee works for one company only.



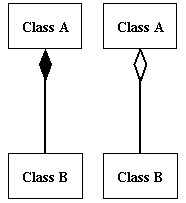
**Constraint**

Place constraints inside curly braces {}.

http://wc1.smartdraw.com/resources/tutorials/images/uml_constraint.gif*Simple Constraint* 

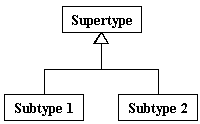
**Composition and Aggregation**

Composition is a special type of aggregation that denotes a strong ownership between Class A, the whole, and Class B, its part. Illustrate **composition** with a filled diamond. Use a hollow diamond to represent a simple **aggregation** relationship, in which the "whole" class plays a more important role than the "part" class, but the two classes are not dependent on each other. The diamond end in both a composition and aggregation relationship points toward the "whole" class or the aggregate



**Generalization**

Generalization is another name for inheritance or an "is a" relationship. It refers to a relationship between two classes where one class is a specialized version of another. For example, Honda is a type of car. So the class Honda would have a generalization relationship with the class car.



In real life coding examples, the difference between inheritance and aggregation can be confusing. If you have an aggregation relationship, the aggregate (the whole) can access only the PUBLIC functions of the part class. On the other hand, inheritance allows the inheriting class to access both the PUBLIC and PROTECTED functions of the super class.

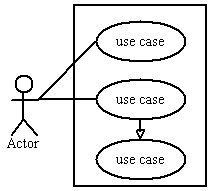
**What is a UML Use Case Diagram?**

Use case diagrams model the functionality of a system using actors and use cases. Use cases are services or functions provided by the system to its users.

**Basic Use Case Diagram Symbols and Notations**

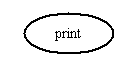
**System**

Draw your system's boundaries using a rectangle that contains use cases. Place actors outside the system's boundaries.



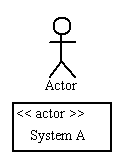
**Use Case**

Draw use cases using ovals. Label with ovals with verbs that represent the system's functions.



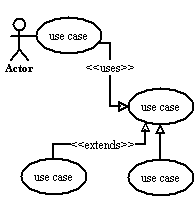
**Actors**

Actors are the users of a system. When one system is the actor of another system, label the actor system with the actor stereotype.



**Relationships**

Illustrate relationships between an actor and a use case with a simple line. For relationships among use cases, use arrows labeled either "uses" or "extends." A "uses" relationship indicates that one use case is needed by another in order to perform a task. An "extends" relationship indicates alternative options under a certain use case.



**Sequence Diagram**

Sequence diagrams describe interactions among classes in terms of an exchange of messages over time.

**Basic Sequence Diagram Symbols and Notations**

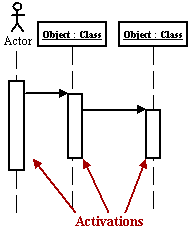
**Class roles**

Class roles describe the way an object will behave in context. Use the UML object symbol to illustrate class roles, but don't list object attributes.

Class roles

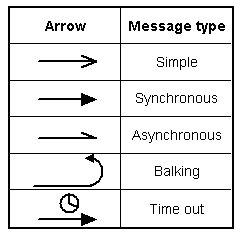
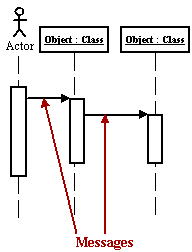
**Activation**

Activation boxes represent the time an object needs to complete a task.



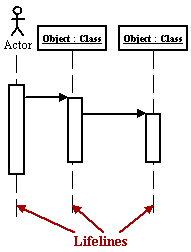
**Messages**

Messages are arrows that represent communication between objects. Use half-arrowed lines to represent asynchronous messages. Asynchronous messages are sent from an object that will not wait for a response from the receiver before continuing its tasks.

  
*Various message types for Sequence and Collaboration diagrams*

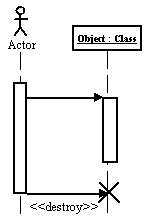
**Lifelines**

Lifelines are vertical dashed lines that indicate the object's presence over time.



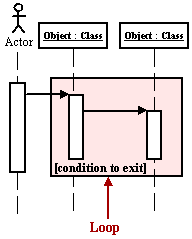
**Destroying Objects**

Objects can be terminated early using an arrow labeled "<< destroy >>" that points to an X.



**Loops**

A repetition or loop within a sequence diagram is depicted as a rectangle. Place the condition for exiting the loop at the bottom left corner in square brackets [ ].



**Collaboration Diagram**

A collaboration diagram describes interactions among objects in terms of sequenced messages. Collaboration diagrams represent a combination of information taken from class, sequence, and use case diagrams describing both the static structure and dynamic behavior of a system.

**Basic Collaboration Diagram Symbols and Notations**

**Class roles**

Class roles describe how objects behave. Use the UML object symbol to illustrate class roles, but don't list object attributes.

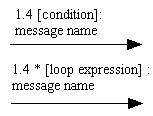
Class roles

**Association roles**

Association roles describe how an association will behave given a particular situation. You can draw association roles using simple lines labeled with stereotypes.  
Association roles

**Messages**

Unlike sequence diagrams, collaboration diagrams do not have an explicit way to denote time and instead number messages in order of execution. Sequence numbering can become nested using the Dewey decimal system. For example, nested messages under the first message are labeled 1.1, 1.2, 1.3, and so on. The a condition for a message is usually placed in square brackets immediately following the sequence number. Use a \* after the sequence number to indicate a loop.  
[Learn how to add arrows to your lines.](http://www.smartdraw.com/resources/tutorials/Lines)



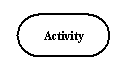
**Activity Diagram**

An activity diagram illustrates the dynamic nature of a system by modeling the flow of control from activity to activity. An activity represents an operation on some class in the system that results in a change in the state of the system. Typically, activity diagrams are used to model workflow or business processes and internal operation. Because an activity diagram is a special kind of state chart diagram, it uses some of the same modeling conventions.

**Basic Activity Diagram Symbols and Notations**

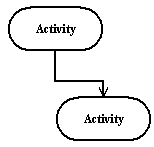
**Action states**

Action states represent the non interruptible actions of objects. You can draw an action state in Smart Draw using a rectangle with rounded corners.



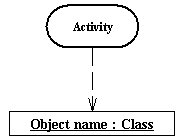
**Action Flow**

Action flow arrows illustrate the relationships among action states.



**Object Flow**

Object flow refers to the creation and modification of objects by activities. An object flow arrow from an action to an object means that the action creates or influences the object. An object flow arrow from an object to an action indicates that the action state uses the object.



**Initial State**

A filled circle followed by an arrow represents the initial action state.

Initial State

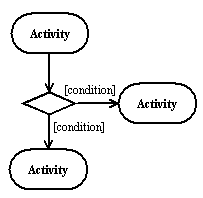
**Final State**

An arrow pointing to a filled circle nested inside another circle represents the final action state.

Final State

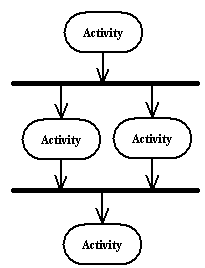
**Branching**

A diamond represents a decision with alternate paths. The outgoing alternates should be labeled with a condition or guard expression. You can also label one of the paths "else."



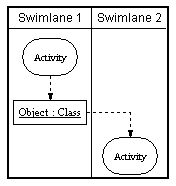
**Synchronization**

A synchronization bar helps illustrate parallel transitions. Synchronization is also called forking and joining.



**Swimlanes**

Swimlanes group related activities into one column.

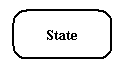
**State chart Diagram**

A state chart diagram shows the behavior of classes in response to external stimuli. This diagram models the dynamic flow of control from state to state within a system.

**Basic State chart Diagram Symbols and Notations**

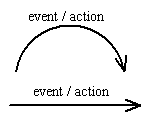
**States**

States represent situations during the life of an object. You can easily illustrate a state in Smart Draw by using a rectangle with rounded corners.



**Transition**

A solid arrow represents the path between different states of an object. Label the transition with the event that triggered it and the action that results from it.



**Initial State**

A filled circle followed by an arrow represents the object's initial state.

Initial State

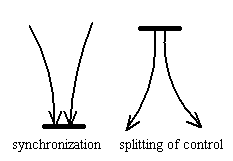
**Final State**

An arrow pointing to a filled circle nested inside another circle represents the object's final state.

Final State

**Synchronization and Splitting of Control**

A short heavy bar with two transitions entering it represents a synchronization of control. A short heavy bar with two transitions leaving it represents a splitting of control that creates multiple states.



**STATE CHART DIAGRAM:**

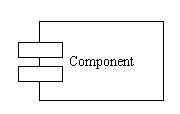
**What is a UML Component Diagram?**

A component diagram describes the organization of the physical components in a system.

**Basic Component Diagram Symbols and Notations**

**Component**

A component is a physical building block of the system. It is represented as a rectangle with tabs.  
[Learn how to resize grouped objects like components.](http://www.smartdraw.com/resources/tutorials/Objects)



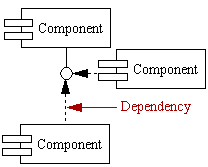
**Interface**

An interface describes a group of operations used or created by components.

Interface

**Dependencies**

Draw dependencies among components using dashed arrows.  
[Learn about line styles in SmartDraw.](http://www.smartdraw.com/resources/tutorials/Lines)



**COMPONENT DIAGRAM:**

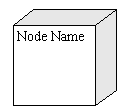
**What is a UML Deployment Diagram?**

Deployment diagrams depict the physical resources in a system including nodes, components, and connections.

**Basic Deployment Diagram Symbols and Notations**

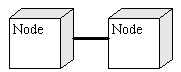
**Component**

A node is a physical resource that executes code components.  
[Learn how to resize grouped objects like nodes.](http://www.smartdraw.com/resources/tutorials/Objects)



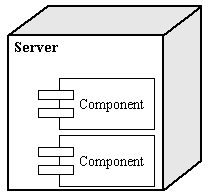
**Association**

Association refers to a physical connection between nodes, such as Ethernet.  
[Learn how to connect two nodes.](http://www.smartdraw.com/resources/tutorials/Lines)

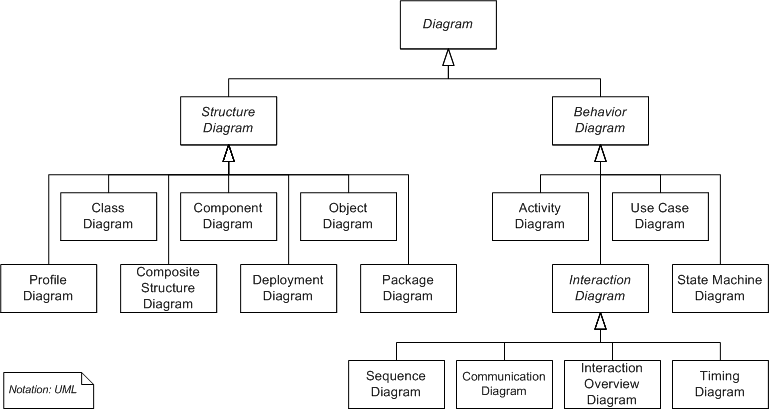


**Components and Nodes**

Place components inside the node that deploys them.



UML Diagrams Overview



UML combines best techniques from data modeling (entity relationship diagrams), business modeling (work flows), object modeling, and component modeling. It can be used with all processes, throughout the software development life cycle, and across different implementation technologies. UML has synthesized the notations of the Booch method, the Object-modeling technique (OMT) and Object-oriented software engineering (OOSE) by fusing them into a single, common and widely usable modeling language. UML aims to be a standard modeling language which can model concurrent and distributed systems.

**UML DIAGRAMS:**

**USECASE DIAGRAM:**

****

**CLASS DIAGRAM:**

****

**SEQUENCE DIAGRAM:**

****

**COLLABORATION DIAGRAM:**

****

**STATECHART DIAGRAM:**

****

**ACTIVITY DIAGRAM:**

****

**COMPONENT DIAGRAM:**

****

**DEPLOYMENT DIAGRAM:**

****

**Review of MRI-based brain tumor image segmentation using deep**

**learning methods**

**INTRODUCTION:**

Cancer can be defined as the uncontrolled, unnatural growth and division of the cells in the body. Occurrence, as a mass, of these unnatural cell growth and division in the brain tissue is called a brain tumor. While brain tumors are not very common, they are one of the most lethal cancers1.

Depending on their initial origin, brain tumors can be considered as either primary brain tumors or metastatic brain tumors. In primary ones, the origin of the cells are brain tissue cells, where in metastatic ones cells become cancerous at any other part of the body and spread into the brain. Gliomas are type of brain tumors that originate from glial cells.

They are the main type of brain tumors that current brain tumor segmentation research focuses on. The term glioma is a general term that is used to describe different types of gliomas ranging from low-grade gliomas like astrocytomas and oligodendrogliomas to the high grade (grade IV) glioblastoma multiform (GBM), which is the most aggressive and the most common primary malignant brain tumor2. Surgery, chemotherapy and radiotherapy are the techniques used, usually in combination, to treat gliomas3.

Early diagnosis of gliomas plays an important role in improving treatment possibilities. Medical Imaging techniques such as Computed Tomography (CT), Single-Photon Emission Computed Tomography (SPECT), Positron Emission Tomography (PET), Magnetic Resonance Spectroscopy (MRS) and Magnetic Resonance Imaging (MRI) are all used to provide valuable information about shape, size, location and metabolism of brain tumors assisting in diagnosis. While these modalities are used in combination to provide the highest detailed information about the brain tumors, due to its good soft tissue contrast and widely availability MRI is considered as the standard technique. MRI is a non-invasive in vivo imaging technique that uses radio frequency signals to excite target tissues to produce their internal images under the influence of a very powerful magnetic field.

Images of different MRI sequences are generated by altering excitation and repetition times during image acquisition. These different MRI modalities produce different types of tissue contrast images, thus providing valuable structural information and enabling diagnosis and segmentation of tumors along with their subregions4. Four standard MRI modalities used for glioma diagnosis include T1-weighted MRI (T1), T2-weighted MRI (T2), T1-weighted MRI with gadolinium contrast enhancement (T1-Gd) and Fluid Attenuated Inversion Recovery (FLAIR) (see Fig. 1). During MRI acquisition, although can vary from device to device, around one hundred and fifty slices of 2D images are produced to represent the 3D brain volume. Furthermore, when the slices from the required standard modalities are combined for diagnosis the data becomes very populated and complicated. Generally, T1 images are used for distinguishing healthy tissues, whereas T2 images are used to delineate the edema region which produces bright signal on the image. In T1-Gd images, the tumor border can easily be distinguished by the bright signal of the accumulated contrast agent (gadolinium ions) in the active cell region of the tumor tissue.

Since necrotic cells do not interact with the contrast agent, they can be observed by hypo intense part of the tumor core making it possible to easily segment them from the active cell region on the same sequence. In FLAIR images, signal of water molecules are suppressed which helps in distinguishing edema region from the Cerebrospinal Fluid (CSF). Before applying any therapy, it is crucial to segment the tumor in order to protect healthy tissues while damaging and destroying tumor cells during the therapy. Brain tumor segmentation involves diagnosing, delineating and separating tumor tissues, such as active cells, necrotic core and edema (Fig. 2) from normal brain tissues including Gray Matter (GM), White Matter (WM) and CSF. In current clinical routine, this task involves manual annotation and segmentation of large amount of multimodal MRI images. However, since manual segmentation is a very time consuming procedure, development of robust automatic segmentation methods, to provide efficient and objective segmentation, became an interesting and popular research area in recent years5. Current high segmentation performances obtained by deep learning methods make them good candidates for achieving this task. The rest of the paper is organized as follows: First we briefly review methods for brain tumor image segmentation in section 2. Then, in section 3, we especially focus on methods based on deep learning algorithms, which provide the state-of-the-art results in recent years. Inparticular, we compare designs of different deep learning methods and their performances. Finally, in conclusions, we assess the current state-of-the-art and provide future directions for development.

****

**2. Methods for Brain Tumor Image Segmentation**

Brain tumor segmentation methods can be classified as manual methods, semi-automatic methods and fully automatic methods based on the level of user interaction required6.

*2.1. Manual Segmentation Methods*

Manual segmentation requires the radiologist to use the multi-modality information presented by the MRI images along with anatomical and physiological knowledge gained through training and experience. Procedure involves the radiologist going through multiple slices of images slice by slice, diagnosing the tumor and manually drawing the tumor regions carefully. Apart from being a time consuming task, manual segmentation is also radiologist dependent and segmentation results are subject to large intra and inter rater variability7. However, manual segmentations are

widely used to evaluate the results of semi-automatic and fully automatic methods.

*2.2. Semi-Automatic Segmentation Methods*

Semi-automatic methods require interaction of the user for three main purposes; initialization, intervention or feedback response and evaluation8. Initialization is generally performed by defining a region of interest (ROI), containing the approximate tumor region, for the automatic algorithm to process. Parameters of pre-processing methods can also be adjusted to suit the input images. In addition to initialization, automated algorithms can be steered towards a desired result during the process by receiving feedbacks and providing adjustments in response. Furthermore, user can evaluate the results and modify or repeat the process if not satisfied. Hamamci et al. proposed the “Tumor Cut” method9. This semi-automatic segmentation method requires the user

to draw the maximum diameter of the tumor on input MRI images. After initialization a cellular automata (CA) based seeded tumor segmentation method run twice, once for tumor seeds provided by the user and once for the background seeds to obtain a tumor probability map. This approach includes separately applying the algorithm to each MRI modality (e.g. T1, T2, T1-Gd and FLAIR), then combining the results to obtain the final tumor volume. A recent semi-automatic method employed a novel classification approach10. In this approach segmentation

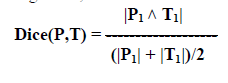
problem was transformed into a classification problem and a brain tumor is segmented by training and classifying within that same brain only. Generally, machine learning classification methods, for brain tumor segmentation, requires large amounts of brain MRI scans (with known ground truth) from different cases to train on. This results in a need to deal with intensity bias correction and other noises. However in this method, user initializes the process by selecting a subset of voxels belonging to each tissue type, from a single case. For these subsets of voxels, algorithm extracts the intensity values along with spatial coordinates as features and train a support vector machine (SVM) that is used to classify all the voxels of the same image to their corresponding tissue type. Despite semi-automatic brain tumor segmentation methods are less time consuming than manual methods and can obtain efficient results, they are still prone to intra and inter rater/user variability. Thus, current brain tumor segmentation research is mainly focused on fully automatic methods.



*BRATS Dataset*

Objective evaluation of the results of various brain tumor image segmentation methods with the state-of-the-art is a difficult task. However, with the development of a widely accepted benchmark, the BRATS benchmark5, for automatic brain tumor segmentation, now it is possible to objectively compare various glioma segmentation methods using this common dataset. Current version (2015) of the BRATS training dataset contains 274 multi-modality MRI

scans of patients with gliomas (both high and low grades) along with their ground truth segmentations for evaluation. As for testing data, 110 scans are available with unknown grades and unknown ground truths. Evaluation on the testing data is only possible with the online evaluation tool. Results are presented by the tool mainly in the form of well-known Dice Score, Sensitivity (true positive rate) and Specificity (true negative rate) for three main tumor regions; whole tumor (all tumor components), core tumor (all tumor components except edema) and active tumor (only active cells). We only report dice scores as performance measures. For each tumor region, *P1* represents the segmented tumor area by the proposed method, and *T1* is the actual tumor area in the ground truth. Then, dice score is calculated by the online tool for each region as;



**Deep Learning Methods**

Recent performances of deep learning methods, specifically Convolutional Neural Networks (CNNs), in several object recognition25 and biological image segmentation26 challenges increased their popularity among researches. In contrast to traditional classification methods, where hand crafted features are fed into, CNNs automatically learn representative complex features directly from the data itself. Due to this property, research on CNN based brain

tumor segmentation mainly focuses on network architecture design rather than image processing to extract features. CNNs take patches extracted from the images as inputs and use trainable convolutional filters and local subsampling to extract a hierarchy of increasingly complex features. Although currently very few in number compared to other traditional brain tumor segmentation methods, due to state-of-the-art results obtained by CNN based brain tumor

segmentation methods, we will focus the review on these methods in this section. Comparison of the reviewed deep learning and traditional glioma segmentation methods is presented in Table 1.

Urban et al. proposed a 3D CNN architecture for the multi-modal MRI glioma segmentation task27. Multimodality 3D patches, basically cubes of voxels, extracted from the different brain MRI modalities are used as inputs to a CNN to predict the tissue label of the center voxel of the cube. Input has 3D spatial intensity information and one additional dimension for MRI modalities. Thus 4D input data is handled effectively by the CNN. While high dimensional processing can better represent 3D nature of biological structures, it also increases processing load of the network. As for the architecture, two different networks are designed. The first one is a four layer CNN with the input layer containing 15 3D filters that have 53 spatial dimensions with an additional 4th dimension accounting for the corresponding MRI modality resulting in a filter shape of 5 x 5 x 5 x 4. Two of the hidden layer filters also have 53 spatial dimensions plus one dimension which corresponds to the number of filters in the preceding layer. Number

of filters in each hidden layer is 25. The last layer, the softmax layer contains 6 filters one for each tissue type to be classified allowing the interpretation of the output as probabilities (see Fig.3. for example architecture). The second network is almost identical with the exception of an additional hidden layer with 40 filters of size 53. Connected components are used to post-process the results. Reported average results of the two proposed networks are promising with BRATS dice scores of 87% for the whole tumor region, 77% for the core tumor region and 73% for

the active tumor region.

In contrast to the high dimensional method of Urban et al., Zikic et al. developed an interpretation method to transform the 4D data, so that standard 2D-CNN architectures can be used to solve the brain tumor segmentation task28. This can remove the burden of high dimensional CNN design while increasing computational efficiency. Interpretation is done by transforming each 4-modalitiy 3D input patch of size (*d1* x *d2* x *d3* x 4) into 4.*d3*-channel of

2D patches of size (*d1* x *d2* x 4*d3*). With this method, input patches of size 19x19x4 (single slice is used for each.

While Urban et al. used hyperbolic tangent function, this method applied rectified linear unit (ReLU) as a nonlinearity term. No post-processing is applied. Reported results indicate BRATS dice scores of 83.7% for the whole tumor region, 73.6% for core tumor region and 69% for active tumor region. It is important to note that, these results are obtained with a limited dataset which might affect the performance. Another novel approach implemented a cascaded two-pathway CNN architecture29. By extracting smaller sized patches and larger sized patches at the same time, a cascaded CNN that process local details of the brain MRI along with larger context of brain tissue is realized. Centred at the same location of the image, patches sized 33 x 33 pixels are extracted from each different MRI modality for local pathway and patches sized 65 x 65 are extracted for global pathway to classify the label of the central pixel. 2D multi-modality global input patches of size 65 x 65 x 4 are first processed by a CNN to output patches of size 33 x 33 x 5. Those output patches are then concatenated with the local patches of size 33x33x4 and fed as an input to a two-pathway CNN with convolutional layers containing 7 x 7 sized filters in one path and 13 x 13 sized filters in the other one. Thus, creating cascaded two-pathway CNN architecture. Several modified architectures of this cascaded CNN method are also proposed. Along with this novel architectural approach, two phase training is also implemented to avoid class imbalances. In first phase, cascaded CNN is trained with balanced distribution of classes and later in the second phase CNN is retrained with a more representative

distribution of the original images. Furthermore, Maxout non-linearity is used and connected components method is implemented as a post-processing step. High BRATS dice scores of 88% for whole tumor region, 79% for core tumor region and 73% for active tumor region are reported. A similar two-pathway approach with only one CNN is also proposed30.

One of the recent CNN approaches31 evaluated the brain tumor segmentation performance of using deeper CNN architectures. This approach is realized by implementing small 3 x 3 sized filters in the convolutional layers. In this way, more convolutional layers can be added to the architecture without reducing the effective receptive field of the

traditional bigger filters. Furthermore, deeper architectures apply more non-linearities and have less filter weights, due to the use of smaller filters, reducing the chance of overfitting. Modified version of ReLU, leaky rectifier linear unit (LReLU) is used as non-linearity activation function. Proposed CNN that has 11 layers of depth (6 convolutional layers followed by 3 fully-connected layers with 2 max-pooling layers dividing them into blocks of three) obtained BRATS dice scores of 88%, 83% and 77% for whole tumor, core tumor and active tumor regions

respectively. Implementation of intensity normalization, intensity bias correction and input patch augmentation as pre-processing operations along with threshold based unwanted cluster removal as post- processing contributed to the state of the art results. Some of the glioma segmentation methods combined CNN application with other classification or clustering techniques. In one method a local structured prediction with CNN is proposed32. Instead of using CNNs to classify

central voxels of input image patches into brain tissue classes, first patches of labels are extracted from ground truth images and then clustered by k-means algorithm into N groups to form a label patch dictionary of size N. Later, a 2D CNN is used to classify multimodal input image patches into one of these clusters. As for the segmentation performance of the method, BRATS dice scores of 83%, 75% and 77% for whole tumor, core tumor and active tumor regions are reported respectively. On the other hand Rao et al.33 extracted multi plane patches around each

pixel and trained four different CNNs each taking input patches from a separate MRI modality image. Outputs of the last hidden layers of those CNNs are then concatenated and used as feature maps to train a RF classifier.

**SYSTEM TESTING AND IMPLEMENTATION**

* 1. **INTRODUCTION**

Software testing is a critical element of software quality assurance and represents the ultimate review of specification, design and coding. In fact, testing is the one step in the software engineering process that could be viewed as destructive rather than constructive.

A strategy for software testing integrates software test case design methods into a well-planned series of steps that result in the successful construction of software. Testing is the set of activities that can be planned in advance and conducted systematically. The underlying motivation of program testing is to affirm software quality with methods that can economically and effectively apply to both strategic to both large and small-scale systems.

**8.2. STRATEGIC APPROACH TO SOFTWARE TESTING**

The software engineering process can be viewed as a spiral. Initially system engineering defines the role of software and leads to software requirement analysis where the information domain, functions, behavior, performance, constraints and validation criteria for software are established. Moving inward along the spiral, we come to design and finally to coding. To develop computer software we spiral in along streamlines that decrease the level of abstraction on each turn.

A strategy for software testing may also be viewed in the context of the spiral. Unit testing begins at the vertex of the spiral and concentrates on each unit of the software as implemented in source code. Testing progress by moving outward along the spiral to integration testing, where the focus is on the design and the construction of the software architecture. Talking another turn on outward on the spiral we encounter validation testing where requirements established as part of software requirements analysis are validated against the software that has been constructed. Finally we arrive at system testing, where the software and other system elements are tested as a whole.

UNIT TESTING

MODULE TESTING

SUB-SYSTEM TESING

SYSTEM TESTING

ACCEPTANCE TESTING

Component Testing

Integration Testing

User Testing

**8.3. UNIT TESTING**

Unit testing focuses verification effort on the smallest unit of software design, the module. The unit testing we have is white box oriented and some modules the steps are conducted in parallel.

**1. WHITE BOX TESTING**

This type of testing ensures that

* All independent paths have been exercised at least once
* All logical decisions have been exercised on their true and false sides
* All loops are executed at their boundaries and within their operational bounds
* All internal data structures have been exercised to assure their validity.

To follow the concept of white box testing we have tested each form .we have created independently to verify that Data flow is correct, All conditions are exercised to check their validity, All loops are executed on their boundaries.

**2. BASIC PATH TESTING**

Established technique of flow graph with Cyclomatic complexity was used to derive test cases for all the functions. The main steps in deriving test cases were:

Use the design of the code and draw correspondent flow graph.

Determine the Cyclomatic complexity of resultant flow graph, using formula:

V(G)=E-N+2 or

V(G)=P+1 or

V (G) =Number Of Regions

Where V (G) is Cyclomatic complexity,

E is the number of edges,

N is the number of flow graph nodes,

P is the number of predicate nodes.

Determine the basis of set of linearly independent paths.

**3. CONDITIONAL TESTING**

In this part of the testing each of the conditions were tested to both true and false aspects. And all the resulting paths were tested. So that each path that may be generate on particular condition is traced to uncover any possible errors.

**4. DATA FLOW TESTING**

This type of testing selects the path of the program according to the location of definition and use of variables. This kind of testing was used only when some local variable were declared. The *definition-use chain* method was used in this type of testing. These were particularly useful in nested statements.

**5. LOOP TESTING**

In this type of testing all the loops are tested to all the limits possible. The following exercise was adopted for all loops:

* All the loops were tested at their limits, just above them and just below them.
* All the loops were skipped at least once.
* For nested loops test the inner most loop first and then work outwards.
* For concatenated loops the values of dependent loops were set with the help of
* connected loop.Unstructured loops were resolved into nested loops or concatenated loops and tested as above.

# 9.1 INTRODUCTION

The protection of computer based resources that include hardware, software, data, procedures and people against unauthorized use or natural Disaster is known as System Security.

System Security can be divided into four related issues:

* Security
* Integrity
* Privacy
* Confidentiality

**SYSTEM SECURITY** refers to the technical innovations and procedures applied to the hardware and operation systems to protect against deliberate or accidental damage from a defined threat.

**DATA SECURITY** is the protection of data from loss, disclosure, modification and destruction.

**SYSTEM INTEGRITY** refers to the power functioning of hardware and programs, appropriate physical security and safety against external threats such as eavesdropping and wiretapping.

**PRIVACY** defines the rights of the user or organizations to determine what information they are willing to share with or accept from others and how the organization can be protected against unwelcome, unfair or excessive dissemination of information about it.

**CONFIDENTIALITY** is a special status given to sensitive information in a database to minimize the possible invasion of privacy. It is an attribute of information that characterizes its need for protection.

**9.3 SECURITY SOFTWARE**

It is the technique used for the purpose of converting communication. It transfers message secretly by embedding it into a cover medium with the use of information hiding techniques. It is one of the conventional techniques capable of hiding large secret message in a cover image without introducing many perceptible distortions.

NET has two kinds of security:

* Role Based Security
* Code Access Security

The Common Language Runtime (CLR) allows code to perform only those operations that the code has permission to perform. So CAS is the CLR's security system that enforces security policies by preventing unauthorized access to protected **resources** and **operations**. Using the Code Access Security, you can do the following:

* Restrict what your code can do
* Restrict which code can call your code
* Identify code

Automatic segmentation of the brain tumors for cancer diagnosis is a challenging task. Recently, availability of public datasets and the well-accepted BRATS benchmark provided a common medium for the researchers to develop and objectively evaluate their methods with the existing techniques. In this paper, we provided a review of the state-of-the-art methods based on deep learning, and a brief overview of traditional techniques. With the reported high performances, deep learning methods can be considered as the current state-of-the-art for glioma segmentation.

In traditional automatic glioma segmentation methods, translating prior knowledge into probabilistic maps or selecting highly representative features for classifiers is challenging task. However, convolutional neural networks (CNN) have the advantage of automatically learning representative complex features for both healthy brain tissues and tumor tissues directly from the multi-modal MRI images.

**Future enhancement:**

Future improvements and modifications in CNN architectures and addition of complementary information from other imaging modalities such as Positron Emission Tomography (PET), Magnetic Resonance Spectroscopy (MRS) and Diffusion Tensor Imaging (DTI) may improve

the current methods, eventually leading to the development of clinically acceptable automatic glioma segmentation methods for better diagnosis.