**Project Report**

**On**

**The Simulation of Queuing Model for Bangkok Rapid Transit Train Ticket System Using Python**

**Project work submitted in partial fulfillment**

**of the requirement for the award of the degree**

**Master of Computer Applications**

**By**

###### XXXXXXXXX

## ( Regd.No: XXXXXXXX )

## Under the Guidance of

## Mr. XXXXXXXX

## (Project Coordinator, XXXXXXXXXXX)

<Paste your university emblem here>

**XXXXXXX UNIVERSITY**

**CERTIFICATE**

This is to certify that this project entitled **“Mass Rapid Transport System”** is a bonafide work carried out by **XXXXXXXX** bearing **Hall Ticket No: XXXXXX** in **XXXXXXXXXXXXX** and submitted to **XXXXX University** in partial fulfillment of the requirements for the award of Master **of Computer Applications.**

**Project Guide External Examiner Principal**

#### ACKNOWLEDGMENT

“Task successful” makes everyone happy. But the happiness will be gold without glitter if we didn’t state the persons who have supported us to make it a success. Success will be crowned to people who made it a reality but the people whose constant guidance and encouragement made it possible will be crowned first on the eve of success. This acknowledgement transcends the reality of formality when we would like to express deep gratitude and respect to all those people behind the screen who guided, inspired and helped me for the completion of our project work.

I consider myself lucky enough to get such a good project. This project would add as an asset to my academic profile. I would like to express my thankfulness to my project guide, **Mr. XXXXX** for his constant motivation and valuable help through the project work, and I express my gratitude to **Mr. XXXXXXX**, Director of XXXXXXXXX, Hyderabad, for his constant supervision, guidance and co-operation throughout the project. I also extend my thanks to my Team Members for their co-operation during my course. Finally I would like to thanks my friends for their co-operation to complete this project.

XXXXXXX

**The Simulation of Queuing Model for Bangkok Rapid Transit Train Ticket System Using Python**

**ABSTRACT:**

This paper proposes the simulation model for ticket system of Bangkok rapid transit train. The proposed analysis model is applied by using queuing theory to analyze main queuing delay problem rapid transit train ticket machines. This analysis model has been developed by using python programming language to create software tool for analyzing the existing ticket system comparing with the desire variable condition or redesign the ticket system. Since, proposed software tool use for analyze the effect of existing ticket system by modification to visualize the queuing, delay time, waiting time and etc. The software tool with the simulation model can visualize the 2D animation results and graph for analyzation. The proposed analysis model could be used to any rapid transit train ticket system for impact analysis own existing ticket system.

**CONTENTS**

1. **INTRODUCTION**
   1. INTRODUCTION TO PROJECT
   2. PURPOSE OF THE PROJECT
   3. PROBLEM IN EXISTING SYSTEM
   4. SOLUTION OF THESE PROBLEMS
2. **SYSTEM ANALYSIS**
   1. STUDY OF THE SYSTEM
   2. PROPOSED SYSTEM
   3. INPUT & OUTPUT
   4. PROCESS MODELS USED WITH JUSTIFICATION
   5. SYSTEM ARCHITECTURE
3. **FEASIBILITY STUDY**
   1. TECHNICAL FEASIBILITY
   2. OPERATIONAL FEASIBILITY
   3. ECONOMIC FEASIBILITY
4. **SOFTWARE REQUIREMENT SPECIFICATIONS**
   1. FUNCIONAL REQUIREMENTS
   2. PERFORMANCE REQUIREMENTS
   3. HARDWARE REQUIREMENTS
   4. SOFTWARE REQUIREMENTS
      1. INTRODUCTION TO PYTHON FRAMEWORK
      2. PYTHON
5. **SYSTEM DESIGN**
   1. INTRODUCTION
   2. NORMALIZATION
   3. DATA FLOW DIAGRAMS
   4. UML DIAGRAMS
6. **OUTPUT SCREENS**
7. **SYSTEM AND IMPLEMENTATION**
   1. INTRODUCTION TO TESTING
   2. TESTING STRATEGIES
   3. IMPLEMENTATION
8. **SYSTEM SECURITY**

8.1 INTRODUCTION

8.2 SECURITY IN SOFTWARE

1. **CONCLUSION**
2. **FUTURE SCOPE**
3. **BIBLIOGRAPHY**

**1.1 INTRODUCTION**

Bangkok is the capital of Thailand and has a population near to ten Million in 2019. Like all other large cities in the world, road traffic congestion is a common and very serious problem. The government of Thailand came to realize that mass rapid transit is necessary to serve as a public transportation system for moving passengers, thereby reducing road congestion. It was not until 19 November 1996, just before the Thailand economic crisis in 1997, the construction of a massive rapid transit system in Bangkok commenced which spans over 19.8 kilometer, covering 18 stations from Bang Sue to Hua Lamphong. After several delays, on 3 July 2004 the line was officially opened. Over the last decades, the number of passengers using mass rapid transit is increasing so significantly that more construction of mass rapid transit lines soon follow using underground, elevated or surface trains. At present, mass rapid transit in Bangkok consists of three networks, the Bangkok Mass Transit System (BTS), the Metropolitan Rapid Transit (MRT) and the Airport Rail Link (ARL). According to the statistics in 2017, this Chaloem Ratchamongkhon (Blue) line carries more than 360,000 passengers daily.

This huge number of passengers has indeed caused negative impact on the ticketing system at each station. At peak hours, passengers face difficulties in purchasing ticket due to inadequate ticket machines resulting long queuing delay. This causes unsatisfaction particularly among tourist travelers. In response to this problem, reference [1] applied some basic queuing model to approximately analyze and able to roughly identify the main causes of delay. An analysis model or simulation model of the ticket machine was proposed [2,3]. They discuss issues that focus on specific criteria in each country. In this paper, we propose to develop a more accurate simulation model that incorporates actual traffic data from major stations during peak hours. While collecting real traffic, we observe that congested and long queues can build up at the entrance/exit gates between the ticket system and the trains. Therefore, this nontrivial observation has been included in to our simulation model to achieve more realistic investigation. The proposed simulation model has been implemented in python and extensively tested with real traffic data. Our developed software tool provides graphical visualization for the queuing analysis with 2D Graph and animation.

**1.2 Purpose of the project**

From the theoretical model by previous section, we propose the simulation model of ticket system by making propose simulation model using python programming. The propose simulation model could be defined by number of machines of the ticket system. The simulation tools can make the comparison of the previous system or the newer system by own designating the parameter. The examples of the simulation tools.

**1.3 Problem statement**

1. Passengers of Bangkok Mass Transit System (BST) are not getting excellent customer service from the Automatic Ticket Vending Machine (ATVM).
2. At peak hours, passengers face difficulties in purchasing ticket due to inadequate ticket machines resulting long queuing delay.

**1.4 Solution for the problem statement:**

In response to this problem, applied some basic queuing model to approximately analyze and able to roughly identify the main causes of delay. An analysis model or simulation model of the ticket machine was proposed. They discuss issues that focus on specific criteria in each country. In this paper, we propose to develop a more accurate simulation model that incorporates actual traffic data from major stations during peak hours. While collecting real traffic, we observe that congested and long queues can build up at the entrance/exit gates between the ticket system and the trains. Therefore, this nontrivial observation has been included in to our simulation model to achieve more realistic investigation.

**2.1 STUDY OF THE SYSTEM**

1. Numpy
2. Pandas
3. Matplotlib
4. Turtle
5. Queuing Theory

**1 . Numpy:**

Numpy is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays.

It is the fundamental package for scientific computing with Python. It contains various features including these important ones:

* A powerful N-dimensional array object
* Sophisticated (broadcasting) functions
* Tools for integrating C/C++ and Fortran code
* Useful linear algebra, Fourier transform, and random number capabilities

Besides its obvious scientific uses, Numpy can also be used as an efficient multi-dimensional container of generic data. Arbitrary data-types can be defined using Numpy which allows Numpy to seamlessly and speedily integrate with a wide variety of databases.

**2.Pandas**

Pandas is an open-source Python Library providing high-performance data manipulation and analysis tool using its powerful data structures. Python was majorly used for data munging and preparation. It had very little contribution towards data analysis. Pandas solved this problem. Using Pandas, we can accomplish five typical steps in the processing and analysis of data, regardless of the origin of data load, prepare, manipulate, model, and analyze. Python with Pandas is used in a wide range of fields including academic and commercial domains including finance, economics, Statistics, analytics, etc.

**3.Matplotlib**

Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, the Python and [IPython](http://ipython.org/) shells, the [Jupyter](http://jupyter.org/) notebook, web application servers, and four graphical user interface toolkits. Matplotlib tries to make easy things easy and hard things possible. You can generate plots, histograms, power spectra, bar charts, error charts, scatter plots, etc., with just a few lines of code. For examples, see the [sample plots](https://matplotlib.org/tutorials/introductory/sample_plots.html) and [thumbnail gallery](https://matplotlib.org/gallery/index.html).

For simple plotting the pyplot module provides a MATLAB-like interface, particularly when combined with IPython. For the power user, you have full control of line styles, font properties, axes properties, etc, via an object oriented interface or via a set of functions familiar to MATLAB users.

**4.Turtle**

Why is it called turtle in Python?

Why is the turtle module in Python called that? As it says in the Python Standard Library Doc, it's because Turtle graphics is a popular way of introducing programming to children. It was part of the original Logo programming language developed by Wally Feurzig and Seymour Papert in 1966.

What is the use of turtle in Python?

“Turtle” is a Python feature like a drawing board, which lets us command a turtle to draw all over it! We can use functions like turtle.forward(…) and turtle.right(…) which can move the turtle around

**5.Queuing Theory**

Queuing theory is the mathematical study of the congestion and delays of waiting in line. Queuing theory (or "queueing theory") examines every component of waiting in line to be served, including the arrival process, service process, number of servers, number of system places, and the number of customers—which might be people, data packets, cars, etc.

## Types of Queues and Types of Service

**First In First Out**, or First Come First Served, is fairly common in banking and commerce. It is the type of queue you get when you have people politely lined up, waiting for their turn.

**Last In First Out** is the opposite scheme; whoever has been waiting for the shortest time is served first. This type of queue management is common in asset management, where assets produced or acquired last are the ones used or disposed of first. For example: the most recent employees are often the ones laid off first.

**Priority** is where customers are served based on their priority level; these levels could be based on status, task urgency, or some other criteria.

**Shortest Job First** is when whoever needs the shortest amount of service gets taken care of first

**Processor Sharing** is when everyone gets served, or half-served, at the same time; service capacity is distributed evenly among everyone waiting.

There may be a **single server**, where a line of people or items must go through a single bottleneck, or **parallel servers**, where the same line is served by several servers.  Or there may be a **tandem queue**, where each of multiple servers has their own queue or line.

**Balking** when a customer decides not to wait for service because the wait time threatens to be too long. **Reneging** is similar, but when a customer who has waited already decides to leave because they’ve wasted too much time. **Jockeying** is when a customer switches between queues in a tandem queue system, trying to orchestrate the shortest wait possible.

## Standard Notation for Queueing Theory

To make life easier, there’s standard notation for queueing theory that is used across the board. These standard symbols include

* λ: the [mean](https://www.statisticshowto.datasciencecentral.com/probability-and-statistics/statistics-definitions/mean-median-mode/#mean)arrival rate.
* μ: the mean service rate.
* n: the number of people in the system.
* A: the arrival process [probability distribution](https://www.statisticshowto.datasciencecentral.com/probability-distribution/).
* B: the service process probability distribution.
* C: the number of servers.
* D: the maximum number of customers allowed in the system at any given time, waiting or being served (without getting bumped).
* E: the maximum number of customers total.

**2.2. PROPOSED SYSTEM**

The propose simulation model could be defined by number of machines of the ticket system. The simulation tools can make the comparison of the previous system or the newer system by own designating the parameter. The examples of the simulation tools.

The proposed analysis model can be used to evaluate the service time of railway companies, which enables for the operators to form a proper policy and decide on different ticket system operations based on cost requirements. The proposed simulation model has been implemented in python software tool for analyzing the effect time delay of queues or average waiting time for passengers by comparing ticket existing system and desire modification system. Since, the proposed simulation of queuing model presents the results to show the impact of time delay in queue of passengers when ticket machines are modified by number or feature upgrading. The proposed analysis model could potentially be used to any rapid transit train ticket system for impact evaluation on the existing ticket system.

we developed accurate simulation model that incorporates actual traffic data from major stations during peak hours. While collecting real traffic, we observe that congested and long queues can build up at the entrance/exit gates between the ticket system and the trains. Therefore, this nontrivial observation has been included in to our simulation model to achieve more realistic investigation. The proposed simulation model has been implemented in python and extensively tested with real traffic data. Our developed software tool provides graphical visualization for the queuing analysis with 2D Graph and animation.

**2.3. INPUT AND OUTPUT**

**Inputs and Outputs**

The following some are the projects inputs and outputs.

**Inputs:**

* Importing the all required packages like Numpy, Pandas, Matplotlib, Queuing theory algorithm ,turtle and required machine learning algorithms packages .
* Setting the dimensions of visualization graph.
* Downloading and importing the libraries.

**Outputs:**

* Displaying of basic statistics of queue i.e. average waiting time, arrival rate etc .
* Simulation of queue and server using turtle python library .
* Visualization of data of previous and proposed system .

**2.4. PROCESS MODELS USED WITH JUSTIFICATION**

**SDLC MODEL:**

**Software Development Life Cycle (SDLC)**



The Software Development Lifecycle(SDLC) for small to medium database application development efforts.

This project uses iterative development lifecycle, where components of the application are developed through a series of tight iteration. The first iteration focus on very basic functionality, with subsequent iterations adding new functionality to the previous work and or correcting errors identified for the components in production.

The six stages of the SDLC are designed to build on one another, taking outputs from the previous stage, adding additional effort, and producing results that leverage the previous effort and are directly traceable to the previous stages. During each stage, additional information is gathered or developed, combined with the inputs, and used to produce the stage deliverables. It is important to not that the additional information is restricted in scope, new ideas that would take the project in directions not anticipated by the initial set of high-level requirements or features that are out-of-scope are preserved for later consideration

.

Too many software development efforts go awry when development team and customer personnel get caught up in the possibilities of automation. Instead of focusing on high priority features, the team can become mired in a sea of nice to have features that are not essential to solve the problem, but in themselves are highly attractive. This is the root cause of large percentage of failed and or abandoned development efforts and is the primary reason the development team utilizes the iterative model.

**Roles and Responsibilities of PDR AND PER**

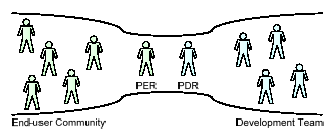
The iterative lifecycle specifies two critical roles that act together to clearly communicate project issues and concepts between the end-user community and the development team.

**Primary End-user Representative (PER)**

The PER is a person who acts as the primary point of contact and principal approver for the end-user community. The PER is also responsible for ensuring that appropriate subject matter experts conduct end-user reviews in a timely manner.

**PER-PDR Relationship**

The PER and PDR are the brain trust for the development effort. The PER has the skills and domain knowledge necessary to understand the issues associated with the business processes to the supported by the application and has a close working relationship with the other members of the end-user community. The PDR has the same advantages regarding the application development process and the other members of the development team together, they act as the concentration points for knowledge about the application to be developed.



The objective of this approach is to create the close relationship that is characteristic of a software project with one developer and one end-user in essence, this approach the “pair programming” concept from Agile methodologies and extends it to the end-user community. While it is difficult to create close relationships between the diverse members of an end-user community and a software development team, it is much simpler to create a close relationship between the lead representatives for each group.

When multiple end-users are placed into relationship with multiple members of a development team, communication between the two groups degrades as the number of participants grows. In this model, members of end-user community may communicate with members of the development team as needed, but it is the responsibility of all participants to keep the PER and PDR apprised of the communications for example, this allows the PER and PDR to resolve conflicts that arise when two different end-users communicate different requirements for the same application feature to different members of the development team.

**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 before the information gets stored in the database media:

Ex: In this project voter either existing or new user data will be stored in database as the inputs given by users….

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

**OUTPUT DESIGN**

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 with

The outputs were needed to be generated as a hard copy and as well as queries to be viewed on the screen. Keeping in view these outputs, the format for the output is taken from the outputs, which are currently being obtained after manual processing. The standard printer is to be used as output media for hard copies.

**DESIGN PRINCIPLES & METHODOLOGY:**

**Object Oriented Analysis And Design**

When Object orientation is used in analysis as well as design, the boundary between OOA and OOD is blurred. This is particularly true in methods that combine analysis and design. One reason for this blurring is the similarity of basic constructs (i.e.,objects and classes) that are used in OOA and OOD. Through there is no agreement about what parts of the object-oriented development process belongs to analysis and what parts to design, there is some general agreement about the domains of the two activities.

The fundamental difference between OOA and OOD is that the former models the problem domain, leading to an understanding and specification of the problem, while the latter models the solution to the problem. That is, analysis deals with the problem domain, while design deals with the solution domain. However, in OOAD subsumed in the solution domain representation. That is, the solution domain representation, created by OOD, generally contains much of the representation created by OOA. The separating line is matter of perception, and different people have different views on it. The lack of clear separation between analysis and design can also be considered one of the strong points of the object-oriented approach the transition from analysis to design is “seamless”. This is also the main reason OOAD methods-where analysis and designs are both performed.

The main difference between OOA and OOD, due to the different domains of modeling, is in the type of objects that come out of the analysis and design process.

**Features of OOAD:**

* It users Objects as building blocks of the application rather functions
* All objects can be represented graphically including the relation between them.
* All Key Participants in the system will be represented as actors and the actions done by them will be represented as use cases.
* A typical use case is nothing bug a systematic flow of series of events which can be well described using sequence diagrams and each event can be described diagrammatically by Activity as well as state chart diagrams.
* So the entire system can be well described using OOAD model, hence this model is chosen as SDLC model.

**THE GENESIS OF UML:**

Software engineering has slowly become part of our everyday life. From washing machines to compact disc player, through cash machines and phones, most of our daily activities use software, and as time goes by, the more complex and costly this software becomes.

The demand for sophisticated software greatly increases the constraints imposed on development teams. Software engineers are facing a world of growing complexity due to the nature of applications, the distributed and heterogeneous environments, the size of programs, the organization of software development teams, and the end-users ergonomic expectations.

To surmount these difficulties, software engineers will have to learn not only how to do their job, but also how to explain their work to others, and how to understand when others work is explained to them. For these reasons, they have (and will always have) an increasing need for methods.

**From Functional to Object-Oriented Methods**

Although object-oriented methods have roots that are strongly anchored back in the 60s, structured and functional methods were the first to be used. This is not very surprising, since functional methods are inspired directly my computer architecture (a proven domain well known to computer scientists). The separation of data and code, just as exists physically in the hardware, was translated into the methods; this is how computer scientists got into the habit of thinking in terms of system functions.

This approach is natural when looked at in its historical context, but today, because of its lack of abstraction, it has become almost completely anachronistic. There is no reason to impose the underlying hardware on a software solution. Hardware should act as the servant of the software that is executed on it, rather than imposing architectural constraints.

**TOWARDS A UNIFIED MODELLING LANGUAGE**

The unification of object-oriented modeling methods became possible as experience allowed evaluation of the various concepts proposed by existing methods. Based on the fact that differences between the various methods were becoming smaller, and that the method wars did not move object-oriented technology forward any longer, Jim Rumbaugh and Grady Booch decided at the end of 1994 to unify their work within a single method: the Unified Method. A year later they were joined by Ivar Jacobson, the father of use cases, a very efficient technique for the determination of requirements.

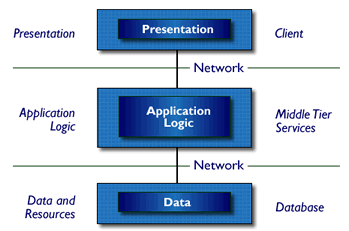
Booch, Rumbaugh and Jacobson adopted four goals:

* To represent complete systems (instead of only the software portion) using object oriented concepts.
* To establish an explicit coupling between concepts and executable code.
* To take into account the scaling factors that are inherent to complex and critical systems.
* To creating a modeling language usable by both humans and machines.

**2.6. SYSTEM ARCHITECTURE**

The current application is being developed by taking the 3-tier architecture as a prototype. The 3-tier architecture is the most common approach used for web applications today. In the typical example of this model, the web browser acts as the client, IIS handles the business logic, and a separate tier MS-SQL Server handles database functions.

Although the 3-tier approach increases scalability and introduces a separation of business logic from the display and database layers, it does not truly separate the application into specialized, functional layers. For prototype or simple web applications, the 3-tier architecture may be sufficient. However, with complex demands placed on web applications, a 3-tiered approach falls short in several key areas, including flexibility and scalability. These shortcomings occur mainly because the business logic tier is still too broad- it has too many functions grouped into one tier that could be separated out into a finer grained model.



**Three-Tier Architecture**

* Tier 1: the client contains the presentation logic, including simple control and user input validation. This application is also known as a thin client. The client interface is developed using ASP.Net Server Controls and HTML controls in some occasions
* Tier 2: the middle tier is also known as the application server, which provides the business processes logic and the data access. The business logic/ business rules can be written either with C#.Net or VB.Net languages. These business runes will be deployed as DLL’s in IIS web server.
* Tier 3: the data server provides the business data. MS-SQL server acts as Tier-3, which is the database layer.

These are some of the advantages of three-tier architecture:

* It is easier to modify or replace any tier without affecting the other tiers.
* Separating the application and database functionality means better load balancing.
* Adequate security policies can be enforced within the server tiers without hindering the clients.

The proposed system can be designed perfectly with the three tier model, as all layers are perfectly getting set as part of the project. In the future, while expanding the system, in order to implement integration touch points and to provide enhanced user interfaces, the n-tier architecture can be used.

FEASIBILITY

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

**3.1 TECHINICAL 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 2008 and SQL Server 2005. These are free software that would be downloaded from web.

Visual Studio 2008 –it is tool or technology.

**3.2 OPERATIONAL FEASIBILITY:**

Not only must an application make economic and technical sense, it must also make operational sense.

**Issues to consider when determining the operational feasibility of a project.**

|  |  |
| --- | --- |
| **Operations Issues** | **Support Issues** |
| * What tools are needed to support operations? * What skills will operators need to be trained in? * What processes need to be created and/or updated? * What documentation does operations need? | * What documentation will users be given? * What training will users be given? * How will change requests be managed? |

Very often you will need to improve the existing operations, maintenance, and support infrastructure to support the operation of the new application that you intend to develop.  To determine what the impact will be you will need to understand both the current operations and support infrastructure of your organization and the operations and support characteristics of your new application.

To operate this application this system that the user no needs to require any technical knowledge that we are used to develop this project is. Asp.net, C#.net. that the application providing rich user interface by user can do the operation in flexible manner.

**3.3 ECONOMIC FEASIBILITY:**

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

In the present system, the development of new product greatly enhances the accuracy

of the system and cuts short the delay in the processing this application. The errors can be greatly reduced and at the same time providing a great level of security. Here we don’t need any additional equipment except memory of required capacity.

No need for spending money on client for maintenance because the database used is web enabled database.

A **Software Requirements Specification** (**SRS**) – a [requirements specification](http://en.wikipedia.org/wiki/Requirements_specification) for a [software system](http://en.wikipedia.org/wiki/Software_system) – is a complete description of the behavior of a system to be developed. It includes a set of [use cases](http://en.wikipedia.org/wiki/Use_case) that describe all the interactions the users will have with the software. In addition to use cases, the SRS also contains non-functional requirements. [Non-functional requirements](http://en.wikipedia.org/wiki/Non-functional_requirements) are requirements which impose constraints on the design or implementation (such as [performance engineering](http://en.wikipedia.org/wiki/Performance_engineering) requirements, [quality](http://en.wikipedia.org/wiki/Quality_%28business%29) standards, or design constraints).

**System requirements specification:** A structured collection of information that embodies the requirements of a system. A [business analyst](http://en.wikipedia.org/wiki/Business_analyst), sometimes titled [system analyst](http://en.wikipedia.org/wiki/System_analyst), is responsible for analyzing the business needs of their clients and stakeholders to help identify business problems and propose solutions. Within the [systems development life cycle](http://en.wikipedia.org/wiki/Systems_development_life_cycle) domain, typically performs a liaison function between the business side of an enterprise and the information technology department or external service providers. Projects are subject to three sorts of requirements:

* [**Business requirements**](http://en.wikipedia.org/wiki/Business_requirements) describe in business terms *what* must be delivered or accomplished to provide value.
* **Product requirements** describe properties of a system or product (which could be one of

Several ways to accomplish a set of business requirements.)

* **Process requirements** describe activities performed by the developing organization. For instance, process requirements could specify specific methodologies that must be followed, and constraints that the organization must obey.

Product and process requirements are closely linked. Process requirements often specify the activities that will be performed to satisfy a product requirement. For example, a maximum development cost requirement (a process requirement) may be imposed to help achieve a maximum sales price requirement (a product requirement); a requirement that the product be maintainable (a Product requirement) often is addressed by imposing requirements to follow particular development styles

**Purpose**

An systems engineering, a **requirement** can be a description of *what* a system must do, referred to as a [Functional Requirement](http://en.wikipedia.org/wiki/Functional_requirements). This type of requirement specifies something that the delivered system must be able to do. Another type of requirement specifies something about the system itself, and how well it performs its functions. Such requirements are often called [Non-functional requirements](http://en.wikipedia.org/wiki/Non-functional_requirements), or 'performance requirements' or 'quality of service requirements.' Examples of such requirements include usability, availability, reliability, supportability, testability and maintainability.

A collection of requirements define the characteristics or features of the desired system. A 'good' list of requirements as far as possible avoids saying *how* the system should implement the requirements, leaving such decisions to the system designer. Specifying how the system should be implemented is called "implementation bias" or "solution engineering". However, *implementation constraints* on the solution may validly be expressed by the future owner, for example for required interfaces to external systems; for interoperability with other systems; and for commonality (e.g. of user interfaces) with other owned products.

In software engineering, the same meanings of requirements apply, except that the focus of interest is the software itself.

**4.1 FUNCTIONAL Requirements**

Accuracy of model should be high then only we can get perfect results

Need to be analyze the data and remove the unwanted data, if there is any missing values there need to remove those missing values or else has to put suitable for it

Feature selection is the major part of the data analysis get the perfect feature to build model

**4.2 NON Functional Requirements**

The major non-functional Requirements of the system are as follows

**Usability**

The system is designed with completely automated process hence there is no or less user intervention.

**Reliability**

The system is more reliable because of the qualities that are inherited from the chosen platform java. The code built by using java is more reliable.

**Performance**

This system is developing in the high level languages and using the advanced front-end and back-end technologies it will give response to the end user on client system with in very less time.

**Supportability**

The system is designed to be the cross platform supportable. The system is supported on a wide range of hardware and any software platform, which is having JVM, built into the system.

**Implementation**

The system is implemented in web environment using struts framework. The apache tomcat is used as the web server and windows xp professional is used as the platform.

Interface the user interface is based on Struts provides HTML Tag

**4.3 Hardware Requirements:**

• RAM: 4GB and Higher

• Processor: Intel i3 and above

• Hard Disk: 500GB: Minimum

**4.4 Software Requirements:**

* OS: Windows or Linux
* Python IDE : python 2.7.x and above
* Setup tools and pip to be installed for 3.6 and above
* Language : Python Scripting

# 4.4.1. INTRODUCTION TO PYTHON

**Python**

### What Is A Script?

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

**Scripts are reusable**

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

**Scripts are editable**

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

**You will need a text editor**

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

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

**Difference between a script and a program**

Script:

Scripts are distinct from the core code of the application, which is usually written in a different language, and are often created or at least modified by the end-user. Scripts are often interpreted from source code or byte code, 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
  + 1. **Datascience**

**DATASETS**

**DataSets**  
The **DataSet** object is similar to the ADO **Recordset** object, but more powerful, and with one other important distinction: the **DataSet** is always disconnected. The **DataSet** object represents a cache of data, with database-like structures such as tables, columns, relationships, and constraints. However, though a **DataSet** can and does behave much like a database, it is important to remember that **DataSet** objects do not interact directly with databases, or other source data. This allows the developer to work with a programming model that is always consistent, regardless of where the source data resides. Data coming from a database, an XML file, from code, or user input can all be placed into **DataSet** objects. Then, as changes are made to the **DataSet** they can be tracked and verified before updating the source data. The **GetChanges** method of the **DataSet** object actually creates a second **DatSet** that contains only the changes to the data. This **DataSet** is then used by a **DataAdapter** (or other objects) to update the original data source.

The **DataSet** has many XML characteristics, including the ability to produce and consume XML data and XML schemas. XML schemas can be used to describe schemas interchanged via WebServices. In fact, a **DataSet** with a schema can actually be compiled for type safety and statement completion.

**5.1. 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.

During design, progressive refinement of data structure, program structure, and procedural details are developed reviewed and documented. System design can be viewed from either technical or project management perspective. From the technical point of view, design is comprised of four activities – architectural design, data structure design, interface design and procedural design.

**5.2. NORMALIZATION**

It is a process of converting a relation to a standard form. The process is used to handle the problems that can arise due to data redundancy i.e. repetition of data in the database, maintain data integrity as well as handling problems that can arise due to insertion, updation, deletion anomalies.

Decomposing is the process of splitting relations into multiple relations to eliminate anomalies and maintain anomalies and maintain data integrity. To do this we use normal forms or rules for structuring relation.

**Insertion anomaly**: Inability to add data to the database due to absence of other data.

**Deletion anomaly**: Unintended loss of data due to deletion of other data.

**Update anomaly**: Data inconsistency resulting from data redundancy and partial update

**Normal Forms**: These are the rules for structuring relations that eliminate anomalies.

**FIRST NORMAL FORM**:

A relation is said to be in first normal form if the values in the relation are atomic for every attribute in the relation. By this we mean simply that no attribute value can be a set of values or, as it is sometimes expressed, a repeating group.

**SECOND NORMAL FORM**:

A relation is said to be in second Normal form is it is in first normal form and it should satisfy any one of the following rules.

1. Primary key is a not a composite primary key
2. No non key attributes are present
3. Every non key attribute is fully functionally dependent on full set of primary key.

**THIRD NORMAL FORM**:

A relation is said to be in third normal form if their exits no transitive dependencies.

**Transitive Dependency**: If two non key attributes depend on each other as well as on the primary key then they are said to be transitively dependent.

The above normalization principles were applied to decompose the data in multiple tables thereby making the data to be maintained in a consistent state.

**5.3. E – R DIAGRAMS**

* + The relation upon the system is structure through a conceptual ER-Diagram, which not only specifics the existential entities but also the standard relations through which the system exists and the cardinalities that are necessary for the system state to continue.
  + The entity Relationship Diagram (ERD) depicts the relationship between the data objects. The ERD is the notation that is used to conduct the date modeling activity the attributes of each data object noted is the ERD can be described resign a data object descriptions.
  + The set of primary components that are identified by the ERD are

◆ Data object ◆ Relationships

◆ Attributes ◆ Various types of indicators.

The primary purpose of the ERD is to represent data objects and their relationships.

**5.4. DATA FLOW DIAGRAMS**

A data flow diagram is graphical tool used to describe and analyze movement of data through a system. These are the central tool and the basis from which the other components are developed. The transformation of data from input to output, through processed, may be described logically and independently of physical components associated with the system. These are known as the logical data flow diagrams. The physical data flow diagrams show the actual implements and movement of data between people, departments and workstations. A full description of a system actually consists of a set of data flow diagrams. Using two familiar notations Yourdon, Gane and Sarson notation develops the data flow diagrams. Each component in a DFD is labeled with a descriptive name. Process is further identified with a number that will be used for identification purpose. The development of DFD’S is done in several levels. Each process in lower level diagrams can be broken down into a more detailed DFD in the next level. The lop-level diagram is often called context diagram. It consists a single process bit, which plays vital role in studying the current system. The process in the context level diagram is exploded into other process at the first level DFD.

The idea behind the explosion of a process into more process is that understanding at one level of detail is exploded into greater detail at the next level. This is done until further explosion is necessary and an adequate amount of detail is described for analyst to understand the process.

Larry Constantine first developed the DFD as a way of expressing system requirements in a graphical from, this lead to the modular design.

A DFD is also known as a “bubble Chart” has the purpose of clarifying system requirements and identifying major transformations that will become programs in system design. So it is the starting point of the design to the lowest level of detail. A DFD consists of a series of bubbles joined by data flows in the system.

**DFD SYMBOLS:**

In the DFD, there are four symbols

1. A square defines a source(originator) or destination of system data
2. An arrow identifies data flow. It is the pipeline through which the information flows
3. A circle or a bubble represents a process that transforms incoming data flow into outgoing data flows.
4. An open rectangle is a data store, data at rest or a temporary repository of data

Process that transforms data flow.

Source or Destination of data

Data flow

Data Store

**CONSTRUCTING A DFD:**

Several rules of thumb are used in drawing DFD’S:

1. Process should be named and numbered for an easy reference. Each name should be representative of the process.
2. The direction of flow is from top to bottom and from left to right. Data traditionally flow from source to the destination although they may flow back to the source. One way to indicate this is to draw long flow line back to a source. An alternative way is to repeat the source symbol as a destination. Since it is used more than once in the DFD it is marked with a short diagonal.
3. When a process is exploded into lower level details, they are numbered.
4. The names of data stores and destinations are written in capital letters. Process and dataflow names have the first letter of each work capitalized

A DFD typically shows the minimum contents of data store. Each data store should contain all the data elements that flow in and out.

Questionnaires should contain all the data elements that flow in and out. Missing interfaces redundancies and like is then accounted for often through interviews.

**SAILENT FEATURES OF DFD’S**

1. The DFD shows flow of data, not of control loops and decision are controlled considerations do not appear on a DFD.
2. The DFD does not indicate the time factor involved in any process whether the dataflow take place daily, weekly, monthly or yearly.
3. The sequence of events is not brought out on the DFD.

**TYPES OF DATA FLOW DIAGRAMS**

1. Current Physical
2. Current Logical
3. New Logical
4. New Physical

**CURRENT PHYSICAL:**

In Current Physical DFD process label include the name of people or their positions or the names of computer systems that might provide some of the overall system-processing label includes an identification of the technology used to process the data. Similarly data flows and data stores are often labels with the names of the actual physical media on which data are stored such as file folders, computer files, business forms or computer tapes.

**CURRENT LOGICAL:**

The physical aspects at the system are removed as mush as possible so that the current system is reduced to its essence to the data and the processors that transforms them regardless of actual physical form.

**NEW LOGICAL**:

This is exactly like a current logical model if the user were completely happy with he user were completely happy with the functionality of the current system but had problems with how it was implemented typically through the new logical model will differ from current logical model while having additional functions, absolute function removal and inefficient flows recognized.

**NEW PHYSICAL:**

The new physical represents only the physical implementation of the new system.

**RULES GOVERNING THE DFD’S**

**PROCESS**

1. No process can have only outputs.
2. No process can have only inputs. If an object has only inputs than it must be a sink.
3. A process has a verb phrase label.

**DATA STORE**

1. Data cannot move directly from one data store to another data store, a process must move data.
2. Data cannot move directly from an outside source to a data store, a process, which receives, must move data from the source and place the data into data store
3. A data store has a noun phrase label.

**SOURCE OR SINK**

The origin and / or destination of data.

1. Data cannot move direly from a source to sink it must be moved by a process
2. A source and /or sink has a noun phrase land

**DATA FLOW**

1. A Data Flow has only one direction of flow between symbols. It may flow in both directions between a process and a data store to show a read before an update. The later is usually indicated however by two separate arrows since these happen at different type.
2. A join in DFD means that exactly the same data comes from any of two or more different processes data store or sink to a common location.
3. A data flow cannot go directly back to the same process it leads. There must be atleast one other process that handles the data flow produce some other data flow returns the original data into the beginning process.
4. A Data flow to a data store means update (delete or change).
5. A data Flow from a data store means retrieve or use.

A data flow has a noun phrase label more than one data flow noun phrase can appear on a single arrow as long as all of the flows on the same arrow move together as one package.

**5.5. UML DIAGRAMS**

The Unified Modeling Language (UML) is used to specify, visualize, modify, construct and document the artifacts of an object-oriented software intensive system under development. UML offers a standard way to visualize a system's architectural blueprints, including elements such as:

* actors
* business processes
* (logical) components
* activities
* programming language statements
* database schemas, and
* Reusable software components.

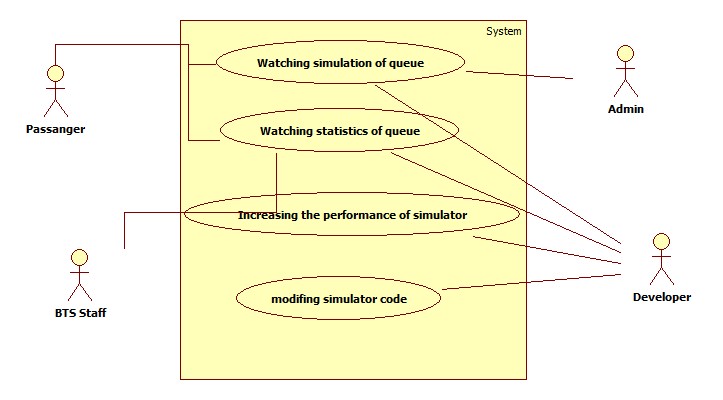
UML combines best techniques from data modeling (entity relationship diagrams), business modeling (work flows), object modeling, and component modeling. It can be used with all processes, throughout the software development life cycle, and across different implementation technologies. UML has synthesized 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.

**Use Case Diagram:**

**Over View Use Case Diagram:**

**Server:**

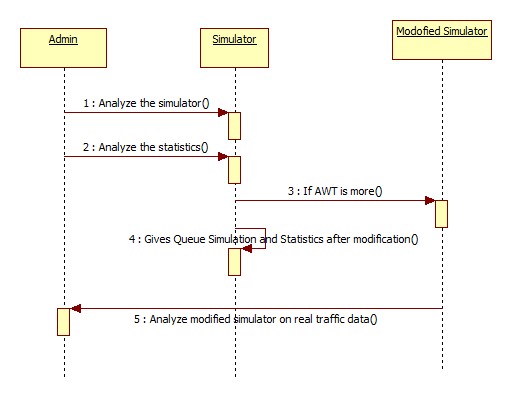
Use Case Diagram:



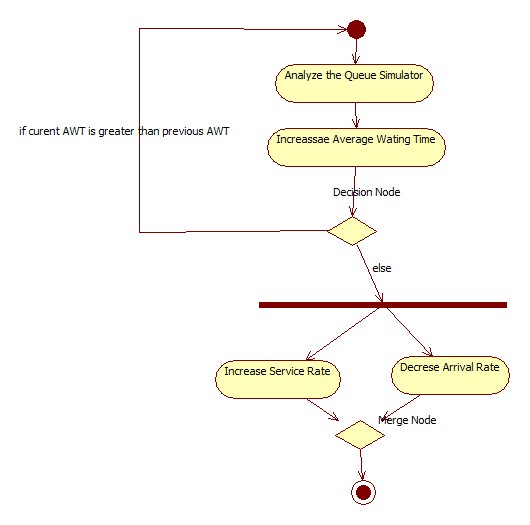
Sequence diagram:

Sequence Diagrams Represent the objects participating the interaction horizontally and time vertically. A Use Case is a kind of behavioral classifier that represents a declaration of an offered behavior. Each use case specifies some behavior, possibly including variants that the subject can perform in collaboration with one or more actors. Use cases define the offered behavior of the subject without reference to its internal structure. These behaviors, involving interactions between the actor and the subject, may result in changes to the state of the subject and communications with its environment. A use case can include possible variations of its basic behavior, including exceptional behavior and error handling.

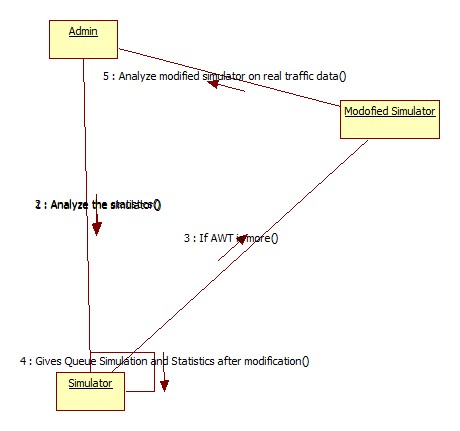
Sequence:



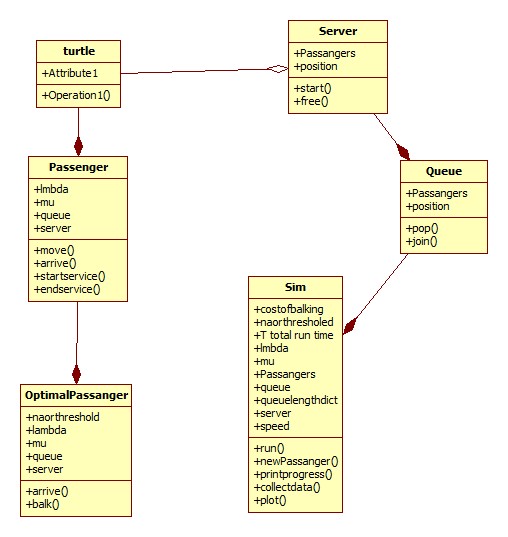
**Activity diagram:**



**Collaboration Diagram:**

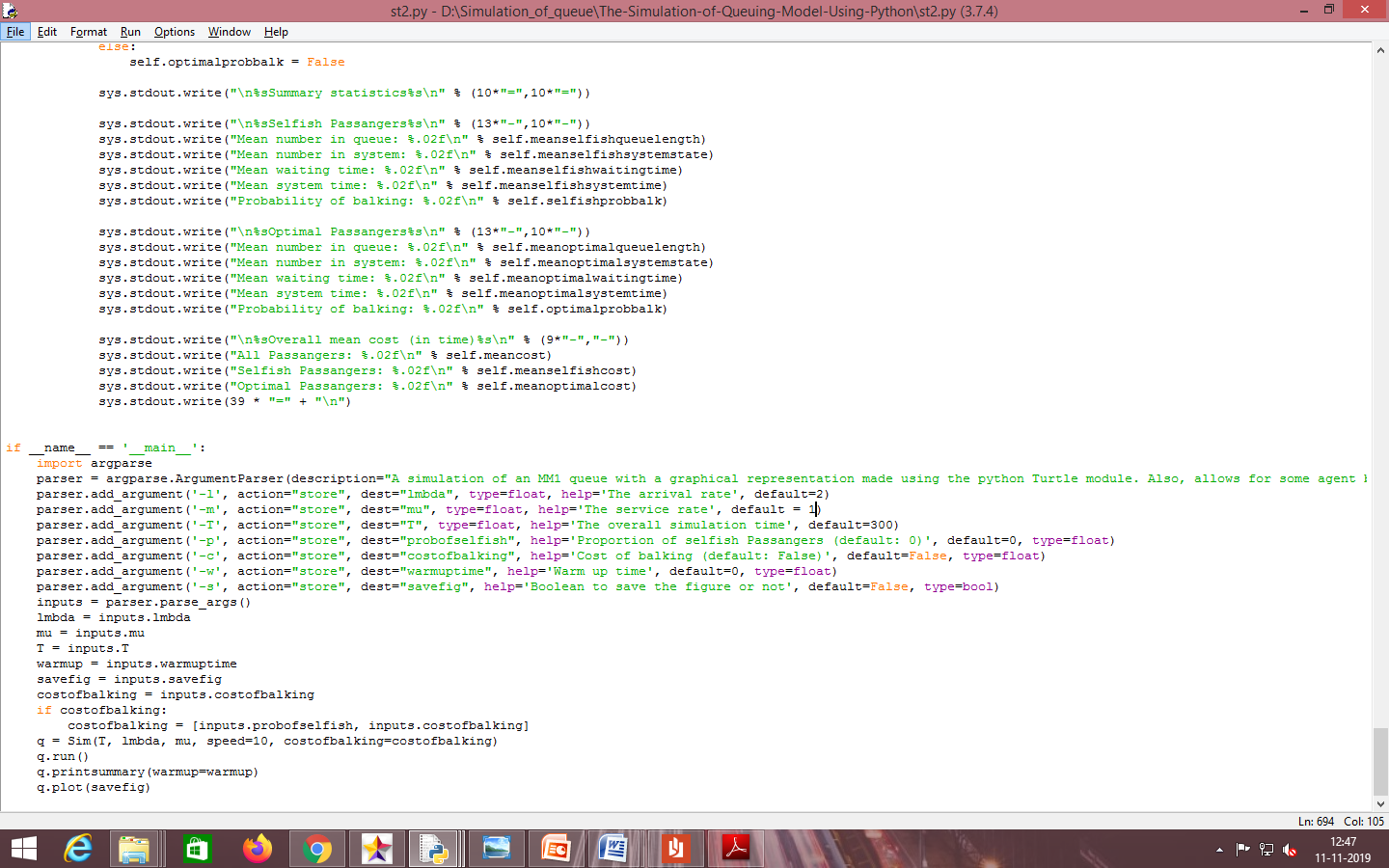


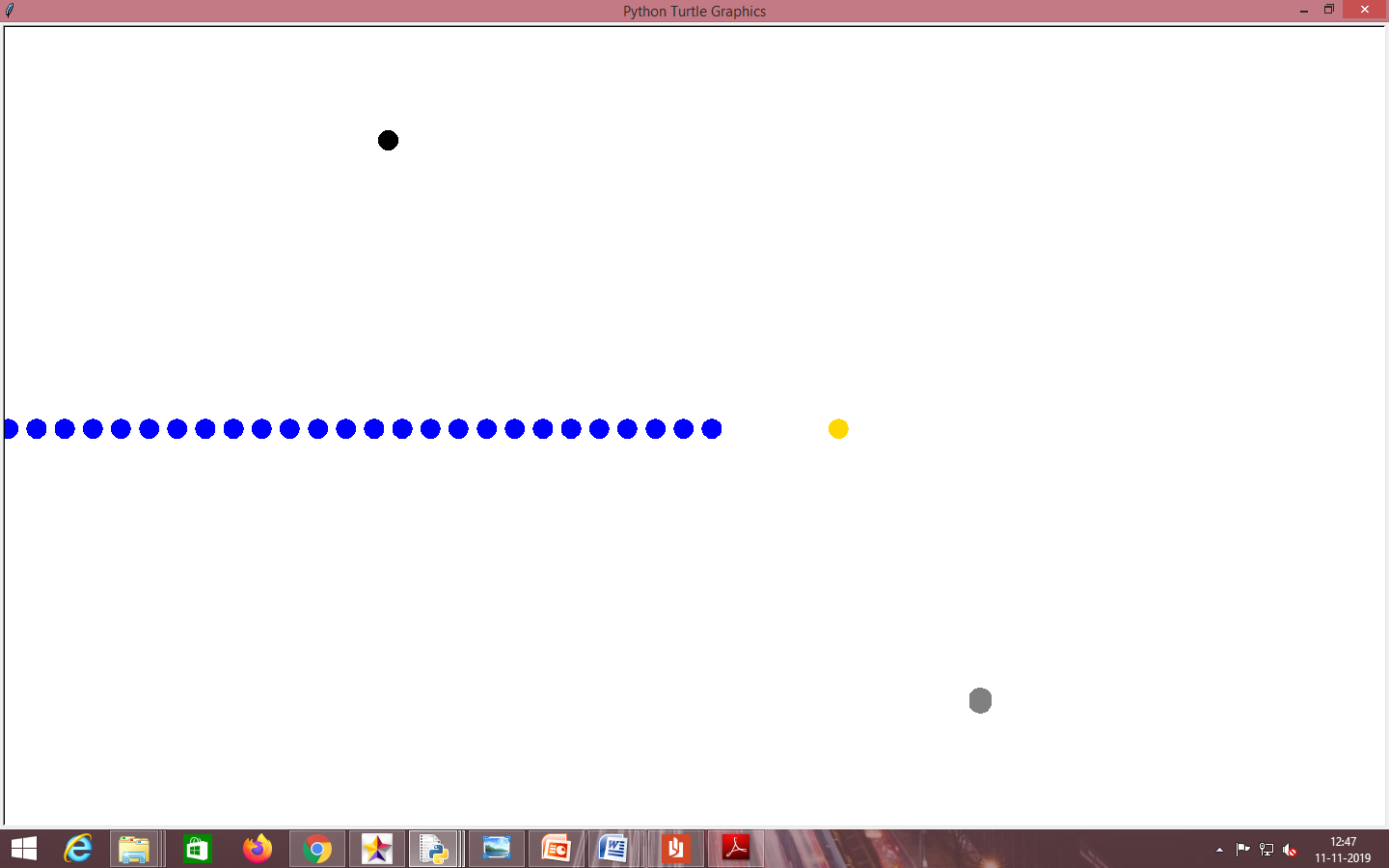
**Class Diagram:**



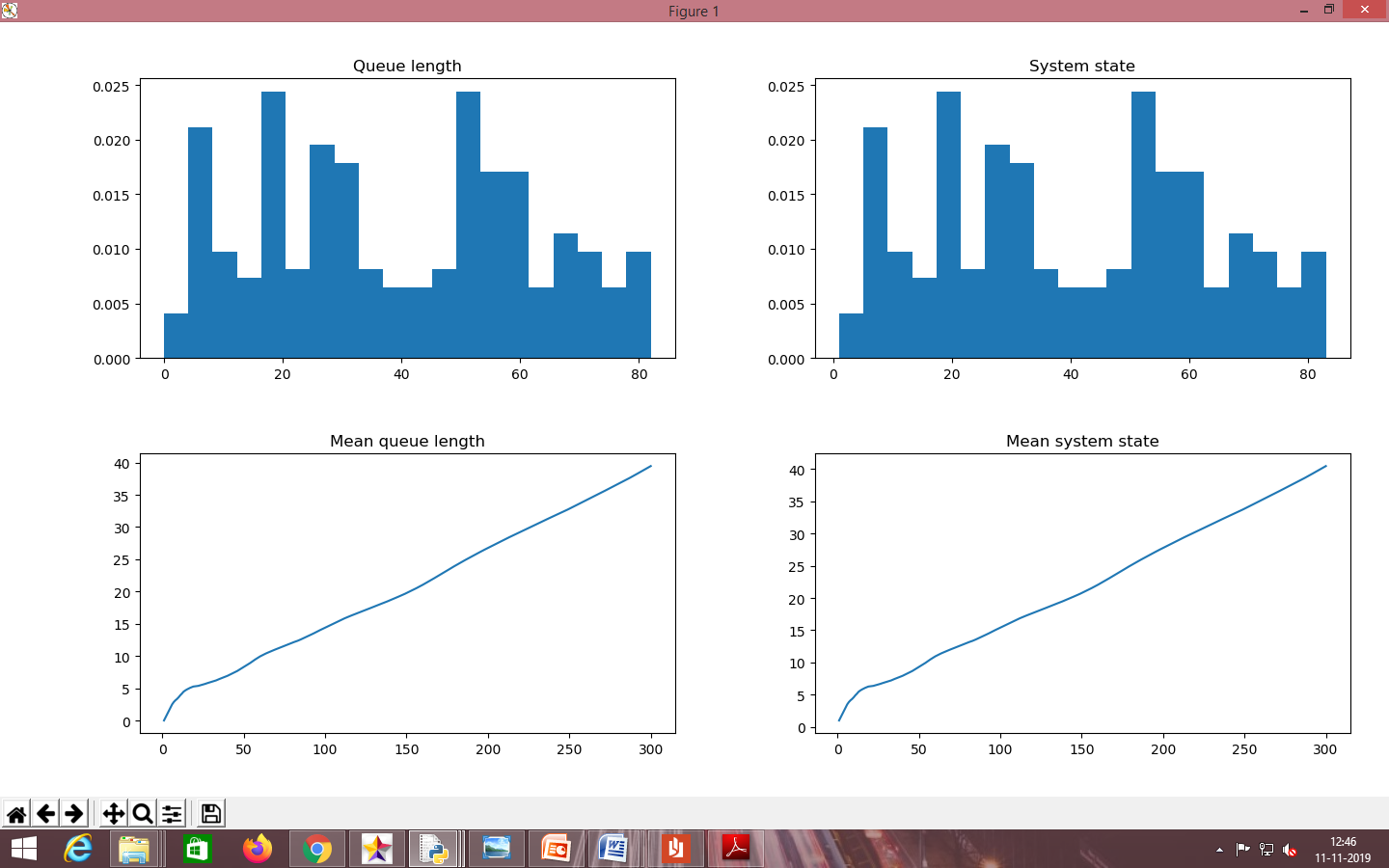
**The Simulation of Queuing Model for Bangkok Rapid Transit Train Ticket System Using Python**

**Existing System:**

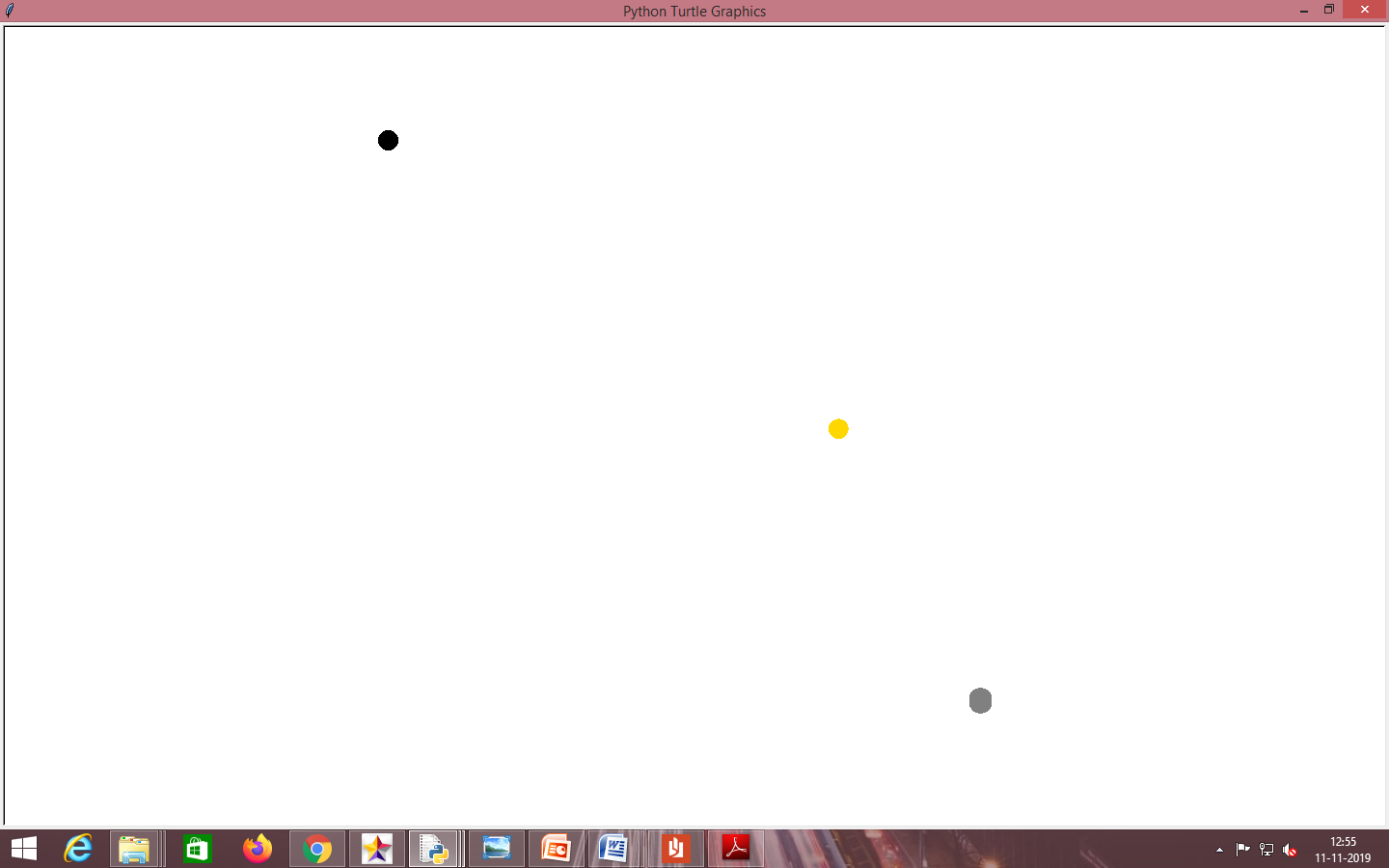


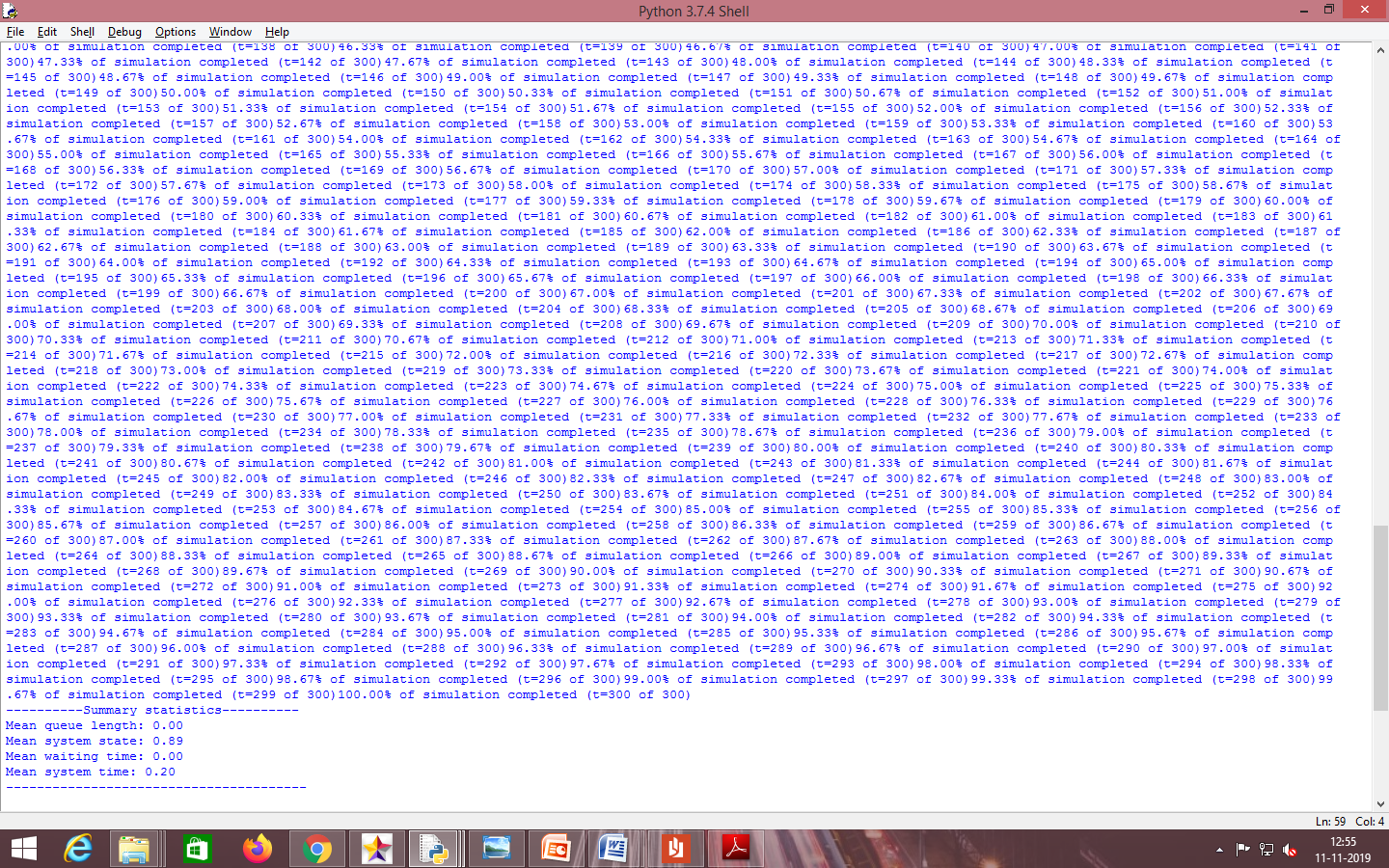


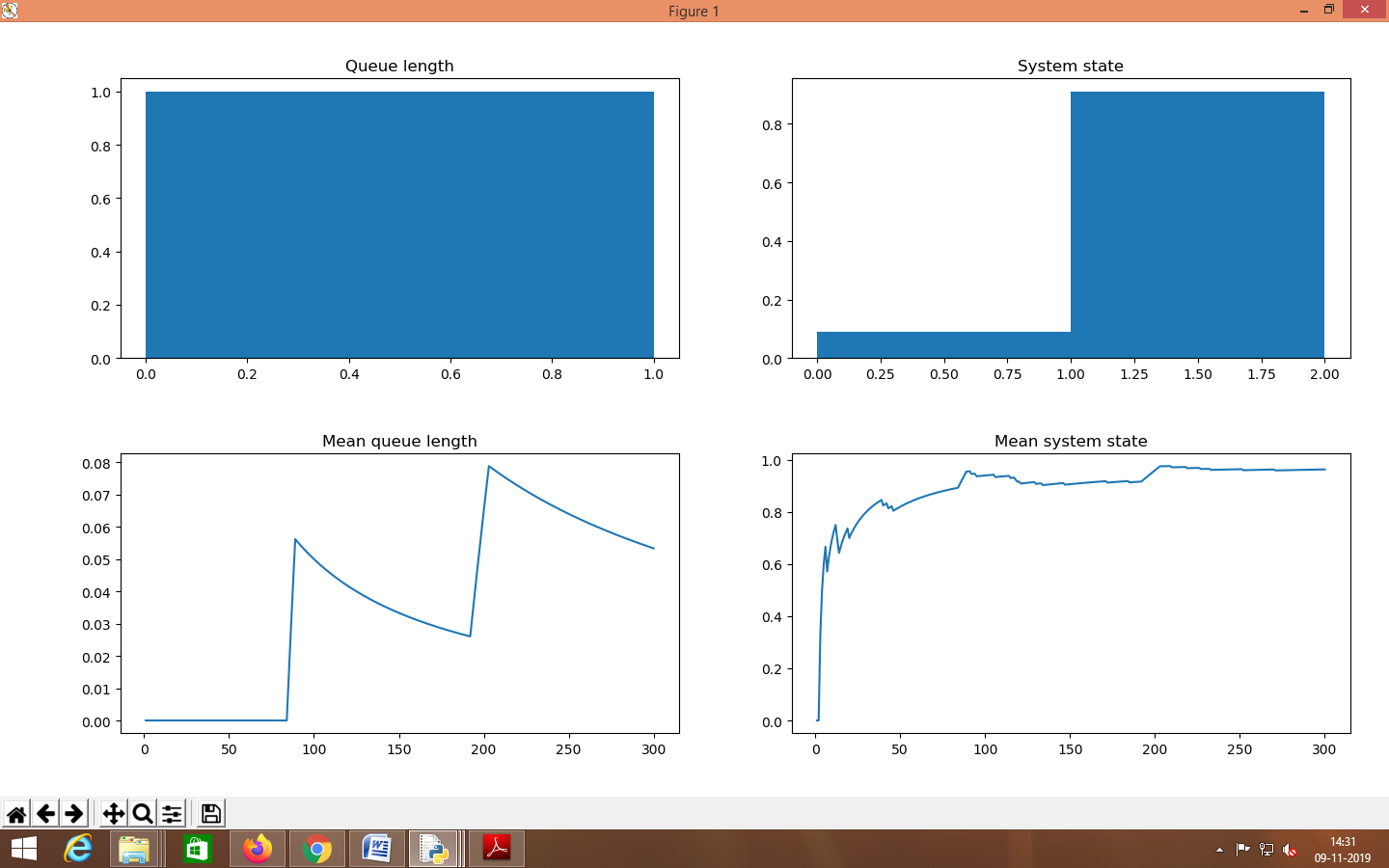




**Proposed System :**

****

****

****

**7.1 INTRODUCTION TO TESTING**

Software testing is a critical element of software quality assurance and represents the ultimate review of specification, design and coding. The increasing visibility of software as a system element and attendant costs associated with a software failure are motivating factors for we planned, through testing. Testing is the process of executing a program with the intent of finding an error. The design of tests for software and other engineered products can be as challenging as the initial design of the product itself.

There of basically **two types of testing** approaches.

One is ***Black-Box testing*** *–* the specified function that a product has been designed to perform, tests can be conducted that demonstrate each function is fully operated.

The other is ***White-Box testing*** *–* knowing the internal workings of the product ,tests can be conducted to ensure that the internal operation of the product performs according to specifications and all internal components have been adequately exercised.

White box and Black box testing methods have been used to test this package. The entire loop constructs have been tested for their boundary and intermediate conditions. The test data was designed with a view to check for all the conditions and logical decisions. Error handling has been taken care of by the use of exception handlers.

**7.2 TESTING STRATEGIES**:

Testing is a set of activities that can be planned in advanced and conducted systematically. A strategy for software testing must accommodation low-level tests that are necessary to verify that a small source code segment has been correctly implemented as well as high-level tests that validate major system functions against customer requirements.

Software testing is one element of verification and validation. Verification refers to the set of activities that ensure that software correctly implements as specific function. Validation refers to a different set of activities that ensure that the software that has been built is traceable to customer requirements.

The main objective of software is testing to uncover errors. To fulfill this objective, a series of test steps unit, integration, validation and system tests are planned and executed. Each test step is accomplished through a series of systematic test technique that assist in the design of test cases. With each testing step, the level of abstraction with which software is considered is broadened.

Testing is the only way to assure the quality of software and it is an umbrella activity rather than a separate phase. This is an activity to be preformed in parallel with the software effort and one that consists of its own phases of analysis, design, implementation, execution and maintenance.

UNIT TESTING:

This testing method considers a module as single unit and checks the unit at interfaces and communicates with other modules rather than getting into details at statement level. Here the module will be treated as a black box, which will take some input and generate output. Outputs for a given set of input combination are pre-calculated and are generated by the module.

SYSTEM TESTING:

Here all the pre tested individual modules will be assembled to create the larger system and tests are carried out at system level to make sure that all modules are working in synchronous with each other. This testing methodology helps in making sure that all modules which are running perfectly when checked individually are also running in cohesion with other modules. For this testing we create test cases to check all modules once and then generated test combinations of test paths through out the system to make sure that no path is making its way into chaos.

INTEGRATED TESTING

Testing is a major quality control measure employed during software development. Its basic function is to detect errors. Sub functions when combined may not produce than it is desired. Global data structures can represent the problems. Integrated testing is a systematic technique for constructing the program structure while conducting the tests. To uncover errors that are associated with interfacing the objective is to make unit test modules and built a program structure that has been detected by design. In a non - incremental integration all the modules are combined in advance and the program is tested as a whole. Here errors will appear in an end less loop function. In incremental testing the program is constructed and tested in small segments where the errors are isolated and corrected.

Different incremental integration strategies are top – down integration, bottom – up integration, regression testing.

TOP-DOWN INTEGRATION TEST

Modules are integrated by moving downwards through the control hierarchy beginning with main program. The subordinate modules are incorporated into structure in either a breadth first manner or depth first manner. This process is done in five steps:

* Main control module is used as a test driver and steps are substituted or all modules directly to main program.
* Depending on the integration approach selected subordinate is replaced at a time with actual modules.
* Tests are conducted.
* On completion of each set of tests another stub is replaced with the real module
* Regression testing may be conducted to ensure trha5t new errors have not been introduced.

This process continuous from step 2 until entire program structure is reached. In top down integration strategy decision making occurs at upper levels in the hierarchy and is encountered first. If major control problems do exists early recognitions is essential.

If depth first integration is selected a complete function of the software may be implemented and demonstrated.

Some problems occur when processing at low levels in hierarchy is required to adequately test upper level steps to replace low-level modules at the beginning of the top down testing. So no data flows upward in the program structure.

BOTTOM-UP INTEGRATION TEST

Begins construction and testing with atomic modules. As modules are integrated from the bottom up, processing requirement for modules subordinate to a given level is always available and need for stubs is eliminated. The following steps implements this strategy.

* Low-level modules are combined in to clusters that perform a specific software sub function.
* A driver is written to coordinate test case input and output.
* Cluster is tested.
* Drivers are removed and moving upward in program structure combines clusters.

Integration moves upward, the need for separate test driver’s lesions.

If the top levels of program structures are integrated top down, the number of drivers can be reduced substantially and integration of clusters is greatly simplified.

REGRESSION TESTING

Each time a new module is added as a part of integration as the software changes. Regression testing is an actually that helps to ensure changes that do not introduce unintended behavior as additional errors.

Regression testing maybe conducted manually by executing a subset of all test cases or using automated capture play back tools enables the software engineer to capture the test case and results for subsequent playback and compression. The regression suit contains different classes of test cases.

A representative sample to tests that will exercise all software functions.

Additional tests that focus on software functions that are likely to be affected by the change.

**7.3 IMPLEMENTATION**

Implementation is the process of converting a new or revised system design into operational one. There are three types of Implementation:

* Implementation of a computer system to replace a manual system. The problems encountered are converting files, training users, and verifying printouts for integrity.
* Implementation of a new computer system to replace an existing one. This is usually a difficult conversion. If not properly planned there can be many problems.
* Implementation of a modified application to replace an existing one using the same computer. This type of conversion is relatively easy to handle, provided there are no major changes in the files.

Implementation in Generic tool project is done in all modules. In the first module User level identification is done. In this module every user is identified whether they are genuine one or not to access the database and also generates the session for the user. Illegal use of any form is strictly avoided.

In the Table creation module, the tables are created with user specified fields and user can create many table at a time. They may specify conditions, constraints and calculations in creation of tables. The Generic code maintain the user requirements through out the project.

In Updating module user can update or delete or Insert the new record into the database. This is very important module in Generic code project. User has to specify the filed value in the form then the Generic tool automatically gives whole filed values for that particular record.

In Reporting module user can get the reports from the database in 2Dimentional or 3Dimensional view. User has to select the table and specify the condition then the report will be generated for the user.

# 9.1. INTRODUCTION

The protection of computer based resources that includes 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.

## 9.2. SECURITY IN SOFTWARE

System security refers to various validations on data in form of checks and controls to avoid the system from failing. It is always important to ensure that only valid data is entered and only valid operations are performed on the system. The system employees two types of checks and controls:

**CLIENT SIDE VALIDATION**

Various client side validations are used to ensure on the client side that only valid data is entered. Client side validation saves server time and load to handle invalid data. Some checks imposed are:

* JavaScript in used to ensure those required fields are filled with suitable data only. Maximum lengths of the fields of the forms are appropriately defined.
* Forms cannot be submitted without filling up the mandatory data so that manual mistakes of submitting empty fields that are mandatory can be sorted out at the client side to save the server time and load.
* Tab-indexes are set according to the need and taking into account the ease of user while working with the system.

**SERVER SIDE VALIDATION**

Some checks cannot be applied at client side. Server side checks are necessary to save the system from failing and intimating the user that some invalid operation has been performed or the performed operation is restricted. Some of the server side checks imposed is:

* Server side constraint has been imposed to check for the validity of primary key and foreign key. A primary key value cannot be duplicated. Any attempt to duplicate the primary value results into a message intimating the user about those values through the forms using foreign key can be updated only of the existing foreign key values.
* User is intimating through appropriate messages about the successful operations or exceptions occurring at server side.
* Various Access Control Mechanisms have been built so that one user may not agitate upon another. Access permissions to various types of users are controlled according to the organizational structure. Only permitted users can log on to the system and can have access according to their category. User- name, passwords and permissions are controlled o the server side.
* Using server side validation, constraints on several restricted operations are imposed.

In this paper, proposed analysis model can be used to evaluate the service time of railway companies, which enables for the operators to form a proper policy and decide on different ticket system operations based on cost requirements. The proposed simulation model has been implemented in python software tool for analyzing the effect time delay of queues or average waiting time for passengers by comparing ticket existing system and desire modification system. Since, the proposed simulation of queuing model presents the results to show the impact of time delay in queue of passengers when ticket machines are modified by number or feature upgrading. The proposed analysis model could potentially be used to any rapid transit train ticket system for impact evaluation on the existing ticket system.

Proposed future system will include artificial intelligence to classify between passenger who are in hurry or not. Those who will be hurry will get the priority for the service

**For software installation:**

<https://www.anaconda.com/download/>

<https://www.python.org/downloads/release/python-360/>

**Modules:**

* Install numpy
* Install pandas
* Install matplotlib
* Install Turtle

**References:**

[1] S. Poomrittigul, P. Tuwanut, L.Wuttisittikulkij,"Queuing Theory Analysis for Bangkok Rapid Transit Train Ticket System" The 32st International Technical Conference on Circuits/Systems, Computers and Communications (ITC-CSCC2017), pp.621-623, July 2017.

[2] Y. Jiang, and X. Lin, “Simulation and Optimization of the Ticket Vending Machine Configuration in Metro Stations Based on Anylogic Software”, ICTE 2013. 2013. p.754-760.

[3] A. Khattak, Y. Jiang, H. Lu, J. Zhu, “Modeling and Simulation of Metro Transit Station Walkway as a State-dependent Queuing System based on the Phase-Type Distribution”, International Journal of Traffic and Transportation Engineering, Vol. 5 No. 5, 2016, pp. 103-111.

[4] M. Zukerman. “Introduction to Queueing Theory and Stochastic Teletraffic Models”, City University of Hongkong, 2012.