

DESIGN PROJECT 3 Final Raport

Prepared by: Israel Mbombole

219120498

Lecturer: Mr V Moyo

**Airline Reservation System**

May 2021

Table of Contents

[GLOSARRY 3](#_Toc64481390)

[INTRODUCTION 4](#_Toc64481391)

[1. PROJECTS IDEAS 4](#_Toc64481392)

[1.1. AIRLINE TICKET RESERATION SYSTEM 4](#_Toc64481393)

[1.2. Operational feasibility 8](#_Toc64481394)

[1.3. Technical feasibility 8](#_Toc64481395)

[1.4. Economic feasibility 8](#_Toc64481396)

[1.5. Schedule feasibility 8](#_Toc64481397)

[2. OBSTACLE AVOIDING CAR ARDUINO 9](#_Toc64481398)

[2.1. Operation feasibility 9](#_Toc64481399)

[2.2. Technical feasibility 9](#_Toc64481400)

[2.3. Economic feasibility 10](#_Toc64481401)

[2.4. Schedule feasibility 10](#_Toc64481403)

[3. NETWORK PACKET SNIFFER 10](#_Toc64481404)

[3.1. Operational feasibility 11](#_Toc64481405)

[3.2. Technical feasibility 11](#_Toc64481406)

[3.3. Economic feasibility 12](#_Toc64481407)

[3.4. Schedule feasibility 12](#_Toc64481408)

[3.4.1. Airline ticket Reservation System 12](#_Toc64481409)

[3.4.2. Obstacle avoiding car Arduino 13](#_Toc64481410)

[3.4.3. Network Packet Sniffer 13](#_Toc64481411)

[4. Motivation 13](#_Toc64481412)

[5. Overview of airline ticket reservation System 14](#_Toc64481413)

[6. Gant Chart 14](#_Toc64481414)

[7. References 15](#_Toc64481415)

# Table of Figures

[Figure 1Login 6](#_Toc64481498)

[Figure 2user Creation 6](#_Toc64481499)

[Figure 3Add Customer 7](#_Toc64481500)

[Figure 4Search Customer 7](#_Toc64481501)

[Figure 5Add flight 8](#_Toc64481502)

[Figure 6Book ticket 8](#_Toc64481503)

[Figure 7 Obstacle Avoiding car 10](#_Toc64481504)

[Figure 8Network Packet Sniffer 12](#_Toc64481505)

[Figure 9Grantt chart 15](#_Toc64481506)

# List of Tables

[Table 1Airline ticket reservation System equipment 10](#_Toc64481815)

[Table 2Obstacle avoiding car Arduino component 12](#_Toc64481816)

[Table 3Network Packet Sniffer equipment 14](#_Toc64481817)

[Table 4 pros and cos of an airline ticket reservation System 14](#_Toc64481818)

[Table 5 pros and cos of Obstacle avoiding car Arduino 15](#_Toc64481819)

[Table 6 pros and cos of Network Packet Sniffer 15](#_Toc64481820)

# GLOSARRY

* **Database:** A database is an organized collection of information, or organized data, that's recorded electronically in a computer framework. A database makes a difference the information to be effortlessly overseen, gotten to, altered, overhauled, controlled, and organized. [1]
* **Packet:** A packet may be a little fragment of a bigger message. Information sent over computer systems, such as the web is partitioned into bundles and these bundles are recombined by the computer or gadget that gets them. [2]
* **Microcontroller:** A microcontroller may be a compact coordinates circuit planned to administer a particular operation in an implanted framework. A microcontroller incorporates input/output peripherals, memory, and a processor on a single chip. [3]
* **Arduino:** Arduino is an open-source stage utilized for building hardware venture. It comprises of a physical programmable circuit board and a chunk of program, or coordinates improvement environment that runs on a computer utilized to type in and transfer code to the physical board. [4]
* **Ultrasonic sensor HC-SR04:** The HC-SR04 ultrasonic sensor employments SONAR to decide the separate of an protest a bit like the bats do. It offers great non-contact run location with tall exactness and steady readings in an easy-to-use bundle from 2 cm to 400 cm. [5]

## INTRODUCTION

I am required as an understudy in computer framework understudy in electrical designing to plan a extend based on all the information that I have gotten since the starting of my studies. As we will see in our world innovation gets to be a fundamentals element in numerous things in our life and we got to keep on track the current and future improvements of the technology.

Technology in cars is creating exceptionally fast and to nowadays a car can be plan for any I have been inquisitive about microcontroller since able to do a parcel of assignments with them depending on how we program them, I am moreover fascinated by electronic and coding creating application. I did me inquire about on 3 ventures that requires programming aptitudes such as C, C++ and Java. One will be managing with microcontrollers and the others with JAVA.

## PROJECTS IDEAS

# AIRLINE TICKET RESERATION SYSTEM

This project aims to provide a platform of booking and making reservation of a flight ticket

And customer details

Airline reservation systems incorporate airline schedules, [fare tariffs](https://en.wikipedia.org/wiki/Fare_basis_code), passenger reservations and ticket records. An airline's direct distribution works within their own reservation system, as well as pushing out information to the GDS. The second type of direct distribution channel are consumers who use the internet or mobile applications to make their own reservations. Travel agencies and other indirect distribution channels access the same GDS as those accessed by the airline reservation systems, [8]

Therefor with this project we will try to overcome these issues by providing perfect connection between Airline company and customers via an accurate platform. [9]

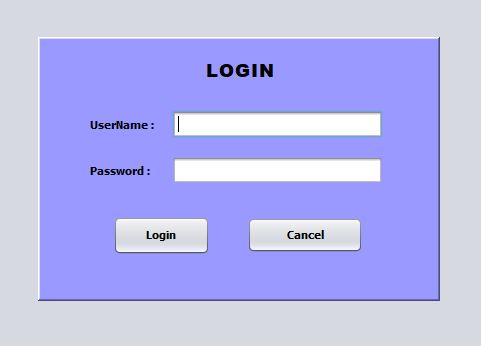


Figure 1Login

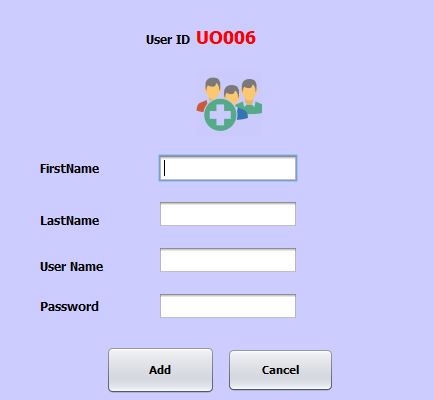


Figure 2user Creation

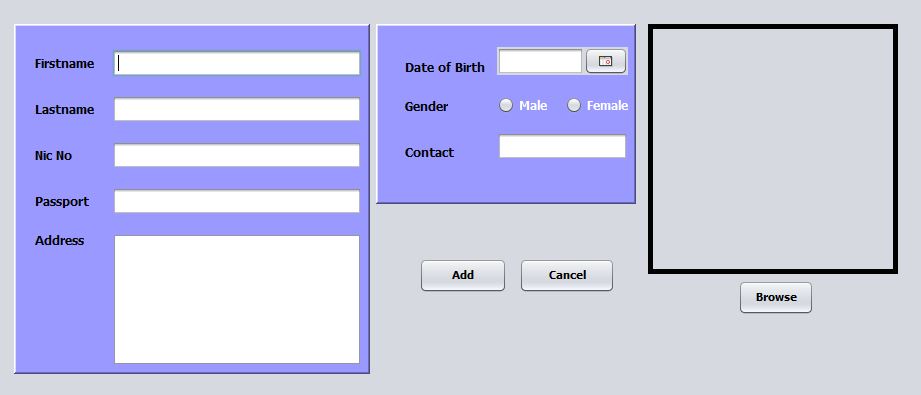


Figure 3Add Customer

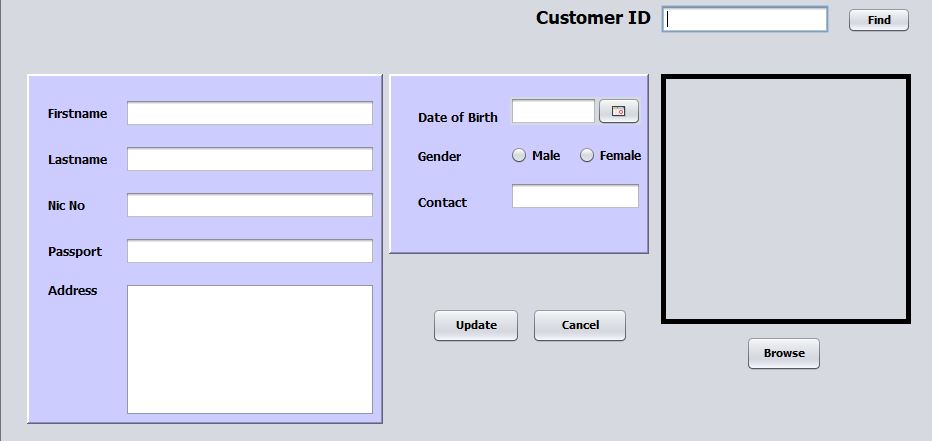


Figure 4Search Customer

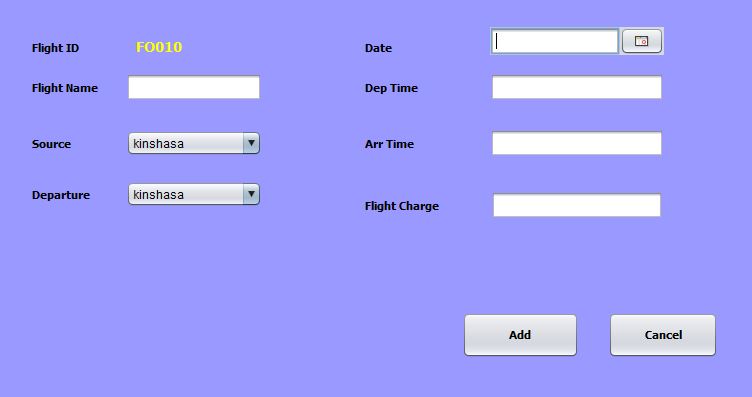


Figure 5Add flight

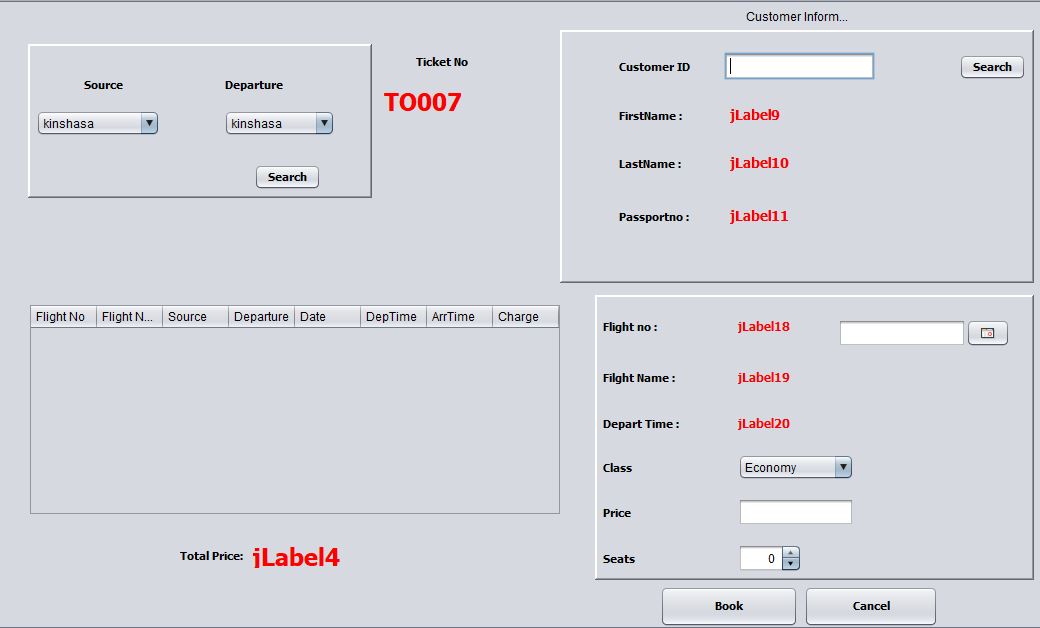


Figure 6Book ticket

# Operational feasibility

This project will ease the customer to make their reservations and book their flight on time with an airline company of their choice, and the airline company can then keep truck of the flight tickets, flight suggestions, and airline manager can view the customer’s records, flights, updating them and also the users or customer can have a look to find and connect with their flight.

# Technical feasibility

This project will require the following:

|  |  |
| --- | --- |
| N | Equipment |
| 1 | NetBeans 8.1 Software |
| 2 | MySQL Database |
| 3 | Airline Customer manager |
| 4 | Internet connection |
| 5 | Browser installed (Firefox, IE) |

Table 1Airline ticket reservation System equipment

# Economic feasibility

This project will not require development cost, it will increase the number of patients who will appreciate this well-organized structure and will help the airline companies in working fast in finding someone ticket reports. Also, in some condition where the person cannot move easily it’s going to be simple to contact the flight center and make their reservations.

# Schedule feasibility

This project will require an active presence in airline company structure to understand how it does work, to know all the airline terms and how everything is managed, which is easy to find on internet.

By considering the amount of time given to us, it will be easy to finish this project on time

# OBSTACLE AVOIDING CAR ARDUINO

Many new cars can be driving by themselves everywhere you want it to go and park itself in parking, one of the main parts of this technology is done by the sensor which detect all obstacle and avoid them easily. [5]

By connecting a motor driver L298N to an Arduino [4] board mounted on a 4WD car, this design will allow the car to travel in unknown environment by the instructions given to the motor driver and the sensor. The ultrasonic sensor [6] will help to avoid obstacles by sensing them in front of it and turning the car in another direction.

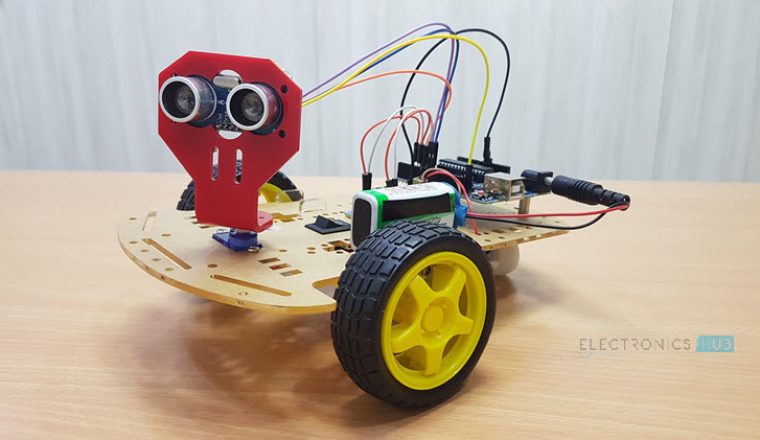


Figure 7 Obstacle Avoiding car

# Operation feasibility

the concept of an avoiding obstacle [5] car is achieving with an Arduino, a driver motor, and a sensor. We will understand also how the sensor works and how to write C++ basics program that will be apply to give instructions to the car.

# Technical feasibility

The component required in the project are the following

|  |  |
| --- | --- |
| No | **Components** |
| 1 | ArduinoUNO |
| 2 | HC-SR04 Ultrasonic Sensor |
| 3 | LM298N Motor Driver Module |
| 4 | 5V DC Motors |
| 5 | Battery |
| 6 | Wheels |
| 7 | Chassis |
| 8 | Breadboard and Jumper wires |

Table 2Obstacle avoiding car Arduino component

# Economic feasibility

### This project does not require numerous human powers to complete it, one individual can do it without issue. This extend will require to purchase a few components, patching wires, and mounting the car chassis, in any case the engine can be made from scratch or able to purchase it this will offer assistance the economy of a shop.

# Schedule feasibility

This project does require much time to complete it, but waiting for components to arrive and learning a bit of C++ programming can take a bit longer. And require more research in order to be completed on time, which will make it difficult to be completed on the scheduled time.

# NETWORK PACKET SNIFFER

In a word where numeric and digital are taking much place over the year, such as working, study make payment from home network become a helpful system that is used to transmit information from a point A to point B, this project will be used to monitor network traffic.

Network packet sniffer [7] is a web-based Java application uses to facilitates the monitoring of network packets [2] that travel across a system network

With this project companies and people will feel more secure with the rules that can be establish to prevent hackers from attacking the system software with malware and viruses.

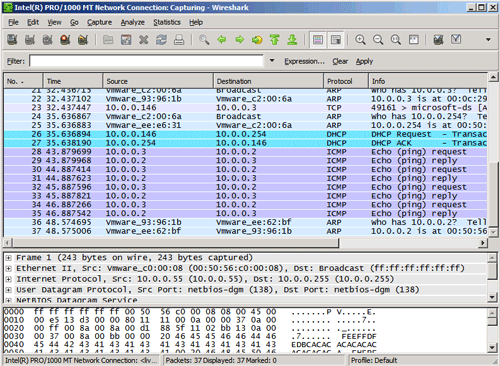


Figure 8Network Packet Sniffer

# Operational feasibility

In some environment, it is very difficult to work on internet freely and feel secure because hackers who can attack the system and steal your personal information.

With this project we will prevent the system from being infected with viruses and malware from hackers, it will help customers and compagnies to work in secure environment without worrying about an outside intrusion their work system.

# Technical feasibility

To accomplish this project the following are required:

|  |  |
| --- | --- |
| N | Equipment |
| 1 | JCreator Software |
| 2 | Networking Protocol |
| 3 | Microsoft Excel |
| 4 | Database |

Table 3Network Packet Sniffer equipment

# Economic feasibility

As we can see in the description of the project, I will need a network technician and network topology to accomplish this project, the project can be risky because we will have to change a security that is already installed in a company to try this one, therefor the benefits of this project are not guarantee.

# Schedule feasibility

To fully accomplish this project, we have to understand perfectly a network and know how the network is configure and working, to understand that we have to have a network and technician, which could be time consuming with the time we have to complete this project

To decide project, I am going to work on I have to consider the best advantages and disadvantages out of the three projects:

## Airline ticket Reservation System

|  |  |
| --- | --- |
| **Advantages** | **Disadvantages** |
| No development cost | Longer to complete |
| Useful | Require a good amount of time development |
| Save appointment time | Need a good understanding of an aviation reservation system |
| Easy to build |  |
| Easy to use |  |
| Secured |  |
| Opportunity to learn new concepts |  |
| Quick to understand |  |

Table 4 pros and cos of an airline ticket reservation System

## Obstacle avoiding car Arduino

|  |  |
| --- | --- |
| **Advantages** | **Disadvantages** |
| Easy to build | Component can be damaged |
| Cheaper to build and maintain | Learning basics of C++ |
| All component can be found | Waiting for component to arrive |
|  | More time required to complete |
|  | Need a good understanding of Arduino |
|  | Lot of the libraries of Arduino are needs to address issues like Ethernet, ZigBee. |
|  | it’s not optimized for performance |
|  | limited to a small number of MCUs |

Table 5 pros and cos of Obstacle avoiding car Arduino

## Network Packet Sniffer

|  |  |
| --- | --- |
| **Advantages** | **Disadvantages** |
| No development cost | Require internet connection |
| Useful | Require amount of time development |
| Will help companies against hackers | Need a good understanding of a network company |
|  | Require a network technician |
|  | Complex with Java |
|  | Longer to complete |

Table 6 pros and cos of Network Packet Sniffer

# Motivation

I have decided to choose the **airline ticket reservation System** project because it a project clearer than all other and can surely be completed in the amount of time given to us, and some component require for the project can be found locally and other can be downloaded. I also choose this project because I am enjoying working with NetBeans and JAVA and learning new concept such as database concepts

The schedule of this project will not be long than other and can be completed on time while the other project requires more time and need some research, this project has a financial benefits and the same concept can be applying on our daily life and improve maneuvers of forgetting patient details.

# Overview of airline ticket reservation System

The airline reservation system is a platform of booking solution that helps in consolidating data from all airlines through the use of a global distribution system. The system provides inventory and rates in real-time to customers as well as travel agents. Lately airline reservation system is being not only to make flight reservations but also being used to assist with different airline management tasks and to meet from the initial booking all the way to complete the flight. Airline reservations system (ARS) is online application software used to retain, retrieve information and perform transactions related to air travel.

# Gant Chart

Figure 9Grantt chart

# 7.1 DFD for Airline Reservation System

**First Level of Data Flow Diagram for**

**System Login**

Password Verification

User Name

Log In

**SYSTEM**

**ADMINSTRATOR**

Verification

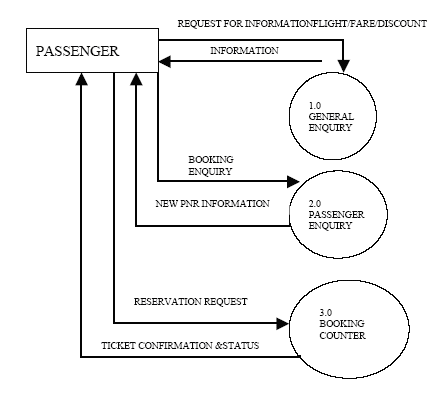
Fail

Verification

Success

**Second Level of Data Flow Diagram for**

**General Inquiry System**

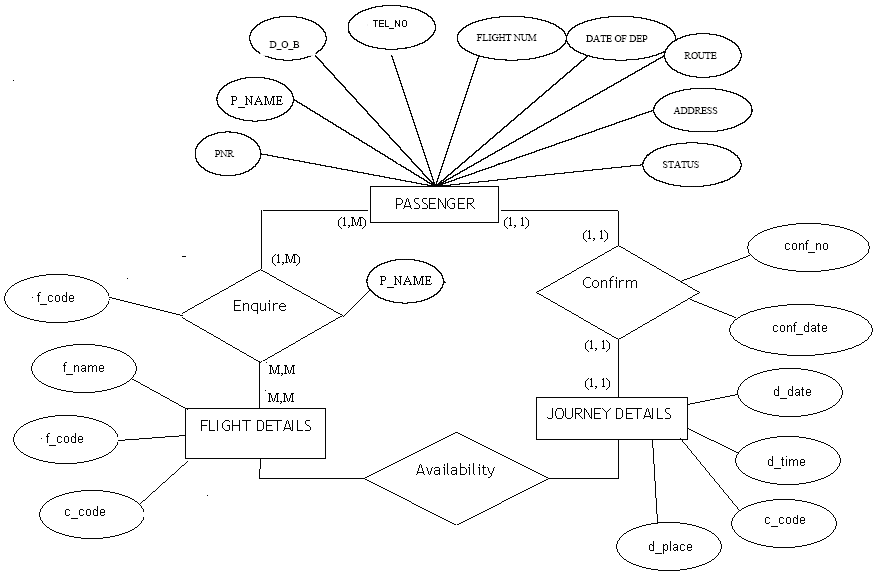
****

**Third Level DATA FLOW DIAGRAM**

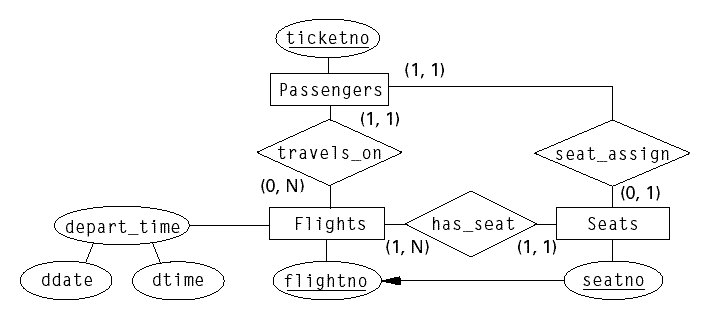
**OF BOOKING SECTION**



**E-R Diagram**

****

E-R Diagram for Airline Reservation System**ER-DIAGRAM FOR PASSENGER**

****

**EVENT TRACE DIAGRAM**

New Airline Registration Reservation Flight

Customer Reservation

system

Enters Get Response Cust\_id Login request for

Reservation Interacts Response Gives request response Request operation Interacts

Gives request

Response Response Flight Interacts Gives response response

**EVENT FLOW DIAGRAM**  **Login** **Performs** **Gives Response**

**Airline Reservation system**

**Customer**

**Operations**

# 7.2 STATE DIAGRAM

A state diagram relates events and states. When an event is received , the next state depends on the current state as well as the event: A change of state caused by an event is called a transition. A state diagram is a graph whose nodes are states and whose directed arcs are transitions labeled by event names. A state is drawn as a rounded box containing an optional name. A transition is drawn as an arrow from the receiving state to the target state: The label on the arrow is the name of the event causing the transition. All the transitions leaving a state must correspond to different events.

Figure below shows a state diagram describing the behavior of Airline Reservation System. A state diagram describes the behavior of a single class of objects. Since all the instances of a class of objects. Since all the instances of a class have the same behavior, they all share the same state diagram, as they all share the same class features.

**STATE DIAGRAM:**

**Exit**

**●Enter**

Performs Performs Performs

System

do : Enter the system

Operations

Exit

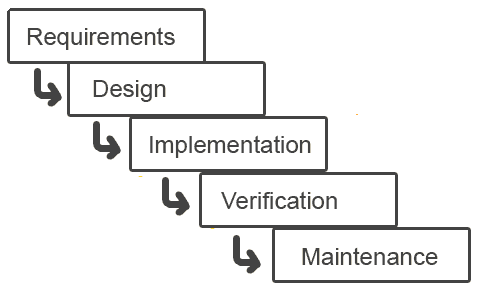
Reservation

Flight Operation

# 8. SOFTWARE ENGINEERING APPROACH

The field of software engineering is related to the development software in systematic manner unlike simple programs which can be developed in isolation and there may not be any systematic approach being followed. As there is large difference between programming and software engineering. As it provide models that lead to the production of well documented software in a manner that is predictable. For a mature process, it should be possible to determine in advance how much time and effort will be required to produce the final product. To develop successful software I have to follow some models, which act as guidelines.

The model I have used is **Waterfall Model or Classic Life Cycle**. In this model first of all the existed system is observed. Then customer requirements are taken in consideration then planning, modeling, construction and finally deployment.

****

**Fig.1. Waterfall Model**

# 9. SYSTEM DESIGN

**Introduction**

Analysis collects a great deal of unstructured data through interviews, questionnaires, on-site observations, and procedural manuals and like. It is required to organize and convert the data through system flowcharts, data flow diagrams, structured English, decision tables and the like which support future development of the system.

The Data flow diagrams and various processing logic techniques show how, where, and when data are used or changed in an information system, but these techniques do not show the definition, structure and relationships within the data.

It is a way to focus on functions rather than the physical implementation. This is analogous to the architect’s blueprint as a starting point for system design. The design is a solution, a “how to” approach, compared to analysis, a “what is” orientation.

System design is a highly creative process. This system design process is also referred as data modeling. The most common formatted used the E-R notation explains the characteristics and structure of data independent of how the data may be stored in computer memories.

The process of system design can be divided into three stages. They are:

* Structure design (already discussed)
* Database design
* Interface design

As we know that system design is a solution to “How to approach to the creation of new system”. It provides the understudying and procedural details necessary for implementing the system. The steps involved during system design were as follow: -

**LOGICAL AND PHYSICAL DESIGN**

The current physical system was thoroughly reviewed from point of view how the data flow, what are file contents, its volumes and frequency etc.

After this input, output specifications security & control specification were prepared. It was also decided that how physical information will flow through the system and a physical design walkthrough.

**OUTPUT DESIGN**

The format of outputs is designed in such a way that it is simple to read and interpret In the present output we have clearly labeled title it contains date and time and all the fields are clearly mentioned (labeled).

**INPUT DESIGN**

.Input should be as simple as possible. It is design to reduce possibility of incorrect data being enter and the need of system user are considered with this view of mind several human factor is evaluated.

**SCREEN DESIGN**

The screen design for inputting the inputs were also panned as the format of inputs.

# 9.1 DATABASE DESIGN

**Table name: customer**

|  |  |  |  |
| --- | --- | --- | --- |
| # | Name | Type | Collation |
| 1 | **id Primary** | varchar(255) | latin1\_swedish\_ci |
| 2 | **firstname** | varchar(255) | utf8mb4\_general\_ci |
| 3 | **lastname** | varchar(255) | utf8mb4\_general\_ci |
| 4 | **nic** | varchar(255) | utf8mb4\_general\_ci |
| 5 | **passport** | varchar(255) | utf8mb4\_general\_ci |
| 6 | **address** | text | utf8mb4\_general\_ci |
| 7 | **dob** | varchar(255) | utf8mb4\_general\_ci |
| 8 | **gender** | varchar(255) | utf8mb4\_general\_ci |
| 9 | **contact** | int(11) |  |
| 10 | **photo** | longblob |  |

**Table name : flight**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | | # | Name | | Type | | Collation |
| 1 | **id** | | | varchar(255) | | utf8mb4\_general\_ci | | |
| 2 | **flightname** | | | varchar(255) | | utf8mb4\_general\_ci | | |
| 3 | **source** | | | varchar(255) | | utf8mb4\_general\_ci | | |
| 4 | **depart** | | | varchar(255) | | utf8mb4\_general\_ci | | |
| 5 | **date** | | | varchar(255) | | utf8mb4\_general\_ci | | |
| 6 | **deptime** | | | varchar(255) | | utf8mb4\_general\_ci | | |
| 7 | **arrtime** | | | varchar(255) | | utf8mb4\_general\_ci | | |
| 8 | **flightcharge** | | | varchar(255) | | utf8mb4\_general\_ci | | |

**Table name: ticket**

|  |  |  |  |
| --- | --- | --- | --- |
| # | Name | Type | Collation |
| 1 | **id** | varchar(255) | utf8mb4\_general\_ci |
| 2 | **flightid** | varchar(255) | utf8mb4\_general\_ci |
| 3 | **custid** | varchar(255) | utf8mb4\_general\_ci |
| 4 | **class** | varchar(255) | utf8mb4\_general\_ci |
| 5 | **price** | int(11) |  |
| 6 | **seats** | int(11) |  |
| 7 | **date** | varchar(255) | utf8mb4\_general\_ci |

**Table name: user**

|  |  |  |  |
| --- | --- | --- | --- |
| # | Name | Type | Collation |
| 1 | **id** | varchar(255) | utf8mb4\_general\_ci |
| 2 | **firstname** | varchar(255) | utf8mb4\_general\_ci |
| 3 | **lastname** | varchar(255) | utf8mb4\_general\_ci |
| 4 | **username** | varchar(255) | utf8mb4\_general\_ci |
| 5 | **password** | varchar(255) | utf8mb4\_general\_ci |

# 9.2 TESTING

**Software Testing**

Software testing is a process of *verifying* and *validating* that a software application or program. Software testing

**1.** Meets the business and technical requirements that guided its design and development, and

**2.** Works as expected.

Software testing also identifies important *defects*, flaws, or errors in the application code that must be fixed. The modifier “important” in the previous sentence is, well, important because defects must be categorized by severity.

During test planning we decide what an important defect is by reviewing the requirements and design documents with an eye towards answering the question “Important to whom?” Generally speaking, an important defect is one that from the customer’s perspective affects the usability or functionality of the application. Using colors for a traffic lighting scheme in a desktop dashboard may be a no-brainer during requirements definition and easily implemented during development but in fact may not be entirely workable if during testing we discover that the primary business sponsor is color blind. Suddenly, it becomes an important defect. (About 8% of men and .4% of women have some form of color blindness.)

The quality assurance aspect of software development—documenting the degree to which the developers followed corporate standard processes or best practices—is not addressed in this paper because assuring quality is not a responsibility of the testing team. The testing team cannot improve quality; they can only measure it, although it can be argued that doing things like designing tests before coding begins will improve quality because the coders can then use that information while thinking about their designs and during coding and debugging.

Software testing has three main purposes: verification, validation, and defect finding.

* The *verification* process confirms that the software meets its technical specifications. A “specification” is a description of a function in terms of a measurable output value given a specific input value under specific preconditions. A simple specification may be along the line of “a SQL query retrieving data for a single account against the multi-month account-summary table must return these eight fields <list> ordered by month within 3 seconds of submission.”
* The *validation* process confirms that the software meets the business requirements. A simple example of a business requirement is “After choosing a branch office name, information about the branch’s customer account managers will appear in a new window. The window will present manager identification and summary information about each manager’s customer base: <list of data elements>.” Other requirements provide details on how the data will be summarized, formatted and displayed.
* A *defect* is a variance between the expected and actual result. The defect’s ultimate source may be traced to a fault introduced in the specification, design, or development (coding) phases.

## Testing methods

Software testing methods are traditionally divided into black box testing and white box testing. These two approaches are used to describe the point of view that a test engineer takes when designing test cases.

#### Black box testing

Black box testing treats the software as a "black box"—without any knowledge of internal implementation. Black box testing methods include: equivalence partitioning, boundary value analysis, all-pairs testing, fuzz testing, model-based testing, traceability matrix, exploratory testing and specification-based testing.

**Specification-based testing**: Specification-based testing aims to test the functionality of software according to the applicable requirements. Thus, the tester inputs data into, and only sees the output from, the test object. This level of testing usually requires thorough test cases to be provided to the tester, who then can simply verify that for a given input, the output value (or behavior), either "is" or "is not" the same as the expected value specified in the test case.

Specification-based testing is necessary, but it is insufficient to guard against certain risks.

**Advantages and disadvantages**: The black box tester has no "bonds" with the code, and a tester's perception is very simple: a code *must* have bugs. Using the principle, "Ask and you shall receive," black box testers find bugs where programmers do not. *But,* on the other hand, black box testing has been said to be "like a walk in a dark labyrinth without a flashlight," because the tester doesn't know how the software being tested was actually constructed. As a result, there are situations when (1) a tester writes many test cases to check something that could have been tested by only one test case, and/or (2) some parts of the back-end are not tested at all.

Therefore, black box testing has the advantage of "an unaffiliated opinion," on the one hand, and the disadvantage of "blind exploring," on the other.

#### White box testing

White box testing is when the tester has access to the internal data structures and algorithms including the code that implement these.

Types of white box testing

* API testing (application programming interface) - Testing of the application using Public and Private APIs
* Code coverage - creating tests to satisfy some criteria of code coverage (e.g., the test designer can create tests to cause all statements in the program to be executed at least once)
* Fault injection methods - improving the coverage of a test by introducing faults to test code paths
* Mutation testing methods
* Static testing - White box testing includes all static testing

### A sample testing cycle

Although variations exist between organizations, there is a typical cycle for testing:

* **Requirements analysis**: Testing should begin in the requirements phase of the software development life cycle. During the design phase, testers work with developers in determining what aspects of a design are testable and with what parameters those tests work.
* **Test planning**: Test strategy, test plan, tested creation. Since many activities will be carried out during testing, a plan is needed.
* **Test development**: Test procedures, test scenarios, test cases, test datasets, test scripts to use in testing software.
* **Test execution**: Testers execute the software based on the plans and tests and report any errors found to the development team.
* **Test reporting**: Once testing is completed, testers generate metrics and make final reports on their test effort and whether or not the software tested is ready for release.
* **Test result analysis**: Or Defect Analysis, is done by the development team usually along with the client, in order to decide what defects should be treated, fixed, rejected (i.e. found software working properly) or deferred to be dealt with later.
* **Defect Retesting**: Once a defect has been dealt with by the development team, it is retested by the testing team.
* **Regression testing**: It is common to have a small test program built of a subset of tests, for each integration of new, modified, or fixed software, in order to ensure that the latest delivery has not ruined anything, and that the software product as a whole is still working correctly.
* **Test Closure**: Once the test meets the exit criteria, the activities such as capturing the key outputs, lessons learned, results, logs, documents related to the project are archived and used as a reference for future projects.

# 10.Conclusion:

This package is designed and developed in a compact manner, which is ready to meet the user’s specification and to serve them in an effective as well as in an enhanced manner. The actual problem has been observed with keen interest and it has been defined and analyzed in such a way that it never causes choice to the user. More ever the limitation that has been prevailing in the existing system had been overcome to suit the need of the user.

High precision and care has been taken to design the data base, input forms an output reports since they should be given due importance which could otherwise to serious consequences thus affecting the whole system. The system thus developed has been implemented successfully which has been performed to scrutinized the validation of each data and errors were spotted out and then finally cleared in a sophisticated manner.

The added feature of this system is that it has been provided with many provisions for future enhancement in order to maintain the system in such a way that the future requirement of the user could also be satisfied and upgraded.

# 11.References

[

1 <https://www.oracle.com/database/what-is-database.html>

]

[

2 <https://www.cloudflare.com/learning/network-layer/what-is-a-packet/>

]

[

3 <https://internetofthingsagenda.techtarget.com/definition/microcontroller>

]

[

4 <https://learn.sparkfun.com/tutorials/what-is-an-arduino/all>

]

[

5 <https://www.instructables.com/id/Obstacle-Avoiding-Robot-Arduino-1/#:~:text=Introduction%3A%20Obstacle%20Avoiding%20Robot%20(Arduino)&text=An%20obstacle%20avoiding%20robot%20is,and%20makes%20a%20step%20back>.

]

[

6 <https://www.tutorialspoint.com/arduino/arduino_ultrasonic_sensor.htm#:~:text=Advertisements,or%201%E2%80%9D%20to%2013%20feet>.

]

[

7<https://www.codewithc.com/network-packet-sniffer-java-project/>

]

[

8<https://www.codewithc.com/online-medical-management-system-java-project/>

]

[

9 [Airline reservations system - Wikipedia](https://en.wikipedia.org/wiki/Airline_reservations_system)

]

## <https://internetofthingsagenda.techtarget.com/definition/microcontroller>

## Books

* Mastering Visual Basic 6 (Paperback)
* Visual Basic Black Book (Paperback)
* SQL Bible, 2nd Edition (Paperback)
* Database Development in Visual Basic