

SILIGURI INSTITUTE OF TECHNOLOGY

CS 892

Supermarket Billing System

BY

CSE_PROJ_2021_03

Name of Students	Roll No.
1. Rinki Kundu	11900117032
2. Suswagata Chakraborty	11900117012
3. Shweta Das	11900117018
4. Tanushree Pandit	11900117010

Under the Guidance

of

Prof. Debajyoti Guha

Submitted to the Department of **Computer Science & Engineering** in partial fulfillment of the requirements for the award of the degree Bachelor of Technology in **Computer Science & Engineering**.

Year of Submission: 2021



Siliguri Institute of Technology

P.O. SUKNA, SILIGURI, DIST. DARJEELING, PIN: 734009

Tel: (0353)2778002/04, Fax: (0353) 2778003

DECLARATION

-

This is to certify that Report entitled "Supermarket Billing System" which is submitted by me in partial fulfillment of the requirement for the award of degree B.Tech. in **Computer Science Engineering** at **Siliguri Institute of Technology** under **Maulana Abul Kalam Azad University of Technology**, West Bengal. We took the help of other materials in our dissertation which have been properly acknowledged. This report has not been submitted to any other Institute for the award of any other degree.

Date: 05.07.21

SN	Name of the Student	Roll No	Signature
1	Rinki Kundu	11900117032	Rinki Kundu 05/07/2021
2	Suswagata Chakraborty	11900117012	Suswagata Chakraborty 5/7/21
3	Shweta Das	11900117018	Shweta Das 05/07/2021
4	Tanushree Pandit	11900117010	Tanushree Pandit 05/07/21

CERTIFICATE

This is to certify that the project report entitled “Supermarket Billing System” submitted to **Department of Computer Science & Engineering of Siliguri Institute of Technology** in partial fulfilment of the requirement for the award of the degree of **Bachelor of Technology in Computer Science & Engineering** during the academic year **2019-20**, is a bonafide record of the project work carried out by them under my guidance and supervision.

Project Group Number : 03			
SN	Name of the students	Registration No	Roll No
1.	Rinki Kundu	171190110064	11900117032
2.	Suswagata Chakraborty	171190110084	11900117012
3.	Shweta Das	171190110078	11900117018
4.	Tanushree Pandit	171190110086	11900117010

Signature of Project Guide

Name of the Guide:

Signature of the HOD

Department of Computer Science & Engineering

Acknowledgement

Success of any project depends largely on the encouragement and guidelines of many others. We take this sincere opportunity to express our gratitude to the people who have been instrumental in the successful completion of this project work.

We would like to show our greatest appreciation to our project guide Prof. Debajyoti Guha. We always feel motivated and encouraged every time by his valuable advice and constant inspiration; without his encouragement and guidance this project would not have materialized. We would give special thanks to our HOD Mr. Anupam Mukherjee and all other teachers for their assistance during the various stages of the project. We are also very thankful to our respected director Dr. Pradosh Kumar Adhvaryu.

Words are inadequate in offering our thanks to the other classmates and project assistants of our college for their encouragement and cooperation in carrying out this project work. The guidance and support received from all the members and who are contributing to this project, was vital for the success of this project.

Signature of all the group members with date

Rinki Kundu
05/07/2021

Suswagata Chakraborty
5/7/21.

Shweta Das
05/07/2021

Tanushree Pandit
05/07/21

Table of Contents

- 1. Introduction and objectives
 - 1.1 Introduction.....8
 - 1.2 Objectives.....9
- 2. System ANALYSIS.....9
 - 2.1 Identification of Need..... 9
 - 2.2 Preliminary Investigation 9
 - 2.3 Feasibility study..... 10
 - 2.3.1 Technical Feasibility.....10
 - 2.3.2 Economic Feasibility.....10
 - 2.3.3 Legal Feasibility.....10
 - 2.3.4 Operational Feasibility.....11
 - 2.3.5 Scheduling Feasibility.....11
 - 2.4 Project planning..... 11
 - 2.5 Project Scheduling.....12
 - 2.6 Software Requirement Specification.....12
 - 2.6.1 Introduction.....12

2.6.1.1 Purpose.....	12
2.6.1.2 Project Scope.....	12
2.6.1.3 Document Conventions.....	13
2.6.2 Overall Description.....	13
2.6.2.1 Product Features.....	13
2.6.2.2 Assumptions and Dependencies.....	13
2.6.2.3 Design and Implementation Constraints.....	13
2.6.3 Functional Requirements.....	13
2.6.3.1 Software Requirements.....	13
2.6.3.2 Language used.....	13
2.6.4 Non-Functional Requirements.....	14
2.6.4.1 Usability.....	14
2.6.4.2 Performance.....	14
2.6.4.3 Availability.....	14
2.7 Software Engineering Paradigm Applied.....	14
3. System Design.....	14
3.1 Definition.....	15
3.2 Input.....	15
3.3 Output.....	15
4. Coding.....	18

4.1 Code	
Efficiency.....	18
4.2 Coding	
Standards.....	18
4.3 Codes.....	19
5. Testing.....	22
5.1 Testing	
Objectives.....	22
5.2 Testing	
Principles.....	22
5.3 Software Testing	
Techniques.....	22
5.3.1 White Box	
Testing.....	22
5.3.2 Black Box	
Testing.....	23
5.4 Software Testing Strategies Used.....	25
5.4.1 Unit	
Testing.....	25
5.4.2 Integration	
testing.....	26
5.4.3 System	
Testing.....	26
5.4.4 Acceptance	
Testing.....	26
5.5 Testing for this	
Project.....	26
6. Cost Estimation of the	
Project.....	26

7. Reports.....	28
8. Charts.....	28
8.1 PERT	
Chart.....	28
8.1.1 Steps to Implement a PERT	
Chart.....	28
8.2 GANTT	
Chart.....	28
9. Limitations and	
Conclusion.....	29
9.1 Limitations in the Current	
Version.....	29
9.2 Conclusion.....	29
10. References.....	30

Abstract

Billing systems are the most widely recommended legal technologies, which the corporate might and merchants urge to utilise. Over the years Billing systems have made quite a reputation in any or most industrial practices, the need for precise information is increasing and when it comes to satisfying client’s growing demand, regarding their transactions, it does not discriminate.

Why use a billing system?

- Administrative control using a billing system as it keeps the invoice process optimum and well managed.
- Creating transparency between the organisation and the customer.
- Operation like re-funds, discounters, renewal, one-off transaction and free trials are much more easier.

1.Introduction & Objectives

1.1 Introduction:

This software project is a traditional supermarket billing system with some added functionality. This system is built for fast data processing and bill generation for supermarket customers. The billing system consists of a Mysql database, java JFrame and the effective front end designed in Netbeans IDE. The billing database is a vast collection of product name, price and other product specific data. A product when billed is searched from the database and its price is added to the bill based upon the product quantity. The supermarket billing system is built to help supermarkets calculate and display bills and serve the customer in a faster and efficient manner. This software project consists of an effective and easy Gui to help the employee in easy bill calculation and providing an efficient customer service.

1.2 Objectives:

We aim to create and develop a system that is capable and reliable in the whole transaction flow such as tracking, retrieving and storing data in an appropriate way. In particular it aims to:

- Provide a database that will store information.
- Develop a system that will lessen process delay in terms of receipts and bills. Provide summary reports daily and monthly sales including a reservation report system that can accommodate reservation transactions from customers.

2. System Analysis

2.1 Identification of Need

Identification of Need is a process of determining what and how an end-user would want a product to perform. The needs of an end-user are often non-technical, and they reflect the user's perception of the product, not the actual design specifications, although frequently they are closely related. Identification of needs has two major goals:

- To keep the product focused on the needs of an end user.
- To identify not just the explicit needs of the customer, but also the latent needs.

With our project, we have tried to produce an architecture which keeps both the above points into consideration. From the end user point of view, we have tried to keep the project simple and intuitive enough so that it can easily be handled.

Keeping in mind the latent needs, we have also made the system capable of upscaling.

In addition, as mentioned in the future scope, the system can be further made better by embedding new techniques like steganography using audio and video files.

2.2 Preliminary Investigation

One of the major purposes of the preliminary investigation is the evaluation of the project request. It is the process of collecting information which helps the evaluator to identify the merits of the project and make a judgement on the feasibility of the proposed project.

As the core of any system analysis is a detailed understanding of all-important facts of the topic under investigation. Few of the key questions to be answered are as listed below:

- What is being done?
- How is it being done?
- Who are the end users?
- Is there any problem with the existing system?
- How well does the new system address the current problems?
- Does the new system pose a risk? If yes, can it be overcome?

To answer these questions, first of all, we did an analysis of the topic of interest and then worked on identifying the lacunas in the currently available implementation and the needs or requirements that could be added to improvise the current system and at the same time providing a better result or outcome to the end user.

Here, the biggest challenge we faced was to identify how to generalize our system to address a broader category of users in the best possible manner rather than we got multiple use cases where this system could be validated, as is mentioned already in the introduction and abstract. Keeping in mind the questions mentioned above, we sat together for a session where we discussed what could be the requirements that the new system proposed should address and how the end user would like the product to be and at the end of the session we came up with a list of features that could be incorporated into the system.

2.3 Feasibility Study

A feasibility analysis evaluates the project’s potential for success; therefore, perceived objectivity is an essential factor in the credibility of the study for potential investors and lending institutions. There are five types of feasibility study—separate areas that a feasibility study examines, described below.

2.3.1 Technical Feasibility

This assessment focuses on the technical resources available to the organization. It helps organizations determine whether the technical resources meet capacity and whether the technical team is capable of converting the ideas into working systems. Technical feasibility also involves the evaluation of the hardware, software, and other technical requirements of the proposed system. As an exaggerated example, an organization wouldn’t want to try to put Star Trek’s transporters in their building—currently, this project is not technically feasible.

2.3.2 Economic Feasibility

This assessment typically involves a cost/ benefits analysis of the project, helping organizations determine the viability, cost, and benefits associated with a project before financial resources are allocated. It also serves as an independent project assessment and enhances project credibility—helping decision-makers determine the positive economic benefits to the organization that the proposed project will provide.

2.3.2 Legal Feasibility

This assessment investigates whether any aspect of the proposed project conflicts with legal requirements like zoning laws, data protection acts or social media laws. Let’s say an organization wants to construct a new office building in a specific location. A feasibility study might reveal the organization’s ideal location isn’t zoned for that type of business. That organization has just saved considerable time and effort by learning that their project was not feasible right from the beginning.

2.3.4 Operational Feasibility

This assessment involves undertaking a study to analyse and determine whether—and how well—the organization’s needs can be met by completing the project. Operational feasibility studies also examine how a project plan satisfies the requirements identified in the requirements analysis phase of system development.

2.3.5 Scheduling Feasibility

This assessment is the most important for project success; after all, a project will fail if not completed on time. In scheduling feasibility, an organization estimates how much time the project will take to complete.

When these areas have all been examined, the feasibility analysis helps identify any constraints the proposed project may face, including:

Internal Project Constraints: Technical, Technology, Budget, Resource, etc.

Internal Corporate Constraints: Financial, Marketing, Export, etc.

External Constraints: Logistics, Environment, Laws, and Regulations, etc.

2.4 Project planning

Project planning is the most important management activity. The basic goal of planning is to look into the future, identify the activities that need to be done to complete the project successfully and plan the scheduling and resources. Planning of the project starts long before its development has started. Without a proper plan, no real monitoring or controlling of the project. Lack of proper planning leads to failure of the project later on. The major issues project planning addresses are:

- Process planning
- Effort estimation
- Schedule and resource estimation
- Quality plans
- Risk Management

In our project, we tried to ensure that all the above points are taken care of. In order to achieve process planning, we had devised workflow diagrams and system architecture diagrams to understand the interfacing and data flow between various components/modules and specified the tasks needed to be carried out in each stage. Effort and Cost Estimation was done by taking into account almost all the possible requirements and the cost was calculated accordingly. The schedule of the project was devised and all the resources needed to develop the project were determined. The potential risks that may evolve during the development of the project were noted down and steps were devised to tackle the problems that may arise as the project progresses. Plans have been made to constantly monitor the working of the project and are flexible enough for making changes later on if required. In addition, to cater to the requirements of the project, we kept the provision of syncing up with the initial plan and making minor adjustments to it, if found necessary for the project.

2.5 Project Scheduling

Software project scheduling is an activity that distributes efforts across the planned project duration by allocating the effort to specific software engineering tasks. In order to prepare the schedule, we first identified all the major tasks needed to complete the project and these tasks were broken down into simpler and smaller sub-tasks. All the tasks were allotted the necessary resources and the time estimates were established to complete the tasks.

PERT and GANTT charts were developed to graphically represent the scheduling of different tasks to be performed. The different schedules devised by us are:

Schedule 1: Determine all the entities of the project and interfacing between them. Also determine the software requirements for the system.

Schedule 2: Develop the software part to be embedded in the system.

Schedule 3: Test the software for proper functionality as independent entities.

Schedule 4: Integration of software via the necessary drivers.

Schedule 5: Testing of the integrated system.

Schedule 6: Monitoring the project for maintenance and any further changes if required, keeping the scope of improvement always open.

2.6 Software Requirements Specifications

2.6.1 Introduction

This software project is a traditional supermarket billing system with some added functionality. This system is built for fast data processing and bill generation for supermarket customers. The billing system consists of an sql database and effective front end designed in Asp.net. The billing database is a vast collection of product name, price and other product specific data. A product when billed is searched from the database and its price is added to the bill based upon the product quantity. The system also contains discounts on various products so that the product is offered at discounted price while billing. The supermarket billing system is built to help supermarkets calculate and display bills and serve the customer in a faster and efficient manner. This software project consists of an effective and easy Gui to help the employee in easy bill calculation and providing an efficient customer service.

2.6.1.1 Purpose

This billing system focuses on the development of an information system that will automate manual transactions in the supermarkets.

2.6.1.2 Project Scope

The proposed automated system should generate reports of daily and monthly sales including reservation transactions of products. The user by using this software can add information about new buyers, can update buyers details, can show all details about buyers, can also delete details of a buyer if required. The system can add new products, show all details about products, and can also delete product details. It will generate receipts on every transaction inputted to the system. The software will display a view of calculations of every transaction. For security and privacy of the management, the billing system complies login users with username and password.

2.6.2 Overall Description

2.6.2.1 Product Features

- User-friendly UI
- It works with different algorithms to keep safe from intruders
- The software provides the option of scalability with minimal changes in the code

2.6.2.2 Assumptions and Dependencies

During the design, few scenarios were encountered which required further implementation of a few things which did not fit into the time constraints as described later. Below are a few assumptions that have been taken in.

- The software works fine with the basic skeletal features but one needs to manually copy the text areas to make the system work.
- The image after encoding should be in its raw form, if compressed or resized can lose the information inside it.

2.6.2.3 Design and Implementation Constraints

When we work on a certain project it is not always necessary that we have all the things at our disposal for implementing the same. Below are some of the constraints that we got restricted with during the development of the project.

- Time: working for a project with a deadline time is always a constraint as we have to deliver the output within a stipulated time and hence, we faced some limitation to some extent
- Research: When developing algorithms, one needs ample amount of research on the topic which was limited due to the first constraint.

2.6.3 Functional Requirements

2.6.3.1 Software Requirements

- We are using Netbeans IDE to develop the project.
- Mysql for the database and Java JFrame.

2.6.3.2 Language Used

- Java

2.6.4 Non-Functional Requirements

2.6.4.1 Usability

The system shall be easy to use.

The output produced by the system should be easily understood by the user.

2.6.4.2 Performance

- User friendliness: It was ensured by the end user should be having a very limited amount of time to operate

- Robustness: This is one of the most important things that one tries to make a system capable of, i.e. to be robust and to cater the requirements of all kinds of scenarios. We tried to make the software as robust as possible but some things can be done which were restricted due to the factors mentioned earlier.
- Error handling: Response to user errors and undesired situations have been taken care of to ensure that the system operates without halting.

2.6.4.3 Availability

- Other than the point that we depend on an uninterrupted local server running.

3. System design

3.1 Definition

The most creative and challenging phase of development is System Design. It provides the understanding and procedural details necessary for the logical and physical stages of development. In designing a new system, the system analyst must have a clear understanding of the objectives, which the design aims to fulfil.

The first step is to determine how the output is to be produced and in what format. Second, input data and the master files have to be designed to make the requirements of the proposed output. The operational phases are handled through programme construction and testing.

3.1.1. Input

- Users can input new buyers details, update buyers details.
- Quantity of item purchased by buyer.

3.1.2. Output

- Calculate buyer's bill
- Store and organise the customer's reservation.
- Organise and monitor the sales report.

New Buyer:

The New Buyer page takes details of the buyer. The record is stored in the database and can be viewed with the help of Buyer Detail option.

New Buyer

Name:

Contact No:

Email:

Address:

Gender:

Fig: New Buyer

Update Buyer:

Similarly, if the cashier wants to update information regarding any particular customer then it can be done with the help of the update buyer option by typing in the contact number and then making the necessary changes.

Update Buyer

Contact No:

Name:

Contact No:

Email:

Address:

Gender:

Fig: Update Buyer

Delete Buyer:

The customer information can easily be deleted with the help of the delete buyer option.

The image shows a web form titled "Delete Buyer". At the top left, there is a small icon of a person with a red 'X' over it. The title "Delete Buyer" is in bold black text. Below the title, there is a search bar labeled "Contact No." with a "Search" button to its right. Underneath the search bar, there are five input fields stacked vertically, labeled "Name", "Contact No.", "Email", "Address", and "Gender". At the bottom of the form, there are three buttons: "Delete" (with a red 'X' icon), "Reset" (with a circular arrow icon), and "Close" (with a red 'X' icon).

Fig: Delete Buyer

4. CODING

Coding is the use of computer programming languages to give computers and machines instructions on what actions to perform. Coding is the way humans communicate with machines, and it allows us to create software like programs, operating systems, and mobile apps. Computer programming is the process of designing and building an executable computer program to accomplish a specific computing result. Programming involves tasks such as: analysis, generating algorithms, profiling algorithms' accuracy and resource consumption, and the implementation of algorithms in a chosen programming language (commonly referred to as coding). The source code of a program is written in one or more languages that are intelligible to programmers, rather than machine code, which is directly executed by the central processing unit. The purpose of programming is to find a sequence of instructions that will automate the performance of a task on a computer, often for solving a given problem.

4.1 CODE EFFICIENCY

Code efficiency plays a significant role in applications in a high-execution-speed environment where performance and scalability are paramount. One of the recommended best practices in coding is to ensure

good code efficiency. Well-developed programming codes should be able to handle complex algorithms.

Recommendations for code efficiency include:

- to make use of reusable components whenever possible.
- to make use of error and exception handling at all layers of software, such as UI, logic and data flow.
- to make programming codes that ensure data integrity and consistency.
- to develop programming code that's compliant with the design logic and flow.
- to make use of coding practices applicable to the related software.
- to optimize the use of data access and data management practices.
- to use the best keywords, data types, variables and other available programming concepts to implement the related algorithm.

1.2 Coding Standards

They are a series of procedures that can be defined for a particular programming language specifying a programming style, the methods, & different procedures. These procedures can be for various aspects of the program written in that language. They can be considered as essential attributes of software development. A coding standard makes sure that all the developers working on the project are following certain specified guidelines. The code can be easily understood and proper consistency is maintained. Consistency has a positive impact on the quality of the program and one should maintain it while coding. Also, it should be taken care that the guidelines are homogeneously followed across different levels of the system and they do not contradict each other. The finished program code should look like that it has been written by a single developer, in a single session.

Coding standards are important in Software Development because:

If the coding standards are not defined, the developers could use their own methods which might lead to certain negative effects such as:

1. Security Concerns:

Software becomes vulnerable to attacks if it is inconsistent, contains bugs and errors in logic. Most of the aforementioned problems arise due to the faulty programming code that might have resulted from poor coding practices.

2. Performance Issues:

Poor coding has an adverse effect on the performance of the site. The performance issues comprise a multitude of things like when the user is interacting with the site, server response issues, reusability & flow of the code, etc.

1.3 Codes

The code used in our project is given below:

login.java:

```
private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    int a;
    a = JOptionPane.showConfirmDialog(null,"Do you really want to Close the Application?","Select",
JOptionPane.YES_NO_OPTION );
    if(a==0){
        System.exit(0);
    }
}

private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    if(jTextField1.getText().equals("bms") && jPasswordField1.getText().equals("admin")){
        setVisible(false);
        new home().setVisible(true);
    }
    else
        JOptionPane.showMessageDialog(null,"Incorrect Username or Password");
}

private void jCheckBox1ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    if(jCheckBox1.isSelected()){
        jPasswordField1.setEchoChar((char)0);
    }
    else
        jPasswordField1.setEchoChar('*');
}
}
```

home.java:

```
private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    new updateBuyer().setVisible(true);
}

private void jButton4ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    new buyersDetails().setVisible(true);
}

private void jButton8ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    new productDetails().setVisible(true);
}
}
```

```

private void jButton10ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    new billing().setVisible(true);
}

private void jButton11ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    int a = JOptionPane.showConfirmDialog(null,"Do you really want to Logout?","Select",
JOptionPane.YES_NO_OPTION);
    if(a==0){
        setVisible(false);
        new login().setVisible(true);
    }
}

private void jButton12ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    int a = JOptionPane.showConfirmDialog(null,"Do you really want to Logout?","Select",
JOptionPane.YES_NO_OPTION);
    if(a==0){
        System.exit(0);
    }
}

private void jButton1MouseClicked(java.awt.event.MouseEvent evt) {
    // TODO add your handling code here:
    if(z==0){
        try {
            Thread.sleep(250);
            jButton2.setVisible(true);
            jLabel1.setVisible(true);
        }
        catch(InterruptedException e)
        {}
    }
    else{
        jButton2.setVisible(false);
        jButton3.setVisible(false);
        jButton4.setVisible(false);
        jButton5.setVisible(false);
        jButton6.setVisible(false);
        jButton7.setVisible(false);
        jButton8.setVisible(false);
        jButton9.setVisible(false);
        jButton10.setVisible(false);
        jButton11.setVisible(false);
        jButton12.setVisible(false);
    }
}

```

```

jLabel1.setVisible(false);
jLabel2.setVisible(false);
jLabel3.setVisible(false);
jLabel4.setVisible(false);
jLabel5.setVisible(false);
jLabel6.setVisible(false);
jLabel7.setVisible(false);
jLabel8.setVisible(false);
jLabel9.setVisible(false);
jLabel10.setVisible(false);
jLabel11.setVisible(false);
z=0;
}
}

private void jButton2ComponentShown(java.awt.event.ComponentEvent evt) {
// TODO add your handling code here:
try{
    Thread.sleep(250);
    jButton3.setVisible(true);
    jLabel2.setVisible(true);
}
catch(InterruptedException e)
{}
}

private void jButton3ComponentShown(java.awt.event.ComponentEvent evt) {
// TODO add your handling code here:
try{
    Thread.sleep(250);
    jButton4.setVisible(true);
    jLabel3.setVisible(true);
}
catch(InterruptedException e)
{}
}

private void jButton4ComponentShown(java.awt.event.ComponentEvent evt) {
// TODO add your handling code here:
try{
    Thread.sleep(250);
    jButton5.setVisible(true);
    jLabel4.setVisible(true);
}
catch(InterruptedException e)
{}
}

private void jButton5ComponentShown(java.awt.event.ComponentEvent evt) {

```

```

// TODO add your handling code here:
try{
    Thread.sleep(250);
    jButton6.setVisible(true);
    jLabel5.setVisible(true);
}
catch(InterruptedException e)
{}
}

private void jButton6ComponentShown(java.awt.event.ComponentEvent evt) {
// TODO add your handling code here:
try{
    Thread.sleep(250);
    jButton7.setVisible(true);
    jLabel6.setVisible(true);
}
catch(InterruptedException e)
{}
}

private void jButton7ComponentShown(java.awt.event.ComponentEvent evt) {
// TODO add your handling code here:
try{
    Thread.sleep(250);
    jButton8.setVisible(true);
    jLabel7.setVisible(true);
}
catch(InterruptedException e)
{}
}

private void jButton8ComponentShown(java.awt.event.ComponentEvent evt) {
// TODO add your handling code here:
try{
    Thread.sleep(250);
    jButton9.setVisible(true);
    jLabel8.setVisible(true);
}
catch(InterruptedException e)
{}
}

private void jButton9ComponentShown(java.awt.event.ComponentEvent evt) {
// TODO add your handling code here:
try{
    Thread.sleep(250);
    jButton10.setVisible(true);
    jLabel9.setVisible(true);
}
}

```

```

    }
    catch(InterruptedExceotion e)
    {}
}

private void jButton10ComponentShown(java.awt.event.ComponentEvent evt) {
    // TODO add your handling code here:
    try{
        Thread.sleep(250);
        jButton11.setVisible(true);
        jLabel10.setVisible(true);
    }
    catch(InterruptedExceotion e)
    {}
}

private void jButton11ComponentShown(java.awt.event.ComponentEvent evt) {
    // TODO add your handling code here:
    try{
        Thread.sleep(250);
        jButton12.setVisible(true);
        jLabel11.setVisible(true);
        z=1;
    }
    catch(InterruptedExceotion e)
    {}
}

private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    /*setVisible(false);*/
    new newBuyer().setVisible(true);
}

private void jButton5ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    new deleteBuyer().setVisible(true);
}

private void jButton6ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    new newProduct().setVisible(true);
}

private void jButton7ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    new updateProduct().setVisible(true);
}

```



```

private void jButton9ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    new deleteProduct().setVisible(true);
}

```

newBuyer.java:

```

private void jTextField1FocusGained(java.awt.event.FocusEvent evt) {
    // TODO add your handling code here:
    if(jTextField1.getText().equals("Enter Name")){
        jTextField1.setText("");
        jTextField1.setForeground(new Color(0,0,0));
    }
}

```

```

private void jTextField1FocusLost(java.awt.event.FocusEvent evt) {
    // TODO add your handling code here:
    if(jTextField1.getText().equals("")){
        jTextField1.setText("Enter Name");
        jTextField1.setForeground(new Color(153,153,153));
    }
}

```

```

private void jTextField2FocusGained(java.awt.event.FocusEvent evt) {
    // TODO add your handling code here:
    if(jTextField2.getText().equals("Enter Contact No")){
        jTextField2.setText("");
        jTextField2.setForeground(new Color(0,0,0));
    }
}

```

```

private void jTextField2FocusLost(java.awt.event.FocusEvent evt) {
    // TODO add your handling code here:
    if(jTextField2.getText().equals("")){
        jTextField2.setText("Enter Contact No");
        jTextField2.setForeground(new Color(153,153,153));
    }
}

```

```

private void jTextField3FocusGained(java.awt.event.FocusEvent evt) {
    // TODO add your handling code here:
    if(jTextField3.getText().equals("Enter Email")){
        jTextField3.setText("");
        jTextField3.setForeground(new Color(0,0,0));
    }
}

```

```

private void jTextField3FocusLost(java.awt.event.FocusEvent evt) {

```

```

// TODO add your handling code here:
if(jTextField3.getText().equals("")){
    jTextField3.setText("Enter Email");
    jTextField3.setForeground(new Color(153,153,153));
}
}

private void jTextField4FocusGained(java.awt.event.FocusEvent evt) {
// TODO add your handling code here:
if(jTextField4.getText().equals("Enter Address")){
    jTextField4.setText("");
    jTextField4.setForeground(new Color(0,0,0));
}
}

private void jTextField4FocusLost(java.awt.event.FocusEvent evt) {
// TODO add your handling code here:
if(jTextField4.getText().equals("")){
    jTextField4.setText("Enter Address");
    jTextField4.setForeground(new Color(153,153,153));
}
}

private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
// TODO add your handling code here:
setVisible(false);
new newBuyer().setVisible(true);
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
// TODO add your handling code here:
String name=jTextField1.getText();
String contactNo=jTextField2.getText();
String email = jTextField3.getText();
String address = jTextField4.getText();
String gender = (String)jComboBox1.getSelectedItem();

try{
//Class.forName("com.mysql.jdbc.Driver");
Connection con =
ConnectionProvider.getCon();//DriverManager.getConnection("jdbc:mysql://localhost:3306/bms","root",
"password");
Statement st = con.createStatement();
st.executeUpdate("insert into buyer
values("+name+"",""+contactNo+"",""+email+"",""+address+"",""+gender+"");
JOptionPane.showMessageDialog(null,"Successfully Updated");
setVisible(false);
new newBuyer().setVisible(true);
}
catch(HeadlessException | SQLException e){
JOptionPane.showMessageDialog(null, "Mobile number is already Exist");
}
}

```

```

    }
updatebuyer.java:
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    String contactNo=jTextField1.getText();
    try{
        Connection con = ConnectionProvider.getCon();
        Statement st = con.createStatement();
        ResultSet rs = st.executeQuery("select * from buyer where contactNo='"+contactNo+"'");
        if(rs.next()){
            jTextField2.setText(rs.getString(1));
            jTextField3.setText(rs.getString(2));
            jTextField4.setText(rs.getString(3));
            jTextField7.setText(rs.getString(4));
            jTextField8.setText(rs.getString(5));
            jTextField1.setEditable(false);
        }
        else{
            JOptionPane.showMessageDialog(null, "Contact No. does not Exist");
        }
    }
    catch(HeadlessException | SQLException e){
        JOptionPane.showMessageDialog(null,e);
    }
}

private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    String contactNo1=jTextField1.getText();
    String name = jTextField2.getText();
    String contactNo = jTextField3.getText();
    String email = jTextField4.getText();
    String address = jTextField7.getText();
    String gender = jTextField8.getText();

    try{
        Connection con = ConnectionProvider.getCon();
        Statement st = con.createStatement();
        st.executeUpdate("update buyer set
name='"+name+"',contactNo='"+contactNo+"',email='"+email+"',address='"+address+"',gender='"+gende
r+"' where contactNo='"+contactNo1+"'");
        JOptionPane.showMessageDialog(null,"Successfully Updated");
        setVisible(false);
        new updateBuyer().setVisible(true);
    }
    catch(HeadlessException | SQLException e){
        JOptionPane.showMessageDialog(null, e);
    }
}
}

```

buyerDetails.java

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    try{
        jTable1.print(JTable.PrintMode.NORMAL);
    }
    catch(Exception e){
        JOptionPane.showMessageDialog(null, e);
    }
}

private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    setVisible(false);
}

private void formComponentShown(java.awt.event.ComponentEvent evt) {
    // TODO add your handling code here:
    try{

        Connection con = ConnectionProvider.getCon();
        Statement st = con.createStatement();
        ResultSet rs=(ResultSet) st.executeQuery("select * from buyer");

        jTable1.setModel(DbUtils.resultSetToTableModel((ResultSet) rs));
    }
    catch(SQLException e){
        JOptionPane.showMessageDialog(null, e);
    }
}
```

deleteBuyer.java

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    String contactNo=jTextField1.getText();
    try{
        Connection con = ConnectionProvider.getCon();
        Statement st = con.createStatement();
        ResultSet rs = st.executeQuery("select * from buyer where contactNo='"+contactNo+"'");
        if(rs.next()){
            jTextField2.setText(rs.getString(1));
            jTextField3.setText(rs.getString(2));
            jTextField4.setText(rs.getString(3));
            jTextField5.setText(rs.getString(4));
            jTextField6.setText(rs.getString(5));
        }
    }
}
```

```

        jTextField1.setEditable(false);
    }
    else{
        JOptionPane.showMessageDialog(null, "Contact No. does not Exist");
    }
}
catch(Exception e){
    JOptionPane.showMessageDialog(null,e);
}
}

private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    String contactNo = jTextField1.getText();
    int a= JOptionPane.showConfirmDialog(null, "Do you really want to Delete?", "Select",
JOptionPane.YES_NO_OPTION);
    if(a==0){
        try{
            Connection con = ConnectionProvider.getCon();
            Statement st = con.createStatement();
            st.executeUpdate("delete from buyer where contactNo='"+contactNo+"'");
            setVisible(false);
            new deleteBuyer().setVisible(true);
        }
        catch(Exception e){
            JOptionPane.showMessageDialog(null,e);
        }
    }
}
}
}

```

newProduvt.java:

```

private void formComponentShown(java.awt.event.ComponentEvent evt) {
    // TODO add your handling code here:
    try{
        Connection con = ConnectionProvider.getCon();
        Statement st = con.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE,
ResultSet.CONCUR_READ_ONLY);
        ResultSet rs = st.executeQuery("select max(pId) from product");
        if(rs.first()){
            int id = rs.getInt(1);
            id = id+1;
            String str = String.valueOf(id);
            jLabel4.setText(str);
        }
        else{
            jLabel4.setText("1");
        }
    }
}
}

```

```

        catch(Exception e){
            JOptionPane.showMessageDialog(null, e);
        }
    }

private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    setVisible(false);
    new newProduct().setVisible(true);
}

private void jTextField1FocusGained(java.awt.event.FocusEvent evt) {
    // TODO add your handling code here:
    if(jTextField1.getText().equals("Enter Product Name")){
        jTextField1.setText("");
        jTextField1.setForeground(new Color(0,0,0));
    }
}

private void jTextField1FocusLost(java.awt.event.FocusEvent evt) {
    // TODO add your handling code here:
    if(jTextField1.getText().equals("")){
        jTextField1.setText("Enter Product Name");
        jTextField1.setForeground(new Color(153,153,153));
    }
}

private void jTextField2FocusGained(java.awt.event.FocusEvent evt) {
    // TODO add your handling code here:
    if(jTextField2.getText().equals("Enter Rate")){
        jTextField2.setText("");
        jTextField2.setForeground(new Color(0,0,0));
    }
}

private void jTextField2FocusLost(java.awt.event.FocusEvent evt) {
    // TODO add your handling code here:
    if(jTextField2.getText().equals("")){
        jTextField2.setText("Enter Rate");
        jTextField2.setForeground(new Color(153,153,153));
    }
}

private void jTextField3FocusGained(java.awt.event.FocusEvent evt) {
    // TODO add your handling code here:
    if(jTextField3.getText().equals("Enter Description")){
        jTextField3.setText("");
    }
}

```

```

        jTextField3.setForeground(new Color(0,0,0));
    }
}

private void jTextField3FocusLost(java.awt.event.FocusEvent evt) {
    // TODO add your handling code here:
    if(jTextField3.getText().equals("")){
        jTextField3.setText("Enter Description");
        jTextField3.setForeground(new Color(153,153,153));
    }
}

private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    setVisible(false);
}

private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    String pId = jLabel4.getText();
    String pName = jTextField1.getText();
    String rate = jTextField2.getText();
    String description = jTextField3.getText();
    String activate = (String)jComboBox1.getSelectedItem();
    try{
        Connection con = ConnectionProvider.getCon();
        Statement st = con.createStatement();
        st.executeUpdate("insert into product
values("+pId+"",""+pName+"",""+rate+"",""+description+"",""+activate+"")");
        JOptionPane.showMessageDialog(null,"Successfully Updated");
        setVisible(false);
        new newProduct().setVisible(true);
    }
    catch(Exception e){
        JOptionPane.showMessageDialog(null, e);
    }
}

```

updateProduct.java

```

private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    String pId=jTextField1.getText();
    try{
        Connection con = ConnectionProvider.getCon();
        Statement st = con.createStatement();
        ResultSet rs = st.executeQuery("select * from product where pId='"+pId+"'");
        if(rs.next()){
            jTextField2.setText(rs.getString(2));

```

```

        jTextField3.setText(rs.getString(3));
        jTextField4.setText(rs.getString(4));
        jTextField5.setText(rs.getString(5));
        jTextField1.setEditable(false);
    }
    else {
        JOptionPane.showMessageDialog(null, "Product ID does not Exist");
    }
}
catch(Exception e){
    JOptionPane.showMessageDialog(null, e);
}
}

private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    String pId = jTextField1.getText();
    String pName = jTextField2.getText();
    String rate = jTextField3.getText();
    String description = jTextField4.getText();
    String activate = jTextField5.getText();
    try {
        Connection con = ConnectionProvider.getCon();
        Statement st = con.createStatement();
        st.executeUpdate("update product set
pName='"+pName+"',rate='"+rate+"',description='"+description+"',activate='"+activate+"' where
pId='"+pId+"'");
        JOptionPane.showMessageDialog(null, "Successfully Updated");
        setVisible(false);
        new updateProduct().setVisible(true);
    }
    catch(Exception e){
        JOptionPane.showMessageDialog(null, e);
    }
}
}

```

productDetails.java:

```

private void formComponentShown(java.awt.event.ComponentEvent evt) {
    // TODO add your handling code here:
    try {
        Connection con = ConnectionProvider.getCon();
        Statement st = con.createStatement();
        ResultSet rs = st.executeQuery("select * from product");
        jTable1.setModel(DbUtils.resultSetToTableModel(rs));
    }
    catch(Exception e){
        JOptionPane.showMessageDialog(null, e);
    }
}
}

```



```

}

private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    setVisible(false);
}

```

```

private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    try{
        jTable1.print(JTable.PrintMode.NORMAL);
    }
    catch(Exception e){
        JOptionPane.showMessageDialog(null, e);
    }
}

```

deleteProduct.java:

```

private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    String pId = jTextField1.getText();
    try{
        Connection con = ConnectionProvider.getCon();
        Statement st = con.createStatement();
        ResultSet rs = st.executeQuery("select * from product where pId='"+pId+"'");
        if(rs.next()){
            jTextField2.setText(rs.getString(2));
            jTextField3.setText(rs.getString(3));
            jTextField4.setText(rs.getString(4));
            jTextField5.setText(rs.getString(5));
            jTextField1.setEditable(false);
        }
        else{
            JOptionPane.showMessageDialog(null,"ProductId does not Exist");
        }
    }
    catch(Exception e){

    }
}

```

```

private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    String pId = jTextField1.getText();
    int a = JOptionPane.showConfirmDialog(null,"Do you want to
Delete?","Select",JOptionPane.YES_NO_OPTION);
    if(a==0){
        try{
            Connection con = ConnectionProvider.getCon();

```

```

        Statement st = con.createStatement();
        st.executeUpdate("delete from product where pId='"+pId+"'");
        setVisible(false);
        new deleteProduct().setVisible(true);
    }
    catch(Exception e){
        JOptionPane.showMessageDialog(null, e);
    }
}
}

```

billing.java:

```

private void jTextField1ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    String name = jTextField1.getText();
    try{
        Connection con = ConnectionProvider.getCon();
        Statement st = con.createStatement();
        ResultSet rs = st.executeQuery("select * from buyer where name like '"+name+"%";
        if(rs.next()){
            jTextField1.setText(rs.getString(1));
            jTextField2.setText(rs.getString(2));
            jTextField3.setText(rs.getString(3));
            jTextField4.setText(rs.getString(4));
        }
        else{
            jTextField2.setText("");
            jTextField3.setText("");
            jTextField4.setText("");
        }
    }
    catch(Exception e){
        JOptionPane.showMessageDialog(null, e);
    }
}

```

```

private void jTextField5ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    String pId = jTextField5.getText();
    try{
        Connection con = ConnectionProvider.getCon();
        Statement st = con.createStatement();
        ResultSet rs = st.executeQuery("select * from product where pId = '"+pId+"'");
        if(rs.next()){
            jTextField6.setText(rs.getString(2));
            jTextField7.setText(rs.getString(3));
            jTextField8.setText("1");
            jTextField9.setText(rs.getString(4));
        }
    }
}

```

```

    }
    else{
        jTextField6.setText("");
        jTextField7.setText("");
        jTextField8.setText("");
        jTextField9.setText("");
    }
}
catch(Exception e){
    JOptionPane.showMessageDialog(null, e);
}
}

private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    int price = Integer.parseInt(jTextField7.getText());
    int quantity = Integer.parseInt(jTextField8.getText());
    int total = price*quantity;
    DefaultTableModel model = (DefaultTableModel) jTable3.getModel();
    model.addRow(new Object[] {jTextField6.getText(), jTextField9.getText(), price, quantity,total});
    finalTotal=finalTotal+total;
    String finalTotal1=String.valueOf(finalTotal);
    jTextField10.setText(finalTotal1);
}

private void jTextField11ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    String paidAmount = jTextField11.getText();
    int z = Integer.parseInt(paidAmount);
    finalTotal = z-finalTotal;
    String finalTotal1= String.valueOf(finalTotal);
    jTextField12.setText(finalTotal1);
    jTextField12.setEditable(false);
}

private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    setVisible(false);
    new billing().setVisible(true);
}

private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    String name = jTextField1.getText();
    String contactNo = jTextField2.getText();
    String email = jTextField3.getText();
    String address = jTextField4.getText();
    String path ="D:\\PROJECT\\ALL Bills";

```

```

com.itextpdf.text.Document doc = new com.itextpdf.text.Document();
try{
    PdfWriter.getInstance(doc,new FileOutputStream(path+" "+name+" "+jLabel5.getText()+".pdf"));
    doc.open();
    Paragraph paragraph1 = new Paragraph("                Billing Management System\n
Contact Number:(+91)0123456789\n\n");
    doc.add(paragraph1);
    Paragraph paragraph2 = new Paragraph("Date & Time:"+jLabel5.getText()+
"+jLabel6.getText()+"\nBuyer Details:\nName:"+name+"\nContact
No.:"+contactNo+"\nEmail:"+email+"\nAddress:"+address+"\n\n");
    doc.add(paragraph2);
    PdfPTable tbl = new PdfPTable(5);
    tbl.addCell("Name");
    tbl.addCell("Description");
    tbl.addCell("Rate");
    tbl.addCell("Quantity");
    tbl.addCell("Sub Total");
    for(int i=0;i<jTable3.getRowCount();i++){
        String n = jTable3.getValueAt(i, 0).toString();
        String d = jTable3.getValueAt(i, 1).toString();
        String r = jTable3.getValueAt(i, 2).toString();
        String q = jTable3.getValueAt(i, 3).toString();
        String s = jTable3.getValueAt(i, 4).toString();
        tbl.addCell(n);
        tbl.addCell(d);
        tbl.addCell(r);
        tbl.addCell(q);
        tbl.addCell(s);
    }
    doc.add(tbl);
    Paragraph paragraph3 = new Paragraph("\nTotal:"+jTextField10.getText()+"\nPaid
Amount:"+jTextField11.getText()+"\nReturn Amount"+jTextField12.getText()+"\n\nThank you for
visiting!! Please come agin.\n From Suswagata, Rinki, Shweta, Tanushree");
    doc.add(paragraph3);
    JOptionPane.showMessageDialog(null, "Bill Generated");
    setVisible(true);
    new billing().setVisible(true);
}
catch(Exception e){
    JOptionPane.showMessageDialog(null, e);
}
doc.close();
}

```

2. TESTING

Software testing is a process, to evaluate the functionality of a software application with an intent to find whether the developed software met the specified requirements or not and to identify the defects to ensure that the product is defect-free in order to produce the quality product. Software testing is defined as an activity to check whether the actual results match the expected results and to ensure that the software system is Defect free. It involves execution of a software component or system component to evaluate one or more properties of interest. Software testing also helps to identify errors, gaps or missing requirements in contrast to the actual requirements. It can be either done manually or using automated tools.

5.1 Testing Objectives

- Testing is a process of executing a program with the intent of finding an error.
- A successful test is one that uncovers an as-yet-undiscovered error.
- A good test case is one that has a high probability of finding an as yet undiscovered error.

5.2 Testing Principles

Before applying methods to design effective test cases, we must understand the basic principles that guide software testing. Principles are as follows:

- All tests should meet the customer requirements.
- To make our software, testing should be done by a third party.
- Exhaustive testing is not possible.
- All tests to be conducted should be planned before implementing it.
- Start testing with small parts and extend it to the large parts.

5.3 Software Testing Techniques

The importance of software testing and its impact on software cannot be underestimated. Software testing is a fundamental component of software quality assurance and represents a review of specification, design, and coding. The greater visibility of software systems and the cost associated with software failure are motivating factors for planning, through testing. It is not uncommon for a software organization to spend 40% of its effort on testing.

5.3.1 White Box Testing

White box testing (also called structural testing or glass box testing) is performed to test the program internal structure. To perform white box testing, the tester should have a thorough knowledge of the program internals along with the purpose of developing the software. During this testing, the entire software implementation is also included with the specification. This helps in detecting errors even with unclear or incomplete software specification.

The goal of white box testing is to ensure that the test cases (developed by software testers by using white box testing) exercise each path through a program. That is, test cases ensure that all internal structures in the program are developed according to design specifications. The test cases also ensure the following:

- All independent paths within the program have been exercised at least once.

- All internal data structures have been exercised.
- All loops (simple loops, concatenated loops, and nested loops) have been executed at and within their specified boundaries.
- All segments present between the control's structures (like 'switch' statement) have been executed at least once. Each branch (like 'case' statement) has been exercised at least once.
- All the logical conditions as well as their combinations have been executed at least once for both true and false paths.

There are various types of white box testing strategies. They are:

5.3.1.1 Basis Path Testing

The basic path testing is the same, but it is based on a White Box Testing method, that defines test cases based on the flows or logical path that can be taken through the program. In software engineering, Basis path testing involves execution of all possible blocks in a program and achieves maximum path coverage with the least number of test cases. It is a hybrid of branch testing and path testing methods. The objective behind the basis path in software testing is that it defines the number of independent paths, thus the number of test cases needed can be defined explicitly.

5.3.1.2 Control Structure Testing

Control structure testing is used to increase the coverage area by testing various control structures present in the program. The different types of testing performed under control structure. Different control structure techniques are:

- Condition Testing
- Data Flow Testing
- Loop Testing

5.3.1.3 Unit Testing

Unit testing is a testing technique using which individual modules are tested to determine if there are any issues by the developer himself. It is concerned with functional correctness of the standalone modules. The main aim is to isolate each unit of the system to identify, analyse and fix the defects.

5.3.2 Black Box Testing

Black box testing checks the functional requirements and examines the input and output data of these requirements. When black box testing is performed, only the sets of 'legal' input and corresponding outputs should be known to the tester and not the internal logic of the program to produce that output. Hence to determine the functionality, the outputs produced for the given sets of input are observed.

The black box testing is used to find the errors listed below.

- Interface errors such as functions, which are unable to send or receive data to/from other software.
- Incorrect functions that lead to undesired output when executed.
- Missing functions and erroneous data structures.

- Erroneous databases, which lead to incorrect outputs when the software uses the data present in these databases for processing.
- Incorrect conditions due to which the functions produce incorrect outputs when they are executed.
- Termination errors such as certain conditions due to which a function enters a loop that forces it to execute indefinitely.

In this testing, testers derive various test cases to exercise the functional requirements of the software without considering implementation details of the code. Then, the software is run for the specified sets of input and the outputs produced for each input set is compared against the specifications to conform to the correctness. If they are specified by the user, then the software is considered to be correct, else the software is tested for the presence of errors in it.

5.3.2.1 Functional Testing

Functional testing is a type of software testing that validates the software system against the functional requirements/specifications. The purpose of Functional tests is to test each function of the software application, by providing appropriate input, verifying the output against the Functional requirements.

5.3.2.2 Smoke Testing

Smoke testing is a type of software testing that determines whether the deployed build is stable or not. The purpose of Smoke Tests is to confirm whether the QA team can proceed with further testing. Smoke tests are a minimal set of tests run on each build.

5.3.2.3 Recovery Testing

Recovery testing verifies the system's ability to recover from points of failure like software/hardware crashes, network failures etc. The purpose of Recovery Testing is to determine whether operations can be continued after a disaster or after the integrity of the system has been lost. It involves reverting to a point where the integrity of the system was known and then reprocessing transactions up to the point of failure.

5.3.2.4 Alpha Testing

Alpha testing is a type of acceptance testing; performed to identify all possible issues/bugs before releasing the product to everyday users or the public. The focus of this testing is to simulate real users by using a black box and white box techniques. The aim is to carry out the tasks that a typical user might perform. Alpha testing is carried out in a lab environment and usually, the testers are internal employees of the organization. To put it as simply as possible, this kind of testing is called alpha only because it is done early on, near the end of the development of the software, and before beta testing.

5.3.2.5 Beta Testing

Beta Testing is one of the Acceptance Testing types, which adds value to the product as the end-user (intended real user) validates the product for functionality, usability, reliability, and compatibility. Inputs provided by the end-users helps in enhancing the quality of the product further and leads to its success. This also helps in decision making to invest further in the future products or the same product for improvisation.

In order to systematically test a set of functions, it is necessary to design test cases. Testers can create test cases from the requirement specification document using the following black box testing techniques:

□ Boundary Value Analysis

It is the widely used black-box testing, which is also the basis for equivalence testing. Boundary value analysis tests the software with test cases with extreme values of test data. BVA is used to identify the flaws or errors that arise due to the limits of input data.

□ Equivalence partitioning

This test case designing technique checks the input and output by dividing the input into equivalent classes. The data must be tested at least once to ensure maximum test coverage of data. It is the exhaustive form of testing, which also reduces the redundancy of inputs.

□ State Transition Testing

This testing technique uses the inputs, outputs, and the state of the system during the testing phase. It checks the software against the sequence of transitions or events among the test data.

□ Decision Table Testing

This approach creates test cases based on various possibilities. It considers multiple test cases in a decision table format where each condition is checked and fulfilled, to pass the test and provide accurate output. It is preferred in case of various input combinations and multiple possibilities.

5.4 Software Testing Strategies Used

A strategy for software testing integrates software test case design techniques into a well-planned set of steps that cause the production of software. A software test strategy provides a road map for the software developer, the quality assurance organization and the customer.

5.4.1 Unit Testing

Unit testing is a level of software testing where individual units or components of software are tested. The purpose is to validate that each unit of the software performs as designed. A unit is the smallest testable part of any software. It usually has one or a few inputs and usually a single output. In procedural programming, a unit may be an individual program, function, procedure, etc. In object-oriented programming, the smallest unit is a method, which may belong to a base or super class, abstract class or derived or child class. Unit testing frameworks, drivers, stubs, and mock or fake objects are used to assist in unit testing. Unit testing increases confidence in changing code. If good unit tests are written and if they are run every time any code is changed. Codes are more reusable. In order to make unit testing possible, codes need to be modular. The cost of fixing a defect detected during unit testing is lesser in comparison to that of defects detected at higher levels. Debugging is easy. When a test fails, only the latest changes need to be debugged. Codes are more reliable.

5.4.2 Integration Testing

Integration testing is defined as a type of testing where software modules are integrated logically and tested as a group. A typical software project consists of multiple software modules, coded by different programmers. The purpose of this level of testing is to expose defects in the interaction between these software modules when they are integrated. Integration Testing focuses on checking data communication amongst these modules. Hence it is also termed as 'I & T' (Integration and Testing), 'String Testing' and sometimes 'Thread Testing'.

5.4.3 System Testing

System testing is a level of testing that validates the complete and fully integrated software product. The purpose of a system test is to evaluate the end-to-end system specifications. Usually, the software is only one element of a larger computer-based system. Ultimately, the software is interfaced with other software/hardware systems. System Testing is actually a series of different tests whose sole purpose is to exercise the full computer-based system. System Testing (ST) is a black box testing technique performed to evaluate the complete system's compliance against specified requirements. In System testing, the functionalities of the system are tested from an end-to-end perspective. System Testing is usually carried out by a team that is independent of the development team in order to measure the quality of the system unbiased. It includes both functional and Non-Functional testing.

5.4.4 Acceptance Testing

Acceptance testing is a level of software testing where a system is tested for acceptability. The purpose of this test is to evaluate the system's compliance with the business requirements and assess whether it is acceptable for delivery. Formal testing with respect to user needs, requirements, and business processes conducted to determine whether or not a system satisfies the acceptance criteria and to enable the user, customers or other authorized entity to determine whether or not to accept the system. Internal Acceptance Testing (Also known as Alpha Testing) is performed by members of the organization that developed the software but who are not directly involved in the project (Development or Testing). Usually, it is the members of Product Management, Sales and/or Customer Support. External Acceptance Testing is performed by people who are not employees of the organization that developed the software. Customer Acceptance Testing is performed by the customers of the organization that developed the software. They are the ones who asked the organization to develop the software. [This is in the case of the software not being owned by the organization that developed it.] User Acceptance Testing (Also known as Beta Testing) is performed by the end users of the software. They can be the customers themselves or the customers' customers.

1. Cost Estimation of the Project

For every project, however noble and practical it might be, until it is within the budget of the organization funding or developing the project, it can't be taken up. Thus, it becomes very essential to identify the requirements of the project correctly, choosing the right components and to calculate the cost of the project keeping in mind all the factors.

COCOMO(Constructive Cost Model) is one of the models which is used to calculate the cost involved in making the project. It is a regression model based on LOC, i.e. the **number of Lines of Code**. It is a procedural cost estimate model for software projects and often used as a process of reliably predicting the various parameters associated with making a project such as size, effort, cost, time and quality.

One of the most fundamental calculations in the COCOMO is the use of the effort equation to estimate the number of Person-Months required for developing a project. Most of the other COCOMO results, including the estimates for Requirements and Maintenance, are derived from this quantity.

The most common approach for estimating effort is to make it a function of a single variable. Often these variables are the project size, and the equation of effort is considered.

$$\text{EFFORT}(E) = a + \text{SIZE}^b$$

Where 'a' and 'b' are constants. Values for these constants for a particular process are determined through regression analysis, which is applied to data about the projects that have been performed in the past.

We have used the intermediate COCOMO for determining the cost estimates of the project which incorporates the effort estimate assessment on each step of the software engineering process. The reason for choosing the intermediate model is the fact that our project involves hardware, software tools as well as coding and hence a proper estimate needs to be made considering the effort that has been put into all directions. We estimate the project size based on the number of Lines of Code(LOC), where, LOC is defined such that:

- Only source lines that are "delivered" as part of the product are included i.e., test drivers and other support software are excluded.
- Source Lines are created by the project staff and any code created by application generation is excluded.
- One LOC is defined as one logical line of code.
- LOC includes variable declarations as well.
- Comments are not counted in LOC.

The basic steps in this model are:

- Obtain an initial estimate of the development effort from the estimates of Kilo Lines of Code(KLOC).
- Determine the set of 15 multiplying factors from the different attributes of the project.
- Multiply these estimates to get the Effort Adjustment Factor(EAF).

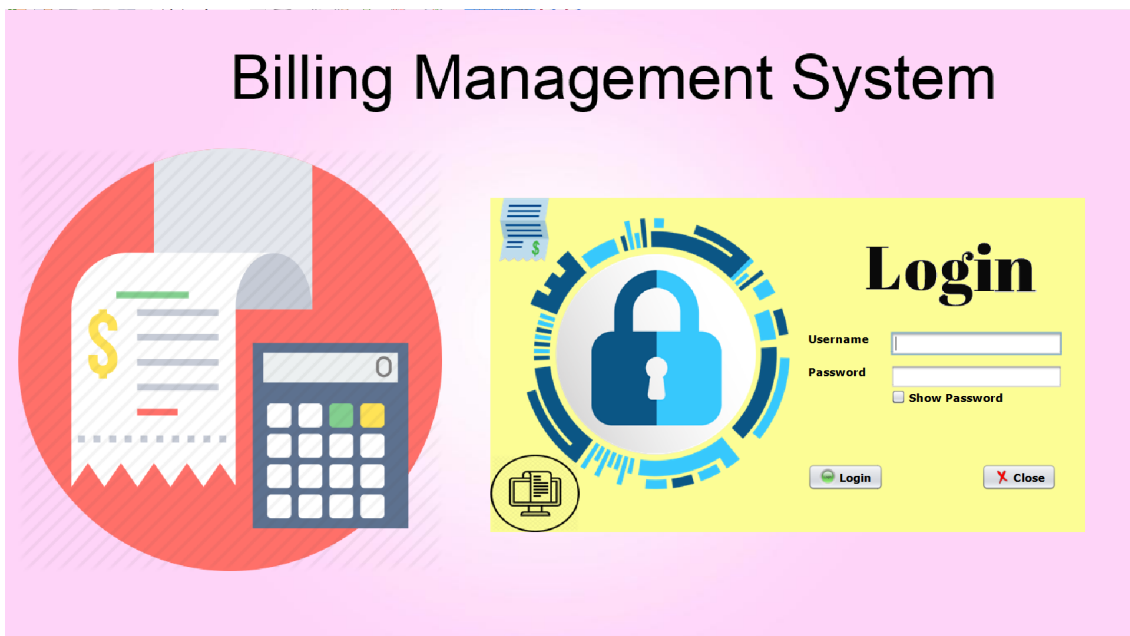
Intermediate COCOMO computes software development effort as a function of program size and a set of "cost drivers" that include subjective assessment of product, hardware, personnel and project attributes. Each of the 15 attributes receives a rating on a six-point scale that ranges from "very low" to "extra high" (in importance or value). An effort multiplier from the table below applies to the rating. The product of all effort multipliers results in an *effort adjustment factor (EAF)*. Typical values for EAF range from 0.9 to 1.4.

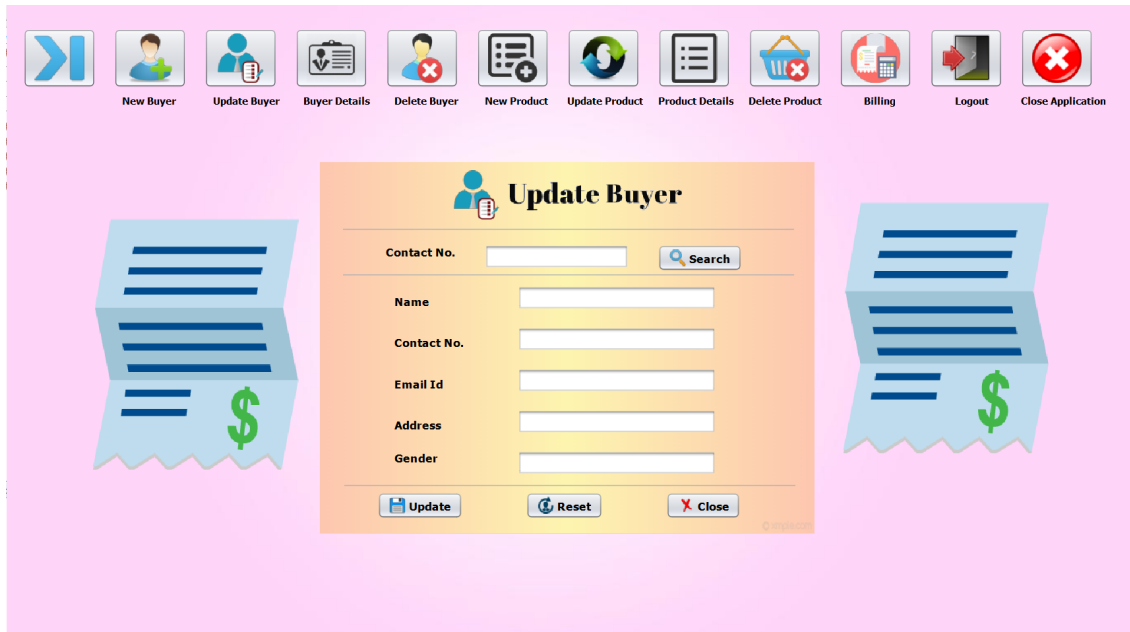
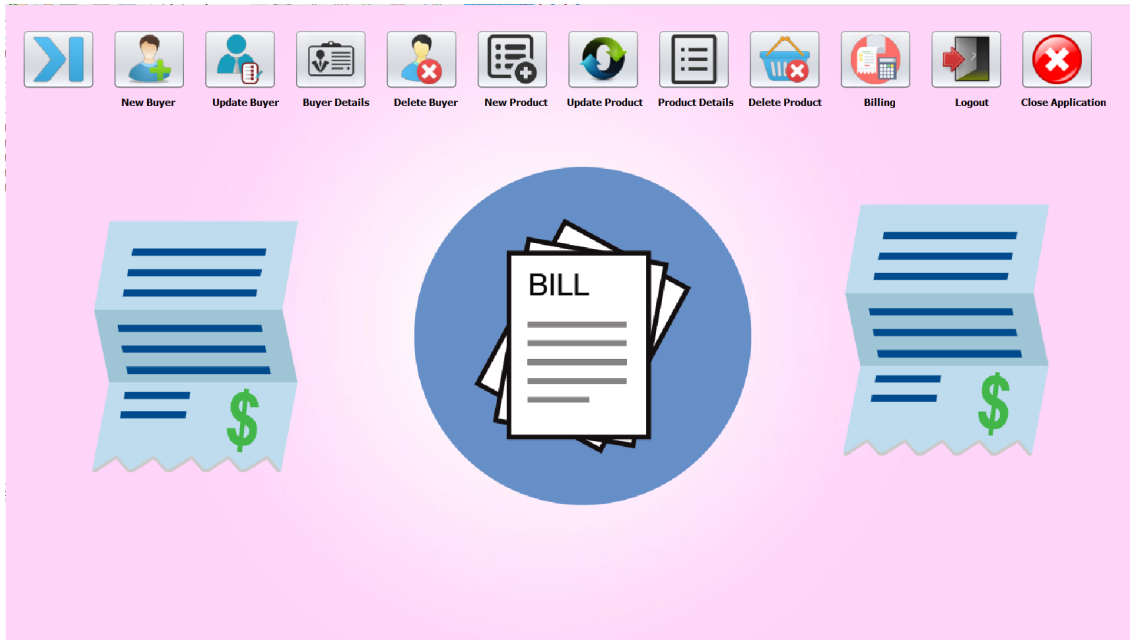
- ✓ The Intermediate Cocomo formula now takes the form:
- ✓ $E = a_i(\text{KLoC})^{b_i}(\text{EAF})$

- ✓ where E is the effort applied in person-months, **KLoC** is the estimated number of thousands of delivered lines of code for the project, and **EAF** is the factor calculated above.

However, in this project, there has been little to no production cost as we used NetBeans IDE as the Gui and java JFrame and Mysql database and didn't develop it for further use yet.

Reports:





Navigation icons: Home, New Buyer, Update Buyer, Buyer Details, Delete Buyer, New Product, Update Product, Product Details, Delete Product, Billing, Logout, Close Application.

Buyers Details

name	contactNo	email	address	gender
Suswagata	000012215	S@gmail.com	Silguri	Female
Rinki Kundu	0123456789	rinki@gmail.com	Sltj	Female
hfh	888	hddh	sjd	Male
Tua C	9434851372	tua@gmail.com	silguri	Female

Buttons: Print, Close

Navigation icons: Home, New Buyer, Update Buyer, Buyer Details, Delete Buyer, New Product, Update Product, Product Details, Delete Product, Billing, Logout, Close Application.

New Buyer

Name:

Contact No.:

Email id:

Address:

Gender:

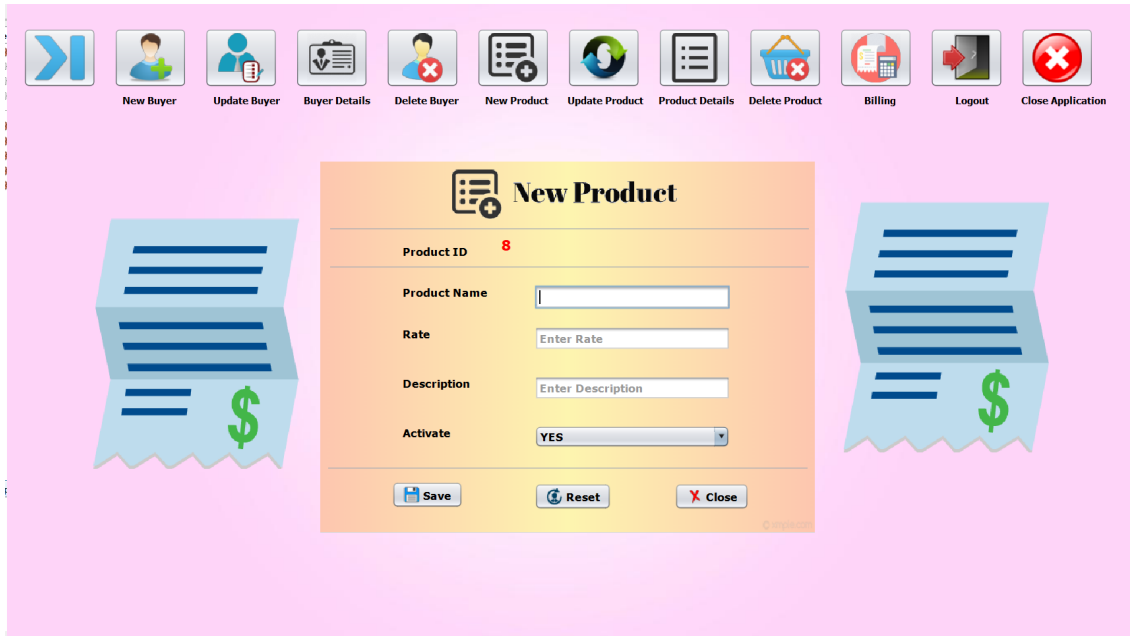
Buttons: Save, Reset, Close

Navigation icons: Home, New Buyer, Update Buyer, Buyer Details, Delete Buyer, New Product, Update Product, Product Details, Delete Product, Billing, Logout, Close Application.

Delete Buyer

Contact No.:

Name:



The interface features a top navigation bar with icons for: Home, New Buyer, Update Buyer, Buyer Details, Delete Buyer, New Product, Update Product, Product Details, Delete Product, Billing, Logout, and Close Application. The main content area is titled "New Product" and contains the following form fields:

- Product ID: 8
- Product Name:
- Rate:
- Description:
- Activate:

At the bottom of the form are three buttons: Save, Reset, and Close. The form is flanked by two decorative blue paper-like elements with a green dollar sign.



The interface features the same top navigation bar as the first screenshot. The main content area is titled "Update Product" and contains the following form fields:

- Product ID:
- Product Name:
- Rate:
- Description:
- Activate:

At the bottom of the form are three buttons: Update, Reset, and Close. The form is flanked by two decorative blue paper-like elements with a green dollar sign.

Product Details

pld	pName	rate	description	activate
3	Pen	3	Use and throw	NO
4	Tea	100	Tata	NO
5	Juice	10	Home Made	YES
6	Cake	50	Suger free	YES
7	jam	100	tasty	YES

Print Close

Delete Product

Product ID Search

Product Name

Rate

Description

Activate

Delete Reset Close

Billing Date: 04-07-2021
Time: 20:37:05

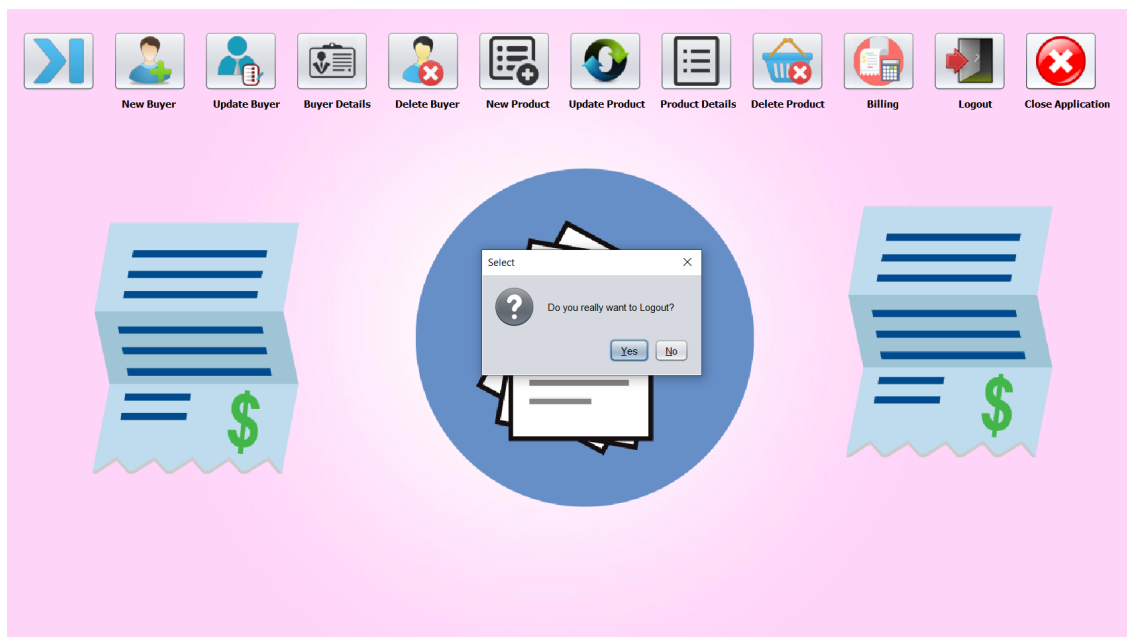
Buyer Details:
Name Contact No. Email Address

Product Details:
Product ID Product Name Rate Quantity Description
Activate

Name	Description	Rate	Quantity	Total

Calculation Details:
Total
Paid Amount
Return Amount

Select [X]
? Do you really want to Logout?



8. Charts

8.1 PERT Chart

PERT is an acronym that stands for Program Evaluation and Review Technique. It's a statistical tool that can be very useful when working on a project, as it analyzes and represents the project's tasks.

A PERT chart is a tool that can help project managers schedule, organize and coordinate tasks in their projects. It's a graphic representation of the timeline of a project, which gives project managers the tools they need to break down each of the project's tasks for analysis. There are milestones for the project indicated on the PERT chart by triangles. Circles represent the individual tasks and are connected by lines to show the duration of that task from start to finish. These are called nodes.

8.1.1 Steps to Implementing a PERT Chart

- Begin by identifying the project milestones and then break those down into individual tasks.
- Then figure out the sequence of the tasks.
- Make the PERT diagram.
- Do an estimate for each task and the time it will take to complete it.
- Calculate the critical path and identify any possible slack.
- Finally, the PERT chart is a living document that must be returned to and revived as needed when the project is in progress.

Pert chart is useful because of the following information:

The PERT chart is used by project managers to estimate the minimum amount of time that will be needed to close a project. This is done by examining the breakdown of the project and the connections there are between tasks, which also helps determine the amount of risk inherent in the project.

One of the purposes of a PERT chart is to help project managers get a handle on complex projects. The nature of the PERT chart and its breakdown structure help to take the complexity of a project and its many parts and visualize the dependencies between each step in the process.

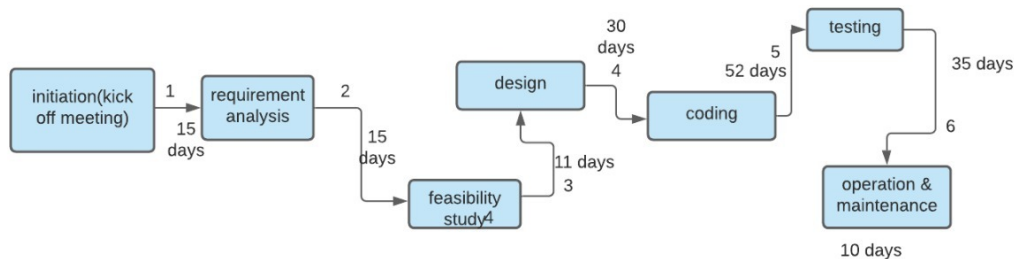


Fig: Pert Chart

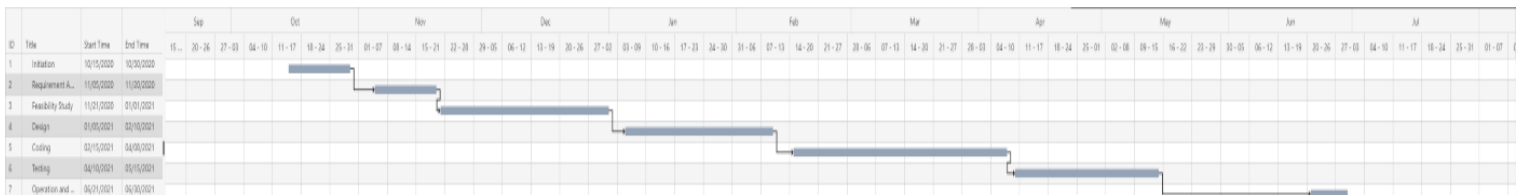


Fig: Gantt Chart

9. Limitations and Conclusion

9.1 Scope and Limitations

This billing system focuses on the development of an information system that will automate manual transactions in the supermarkets.

However, the study has focused on the following:

- The proposed automated system should generate reports of daily and monthly sales including reservation transactions of products.
- The user by using this software can add information about new buyers, can update buyers details, can show all details about buyers, can also delete details of a buyer if required.
- The system can add new products, can update details of products, show all details about products, and can also delete product details.
- It will generate receipts on every transaction inputted to the system.
- The software will display a view of calculations of every transaction.
- For security and privacy of the management, the billing system complies login users with username and password.

9.2 Conclusion

The primary significance of this project is to help the users to well organise every transaction and minimise their workload. The automated system will help the store to compete with other firms in terms of fast service. The system will help the users to minimize time and effort in tending to the receipt. It will provide faultless and accurate billing calculations. This project can be improvised further by sending an electronic bill via email to the customers and make the process eco friendly.

10. References

1. Paul Deitel and Harvey Deitel, "Java How to Program, Late Object", latest edition.
2. Y. Daniel Liang, "Introduction to Java Programming", latest edition.
3. Bruce Eckel, "Thinking in Java", 4th ed, 2007.
4. W3School JavaScript Tutorials, References and Examples @ <http://www.w3schools.com>.
5. Russell Dyer, "MySQL in a Nutshell", 2nd ed, O'reilly, 2008.
6. <https://nevonprojects.com/supermarket-billing-system/>
7. <https://github.com/jjilka/Billing-System-Java->
8. <https://projectworlds.in/java-projects-with-source-code/java-billing-system-project/>
9. <https://code-projects.org/supermarket-billing-system-in-java-with-source-code/>
10. <https://sites.google.com/site/bfcbillingsystem/methodology>

SILIGURI INSTITUTE OF TECHNOLOGY

CS 892

POTHoles AGGREGATOR

BY

CSE_PROJ_2021_17

Name of Students	Roll No.
1. Ricky Saha	11900117033
2. Purabi Das	11900117036
3. Kumar Jayant	11900117053
4. Dibya Jyoti Ghosh	11900117064

Under the Guidance

of

Prof. Jayshree Singha

Submitted to the Department of **Computer Science & Engineering** in partial fulfillment of the requirements for the award of the degree Bachelor of Technology in **Computer Science & Engineering**.

Year of Submission: 2021



Siliguri Institute of Technology

P.O. SUKNA, SILIGURI, DIST. DARJEELING, PIN: 734009

Tel: (0353)2778002/04, Fax: (0353) 2778003

DECLARATION

-

This is to certify that Report entitled "POTHLES AGGREGATOR" which is submitted by us in partial fulfillment of the requirement for the award of degree B.Tech. in **Computer Science Engineering** at **Siliguri Institute of Technology** under **Maulana Abul Kalam Azad University of Technology**, West Bengal. We took the help of other materials in our dissertation which have been properly acknowledged. This report has not been submitted to any other Institute for the award of any other degree.

Date: 5th July 2021

SN	Name of the Student	Roll No	Signature
1	Ricky Saha	11900117033	Ricky Saha 03/07/2021
2	Purabi Das	11900117036	Purabi Das 03/07/21
3	Kumar Jayant	11900117053	Kumar Jayant 03/07/2021
4	Dibya Jyoti Ghosh	11900117064	Dibya Jyoti Ghosh 03/07/21

CERTIFICATE

This is to certify that the project report entitled POTHLES AGGREGATOR submitted to **Department of Computer Science & Engineering of Siliguri Institute of Technology** in partial fulfilment of the requirement for the award of the degree of **Bachelor of Technology in Computer Science & Engineering** during the academic year **2019-20**, is a bonafide record of the project work carried out by them under my guidance and supervision.

Project Group Number: 17			
SN	Name of the students	Registration No	Roll No
1.	Ricky Saha	171190110063	11900117033
2.	Purabi Das	171190110060	11900117036
3.	Kumar Jayant	171190110043	11900117053
4.	Dibya Jyoti Ghosh	171190110032	11900117064

Signature of Project Guide

Name of the Guide: Prof. Jayshree Singha

Signature of the HOD

Department of Computer Science & Engineering

ACKNOWLEDGEMENT

Patience and preservation are the part & parcel to make fulfill any desired motto successful. Not only these two indispensable characters but also kind operation and restful help are always required with which one can be able to reach his ultimate goal after passing through a series of several incidents.

Likewise, we do have the pleasure to expose that we have completed our final year project on "POTHLES AGGREGATOR". So, at the very outset we deeply feel like expressing my indebtedness and gratitude to all concerned, unless who's help, valued suggestions, guidance and moral boosting the presence of the work of ours would not have been possible.

We express our heartfelt gratitude in deep humility to Mrs. Jayshree. Sinha (PROJECT MENTOR & FACULTY, CSE, SIT) who has provided us with the facilities in achieving the objective to prepare the project work and provide valuable and priceless suggestions which helped us at various stages of the project.

Signature of all the group members with date

1. Ricky Saha

Ricky Saha
03/07/2021

2. Purabi Das

Purabi Das
03/07/21

3. Kumar Jayant

Kumar Jayant
03/07/2021

4. Dibya Jyoti Ghosh

Dibya Jyoti Ghosh
03/07/21

Table of Contents

ABSTRACT.....	07
1.	
INTRODUCTION.....	08
2. SYSTEM	
ANALYSIS.....	09-17
2.1 Identification of Need.....	09
2.2 Preliminary Investigation.....	09
2.3 Feasibility Study.....	09
2.4 Project Planning.....	09
2.5 Project Scheduling.....	10
2.6 Software requirement specifications (SRS).....	10
2.7 Software Engineering Paradigm applied.....	10-16
2.8 Data model/Activity Diagrams.....	17
3. SYSTEM	
DESIGN.....	18-23
3.1 Modularization details.....	18
3.2 Data integrity and constraints.....	19
3.3 Database design/Procedural Design/Object Oriented Design.....	20-18
3.4 User Interface Design.....	19-23
4. CODING.....	24-31
5. TESTING.....	32-33

5.1 Testing techniques and test reports.....	32-33
5.2 Debugging and Code improvement.....	33
6. SYSTEM SECURITY MEASURES.....	34-35
6.1 Database/Data Security.....	34
6.2 Creation of User profiles and access rights.....	35
7. COST ESTIMATION OF THE PROJECT.....	36
8. PERT CHART.....	37
9. GANTT CHART.....	37
10. CONCLUSION AND RECOMMENDATIONS.....	38
REFERENCES.....	39

ABSTRACT

Road network has an indispensable **role** in economic activity. Without physical access to resources and markets, economic growth and development would not be possible. An effective road network system is, therefore, a fundamental element in enabling sustainable economic development. But one of the major problems this road network has are potholes.

Every year around 3,597 people die **due to potholes**. More than 30% of people die **due to potholes**. The Ministry of Road Transport and Highways provided figures that over 9300 deaths, 25000 injured in the last three years **due to potholes** and more than 25,000 people are getting injured **due to potholes**. The report of existence of pothole to higher authority and then higher authority call to road contractor for repairing job takes lot of time. By that time lots of mishap occur. So, our application solves this issue of reporting. As the citizen can simply use their smartphone to report to the higher authority directly in real time.

INTRODUCTION

This project acts as an important role in saving the lives of human beings and the overall economy of the country, which is also its main aim. We all use smartphone these days and we the citizen are the one who has to struggle with the pot holes. So, our application is based on Citizen first design. Citizen can now directly connect with the higher authority. And higher authority can act promptly. First the user has to install the application in their smartphone then simple first click a photo with the phone. Then after that they have open the app and browse to upload the pot hole image they captured. After that simply they have clicked the Get My Location button to fetch their geo location and address. And then they can mention their name and simply click on the SUBMIT button. After that the photo and the location data will be sent to the database which can directly access by the higher authority in real-time and work upon it. As the data will be stored in a centrally located database so the authority can analyze and work accordingly.

System Analysis

- **Identification of Need**

Every year around 3,597 people die **due to potholes**. More than 30% of people die **due to potholes**. The Ministry of Road Transport and Highways provided figures that over 9300 deaths, 25000 injured in the last three years **due to potholes** and more than 25,000 people are getting injured **due to potholes**. This app basically connects the Citizen with the Government authority.

- **Preliminary Investigation**

This android application allows us to access the potholes location, readily, scalable and centralized so the data can be easily access by the authority in real time whereas tracking the database is complicated when details are maintained manually. This makes the maintenance of schedule erroneous. Limitations of the Manual system: It is time consuming, it leads to error prone results, it consumes a lot of manpower to better results, it lacks data security, retrieval of data takes a lot of time, and the percentage of accuracy is less. Whereas in the proposed System all the above limitations can be avoided.

- **Feasibility Study**

This app is based on the Crowdsourcing data model collecting data from different data sources and presenting in a useful manner. So, it is highly feasible to create this and even scale up in future.

- **Project Planning**

Front End: XML & JAVA (Android Studio)

Backend: NoSQL (Google Firebase)

Software Requirements:

- Web Browser
- Android Studio
- Windows/Linux/Mac

Hardware Components:

- Processor – Intel i5
- Memory (RAM) – 8 GB
- Secondary – 30 GB
- Android device (Testing)

- **Project Scheduling**

This project is scheduled to be completed in a 3 months period of time which stretched from April to June. The first month was spent on the building of ideas, the second month was spent on the type of tools and framework needed for the implementation. And the final month was spent in execution of the project. This was the most crucial month as all the planning and scheduling were made in this month.

- **Software requirement specifications (SRS)**

This project was prepared to run on Android phones so the frontend was developed on Android SDK. And the coding part was done in Android Studio which was used as an IDE. This software was loaded on a Windows 10 PC to work on. For database implementation we used Google Firebase from a web browser.

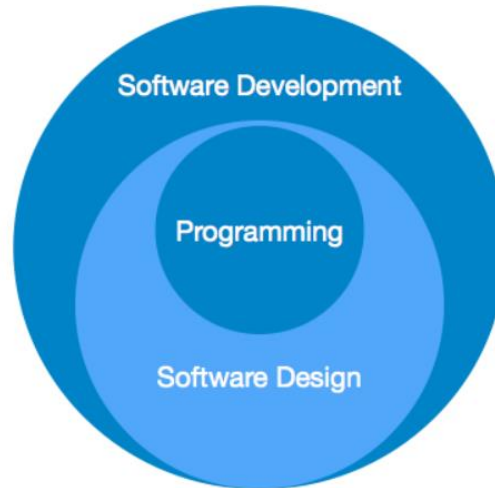
- **Software Engineering Paradigm applied**

Software is more than just a program code. A program is an executable code, which serves some computational purpose. Software is considered to be a collection of executable programming code, associated libraries and documentations. Software, when made for a specific requirement is called a software product.

Engineering on the other hand, is all about developing products, using well-defined, scientific principles and methods.



Software paradigms refer to the methods and steps, which are taken while designing the software. There are many methods proposed and are in work today, but we need to see where in the software engineering these paradigms stand. These can be combined into various categories, though each of them is contained in one another:

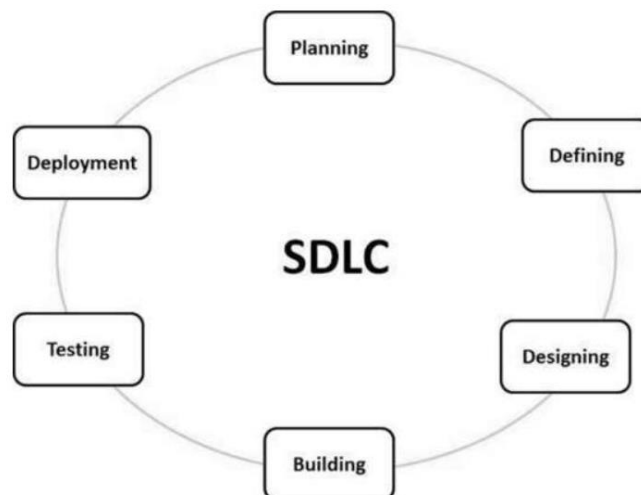


Programming paradigm is a subset of Software design paradigm which is further a subset of Software development paradigm.

Software Development Paradigm:

The software development paradigm helps developer to select a strategy to develop the software. A software development paradigm has its own set of tools, methods and procedures, which are expressed clearly and defines software development life cycle. Software Development Life Cycle (SDLC) is a process used by the software industry to design, develop and test high quality softwares. The SDLC aims to produce high-quality software that meets or exceeds customer expectations, reaches completion within times and cost estimates.

The following figure is a graphical representation of the various stages of a typical SDLC:



A typical Software Development Life Cycle consists of the following stages –

Stage 1: Planning and Requirement Analysis

Requirement analysis is the most important and fundamental stage in SDLC. It is performed by the senior members of the team with inputs from the customer, the sales department, market surveys and domain experts in the industry. This information is then used to plan the basic project approach and to conduct product feasibility study in the economical, operational and technical areas.

Planning for the quality assurance requirements and identification of the risks associated with the project is also done in the planning stage. The outcome of the technical feasibility study is to define the various technical approaches that can be followed to implement the project successfully with minimum risks.

Stage 2: Defining Requirements

Once the requirement analysis is done the next step is to clearly define and document the product requirements and get them approved from the customer or the market analysts. This is done through an SRS (Software Requirement Specification) document which consists of all the product requirements to be designed and developed during the project life cycle.

Stage 3: Designing the Product Architecture

SRS is the reference for product architects to come out with the best architecture for the product to be developed. Based on the requirements specified in SRS, usually more than one design approach for the product architecture is proposed and documented in a DDS - Design Document Specification.

This DDS is reviewed by all the important stakeholders and based on various parameters as risk assessment, product robustness, design modularity, budget and time constraints, the best design approach is selected for the product.

A design approach clearly defines all the architectural modules of the product along with its communication and data flow representation with the external and third-party modules (if any). The internal design of all the modules of the proposed architecture should be clearly defined with the minutest of the details in DDS.

Stage 4: Building or Developing the Product

In this stage of SDLC the actual development starts and the product is built. The programming code is generated as per DDS during this stage. If the design is performed in a detailed and organized manner, code generation can be accomplished without much hassle.

Developers must follow the coding guidelines defined by their organization and programming tools like compilers, interpreters, debuggers, etc. are used to generate the code. Different high-level programming languages such as C, C++, Pascal, Java and

PHP are used for coding. The programming language is chosen with respect to the type of software being developed.

Stage 5: Testing the Product

This stage is usually a subset of all the stages as in the modern SDLC models, the testing activities are mostly involved in all the stages of SDLC. However, this stage refers to the testing only stage of the product where product defects are reported, tracked, fixed and retested, until the product reaches the quality standards defined in the SRS.

Stage 6: Deployment in the Market and Maintenance

Once the product is tested and ready to be deployed it is released formally in the appropriate market. Sometimes product deployment happens in stages as per the business strategy of that organization. The product may first be released in a limited segment and tested in the real business environment (UAT- User acceptance testing).

Then based on the feedback, the product may be released as it is or with suggested enhancements in the targeting market segment. After the product is released in the market, its maintenance is done for the existing customer base.

There are various software development life cycle models defined and designed which are followed during the software development process. These models are also referred to as "Software Development Process Models". Each process model follows a Series of steps unique to its type to ensure success in the process of software development.

The software development paradigm used in this project is defined as follows:

The ***Waterfall Model*** was the first Process Model to be introduced. It is also referred to as a linear-sequential life cycle model. It is very simple to understand and use. In a waterfall model, each phase must be completed before the next phase can begin and there is no overlapping in the phases.

The Waterfall model is the earliest SDLC approach that was used for software development. The waterfall Model illustrates the software development process in a linear sequential flow. This means that any phase in the development process begins only if the previous phase is complete. In this waterfall model, the phases do not overlap.

Waterfall Model - Design

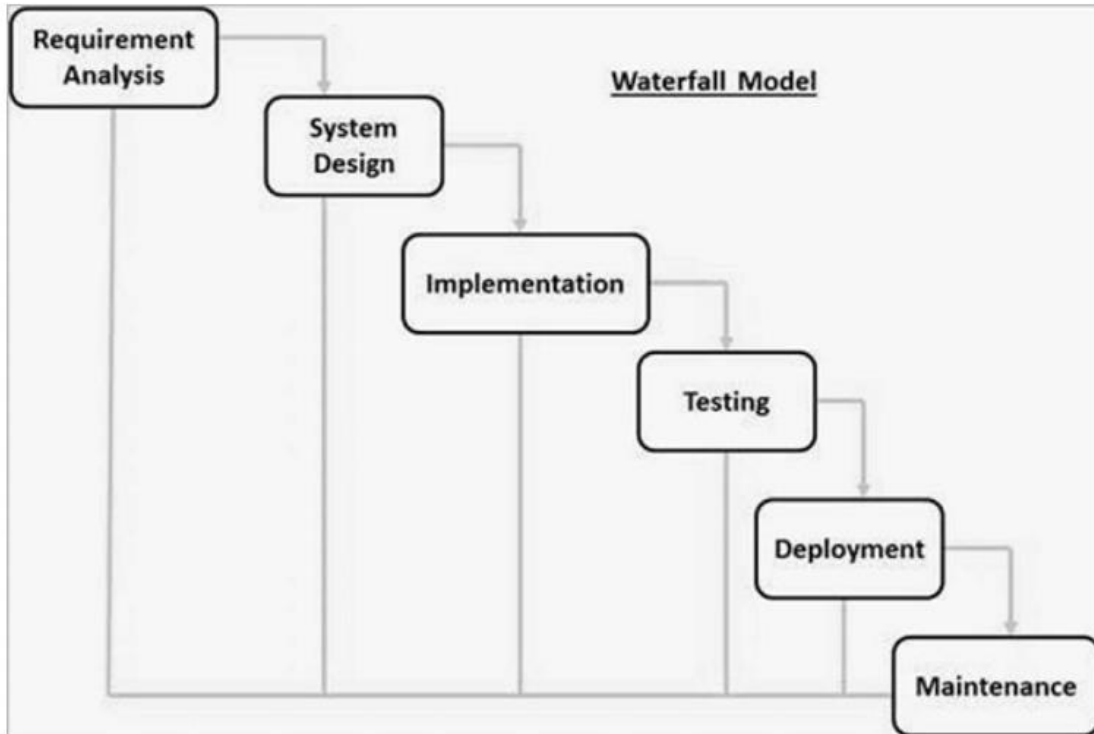
Waterfall approach was the first SDLC Model to be used widely in Software Engineering to ensure success of the project. In "The Waterfall" approach, the whole process of software development is divided into separate phases. In this Waterfall model, typically, the outcome of one phase acts as the input for the next phase sequentially.

The sequential phases in Waterfall model are –

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

All these phases are cascaded to each other in which progress is seen as flowing steadily downwards (like a waterfall) through the phases. The next phase is started only after the defined set of goals are achieved for the previous phase and it is signed off, so the name "Waterfall Model". In this model, phases do not overlap.

The following illustration is a representation of the different phases of the Waterfall Model:



Waterfall Model - Application

Every software developed is different and requires a suitable SDLC approach to be followed based on the internal and external factors. Some situations where the use of Waterfall model is most appropriate are –

- Requirements are very well documented, clear and fixed.
- Product definition is stable.
- Technology is understood and is not dynamic.
- There are no ambiguous requirements.
- Ample resources with required expertise are available to support the product.
- The project is short.

Waterfall Model - Advantages

The advantages of waterfall development are that it allows for departmentalization and control. A schedule can be set with deadlines for each stage of development and a product can proceed through the development process model phases one by one.

Development moves from concept, through design, implementation, testing, installation, troubleshooting, and ends up at operation and maintenance. Each phase of development proceeds in strict order.

Some of the major advantages of the Waterfall Model are as follows –

- Simple and easy to understand and use
- Easy to manage due to the rigidity of the model. Each phase has specific deliverables and a review process.
- Phases are processed and completed one at a time.
- Works well for smaller projects where requirements are very well understood.
- Clearly defined stages.
- Well understood milestones.
- Easy to arrange tasks.
- Process and results are well documented.

Waterfall Model - Disadvantages

The disadvantage of waterfall development is that it does not allow much reflection or revision. Once an application is in the testing stage, it is very difficult to go back and change something that was not well-documented or thought upon in the concept stage.

The major disadvantages of the Waterfall Model are as follows –

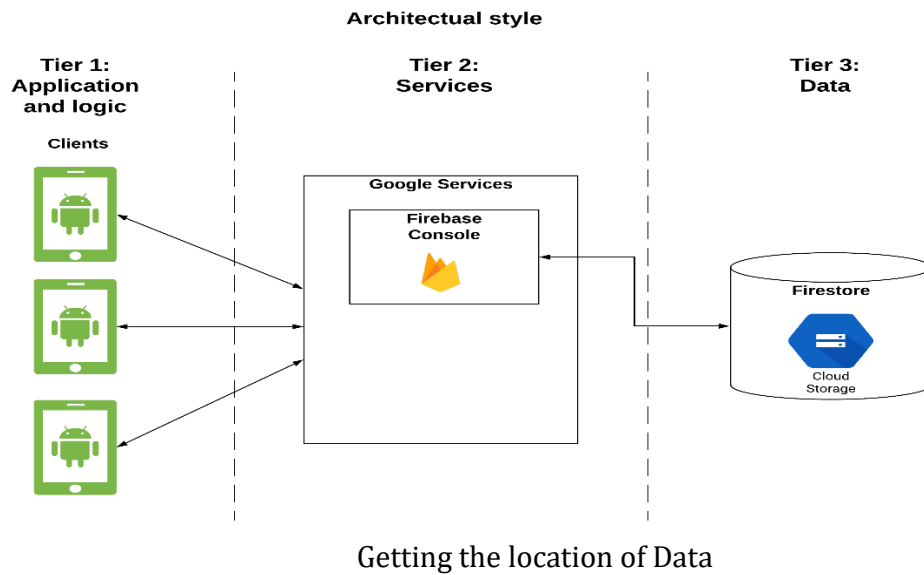
- No working software is produced until late during the life cycle.
- High amounts of risk and uncertainty.
- Not a good model for complex and object-oriented projects.
- Poor model for long and ongoing projects.

Despite these disadvantages, the waterfall model remains popular because it recognizes the inherent sequence in software engineering activities.

- **Data Model**



- **Activity Diagrams**

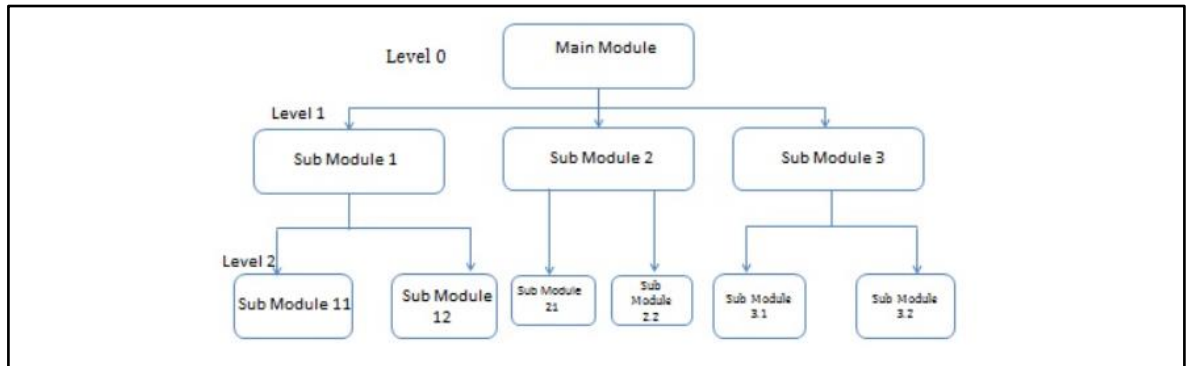


System Design

- **Modularization details**

Structured design partitions the program into small and independent modules. These are organized in a top down manner with the details shown at the bottom.

Thus, structured design uses an approach called **Modularization** or decomposition to minimize the complexity and to manage the problem by subdividing it into smaller segments.



Top-down strategy in Modularization

In our application **“Potholes Aggregator”**, the entire system has been divided into several modules. The main launching page is present in *MainActivity.java*, the code relating to the data loading of the user is present in *dataholder*. The code relating to the design is present in the *activity_main.xml*. The code relating to the color is *colors.xml* and the code relating to the language is in *strings.xml*.

Data integrity and constraints

Integrity constraints are a set of rules. It is used to maintain the quality of information. Integrity constraints ensure that the data insertion, updating, and other processes have to be performed in such a way that data integrity is not affected. Thus, integrity constraint is used to guard against accidental damage to the database.

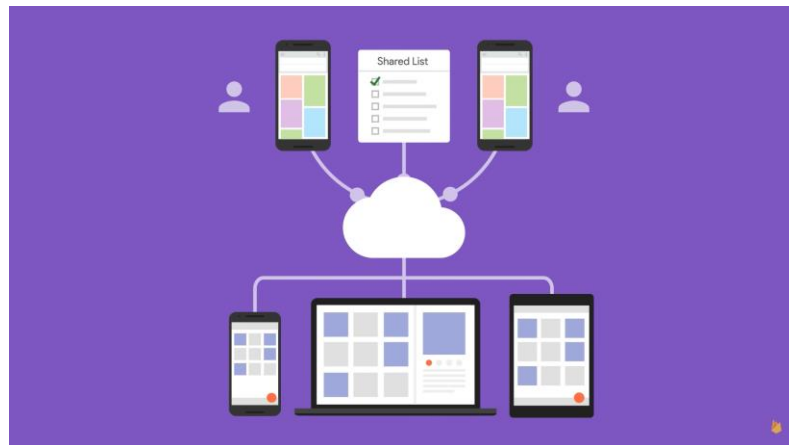
The **entity integrity** constraint states that primary key value can't be null.

The **key constraints** states Keys are the entity set that is used to identify an entity within its entity set uniquely. A primary key can contain a unique and null value in the relational table.

- **Database design/Procedural Design/Object Oriented Design**

The Firebase Realtime Database is a cloud-hosted database in which data is stored as JSON. The data is synchronized in real-time to every connected client. All of our clients share one Realtime Database instances and automatically receive updates with the newest data.

The Firebase Realtime Database is a NoSQL database from which we can store and sync the data between our users in real-time. It is a big JSON object which the developers can manage in real-time. By using a single API, the Firebase database provides the application with the current value of the data and updates to that data. Real-time syncing makes it easy for our users to access their data from any device, be it web or mobile.



Firestore: Realtime Database

- **User Interface Design**

Main Page :

11:30 • Pot Holes Aggregator

Upload the
pot hole image **BROWSE**

GET MY LOCATION

Latitude _____

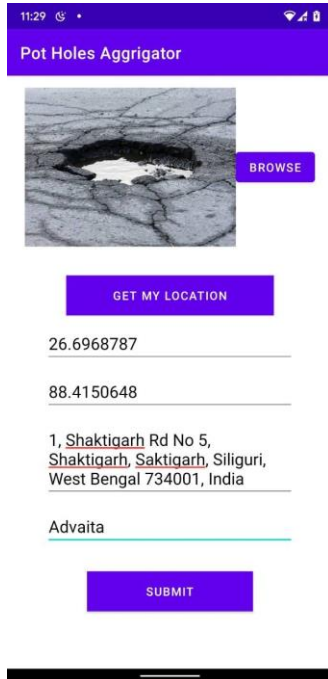
Longitude _____

Additional Info _____

Your Name _____

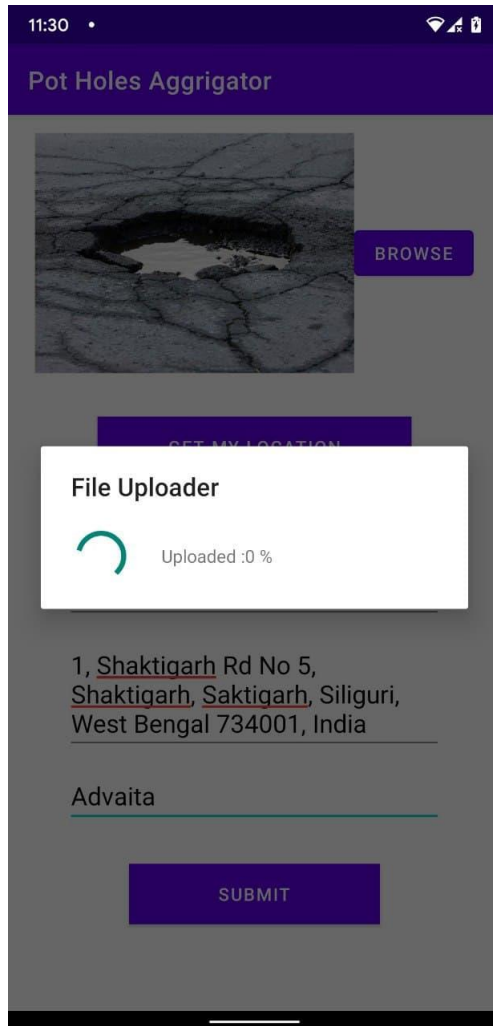
SUBMIT

The initial main page.



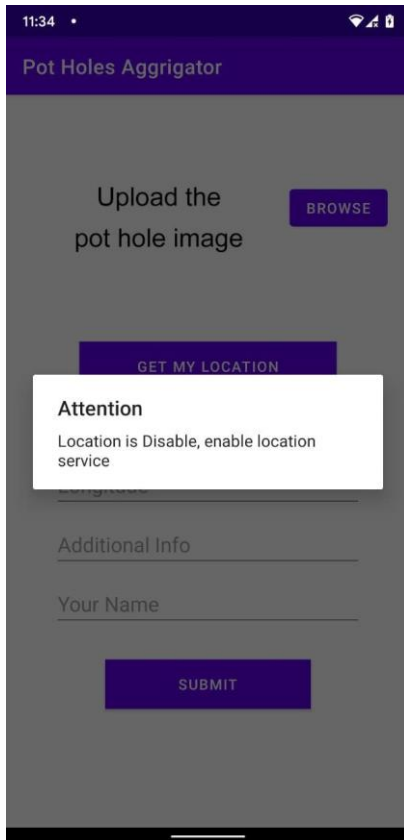
The user attach the pothole image and fetch the geo location of the area and enter his/her name and upload

Uploading Progress :



Showing the user the upload progress.

Location Permission Access Alert:



If the user doesn't switch on their GPS/navigation system and tries to fetch location data then alert will be shown to enable location service.

Coding

The code of the MainActivity page is as follows:

```
package com.potholes;

import android.Manifest;
import android.app.AlertDialog;
import android.app.ProgressDialog;
import android.content.Context;
import android.content.Intent;
import android.content.pm.PackageManager;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.location.Address;
import android.location.Geocoder;
import android.location.Location;
import android.location.LocationListener;
import android.location.LocationManager;
import android.net.Uri;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ImageView;
import android.widget.Toast;

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.app.ActivityCompat;
import androidx.core.content.ContextCompat;

import com.google.android.gms.tasks.OnSuccessListener;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.storage.FirebaseStorage;
import com.google.firebase.storage.OnProgressListener;
import com.google.firebase.storage.StorageReference;
import com.google.firebase.storage.UploadTask;
import com.karumi.dexter.Dexter;
import com.karumi.dexter.PermissionToken;
import com.karumi.dexter.listener.PermissionDeniedResponse;
import com.karumi.dexter.listener.PermissionGrantedResponse;
import com.karumi.dexter.listener.PermissionRequest;
import com.karumi.dexter.listener.single.PermissionListener;

import java.io.IOException;
import java.io.InputStream;
import java.util.ArrayList;
import java.util.List;
import java.util.Locale;
import java.util.Random;

public class MainActivity extends AppCompatActivity {
```

```

// Initializing the variable
EditText lat, lon, info, name;
Uri filepath;
ImageView img;
Button browse, submit, getgps;
Bitmap bitmap;
String latx, lonx;

private boolean gps_enable = false;
private boolean network_enable = false;

// To generate the address
Geocoder geocoder;
List<Address> myaddress;

public LocationManager locationManager;
public LocationListener locationListener = new
MyLocationListner();

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    img = (ImageView) findViewById(R.id.img);
    // Function to fetch location data
    getgps = (Button) findViewById(R.id.getlocation);
    locationManager = (LocationManager)
this.getSystemService(Context.LOCATION_SERVICE);
    getgps.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            getMygps();
        }
    });
    checkLocationPermission();
    submit = (Button) findViewById(R.id.upload);
    browse = (Button) findViewById(R.id.browse);
    // Function to browse the image file
    browse.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {

            Dexter.withActivity(MainActivity.this)

.withPermission(Manifest.permission.READ_EXTERNAL_STORAGE)
                .withListener(new PermissionListener() {
                    @Override
                    public void
onPermissionGranted(PermissionGrantedResponse
permissionGrantedResponse) {

                        Intent intent = new
Intent(Intent.ACTION_PICK);
                        intent.setType("image/*");

```



```

startActivityForResult(Intent.createChooser(intent, "Select Image
File"), 1);
    }

    @Override
    public void
onPermissionDenied(PermissionDeniedResponse permissionDeniedResponse)
{
    }

    @Override
    public void
onPermissionRationaleShouldBeShown(PermissionRequest
permissionRequest, PermissionToken permissionToken) {

permissionToken.continuePermissionRequest();
    }
    }.check();
    }
});
// Function to upload the data in firebase
submit.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        uploadtofirebase();
    }
});
}

// class created for the location manager
class MylocationListner implements LocationListener {

    @Override
    public void onLocationChanged(@NonNull Location location) {

        if (location != null) {
            locationManager.removeUpdates(locationListener);
            latx = "" + location.getLatitude();
            lonx = "" + location.getLongitude();

            lat.setText(latx);
            lon.setText(lonx);

            geocoder = new Geocoder(MainActivity.this,
Locale.getDefault());
            try {
                myaddress =
geocoder.getFromLocation(location.getLatitude(), location.getLongitude(
), 1);
            } catch (IOException e) {
                e.printStackTrace();
            }
            String address1 = myaddress.get(0).getAddressLine(0);

```

```

        info.setText(address1);
    }
}

@Override
public void onStatusChanged(String provider, int status,
Bundle extras) {

}

@Override
public void onProviderEnabled(@NonNull String provider) {

}

@Override
public void onProviderDisabled(@NonNull String provider) {

}
}
// Finction to check location & network permission is enable or
not.
public void getMygps() {
    try {

        gps_enable =
locationManager.isProviderEnabled(locationManager.GPS_PROVIDER);
    } catch (Exception ex) {
    }

    try {

        network_enable =
locationManager.isProviderEnabled(locationManager.GPS_PROVIDER);
    } catch (Exception ex) {
    }
    if (!gps_enable && !network_enable) {
        AlertDialog.Builder builder = new
AlertDialog.Builder(MainActivity.this);
        builder.setTitle("Attention");
        builder.setMessage("Location is Disable, enable location
service");
        builder.create().show();
    }
    if (gps_enable) {
        if (ActivityCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_FINE_LOCATION) !=
PackageManager.PERMISSION_GRANTED &&
ActivityCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_COARSE_LOCATION) !=
PackageManager.PERMISSION_GRANTED) {
            // TODO: Consider calling
            //     ActivityCompat#requestPermissions
            // here to request the missing permissions, and then
overriding
            //     public void onRequestPermissionsResult(int
requestCode, String[] permissions,

```

```

grantResults) // int[]
// to handle the case where the user grants the
permission. See the documentation
// for ActivityCompat#requestPermissions for more
details.
return;
}
if (ActivityCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_FINE_LOCATION) !=
PackageManager.PERMISSION_GRANTED &&
ActivityCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_COARSE_LOCATION) !=
PackageManager.PERMISSION_GRANTED) {
// TODO: Consider calling
// ActivityCompat#requestPermissions
// here to request the missing permissions, and then
overriding
// public void onRequestPermissionsResult(int
requestCode, String[] permissions,
// int[]
grantResults)
// to handle the case where the user grants the
permission. See the documentation
// for ActivityCompat#requestPermissions for more
details.
return;
}
locationManager.requestLocationUpdates(LocationManager.NETWORK_PROVIDE
R, 0, 0, locationManager);
}
if(network_enable){
locationManager.requestLocationUpdates(LocationManager.NETWORK_PROVIDE
R, 0, 0, locationManager);
}
}
private boolean checkLocationPermission(){
int location =
ContextCompat.checkSelfPermission(this,Manifest.permission.ACCESS_FINE
_LOCATION);
int location2 =
ContextCompat.checkSelfPermission(this,Manifest.permission.ACCESS_COAR
SE_LOCATION);
List<String>listPermission= new ArrayList<>();
if(location != PackageManager.PERMISSION_GRANTED){
listPermission.add(Manifest.permission.ACCESS_FINE_LOCATION);
}
if(location2 != PackageManager.PERMISSION_GRANTED){
listPermission.add(Manifest.permission.ACCESS_COARSE_LOCATION);
}
if(!listPermission.isEmpty()){
ActivityCompat.requestPermissions(this,listPermission.toArray(new

```

```

String[listPermission.size()],1 );
    }
    return true;
}
@Override
protected void onActivityResult(int requestCode, int resultCode,
@Nullable Intent data) {
    if(requestCode==1 && resultCode==RESULT_OK)
    {
        filepath=data.getData();
        try {
            InputStream
inputStream=getContentResolver().openInputStream(filepath);
            bitmap= BitmapFactory.decodeStream(inputStream);
            img.setImageBitmap(bitmap);

        }catch (Exception ex)
        {

        }
        super.onActivityResult(requestCode, resultCode, data);
    }
}

// Function to connect and upload the data in firebase
private void uploadtofirebase() {
    final ProgressDialog dialog=new ProgressDialog(this);
    dialog.setTitle("File Uploader");
    dialog.show();

    lat=(EditText)findViewById(R.id.t4);
    lon=(EditText)findViewById(R.id.t1);
    info=(EditText)findViewById(R.id.t2);
    name=(EditText)findViewById(R.id.t3);

    FirebaseStorage storage=FirebaseStorage.getInstance();
    final StorageReference
uploader=storage.getReference("Image1"+new Random().nextInt(50));

    uploader.putFile(filepath)
        .addOnSuccessListener(new
OnSuccessListener<UploadTask.TaskSnapshot>() {
            @Override
            public void onSuccess(UploadTask.TaskSnapshot
taskSnapshot) {

uploader.getDownloadUrl().addOnSuccessListener(new
OnSuccessListener<Uri>() {
                @Override
                public void onSuccess(Uri uri) {
                    dialog.dismiss();
                    FirebaseDatabase
db=FirebaseDatabase.getInstance();
                    DatabaseReference
root=db.getReference("users");

                    dataholder obj=new

```

```

dataholder(info.getText().toString(),lat.getText().toString(),lon.getT
ext().toString(),uri.toString());

root.child(name.getText().toString()).setValue(obj);

// TO reset the Edittext after
completion of the last upload
info.setText("0");
lat.setText("0");
lon.setText("0");
name.setText("0");

img.setImageResource(R.drawable.upload);

Toast.makeText(getApplicationContext(),"Uploaded",Toast.LENGTH_LONG).s
how();
    }
    });
}
})
// Function to progress status of the uploading of data
in firebase
    .addOnProgressListener(new
OnProgressListener<UploadTask.TaskSnapshot>() {
    @Override
    public void onProgress(UploadTask.TaskSnapshot
taskSnapshot) {

        float
percent=(100*taskSnapshot.getBytesTransferred())/taskSnapshot.getTotal
ByteCount();
        dialog.setMessage("Uploaded :"+(int)percent+"
%");

    }
});
}
}
}

```

The code of the data loading page is as follows:

```
package com.potholes;

public class dataholder {
    String info,lat,lon,pimage;
    // Constructor is created
    public dataholder(String info, String lat, String lon, String
pimage) {
        this.info = info;
        this.lat = lat;
        this.lon = lon;
        this.pimage = pimage;
    }
    // Getter & Setter method is created
    public String getInfo() {
        return info;
    }

    public void setInfo(String info) {
        this.info = info;
    }

    public String getLat() {
        return lat;
    }

    public void setLat(String lat) {
        this.lat = lat;
    }

    public String getLon() {
        return lon;
    }

    public void setLon(String lon) {
        this.lon = lon;
    }

    public String getPimage() {
        return pimage;
    }

    public void setPimage(String pimage) {
        this.pimage = pimage;
    }
}
```

Testing

3 Android phones of different manufactures were used for testing different test cases.

- Poco F1 (Running Android 10 & 11)
- Redmi Note 10 (Running Android 10)
- Pixel 2 (AVD) (Running Android 8)

Test Cases	Android Version	Remarks
1. UI (Homepage)	10	Ran smoothly without any hic-ups
	9	Ran smoothly without any hic-ups
	8	Ran smoothly without any hic-ups
2. Entering the values	10	No crashing or force stop while typing the data
	9	No crashing or force stop while typing the data
	8	No crashing or force stop while typing the data
3. Browsing the image file	10	No file size issue while uploading the image
	9	No file size issue while uploading the image
	8	No file size issue while uploading the image
4. Getting the location data	10	No issue while fetching the data from location service
	9	No issue while fetching the data from location service
	8	No issue while fetching the data from location service
5. UI (Night Mode)	10	Only issue was editext color was not visible properly
	9	Only issue was editext color was not visible properly
	8	N/A (OS doesn't support)

6. Location Permission Checking	10	No-issues
	9	No-issues
	8	No-issues

Debugging and Code improvement

The codes were debugged heavily with rigorous testing of every feature on different platforms. The debug method was mostly trial and error, emphasizing on the quality of the app and restricting every possible loophole that came to our notice.

At first much of the coding was hard-coded to get the app up and running with all the basic features. Gradually all the hard-coding were removed to make the app more flexible and scalable in future.

System Security measures (Implementation of security for the project developed)

- **Database/data security**

Security in Firebase is handled by setting the JSON like object inside the security rules. Security rules can be found when we click on Database inside the side menu and then RULES in the tab bar.

In this section, we will go through a couple of simple examples to show you how to secure the Firebase data.

Read and Write

The following code snippet defined inside the Firebase security rules will allow writing access to `/users/'$uid'/` for the authenticated user with the same uid, but everyone could read it.

Example

Let us consider the following example.

```
{
  "rules": {
    "users": {
      "$uid": {
        ".write": "$uid === auth.uid",
        ".read": true
      }
    }
  }
}
```

Validate

We can enforce data to string by using the following example.

Example

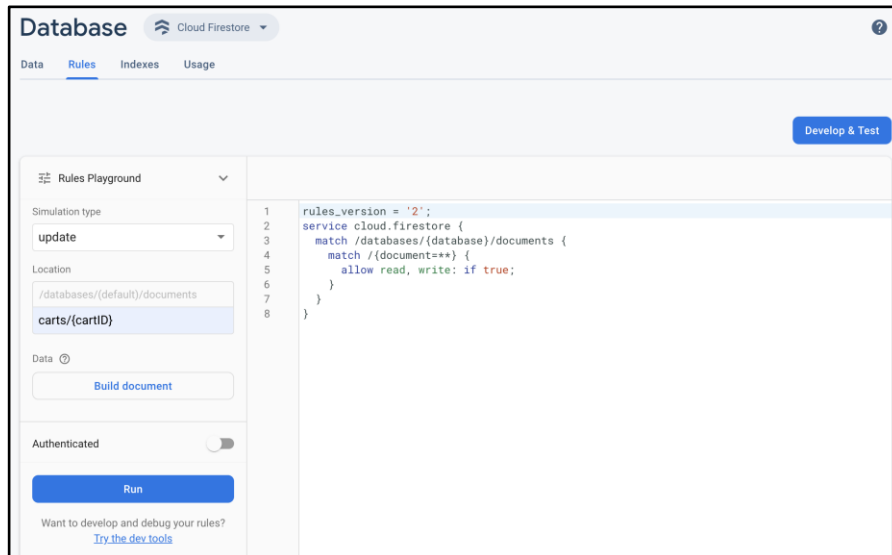
```
{
  "rules": {
    "foo": {
      ".validate": "newData.isString()"
    }
  }
}
```

- **Creation of User profiles and access rights**

In Google Firebase, authenticating your users and writing security rules, you can fully restrict read / write access to your Firebase data.

In a nutshell, Firebase security is enforced by server-side rules, that *you* author, and govern read or write access to given paths in your Firebase data tree.

Firebase security rules are JavaScript-like expressions: easy-to-write expressions that have access to the credentials for the connection, and the view of the Firebase data tree as it exists, along with pending changes on write.

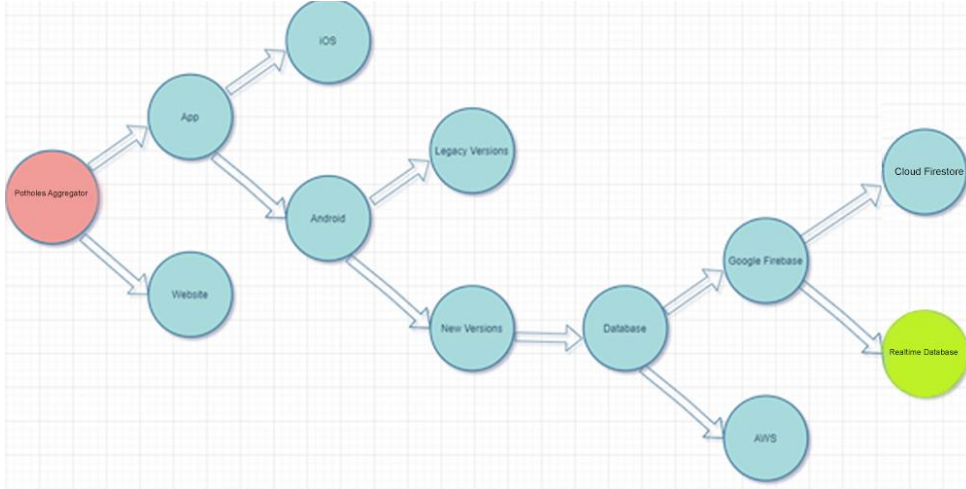


In most cases, your client-side logic, templates, assets, etc. will be static and public. What you're really looking to secure is user and application data, and this is where Firebase Authentication (whether using custom Firebase authentication tokens or Firebase Simple Login) comes in. Firebase Authentication is essentially token generation - taking confirmed, identifiable user data and passing it securely to Firebase so that it cannot be spoofed. This confirmed credential data is then made available in your security rules.

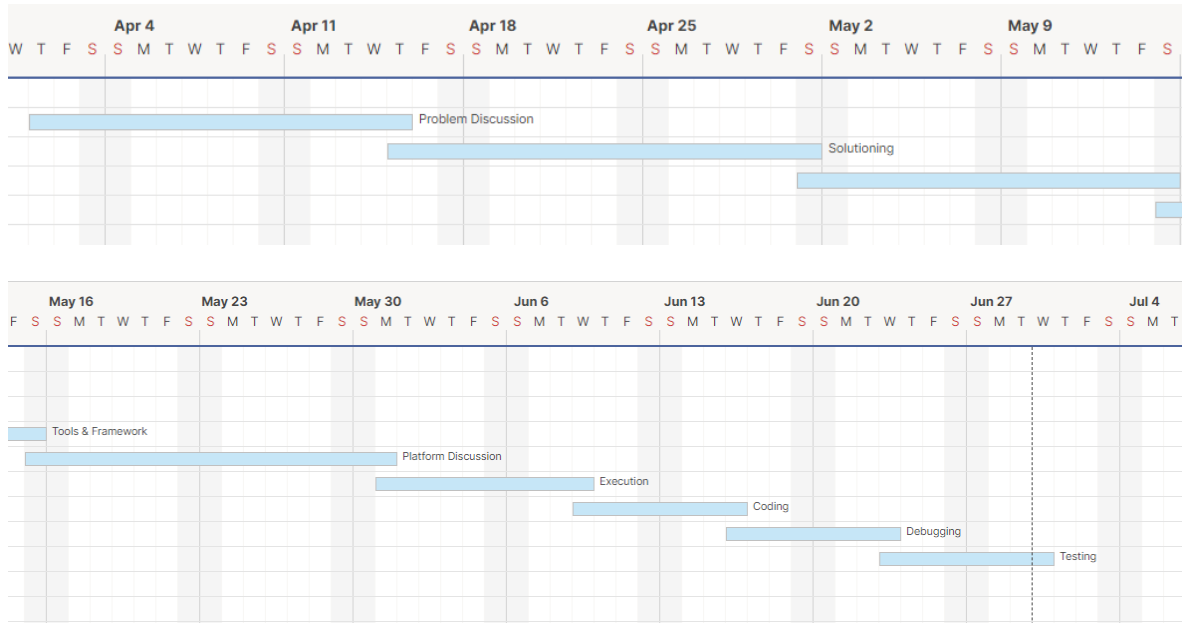
Cost Estimation of the Project

Item	Price
1. Google Firebase Subscription	140 per month
2. Hardware Resource	50000
3. Electricity + Internet Charges	200 (approx.)
TOTAL	50340

PERT Chart



Gantt Chart



Conclusion and Recommendations

In conclusion this application is aimed to connect the Citizen with the higher authority can now directly and transparently. Listing out some advantages, disadvantages and applications of the project: -

Advantages

- Citizen can directly connect with the authority seamless with a click of a button.
- The Road authority can transparently see the road condition from a central location.
- It is very cost effective and increase the overall economy by rapidly working on the road condition.
- Usage of this application will greatly reduce time in survey and corruption by the road contractor if an done so.

Disadvantages

- The android mobile user will not be able to insert or view details if the server goes down. Thus, there is a disadvantage of single point failure.

Applications

- This application can be used by any common person and in regional language too.
- The application can prove very beneficial to the Citizen, Government Authority and the road contractor.

Recommendations - This app can be made better in future by showing the other user's reported location so one can avoid that road.

Reference

1. Google.Inc, Official Firebase Documentation 2021
<https://firebase.google.com/docs>
2. Google.Inc, Official Android Documentation 2021
<https://developer.android.com/docs>
3. Oracle.Inc, Official Java Documentation 2021
<https://docs.oracle.com/en/java/>
4. Aditya, Symbiosis Law School., “Pothole deaths in India” Article
January 5, 2021
<https://blog.ipleaders.in/pothole-deaths-india/>
5. YouTube - Knowledge Extension - get current location android studio
- Apr 24, 2020
https://youtu.be/ByjxO66Y_1I
6. YouTube - Md Jamal - Firebase User Signup with Image - Jul 1, 2020
<https://youtu.be/9bZRMn6mZWc>
7. N. Naveen, 2 S.Mallesh Yadav, 3A.Sontosh Kumar 1Assistant Professor, K G Reddy College of Engineering and Technology, Hyderabad 2,3 “A Study on Potholes and Its Effects on Vehicular Traffic”. 2018 IJCRT | Volume 6, Issue 1 February 2018 | ISSN: 2320-2882
8. EUGENE Y. HUANG, Associate Professor of Civil Engineering, University of Illinois. A Study of Occurrence of Potholes and Washboards on Soil-Aggregate Roads, page 153, 2005.
9. Abhishek Kanoungo, Chitkara University – “Study of Causes of Potholes on Bituminous Roads – A Case study”. Journal of Civil Engineering and Environment Technology (JCEET) ISSN-2349-879X , Number 4 (April-June 2015), pp. 345-349At: JNU, NEW DELHI Volume: Volume 2.