
DROID TOOLS

INDEX


Objectives

The objective of the application is to provide users with a convenient and efficient tool for converting files to different types. By utilizing this app, users can easily transform their documents, images, or any other supported file formats into the desired format.

- **Menus :** The Primary objective of our app is to convert PDF file to text file , text file to PDF , image to text.

- **Data Uploading to Server :** Following few steps we can upload important file or notes into the Server.

- **Access Your File :** Which file we uploaded in the Server , that can be access from any device by login your account.

- **Quick Note :** There is a premium feature in our app Called Quick note. Where we can write anything virtually that is stored in our Database in real time.

CATEGORY

f

PROJECT

CATEGORY OF THE PROJECT WORK:

This is a android-based project.

Hardware and Software Used: -

Hardware: Computer or laptop with internet connection.

Processor: i5 10th gen or Ryzen 5 or upper

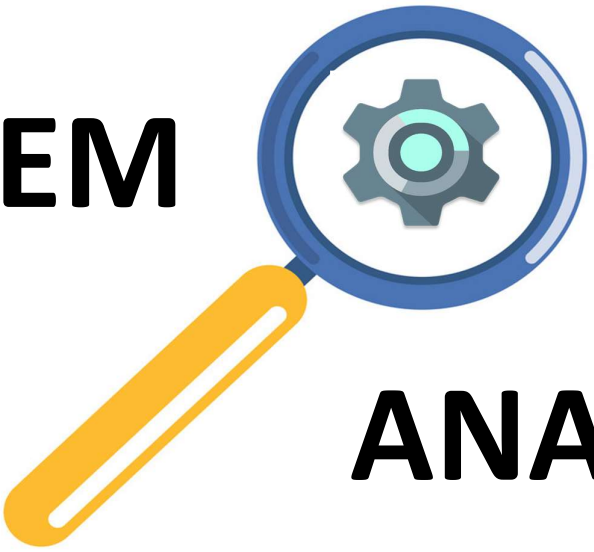
Software Tools: Android Studio, Chrome Browser

Language: JAVA and XML

Platform: Android

Database: Firebase

SYSTEM ANALYSIS



Identification of Need:

The basic elements of system analysis are as follows:

Intention: Determination of the objectives or goals. What do we intend to achieve and what is the purpose of our works.

Requirement Analysis: The emphasis in requirements analysis is on identifying what is needed from the system, not how the system will active its goals. This consists of two phases:

A. Problem analysis: The aim is to understand the problem and its context and the requirements of the new system that is to be developed.

B. Requirement Specification: Once the problem is analyzed and the essentials understood, the requirements must be specified in the requirement specification document. This document must specify all functional and performance requirements; the formats of inputs and outputs and all design constants that exists due to political, economic, environmental and security reasons.

Proper Format: The inputs should be available in proper format.

Economy: The data must be produced at allowable least cost.

File: The word file implies that files are used to store data. Most of the inputs necessary for the system may be historical

Preliminary Investigation:

The first step in the system development life-cycle is the preliminary investigations to determine the system. The purpose of the preliminary investigation is to evaluate project requests. It is not a design study nor does it include the collection of details to describe the business system in all respect. Rather, it is the collecting of information that helps committee member to evaluate the merits if project requests and make an information judgement about the feasibility of the proposed project. Analysts working on preliminary investigation should accomplish the following objectives: -

- Clarify and understand the project request.
- Determine size of project.
- Assets cost and benefits of alternative approaches.
- Determine the technical and operational feasibility of alternative approaches.

Verification and Validation

Verification: This is to ensure that the software being developed implements a specific function. Verification is done against the design document. It verifies that the software being developed implements all the functionality specified in the design document

Verification-"Are we building the product Right"

Validation : This is to ensure that the product being developed is matching with the customer requirements. Validation is done against the SRS (Software Requirement Specification) document. It verifies that the software being developed implements all the requirements specified in the SRS document.

Validation: ”Are we building the right product”

SRS

Software Requirements Specification

Definition

A condition of capability needed by a user to solve a problem or achieve an object. A conditioner a capability that must be met or processed by a system to satisfy a contract, standard, specification or other formally imposed document.

Three major parties interested in a new system are The Developer, The client, The user.

The problem while developing the software faced by the developers and client is that the developer usually does not understand the client's problem and the application area and the client usually does not understand the software development process.

The basic purpose of SRS documentation is to bridge this Communication gap. The Introduction of software requirement Specification states the goals and objectives of the software, describe In its context of computer based system. Actually, the information may be nothing more than the software scoop planning document.

This document in intended for

Developers : In order to be sure they are developing the right Project that fulfil requirements provided in this document.

SOFTWARE ENGINEERING
&
PARADIGMS

The software development strategy is referred as "Software Engineering" Paradigm. The Software development strategy consists of methods, tools, and procedures. There exist various software development strategies or process models. Software engineering is a Layered technology. Process defines the framework for a set of key processing are as that must be established for effective delivery of software engineering technology. The key process areas forms the basis of management control of software projects and establish the context in which technical methods are applied, work products are produced, milestones are established, quality is ensured and change is property managed. The "Software Engineering" methods provide a technical how to's for building software. Methods encompass a broad area of tasks that include requirement analysis, design, program construction, testing and support. Software Engineering relies on basic principles that govern each area of technology and include modeling activities and other descriptive techniques. Software engineering tools provide automated and semi-automated support for the process and methods when tools are integrated so that the information created by one tool can be used by another; a system for the support of software.

SOFTWARE DEVELOPMENT MODELS

- Classical Waterfall Model
- Iterative Waterfall Model
- Prototype Model
- Spiral Model

Classical Waterfall Model

In waterfall model, we proceed from one phase to the next in a sequential manner. Requirement Analysis, Design, Implementation, Testing, Deployment and Maintenance are the different sequential phases. That is once the requirement analysis is completed perfectly we proceed towards design phase. When the design is totally completed, the implementation of that design is performed by the coders. After completing the coding, the testing phase begins. The objective of testing is to uncover errors. The next is the deployment maintenance phase. Thus the waterfall model insists that proceed to the next phase only after completing previous phase perfectly. Water fall model is also known as classic life cycle model

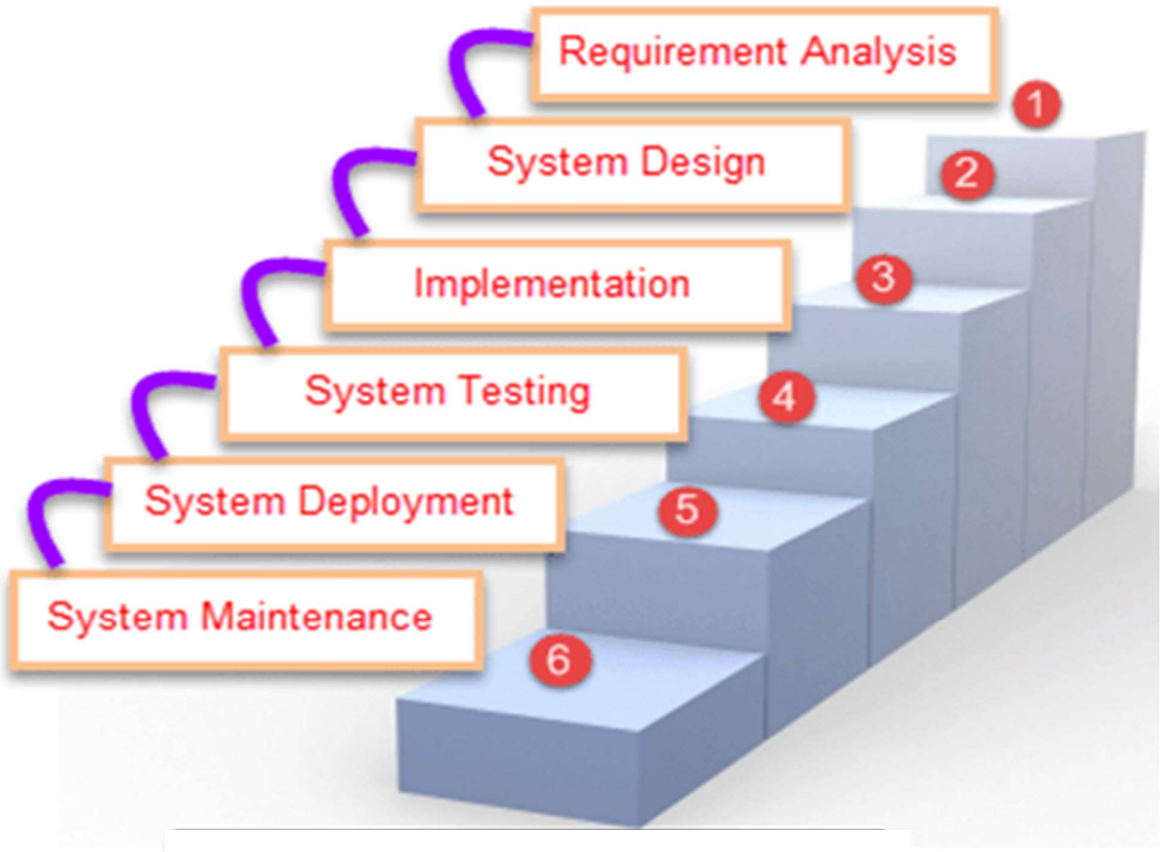


FIGURE: CLASSICAL WATERFALL MODEL

Iterative Water Model

The Iterative water fall model approach overcomes the problems associated with the waterfall model approach. If any difficulty or problem encounter in any phase may require going back to the previous

phase and performing the required modifications and proceeds sequentially. This backtracking allows modifying any corrections or modifications required in the previous phase.

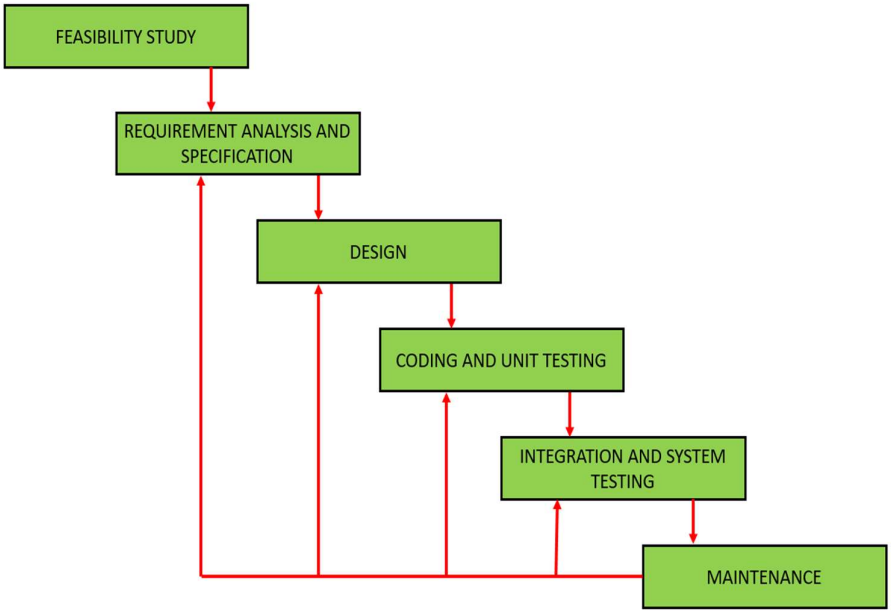


FIGURE: ITERATIVE MODEL

Prototype Model

After the requirement analysis a quick design is done. That is a working model of the intended software is constructed. This focuses on a representation of the software that will be visible to the user. This working model of the software is termed as a prototype. This prototype is evaluated by the customer to ensure that all the requirements are incorporated. The prototype will be modified according to the customer's suggestions. This process will be repeated until we get a fully functional working model that satisfies all user requirements. Ideally prototyping is a mechanism for identifying user requirements.

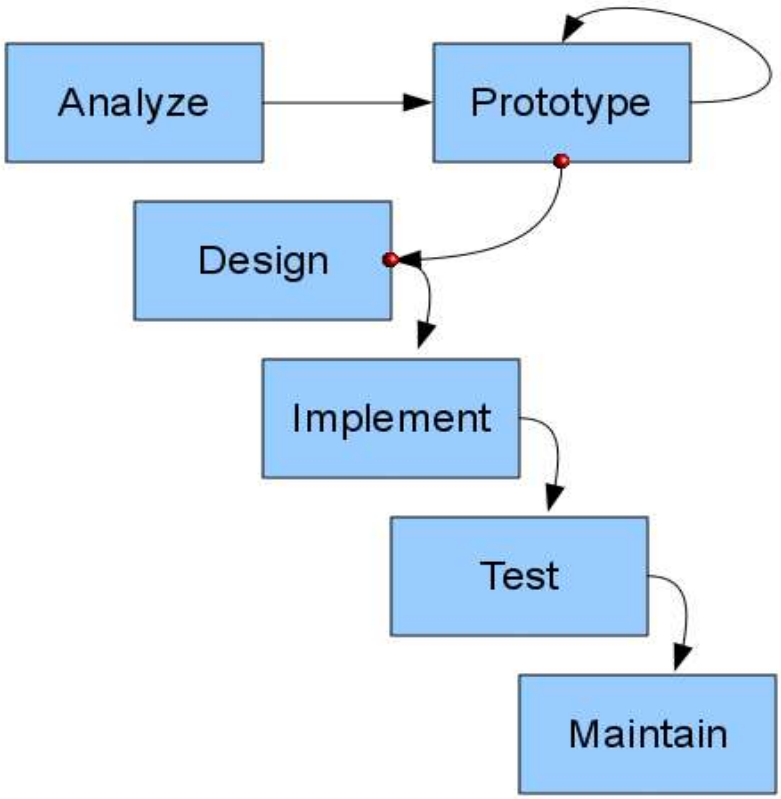


FIGURE: PROTOTYPE MODEL

SPIRAL MODEL

Spiral model is one of the most important Software Development Life Cycle models, which provides support for **Risk Handling**. In its diagrammatic representation, it looks like a spiral with many loops. The exact number of loops of the spiral is unknown and can vary from project to project. Each loop of the spiral is called a **Phase of the software development process.** The exact number of phases needed to develop the product can be varied by the project manager depending upon the project risks. As the project manager dynamically determines the number of phases, so the project manager has an important role to develop a product using the spiral model.

The Spiral Model is a software development life cycle (SDLC) model that provides a systematic and iterative approach to software development. It is based on the idea of a spiral, with each iteration of the spiral representing a complete software development cycle, from requirements gathering and analysis to design, implementation, testing, and maintenance.

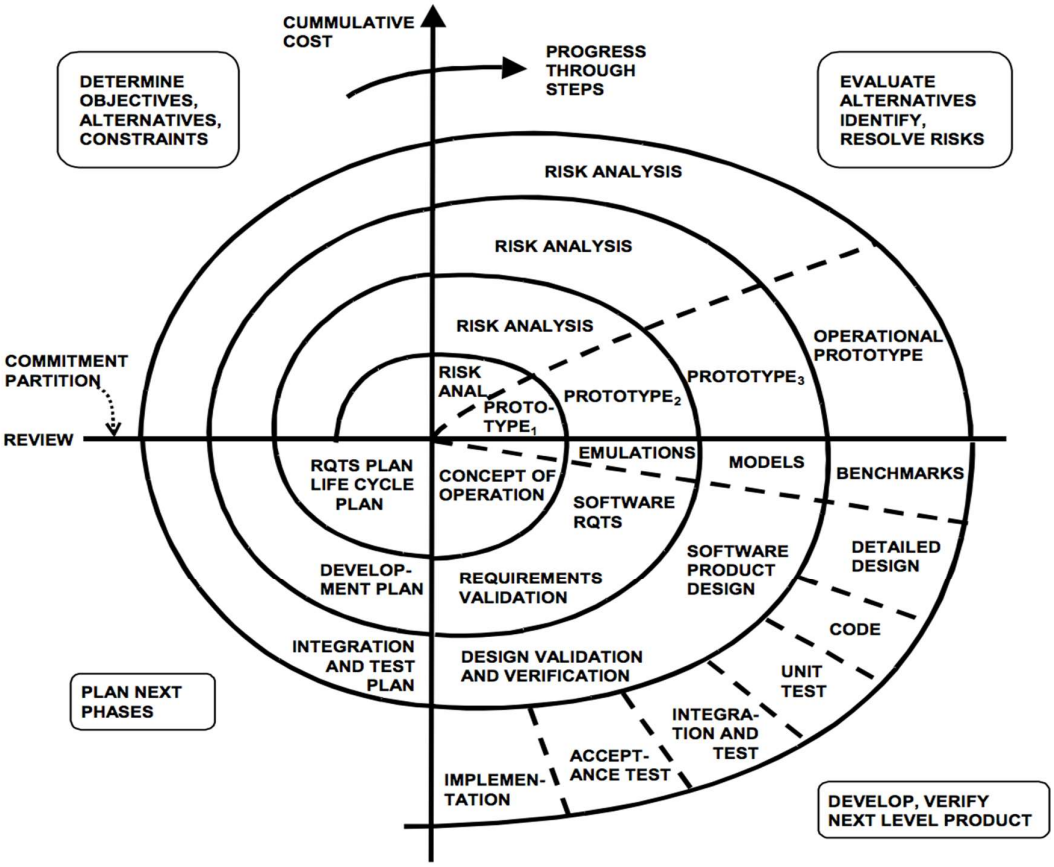
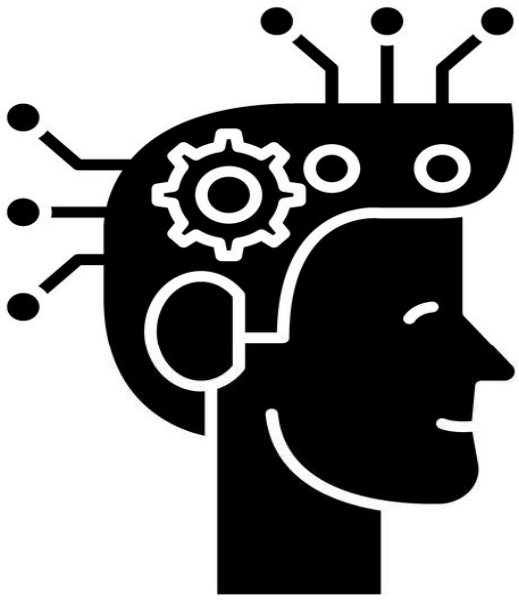


FIGURE: SPIRAL MODEL

SYSTEM PLANNING



System Planning

Project Planning:

The objective of software project planning is to provide a framework that enables the manager to make reasonable estimates of resources, cost, and schedule, These estimates are made within a limited time frame at the beginning of a software project and should be updated

regularly as the project progresses, In addition, estimates should attempt to define best case worst case scenarios that project outcomes can be bounded, The planning objective is achieved through a process of information discovery that leads to reasonable estimates.

Project Scheduling:

Project scheduling is an activity that distributes estimated effort across the planned project duration by allocating the effort to specific software engineering tasks. The number of basic principles guide the project scheduling is as follows:

Compartmentalization: The project must be compartmentalized into a number of manageable activities and tasks. To accomplish compartmentalization, both the product and the process are decomposed.

- **Interdependence:** The interdependence of each compartmentalized activity or task must be determined. Some tasks must occur in-sequence while others can occur in parallel. Some activities cannot commence until the work product produced by another is available. Other activities can occur independently.

-

Time allocation: Each task is scheduled must be allocated some number of work units. In addition, each task must be assigned a start date and a completion date.

Defined outcomes: Every task that is scheduled should have a defined outcome. For software projects the outcome is normally a work product or a part of a work product.

Defined milestones: Every tasks or group of tasks should be associated with a project milestone. A milestone is accomplished when one or more work products has been reviewed for quality and has been approved.

PERT CHART:

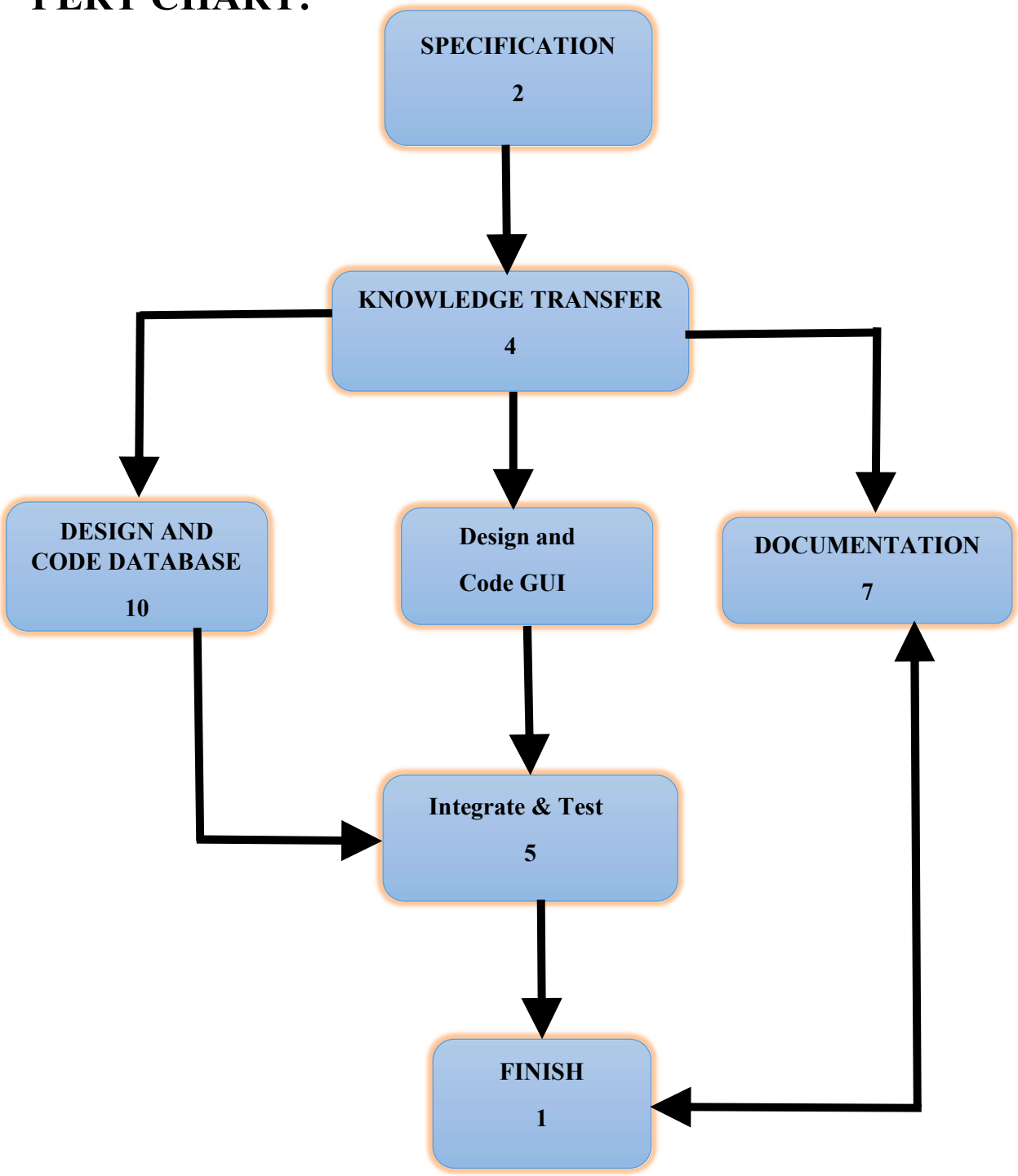


FIGURE: PERT CHART

Program evaluation and review technique (PERT) charts depict task, duration and dependency information. Each chart starts with an initiation node from which the first task or tasks originates. If multiple tasks begin at the same time, they are all started from the node or branch, or fork out from the starting point. Each task is represented by a line which states its name or other identifier, its duration, the number of people assigned to it and in some cases the initials of the personnel assigned. The other end of the task line is terminated by another node which identifies the start of another task, or the beginning of any slack time, that is, waiting time between tasks.

Each task is connected to its successor tasks in this manner forming a network of nodes and connecting lines. The chart is complete when all final tasks come together at the completion node. When slack time exists between the end of one task and the start of another, the usual method is to draw a broken or dotted line between the end of the first task and the start of the next dependent task.

A PERT chart may have multiple parallel or interconnecting networks of tasks. If the schedule project has milestones, checkpoints or review points (all of which are highly recommended in any project schedule); the PERT chart will note – that all tasks up to that point terminate at review node. It should be noted at this point that the project review, approvals, user reviews, and so forth all take time. This time should never be underestimated when drawing up the project plan. It is not unusual for a review to take one or two weeks. Obtaining management and user approvals may even take longer.

PERT charts are usually drawn on ruled paper with the Horizontal-axis indicating 'time-period divisions' in days, weeks, months and so on. Although it is possible to draw a PERT chart for the entire project, the usual practice is to break the plans in smaller, more meaning full parts. This is very helpful if the chart has to be redrawn for any reason, such as script or incorrectly estimated tasks.

Many PERT charts terminate at the major review points, such as the end of the analysis. Many organization include funding reviews in the project life cycle. Where this is the case, each chart terminates in the funding review node.

Funding reviews can affect a project in that they may either increase funding, in case more people have to make available, or they may decrease funding, in which case fewer people may be available. People affect the length of time it takes to complete the project..

GNATT CHART:

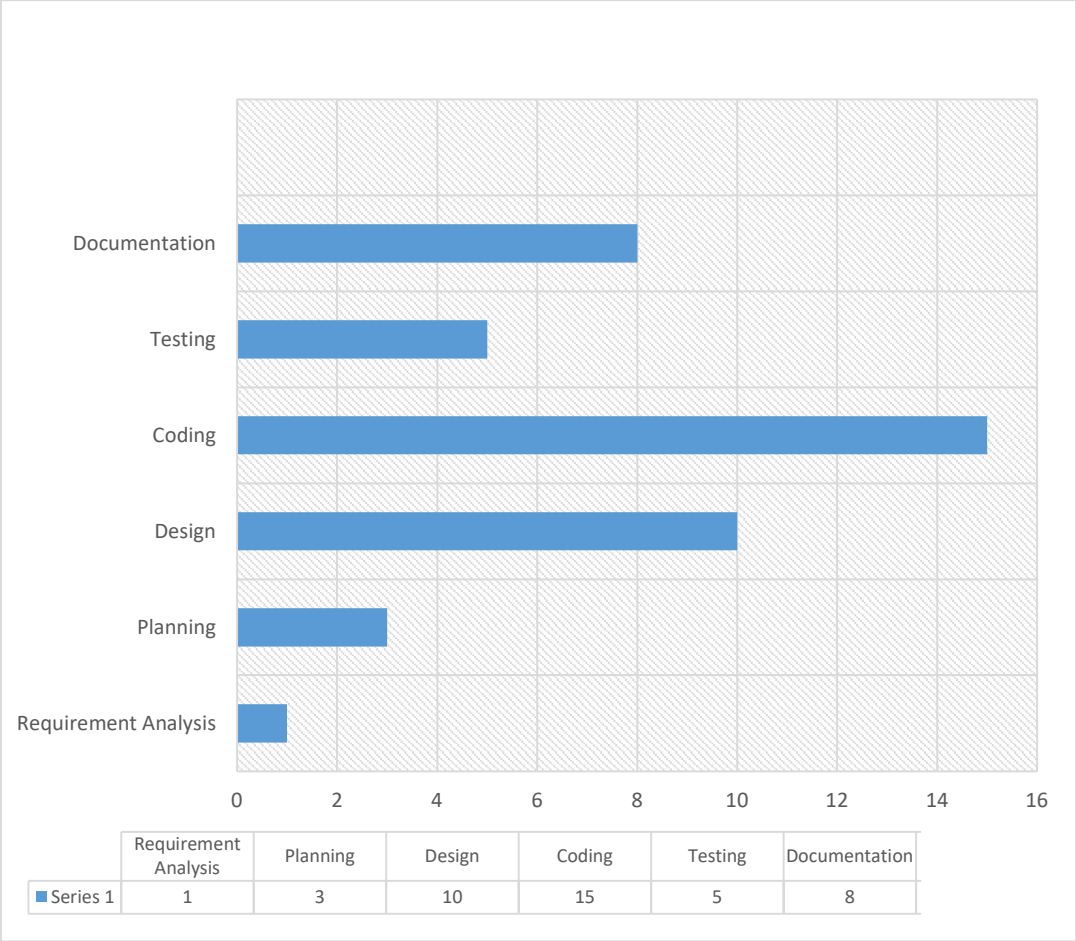


FIGURE: GNATT CHART

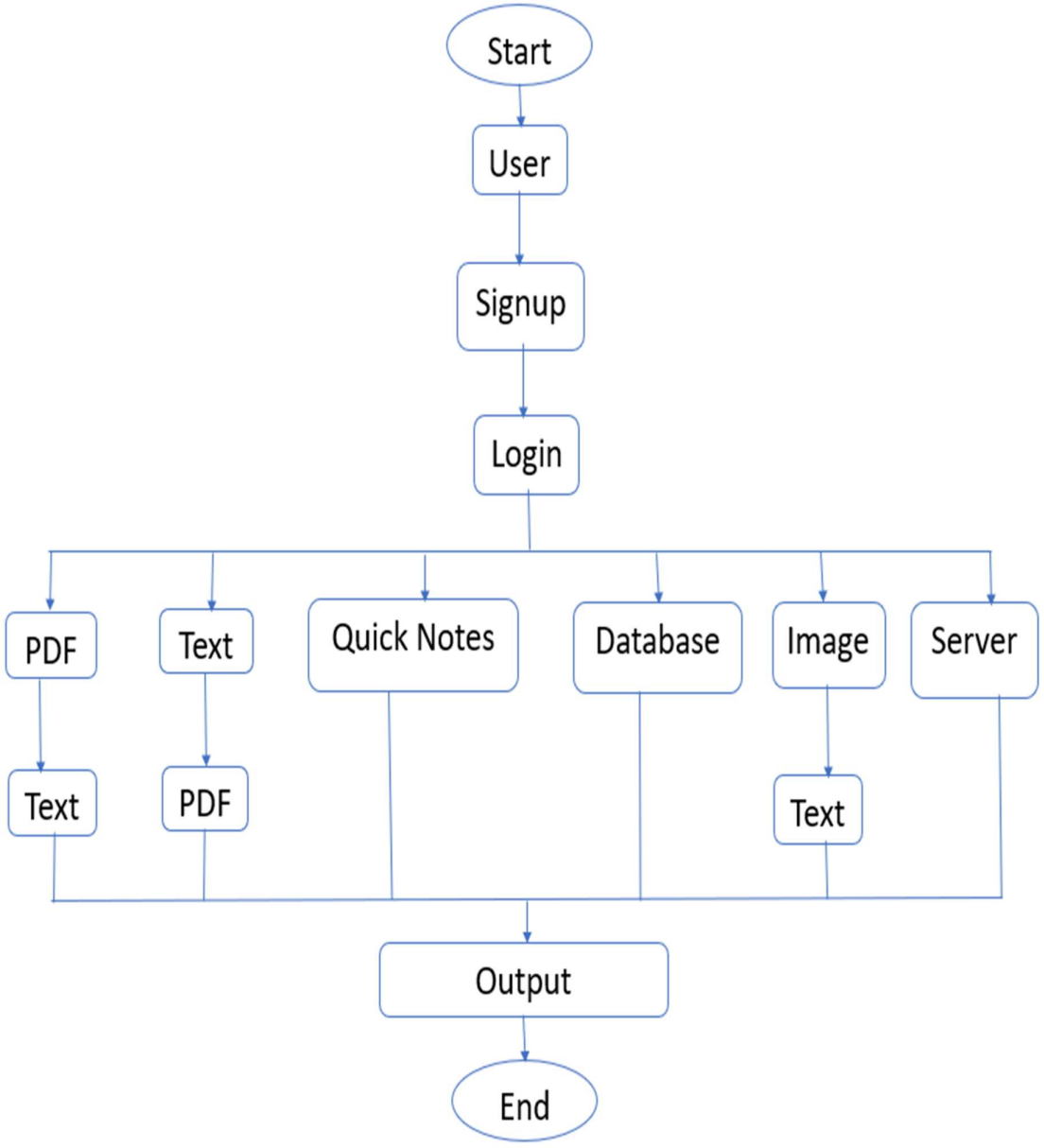
A Gantt chart is a matrix which lists on the vertical axis all the tasks to be performed. Each row contains a single task identification which usually consists of a number and name. The horizontal axis is headed by columns indicating estimated task duration, skill level needed to perform the task and the name of the person assigned to the tasks, followed by one column for each period in the project's duration. Each period may be expressed in hours, days, weeks, months and other time units. In some cases it may be necessary to label the period columns as Period1, Period2 and so on.

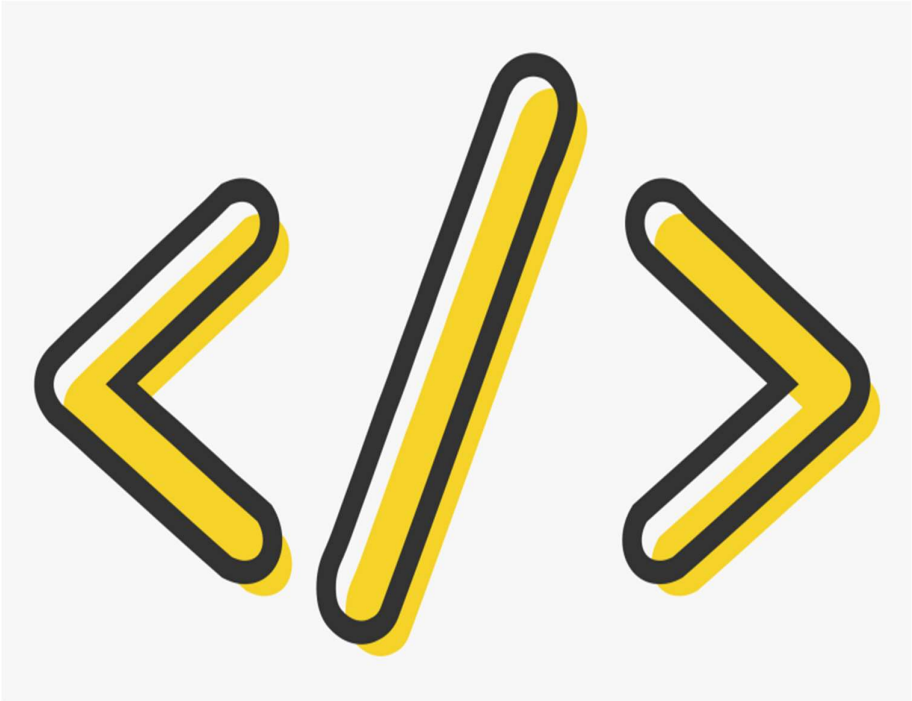
The graphics portion of the Gantt chart consists of a horizontal bar for each task connecting the period-start and period-ending columns. A set of markers is usually used to indicate estimated and actual start and end. Each bar on a separate line and the name of each person assigned to the task is on a separate line.

In many cases when this type of project plan is used, a blank row is left between tasks. When the project is under way, this row is used to indicate progress, indicated by a second bar which starts in the period column when the task is actually started and continues until the task is actually completed. Comparison between estimated start and end and actual start and end should indicate project status on a task-by-task basis.

Variants of this method include a lower chart which shows personnel allocations on a person-by-person basis. For this section the vertical axis contains the number of people assigned to the project and the columns indicating task duration are left blank, as is the column indicating person assigned. The graphics consists of the same bar notation as in the upper chart indicates that the person is working on a task. The value of this lower chart is evident when it shows slack time for the project personnel, that is, times when they are not actually working on any project.

FLOW CHART:


Start
User
Signup
Login
PDF
Text
Quick Notes
Database
Image
Server
Text
PDF
Text
Output
End



CODE

Login Page Code:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".LoginAndSignUp.LoginPage">

    <LinearLayout
        android:id="@+id/linearLayout"
        android:layout_width="match_parent"
        android:layout_height="240dp"
        android:background="@drawable/top_background"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        tools:ignore="MissingConstraints"
        android:orientation="horizontal" />

    <TextView
        android:id="@+id/textview"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="50sp"
```

```
android:textColor="@color/white"
        android:textStyle="bold"
        android:layout_marginTop="36dp"
        android:text="@string/app_name"
        android:fontFamily="sans-serif-condensed"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"/>
<androidx.cardview.widget.CardView
        android:id="@+id/cardView"
        android:layout_width="match_parent"
        android:layout_height="400dp"
        android:layout_marginStart="30dp"
        android:layout_marginEnd="30dp"
        android:layout_marginTop="50dp"
        android:background="@drawable/custom_edittext"
        app:cardCornerRadius="20dp"
        app:cardElevation="21dp"
        app:layout_constraintTop_toBottomOf="@+id/textview">
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_gravity="center_horizontal"
        android:background="@color/white"
         android:orientation="vertical"
         android:padding="24dp">
```

```
<TextView
        android:id="@+id/LginText"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/loginTxt"
        android:textAlignment="center"
        android:textColor="#448AFF"
        android:textSize="36sp"
        android:textStyle="bold" />

 <EditText
        android:id="@+id/mailid"
        android:layout_width="match_parent"
        android:layout_height="50dp"
        android:layout_marginTop="40dp"
        android:background="@drawable/custom_edittext"
        android:drawableStart="@drawable/mail_logo"
        android:drawablePadding="8dp"
        android:hint="@string/e_mail"
        android:importantForAutofill="no"
        android:inputType="textEmailAddress"
        android:padding="8dp"
        android:textColor="@color/black"
        android:textColorHighlight="@color/cardview_dark_background"
        android:textColorHint="#5C6BC0"
        tools:ignore="VisualLintTextFieldSize,TextContrastCheck" />
```

```
<EditText
    android:id="@+id/password"
    android:layout_width="match_parent"
    android:layout_height="50dp"
    android:layout_marginTop="40dp"
    android:autofillHints=""
    android:background="@drawable/custom_edittext"
    android:drawableStart="@drawable/password"
    android:drawablePadding="8dp"
    android:hint="@string/passwordTxt" android:inputType="textPassword"
    android:padding="8dp"
    android:textColor="@color/black"
    android:textColorHighlight="@color/cardview_dark_background"
    android:textColorHint="#5C6BC0"
    tools:ignore="VisualLintTextFieldSize,TextContrastCheck" />

<Button
    android:id="@+id/button"
    android:layout_width="match_parent"
    android:layout_height="60dp"
    android:layout_marginTop="30dp"
    android:backgroundTint="@color/purple"
    android:text="@string/loginTxt"
    android:textColor="#263238"
    android:textSize="18sp"
    app:cornerRadius="20dp"
    tools:ignore="VisualLintButtonSize,DuplicateSpeakableTextCheck" />
</LinearLayout>
```

```
</androidx.cardview.widget.CardView>

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical"
    android:gravity="center"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent">

     <TextView
        android:id="@+id/signupText"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginBottom="20dp"
        android:clickable="true"
        android:minHeight="48dp"
        android:padding="8dp"
        android:text="@string/not_yet_registered_signup_nowTxt"
        android:textAlignment="center"
        android:textColor="#5C6BC0"
        android:textSize="14sp" />

     <TextView
        android:id="@+id/forgotpass"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
```

```
android:layout_marginBottom="20dp"
        android:clickable="true"
        android:minHeight="48dp"
        android:padding="8dp"
        android:text="@string/forgotpass"
        android:textAlignment="center"
        android:textColor="#5C6BC0"
        android:textSize="14sp" />
    </LinearLayout>
```

Signup Page Code:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:gravity="center">
    <LinearLayout
        android:id="@+id/linearLayout"
        android:layout_width="match_parent"
        android:layout_height="240dp"
        android:background="@drawable/top_background"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        tools:ignore="MissingConstraints"
        android:orientation="horizontal" />

    <androidx.cardview.widget.CardView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_margin="30dp"
        android:background="@drawable/custom_edittext"
        android:elevation="20dp"
```

```
app:cardCornerRadius="30dp"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent">

  <LinearLayout
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:layout_gravity="center_horizontal"
      android:background="@color/white"
      android:orientation="vertical"
      android:padding="24dp">

      <TextView
        android:id="@+id/LginText"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/signupTxt"
        android:textAlignment="center"
        android:textColor="#448AFF"
        android:textSize="36sp"
        android:textStyle="bold" />
       <EditText
        android:id="@+id/mail"
        android:layout_width="match_parent"
        android:layout_height="50dp"
        android:layout_marginTop="40dp"
```

```
    android:background="@drawable/custom_edittext"
    android:drawableStart="@drawable/email"
    android:drawablePadding="8dp"android:hint="@string/e_mail"
    android:importantForAutofill="no"
    android:inputType="textEmailAddress"
    android:padding="8dp"
    android:textColor="@color/black"
    android:textColorHighlight="@color/cardview_dark_background"
    android:textColorHint="#5C6BC0"
    tools:ignore="VisualLintTextFieldSize,TextContrastCheck"
    />android:hint="@string/e_mail"
    android:importantForAutofill="no"
    android:inputType="textEmailAddress"
    android:padding="8dp"
    android:textColor="@color/black"
    android:textColorHighlight="@color/cardview_dark_background"
    android:textColorHint="#5C6BC0"
    tools:ignore="VisualLintTextFieldSize,TextContrastCheck" />
<EditText
    android:id="@+id/phonenumber"
    android:layout_width="match_parent"
    android:layout_height="50dp"
    android:layout_marginTop="40dp"
    android:background="@drawable/custom_edittext"
    android:drawableStart="@drawable/phone"
```

```
android:drawablePadding="8dp"
        android:hint="@string/phone_number"
        android:importantForAutofill="no"
        android:inputType="phone"
        android:padding="8dp"
        android:textColor="@color/black"
        android:textColorHighlight="@color/cardview_dark_background"
        android:textColorHint="#5C6BC0"
        tools:ignore="TextContrastCheck,TextFields,VisualLintTextFieldSize" />

    <EditText
        android:id="@+id/username"
        android:layout_width="match_parent"
        android:layout_height="50dp"
        android:layout_marginTop="40dp"
        android:background="@drawable/custom_edittext"
        android:drawableStart="@drawable/user"
        android:drawablePadding="8dp"
        android:hint="@string/usernameTxt"
        android:importantForAutofill="no"
        android:inputType="text"
        android:padding="8dp"
        android:textColor="@color/black"
        android:textColorHighlight="@color/cardview_dark_background"
        android:textColorHint="#5C6BC0"
        tools:ignore="VisualLintTextFieldSize,TextContrastCheck" />
```

```
<EditText
    android:id="@+id/password"
    android:layout_width="match_parent"
    android:layout_height="50dp"
    android:layout_marginTop="40dp"
    android:autofillHints=""
    android:background="@drawable/custom_edittext"
    android:drawableStart="@drawable/password"
    android:drawablePadding="8dp"
    android:hint="@string/passwordTxt"
    android:inputType="textPassword"
    android:padding="8dp"
    android:textColor="@color/black"
    android:textColorHighlight="@color/cardview_dark_background"
    android:textColorHint="#5C6BC0"
    tools:ignore="VisualLintTextFieldSize,TextContrastCheck" />

<EditText
    android:id="@+id/cPassword"
    android:layout_width="match_parent”
     android:layout_height="50dp"
    android:layout_marginTop="40dp"
    android:autofillHints=""
    android:background="@drawable/custom_edittext"
    android:drawableStart="@drawable/confirm_password"
    android:drawablePadding="8dp"
    android:hint="@string/confirm_password"
    android:inputType="textPassword"
```

```
            android:padding="8dp"
            android:textColor="@color/black"
            android:textColorHighlight="@color/cardview_dark_background"
            android:textColorHint="#5C6BC0"
            tools:ignore="DuplicateIds,TextContrastCheck,VisualLintTextFieldSize" />

        <Button
            android:id="@+id/signUpBtn"
            android:layout_width="match_parent"
            android:layout_height="60dp"
            android:layout_marginTop="30dp"
            android:backgroundTint="@color/purple"
            android:text="@string/signupTxt"
            android:textColor="#263238"
            android:textSize="18sp"
            app:cornerRadius="20dp"
            tools:ignore="VisualLintButtonSize,DuplicateSpeakableTextCheck" />

        </LinearLayout>

    </androidx.cardview.widget.CardView>

</androidx.constraintlayout.widget.ConstraintLayout>
```

Quick Note Input Page

```
package com.example.droidtools.Notepad;

import android.content.Intent;

import android.os.Bundle;

import android.view.Menu;

import android.view.MenuItem;

import android.view.View;

import android.widget.EditText;

import android.widget.ProgressBar;

import androidx.annotation.NonNull;

import androidx.appcompat.app.AppCompatActivity;


import com.example.droidtools.DatasOfUsersLogedIn.DataHolder;

import com.example.droidtools.R;

import com.google.firebase.FirebaseApp;

import com.google.firebase.database.DataSnapshot;

import com.google.firebase.database.DatabaseError;

import com.google.firebase.database.DatabaseReference;

import com.google.firebase.database.FirebaseDatabase;

import com.google.firebase.database.ValueEventListener;


import java.time.LocalDate;

import java.time.LocalTime;

import java.time.format.DateTimeFormatter;


public class QuickNoteInputPage extends AppCompatActivity {
```

```
  private EditText noteTitle, noteDesc;
    private FirebaseDatabase database;
    private DatabaseReference reference;
    private ProgressBar progressBar;
    private final String key = DataHolder.uidHolder;
    private String currentTimeString;
    private String currentDateString;
    private String date , time;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_quick_note_input_page);
        FirebaseApp.initializeApp(this);

        init();
    }

    private void init() {
        noteTitle = findViewById(R.id.note_title);
        noteDesc = findViewById(R.id.note_desc);
        progressBar= findViewById(R.id.loading);
        database = FirebaseDatabase.getInstance();
        reference = database.getReferenceFromUrl("https://droidtools-73721-default-
rtdb.firebaseio.com/");
    }
```

```
@Override
  public boolean onCreateOptionsMenu(Menu menu) {
     getMenuInflater().inflate(R.menu.menu_for_quick_notes, menu);
     return true;
 }


   @Override
   public boolean onOptionsItemSelected(@NonNull MenuItem item) {
      int item_id = item.getItemId();
      if (item_id == R.id.save) {
         progressBar.setVisibility(View.VISIBLE);


         getDate();
         getTime();


         String title = noteTitle.getText().toString();
         String description = noteDesc.getText().toString();


         reference.child("Users").addListenerForSingleValueEvent(new
ValueEventListener() {
            @Override
            public void onDataChange(@NonNull DataSnapshot snapshot) {

reference.child("Users").child(key).child("qnotes").child(key+currentDateString+currentT
imeString).child("title").setValue(title);

reference.child("Users").child(key).child("qnotes").child(key+currentDateString+currentT
imeString).child("description").setValue(description);
```

```
reference.child("Users").child(key).child("qnotes").child(key+currentDateString+currentTi
meString).child("date").setValue(date);

reference.child("Users").child(key).child("qnotes").child(key+currentDateString+currentTi
meString).child("time").setValue(time);
            }


            @Override
            public void onCancelled(@NonNull DatabaseError error) {


            }
        });
        if (title.isEmpty() && description.isEmpty()) {
            return false;
        }
        Note result=new Note(title,description,time,date);
        Intent intent=new Intent();
        intent.putExtra("Result",result);
        setResult(RESULT_OK,intent);
        finish();
    }
    return true;
  }
  private void getTime(){
 LocalTime currentTime = LocalTime.now();


    DateTimeFormatter formatter = DateTimeFormatter.ofPattern("HH:mm:ss");
```

```
        currentTimeString = currentTime.format(formatter);
        time = currentTimeString;
        currentTimeString=currentTimeString.replace(":","");
        currentTimeString=currentTimeString.replace(" ","");

        System.out.println("Current time: " + currentTimeString);
    }
    private void getDate(){
        LocalDate currentDate = LocalDate.now();

        DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy-MM-dd");

        currentDateString = currentDate.format(formatter);
        date = currentDateString;
        currentDateString=currentDateString.replace("-","");

        System.out.println("Current date: " + currentDateString);
    }
}
```

App Menu Page:

```
package com.example.droidtools.AppMenu;
import android.annotation.SuppressLint;
import android.app.AlertDialog;
import android.content.Intent;
import android.os.Bundle;
import android.widget.LinearLayout;

import androidx.appcompat.app.AppCompatActivity;

import com.example.droidtools.DatasOfUsersLogedIn.DataHolder;
import com.example.droidtools.FileUploadOnServer.UploadFiles;
import com.example.droidtools.ImageToTextTools.ImageToText;
import com.example.droidtools.LoginAndSignUp.LogOutApp;
import com.example.droidtools.LoginAndSignUp.LoginPage;
import com.example.droidtools.Notepad.NotePadStartInterface;
import com.example.droidtools.PdfToTextTools.PdfToText;
import com.example.droidtools.R;
import com.example.droidtools.ShowUplodedDataFromDatabase.FetchFileFromDB;
import com.example.droidtools.TextToPDFTools.TextToPdf;
import com.example.droidtools.UserDashboard.UserDashBoardSettings;
import com.example.droidtools.UserDashboard.UserDetailsShow;
import com.google.android.material.bottomnavigation.BottomNavigationView;

import java.util.Objects;
public class AppMenuPage extends AppCompatActivity {
    private LinearLayout
```

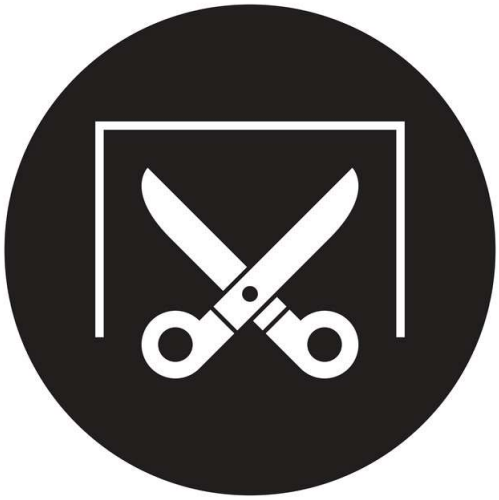
```
quickNoteBtn,txtToPdf,imgtotxt,uploadToServerFile,pdfToTxt,openDbBtn;
    private void init(){
        quickNoteBtn = (LinearLayout) findViewById(R.id.quiNoteBtn);
        imgtotxt = (LinearLayout) findViewById(R.id.imtotxt);
        uploadToServerFile = (LinearLayout) findViewById(R.id.uploadToServerFile);
        txtToPdf = (LinearLayout) findViewById(R.id.txtToPdf);
        pdfToTxt = (LinearLayout) findViewById(R.id.pdfToTxt);
        openDbBtn = (LinearLayout) findViewById(R.id.openDbBtn);
    }

    @SuppressLint("NonConstantResourceId")
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_app_menu_page);
        Objects.requireNonNull(getSupportActionBar()).hide();
        init();

        String key = DataHolder.uidHolder;
        if (key == null || key.equals("null")){
            Intent intent = new Intent(AppMenuPage.this, LoginPage.class)
             startActivity(intent);
        }
quickNoteBtn.setOnClickListener(view -> {
            Intent intent = new Intent(AppMenuPage.this, NotePadStartInterface.class);
            startActivity(intent);
        });
;
```

```
imgtotxt.setOnClickListener(view -> {
        Intent intent = new Intent(AppMenuPage.this, ImageToText.class);
        startActivity(intent);
    });


    uploadToServerFile.setOnClickListener(view -> {
        Intent intent = new Intent(AppMenuPage.this, UploadFiles.class);
        startActivity(intent);
    });

    txtToPdf.setOnClickListener(view -> {
        Intent intent = new Intent(AppMenuPage.this, TextToPdf.class);
        startActivity(intent);
    });

    pdfToTxt.setOnClickListener(view -> {
        Intent intent = new Intent(AppMenuPage.this, PdfToText.class);
        startActivity(intent);
    });

    openDbBtn.setOnClickListener(view -> {
        Intent intent = new Intent(AppMenuPage.this, FetchFileFromDB.class);
        startActivity(intent);
    });
        BottomNavigationView navigationView = (BottomNavigationView)
findViewById(R.id.navbarmain);
    navigationView.setSelectedItemId(R.id.mainmenu);
```

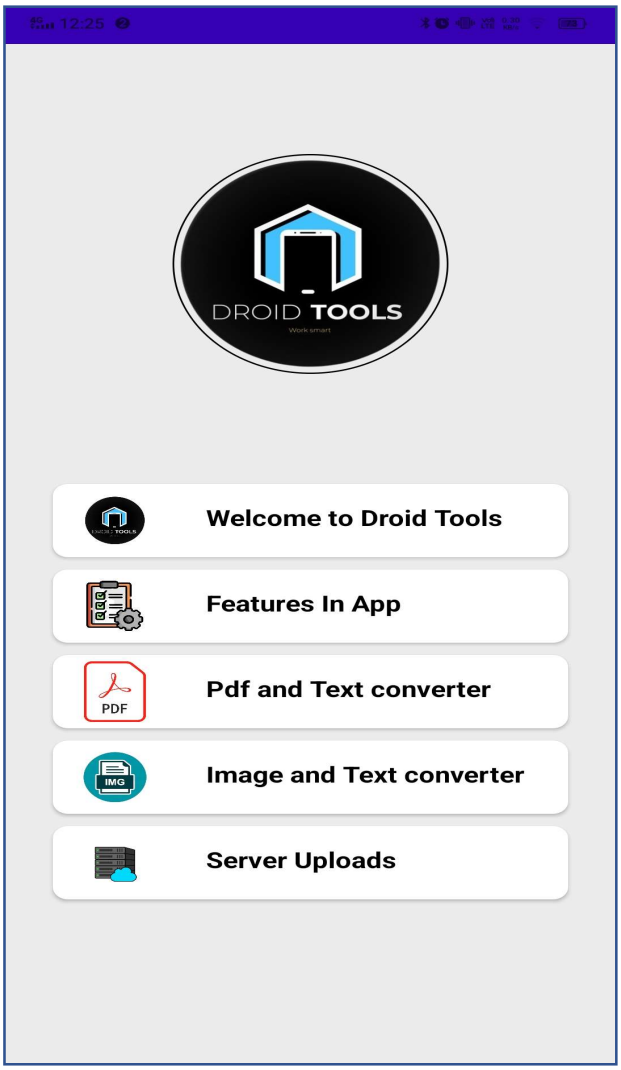
```
//noinspection deprecation
    navigationView.setOnNavigationItemSelectedListener(item -> {
        switch (item.getItemId()){
            case R.id.home:
                Intent intent0 = new Intent(AppMenuPage.this, HomeDashboard.class);
                startActivity(intent0);
overridePendingTransition(0,0);
                return true;
            case R.id.profile:
                Intent intent1 = new Intent(AppMenuPage.this, UserDetailsShow.class);
                startActivity(intent1);
                overridePendingTransition(0,0);
                return true;
            case R.id.mainmenu:
                Intent intent2 = new Intent(AppMenuPage.this, AppMenuPage.class);
                startActivity(intent2);
                overridePendingTransition(0,0);
                return true;
            case R.id.settings:
                Intent intent3 = new Intent(AppMenuPage.this, UserDashBoardSettings.class);
                startActivity(intent3);
                overridePendingTransition(0,0);
                break;
case R.id.logout:
                LogOutApp logout = new LogOutApp();
                new AlertDialog.Builder(this).setMessage("Are you sure to logout?")
                        .setCancelable(false)
```

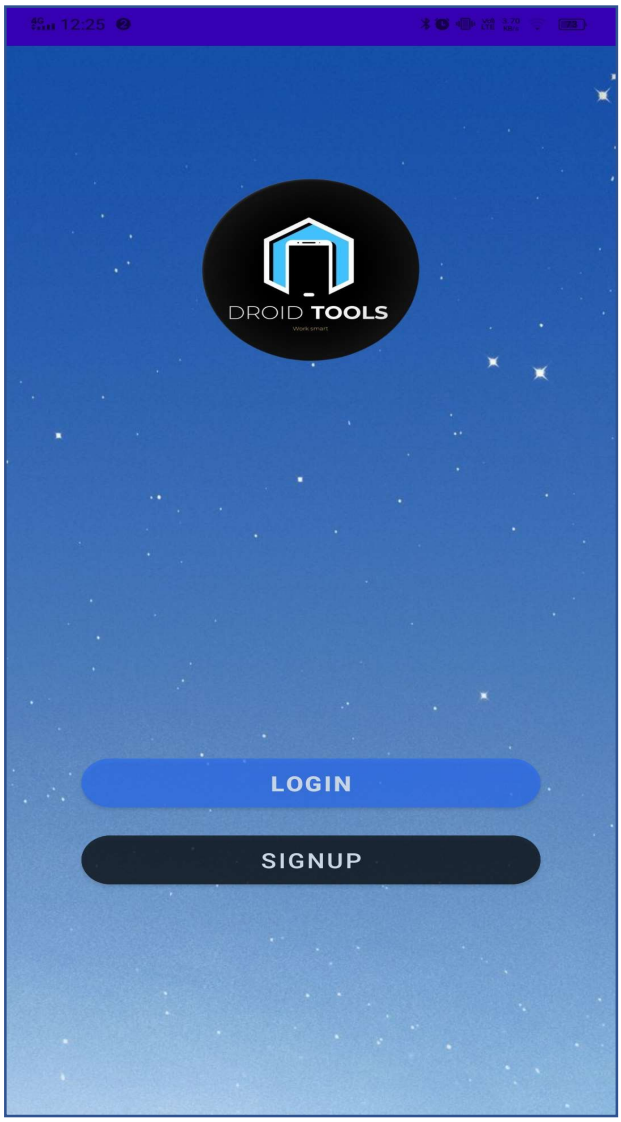
```
.setPositiveButton("Logout", (dialogInterface, i) -> {
                                    logout.logMeOut();
                                    Intent intent31 = new Intent(AppMenuPage.this, LoginPage.class);
                                    startActivity(intent31);
                                    finish();
                                    overridePendingTransition(0,0);
                                })
                    .setNegativeButton("Cancel", (dialogInterface, i) -> {
                        Intent intent21 = new Intent(AppMenuPage.this, AppMenuPage.class);
                        startActivity(intent21);
                        overridePendingTransition(0,0);
                    })
                    .show();


                overridePendingTransition(0,0);
                return true;
            default:
                throw new IllegalStateException("Unexpected value: " + item.getItemId());
        }
        return false;
    });
  }
@Override
  public void onBackPressed() {
    LogOutApp logout = new LogOutApp();
    new AlertDialog.Builder(this).setMessage("Are you sure to logout?")
            .setCancelable(false)
```

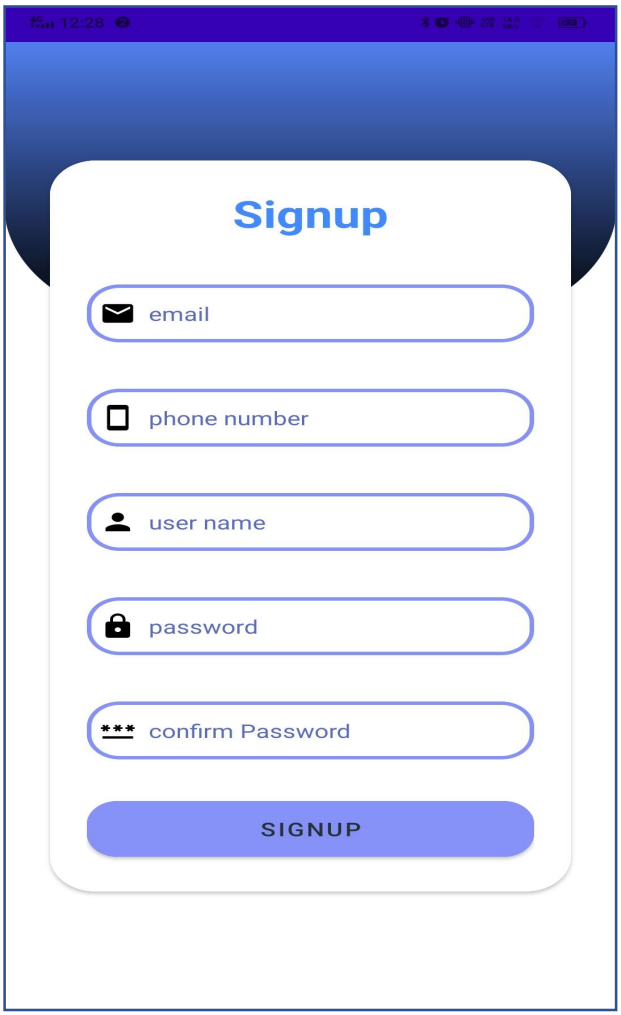
```
.setPositiveButton("Logout", (dialogInterface, i) -> {
                logout.logMeOut();
                Intent intent31 = new Intent(getApplicationContext(), LoginPage.class);
                startActivity(intent31);
                finish();
                overridePendingTransition(0,0);
            })
            .setNegativeButton("Cancel", (dialogInterface, i) -> {
                Intent intent21 = new Intent(getApplicationContext(), AppMenuPage.class);
                startActivity(intent21);
                overridePendingTransition(0,0);
            })
            .show();


        overridePendingTransition(0,0);
    }
}
```

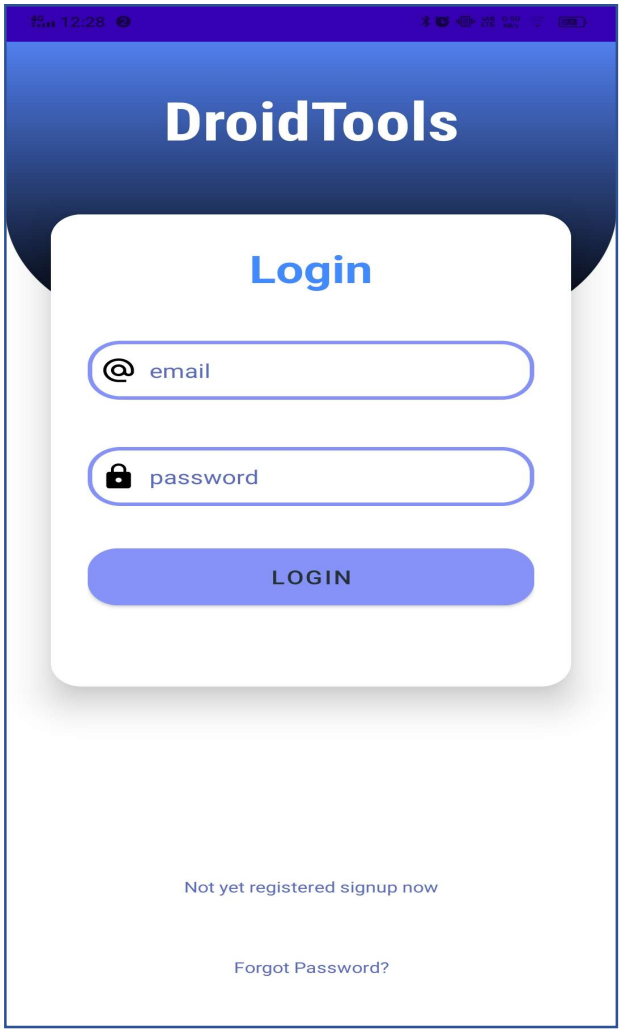


SCREENSHOT

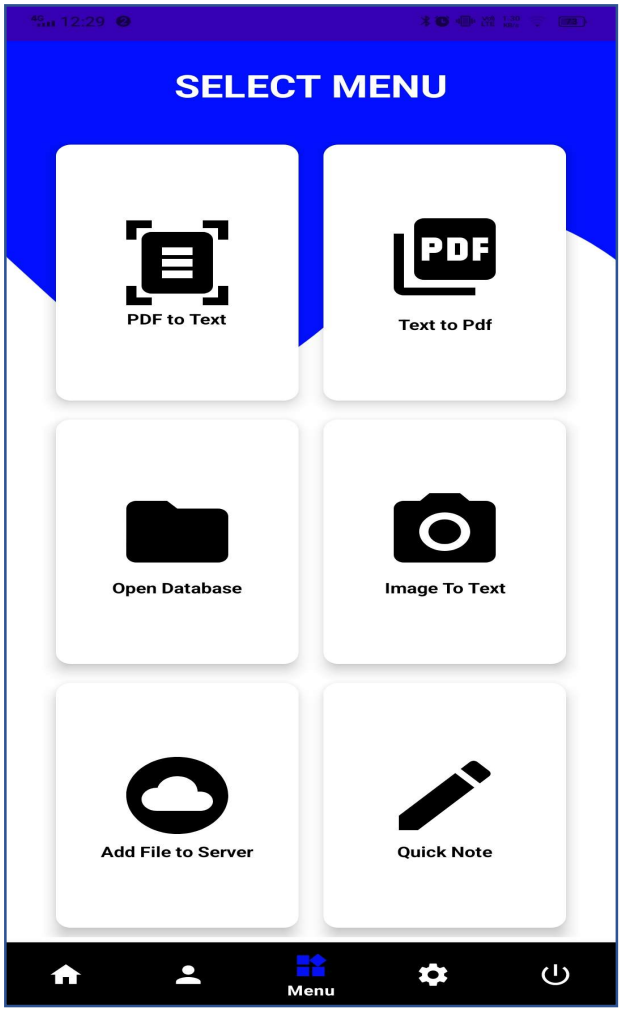

12:25
DROID TOOLS
Work smart
Welcome to Droid Tools
Features In App
PDF
Pdf and Text converter
IMG
Image and Text converter
Server Uploads


12:25
DROID TOOLS
Work smart
LOGIN
SIGNUP

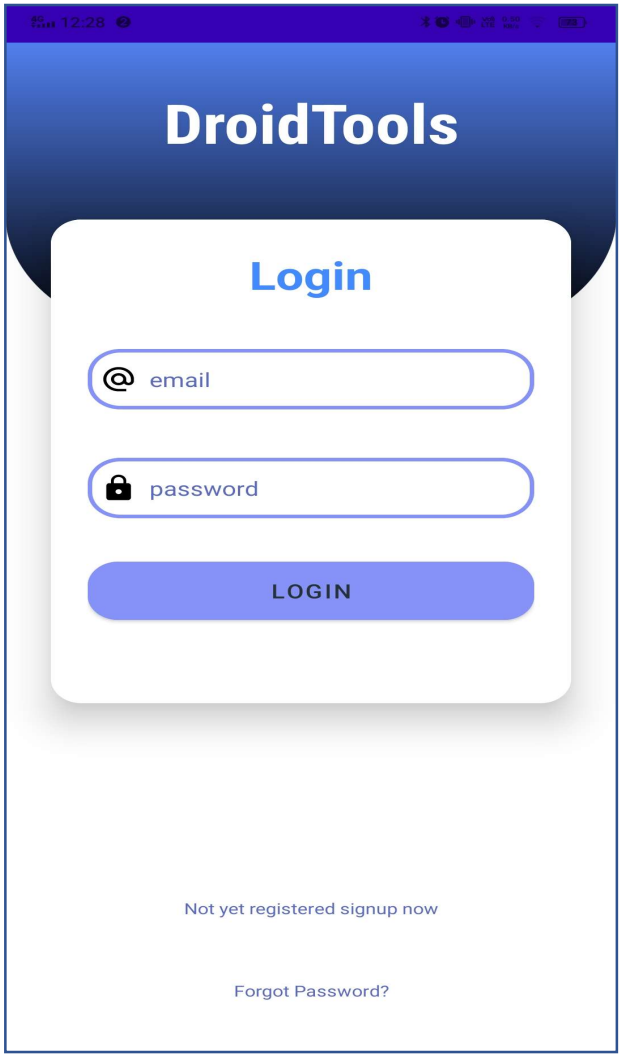

12:28
Signup
email
phone number
user name
password
confirm Password
SIGNUP

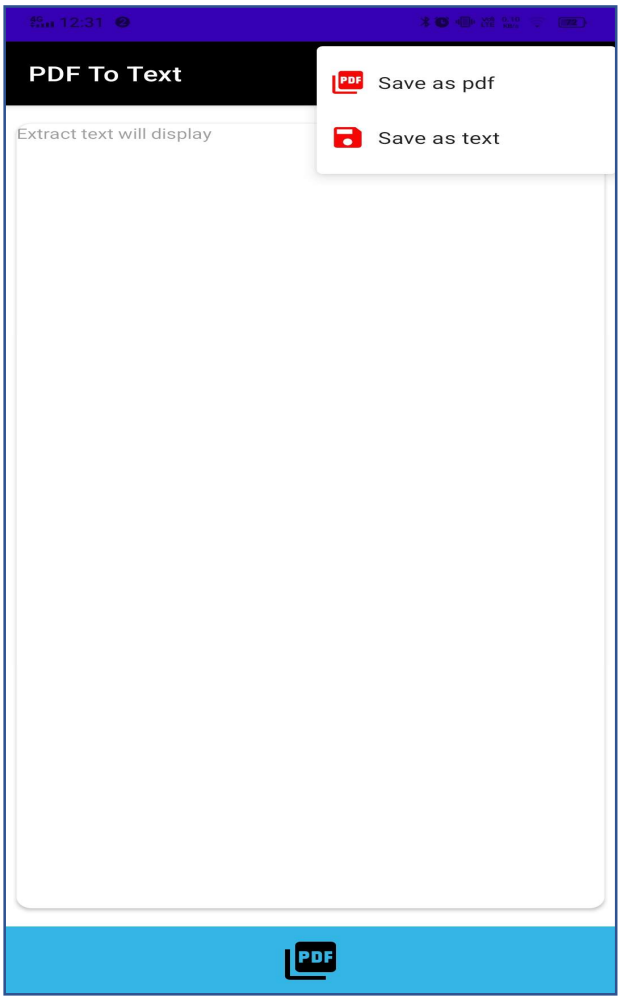

12:28
DroidTools
Login
email
password
LOGIN
Not yet registered signup now
Forgot Password?

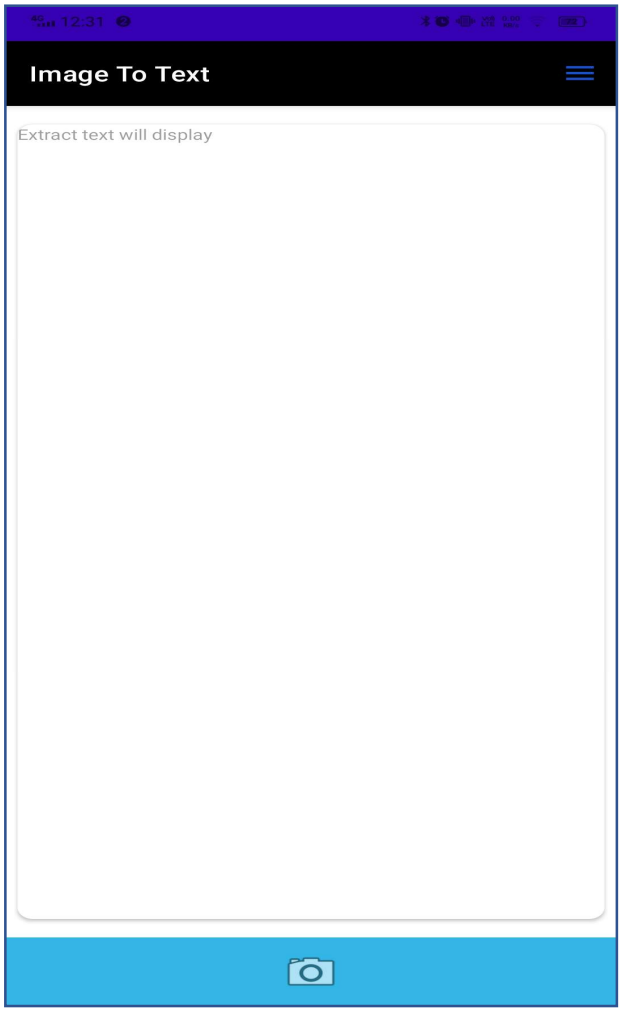

12:29
Home Page
Image To Text
Extract text from image tool.
Save to Database
Upload local file on database to access always.
Text to Pdf
Convert a text to pdf file.
Quick Notes
Make a quick note in the time of emargency.
PDF to Text
Convert a PDF file to text file.
Open Database
Open database to access saved files..
Future Updates
Voice Extractor
Extract voice from any video.
Coming Soon
Home

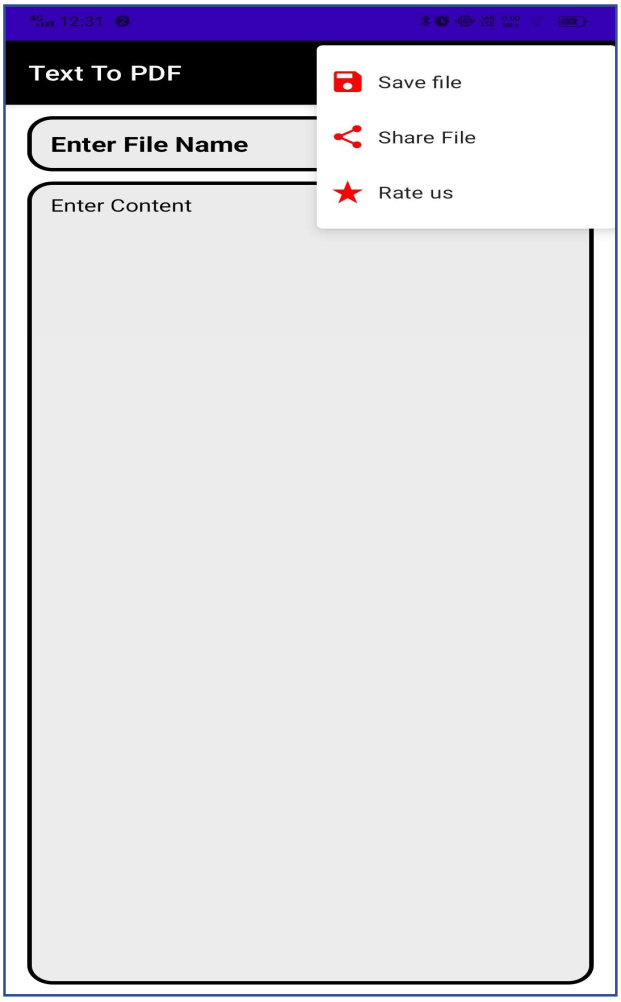

12:29
SELECT MENU
PDF to Text
Text to Pdf
Open Database
Image To Text
Add File to Server
Quick Note
Menu


12:30
DROID TOOLS
Work smart
The User
user1234@gmail.com
9876543210
V8enClFGlJV4EyUIA6uaOfxJ
5sG2
2023-06-29
Profile

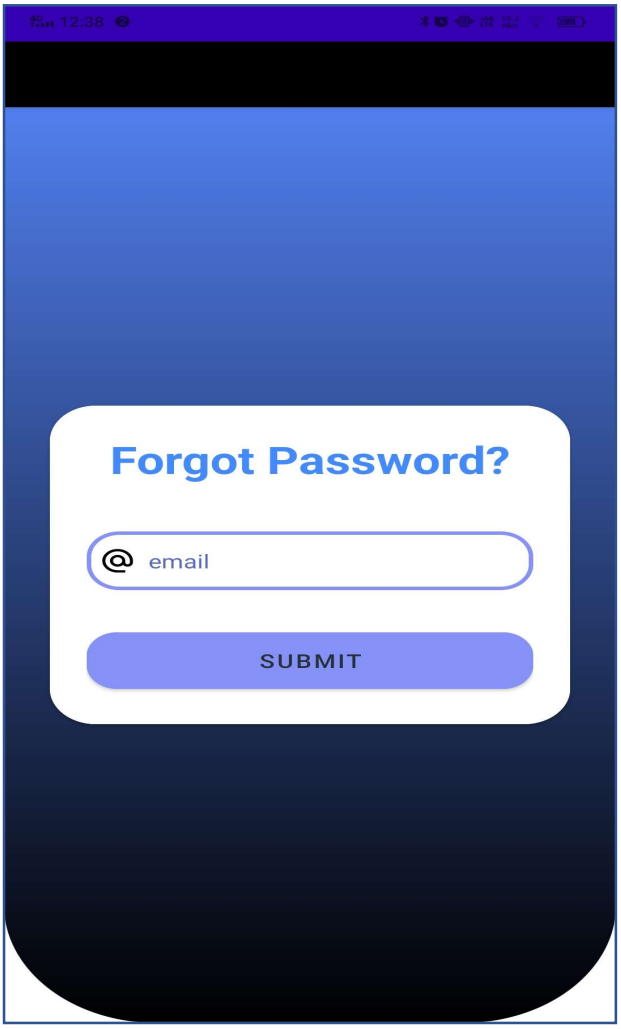

12:28
DroidTools
Login
email
password
LOGIN
Not yet registered signup now
Forgot Password?

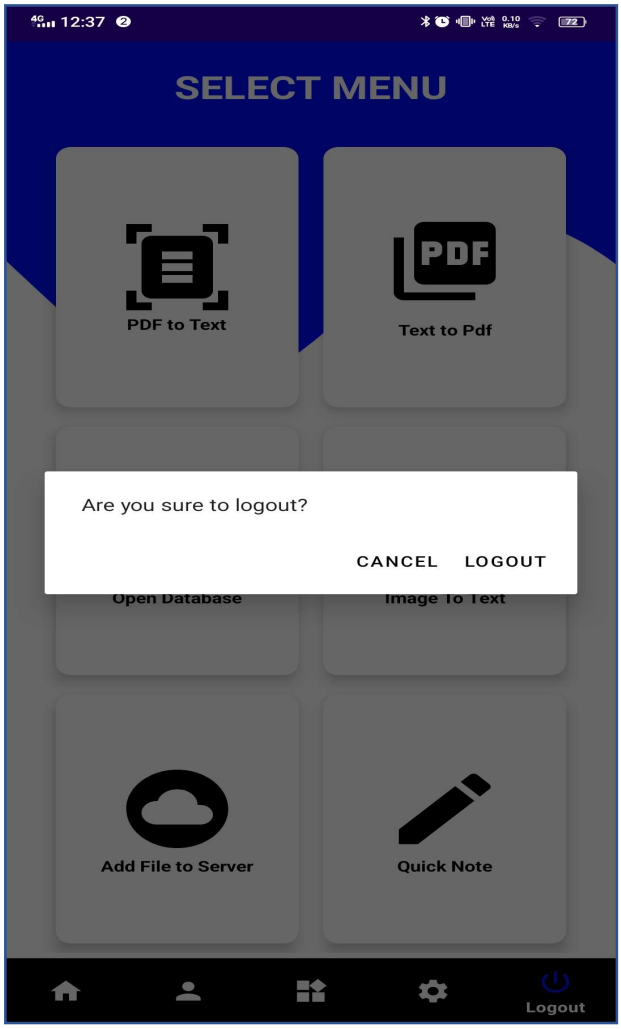

12:31
PDF To Text
Save as pdf
Save as text
Extract text will display
PDF


12:31
Image To Text
Extract text will display

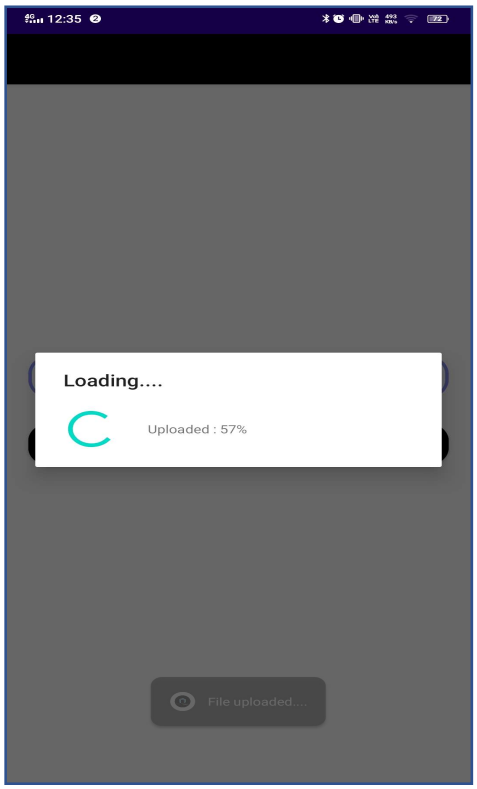

12:31
Text To PDF
Save file
Share File
Rate us
Enter File Name
Enter Content

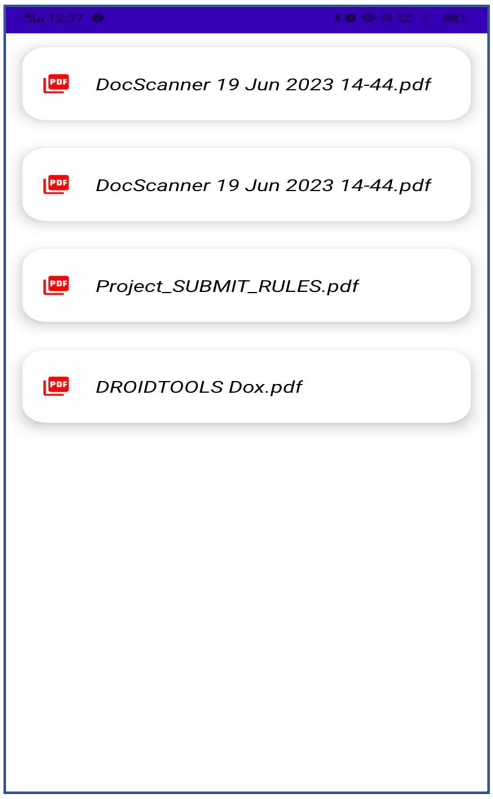

12:32
Quick Notes
note title
note description

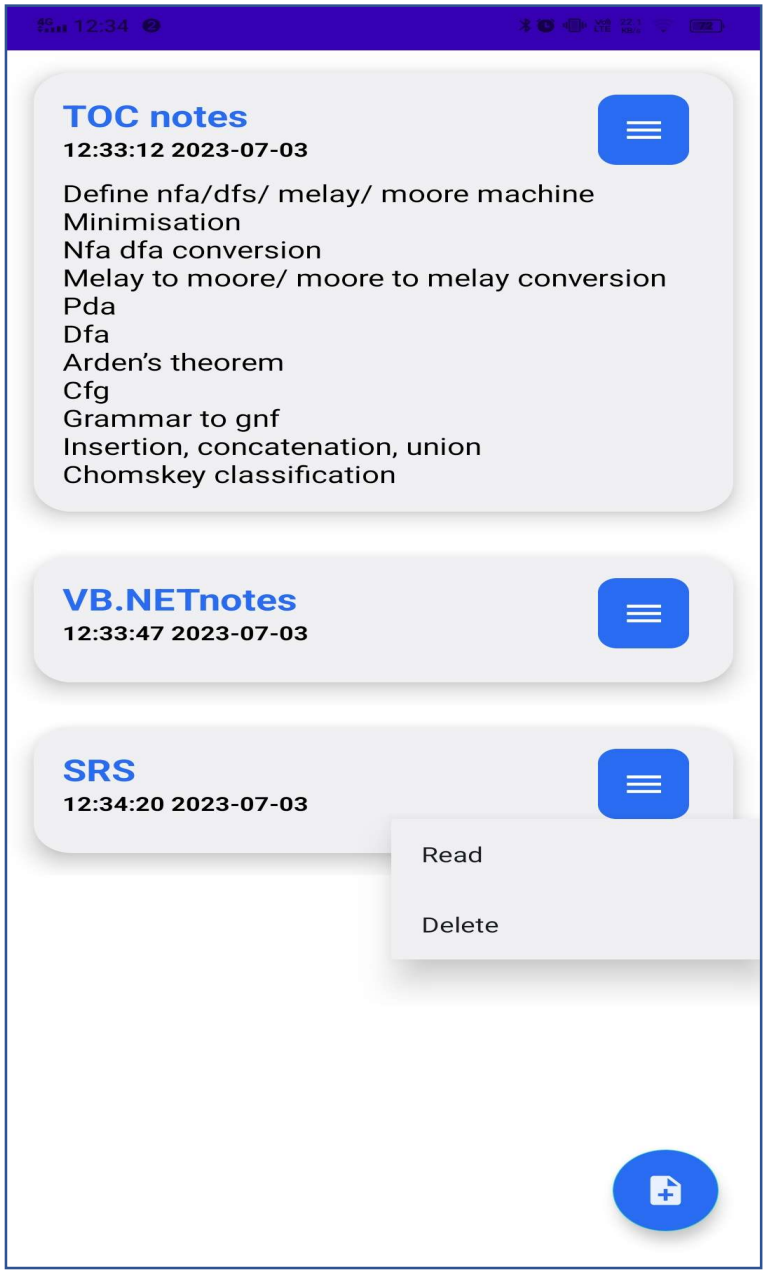

Forgot Password?
email
SUBMIT


SELECT MENU
PDF to Text
Text to Pdf
Are you sure to logout?
CANCEL
LOGOUT
Add File to Server
Quick Note
Logout


DROIDTOOLS.pdf
UPLOAD


Loading....
Uploaded : 57%
File uploaded....


DocScanner 19 Jun 2023 14-44.pdf
DocScanner 19 Jun 2023 14-44.pdf
Project_SUBMIT_RULES.pdf
DROIDTOOLS Dox.pdf


12:34
TOC notes
12:33:12 2023-07-03
Define nfa/dfs/ melay/ moore machine
Minimisation
Nfa dfa conversion
Melay to moore/ moore to melay conversion
Pda
Dfa
Arden's theorem
Cfg
Grammar to gnf
Insertion, concatenation, union
Chomskey classification
VB.NETnotes
12:33:47 2023-07-03
SRS
12:34:20 2023-07-03
Read
Delete

SECURITY ISSUES

Security Issues:

- Security is the probability that attack of a specific type will be repelled. Any computer based system that manages sensitive information may become intentional or unintentional target of illegal penetration. Penetration spans a broad range of activities: disgruntled employees who attempt to penetrate for revenge; dishonest individual who attempt to penetrate for illicit personal gain.

- Security testing attempts to verify protection mechanism built into a system will in fact protect it from improper penetration. The system security must test for invulnerability from frontal attack but also be tested for invulnerability from flank or rear attack. During security testing the tester plays the individual who desires to penetrate the system.

- The tester may attempt to acquire passwords through external clerical means or may attack the system with customer software designed to break down any defences that have constructed. The

MAINTENANCE

Software maintenance is widely accepted part of SDLC now a days. It stands for all the modifications and updations done after the delivery of software product. There are number of reasons, why modifications are required,some of them are briefly mentioned below.

Market Conditions- Policies, which changes over the time, such as taxation and newly introduced constraints like, how to Maintain book keeping, may trigger need for modification.

Client Requirements- Over the time, customer may ask for new features or functions in the software.

Host Modifications- If any of the hardware and/or platform (such as operating system)of the target host changes, software changes are needed to keep adaptability.

Organization Changes - If there is any business level change at client end, such as reduction of organization strength, acquiring another company, organization venturing into new business, need to modify in the original software may arise.

Types of maintenance

In a software lifetime, type of maintenance may vay based on its nature.

Following are characteristics: some types of maintenance based on

their characteristics.

Corrective Maintenance- This includes modifications and update done in order to correct or fix problems, which are either discovered by user or concluded by user error reports.

Adaptive Maintenance- This includes modifications and update applied to keep the software product UpToDate and tuned to the ever change world of technology and business environment

Perfective Maintenance- This includes modifications and updates done in order to keep the software usable over long period of time. It includes new features, new user requirements for refining the software and improve its reliability and performance.

SYSTEM DESIGN

Software design is an interactive process through which requirements are translated into a blue print for constructing the software. The design provides a complete picture of the software addressing the

data, functional and behavioral domains from an implementation perspective. The design in this project is modular, i.e., the software can be locally sub-divided into sub-modules; each having a non overlapping definite set of functions. The design contains distinct representation of data architecture interface and modules. It is the way in which we can accurately translate customer requirements into

finished product or system.

The principles of design are as follows:-

- The design process does not suffer from ant tunnel vision. Al the alternative approaches are considered and are judged each based on the requirements of the problems.

- The design is traceable to the analysis model.

- The design does not reinvent the wheel.

- The design “minimizes the intellectual distance” between the software and the problem as it exists between the real worlds.

- The design should exhibit uniformity and integration.

- A design should be structured in such a way that – limited amount of cost should be incurred to reflect the modification.

- **Procedural abstraction:** It is a name sequence of instruction, which has specific and limited function. For example: the word "click for a module to get activated".
- **Data abstraction:** It is a collection of data that describes a data object.
- **Control abstraction:** It implies a program control mechanism without specifying the internal details.

Refinement:-Refinement is a process of elaboration. The statement describes a function of information conceptually but provides no information about the internal workings of the function or the internal structure of the information. This project work provides a semantic mechanism for decomposing the problem into sub problems that will reduce the complexity of the overall problems. Thus it can be stated that **modular decomposability** is present. The design methodology in this project work enables existing design components to be assembled into a new system. Hence it can be stated that **modular compatibility** is also present. Each module can be understood as a standard unit – which makes it easier to easier to modify and change. Thus this project

Modularization Details:

Data Integrity and Constraints: The primary activity during data design is to select logical representation of data objects identified during the requirements specification and definition phase. The selection process may involve algorithmic analysis of alternative structures in order to determine the most efficient design or may involve the use of the set of modules that provide the desire operation upon some representation of the object; well designed data can lead to better program structure and modularity and reduce procedural complexity. In this project work some principles for data specification have been maintained:

- The systematic analysis principles applied to the function and behaviour is applied to the data.
- All data structures and the operations to each are identified.
- A data dictionary is established and is used to define both data and program design.
- Low level data-design decision is deferred until late in the design process.

Only those modules that must make direct use of data structures know the representations of data

structures.

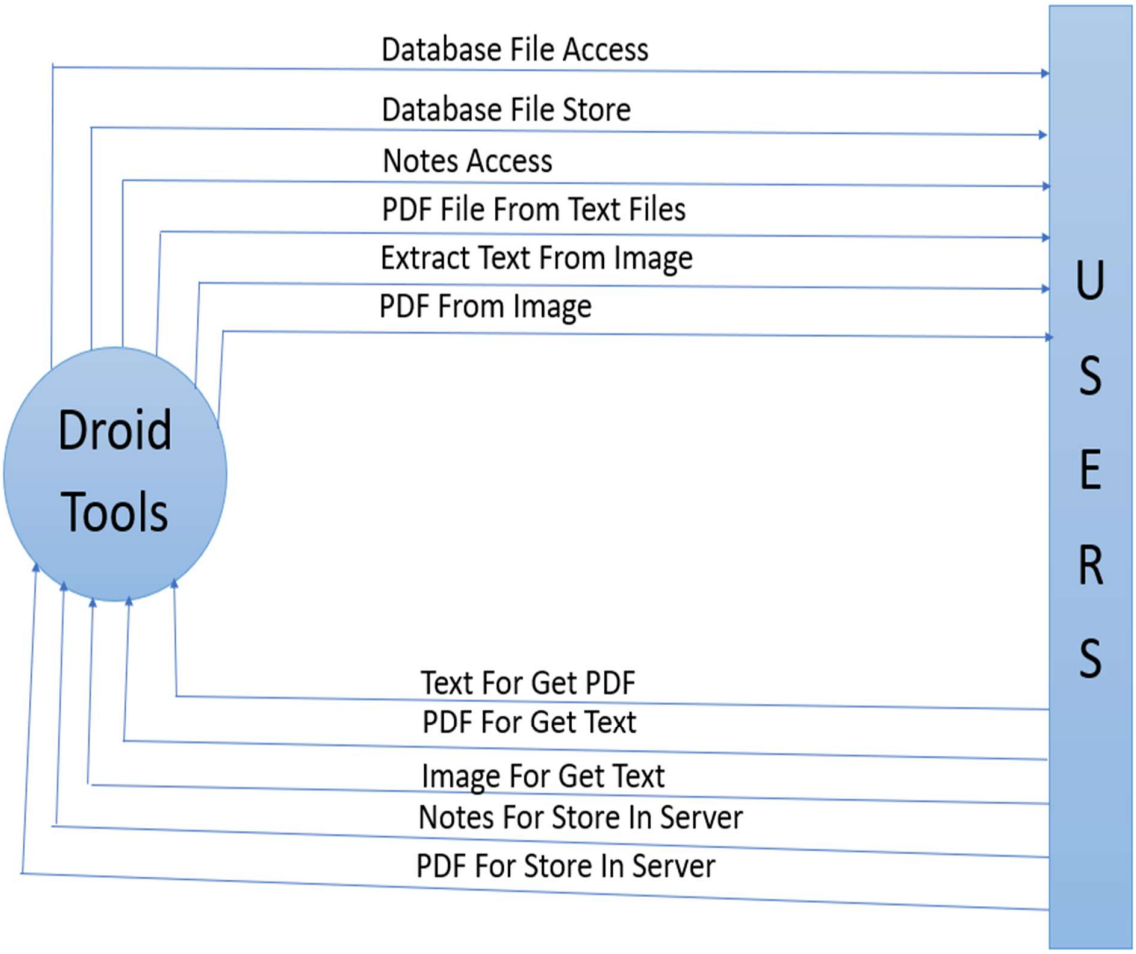
Architectural Design: The primary objective of the architectural design is to develop a modular program structure and represent a control relationship between the modules. In addition architectural design needs program structure and data structure defining interfaces that enable a data to flow throughout the program. Each software design method has strength and weakness. An important selection factor for the design method is the breadth of applications to which it

Information enters the system along the path that transforms external data into an internal form. These paths are identified as an incoming-flow. At the kernel of the software a transition occurs incoming data are passed through a transform center and begin to move along the paths that now lead out of software. Data moving along these paths are called outgoing flow. The overall flow of data occurs in a sequential manner and follows one or only a few straight line paths. Transaction flow is characterized by data moving along the incoming path that converts external works of information into a transaction. The transaction is evaluated and based on its value flow along one of many action paths are initiated. The hub of information flow from which many action paths forms is called Transaction centre.

Creation of data-flow model: The data flow diagram enables the software engineer to develop models of the information domain at the same time. As DFD is referred to greater levels of detail, the analyst performs an implicit functional decomposition of the system.

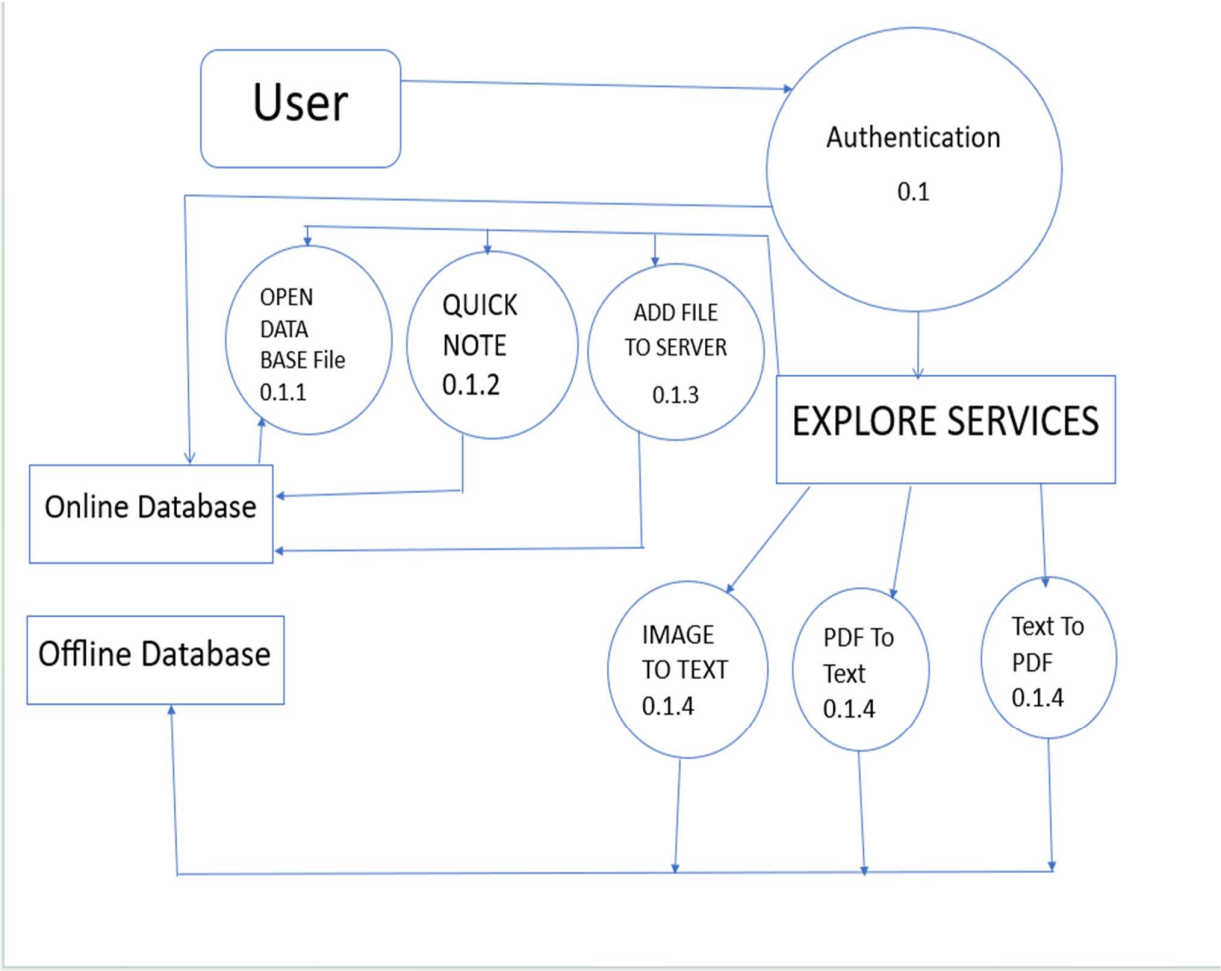
DFD & ERD

DFD:


Database File Access
Database File Store
Notes Access
PDF File From Text Files
Extract Text From Image
PDF From Image
Droid Tools
USERS
Text For Get PDF
PDF For Get Text
Image For Get Text
Notes For Store In Server
PDF For Store In Server

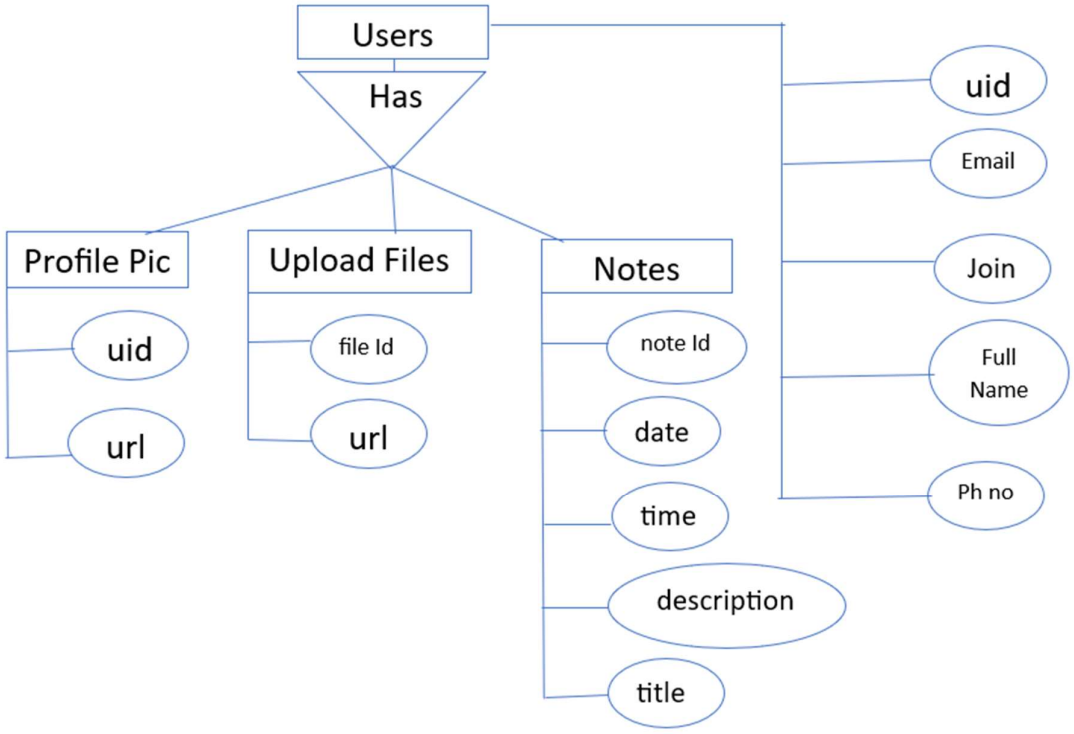
Level 0

DFD:

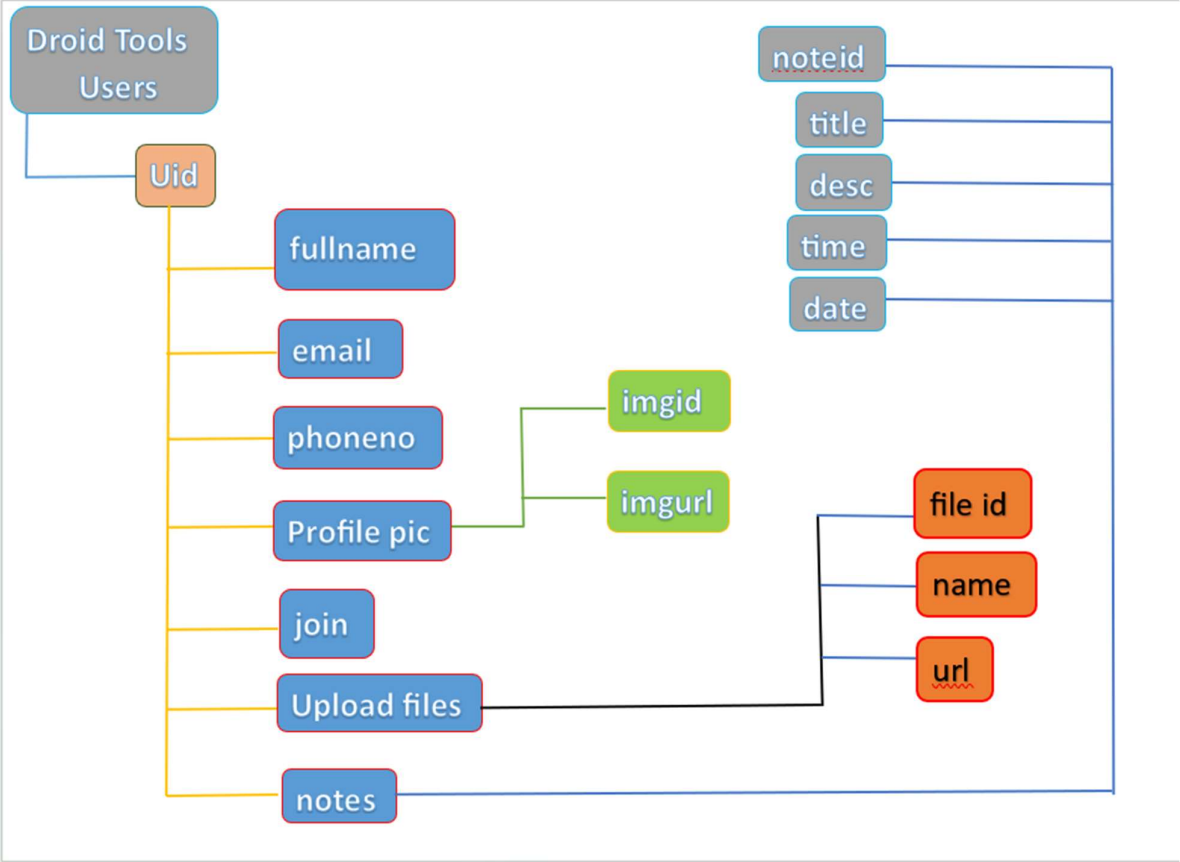

User
Authentication
0.1
OPEN
DATA
BASE File
0.1.1
QUICK
NOTE
0.1.2
ADD FILE
TO SERVER
0.1.3
EXPLORE SERVICES
Online Database
Offline Database
IMAGE
TO TEXT
0.1.4
PDF To
Text
0.1.4
Text To
PDF
0.1.4

Level 1

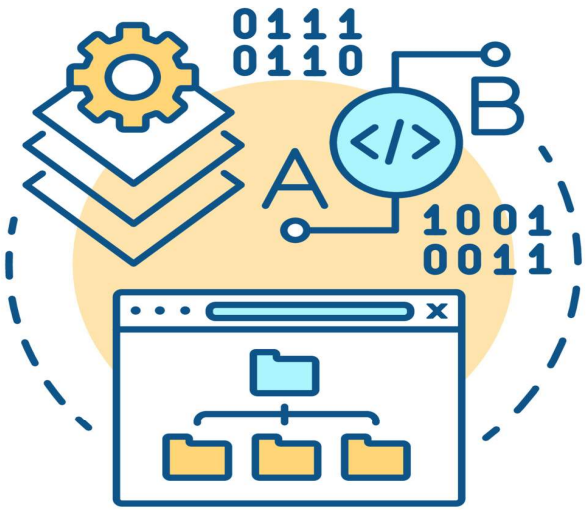
ERD:


Users
Has
uid
Email
Join
Full Name
Ph no
Profile Pic
uid
url
Upload Files
file Id
url
Notes
note Id
date
time
description
title

DATABASE STRUCTURE:



Testing Technique
And
Testing Strategies



Testing Objectives:-

- Testing is the process of executing a program/system with the intention of finding error(s).
- A good testing strategy is one that has a high probability of finding an ; as-yet-undiscovered error.
- A testing strategy must also signify the list of allowable errors and also the types of seeded errors.

These objectives imply a dramatic change towards the view-point of testing. The main motive of designing Test-scenarios and Test-cases should be that – a minimum amount of cost maximum number of errors should be detected.

Testing Principles:-

- All test should be traceable to customer's requirement.
- Tests should be planned long before test begins.
- Testing should begin "in small" and progress toward testing "in the large".
- Exhaustive testing is not possible; hence proper input domains should be, designed.
- To be most effective "Testing" should be conducted by an independent third party.

Testability: -

Software testability is simply how easily a computer program can be tested.

Operability: -

"The better it works the more efficiently it can be tested".

- The system has few bugs (bugs add analysis and reporting overhead to the test process).

- No bugs block the execution of tests.
- The product evolves in functional stages

Observability:-

"What you see is what you test".

- Distinct output is generated for each input.
- System states are variables are visible or quarriable during execution.
- Past system states and variables are visible or cerebella.
- All factors affecting outputs are visible.
- Incorrect output is easily identified.
- Internal-errors are automatically detected through self-testing mechanisms.
- Source code is accessible.

Controllability: -

"The better we can control the software; the most of the testing can be automated and optimized".

- All possible outputs can be generated through the combination of input.
- All code is executable through some combination of input.
- Software and hardware states and states and variables can be directly controlled by test engineer.
- Input and output formats are consistent and structured.
- Tests can be conveniently specified and reviewed

Decomposability: -

"By controlling the scope of testing we can isolate the problem more quickly and effectively".

- The software is built from independent modules.
- The software modules can be tested independently.

Simplicity: -

"The less there is to be tested the more quickly we can test it".

- Functional simplicity: the requirements in this project work for post office management system is account number, amount, date of deposits, date of withdrawal, which we can get from recurring table.
- Structural simplicity: This project architecture is modularization to limit the propagation of faults.
- Code simplicity: coding structure is adopted for case of inspection and maintenance.

Understand ability: -

- "The more we understand the system/software the smarter will be the test".
- The design should be well understood.
- Dependencies between internal, external and shared components are well understood.
- Changes to this design should be communicated to all the involved personnel.
- Technical document should be accurate, well specified, well organised and instantly accessible.

White box testing:-

White box testing sometimes called glass box testing is a test case design method that uses the control structure of procedural design to derive test cases. Using white-box testing methods the software engineer can derive test cases that:

- Guarantee that all independent paths within a module have been exercised at least once.
- Exercise all logical decisions on either true or false sides.
- Execute all loops at their boundaries and within their operational bounds,
- Exercise internal data structures to ensure their validity

Flow Graph Notation:-

The flow graph depicts logical control flow notation. The structured constructs in flow graph in following from where each circle represents one or more no branching POL or source code statements. A flow graph node represents one or more procedural statements. A sequence of process boxes and a decision diamond can map into a single node. The arrows in a called edges or links represent flow of control and are analogous to flowchart arrows. An edge must terminate at a node even if the node does not represent flow of and are analogous to flowchart arrows. An edge must terminate at a node even if the node does not represent any procedural statements. Areas are bounded by edges and nodes are called regions.

Cyclomatic Complexity: -

Cyclomatic complexity is software metric that provide a quantitative measure of the logical complexity of a program. When used in the context of basic path testing method the value computed for

Cyclomatic complexity defines the number of independent paths in the basis of computer program and provides us an upper bound for independents paths for the basis of the program and provides us the upper bound for the number of tests that must be conducted to ensure that all statements have been executed at least once.

An independent path is any path through the program that introduces at least new set processing statements or condition. When stated in terms of flow graph and independent path can move along at at least one edge that has not been traversed before the path. Complexity is computed in three ways:

- The 'n' no.of regions of flow graph correspond to the cyclomatic complexity.
- Cyclomatic complexity V(G) for a flow grapg G is defined as V(G) = E – N + 2;
- Where E --> no.of flow graph edges N is the no.of flow graph nodes.
- Cyclomatic complexity V(G) for a flow graph G is also defined as V(G) = P + 1;
- Where P-->no.of predicate nodes contained in the flow graph G

Control structure testing:-

The basis path testing technique is no.of technique for control structure testing.

Condition Testing:

Condition testing is a case design method that exercise the logical conditions contained in the program module(s). a simple condition is Boolean variable or a relation expression possibly proceeded with one NOT operator. A relational takes the form E1 < relational operator> E2 where E1 and E2 are arithmetic expression and <relational operator> is one of the following: <, <=, >, >=, =, !=. A compound condition consists of two or more simple conditions. Boolean operators allows in compound condition include OR (||), AND (&) NOT (!). A condition without a relational expression S is referred to Boolean expression.

If the condition is incorrect then at least one of the components of the condition is incorrect. Therefore

types of errors include the following:

- Boolean operator error.
- Boolean variable error.
- Boolean parenthesis error.
- Relational operator error.
- Arithmetic expression error.

Condition testing strategies has two advantages. First measures the test coverage of a condition is simple.

Second the test condition in a program provides a simple guidance for a generation of additional tests of a program.

Data Flow Testing: -

The data flow testing method selects test paths of a program. A no.of data flow testing methods have been studied and compared.

Data flow testing methods are useful for selecting test paths of a program containing if and loop statements.

Loop Testing: -

Loop testing is a White-box testing technique that focuses exclusively on the validity of the loop conditions. Four different types of loops can be defined.

Simple Loops: -

The following tests can be applied to simple loops. Where N is the maximum number of allowable pass through the loop

Nested Loops: -

If we were to extend the test approach for simple loops the number of possible would grow exponentially.

This would result in impractical number of tests

- Start at the inner most loop. Set all other loops to minimum values.
- Conduct simple loop test for the inner most loop while holding the outer loops at their minimum iteration parameter values. Add other test for out-of-range or excluded values.
- Work outward to conduct tests for the next loops but keeping all outer loops at minimum values and nested loops to 'typ

Concatenated Loops: -

Concatenated loops can be tested using the approach defined for simple loops are independent of the other. However if two loops are concatenated and the loop counter for loop one is used as initial value for lop2 then the loops are not independent. When the loops are not independent the approach applied to nested loops is recommended.

Unstructured Loops:

Whenever possible this class of loops should be redesigned to reflect the use of the structured programming constructs.

Black-box Testing: -

Black-box testing also called behavioral testing focus on the functional requirements for a program. That is Black-box testing enables the software engineer to derive sets of input conditions that will fully exercise functional requirements for a program. Black-box testing attempts to find the following errors:

- Incorrect or missing functions.
- Interface errors.
- Errors in data structures
- Behavior or performance errors.
- Initialization or termination errors.

Black-box testing is designed to answer the following questions.

- How is functional validity tested?
- How is system behavior and performance tested?
- What is the class of input that will make good test-cases.
- Is the system sensitive to particular input values?
- How are the boundaries of a data class isolated?
- What data rates and data volumes can the system tolerate?
- What effect will the specific combination of data have on system operation?

Graph based testing methods: -

The first step towards Black-box testing method is to understand the object i.e. modelled in software and the relationship that connects; begin by creating a graph a collection of nodes that represents objects links that represents relationship between objects node weights that describe properties of a node and link weights that describe some characteristic of a link.

Equivalence Class Partitioning:-

Equivalence partitioning is a Black box testing method that decides the input domain of a program into classes of data from which tests can be derived. An ideal test case single handily uncovers a class of errors that might otherwise many case to be executed before the general errors observed. Equivalence partitioning strives to derive a test case to be executed before the general error is observed. Equivalence partitioning strives to derive a test case that uncovers classes of errors there by reducing total no. of test cases that must be developed.

Test case design for equivalence partitioning is based on the evaluation of equivalence classes for in input condition. An Equivalence classes represents a set of valid or invalid states for invalid conditions. Typically input conditions are either a specific number value, a range of values, a set of related values, or a Boolean condition. Equivalence classes may be defined according to following guide lines:

- If an input condition specifies a range that is; one valid and two invalid equivalence classes are one defined.
- If an input condition requires a specific value one valid and two invalid equivalence are defined.
- If an input condition specifies a number of a set one valid and one invalid equivalence classes are defined.
- If an input condition specifies a Boolean one valid and invalid class are defined.

Boundary value Analysis: -

Boundary value analysis is a test case design technique that complements equivalence partitioning. Rather that selecting an equivalence class BVA leads to the selection of test cases at the edges of the class.

1. If an input condition specifies a range bounded by values ‘a’b & ‘b’, test cases should be designed with values ‘a’ & ‘b’ and just above and just below ‘a’ & ‘b’.

2. If an input condition specifies a number of values test case should be developed that exercise minimum and maximum numbers. Values just above and below minimum and maximum number are also tested.

- Apply guide lines 1 and 2 two output conditions.
- If the internal data structures have prescribed boundaries are

  certain to design a test to exercise that boundary

Security Measures:

The directory definition of security compasses a set of measures taken to guard against theft, attack, crime, and espionage or sabotage. Security implies the quality or state of being secure, that is a relief from danger and acting so as to make safe against adverse contingencies. When talking about computers, security can have different meanings, including:

- Securities boundaries
- Physical securities
- Application securities
- Access control
- Message verification.
- Message ownership.

The growing use of and reliance on computer worldwide has resulted in business, Govt. and the military and even at home. Large amount of vital and sensitive data are increasingly becoming entrusted to end store in computers.

Unauthorized access, revelation, or destruction of data can violate industrial privacy. Corruption of stored data can result in significant and potentially catastrophic losses to the concern. It is worth remembering however, that security threats are not unique to computers. They are inevitably present in any form of safeguarding of valuable assets. The potential problem with computers is that the evolving immigrations of assets to them sometimes result in the delayed managerial of the value of and potential dangers of exposures to loss of information store there in.

Thus second security practices and policies of ten trials rather than precede migration of data and assets to computers.

System Security:

By system security we mean the ability to secure the underlying operating system, including the OS kernel and the essential utilities that make up the rest of the operating system. Limit number of user accounts that you give out and only allow them access to your systems via secure authentication mechanism.

Application Security:

By application security we should do defensive programming. Make sure that we take care of your memory. This means freeing all pointers, and preventing buffer overflows. Many security holes exist because we can overflow the buffer of a program and in turn that allows executing commands in the stack space of the program. If that programs happens to have super user privileges, the amount of damage that is possible to quite large. In our proposed project we must take extra care when making system calls to external programs.

Access Control:

Access control is the process you perform to determine if a particular connection belongs to a particular user and then to determine if they have the right to perform the requested action.

User Authentication:

User authentication is the process we preform to verify that a user is who they say they are. This is accomplished in this project work with the help of password.

User Authorization:

User authorization is the process we perform that determines whether someone has access to a particular system, application or data. Authorization may or may not require user authentication


CONCLUSION

- In conclusion, the application we have designed offers a comprehensive suite of services that cater to the diverse needs of our users. Throughout this presentation, we have highlighted the key features and benefits of our app, emphasizing its ability to convert text to PDF and vice versa, convert images to text, provide cloud storage, and enable quick note-taking.

- Furthermore, our app's image-to-text conversion feature offers a convenient solution for extracting text from images. Users can simply capture or upload an image, and our app will convert it into editable text, making it easy to extract information, save time, and enhance productivity.

- The cloud storage facility provided by our application offers a secure and convenient way for users to store and access their files.

- Additionally, our app includes a quick note-taking feature that allows users to capture and save important thoughts, ideas, or reminders on the go.

- We are confident that our application will meet the needs of users seeking a comprehensive tool for their file management, conversion, and note-taking requirements.

BIBLIOGRAPHY

1. Fundamental of Software Engineering
   by **Rajib Mall**

2. Google

3. W3SCHOOL.com


GLOSSARY

Glossary

Analysis:-

A description of the way in which a system work.

Business process:-

A set of steps used to achieve a business a business goal.

Class:-

An object that describes a set of objects with the same features.

Class instance:-

An object that belong to the class.

Clients:-

People from outside an organization that deal with the organization.

Computer- based information system:-

An information system that uses computer.

Context diagram:-

A diagram that shows that the input outputs of a system.

Data flow:-

Data flowing between processes, data stores and external entities.

Data flow diagram (DFD):-

A method to illustrate how data flow in the system.

Data store:-

A component of a DFD that describes the responsibility of data in a system.

.

Database:-

An organization store of data.

Design:-

Creation of an artefact.

Development process:-

A set of steps used to build a system.

Economic feasibility:-

An evaluation to determine whether a system is economically acceptable.

Employee:-

One who works for another for wages or salary.

Encapsulation:-

Inclusion of many features in the one object.

Entity:-

A distinct object in a system.

Feasibility analysis:-

An evaluation of whether it is worthwhile to proceed with a project.

Features:-

A characteristic of a class.

Feedback:-

Using variations from a system goal to change system behavior.

Function:-

A part that produces well defined outputs from given inputs

Functional dependency:-

Where one value of an attribute determines a single value of another attribute.

Global:-

The whole world or all people/world wide.

Information system:-

A system that provides information to people in an organization.

Inheritance:-

Using same features as another object.

Instance:-

A unique occurrence of a type of object.

Interviewing:-

Gathering information by asking questions.

Management:-

People responsible for organizing and allocating resources.

Methods:-

A feature that describes programs within an object.

On-line transaction:-

A transaction made through a terminal.

Operational Feasibility: -

An evaluation to determine whether a system operationally acceptable or not.

Organization:

Grouping and arranging into one whole or that which is grouped and arranged for a special purpose.

Polymorphism : -

Selecting the method appropriate for the class of the object called.

Process: -

A set of steps that define how things are done.

Programmer: -

A person who writes computer program.

Project: -

A scheme of something to be done.

Quality Assurance: -

A process to ensure that development of quality products.

Re-engineering: -

Changing an existing system.

Relation:

A table or list of values.

Security: -

Ensuring that computer system faults do not destroy the information stored about a system.

Software Configuration: -

The documents used in a development process.

Strategy: -

The broad objective for an organization.

System: -

A collection of components that work together to realize an objective.

System Development Cycle: -

Another term for project dictionary.

System Procedure: -

Defines actions to be taken to accomplish a system tasks.

System specification: -

A precise description of what the system must do.

System Analysis: -

Find out what a system does and what its needs are.

Technical Feasibility: -

An evaluation to determine whether the system can be technically built.

Testing: -

Checking to see if a system does what it is supposed to do.

Transaction: -

A simple interaction with a computer database.

User requirements: -

What user expects the system to do for them.