VIVEKANAND EDUCATION SOCIETY'S POLYTECHNIC,
SINDHI SOCIETY, CHEMBUR, MUMBAI - 400 071

PROJECT REPORT ON

MINI KINDLE

SUBMITTED BY

ANUJ DUBEY [65]

ATHARVA KENY [66]

MUNTZAR SAYYED [67]

SAGAR SONI [68]

GUIDED BY

MRS MEENA TALELE

NBA ACCREDITED FROM 2017 TO 2018

DEPARTMENT OF COMPUTER TECHNOLOGY

2018-2019

VIVEKANAND EDUCATION SOCIETY'S POLYTECHNIC, SINDHI SOCIETY, CHEMBUR, MUMBAI - 400 071

CERTIFICATE OF APPROVAL OF PROJECT

This is to certify that,

Mr./Ms.- Anoj Dubey, Atharva Keny, Muntzar Sayyed, Sagar Soni.

Roll No. – 65, 66.67 & 68of VI Semester Diploma in Computer Technology have completed the term satisfactorily in Industrial Project Development for academic year 2018-2019 as prescribed in the MSBTE curriculum.

Place: Chembur, Mumbai. Enrollment No:1700040231

1700040232

1700040233

1700040234

Date: Exam No:

Principal IC. Head of Department

Seal of Institutio

Project Guide Examiner

ACKNOWLEGMENT

We would like to thank all those people who helped us in successful completion of the project "MiniKindle (Online Library)"

We would like to thank **Mr. Vikrant Joshi**, Principal of V.E.S Polytechnic for his immense support and motivation.

We would like to thank **Mr. Avinash Dangwani**, IC. Head of the Department, Computer Technology of V.E.S Polytechnic, for his guidance. We are wholeheartedly thankful to him for giving us his valuable time & attention and for providing us a systematic way for completing our Project in time.

We would like to thank **Mrs. Meena Talele,** our project guide for his guidance. We have learn so many things from him and he motivated us and strengthened our confidence in doing the thesis. We express our deepest gratitude for his valuable suggestions and constant motivation that greatly helped the project work to successfully complete. Throughout the project work, his useful suggestions, constant encouragement has given us a right direction and shape to our learning.

We would also thank all the faculty members who have been a constant source and encouragement during the entire course of our study in this college.

TABLE OF CONTENT

CHAPTER 1

INTRODUCTION

Chapter 1

INTRODUCTION

1.1 OVERVIEW OF THE PROJECT

Our Project uses Android Development to store, organize and display books segregated and categorized according to their genres and themes. It assigns profiles to users so they can have individual recommendations of books and series according to their previous interest. It remembers the chapters the user has read so they can continue to read from the exact point they stopped reading (Bookmarks).

1.2 EXISTING SYSTEM

E-bookstore mainly in the book content providers in the role of the consumer's vision. Compared to the traditional paper books, its advantage is low cost, massive storage, convenient search positioning, personalized service, low carbon environmental protection and a variety of services. In general, operators in addition to the need to provide a large amount of reading data, to provide a perfect terminal reading application is also very important.

1.3 LIMITATIONS OF EXISTING SYSTEM

- Lack of security of data.
- More man power.
- Time consuming.
- Difficult to manage the financial details.

1.2 PROPOSED SYSTEM

Our system will give the ease to read the books as per the user's need. Updates will be sent to all the members via mobile messages and emails. Updating of the server with new books will be sent to the user. Basically, the application provides different magazines, novels, comics, manga etc.

1.3 ADVANTAGES OF PROPOSED SYSTEM

The system is very simple in design and to implement. The system requires very low system resources and the system will work in almost all configurations. It has got following features:

- Security of data.
- Ensure data accuracy’s.
- Proper control of the higher authority.
- Minimize manual data entry.
- Minimum time needed for the various processing.
- Greater efficiency.
- Better service.
- User friendliness and interactive
- Minimum time required
- Bookmark for the last page you read.
- Recommendation.

.

CHAPTER 2

SYSTEM DEVELOPMENT

Chapter 2

SYSTEM DEVELOPMENT

2.1 SYSTEM SPECIFICATION

2.1.1 Hardware Requirements

(Minimum of the hardware required)

- Processor: Min Intel Core i3.
- RAM Capacity: Min 2 GB
- Hard Disk Capacity: Min 10 GB.

2.1.2 Software Requirements

(Any higher version will do)

- Operating System: Android 4.4.0 KitKat
- Coding Language: Java, XML

2.2 TECHNOLOGY USED

2.2.1 Java

What is Java?

Java is a general-purpose computer-programming language that is concurrent, class-based, object-oriented, and specifically designed to have as few implementation dependencies as possible. It is intended to let application developers "write once, run anywhere" (WORA), meaning that compiled Java code can run on all platforms that support Java without the need for recompilation. Java applications are typically compiled to bytecode that can run on any Java virtual machine (JVM) regardless of computer architecture. As of 2016, Java is one of the most popular programming languages in use, particularly for client-server web applications, with a reported 9 million developers. Java was originally developed by James Gosling at Sun Microsystems (which has since been acquired by Oracle Corporation) and released in 1995 as a core component of Sun Microsystem's Java platform. The language derives much of its original features from Small Talk, with a syntax similar to C and C++, but it has fewer low-level facilities than either of them

2.2.2 Android Studio

What is Android Studio?

Android Studio is the official integrated development environment (IDE) for Google's Android operating system, built on JetBrains' IntelliJ IDEA software and designed specifically for development. It is available for download on Windows, macOS and Linux based operating systems. It is a replacement for the Eclipse Android Development Tools (ADT) as the primary IDE for native Android application development. Android Studio was announced on May 16, 2013 at the Google I/O conference. It was in early access preview stage starting from version 0.1 in May 2013, then entered beta stage starting from version 0.8 which was released in June 2014. The first stable build was released in December 2014, starting from version 1.0. The current stable version is 3.3, which was released in January 2019.

2.2.3 XML

What IS XML?

Extensible Markup Language (XML) is a markup language that defines a set of rules for encoding documents in a format that is both human-readable and machine-readable. The W3C's XML 1.0 Specification and several other related specifications all of them free open standards—define XML.The design goals of XML emphasize simplicity, generality, and usability across the Internet. It is a textual data format with strong support via Unicode for different human languages. Although the design of XML focuses on documents, the language is widely used for the representation of arbitrary data structures such as those used in web services.

2.2.4 FIREBASE

What IS Firebase?

Firebase provides a real time database and backend as a service. The service provides application developers an API that allows application data to be synchronized across clients and stored on Firebase's cloud. The company provides client libraries that enable integration with Android, iOS, JavaScript, Java, Objective-C, Swift and Node.js applications. The database is also accessible through a REST API and bindings for several JavaScript frameworks such as AngularJS, React, Ember.js and Backbone.js The REST API uses the Server-Sent Events protocol, which is an API for creating HTTP connections for receiving push notifications from a server. Developers using the real time database can secure their data by using the company's server-side-enforced security rules. Cloud Firestore which is Firebase's next generation of the Real time Database was released for beta use

CHAPTER 3

FEATURES

Chapter 3

Features

In today's world, it is very important to be technically advanced in all fields so by using this software , we can have an organized system to provide data and security .This model will help the user to read the various types of books..

1. Security of data.
2. Ensure data accuracy's.
3. Proper control of the higher authority.
4. Minimize manual data entry.
5. Minimum time needed for the various processing.
6. Greater efficiency.
7. Better service.
8. User friendliness and interactive.
9. Minimum time required.
10. Cost Efficient.

CHAPTER 4

CLASS DIAGRAM

Chapter 4

CLASS DIAGRAM

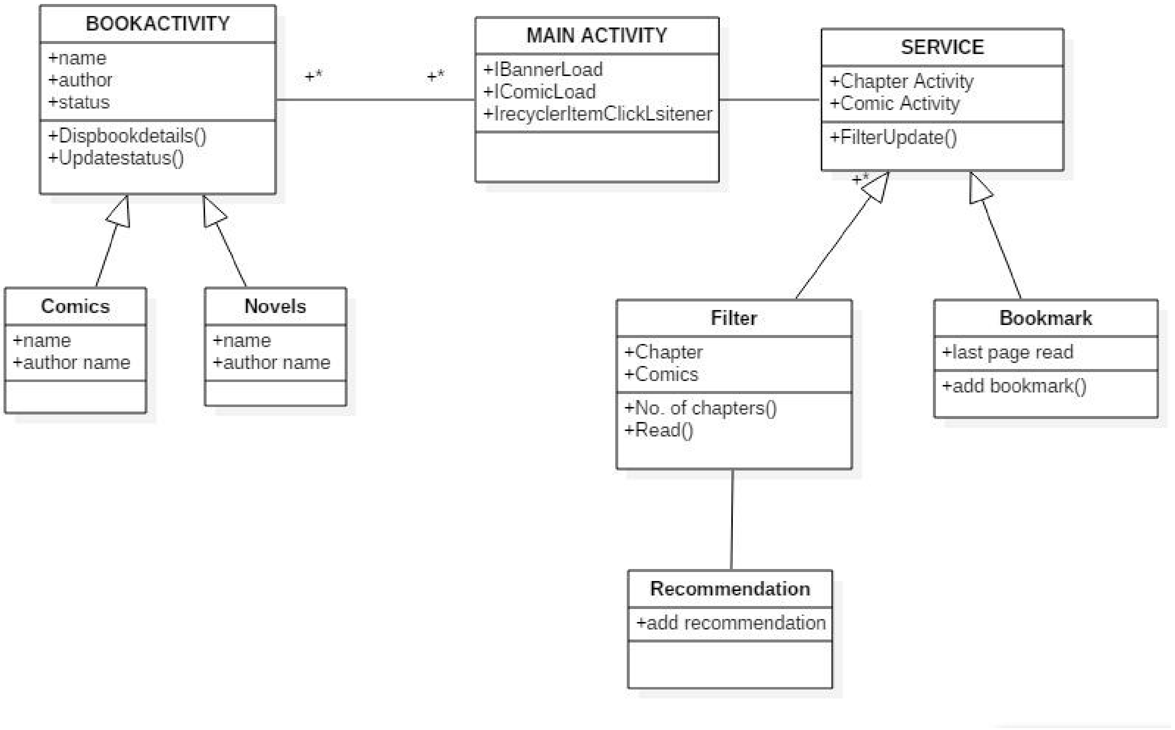


Fig 4.1 Use Case Diagram

CHAPTER 6

FLOWCHART

Chapter 6

FLOWCHART

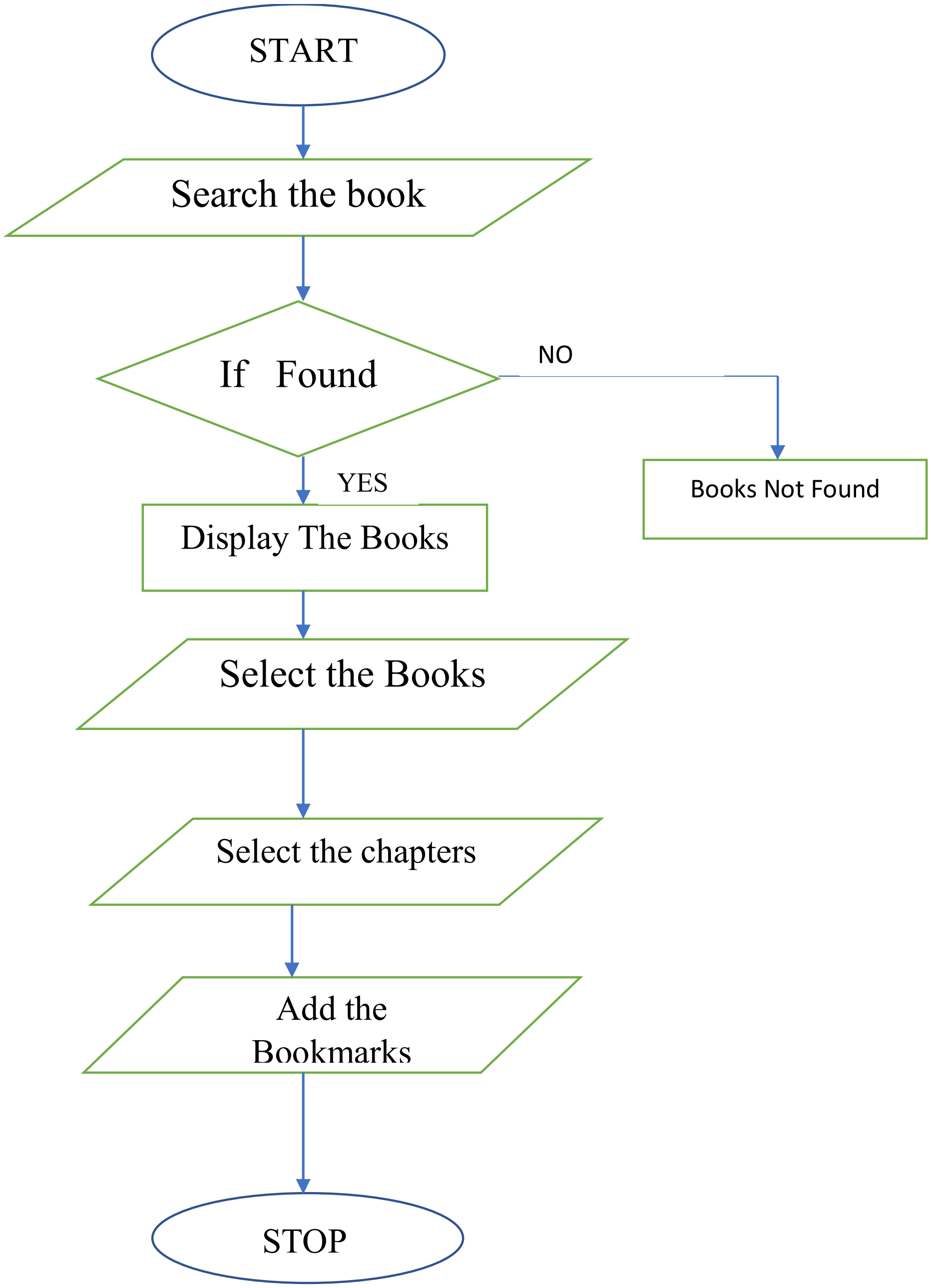


Fig 6.1 Flowchart for User’s Activity

CHAPTER 7

ALGORITHM

Chapter 7

ALGORITHM

1. Start
2. Open the application.
3. Select any of the book you want to read.
4. Search the Name of the Book.
5. Select the chapters from the book.
6. Filter the Book.
7. Add the bookmark on the last page you read, to continue from the from the last page you read.
8. Add your Feedback on the feedback panel.
9. Stop.

CHAPTER 8

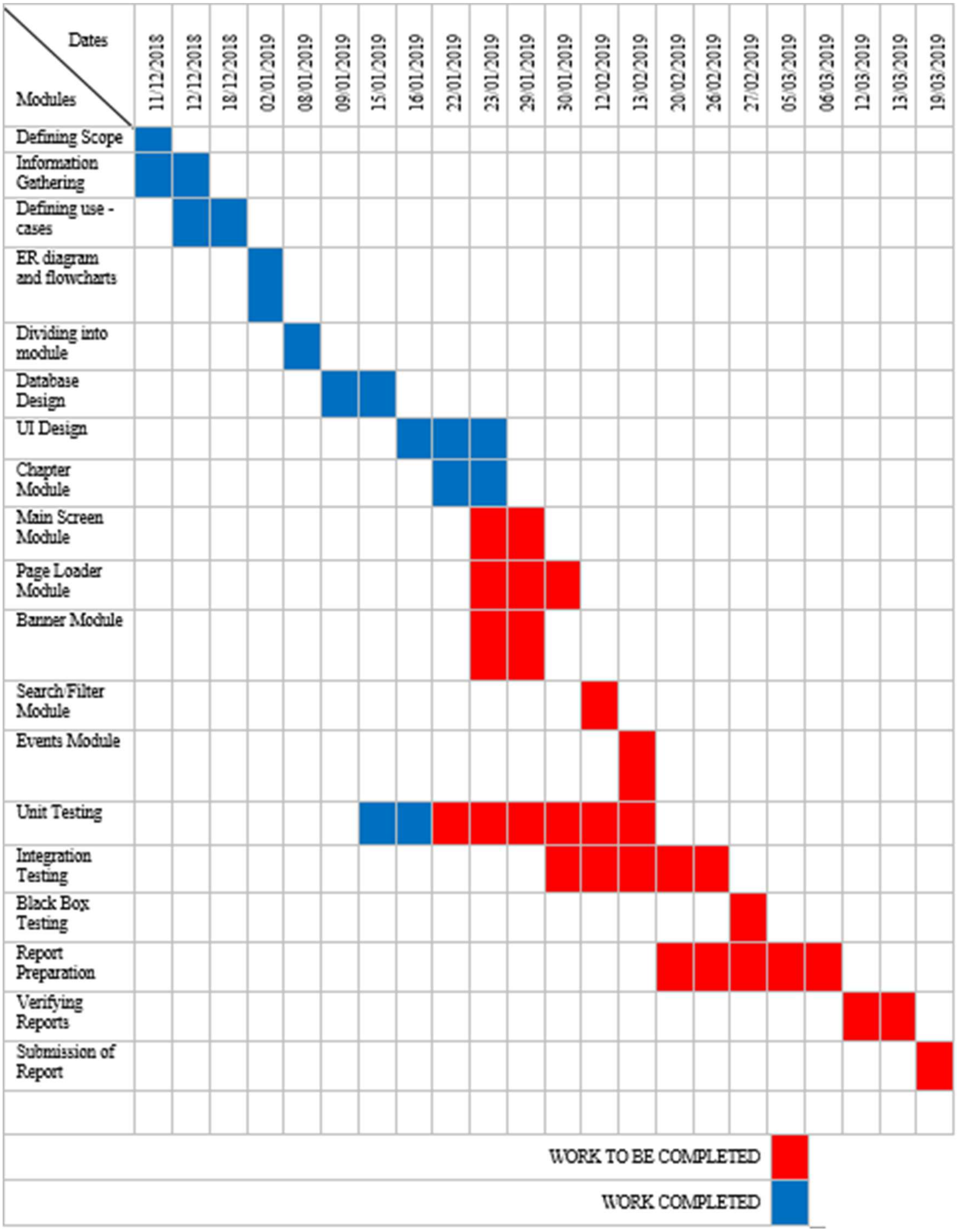
GANTT CHART

Chapter 8

GANTT CHART

- A **Gantt Chart** is a type of bar chart that illustrates a project schedule.
- Gantt charts illustrate our start and finish dates of the terminal elements and summary elements of a project.
- It is one of the most popular and useful ways of showing activities (tasks or events) displayed against time.

GANTT CHART-1



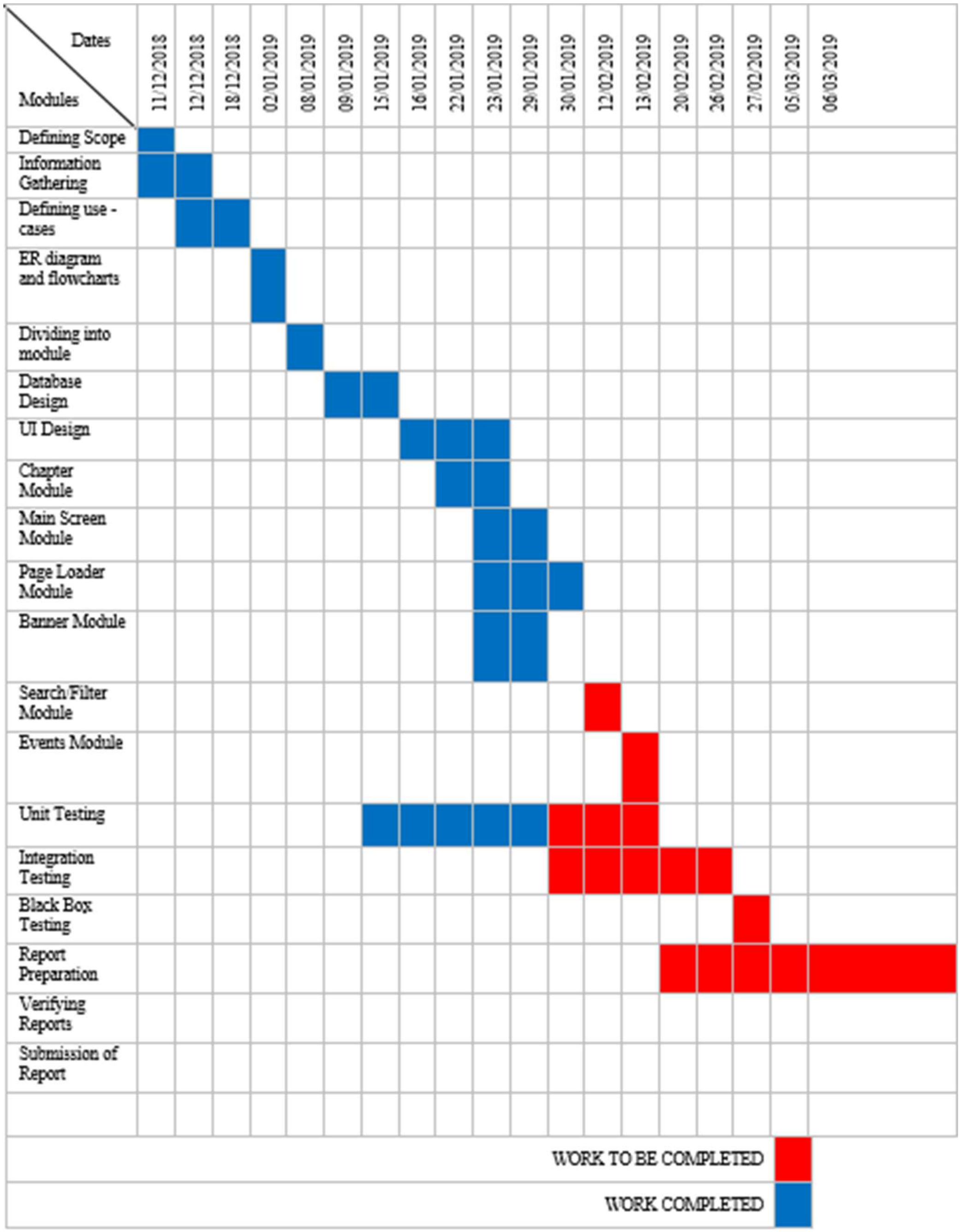
Dates Modules	11/12/2018	12/12/2018	18/12/2018	02/01/2019	08/01/2019	09/01/2019	15/01/2019	16/01/2019	22/01/2019	23/01/2019	29/01/2019	30/01/2019	12/02/2019	13/02/2019	20/02/2019	26/02/2019	27/02/2019	05/03/2019	06/03/2019	12/03/2019	13/03/2019	19/03/2019
Defining Scope	🟦																					
Information Gathering	🟦	🟦																				
Defining use - cases		🟦	🟦																			
ER diagram and flowcharts				🟦																		
Dividing into module					🟦																	
Database Design						🟦	🟦															
UI Design								🟦	🟦	🟦												
Chapter Module									🟦	🟦												
Main Screen Module										🟥	🟥											
Page Loader Module										🟥	🟥	🟥										
Banner Module										🟥	🟥											
Search/Filter Module													🟥									
Events Module														🟥								
Unit Testing							🟦	🟦	🟥	🟥	🟥	🟥	🟥	🟥								
Integration Testing												🟥	🟥	🟥	🟥	🟥						
Black Box Testing																	🟥					
Report Preparation															🟥	🟥	🟥	🟥	🟥			
Verifying Reports																				🟥	🟥	
Submission of Report																						🟥

WORK TO BE COMPLETED 🟥

WORK COMPLETED 🟦

Fig 8.1 – GANTT CHART 1

GANTT CHART-2

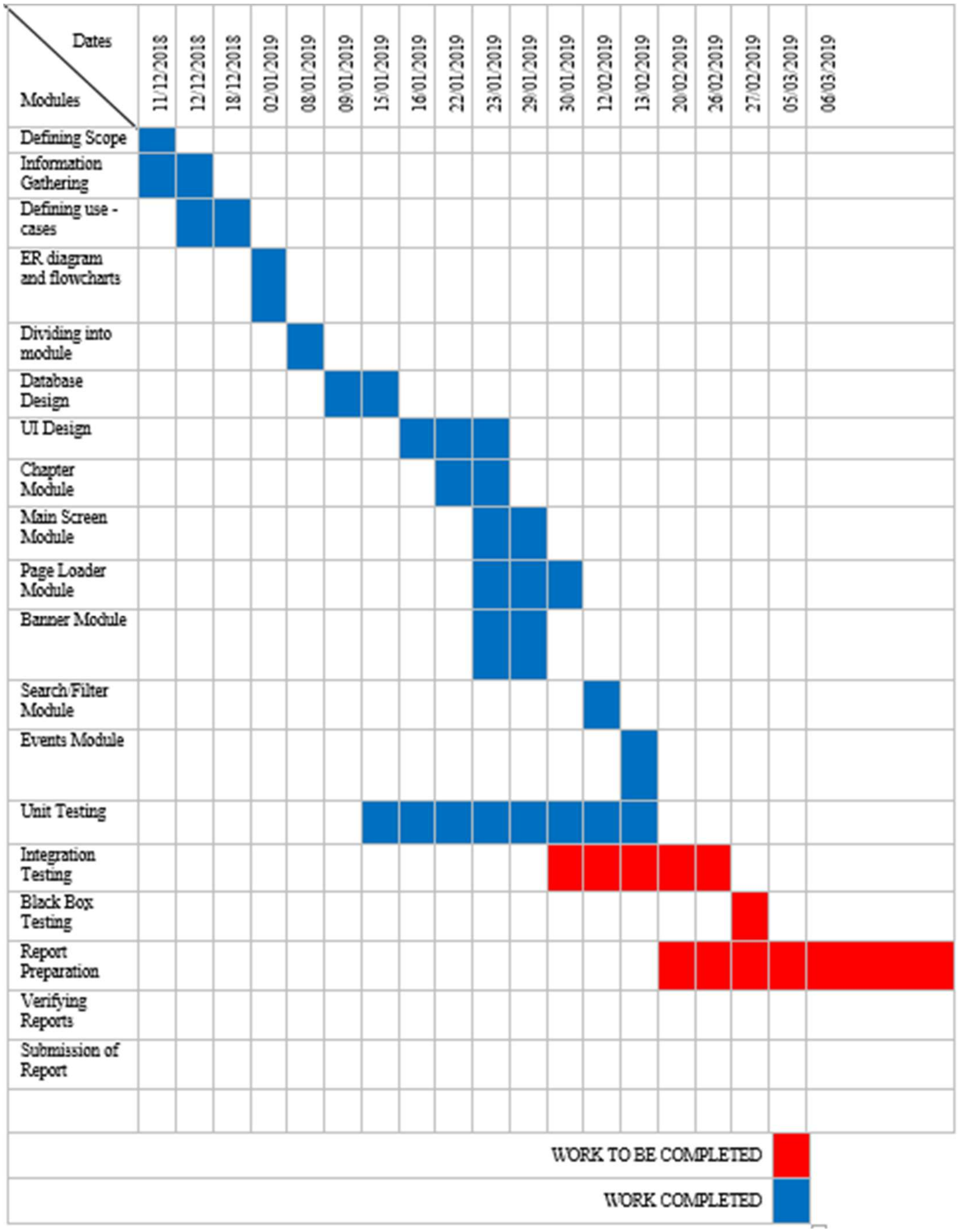


Dates Modules	11/12/2018	12/12/2018	18/12/2018	02/01/2019	08/01/2019	09/01/2019	15/01/2019	16/01/2019	22/01/2019	23/01/2019	29/01/2019	30/01/2019	12/02/2019	13/02/2019	20/02/2019	26/02/2019	27/02/2019	05/03/2019	06/03/2019
Defining Scope	🟦																		
Information Gathering	🟦	🟦																	
Defining use - cases		🟦	🟦																
ER diagram and flowcharts				🟦															
Dividing into module					🟦														
Database Design						🟦	🟦												
UI Design								🟦	🟦	🟦									
Chapter Module									🟦	🟦									
Main Screen Module										🟦	🟦								
Page Loader Module										🟦	🟦	🟦							
Banner Module										🟦	🟦								
Search/Filter Module													🟥						
Events Module														🟥					
Unit Testing							🟦	🟦	🟦	🟦	🟦	🟥	🟥	🟥					
Integration Testing												🟥	🟥	🟥	🟥	🟥			
Black Box Testing																	🟥		
Report Preparation															🟥	🟥	🟥	🟥	🟥
Verifying Reports																			
Submission of Report																			

WORK TO BE COMPLETED	🟥
WORK COMPLETED	🟦

Fig 8.2– GANTT CHART 2

GANTT CHART-3

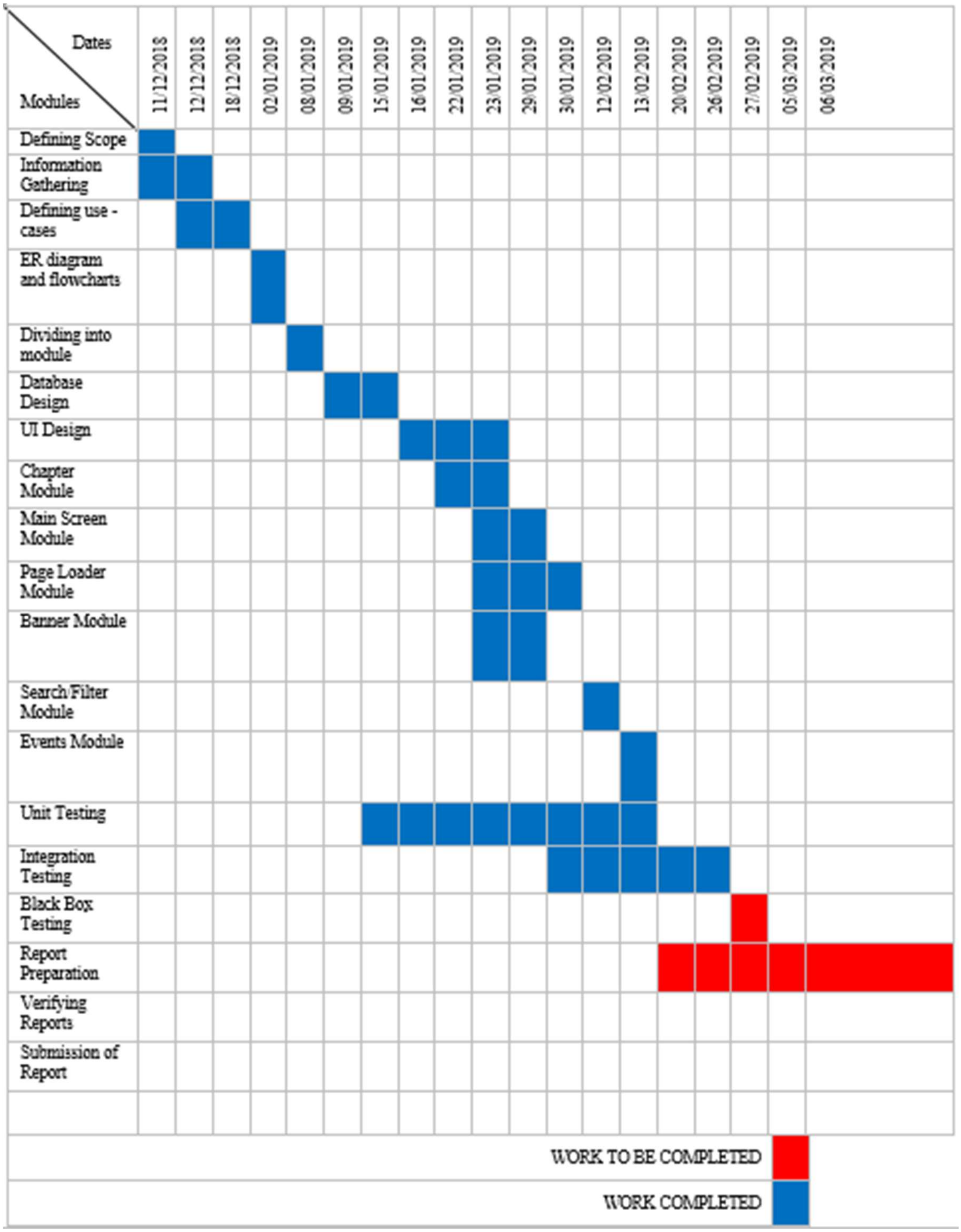


Dates Modules	11/12/2018	12/12/2018	18/12/2018	02/01/2019	08/01/2019	09/01/2019	15/01/2019	16/01/2019	22/01/2019	23/01/2019	29/01/2019	30/01/2019	12/02/2019	13/02/2019	20/02/2019	26/02/2019	27/02/2019	05/03/2019	06/03/2019
Defining Scope	🟦																		
Information Gathering	🟦	🟦																	
Defining use -cases		🟦	🟦																
ER diagram and flowcharts				🟦															
Dividing into module					🟦														
Database Design						🟦	🟦												
UI Design								🟦	🟦	🟦									
Chapter Module									🟦	🟦									
Main Screen Module										🟦	🟦								
Page Loader Module										🟦	🟦	🟦							
Banner Module										🟦	🟦								
Search/Filter Module													🟦						
Events Module														🟦					
Unit Testing							🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦					
Integration Testing												🟥	🟥	🟥	🟥	🟥			
Black Box Testing																	🟥		
Report Preparation															🟥	🟥	🟥	🟥	🟥
Verifying Reports																			
Submission of Report																			

WORK TO BE COMPLETED	🟥
WORK COMPLETED	🟦

Fig 8.3 – GANTT CHART 3

GANTT CHART-4



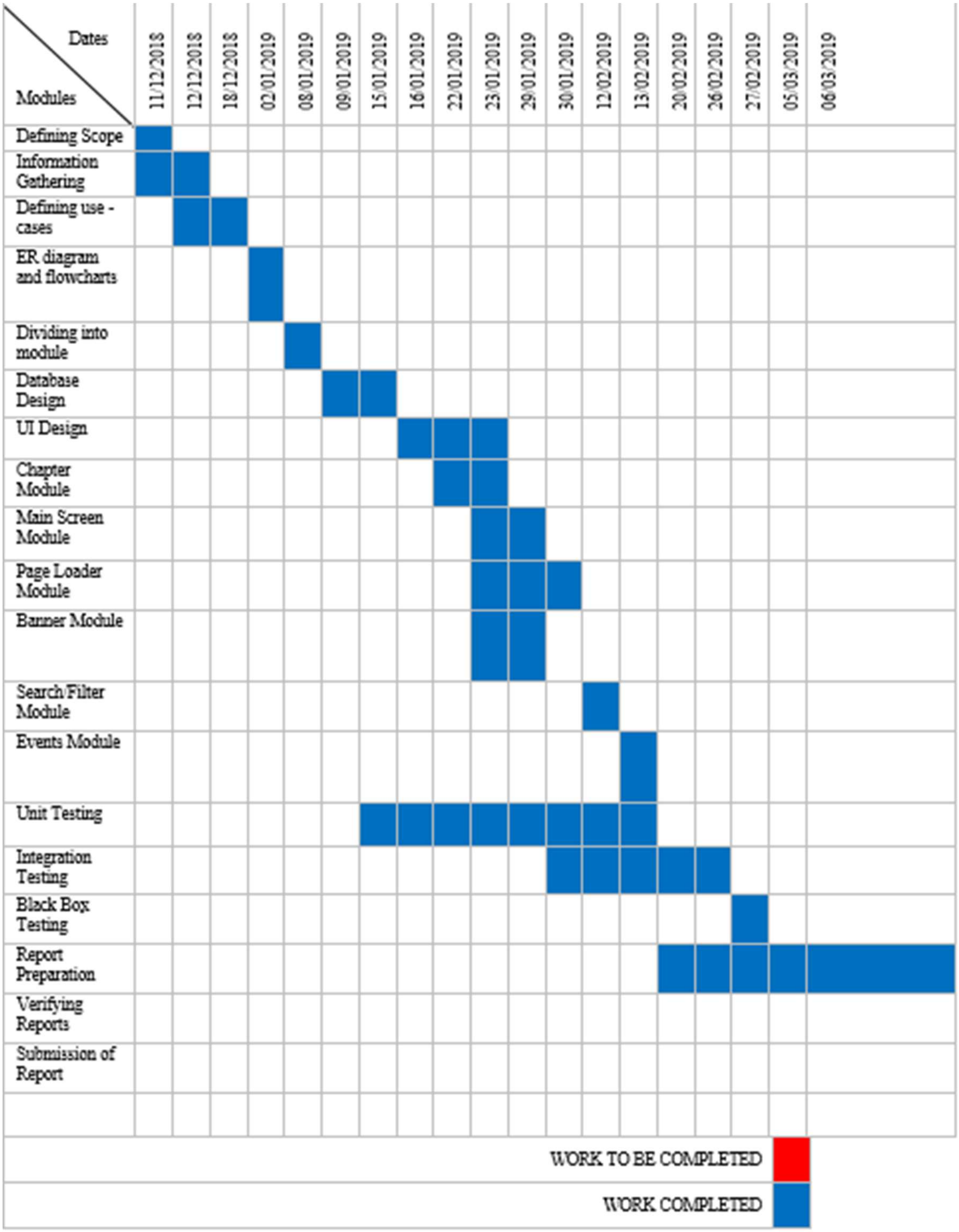
Dates Modules	11/12/2018	12/12/2018	18/12/2018	02/01/2019	08/01/2019	09/01/2019	15/01/2019	16/01/2019	22/01/2019	23/01/2019	29/01/2019	30/01/2019	12/02/2019	13/02/2019	20/02/2019	26/02/2019	27/02/2019	05/03/2019	06/03/2019
Defining Scope	🟦																		
Information Gathering	🟦	🟦																	
Defining use - cases		🟦	🟦																
ER diagram and flowcharts				🟦															
Dividing into module					🟦														
Database Design						🟦	🟦												
UI Design								🟦	🟦	🟦									
Chapter Module									🟦	🟦									
Main Screen Module										🟦	🟦								
Page Loader Module										🟦	🟦	🟦							
Banner Module										🟦	🟦								
Search/Filter Module													🟦						
Events Module														🟦					
Unit Testing							🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦					
Integration Testing												🟦	🟦	🟦	🟦	🟦			
Black Box Testing																	🟥		
Report Preparation															🟥	🟥	🟥	🟥	🟥
Verifying Reports																			
Submission of Report																			

WORK TO BE COMPLETED 🟥

WORK COMPLETED 🟦

Fig 8.4 – GANTT CHART 4

GANTT CHART-5



Dates Modules	11/12/2018	12/12/2018	18/12/2018	02/01/2019	08/01/2019	09/01/2019	15/01/2019	16/01/2019	22/01/2019	23/01/2019	29/01/2019	30/01/2019	12/02/2019	13/02/2019	20/02/2019	26/02/2019	27/02/2019	05/03/2019	06/03/2019
Defining Scope	🟦																		
Information Gathering	🟦	🟦																	
Defining use -cases		🟦	🟦																
ER diagram and flowcharts				🟦															
Dividing into module					🟦														
Database Design						🟦	🟦												
UI Design								🟦	🟦	🟦									
Chapter Module									🟦	🟦									
Main Screen Module										🟦	🟦								
Page Loader Module										🟦	🟦	🟦							
Banner Module										🟦	🟦								
Search/Filter Module													🟦						
Events Module														🟦					
Unit Testing							🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦					
Integration Testing												🟦	🟦	🟦	🟦	🟦			
Black Box Testing																	🟦		
Report Preparation															🟦	🟦	🟦	🟦	🟦
Verifying Reports																			
Submission of Report																			

WORK TO BE COMPLETED 🟥

WORK COMPLETED 🟦

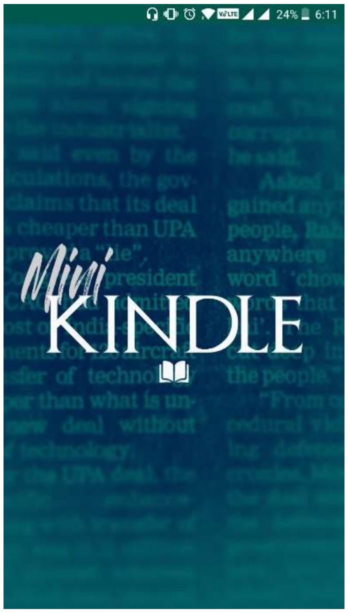
FIG 8.5 – GANTT CHART 5

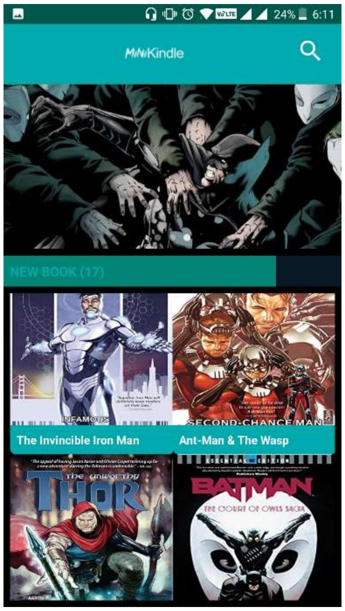
CHAPTER 9

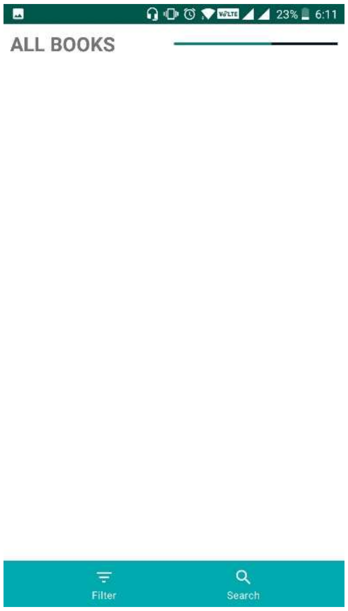
SNAPSHOTS

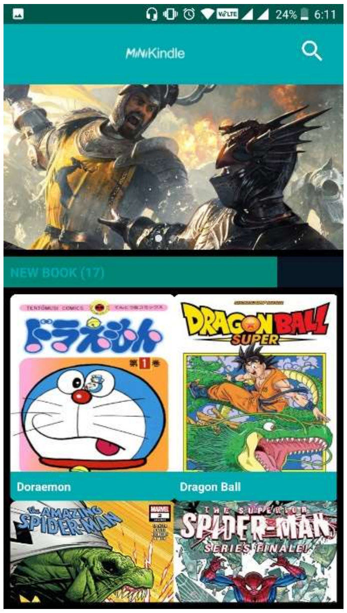
CHAPTER 9

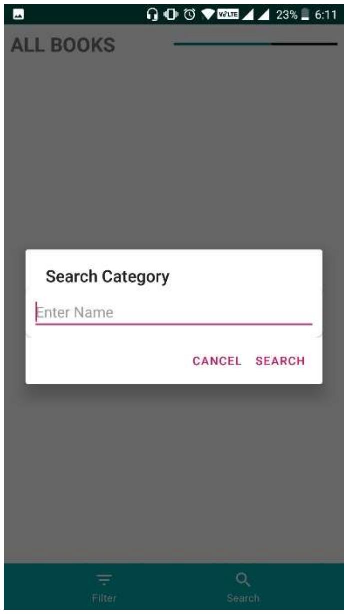
SNAPSHOTS

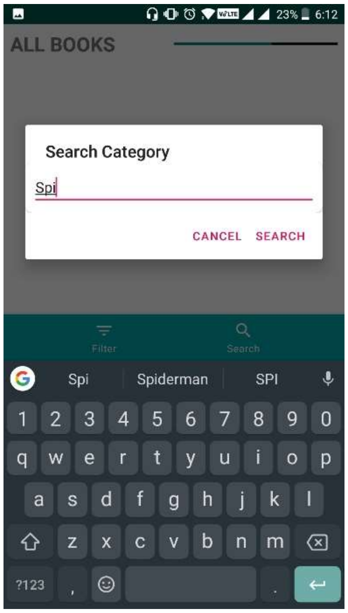


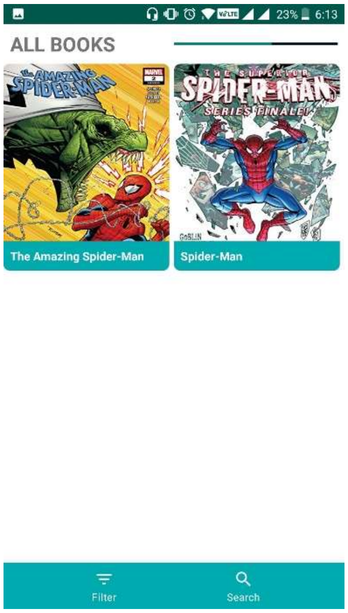


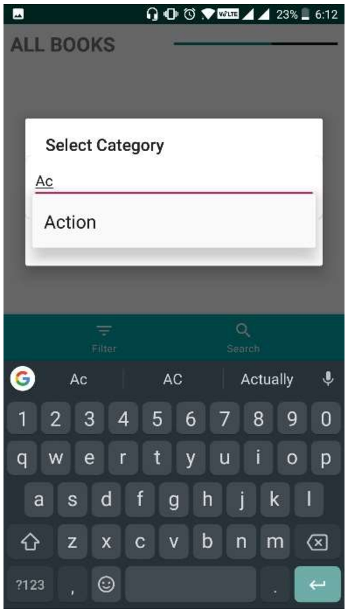


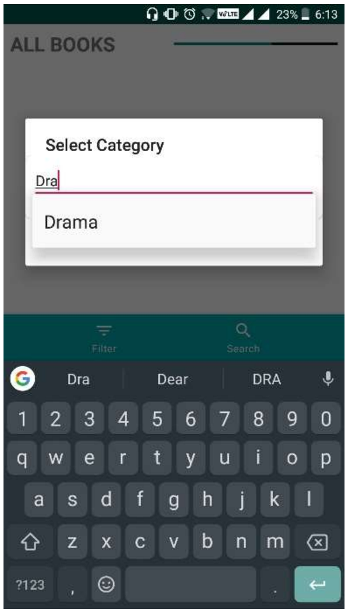


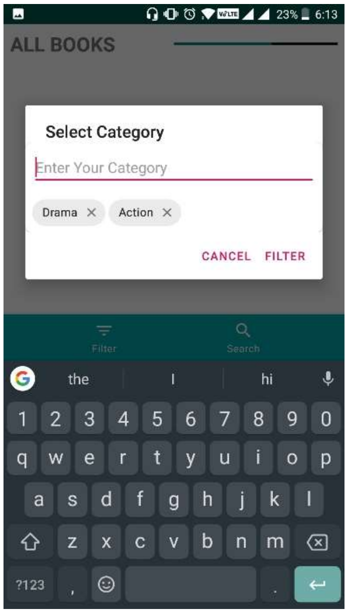

23% 6:11
ALL BOOKS
Search Category
Enter Name
CANCEL
SEARCH
Filter
Search

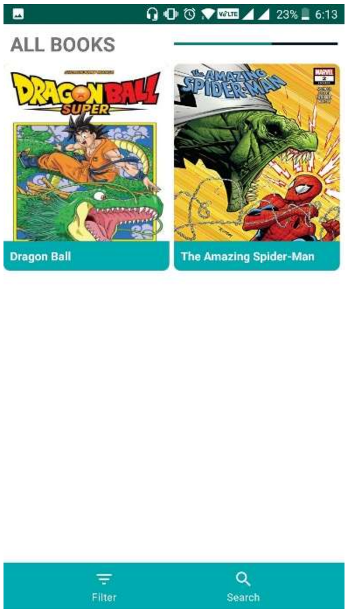

23% 6:12
ALL BOOKS
Search Category
Spi
CANCEL
SEARCH
Filter
Search
Spi
Spiderman
SPI
?123

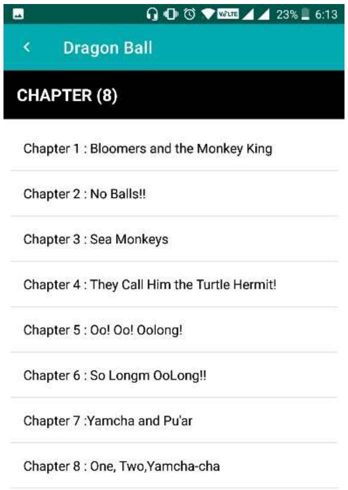

23% 6:13
ALL BOOKS
The Amazing Spider-Man
Spider-Man
Filter
Search


23% 6:12
ALL BOOKS
Select Category
Ac
Action
Filter
Search
Ac
AC
Actually
?123









CHAPTER (8)

Chapter 1 : Bloomers and the Monkey King

Chapter 2 : No Balls!!

Chapter 3 : Sea Monkeys

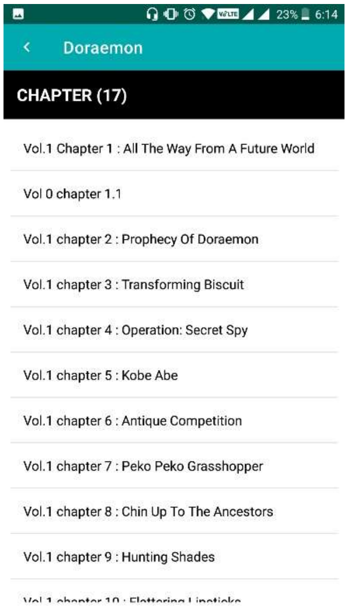
Chapter 4 : They Call Him the Turtle Hermit!

Chapter 5 : Oo! Oo! Oolong!

Chapter 6 : So Longm OoLong!!

Chapter 7 :Yamcha and Pu'ar

Chapter 8 : One, Two,Yamcha-cha



Doraemon

CHAPTER (17)

Vol.1 Chapter 1 : All The Way From A Future World

Vol 0 chapter 1.1

Vol.1 chapter 2 : Prophecy Of Doraemon

Vol.1 chapter 3 : Transforming Biscuit

Vol.1 chapter 4 : Operation: Secret Spy

Vol.1 chapter 5 : Kobe Abe

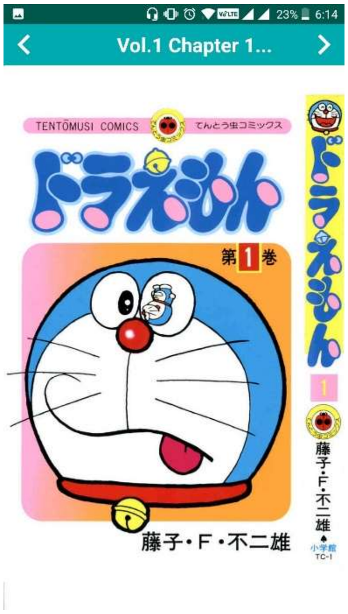
Vol.1 chapter 6 : Antique Competition

Vol.1 chapter 7 : Peko Peko Grasshopper

Vol.1 chapter 8 : Chin Up To The Ancestors

Vol.1 chapter 9 : Hunting Shades

Vol.1 chapter 10 : Flattering Lipsticks



Vol.1 Chapter 1...

CHAPTER 10

SOURCE CODE

```
package com.example.minikindle;

import android.content.Intent; import
android.os.Bundle;
import android.support.annotation.NonNull;
import android.support.v4.widget.SwipeRefreshLayout; import
android.support.v7.app.AppCompatActivity; import
android.support.v7.widget.GridLayoutManager; import
android.support.v7.widget.RecyclerView; import android.view.View;
import android.widget.ImageView; import
android.widget.TextView; import
android.widget.Toast;

import com.example.minikindle.Adapter.MyComicAdapter; import
com.example.minikindle.Adapter.MySliderAdapter; import
com.example.minikindle.Common.Common;
import com.example.minikindle.Interface.IBannerLoadDone; import
com.example.minikindle.Interface.IComicLoadDone; import
com.example.minikindle.Model.Comic;
import com.example.minikindle.Service.PicassoLoadingService; 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.util.ArrayList; import
java.util.List;

import dmax.dialog.SpotsDialog; import
ss.com.bannerslider.Slider;

public class MainActivity extends AppCompatActivity implements
IBannerLoadDone, IComicLoadDone {
      Slider slider;
      SwipeRefreshLayout swipeRefreshLayout; RecyclerView
      recycler_comic;
      TextView txt_comic; ImageView
      btn_filter_search;

      //Database
      DatabaseReference banners,comics;

      //Listener
```

```
    IBannerLoadDone            bannerListener;
    IComicLoadDone comicListener;


    android.app.AlertDialog alertDialog; @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        //init Database
        banners       =       FirebaseDatabase.getInstance().
getReference("Banners");
        comics = FirebaseDatabase.getInstance().getReference ("Comic");

        //init Listener bannerListener =
        this; comicListener = this;

        btn_filter_search          =          (ImageView)findViewById(R.id.
btn_show_filter_search);
        btn_filter_search.setOnClickListener(new View.
OnClickListener(){
            @Override
            public void onClick(View v){
                startActivity(new                    Intent(MainActivity.this,
FilterSearchActivity.class));
            }
        });

                slider  =  (Slider)  findViewById(R.id.slider);  Slider.init(new
        PicassoLoadingService());

        swipeRefreshLayout        =        (SwipeRefreshLayout)
findViewById(R.id.swipe_to_refresh);
        swipeRefreshLayout.setColorSchemeResources(R.color. colorPrimary,
                R.color.colorPrimaryDark);

        swipeRefreshLayout.setOnRefreshListener(new
SwipeRefreshLayout.OnRefreshListener() {
            @Override
            public    void    onRefresh()    {
                loadBanner();
```

```
                    loadComic();
                }
            });
            swipeRefreshLayout.post(new Runnable() { @Override
                public void run() {
                    loadBanner();
                    loadComic();
                }
            });

            recycler_comic = (RecyclerView)findViewById(R.id. recycler_comic);
            recycler_comic.setHasFixedSize(true);
            recycler_comic.setLayoutManager(new
GridLayoutManager(this,2));

            txt_comic = (TextView)findViewById(R.id.txt_comic);

        }

        private void loadComic() {
            //Show Dialog
            alertDialog = new SpotsDialog.Builder().setContext(
this)
                    .setCancelable(false)
                    .setMessage("Please Wait...")
                    .build();

            if(!swipeRefreshLayout.isRefreshing())
                alertDialog.show();

            comics.addListenerForSingleValueEvent(new
ValueEventListener() {
                List<Comic> comic_load = new ArrayList<>(); @Override
                public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
                    for(DataSnapshot comicSnapShot:dataSnapshot.
getChildren()){

                        Comic comic = comicSnapShot.getValue(
Comic.class);

                        comic_load.add(comic);

                    }
                    comicListener.onComicLoadDoneListener(
comic_load);
                  }

                  @Override
                  public void onCancelled(@NonNull DatabaseError databaseError) {
                        Toast.makeText(MainActivity.this, ""+
databaseError.getMessage(), Toast.LENGTH_SHORT).show();
                  }
            });
      }

      private void loadBanner() {
            banners.addListenerForSingleValueEvent(new
ValueEventListener() {
                  @Override
                  public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
                        List<String> bannerList = new ArrayList<>(); for(DataSnapshot
                        bannerSnapShot:dataSnapshot
.getChildren())
                        {
String.class);                    String image = bannerSnapShot.getValue(

                              bannerList.add(image);

                        }
                        //Call Listener
bannerList);            bannerListener.onBannerLoadDoneListener(
                   }


                  @Override
                  public void onCancelled(@NonNull DatabaseError databaseError) {
                        Toast.makeText(MainActivity.this, ""+
databaseError.getMessage(), Toast.LENGTH_SHORT).show();
                  }
            });

      }

      @Override
      public void onBannerLoadDoneListener(List<String> banners) {
            slider.setAdapter(new MySliderAdapter(banners));

        }

        @Override
        public void onComicLoadDoneListener(List<Comic> comicList) {
            Common.comicList = comicList;

            recycler_comic.setAdapter(new MyComicAdapter(
    getBaseContext(),comicList));

            txt_comic.setText(new StringBuilder("NEW BOOK (")
                    .append(comicList.size())
                    .append(")"));

            if(!swipeRefreshLayout.isRefreshing())
                alertDialog.dismiss();


        }
    }
```

```
package com.example.minikindle;

import android.content.Intent; import
android.os.Handler;
import android.support.v7.app.AppCompatActivity; import
android.os.Bundle;


public class SplashScreen extends AppCompatActivity { @Override
      protected void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);

            new Handler().postDelayed(new Runnable() { @Override
                  public void run() {
                        startActivity(new Intent(SplashScreen.this,
MainActivity.class));
                  }
            },3000);
      }
}
```

```
package com.example.minikindle;

import android.os.Build;
import android.support.annotation.RequiresApi; import
android.support.v7.app.AppCompatActivity; import android.os.Bundle;
import android.support.v7.widget.DividerItemDecoration; import
android.support.v7.widget.LinearLayoutManager; import
android.support.v7.widget.RecyclerView;
import android.view.View; import
android.widget.TextView;
import android.support.v7.widget.Toolbar;

import com.example.minikindle.Adapter.MyChapterAdapter; import
com.example.minikindle.Common.Common;
import com.example.minikindle.Model.Comic;


public class ChaptersActivity extends AppCompatActivity { RecyclerView


      recycler_chapter;
      TextView txt_chapter_name;
      LinearLayoutManager layoutManager;

      @RequiresApi(api = Build.VERSION_CODES.LOLLIPOP) @Override
      protected void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.activity_chapters);

            //View
            txt_chapter_name = (TextView)findViewById(R.id. txt_chapter_name);
            recycler_chapter = (RecyclerView)findViewById(R.id. recycler_chapter);
            recycler_chapter.setHasFixedSize(true); layoutManager = new
            LinearLayoutManager(this);
            recycler_chapter.setLayoutManager(layoutManager);
            recycler_chapter.addItemDecoration(new
DividerItemDecoration(this,layoutManager.getOrientation()));



            Toolbar toolbar = (Toolbar)findViewById(R.id.toolbar)
;
            toolbar.setTitle(Common.comicSelected.Name);
            //SetIcon toolbar.setNavigationIcon(R.drawable.
ic_chevron_left_black_24dp); toolbar.setNavigationOnClickListener(new View.
OnClickListener() {
                  @Override
                  public void onClick(View v) { finish();
                  }
            });

            fetchChapter(Common.comicSelected);
      }

      private void fetchChapter(Comic comicSelected) { Common.chapterList =
            comicSelected.Chapters; recycler_chapter.setAdapter(new
            MyChapterAdapter(
this,comicSelected.Chapters));
            txt_chapter_name.setText(new StringBuilder("CHAPTER
  (").append(comicSelected.Chapters.size()).append(")"));
      }
}
```

```
package com.example.minikindle;

import android.support.v4.view.ViewPager;  import
android.support.v7.app.AppCompatActivity; import android.os.Bundle;
import android.view.View; import
android.widget.TextView; import
android.widget.Toast;

import com.example.minikindle.Adapter.MyViewPagerAdapter; import
com.example.minikindle.Common.Common;
import com.example.minikindle.Model.Chapter; import
com.wajahatkarim3.easyflipviewpager. BookFlipPageTransformer;


public class ViewComicActivity extends AppCompatActivity { ViewPager viewPager;
    TextView txt_chapter_name;
    View back,next;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_view_comic);


        viewPager = (ViewPager)findViewById(R.id.view_pager)
;
        txt_chapter_name = (TextView)findViewById(R.id.
txt_chapter_name);
        back = findViewById(R.id.chapter_back); next =
        findViewById(R.id.chapter_next);

        back.setOnClickListener(new View.OnClickListener() { @Override
            public void onClick(View view) {
                if(Common.chapterIndex == 0)
                    Toast.makeText(ViewComicActivity.this, " You Are Reading
First Chapter", Toast.LENGTH_SHORT).show();
                else
                {
                    Common.chapterIndex--;
                    fetchLinks(Common.chapterList.get(Common
.chapterIndex));
                 }
            }
        });
        next.setOnClickListener(new View.OnClickListener() { @Override
            public void onClick(View view) { if(Common.chapterIndex ==
                Common.chapterList.
size()-1)
                    Toast.makeText(ViewComicActivity.this, "
You Are Reading Last Chapter", Toast.LENGTH_SHORT).show(); else
                {
                    Common.chapterIndex++;
                    fetchLinks(Common.chapterList.get(Common
.chapterIndex));
                 }


            }
        });
        fetchLinks(Common.chapterSelected);
    }

    private void fetchLinks(Chapter chapter) { if(chapter.Links
        != null)
        {
            if(chapter.Links.size() > 0)
            {
                MyViewPagerAdapter adapter = new
MyViewPagerAdapter(getBaseContext(),chapter.Links);
                viewPager.setAdapter(adapter);

                txt_chapter_name.setText(Common.formatString
(Common.chapterSelected.Name));

                //Animation
                BookFlipPageTransformer
bookFlipPageTransformer = new BookFlipPageTransformer();
                bookFlipPageTransformer.
setScaleAmountPercent(10f);
                viewPager.setPageTransformer(true,
bookFlipPageTransformer);
            }
            else{
                Toast.makeText(this, "No Image Here", Toast.
LENGTH_SHORT).show();
            }


        }
        else{
            Toast.makeText(this, "This Chapter Is
Translating...", Toast.LENGTH_SHORT).show();
```

```
package com.example.minikindle;
 import android.content.DialogInterface; import
 android.support.annotation.NonNull; import
 android.support.design.chip.Chip; import
 android.support.design.chip.ChipGroup;
 import android.support.design.widget.BottomNavigationView; import
 android.support.v7.app.AlertDialog;
 import android.support.v7.app.AppCompatActivity; import
 android.os.Bundle;
 import android.support.v7.widget.GridLayoutManager; import
 android.support.v7.widget.RecyclerView; import
 android.view.LayoutInflater;
 import android.view.MenuItem; import
 android.view.View;
 import android.widget.AdapterView; import
 android.widget.ArrayAdapter;
 import android.widget.AutoCompleteTextView; import
 android.widget.EditText;
 import android.widget.TextView; import
 android.widget.Toast;

 import com.example.minikindle.Adapter.MyComicAdapter; import
 com.example.minikindle.Common.Common;
 import com.example.minikindle.Model.Comic;

 import java.util.ArrayList; import
 java.util.Collection; import
 java.util.Collections; import
 java.util.List;

 public class FilterSearchActivity extends AppCompatActivity
 {

       BottomNavigationView bottomNavigationView; RecyclerView
       recycler_filter_search;


       @Override
       protected void onCreate(Bundle savedInstanceState) {
             super.onCreate(savedInstanceState);
             setContentView(R.layout.activity_filter_search);

             recycler_filter_search = (RecyclerView) findViewById
 (R.id.recycler_filter_search);
```

```
        recycler_filter_search.setHasFixedSize(true);
        recycler_filter_search.setLayoutManager(new
GridLayoutManager(this, 2));

        bottomNavigationView = (BottomNavigationView)
findViewById(R.id.bottom_nav);
        bottomNavigationView.inflateMenu(R.menu.main_menu);
        bottomNavigationView.
setOnNavigationItemReselectedListener(new
BottomNavigationView.OnNavigationItemReselectedListener() {
            @Override
            public void onNavigationItemReselected(@NonNull MenuItem menuItem)
{
                switch (menuItem.getItemId()) { case
                    R.id.action_filter:
                        showFiltersDialog(); break;
                    case R.id.action_search:
                        showSearchDialog(); break;
                    default:
                        break;
                }
            }
        });
    }

    private void showSearchDialog() {
        final AlertDialog.Builder alertDialog = new
AlertDialog.Builder(FilterSearchActivity.this);
        alertDialog.setTitle("Search Category");

        final LayoutInflater inflater = this.
getLayoutInflater();
        final View search_layout = inflater.inflate(R.layout
.dialog_search, null);

        final EditText edt_search = (EditText) search_layout
.findViewById(R.id.edt_search);

        alertDialog.setView(search_layout);
        alertDialog.setNegativeButton("CANCEL", new
DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterfac
dialogInterface, int which) {
                        dialogInterface.dismiss();
                    }
                    });
                alertDialog.setPositiveButton("SEARCH",new
 DialogInterface.OnClickListener(){
                    @Override
                            public void onClick(DialogInterface dialogInterface,int i){
                                fetchSearchComic(edt_search.getText(
 ).toString());
                        }
                });
                alertDialog.show();
        }
        private void fetchSearchComic(String query){ List<Comic> comic_search
                = new ArrayList<>(); for(Comic comic:Common.comicList)
                {
                    if(comic.Name.contains(query))
                        comic_search.add(comic);
                }
                if(comic_search.size() > 0)
                    recycler_filter_search.setAdapter(new
 MyComicAdapter(getBaseContext(),comic_search)); else
                    Toast.makeText(this, "No result", Toast.
 LENGTH_SHORT).show();

        }

        private void showFiltersDialog() {
                final AlertDialog.Builder alertDialog = new
 AlertDialog.Builder(FilterSearchActivity.this);
                alertDialog.setTitle("Select Category");

                final LayoutInflater inflater = this.
 getLayoutInflater();
                final View filter_layout = inflater.inflate(R.layout
 .dialog_options, null);

                final AutoCompleteTextView txt_category = ( AutoCompleteTextView)
 filter_layout.findViewById(R.id. txt_category);
                final ChipGroup chipGroup = (ChipGroup)
```

```
filter_layout.findViewById(R.id.chipGroup);

            //Create Autocomplete
            ArrayAdapter<String> adapter = new ArrayAdapter<>( this,
android.R.layout.select_dialog_item, Common.categories
);
            txt_category.setAdapter(adapter);
            txt_category.setOnItemClickListener(new AdapterView.
OnItemClickListener() {
                              @Override
                              public void onItemClick(AdapterView<?> parent, View
view, int position, long id) {
                                    //Clear txt_category.setText("");

                                      //Create tags
                                    Chip chip = (Chip)inflater.inflate(R
.layout.chip_item,null,false);
                                    chip.setText(((TextView)view).
getText());
                                    chip.setOnCloseIconClickListener(new
 View.OnClickListener() {
                                          @Override
                                          public void onClick(View view) {
                                              chipGroup.removeView(view);
                                          }
                                     });
                                    chipGroup.addView(chip);
                               }
                        });

                        alertDialog.setView(filter_layout);
                        alertDialog.setNegativeButton("CANCEL", new
DialogInterface.OnClickListener() {
                              @Override
                              public void onClick(DialogInterface
dialogInterface, int which) {
                                    dialogInterface.dismiss();
                               }
                        });
                        alertDialog.setPositiveButton("FILTER", new
DialogInterface.OnClickListener() {
                                @Override
                              public void onClick(DialogInterface
dialog, int which) {
 ArrayList<>();
                                     List<String> filter_key = new StringBuilder
 StringBuilder("");
                                     filter_query = new for(int j=0;j<chipGroup.
 getChildCount();j++)
                                     {
                                           Chip chip = (Chip)chipGroup.
 getChildAt(j);
                                           filter_key.add(chip.getText().
 toString());
                                     }
                                     //Because in our Database Chapters
 are sorted A->Z
                                     //So we need sort our filter_key
                                     Collections.sort(filter_key);
                                     //Convert list to string for(String
                                     key:filter_key)
                                     {
                                           filter_query.append(key).append(
 ",");
                                     }

                                     //Remove Last ","
                                     filter_query.setLength(filter_query.
 length()-1);


                                     //Filter by this query
                                     fetchFilterCategory(filter_query.
 toString());
                                }
                         });

                         alertDialog.show();
                   }

       private void fetchFilterCategory(String query) { List<Comic>
             comic_filtered = new ArrayList<>(); for(Comic
             comic:Common.comicList){
                   if(comic.Category !=null)
                         if(comic.Category.contains(query))
                         comic_filtered.add(comic);
             }
             if(comic_filtered.size() > 0)
                   recycler_filter_search.setAdapter(new
 MyComicAdapter(getBaseContext(),comic_filtered)); else
Toast.makeText(this, "No result", Toast.
 LENGTH_SHORT).show();
       }

 }
```

CHAPTER 11

CONCLUSION

CHAPTER 11

CONCLUSION

Mini Kindle is a simple application to let the users read the magazines,books or comics whenever they want or depends what they want.

The filter search is available with the specified requirements in the provided system.

This decreases almost all the disadvantages of searching a book for longer time.

This system helps the user to provide suggestions and read all the required book available at the recent versions .

Provides security to the user's information.

Mini Kindle lets the user to interact with an easy UI, letting them interact with the application easily.

CHAPTER 12

FUTURE SCOPE

CHAPTER 12

FUTURE SCOPE

Future Scope means the forms our project can take in the future. It’s the prediction of the successes of our project.

The Future Scope of our project is as follow:

- User will be able to provide a bookmark in the future updates of the project.
- User will be able to save the books offline.
- Extending this application by providing Authorization service.
- Sending user the updates via Email or a text message.
- Making the application compatible on various platforms i.e. iOS, windows,etc.

CHAPTER 13

BIBLIOGRAPHY

CHAPTER 13

BIBLIOGRAPHY

a) **www.google.com**

b) **www.wikipedia.org**

c) **www.tutorialpoints.com**

d) **www.w3schools.com**

e) **www.w3schools.in**

f) www.youtube.com

g) www.youtube.com

h) **http://www.pcspl.in/**

CHAPTER 14

TESTING

CHAPTER 14

TESTING

13.1 Testing Details

Testing is a verification process for quality assessment and improvement. Testing is basically done to find errors, faults in the system. The basic goal of software development process is to produce the software that has very few or no errors. In an effort to detect errors soon after they are introduced each phase ends with verification activity such as reviews. However most of these verification activities in the early phase of the software development are based on human evaluation and cannot detect all the errors. Testing plays an important role in quality assurance for the software. It is a dynamic method for the verification and validation, where the system to be tested is executed and the behavior of the system is observed.

13.1.1 Black box testing

Black-box testing is a method ofsoftware testing that examines the functionality of an application (e.g. what the software does) without peering into its internal structures or workings (see white-box testing). This method of test can be applied to virtually every level of software testing: unit, integration system and acceptance. It typically comprises most if not all higher level testing, but can also dominate unit testing as well.

13.1.2 White box testing

White-box testing (also known as clear box testing, glass box testing, transparent box testing, and structural testing) is a method of testing software that tests internal structures or workings of an application, as opposed to its functionality (i.e. black-box testing). In white-box testing an internal perspective of the system, as well as programming skills, are used to design test cases. The tester chooses inputs to exercise paths through the code and determine the appropriate outputs.

13.1.3 Unit testing

Unit testing, also known as component testing, refers to tests that verify the functionality of a specific section of code, usually at the function level. In an object-oriented environment, this is usually at the class level, and the minimal unit tests include the constructors and destructors.

These types of tests are usually written by developers as they work on code (white-box style), to ensure that the specific function is working as expected. One function might have multiple tests, to catch corner cases or other branches in the code. Unit testing alone cannot verify the functionality of a piece of software, but rather is used to ensure that the building blocks of the software work independently from each other.

Unit testing is a software development process that involves synchronized application of a broad spectrum of defect prevention and detection strategies in order to reduce software development risks, time, and costs. It is performed by the software developer or engineer during the construction phase of the software development life cycle. Rather than replace traditional QA focuses, it augments it. Unit testing aims to eliminate construction errors before code is promoted to QA; this strategy is intended to increase the quality of the resulting software as well as the efficiency of the overall development and QA process.

Depending on the organization's expectations for software development, unit testing might include static code analysis, data flow analysis, metrics analysis, peer code reviews, code coverage analysis and other software verification practices.

13.1.4 Integration Testing

Integration testing is any type of software testing that seeks to verify the interfaces between components against a software design. Software components may be integrated in an iterative way or all together ("big bang"). Normally the former is considered a better practice since it allows interface issues to be located more quickly and fixed.

Integration testing works to expose defects in the interfaces and interaction between integrated components (modules).

13.1.5 Component interface testing

The practice of component interface testing can be used to check the handling of data passed between various units, or subsystem components, beyond full integration testing between those units. The data being passed can be considered as "message packets" and the range or data types can be checked, for data generated from one unit, and tested for validity before being passed into another unit. One option for interface testing is to keep a separate log file of data items being passed, often with a time stamp logged to allow analysis of thousands of cases of data passed between units for days or weeks. Tests can include checking the handling of some extreme data values while other interface variables are passed as normal values. Unusual data values in an interface can help explain unexpected performance in the next unit. Component interface testing is a variation of black-box testing, with the focus on the data values beyond just the related actions of a subsystem component.

13.1.6 System testing

System testing, or end-to-end testing, tests a completely integrated system to verify that it meets its requirements.For example, a system test might involve testing a logon interface, then creating and editing an entry, plus sending or printing results, followed by summary processing or deletion (or archiving) of entries, then logoff.

In addition, the software testing should ensure that the program, as well as working as expected, does not also destroy or partially corrupt its operating environment or cause other processes within that environment to become inoperative (this includes not corrupting shared memory, not consuming or locking up excessive resources and leaving any parallel processes unharmed by its presence).

13.1.7 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 public. The focus of this testing is to simulate real users by using blackbox and whitebox techniques.

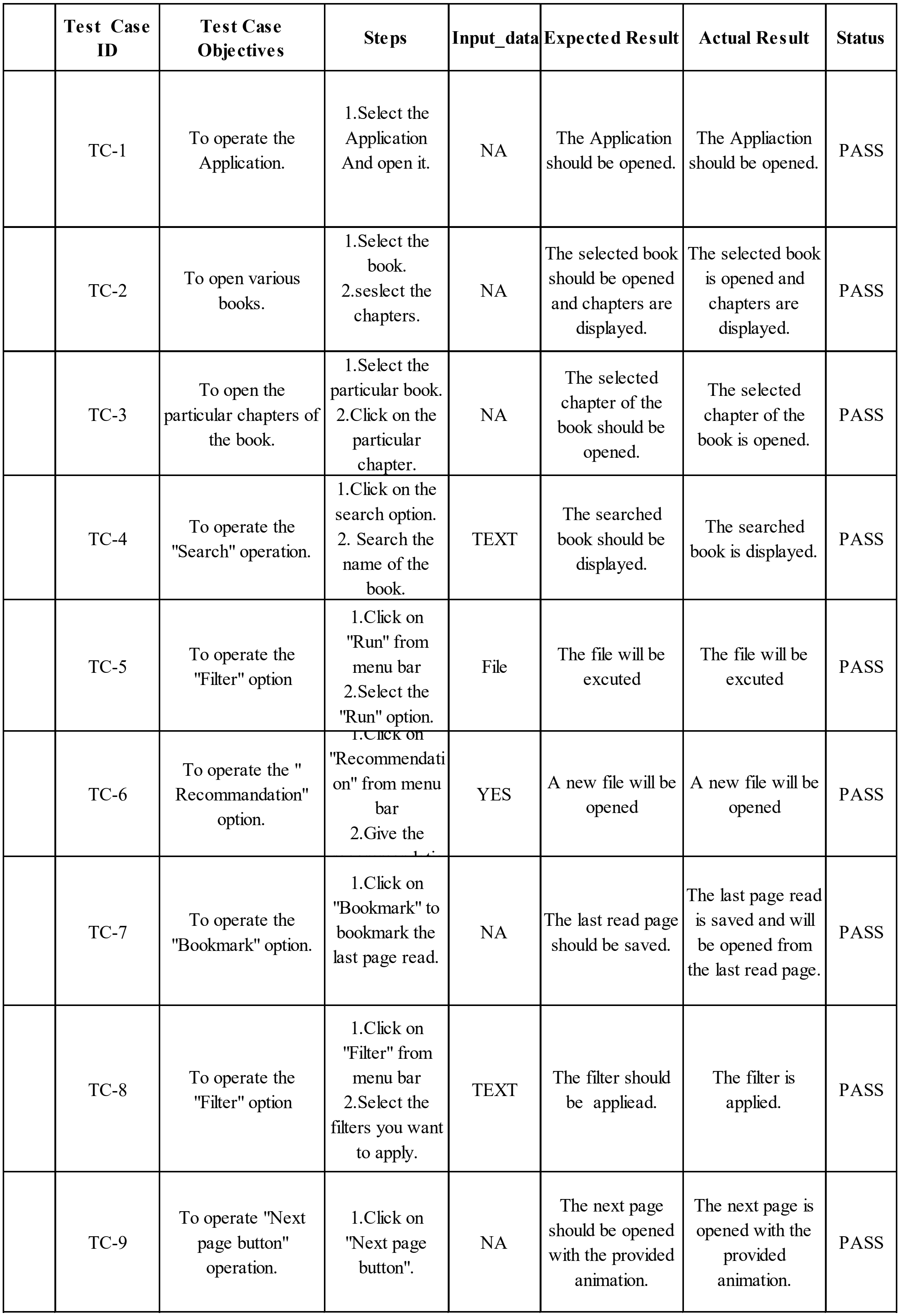
The aim is to carry out the tasks that a typical user might perform. Alpha testing is carried out in a lab environment and usually the testers are internal employees of the organization. To put it as simple as possible, this kind of testing is called alpha only because it is done early on, near the end of the development of the software, and before beta testing.

13.1.8 Beta Testing

Beta Testing of a product is performed by "real users" of the software application in a "real environment" and can be considered as a form of external user acceptance testing.

Beta version of the software is released to a limited number of end-users of the product to obtain feedback on the product quality. Beta testing reduces product failure risks and provides increased quality of the product through customer validation.

It is the final test before shipping a product to the customers. Direct feedback from customers is a major advantage of Beta Testing. This testing helps to tests the product in real time environment.



	Test Case ID	Test Case Objectives	Steps	Input_data	Expected Result	Actual Result	Status
	TC-1	To operate the Application.	1.Select the Application And open it.	NA	The Application should be opened.	The Appliaction should be opened.	PASS
	TC-2	To open various books.	1.Select the book. 2.seslect the chapters.	NA	The selected book should be opened and chapters are displayed.	The selected book is opened and chapters are displayed.	PASS
	TC-3	To open the particular chapters of the book.	1.Select the particular book. 2.Click on the particular chapter.	NA	The selected chapter of the book should be opened.	The selected chapter of the book is opened.	PASS
	TC-4	To operate the "Search" operation.	1.Click on the search option. 2. Search the name of the book.	TEXT	The searched book should be displayed.	The searched book is displayed.	PASS
	TC-5	To operate the "Filter" option	1.Click on "Run" from menu bar 2.Select the "Run" option.	File	The file will be excuted	The file will be excuted	PASS
	TC-6	To operate the " Recommandation" option.	1.Click on "Recommendation" from menu bar 2.Give the	YES	A new file will be opened	A new file will be opened	PASS
	TC-7	To operate the "Bookmark" option.	1.Click on "Bookmark" to bookmark the last page read.	NA	The last read page should be saved.	The last page read is saved and will be opened from the last read page.	PASS
	TC-8	To operate the "Filter" option	1.Click on "Filter" from menu bar 2.Select the filters you want to apply.	TEXT	The filter should be applicad.	The filter is applied.	PASS
	TC-9	To operate "Next page button" operation.	1.Click on "Next page button".	NA	The next page should be opened with the provided animation.	The next page is opened with the provided animation.	PASS

CHAPTER 15

BUG REPORT

CHAPTER 15

BUG REPORT

GROUP. **BUG:** - #0001

SOFTWARE: **MiniKindle (Online Library App)**

RELEASE VERSION: 1.0

TESTER: Sagar Soni **DATE:** 25-02-2019

ASSIGNED TO: Atharva Keny **PRIORITY:** 1

TITLE: App does not refresh after swipe up.

DESCRIPTION:

When the user swipes up to refresh the home screen to load new books into recommendations, it continues to load and then eventually crash.

RESOLUTION: FIXED

DATE RESOLVED: 29-02-2019 **RESOLVED BY: Atharva Keny**

VERSION: 1.0

RETESTED BY: Atharva Keny **VERSION TESTED:**1.0

RETEST COMMENT: THE BUG WAS FIXED PROPERLY AND THERE ARE NO MORE ISSUES OF VALIDATION.

SIGNATURES

ORIGINATOR: _______________ TESTER: _______________

PROGRAMMER: _______________ PROJECT MANAGER: _______________