WOMEN'S CHRISTIAN COLLEGE, CHENNAI – 06.

A CLASSIC CORNER

A PROJECT REPORT

Submitted by

GLENYS ABIGAIL SHAMITHA

16BCA10

in partial fulfillment for the award of the degree of

BACHELOR OF COMPUTER APPLICATIONS

Under the guidance of

Mrs. D. Sylvia Mary

DEPARTMENT OF COMPUTER APPLICATIONS (BCA)

WOMEN'S CHRISTIAN COLLEGE, CHENNAI-06.

MARCH 2019.

CERTIFICATE

This is to certify that the project work titled "A CLASSIC CORNER", is a bonafide work done by GLENYS ABIGAIL SHAMITHA (16BCA10), in partial fulfillment of the requirement for the award of the degree of Bachelor of Computer Applications during the academic year 2018-2019.

Date	**External Examiner**	**Internal Guide**

Submitted for the Viva-Voce Examination held on 25 – 03 – 2019.

ACKNOWLEDGEMENT

I thank and praise God almighty for the wonderful life that I am blessed with. I thank Him for guiding me through each and every step that I have taken in my life, and especially during the past few months when I was working on this project. I am grateful for His grace that guided me through this period of time.

I extend my deepest gratitude to our esteemed Principal **Dr. Lilian I Jasper**, for providing the means to attain this goal of mine, by giving me permission and equipping me with the necessary material to do the project work.

I am most delighted and honored to express my sincere gratitude to **Mrs. Sylvia Mary, M.C.A., M.Phil., NET.,** Head of the Department of Computer Applications (BCA), for her consent, guidance and support. I specially thank her for being my guide and for the continuous constructive criticism and encouragement offered by her. I am also grateful to all the other staff members who readily helped me with any doubts I had.

Last but never the least, I thank all of my family members and friends, for their constant support and encouragement that enabled me to work hard, persevere and succeed.

ABSTRACT

The development of the ‘A Classic Corner’ took place with the intention of enabling readers to access information of any book - which falls under the ‘Classics’ genre - with ease. This is the ultimate destination for bibliophiles. This app, gives the user an illusion of walking through the Classics’ Hall of Fame. It provides a summary of the chosen novel, along with more details such as publication date, the best edition and so on. Thus, giving the user a little insight, on what to expect.

The app is themed in a special way, to make the user feel like they have travelled back in time, while using the app. From various editions to original illustrations, this application has everything one needs to get started on a journey through the pages. Designed with simplicity and clear labels that lead to the expected information is the application. Every page has a unique quality that makes the user feel like they have entered the Victorian era with just a single click. This will in turn display the content pertaining to the title chosen, thus commencing the journey.

TABLE OF CONTENTS

1. INTRODUCTION

1.1 Problem Definition

These days, most bibliophiles depend on Google to refer the plot summary before they dive into the novel. They prefer to read reviews from websites they trust. Influenced by other readers with similar taste, they move to the next step of purchasing the book. Though we have stepped into the digital era, with various applications to make every task easier, we rely on Google's search engine for this function.

The main advantage of applications, that are not based in the cloud, is the fact that they can be accessed anywhere, at any time, without the requirement of an Internet connection. If one were to have a catalogue that could be referred to, when in need of a second opinion, it would simplify the process of logging into the system, and navigating through the phone to Google in order to search for the specific title.

As mentioned above, in the digital age, the digitized version of such a catalogue is expected. This is where the application 'A Classic Corner' comes in.
Created using Android Studio, this application is easy to download and use. Either a millennial or a technology challenged adult can find their way through this app with ease. The user can directly access the content of the application by just clicking the desired book.

Many book apps present the content of the book along with reviews and ratings. If one is wondering why this application does not, it is because this application is for the people who enjoy holding the novel between their hands as the unique scent from the pages wafts through the air, resulting in good read.

2. SYSTEM REQUIREMENTS

2.1 Hardware and Software Requirements

Hardware Requirements

Processor	**:**	Intel Core i5-7200U
RAM	**:**	8 GB
Hard Disk	**:**	201 GB
Monitor	**:**	15” Color Monitor
Mouse	**:**	3 Button Optical Mouse
Keyboard	**:**	104 Keys

Software Requirements

Operating System	**:**	Windows 10
Backend	**:**	SQLite in Android Studio / SQLite Studio
Programming Language	**:**	Java
Frontend	**:**	Android Studio 3.0.1

2.2 Technical Profile

Android

Android is a software stack for mobile devices that includes an operating system, middleware and key applications. Google Inc. purchased the initial developer of the software, Android Inc., in 2005.Android's mobile operating system is based on the Linux kernel. Google and other members of the Open Handset Alliance collaborated on Android's development and release.

The Android Open Source Project (AOSP) is tasked with the maintenance and further development of Android. The Android operating system is the world's bestselling Smartphone platform.

The Android SDK provides the tools and APIs necessary to begin developing applications Android platform using the Java programming language. Android has a large community of developers writing applications ("apps") that extend the functionality of the devices. There are currently over 250,000 apps available for Android Features.

- **Application framework** enabling reuse and replacement of components
- **Dalvik virtual machine** optimized for mobile devices
- **Integrated browser** based on the open source WebKit engine
- **Optimized graphics** powered by a custom 2D graphics library; 3D graphics based on the OpenGL ES 1.0 specification (hardware acceleration optional)
- **SQLite** for structured data storage
- **Media support** for common audio, video, and still image formats (MPEG4, H.264, MP3, AAC, AMR, JPG, PNG, GIF)
- **GSM Telephony** (hardware dependent)
- **Bluetooth, EDGE, 3G, and WiFi** (hardware dependent)
- **Camera, GPS, compass, and accelerometer** (hardware dependent)

- **Rich development environment** including a device emulator, tools for debugging, memory and performance profiling, and a plugin for the Eclipse IDE.

Libraries

Android includes a set of C/C++ libraries used by various components of the Android system. These capabilities are exposed to developers through the Android application framework. Some of the core libraries are listed below:

- **System C library** - a BSD-derived implementation of the standard C system library (libc), tuned for embedded Linux-based devices
- **Media Libraries** - based on PacketVideo'sOpenCORE; the libraries support playback and recording of many popular audio and video formats, as well as static image files, including MPEG4, H.264, MP3, AAC, AMR, JPG, and PNG
- **Surface Manager** - manages access to the display subsystem and seamlessly composites 2D and 3D graphic layers from multiple applications
- **LibWebCore** - a modern web browser engine which powers both the Android browser and an embeddable web view
- **SGL** - the underlying 2D graphics engine
- **3D libraries** - an implementation based on OpenGL/ES 1.0 APIs; the libraries use either hardware 3D acceleration (where available) or the included, highly optimized 3D software rasterizer
- **FreeType** - bitmap and vector font rendering
- **SQLite** - a powerful and lightweight relational database engine available to all applications

Android Runtime

Android includes a set of core libraries that provides most of the functionality available in the core libraries of the Java programming language.

Every Android application runs in its own process, with its own instance of the Dalvik virtual machine. Dalvik has been written so that a device can run multiple VMs efficiently. The Dalvik VM executes files in the Dalvik Executable (.dex) format that is optimized for minimal memory footprint. The VM is register based, and runs classes compiled by a Java language compiler that have been transformed into the .dex format by the included "dx" tool.

The Dalvik VM relies on the Linux kernel for underlying functionality.

Android Operating System

Android is an operating system based on Linux with a Java programming interface. It provides tools e.g., a compiler, debugger and a device emulator as well as its own Java Virtual machine (Dalvik Virtual Machine - DVM). Android is created by the Open Handset Alliance, which is led by Google.

Android uses a special virtual machine, e.g. the Dalvik Virtual Machine. Dalvik uses special bytecode. Therefore, you cannot run standard Java bytecode on Android. Android provides a tool "dx" which allows converting Java Class files into "dex" (Dalvik Executable) files. Android applications are packed into an .apk (Android Package) file by the program "aapt" (Android Asset Packaging Tool) To simplify development Google provides the Android Development Tools (ADT) for Eclipse. The ADT performs automatically the conversion from class to dex files and creates the apk during deployment.

Android supports 2-D and 3-D graphics using the OpenGL libraries and supports data storage in a SQLite database.

Every Android applications runs in its own process and under its own userid, whichis generated automatically by the Android system during deployment. Therefore, the application is isolated from other running applications and a misbehaving application cannot easily harm other Android applications.

Downloading the Android SDK

To set up the SDK, the following is done. If the Android SDK already exists, it must be updated to the latest tools or platform using the ***Android SDK and AVD Manager***, rather than downloading a new SDK starter package.

Here is an overview of the steps one must follow to set up the Android SDK:

1. Prepare your development computer and ensure it meets the system requirements.
2. Install the SDK starter package from the table above. (If you are on Windows, download the installer for help with the initial setup.)
3. Install the ADT Plugin for Eclipse (if you will be developing in Eclipse).
4. Add Android platforms and other components to your SDK.
5. Explore the contents of the Android SDK (optional).

Android Studio

Android Studio is an integrated development environment (IDE) from Google that provides developers with tools needed to build applications for the Android OS platform. Android Studio is available for download on Windows, Mac, and Linux. A one-time, $25 developer's license is required to publish apps to Google Play App Store. The foundation for Android Studio is based on IntelliJ IDEA.

The Android Studio IDE is free to download and use. It has a rich UI development environment with templates to give new developers a launching pad into Android development. Developers will find that Android Studio gives them the tools to build phone and tablet solutions as well as emerging technology solutions for Android TV, Android Wear, Android Auto, Glass and additional contextual models.

Android Studio is intended to be used by development teams as small as one person or as large as global teams. The Android Studio IDE can be linked to larger teams with GIT or similar version control services for larger teams. Mature Android developers will find tools that are necessary for large teams to deliver solutions rapidly to their customers.

Android solutions can be developed using either Java or C++in Android Studio. The workflow for Android Studio is built around the concept of Continuous integration. Continuous Integration allows for teams to test their code each and every time a developer checks in their work.

Issues can be captured and reported to the team immediately. The concept of continuously checking codes provides actionable feedback to the developers with the goal of releasing versions of a mobile solution faster to the Google Play App Store. To this end, there is rigorous support for LINT tools, Pro-Guard and App Signing tools.

Performance tools provide access to view how well an Android application package file (APK) is going. The performance and profiling tools display a color coded image to show how often the same pixel is drawn on a screen to reduce rendering overhead. The GPU rendering shows how well the app does in maintaining Google's 16-ms-per-frame benchmark. Memory tools visualize where and when the app will use too much system RAM and when garbage collection occurs, battery analysis tools present how much drain one is placing on the device.

Features

Below are some of the features of Android Studio:

- **Visual layout editor**

Create complex layouts with `ConstraintLayout` by adding constraints from each view to other views and guidelines. Then preview your layout on any screen size by selecting one of various device configurations or by simply resizing the preview window.

- **APK Analyzer**

Find opportunities to reduce your Android app size by inspecting the contents of your app APK file, even if it wasn't built with Android Studio. Inspect the manifest file, resources, and DEX files. Compare two APKs to see how your app size changed between app versions.

- **Fast emulator**

Install and run your apps faster than with a physical device and simulate different configurations and features, including ARCore, Google's platform for building augmented reality experiences.

- **Intelligent code editor**

Write better code, work faster, and be more productive with an intelligent code editor that provides code completion for Kotlin, Java, and C/C++ languages.

- **Flexible build system**

Powered by Gradle, Android Studio's build system allows you to customize your build to generate multiple build variants for different devices from a single project.

- **Realtime profilers**

The built-in profiling tools provide realtime statistics for your app's CPU, memory, and network activity. Identify performance bottlenecks by recording method traces, inspecting the heap and allocations, and see incoming and outgoing network payloads.

System Requirements for Android Studio

For Windows

- Microsoft® Windows® 7/8/10 (32- or 64-bit)
- 3 GB RAM minimum, 8 GB RAM recommended; plus 1 GB for the Android Emulator
- 2 GB of available disk space minimum,
  4 GB Recommended (500 MB for IDE + 1.5 GB for Android SDK and emulator system image)
- 1280 x 800 minimum screen resolution

For Mac

- Mac® OS X® 10.10 (Yosemite) or higher, up to 10.13 (macOS High Sierra)

- 3 GB RAM minimum, 8 GB RAM recommended; plus 1 GB for the Android Emulator
- 2 GB of available disk space minimum,
  4 GB Recommended (500 MB for IDE + 1.5 GB for Android SDK and emulator system image)
- 1280 x 800 minimum screen resolution

For Linux

- GNOME or KDE desktop

  Tested on Ubuntu® 14.04 LTS, Trusty Tahr (64-bit distribution capable of running 32-bit applications)
- 64-bit distribution capable of running 32-bit applications
- GNU C Library (glibc) 2.19 or later
- 3 GB RAM minimum, 8 GB RAM recommended; plus 1 GB for the Android Emulator
- 2 GB of available disk space minimum,
  4 GB Recommended (500 MB for IDE + 1.5 GB for Android SDK and emulator system image)
- 1280 x 800 minimum screen resolution

Android Studio 3.3

Android Studio 3.3 includes these new enhancements & features:

Develop

- Navigation Editor
- IntelliJ 2018.2.2 Platform Update
- Kotlin 1.3.11 Update
- Clang-Tidy Support for C++
- New Project Wizard update

- Delete Unused IDE Directories
- IDE User Feedback

Build

- Improved incremental Java compilation when using annotation processors
- Lazy task configuration:
- Single-variant project sync
- Android App Bundles now supports Instant Apps

Test

- Multiple Emulator AVD instance Launch
- Android 9 Pie - Emulator System Images
- Emulator Snapshot Save Speed Improvement

Optimize

- Profiler Performance Improvements
- Memory Profiler: Allocation Tracking Options
- Network Profiler: Formatted Text
- CPU Profiler: Frame Rendering Data

SQLite

SQLite is a relational database management system contained in a C programming library. In contrast to many other databases management systems, SQLite is not a client-server database engine. Rather, it is embedded into the end program.

SQLite is a popular choice as embedded database software for local/client storage in application software such as web browsers. It is arguably the most widely deployed database engine, as it used today by several widespread browsers, operating systems, and embedded systems (such as mobile phones) and so on. SQLite has bindings to many programming languages.

Android SQLite

SQLite is an open source SQL database that stores data to a text file on a device. Android comes with built-in SQLite database implementation. SQLite supports all the relational database features. In order to access this database, one does not need to establish any kind of connections for it like JDBC, ODBC etc.

Database – Package

The main package is android.database.SQLite that contains the classes to manage your own databases.

Database – Creation

In order to create a database the user just needs to call the method openOrCreateDatabase with the database name and mode as a parameter. It returns an instance of SQLite database which the user has to receive in a created object

Database – Insertion

A table can be created or have data inserted into it using the execSQL method defined in SQLite Database class.

Database – Helper class

For managing all the operations related to the database, a helper class has been given and is called SQLiteOpenHelper. It automatically manages the creation and updating of the database.

SQLite Studio

- **SQLite Studio** is a SQLite database manager with the following features:
- **Portable -** no need to install or uninstall. Just download, unpack and run.
- **Intuitive interface**,
- **Powerful**, yet **light** and **fast**,
- All **SQLite3** and **SQLite2** features wrapped within **simple GUI**,

- **Cross-platform** - runs on Windows 9x/2k/XP/2003/Vista/7, Linux, MacOS X and should work on other Unixes (not tested yet).
- **Exporting** to various formats (SQL statements, CSV, HTML, XML, PDF, JSON),
- **Importing** data from various formats (CSV, custom text files [regular expressions]),
- Numerous small additions, like **formatting code**, **history of queries** executed in editor windows, on-the-fly syntax checking, and more,
- **Unicode support**,
- **Skinnable** (interface can look native for Windows 9x/XP, KDE, GTK, Mac OS X, or draw widgets to fit for other environments, WindowMaker, etc),
- **Configurable** colors, fonts and shortcuts.
- **Open source and free** - Released under GPLv3 license.

SQLite is a software library that provides a relational database management system. The ‘lite’ in SQLite means light weight in terms of setup, database administration, and required resource. SQLite has the following noticeable features:

i) serverless

ii) self-contained

iii) zero-configuration

iv) transactional

Serverless

Normally, an RDBMS such as MySQL, PostgreSQL, etc., requires a separate server process to operate. The applications that want to access the database server use TCP/IP protocol to send and receive requests. This is called client/server architecture.

Self-Contained

SQLite is self-contained means it requires minimal support from the operating system or external library. This makes SQLite usable in any environments especially in embedded devices like iPhones, Android phones, game consoles, handheld media players, etc.

SQLite is developed using ANSI-C. The source code is available as a big sqlite3.c and its header file sqlite3.h. If you want to develop an application that uses SQLite, you just need to drop these files into your project and compile it with your code.

Zero-configuration

Because of the serverless architecture, you don't need to "install" SQLite before using it. There is no server process that needs to be configured, started, and stopped.
In addition, SQLite does not use any configuration files.

Transactional

All transactions in SQLite are fully ACID-compliant. It means all queries and changes are Atomic, Consistent, Isolated, and Durable. In other words, all changes within a transaction take place completely or not at all even when an unexpected situation like application crash, power failure, or operating system crash occurs.

SQLite uses dynamic types for tables. It means you can store any value in any column, regardless of the data type. SQLite allows a single database connection to access multiple database files simultaneously. This brings many nice features like joining tables in different databases or copying data between databases in a single command.
SQLite is capable of creating in-memory databases which are very fast to work with.

3. SYSTEM DESIGN

System design is the process of defining elements of a system like modules, architecture, components and their interfaces and data for a system based on the specified requirements. It is the process of defining, developing, and designing systems, which satisfies the specific needs and requirements.

3.1 Database Design

Database design is the process of mapping out a detailed data model of the database to be created. This data model must contain all the needed logical and physical design choices and physical storage parameters needed to generate a design in a data definition language, which can then be used to create the database.

Table Name: **book_table**

Column Name	Data Type	Size	Constraints
book_name	varchar	100	Not Null
book_id	varchar	10	Primary Key
book_details	varchar	500	Not Null

4. PROJECT DESCRIPTION

The basic function of this app is to provide the user with the desired details about the chosen novel. The modules have been designed with simplicity and sophistication. Each page is distinctively themed to compliment the novel’s plot. The buttons that point to the next frame are all marked with clarity.

Pros and Cons

Pros

- ➢ The color scheme and background image for every page
- ➢ Simple yet sophisticated design of the application
- ➢ Extra details about each novel
- ➢ Direct link to Amazon.in
- ➢ Displays a summary for each novel

Cons

- ➢ Does not display the entire novel
- ➢ Does not display a list of the available novels
- ➢ Displays details of only selected novels

Modules

- ❖ Title Page
- ❖ List of Books
- ❖ Book Information

Title Page

This is the first screen that the user will see when they open the application. The customization of the application to make the user feel like they have entered a ‘Classic’ application, begins with the background of the first screen. This image is of an old scroll that was used for writing in the olden days.

The page consists of the title of the application, placed in a TextView element. Below this, a Button element is placed with the label ‘Go To Books’. This button has a function that is called when the button is clicked. The function of this button is to lead the user to the next module which is the List of Books.

List of Books

The second module is the list of all the novels whose information the application is equipped with. This module consists of a ScrollView element, with which the user can review all the novels. To direct the user to the information of the desired novel, image buttons, with the image as the novel’s front cover, is placed inside the ScrollView element. Accessing the information can be done by clicking the image button according to the user’s choice. The click of the button immediately leads the user to the next module, which is a customized activity for every novel.

Book Information

This page contains all the details of the book. It is themed according to each novel. A specially chosen background image is added to the activity, to customize it according to the novel’s theme. The background image is used to represent an important part of the novel, or what the novel is based around. Each page also has a color scheme to match the background image of the activity. This complements the design of the application.

In the beginning of the activity, there is a TextView element placed to display the title of the novel, along with the author’s name. Below this, there is a Button element placed with the label ‘Click for Summary’. When the user clicks this button, they will be able to view the summary for that particular novel. This button has an extra feature which is, to display the and hide the text, by clicking the same button i.e. by clicking the button once the summary appears, and if the user clicks the button again, the summary disappears.

The next section of the page is the part that has database connectivity. This is to display More Details of the book. In an EditText element, the title of the novel is typed and a Button element which is placed beneath it is clicked. The label of this Button element is ‘Search’. Clicking the button will call the function of the button, which is to access the database and display the extra information that the user has requested. As soon as the button is clicked the database is accessed and the additional details for the novel, which was added to the database, will appear in a TextView element placed above the EditText element.

The last element on the page is a Button element, with the label ‘Buy Book’. This button, has a function, which is to direct the user to the Amazon.in website, when it is clicked. This is done using an Intent.

All the above-mentioned elements on the page are placed inside a ScrollView element. The structure of this module, remains the same for every novel. This creates a uniform yet customized feel of the application.

5. TESTING AND IMPLEMENTATION

Taxonomy of Testing

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub-assemblies, assemblies and/or a finished product. It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

Types of Tests

Unit Testing

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

Integration Testing

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent.

Functional Testing

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centred on the following items:

Valid Input	: identified classes of valid input must be accepted.
Invalid Input	: identified classes of invalid input must be rejected.
Functions	: identified functions must be exercised.
Output	: identified classes of application outputs must be exercised.
Systems/Procedures	: interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

System Testing

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration-oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

- **White Box Testing**

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

- **Black Box Testing**

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box you cannot ―see‖ into it. The test provides inputs and responds to outputs without considering how the software works

Acceptance Testing

After the developer has completed all rounds of testing and he/she is satisfied with the system, then the user takes over and re-test the system from his/her point of view to judge whether it is acceptable according to some previously identified criteria. This is almost always a tricky situation in the project because of the inherent conflict between the developer and the user.

6. FUTURE ENHANCEMENTS

This project has been developed to benefit the users who prefer to read a book than an e-book. After careful consideration and research, we have designed the app with specifications that will be appreciated by the targeted group of bibliophiles. It provides the right number of features that such a reader may except in an application. However, to further our study we would like to expand and test our boundaries by:

- Adding more novels
- Customizing each page with more details
- Adding more features, such as an option for adding reviews about the novel.

These options can be included in the application if the group's votes are unanimous. In other words, if the targeted group of people wish to add some of the features and not all, their opinion will be taken into consideration, and the changes will be made accordingly.

In the future, we also wish to give our users an exclusive access to an account that they can create on the application. We also look forward to expanding our users, so that we can be partners with a publication house or any website online, in order to shower our users with added benefits such as discounts on the price of certain books, preordered books being delivered the day it is released and many more exciting features.

7. CONCLUSION

As mentioned previously, the idea for the development of this application relied on the desire of bibliophiles to extract information about the novels from a particular genre, without the option of reading it online, or on the same application, or without being persuaded to purchase an E-book. This was achieved by creating a state-of-the-art book-related application, that provides the users with the basic details of a chosen novel – under the classic's genre – like a short summary, details about the publication and best edition and a link that will redirect the user to 'amazon.in', where they can purchase the book. Each page was successfully designed to complement the novel it represents, through the background images and layouts.

The application was tested thoroughly by all the suggested methods of testing. The code though simple, proved to function well like a well-oiled machine. All in all, the application was developed and implemented successfully and further enhancements for its betterment are being studied.

8. BIBLIOGRAPHY

Book References

- Griffiths, Dawn & Griffiths, David (2017). Head First Android Development. USA, Massachusetts: O'Reilly Media Inc.
- Lee, Wei-Meng (2012). Beginning Android Application Development. USA, New Jersey: Wiley.

Web References

- https://stackoverflow.com/
- https://developer.android.com/
- https://www.enotes.com/
- https://en.wikipedia.org/
- https://www.britannica.com/
- https://www.shmoop.com/
- https://www.sparknotes.com/
- https://www.thebestnotes.com/
- https://www.gradesaver.com/
- https://www.chiefessays.net/
- https://www.bl.uk/
- https://www.schoolbytes.com/

9. APPENDICES

9.1 Sample Code

activity_mail.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity"
    android:orientation="vertical"
    android:background="@drawable/bgparchment">
    <ScrollView
        android:layout_width="match_parent"
        android:layout_height="match_parent">
      <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_margin="@android:dimen/notification_large_icon_height"
        android:orientation="vertical">
     <ImageButton
     android:id="@+id/peterbtn"
     android:layout_width="252dp"
     android:layout_height="324dp"
     android:onClick="peterClick"
     android:src="@drawable/peterpan" />
```

```
<TextView
    android:id="@+id/tv1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/t1" />
<ImageButton
    android:id="@+id/pridebtn"
    android:layout_width="210dp"
    android:layout_height="335dp"
    android:onClick="prideClick"
    android:src="@drawable/pride" />
<TextView
    android:id="@+id/tv4"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/t3" />
<ImageButton
    android:id="@+id/mobybtn"
    android:layout_width="249dp"
    android:layout_height="380dp"
    android:onClick="mobyClick"
    android:src="@drawable/moby" />
<TextView
    android:id="@+id/tv8"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/t8" />
    <ImageButton
        android:id="@+id/draculabtn"
        android:layout_width="260dp"
        android:layout_height="360dp"
        android:onClick="draculaClick"
        android:src="@drawable/dracula" />
    <TextView
        android:id="@+id/tv11"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/t11" />
</LinearLayout>
</ScrollView>
</LinearLayout>
```

MainActivity.java

```
package com.example.aclassiccorner;
import android.content.Intent;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);   }
    public void peterClick(View view)
    {
        Intent i = new Intent (this, peter_details.class);
        startActivity(i);
    }
    public void prideClick(View view)
    {
        Intent i = new Intent(this, pride_details.class);
        startActivity(i);
    }
    public void mobyClick(View view)
    {
        Intent i = new Intent(this, moby_details.class);
        startActivity(i);
    }
    public void draculaClick(View view)
    {
        Intent i = new Intent(this, dracula_details.class);
        startActivity(i);
    }
```

DatabaseOpenHelper.java

```
package com.example.aclassiccorner;
import android.content.Context;
import com.readystatesoftware.sqliteasset.SQLiteAssetHelper;
public class DatabaseOpenHelper extends SQLiteAssetHelper {
    private static final String DATABASE_NAME="bookapp2.db";
    private static final int DATABASE_VERSION=1;
    public DatabaseOpenHelper (Context context) {
        super (context,DATABASE_NAME, null, DATABASE_VERSION);
    }
}
```

DatabaseAccess.java

```
package com.example.aclassiccorner;
import android.content.Context;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;
public class DatabaseAccess {
    private SQLiteOpenHelper openHelper;
    private SQLiteDatabase db;
    private static DatabaseAccess instance;
    Cursor c = null;
    private DatabaseAccess (Context context)  {
        this.openHelper= new DatabaseOpenHelper(context);
    }
    public static DatabaseAccess getInstance(Context context)  {
        if (instance==null)  {
            instance= new DatabaseAccess(context);
        }     return instance;
    }
    public void  open ()  {
        this.db=openHelper.getWritableDatabase();       }
    public void close()  {
        if(db!=null)  {
            this.db.close();
        }  }
    public String getBookContent(String name)  {
        c=db.rawQuery("select book_details from book_table where book_name =
'"+name+"'", new String[] {});
        StringBuffer buffer = new StringBuffer();
        while (c.moveToNext()){
            String bookdet = c.getString(0);
            buffer.append(""+bookdet);                    }
        return buffer.toString();
    }  }
```

activity_peter_details.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout 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=".peter_details"
    android:background="@drawable/peterpanbg">
    <ScrollView
        android:layout_width="match_parent"
        android:layout_height="match_parent">
    <LinearLayout
```

```
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_margin="@android:dimen/notification_large_icon_height"
android:orientation="vertical">
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginBottom="40dp"
    android:background="@android:color/black"
    android:text="@string/t1"
    android:textColor="@android:color/white" />
<Button
    android:id="@+id/B1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:background="@android:color/black"
    android:text="@string/click"
    android:textColor="@android:color/white" />
<TextView
    android:id="@+id/Text1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="73dp"
    android:background="@android:color/black"
    android:text="@string/sum "
    android:textColor="@android:color/white" />
<TextView
```

```
    android:id="@+id/display1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="73dp"
    android:background="@android:color/black"
    android:text="@string/more"
    android:textColor="@android:color/white" />
<EditText
    android:id="@+id/edit1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
android:layout_marginTop="@android:dimen/notification_large_icon_height"
    android:background="@android:color/black"
    android:text="Peter Pan"
    android:textColor="@android:color/white" />
<Button
    android:id="@+id/search1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="54dp"
    android:background="@android:color/black"
    android:text="@string/search"
    android:textColor="@android:color/white" />
<Button
    android:id="@+id/buybook"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
```

```
            android:layout_marginTop="54dp"
            android:layout_marginBottom="80dp"
            android:background="@android:color/black"
            android:text="@string/buy"
            android:textColor="@android:color/white" />
        </LinearLayout>
        </ScrollView>
        </RelativeLayout>
```

peter_details.java

```
package com.example.aclassiccorner;
import android.content.Intent;
import android.net.Uri;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
import java.io.IOException;
import java.io.InputStream;
public class peter_details extends AppCompatActivity {
Button btn,buybook,B1;
EditText name;
TextView details,Text1;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_peter_details);
buybook = (Button)findViewById(R.id.buybook);
B1 = (Button) findViewById(R.id.B1);
Text1=(TextView)findViewById(R.id.Text1);
        btn = (Button) findViewById(R.id.search1);
        name = (EditText) findViewById(R.id.edit1) ;
        details = (TextView) findViewById(R.id.display1);
        btn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                DatabaseAccess dbAccess =
DatabaseAccess.getInstance(getApplicationContext());
                dbAccess.open();
                String n = name.getText().toString();
                String moredetails = dbAccess.getBookContent(n);
                details.setText(moredetails);
            }});
        B1.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                String text="";
                try{
                    InputStream is = getAssets().open("peterdoc.txt");
                    int size = is.available();
                    byte [] buffer = new byte[size];
                    is.read(buffer);
                    is.close();
                    text= new String(buffer);
                }
                catch(IOException ex)  {
                    ex.printStackTrace();
                }
                Text1.setText(text);
                Text1.setVisibility((Text1.getVisibility() == View.VISIBLE)
                        ? View.GONE : View.VISIBLE);
            }});
        buybook.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent buy= new Intent(Intent.ACTION_VIEW,
Uri.parse("https://www.amazon.in/"));
                startActivity(buy);
            }});
      }}
```

Strings.xml

```
<resources>
    <string name="app_name">A Classic Corner</string>
    <string name="t1">PETER PAN by J. M. Barrie </string>
    <string name="t3">PRIDE AND PREJUDICE by Jane Austen </string>
    <string name="t8">MOBY DICK by Herman Melville </string>
    <string name="t11">DRACULA by Bram Stoker </string>
    <string name="sum"> Summary </string>
    <string name="click"> Click for Summary </string>
    <string name="buy"> Buy Book </string>
    <string name="search"> Search </string>
    <string name="more"> MORE DETAILS </string>
</resources>
```

AndroidManifest.xml

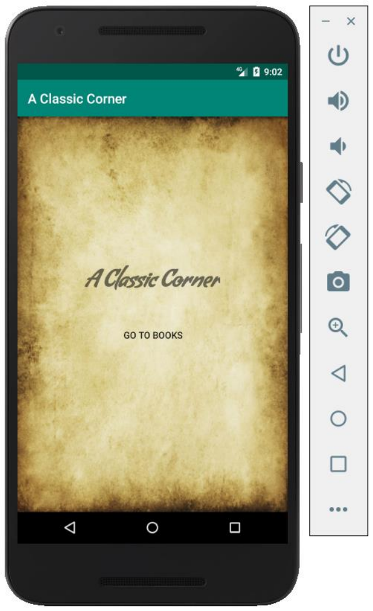
```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.aclassiccorner">
    <uses-permission android:name="android.permission.INTERNET" />
</manifest>
```

build.gradle (Module: app)

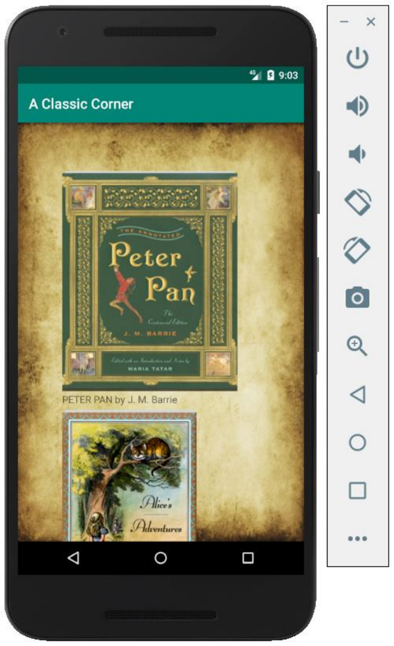
```
dependencies {
    implementation fileTree(dir: 'libs', include: ['*.jar'])
    implementation 'com.android.support:appcompat-v7:28.0.0'
    implementation 'com.android.support.constraint:constraint-layout:1.1.3'
    testImplementation 'junit:junit:4.12'
    androidTestImplementation 'com.android.support.test:runner:1.0.2'
    androidTestImplementation 'com.android.support.test.espresso:espresso-core:3.0.2'

    implementation 'com.readystatesoftware.sqliteasset:sqliteassethelper:+'
}
```

9.2 Screenshots

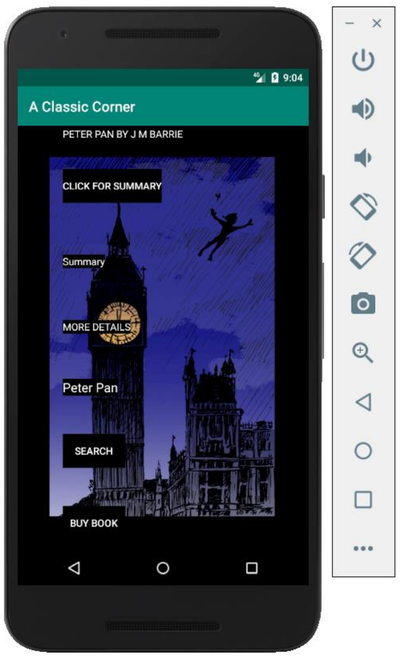
Title Page

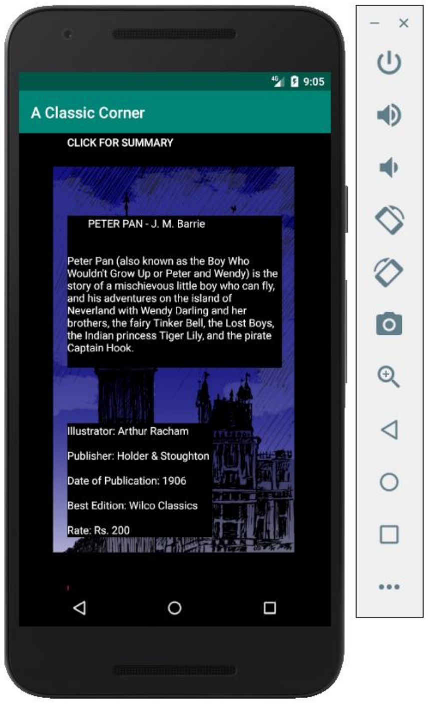


List of Books



Book Information





PETER PAN - J. M. Barrie

Peter Pan (also known as the Boy Who Wouldn't Grow Up or Peter and Wendy) is the story of a mischievous little boy who can fly, and his adventures on the island of Neverland with Wendy Darling and her brothers, the fairy Tinker Bell, the Lost Boys, the Indian princess Tiger Lily, and the pirate Captain Hook.

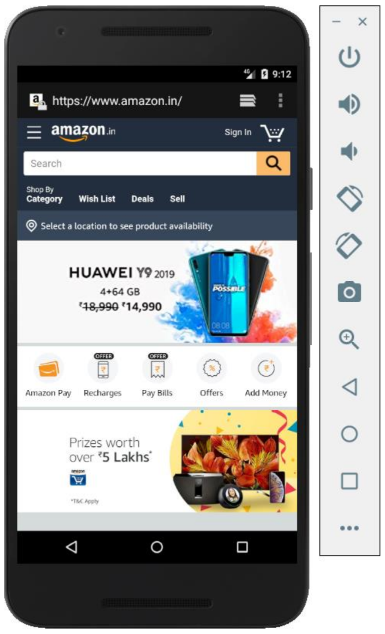
Illustrator: Arthur Racham

Publisher: Holder & Stoughton

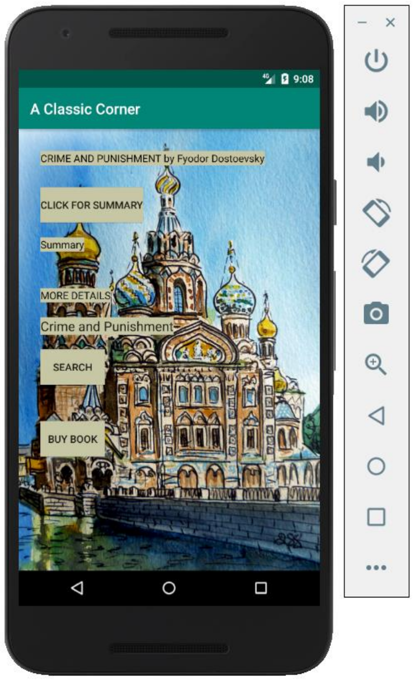
Date of Publication: 1906

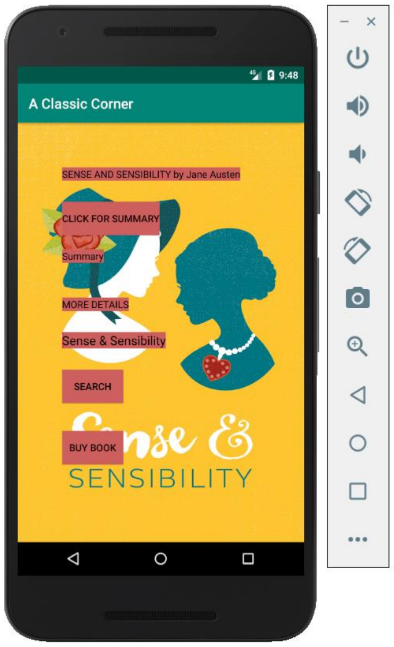
Best Edition: Wilco Classics

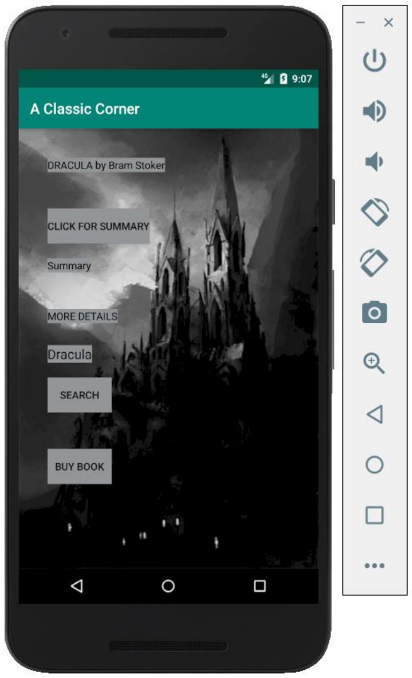
Rate: Rs. 200



Book Information Layout




9:48
A Classic Corner
SENSE AND SENSIBILITY by Jane Austen
CLICK FOR SUMMARY
Summary
MORE DETAILS
Sense & Sensibility
SEARCH
BUY BOOK
Sense &
SENSIBILITY


9:07
A Classic Corner
DRACULA by Bram Stoker
CLICK FOR SUMMARY
Summary
MORE DETAILS
Dracula
SEARCH
BUY BOOK