

CHAPTER 1: INTRODUCTION

1.1 Project Overview

Currently the cheque deposit process starts by manually depositing the cheque in any branch of the same bank, then it is sent for authorization and finally it gets submitted.

Our system will improve upon the present scenario. At the doorstep, the user will be facilitated with the services by digitally submitting the cheque through internet and it will be verified manually by the bank. For this purpose the cheque will contain payee details and there will be a unique QR code which will have the information of the registered payer. Authentication and validation will be done by QR Codes. The user will also be able to check the history of deposited cheques.

1.2 Relevance & Importance

The objective of this project is to develop a bank customer management system to the best satisfaction of the customer and for profit maximization to the Banks.

1. To create a banking system that is easily via internet
2. Reduce the flow of human traffic and long queues at banks
3. Reduce the time wasted in going to banks to update personal details.
4. To develop a bank customer management system with a multi-level security measure that will restore the customers' confidence.

The scope of this project is limited to some activities of the operations unit of a banking system which include registering account, transferring and receiving of cheques and updating personal details. This application does not focus on other online banking services such as; bill payment loan application etc.

1.3 Organization of Report

Chapter 1 covers the project overview, relevance and importance of project, and the organization of the report as stated (**1.3 above**).

Chapter 2 deals with the cross-platform application and requirements

Chapter 3 is about the design phase, database and the snapshots of the application.

Chapter 4 provides the system analysis, results and the discussions that is the Testing phase

Chapter 5 will be conclusion; which includes the concluding remarks, contribution, and limitation of the system.

Chapter 6 Suggestion of the future work, and summary.

CHAPTER 2: BACKGROUND MATERIAL

2.1 Cross Platform Application

2.1.1 Android Development

Android is a software stack for mobile devices such as Smart phone and Tablet that include an operating systems (Linux kernel based), middleware and key applications (Mobile app s/w). Goal of Android Project (s/w stack) is to create a successful real world product that improves mobile experience for mobile end user. It can be thought of a mobile OS but it is not limited to mobile only it is currently being used in various devices such as mobiles, tablet, television, wears etc.

The word was coined from Greek word “Avopoid” a combination of



Figure 2.1 Android Logo

Avop = Human Oid = having the form of.

Figure 2.1 shows the logo of Android.

An Android OS is a Robot design to look and behave like man that's why it has a logo of Robot. 'Android' word patent and copyright by Google in 2007.

Google Acquires Android Inc.

In July 2005, Google acquired Android Inc., a small startup company based in Palo Alto, CA.

Android's co-founders who went to work at Google included Andy Rubin (co-founder of Danger), Rich Miner (co-founder of Wildfire Communications, Inc.), Nick Sears (once VP at TMobile), and Chris White (one of the first engineers at WebTV). At the time, little was known about the functions of Android Inc. other than they made software for mobile phones.

Open Handset Alliance Founded

On 5 November 2007, the Open Handset Alliance, a consortium of several companies which include Google, HTC, Intel, Motorola, Qualcomm, T-Mobile, Sprint Nextel and NVIDIA, was unveiled with the goal to develop open standards for mobile devices. Along with the formation of the Open Handset Alliance, the OHA also unveiled their first product, Android, an open source mobile device platform based on the Linux operating system.

2.1.1.1 Features of Android

Table 2.1 shows us the features of android.

Table- 2.1 Features of Android

Feature	Description
Beautiful UI	Android OS basic screen provides a beautiful and intuitive user interface.
Connectivity	GSM/EDGE, IDEN, CDMA, EV-DO, UMTS, Bluetooth, Wi-Fi, LTE, NFC and WiMAX.
Storage	SQLite, a lightweight relational database, is used for data storage purposes.
Media support	H.263, H.264, MPEG-4 SP, AMR, AMR-WB, AAC, HE-AAC, AAC 5.1, MP3, MIDI, Ogg Vorbis, WAV, JPEG, PNG, GIF, and BMP
Messaging	SMS and MMS

Web browser	Based on the open-source WebKit layout engine, coupled with Chrome's V8 JavaScript engine supporting HTML5 and CSS3.
Multi-touch	Android has native support for multi-touch which was initially made available in handsets such as the HTC Hero.
Multi-tasking	User can jump from one task to another and same time various applications can run simultaneously.
Resizable widgets	Widgets are resizable, so users can expand them to show more content or shrink them to save space
Multi-Language	Supports single direction and bi-directional text.
Wi-Fi Direct	A technology that lets apps discover and pair directly, over a high-bandwidth peer-to-peer connection.

2.1.1.2 Android framework

Android is one of an Open source platforms. It is created by Google and owned by Open Handset Alliance. It is designed with goal “accelerate innovation in mobile” As such android has taken over a field of mobile innovation. It is free and open platform that differs hardware from software that runs on it. It results for much more devices be running the same application. Android it is complete software package for a mobile device. Since the beginning android team offers the developing kit (tool and frameworks) for creating mobile applications quick and easy as possible. In some cases you do not specially need an android phone but you are very welcome to have one. It can work right out of the box, but of course users can customize it for their particular

needs. For manufactures it is ready and free solution for their devices. Except specific drivers android community provides everything else to create their devices.

The Android framework includes the following key services –

- **Activity Manager** – Controls all aspects of the application lifecycle and activity stack.
- **Content Providers** – Allows applications to publish and share data with other applications.
- **Resource Manager** – Provides access to non-code embedded resources such as strings, color settings and user interface layouts.
- **Notifications Manager** – Allows applications to display alerts and notifications to the user.
- **View System** – An extensible set of views used to create application user interfaces.

2.1.1.3 Android Architecture

The following diagram (Figure 2.2) shows the major components of Android

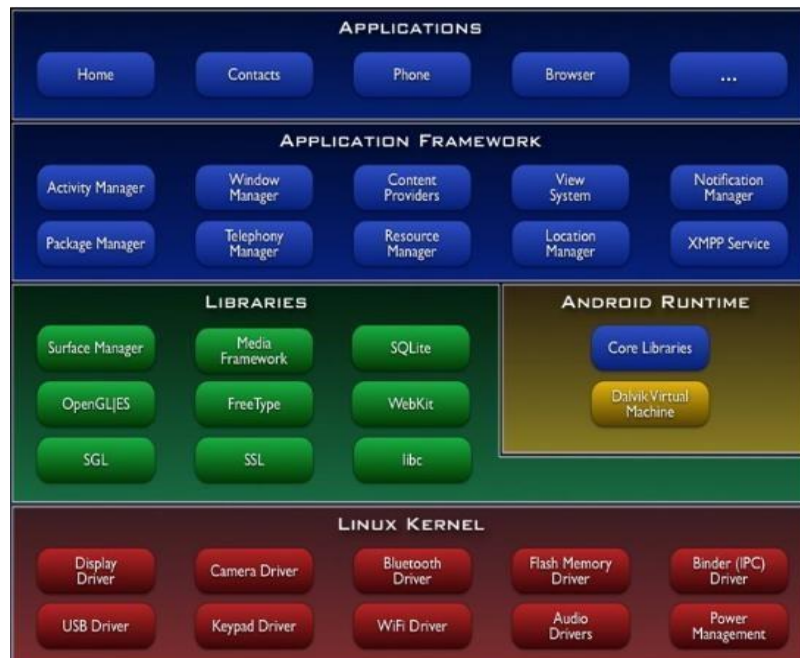


Figure 2.2 Architecture of Android OS.

2.1.1.4 Different Versions of Android (Figure 2.3)



Figure 2.3 Versions of Android

2.1.1.5 Software/Tools

Android Studio- The Official IDE for Android

Android Studio provides the fastest tools for building apps on every type of Android device. World-class code editing, debugging, performance tooling, a flexible build system, and an instant build/deploy system all allow to focus on building unique and high quality apps.

Android Environment Setup: - for development of android app. We require an environment that is called Android Environment Setup. We will be glad to know that we can start our app development either of the following OS---

- 1- M.S. windows XP or later version.
- 2- MAC OS or later version with Intel chip.
- 3- Linux

All required tools to develop Android apps are freely available and we can download from internet. Following is the list of all required tools to develop Android Apps—

- 1- JDK 5 or later version.
- 2- Android SDK (Software Development Kit).
- 3- IDE. OR ADT-Bundle Tool + JDK OR Android Studio.exe + JDK

The Android SDK stands for Software Development Kit set that is used to develop application. The Android SDK includes following things.—

- 1- API (required libraries) android.jar
- 2- Relevant Documentations of APIs (tutorials)
- 3- Build Tools (like aapt, aidl etc.)
- 4- Extra libraries (Support and Design library like appcompat-v7, support-v4, design library etc.)
- 5- An Emulator (AVD)
- 6- Platform tools (like adb)
- 7- Etc.

Dalvik virtual machine (DVM) :- (replaced by AVM) - DVM is a software that actually executes the android application.

- As we know android application is written in java and then compiled into the bytecode files.
- The DX–Compiler converts java byte code files to a single DEX BYTECODE file (classes.dex).
- When user launches application then AVM executes DEX bytecode file.

Android Emulator-You can launch an app on the emulator when you run your project, or you can drag an APK file onto the emulator to install it. As with a hardware device, after you install an app on a virtual device, it remains until you uninstall or replace it. If needed, you can test how multiple apps, such as your own or system apps, work with each other.

2.1.2 Learning Outcome

2.1.2.1 Event

An Event is the change of state of any object either by user or by application itself. The event is broadly classified into two categories-

1. **Foreground Events-** Those events which require the direct interaction of User. They are generated as consequences of a person interacting with the graphical components in GUI. For example – clicking a Button, checked/unchecked a RadioButton, selecting an item of list etc.
2. **Background Events -** Those events which don't require the direct interaction of User. For example- OS interrupts, Hardware/Software failure etc.

General Steps for implementation of Event handling –

1. Identify Event type.
2. Create Listener class by implementing Listener interface.
3. Override callback methods.
4. Create listener Object of listener class.
5. Identify source of event.
6. Do registration of listener object with source of event.

Listener Class: - listener class provides implementation of listener interface. It means we will provide definition of callback methods in listener class.

Listener object: - it is an object of Listener class.

Source of event: - It is any android widget (Object) which is responsible for event generating. Ex. Button, ImageButton, TextView, EditText etc.

2.1.2.2 Popup Messages

There are many situations where you might want your app to show a quick message to the user, without necessarily waiting for the user to respond. For example, when a user performs an action like sending an email or deleting a file, your app should show a quick confirmation to the user.

Android provides two ways to show pop-up messages -

Toast: - It is a class defined in android.widget package. An object of this class provides facility of showing simple feedback message about an operation in a small popup. It only fills the amount of space required for the message and current activity remains visible & interactive.

Syntax: - `Toast.makeText(context, "message saved in draft", Toast.LENGTH_SHORT);`

Context: - It is a class defined in android.content package. An object of this class is representing a buffer that has the references of all data & resources about an application. It is basically an application environment to our android app, by which any widget or component can access required data & resources from it.

SnackBar – It is a class defined in android.support.design.widget package. An object of this class provides lightweight feedback about an operation. It shows a brief message at the bottom of the screen on mobile and lower left on larger devices. Snack bars appear above all other elements on screen and only one can be displayed at a time.

A **Snackbar** shows a message at the bottom of the activity, but the rest of the activity is still usable. **Snackbars** can contain an action which is set via `setAction(CharSequence, android.view.View.OnClickListener)`.

2.1.2.3 Widgets

2.1.2.3.1 Floating Action Button

It is a class defined in `android.support.design.widget` package (Design support library), an object of this class is represented by a circled icon floating above the UI.

2.1.2.3.2 Toggle Button

This is a class defined in `android.widget`. An object of this class represents functionality of an individual toggle button that allows user to change setting between two states. We can define a **Toggle button** in xml Layout file with `<Toggle Button>` Tag. Figure 2.4 shows the **Toggle Button**.



Figure 2.4 Toggle Button

2.1.2.4 RadioButton

This is a class defined in `android.widget`. An object of this class provides functionality of an individual **Radio Button**. **Radio Button** allow user to select one option from multiple. Figure 2.5 shows **Radio Button**.

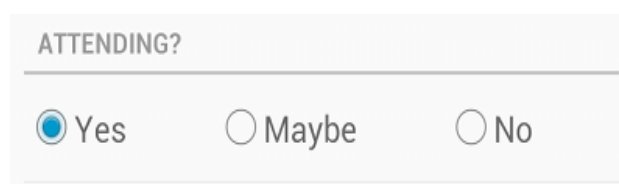


Figure 2.5 Radio Button

For each **Radio Button** option, we have to define `<Radio Button>` tag in your Layout file. We should use radio buttons for optional sets that are mutually exclusive then we must group them together inside a `<Radio Group>` to make **Radio Buttons** mutually exclusive. By grouping them together the system ensure that only one **Radio Button** can be selected at that time.

Responding to click event: Whenever the user select any **Radio Button**, the corresponding **Radio Button** receives `onClick` event.

2.1.2.5 CheckBox

This is a class defined in android.widget package. An object of this class represents functionality of Check Box which allows the user to select one or more option for selection set.

Responding to onCheckedChangeListener event: When the user checked a CheckBox, object receive onCheckedChangeListener event.

Listener- OnCheckedChangeListener .

Event Handler – OnCheckedChangeListener(CompoundButton cb, boolean isChecked).

2.1.2.6 AlertDialog

A Dialog is a small window that prompts the user to make a decision or enter additional information. A dialog does not fill the screen and is normally used for events that require users to take an action before they can proceed.

Types of Dialog:

1. AlertDialog: A dialog that can show a title, an icon and up to three buttons, a list of selectable items.
2. DatePickerDialog: A dialog with a pre-defined UI that allows the user to select a date.
3. TimePickerDialog: A dialog with a pre-defined UI that allows the user to select a time.

AlertDialog- It is a subclass of Dialog (defined in android.app package) that can display a dialog with one, two or three buttons along with Title and Icon. It is used, if you want to ask the user about taking a decision between yes and no or remind me later in response of any particular action taken by the user, by remaining in the same activity and without changing the screen. In order to make an alert dialog, we need to make an object of AlertDialog.Builder which an inner class of AlertDialog. Table 2.2 List the Functions provided by the Builder class to customize the alert dialog.

Table 2.2 Methods

Sr.No	Method type & description
1	setIcon(Drawable icon) This method set the icon of the alert dialog box.
2	setCancelable(boolean cancelable)

	This method sets the property that the dialog can be cancelled or not, when pressing back button or else.
3	setMessage(CharSequence message) This method sets the message to be displayed in the alert dialog
4	setOnCancelListener(DialogInterface.OnCancelListener onCancelListener) This method Sets the callback that will be called if the dialog is canceled.
5	setTitle(CharSequence title) This method set the title to be appear in the dialog

2.1.2.7 Pickers

Android provides controls for the user to pick a time or pick a date as ready-to-use dialogs. Each picker provides controls for selecting each part of the time (hour, minute, AM/PM) or date (month, day, year).

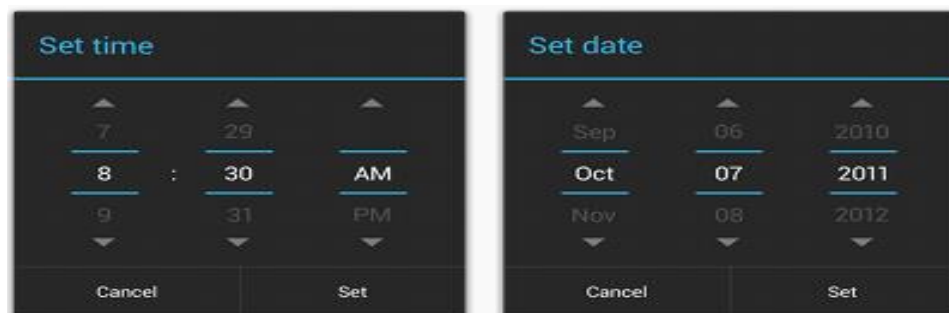


Figure 2.6 Date and Time Pickers

2.1.2.7.1 Date Picker

This class is defined in android.widget package. An object of this class provides facility to render Date picker widget in current Activity with/without the help of DatePickerDialog, which allows the user to select the day, month and year. In order to create DatePicker widget, we have to define <DatePicker> Tag in layout XML file.

Figure 2.6 shows Date and time pickers.

Responding to event: When the date has been adjusted by the user. DatePicker receives OnDateChanged event.

Listener Name: OnDateChangeListener

Event Handler: public void onChanged(DatePicker dp, int selectedYear, int selectedMonth, int selectedDay) ;

Responding to event: When the date has been adjusted by the user and pressed Set Button. DatePickerDialog receives OnDateSet event.

Listener Name: OnDateSetListener

Event Handler: public void onDateSet(DatePicker dp, int selectedYear, int selectedMonth, int selectedDay) ;

2.1.2.7.2 Time Picker

This class is defined in android.widget package. An object of this class provides facility to render Time picker widget in current Activity with/without the help of TimePickerDialog, which allows the user to select the time of day, in either 24 hour or AM/PM mode. In order to create TimePicker widget, we have to define <TimePicker> Tag in layout XML file.

Responding to event: When the time has been adjusted by the user. Time Picker receives OnTimeChanged event.

Listener Name: OnTimeChangedListener

Event Handler: public void onTimeChanged(TimePicker tp, int selectedHour, int selectedMinute);

Responding to event: When the time has been adjusted by the user and pressed Set/cancel Button. TimePickerDialog receives OnTimeSet event.

Listener Name: OnTimeSetListener

Event Handler: public void onTimeSet(TimePicker tp, int selectedHour, int selectedMinute);

2.1.2.8 Adapter

Adapter is an interface defined in android.widget package and implementation of this interface acts as Bridge between an adapter view and data source for the creation of view.

1. The adapter provides access to the data items.
2. It is also responsible for making a child view object for each item in data set.

Adapter View

This is an abstract class which has been defined in android.widget package and object it's any implementing class represents a Parent view whose children are determined by adapter.

Like- ListView, GridView, Spinner, Gallery are commonly used subclass of AdapterView. Figure 2.7 shows ListView.

** Here Adapter View is responsible for taking Child View object from adapter and controlling how to child views should display to the user.

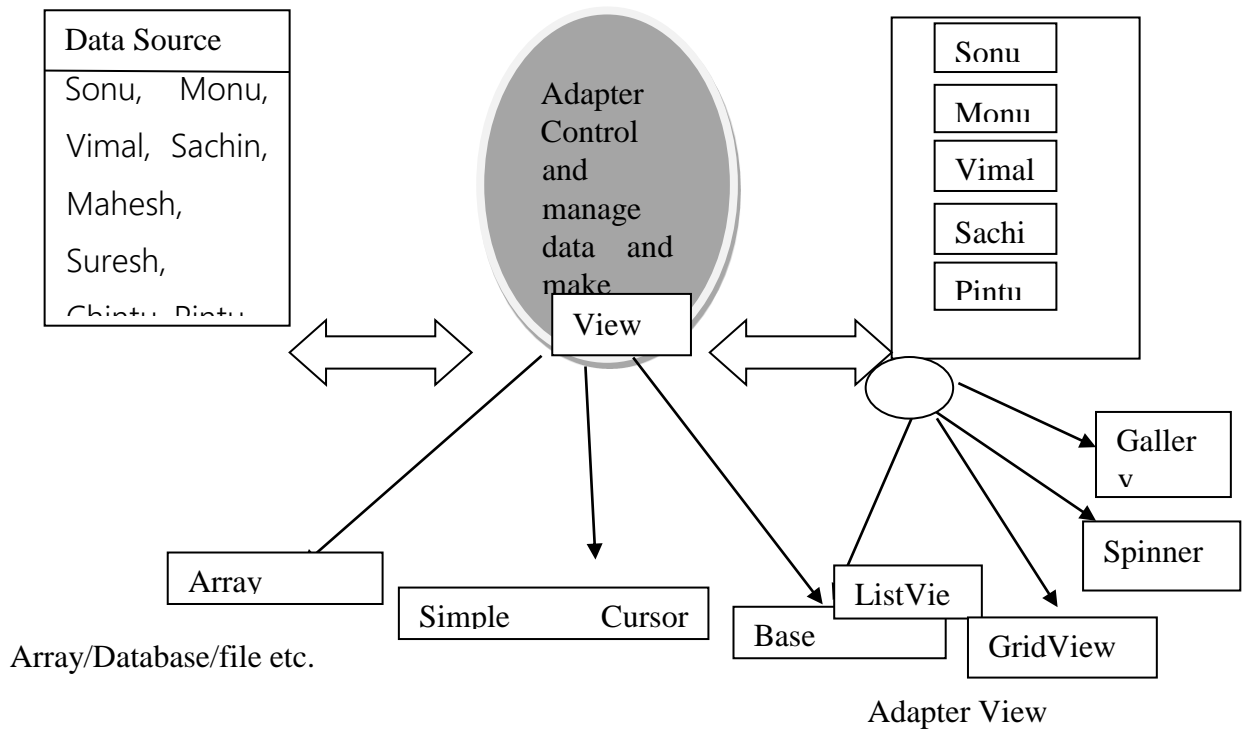


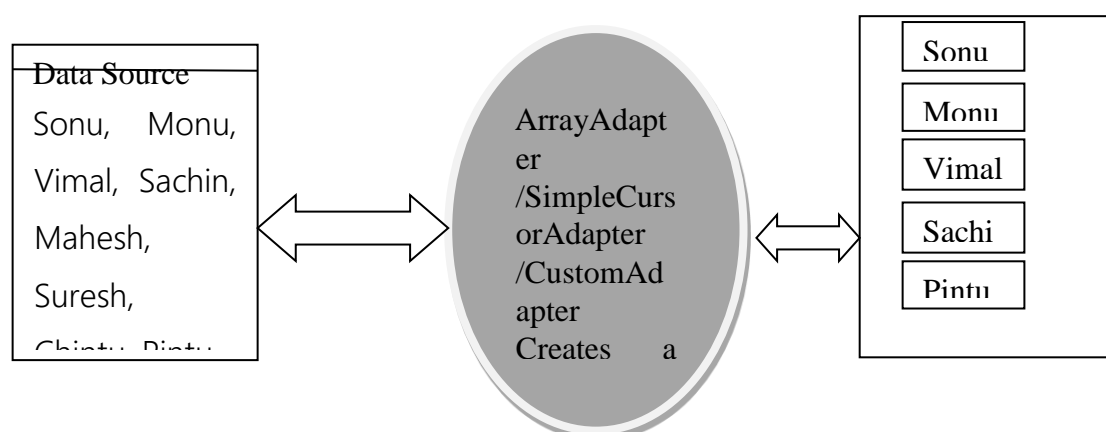
Figure 2.7 List View

2.1.2.8.1 ListView

It displays vertically scrolling list of child View objects with which user can interact. It is class defined in android.widget package. An object of this class created by OS from XML file (inflation) and object of this class provides facility to display multiple child views in vertical scrolling list with which user can interact.

Working of List View in Figure 2.8

Here, A simple architecture to how List View work



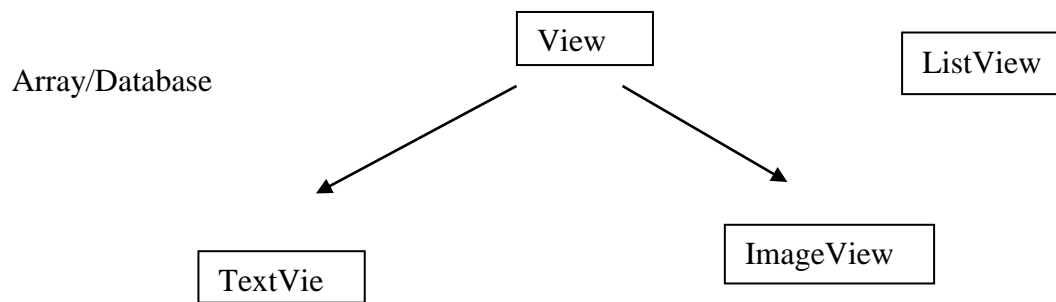


Figure 2.8 Custom View

2.1.2.8.2 GridView

It is class defined in android.widget package. An object of this class created by OS from XML file (inflation) and object of this class provides facility to display multiple child views vertical & horizontal scrolling list (in row-column) with which user can interact.

Why GridView?

- To display lots of information to browse easily.
- As user reaches the end of screen, more results are generated or displayed.
- When the users click on row some action can be performed.

2.1.2.9 Activity

It is class defines in an android.app package an object of this class provides a screen contains different-different UI with which user can interact.

An application can have many screens i. e. many activities, and one of them must be marked as launcher activity in AndroidManifest.xml file, that is presented on screen when the apps lunches.

2.1.2.10 Intent

In general, intent is the standard message in android that expresses the intention of one component to perform the work by any other component. Intent allows the component to interact with other components defining by you or already installed in android OS.

Types of Intent:-

There are basically two types of intents.

1. Explicit intent.
2. Implicit intent.

Explicit intent:-

An intent object will be treated as explicit intent when sender component is aware about which component to call to perform some action.

Implicit Intent:-

An intent object will be treated as implicit when sender component does not aware of which component is to be exactly performed your desired action.

In case of implicit intent, Android OS (package manager) chooses an appropriate component on the basis of action, data, and category fields of the intent object, to perform desired action.

For example: - If you want to display an URL then we have to create an intent object and set action, and data field with ACTION_VIEW and TYPE: <http://www.facebook.com>.

Building an Intent

1. **ComponentName:** - This field of an intent object defines what component receives this intent object.
2. **Action:-**This field of an intent object is a string which defines what operation one component is wanted to perform by another component. This field is mandatory part (When doesn't have ComponentName field) of intent object. The Intent class has number of predefined action (String constant) contains for different-different operations.
3. **Data Field:-**This field of an intent object defines data and type of data (mime type) in the form of URI (Uri object), an action to be performed on it (define by action field) by other components (to whom intent object has sent).
4. **Uri:-**By Using static method parse(String uriString) of Uri class define in android.net.
5. **Extra:-**This field of an Intent object is represented by Bundle Object which provides a facility to add some additional information in it required to accomplish the requested action in the form of key value pair. Just same as few action uses particular kind of data (Uri), few action also uses particular extras.

2.1.2.11 Activity Life Cycle

Android OS calls certain methods on the object of activity class to notify whether your app is currently running or not, or is being stopped, or is being Paused etc.

During the life of an activity, the system calls a core set of lifecycle methods in a sequence similar to a step pyramid. That is, each stage of the activity lifecycle is a

separate step on the pyramid. As the system creates a new activity instance, each callback method moves the activity state one step toward the top. The top of the pyramid is the point at which the activity is running in the foreground and the user can interact with it.

As the user begins to leave the activity, the system calls other methods that move the activity state back down the pyramid in order to dismantle the activity. In some cases, the activity will move only part way down the pyramid and wait (such as when the user switches to another app), from which point the activity can move back to the top (if the user returns to the activity) and resume where the user left off.

Resumed

In this state, the activity is in the foreground and the user can interact with it.

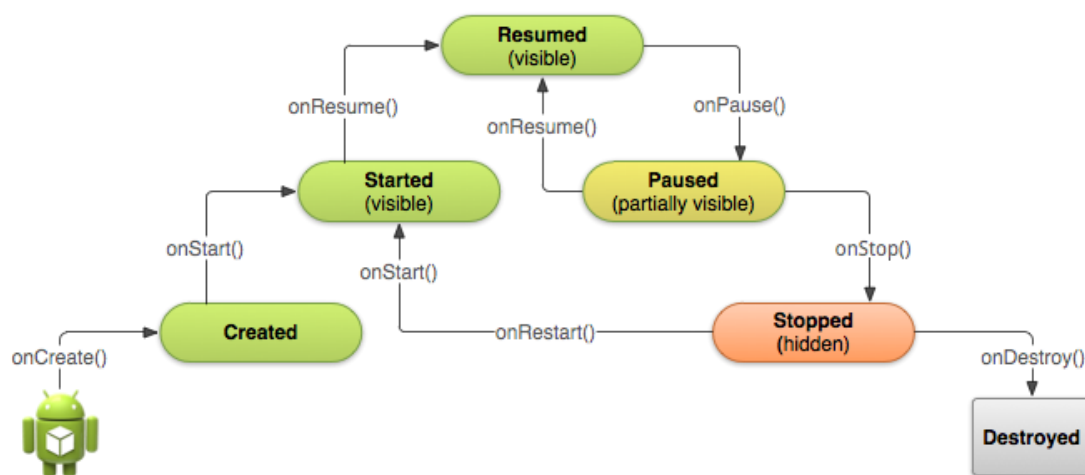


Figure 2.9(1) Life Cycle of Activity

Paused

In this state, the activity is partially obscured by another activity—the other activity that's in the foreground is semi-transparent or doesn't cover the entire screen. The paused activity does not receive user input and cannot execute any code.

Stopped

In this state, the activity is completely hidden and not visible to the user; it is considered to be in the background. While stopped, the activity instance and all its state information such as member variables is retained, but it cannot execute any code.

Figure 1.9(1) shows life cycle of Activity.

Callback methods

1. **onCreate():**- This is first callback which is called by android OS when the activity has been first created

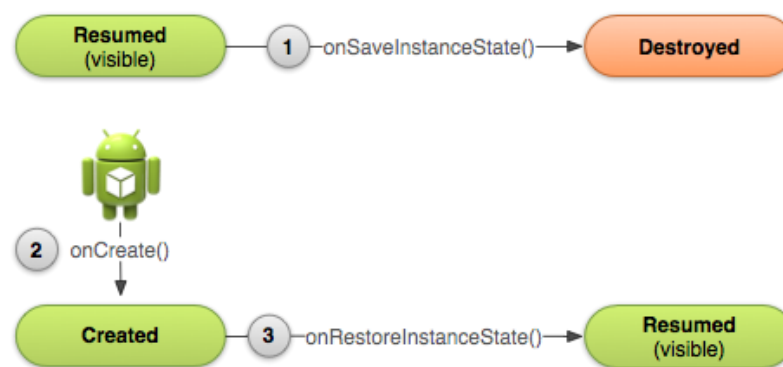


Figure 2.9(2) Part of life cycle.

2. **onStart():**- This callback is called by android OS when the activity is being started.
3. **onResume():**- This callback is called by android OS when the user starts interacting with screen.
4. **onPause():**- This callback is called by android OS when it is being partially visible, i. e. Activity goes in background.
5. **onStop():**- This callback is called by android OS when the Activity is no longer visible.
6. **onRestart():**- This callback is called by android OS when the Activity is restarting after being stopped.
7. **onDestroy():**- This callback is called by android OS when the Activity is being terminated either by OS or by user.

Figure 1.9(2) shows part of activity life cycle when on saved instance and on restore instance are called.

2.1.2.12 Async Task

AsyncTask is an abstract class provided by Android which helps us to use the UI (main) thread properly. This class allows us to perform long/background operations and show its result on the UI thread without having to manipulate threads.

AsyncTask has four Steps:

When an asynchronous task is executed, the task goes through 4 steps:

1. **doInBackground:** Code performing long running operation goes in this method. When onClick method is executed on click of button, it calls execute method which accepts parameters and automatically calls doInBackground method with the parameters passed.
2. **onPostExecute:** This method is called after doInBackground method completes processing. Result from doInBackground is passed to this method.
Note- This method won't be invoked if the task was cancelled.
onPreExecute: This method is called before doInBackground method is called.
3. **onProgressUpdate:** This method is invoked by calling publish Progress anytime from doInBackground call this method.

2.1.3 Java Language

Java is a general purpose, high-level programming language developed by Sun Microsystems. A small team of engineers, known as the **Green Team**, initiated the language in 1991. Java was originally called **OAK**, and was designed for handheld devices and set-top boxes.

Today Java is a commonly used foundation for developing and delivering content on the Web. According to Oracle, there are more than 9 million Java developers worldwide and more than 3 billion mobile phones run Java.

2.1.4 Java Platform

One design goal of Java is portability, which means that programs written for the Java platform must run similarly on any combination of hardware and operating system with adequate runtime support. This is achieved by compiling the Java language code to an intermediate representation called Java bytecode, instead of directly to architecturespecific machine code. Java bytecode instructions are analogous to machine code, but they are intended to be executed by a virtual machine (VM) written

specifically for the host hardware. End users commonly use a Java Runtime Environment (JRE) installed on their own machine for standalone Java applications, or in a web browser for Java applets

2.1.5 Principles

There were five primary goals in the creation of the Java language:

1. It must be "simple, object-oriented and familiar".
2. It must be "robust and secure".
3. It must be "architecture-neutral and portable".
4. It must execute with "high performance".
5. It must be "interpreted, threaded and dynamic".

2.1.6 Servlet

Java Servlet technology provides Web developers with a simple, consistent mechanism for extending the functionality of a Web server and for accessing existing business systems. Servlets are server-side Java EE components that generate responses (typically HTML pages) to requests (typically HTTP requests) from clients. A servlet can almost be thought of as an applet that runs on the server side without a face. Figure 2.10 shows the logo of Java Servlet.



Figure 2.10 Java Servlet Logo

The import statements direct the Java compiler to include all the public classes and interfaces from the packages in the compilation. Packages make Java well suited for large scale applications.

2.2 Requirements

2.2.1 Software Requirements

- **Eclipse IDE:** Eclipse is an integrated development environment used in computer programming, and is the most widely used Java IDE. It contains a base workspace and an extensible plug-in system for customizing the environment.
- **Android Studio:** Android Studio is the official integrated development environment for Google's Android operating system, built on JetBrains' IntelliJ IDEA software and designed specifically for Android development. New features are expected to be rolled out with each release of Android Studio. The following features are provided in the current stable version:
 - Gradle-based build support
 - Android-specific refactoring and quick fixes
 - Lint tools to catch performance, usability, version compatibility and other problems
 - ProGuard integration and app-signing capabilities
 - Template-based wizards to create common Android designs and components
 - A rich layout editor that allows users to drag-and-drop UI components, option to preview layouts on multiple screen configurations.
- **MySQL Browser:- MySQL Query Browser** is a free official GUI for the popular MySQL database server. It fills the gap in MySQL Administrator by allowing you to perform queries directly onto any schema that you choose. You can either write the queries by hand or use the limited query generation that is part of MySQL Query Browser.

2.2.2 Software Development Life Cycle:

Agile SDLC model is a combination of iterative and incremental process models with focus on process adaptability and customer satisfaction by rapid delivery of working software product. Agile Methods (Figure 2.11) break the product into small incremental builds. These builds are provided in iterations.

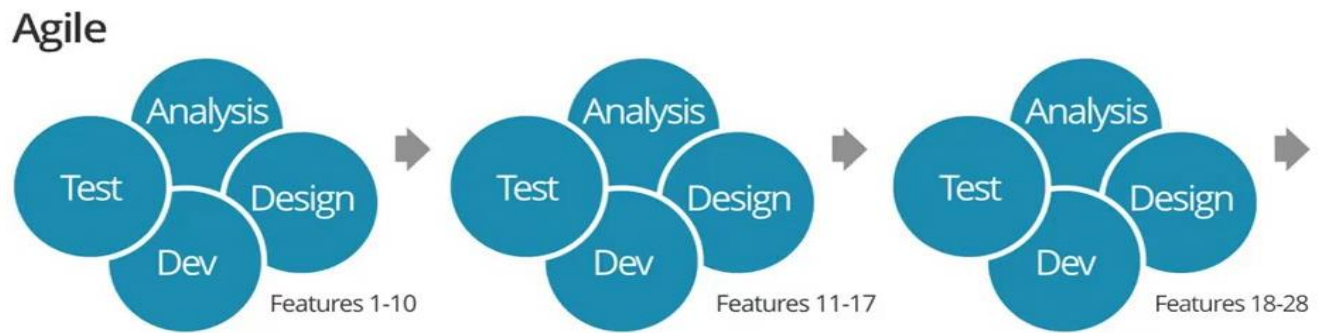


Figure 2.11 Agile Model

2.2.3 Functional Requirements

➤ **User-**

- Register account
- Login account
- Pay a cheque
- Receive a cheque
- View account details
- Manage profile
- View History
- Logout

2.2.3 Non- Functional Requirements

- Android Studio
- Eclipse IDE
- MySQL

CHAPTER 3: CONTRIBUTIONAL WORK

3.1 Description

Available approaches for depositing a cheque :

- Submitting in bank.
- Using website.
- Using ATM machines.
- Using android applications.

The current system of depositing a cheque follows the process of scanning the cheque and uploading its images. Therefore, verifying and depositing the cheque. While, on the other hand our system does not follow this conventional method.

Advantages:

- Easy access
- Cost Effective
- Vast coverage
- Reduction of paper usage
- 24x7 request generation
- Speed depository

Services

- Cheque Deposit
 - Pay
 - Receive
- Cheque Transaction History
- Limiting Cheque Bounce
- Cheque Cancellation Facility
- Security
 - Account linked through registered phone number.

3.2 Use Case Diagram

Use case diagrams are usually referred to as behavior diagrams used to describe a set of actions (use cases) that some system or systems (subject) should or can perform in collaboration with one or more external users of the system (actors). The Figure 3.1 shows the use case diagram.

In brief, the purposes of use case diagrams can be said to be as follows –

- Used to gather the requirements of a system.
- Used to get an outside view of a system.
- Identify the external and internal factors influencing the system.
- Show the interaction among the requirements and actors.

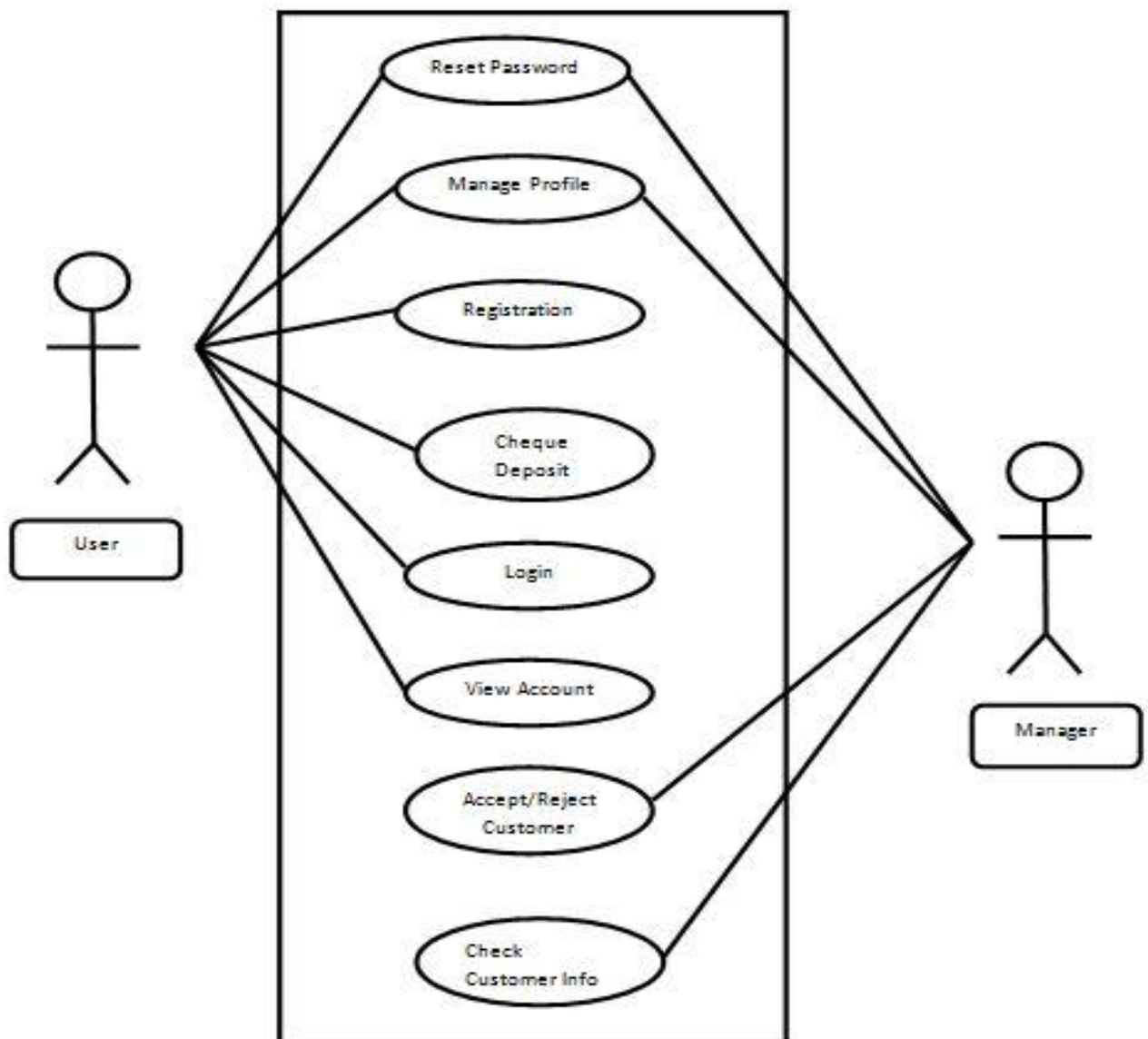


Figure 3.1 Use Case Diagram

3.3 Sequence Diagram

A sequence diagram is an interaction diagram that shows how objects operate with one another and in what order. It is a construct of a message sequence chart.

A sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. Sequence diagrams are typically associated with use case realizations in the Logical View of the system under development. Sequence diagrams are sometimes called event diagrams or event scenarios. Figure 3.2, 3.2, 3.4, 3.5, 3.6 and 3.7 represent the sequence diagram related to the system.

3.3.1 Registration

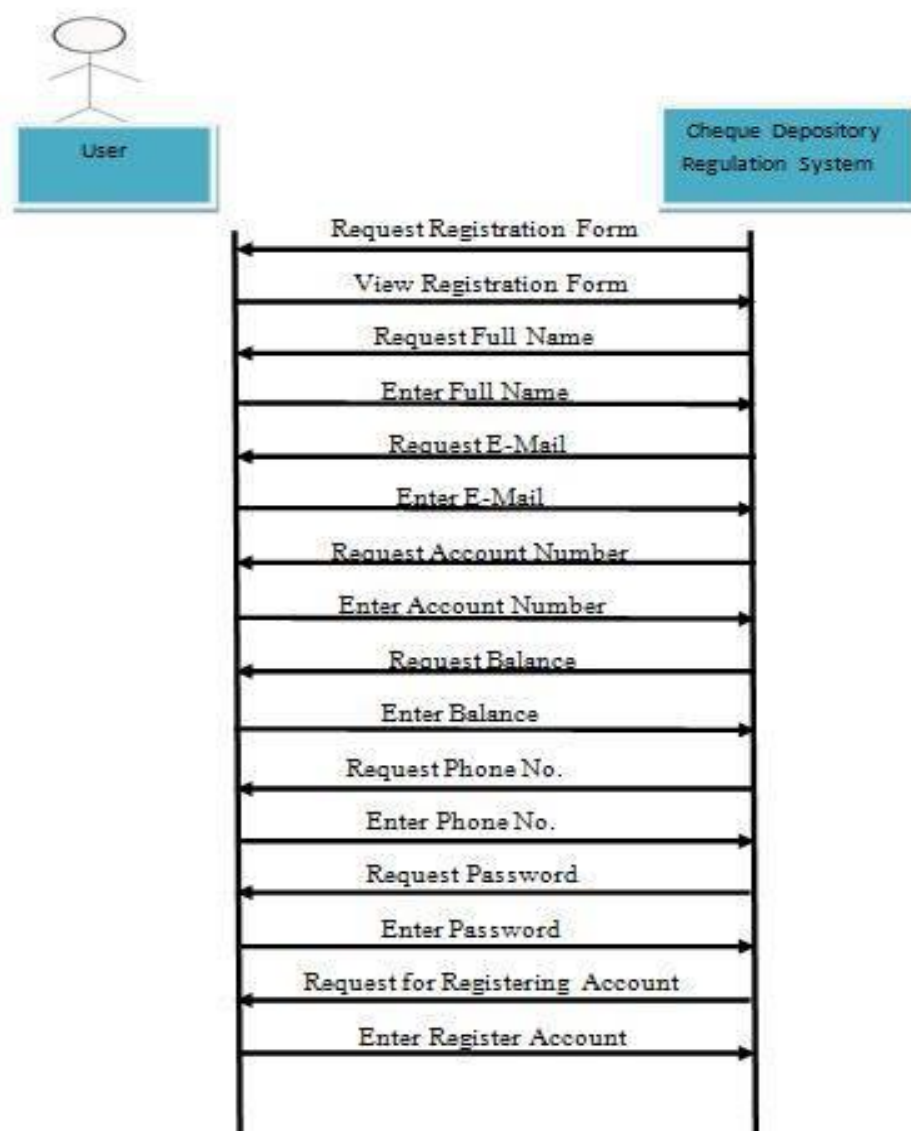


Figure 3.2 Sequence Diagram for Registration

3.3.2 Login

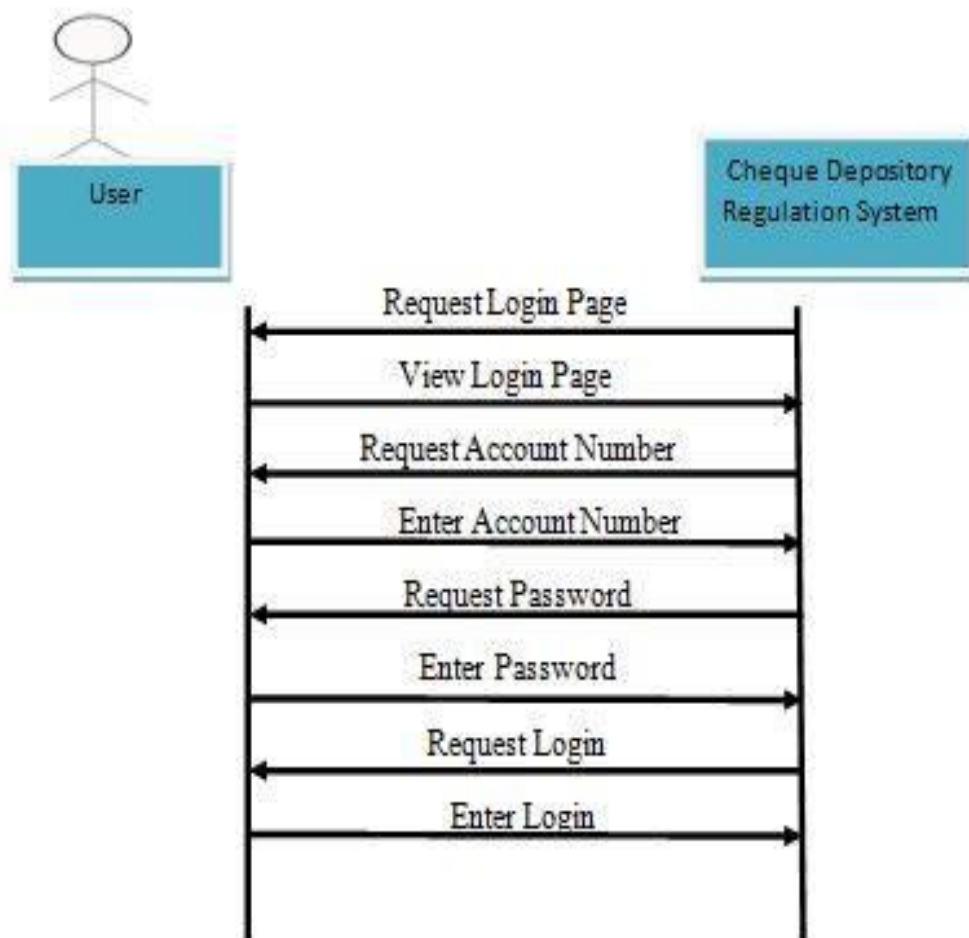


Figure 3.3 Sequence Diagram for Login

3.3.3 Pay

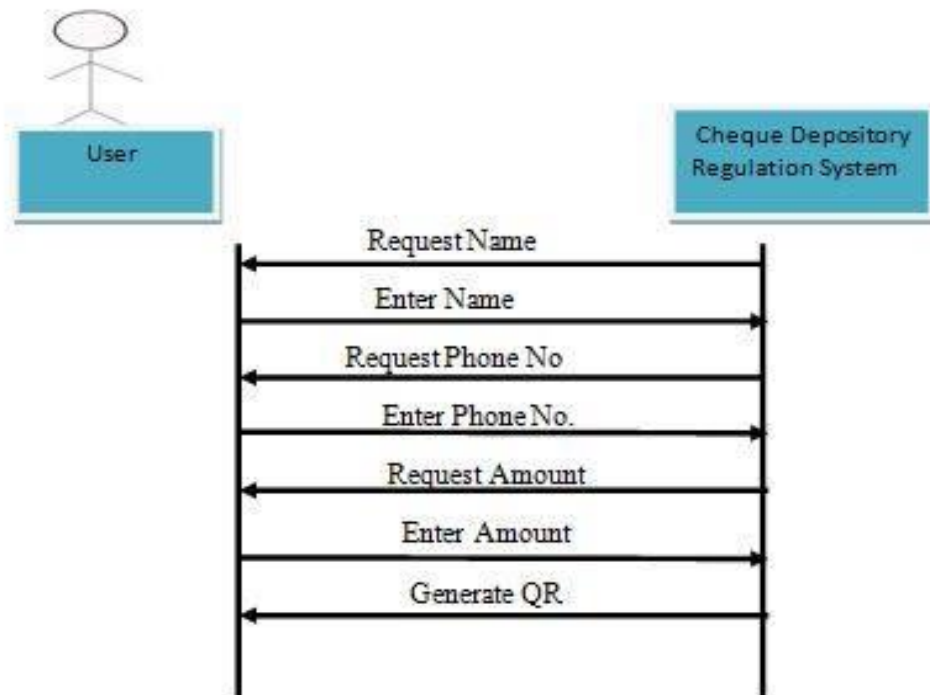


Figure 3.4 Sequence Diagram for Paying Cheque

3.3.4 Receive

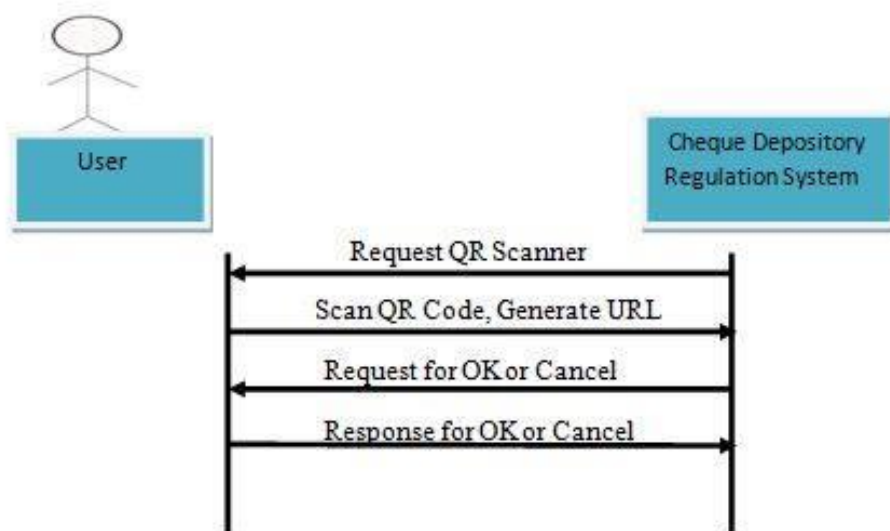


Figure 3.5 Sequence Diagram for Receiving Cheque

3.3.5 View History

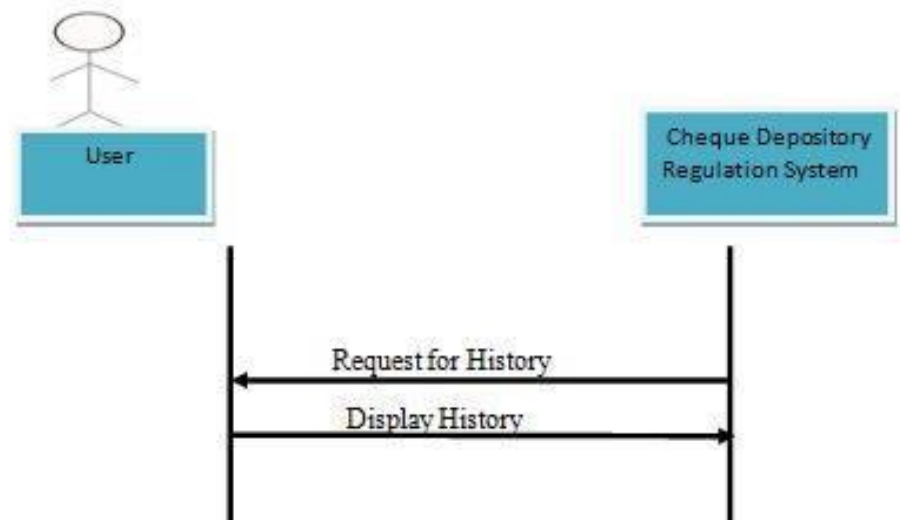


Figure 3.6 Sequence Diagram for Viewing History

3.3.6 Logout

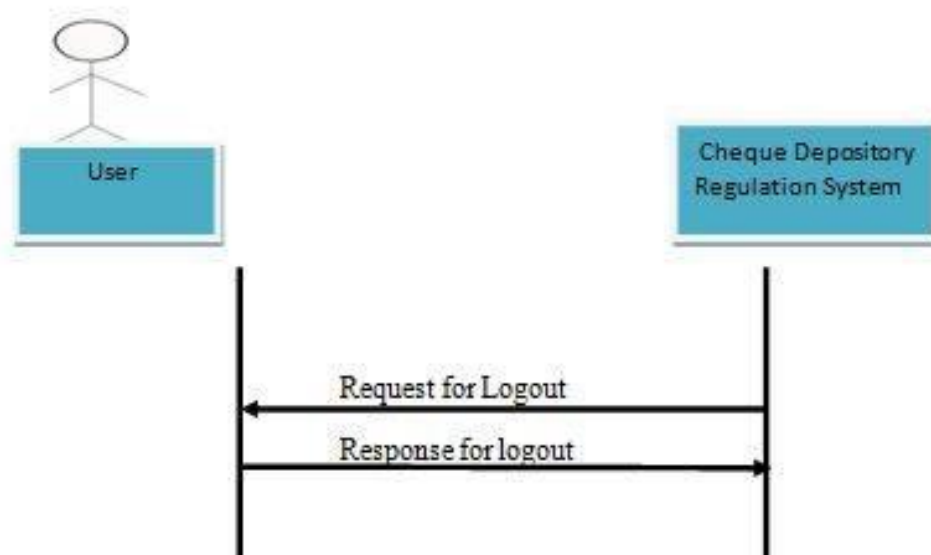


Figure 3.7 Sequence Diagram for Logout

3.4 Activity Diagram

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams are intended to model both computational and organizational processes (i.e. workflows). Activity diagrams show the overall flow of control. Figure 3.8, 3.9 and 3.10 show the activity diagram for the system.

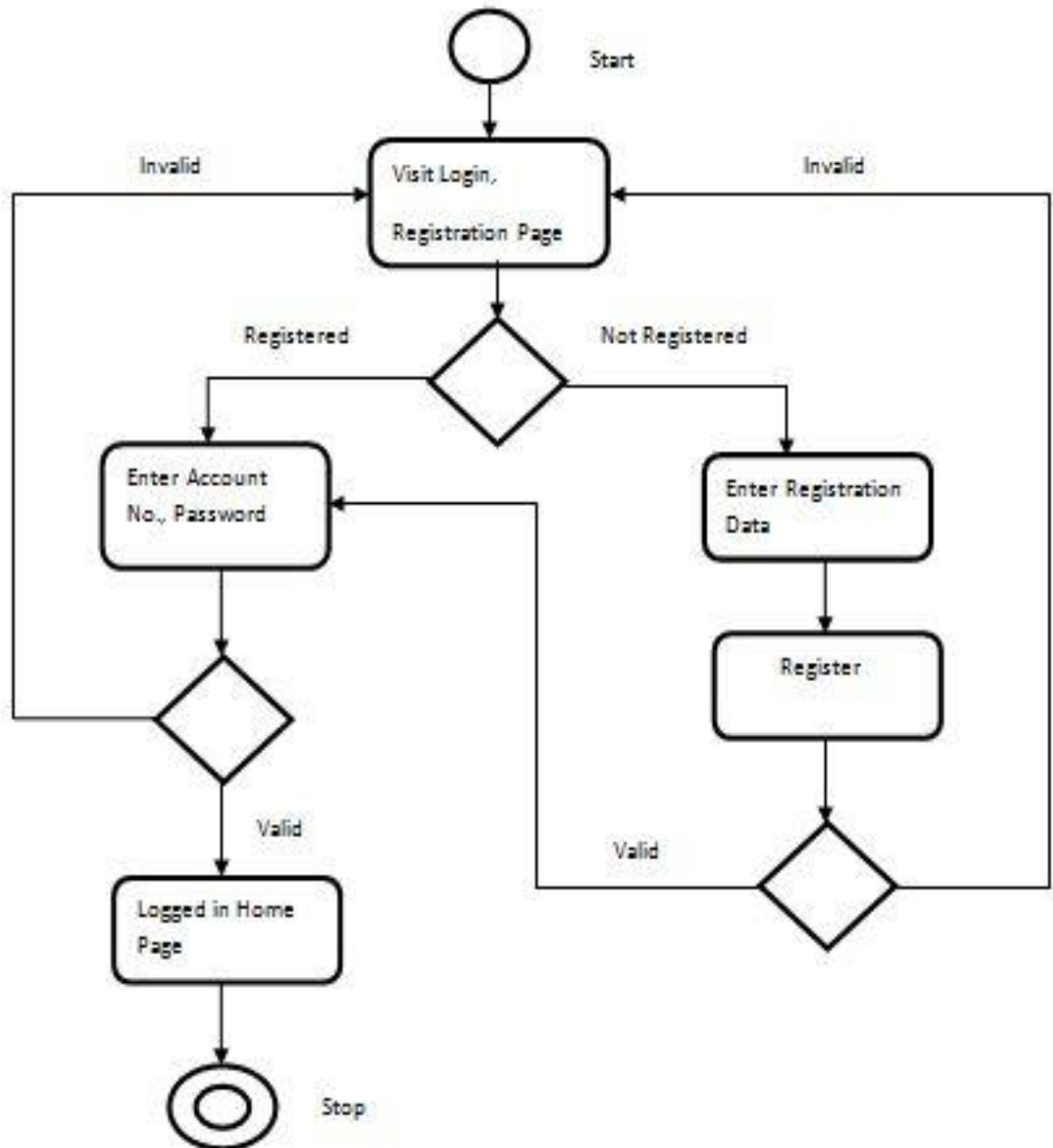


Figure 3.8 Activity Diagram for Registration and Login.

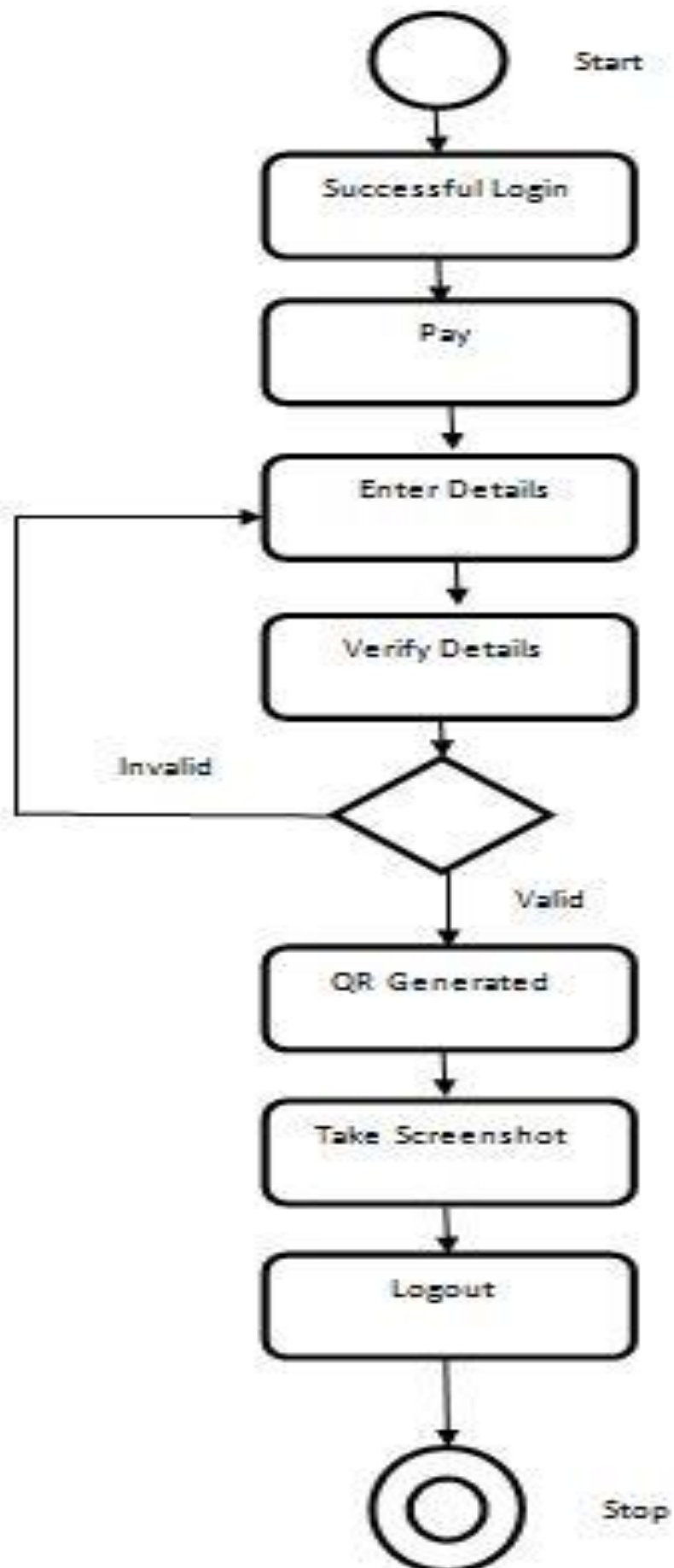


Figure 3.9 Activity Diagram for Paying a Cheque

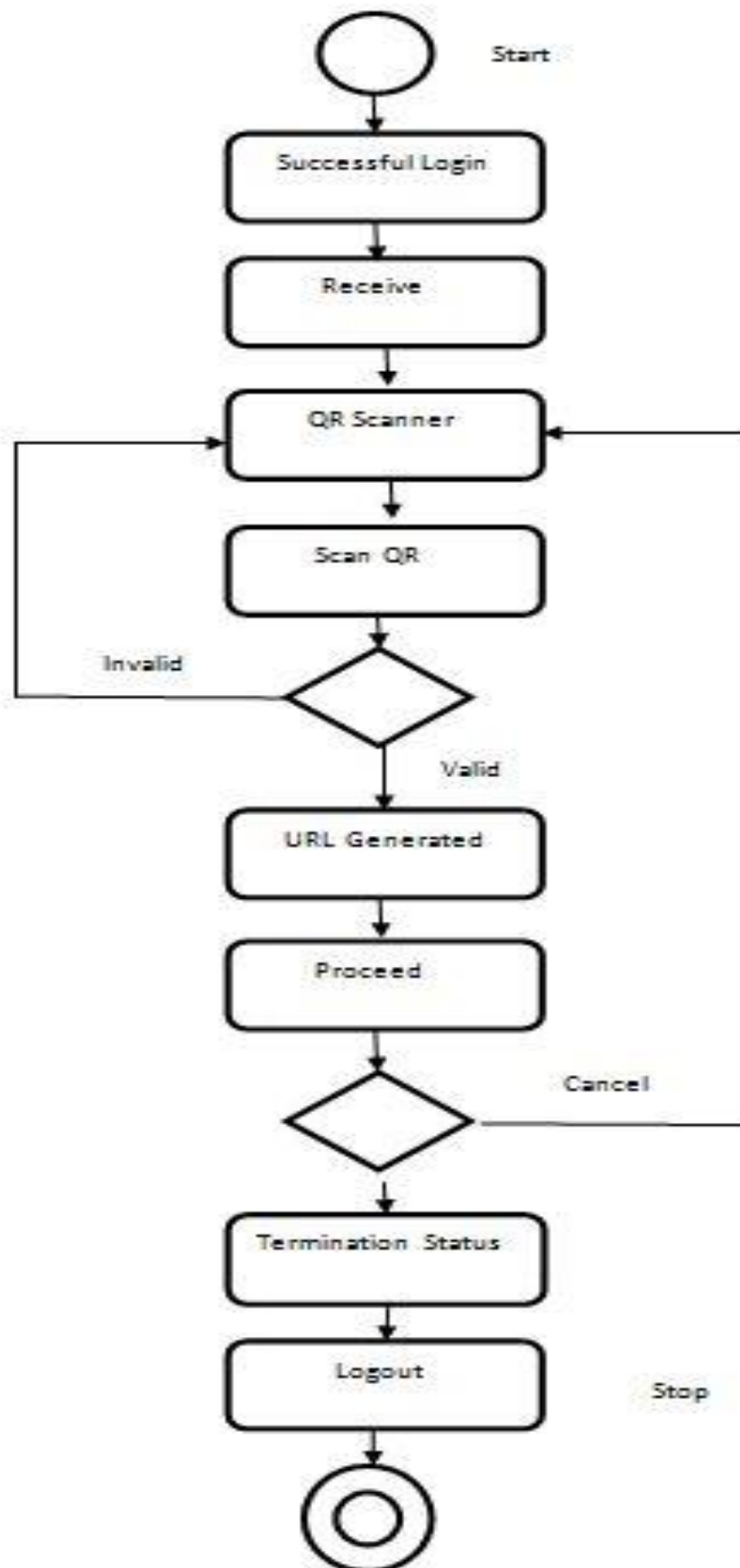


Figure 3.10 Activity Diagram for Receive a Cheque.

3.5 Data Flow Diagram

A data flow diagram (DFD) is a graphical representation of the "flow" of data through an information system, modeling its *process* aspects. A DFD is often used as a preliminary step to create an overview of the system without going into great detail, which can later be elaborated. DFDs can also be used for the visualization of data processing (structured design).

A DFD shows what kind of information will be input to and output from the system, how the data will advance through the system, and where the data will be stored. It does not show information about process timing or whether processes will operate in sequence or in parallel, unlike a traditional structured flowchart which focuses on control flow, or a UML activity workflow diagram, which presents both control and data, flows as a unified model.

3.5.1 Level 0 DFD

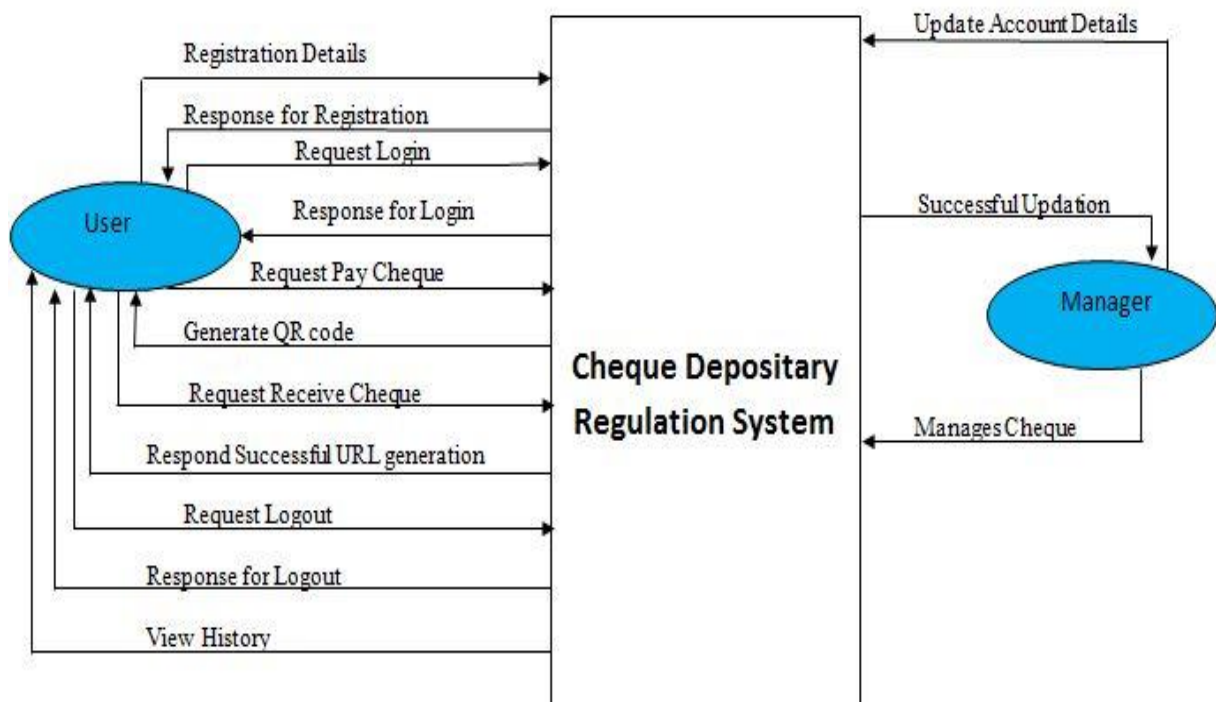


Figure 3.11 Level 0 DFD

A context diagram is a top **level** (also known as "**Level 0**") **data flow diagram**. It only contains one process node ("Process 0") that generalizes the function of the entire system in relationship to external entities. Draw data flow diagrams can be made in several nested layers as in Figure 3.11.

3.5.2 Level 1 DFD

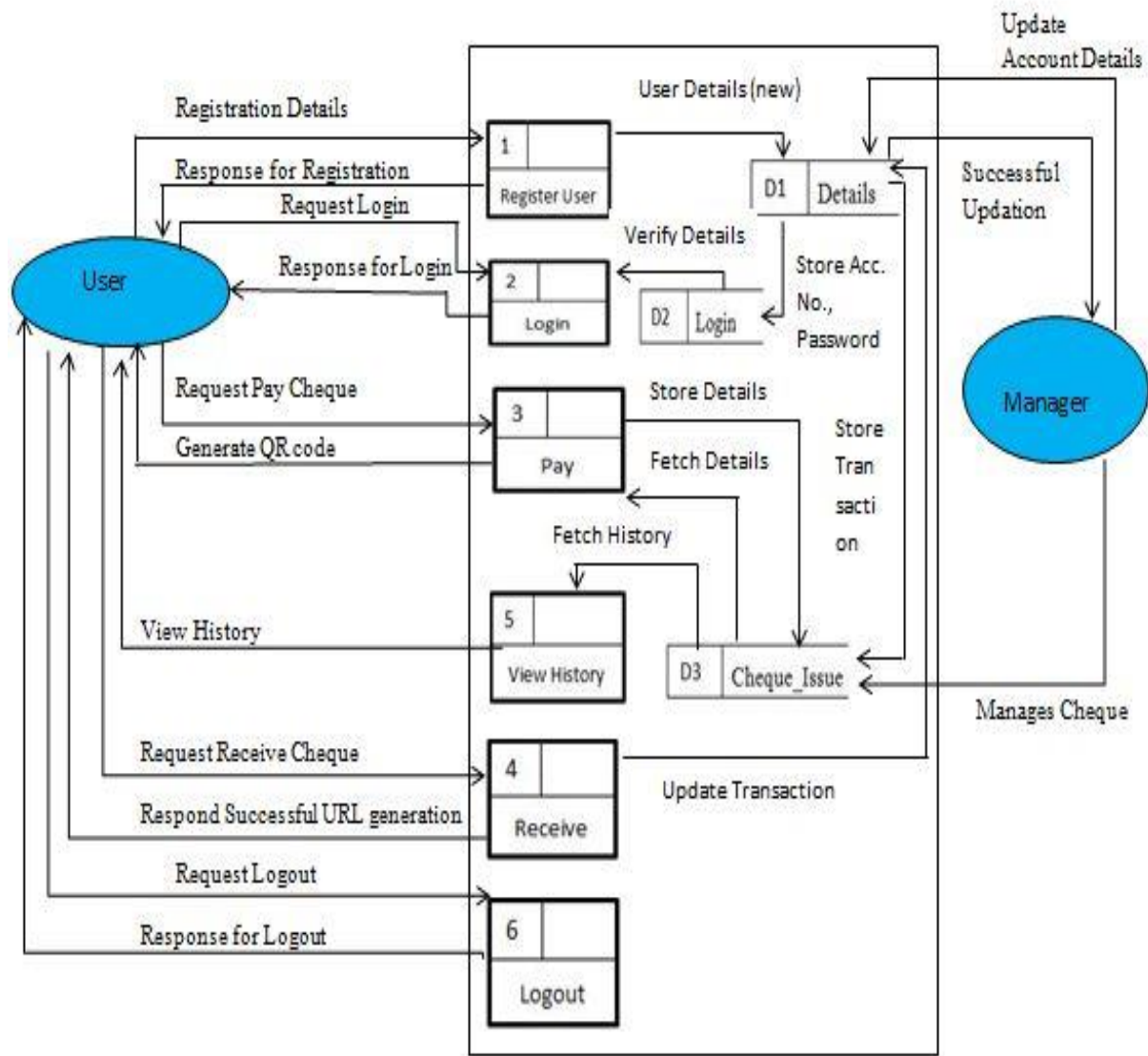


Figure 3.12 Level 1 DFD

The Level 1 DFD shows how the system is divided into sub-systems (processes), each of which deals with one or more of the data flows to or from an external agent, and which together provide all of the functionality of the system as a whole as in Figure 3.12.

3.5.3 Level 2 DFD

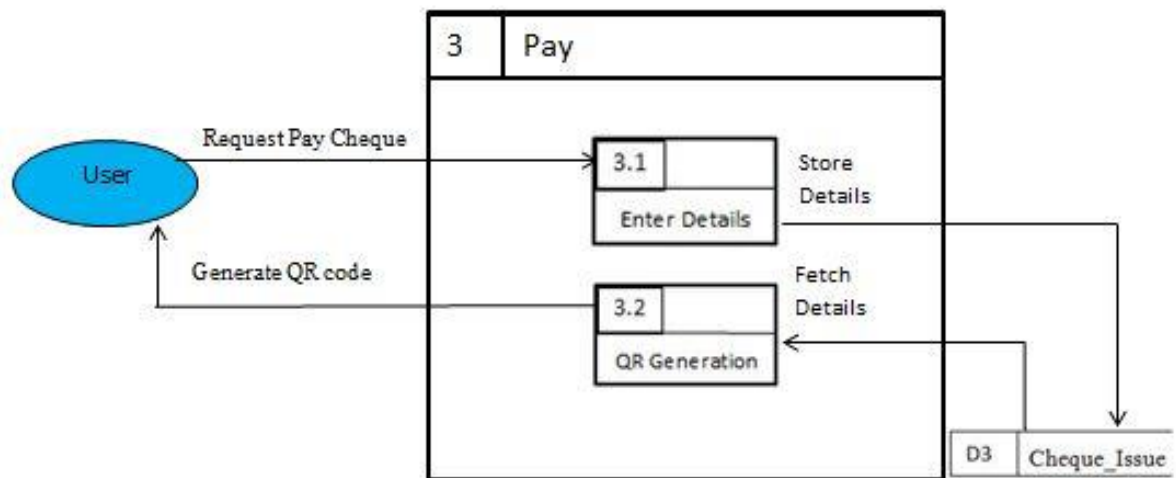


Figure 3.13(1) Level 2 DFD

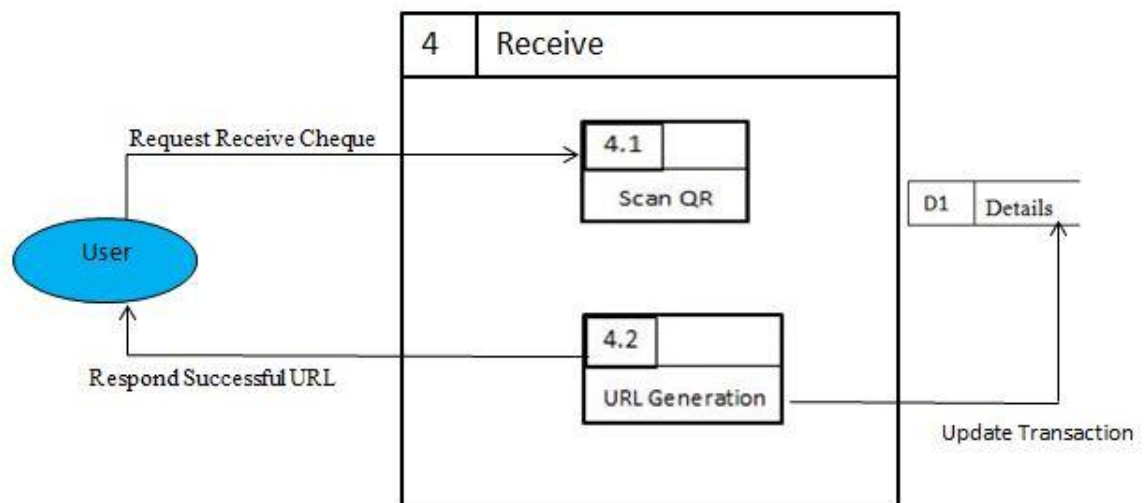


Figure 3.13(2) Level 2 DFD

A level 2 data flow diagram (DFD) offers a more detailed look at the processes that make up an information system than a level 1 DFD does. It can be used to plan or record the specific makeup of a system as in Figure 3.13(1) and 3.13(2).

3.6 Database

3.6.1 Entity Relationship Diagram

An entity–relationship model (ER model) describes inter-related things of interest in a specific domain of knowledge. An ER model is composed of entity types (which classify the things of interest) and specifies relationships that can exist between instances of those entity types. An entity–relationship model is usually the result of systematic analysis to define and describe what is important to processes in an area of a business. It does not define the business processes; it only presents a business data schema in graphical form. It is usually drawn in a graphical form as boxes (*entities*) that are connected by lines (*relationships*) which express the associations and dependencies between entities as shown in Figure 3.14.

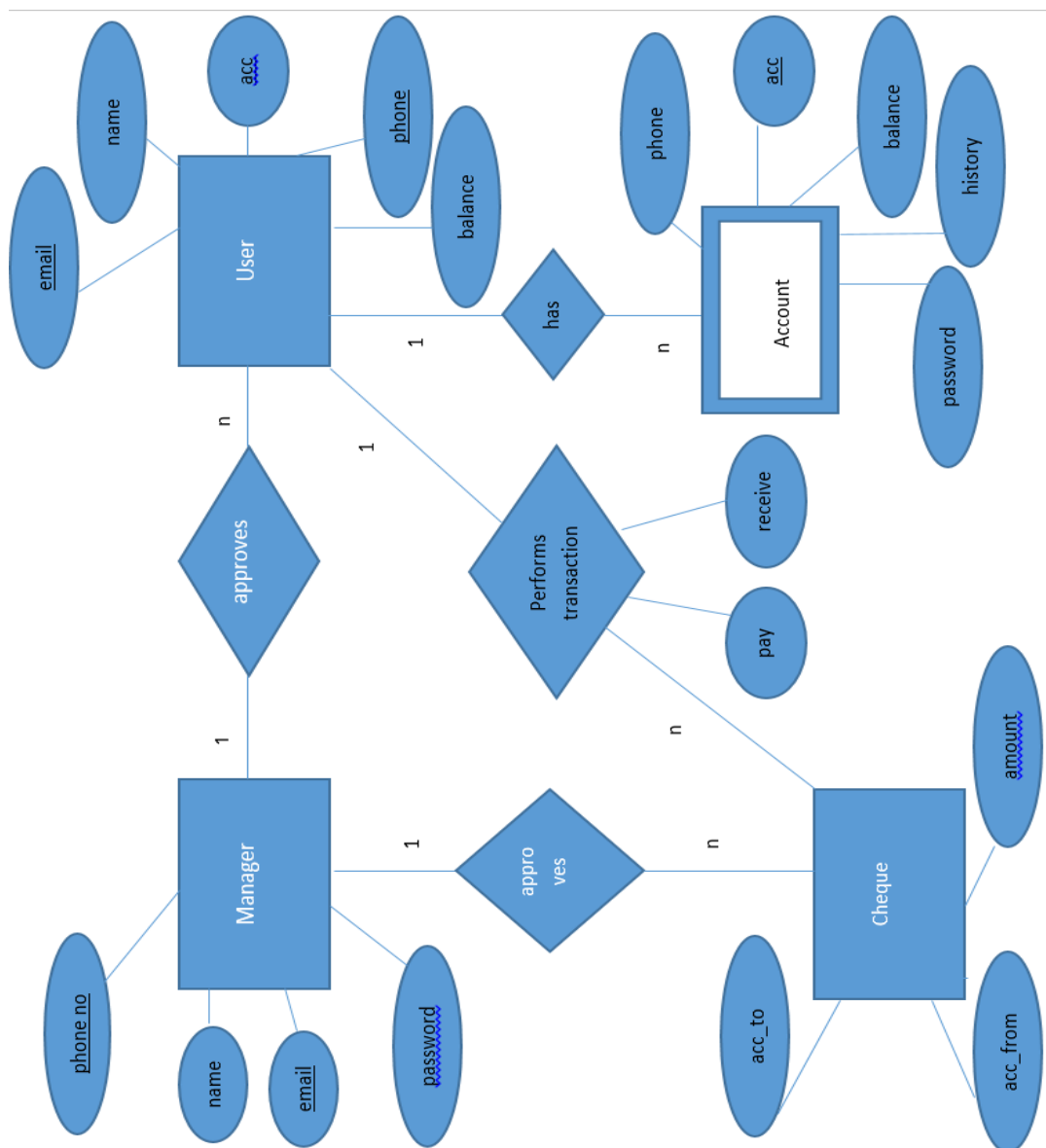


Figure 3.14 Entity Relationship Diagram

3.5.2 Database Tables

The following tables (Table 3.1, 3.2 and 3.3) show the database tables of the system.

Table 3.1 Login

id	acc	password
11	42007	tanmay123
12	008	iyoti123
13	009	deepak123
14	1996	sakshi123

Table 3.2 Details

id	acc	name	email	phone	balance
10	201997	Divya Shah	divya20cancer@gmail.com	9473585479	10000
11	42007	Tanmay Sah	tanmay123@gmail.com	8963950021	11550
12	008	Jyoti Sah	iyotisah67@gmail.com	9451250315	85500
13	009	Deepak Shah	deepakjyotiko@gmail.com	9450654187	81000
14	1996	Sakshi Saxena	saxena.sakshi@gmail.com	7247368398	5000

Table 3.3 Cheque_Issue

id	acc_from	acc_to	image_name	qr_data	amount	cheque_id
48	42007	008	NULL	NULL	500	NULL
49	42007	009	NULL	NULL	1000	NULL
50	008	42007	NULL	NULL	5000	NULL

3.7 Snapshots

3.7.1 Start Activity

This activity shows the start page where the user is given the choice (Figure 3.15).

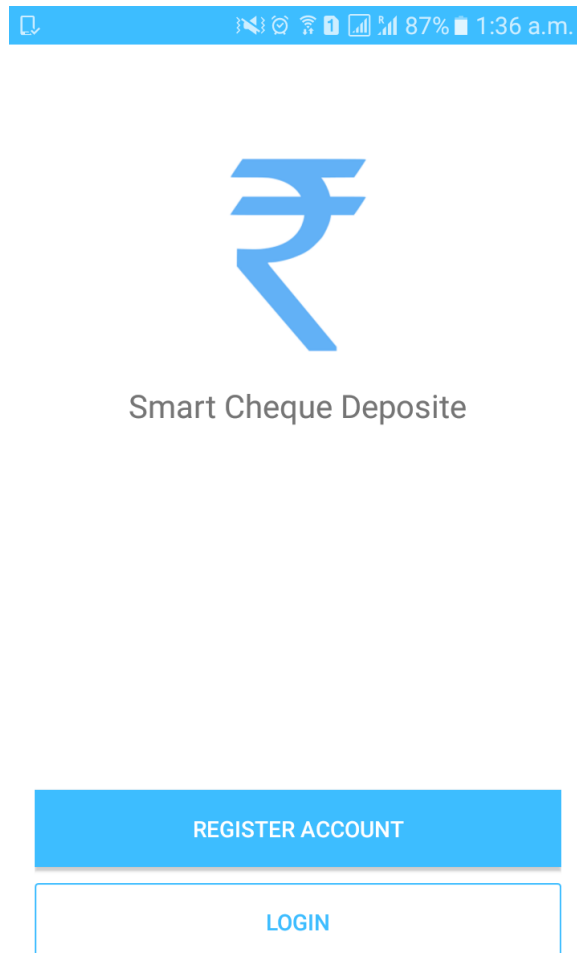


Figure 3.15 Start Activity

3.7.2 Registration Activity

This activity shows the registration form where the user can register by entering the detail (Figure3.16).

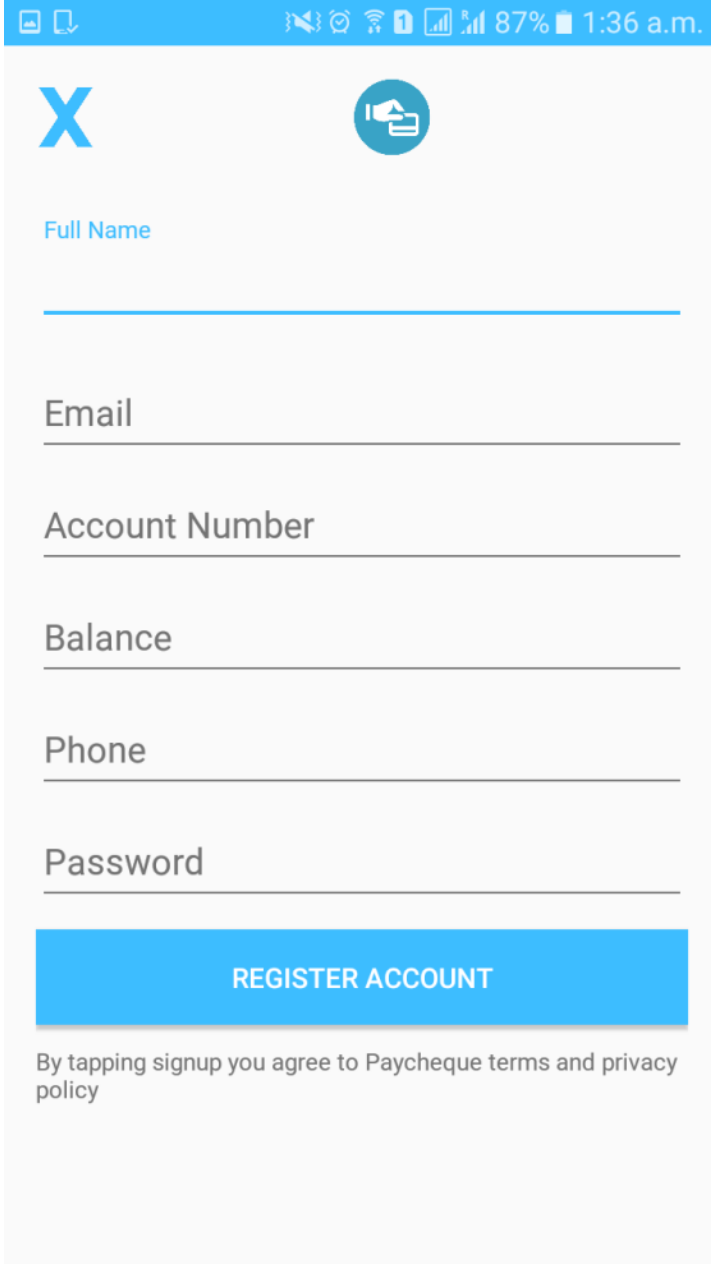
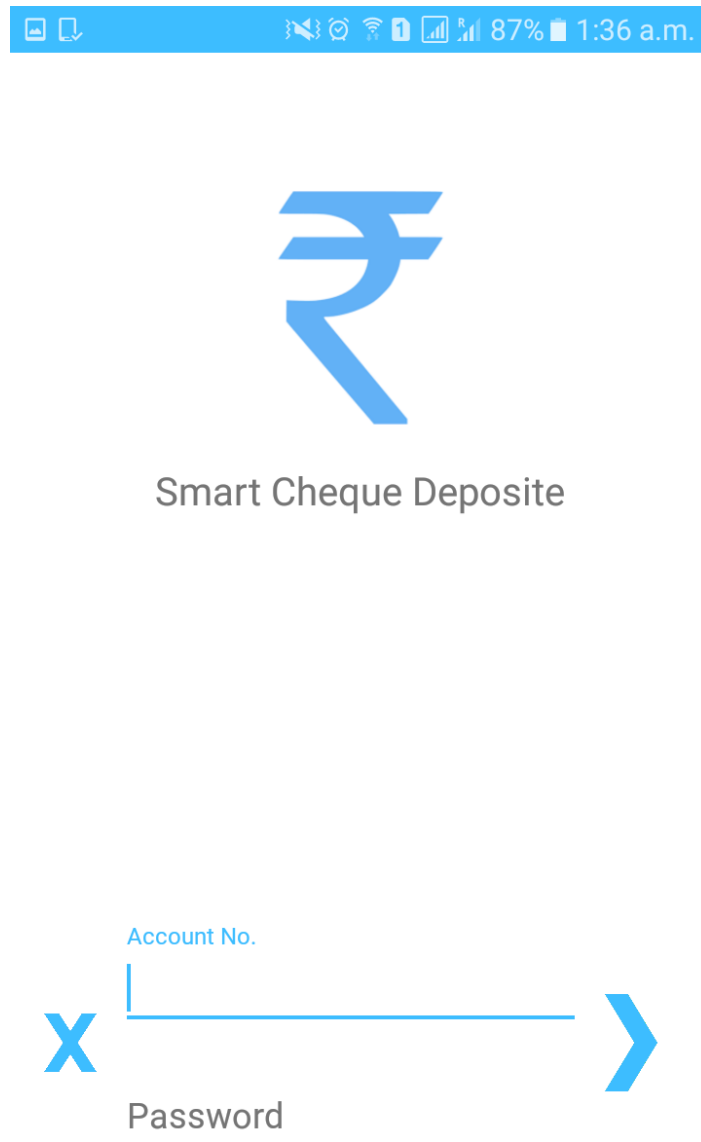
A screenshot of a mobile application's registration screen. At the top is a blue status bar with icons for signal, Wi-Fi, battery, and time (1:36 a.m.). Below the status bar is a header with a large blue 'X' logo on the left and a circular icon with a hand holding a card on the right. The main form area has a light gray background and contains six text input fields with labels: 'Full Name', 'Email', 'Account Number', 'Balance', 'Phone', and 'Password'. Each label is in blue text above its corresponding input field. Below the input fields is a prominent blue button with the text 'REGISTER ACCOUNT' in white. At the bottom of the form, there is a line of small gray text: 'By tapping signup you agree to Paycheque terms and privacy policy'.

Figure 3.16 Registration Activity

3.7.3 Login Activity

This activity shows the login page where the registered user can login by entering the details(Figure 3.17).



The image shows a mobile application interface for "Smart Cheque Deposit". At the top is a blue status bar with icons for camera, gallery, notifications, Wi-Fi, cellular signal, and battery at 87% with the time 1:36 a.m. Below the status bar is a large blue Indian Rupee symbol (₹). Underneath the symbol, the text "Smart Cheque Deposit" is displayed in a grey font. The login form consists of two input fields. The first field is labeled "Account No." in blue text and has a blue "X" icon to its left. The second field is labeled "Password" in grey text. To the right of both input fields is a large blue right-pointing arrow.

Figure 3.17 Login Activity

3.7.4 Home Page Activity

This activity shows the home page where the registered user can pay , receive , view history and logout of the application (Figure 3.18).

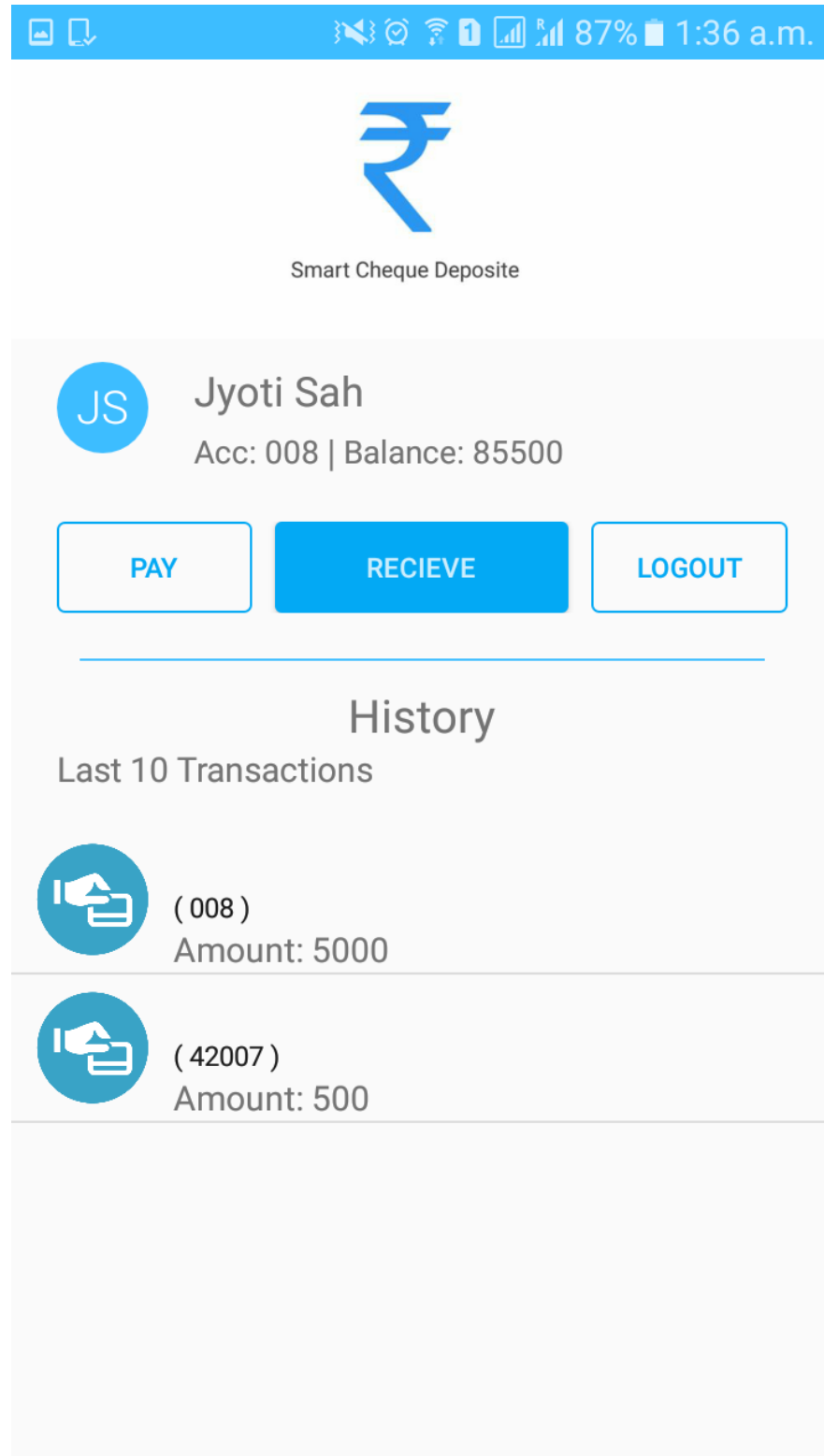
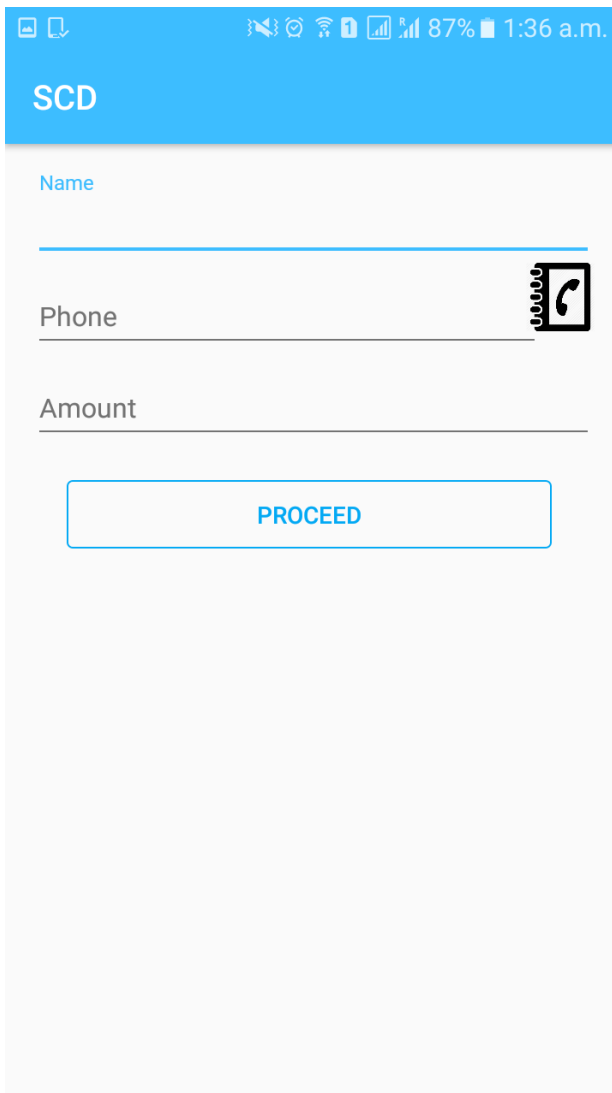


Figure 3.18 Home Page Activity

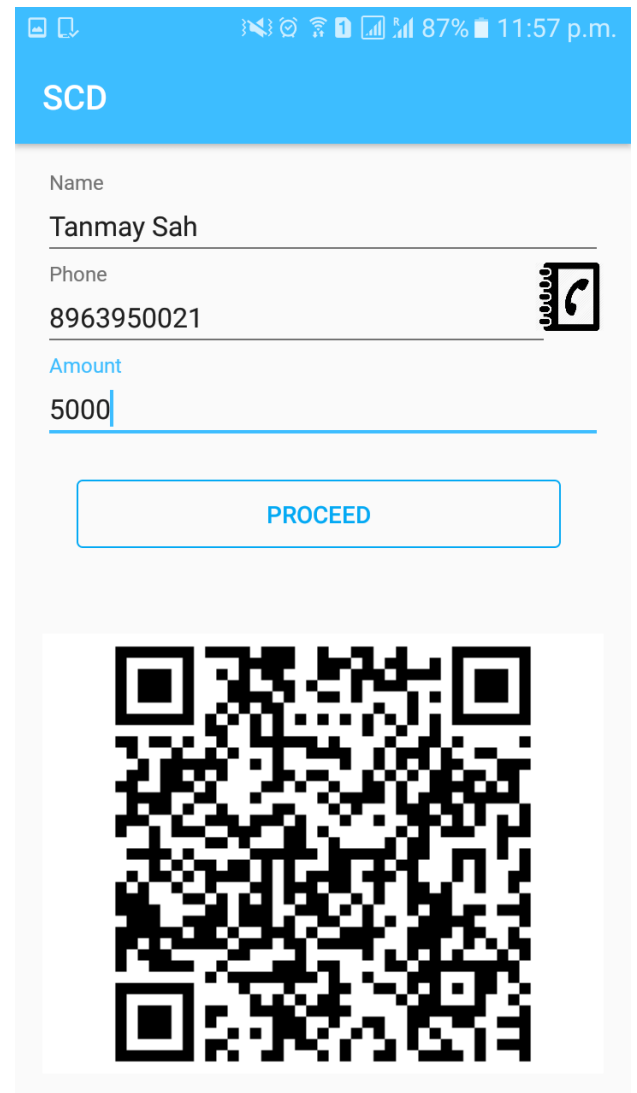
3.7.5 Pay Activity

This activity shows the paying of cheque where the registered user enter the details and on pressing the Proceed button a QR code is generated(Figure 3.19 1nd 3.20).



The screenshot shows the SCD application interface. At the top, there is a blue header with the text "SCD". Below the header, there are three input fields: "Name", "Phone", and "Amount". Each field has a corresponding label above it. To the right of the "Phone" field, there is a small icon of a telephone handset. Below the input fields, there is a blue button labeled "PROCEED". The status bar at the top of the screen shows the time as 1:36 a.m. and the battery level as 87%.

Figure 3.19 Pay Activity(Before entering details)



The screenshot shows the SCD application interface after the user has entered details. The "Name" field now contains the text "Tanmay Sah", the "Phone" field contains "8963950021", and the "Amount" field contains "5000". The "PROCEED" button is still present. Below the button, a QR code is displayed. The status bar at the top of the screen shows the time as 11:57 p.m. and the battery level as 87%.

Figure 3.20 Pay Activity(After entering details)

3.7.6 Receive Activity

This activity shows the receiving of cheque where the registered user presses the Receive button and then a QR scanner appears (Figure 3.21, 3.22, 3.23 and 3.24).

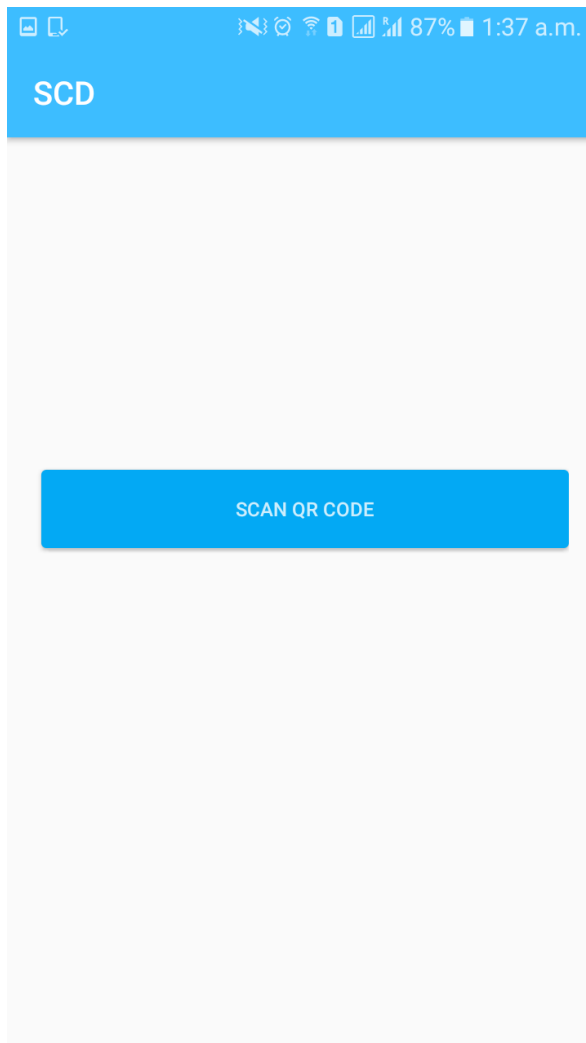


Figure 3.21 Receive Activity

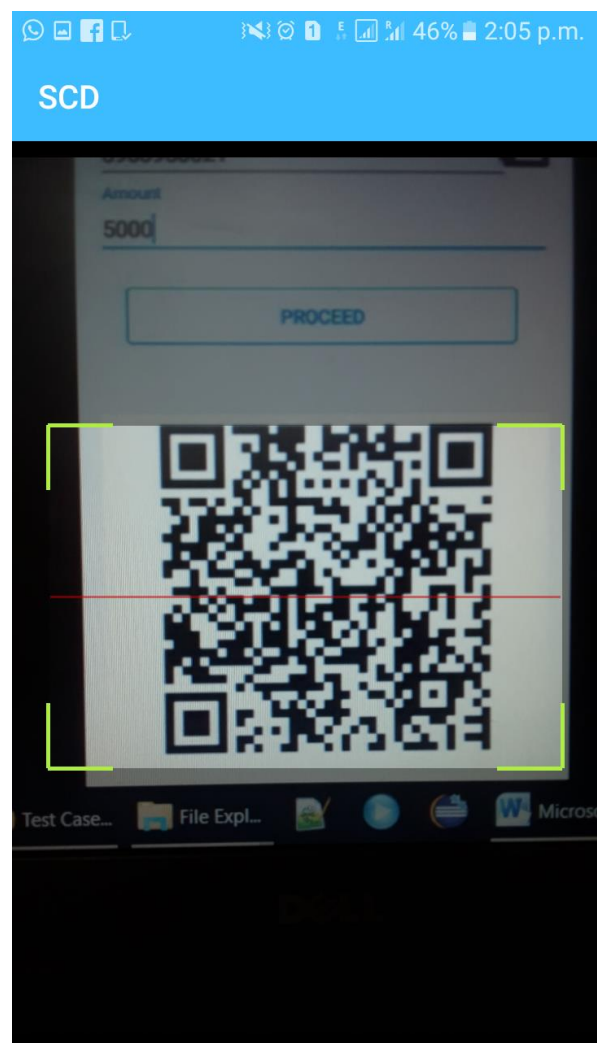


Figure 3.22 QR scanner Activity

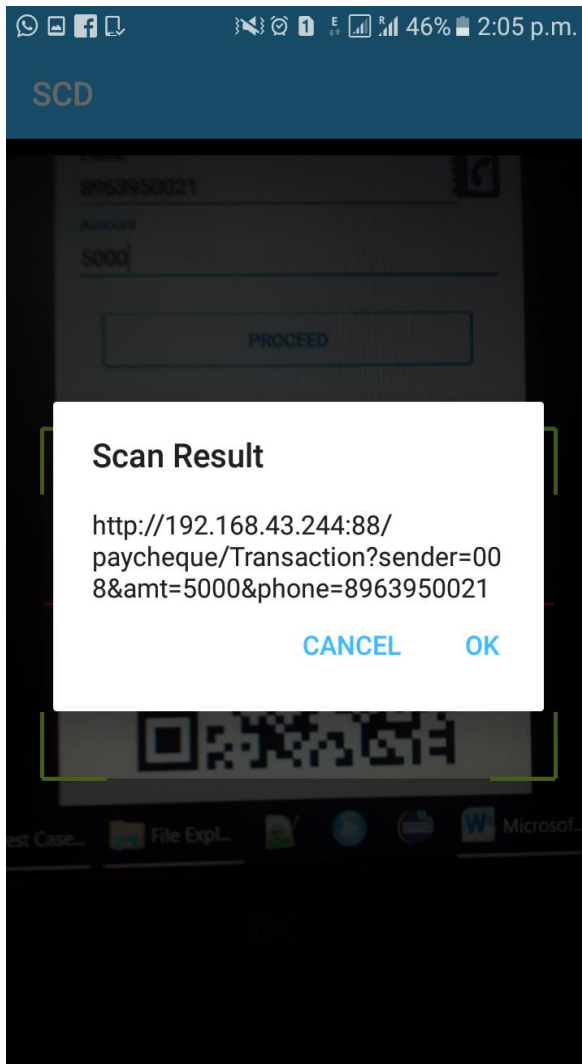
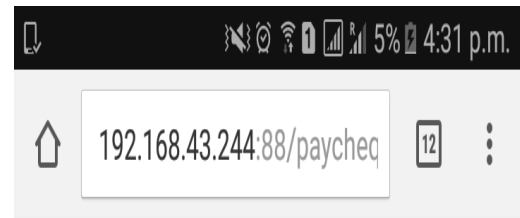


Figure 3.23 URL fetched



Successful

Figure 3.24 Successful Transaction

3.7.7 Logout Activity

This activity shows that if we have not logged out then security is provided (Figure 3.25).

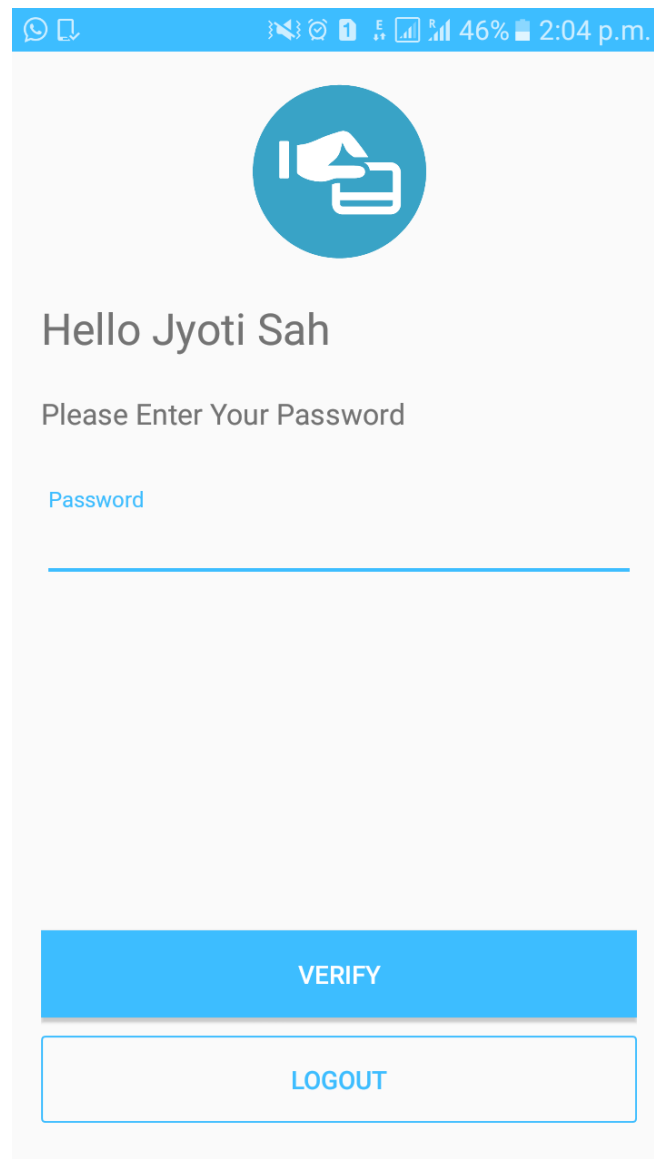


Figure 3.25 Logout Activity

CHAPTER 4 – TESTING PHASE

4.1 Types of Testing

There are different methods that can be used for software testing. This chapter briefly describes the methods available.

Black-Box Testing

Is a testing strategy in which the functionality of the software is tested without any reference to the internal design, code, or algorithm used in the program. Black box testing strategy ignores the internal mechanism of a system or component and focuses solely on the outputs generated in response to selected inputs and execution conditions. A testing strategy without having any knowledge of the internal workings of the application is Black Box testing strategy. The tester is oblivious to the system architecture and does not have access to the source code. Typically, when performing a black box test, a tester will interact with the system's user interface by providing inputs and examining outputs without knowing how and where the inputs are worked upon. It is also called functional testing.

White-Box Testing

This is a testing strategy developers takes into account the internal mechanism of a system, application or component to check if the source code is working as expected or not . The white box testing strategy incorporate coverage of the code written, branches, paths, statements and internal logic of the code, etc. However, in order to implement white box testing strategy, the tester has to deal with the code, and hence is required to possess knowledge of coding and logic i.e., internal working of the code. It also called structural testing, glass box testing or clear box testing

Grey-Box Testing

Grey-box testing is a technique to test the application with having a limited knowledge of the internal workings of an application. In software testing, the phrase the more you know, the better carries a lot of weight while testing an application.

Mastering the domain of a system always gives the tester an edge over someone with limited domain knowledge. Unlike black-box testing, where the tester only tests the application's user interface; in grey-box testing, the tester has access to design documents and the database. Having this knowledge, a tester can prepare better test data and test scenarios while making a test plan.

4.2 Test Cases

4.2.1 User Registration

Input- Valid user details

Status-

1. Valid
2. Invalid

Expected output-

1. User is registered successfully.
2. Generates an error corresponding to the particular field.

Input- Fields left empty

Status-

1. Invalid

Expected output-

1. Generates an error displaying-“Please fill out the fields”.

4.2.2 Login

Input-Valid account no and password

Status-

1. Valid
2. Invalid

Expected output-

1. Redirect you to user home page.
2. Generates an error displaying-“Invalid account no and password”.

Input-Pay

Status-

1. Valid

Expected output-

1. On entering name ,phone no, amount generates a unique qr code.

Input-Receive

Status-

1. Valid
2. Invalid

Expected output-

1. If the details are correct will take you to the url for successful transaction.
2. If any of the field's data does not match exactly to the person you are paying results in "unsuccessful transaction".

Input-History**Status-**

1. Update

Expected output-

1. If any of the transaction occurs successfully history is updated.

Input- If not logout and close application.**Status-**

1. Requests for password

Expected output-

1. If password entered is correct you are logged in otherwise not.

Input-Logout**Status-**

1. Valid

Expected output-

1. Logs you out of the application.

CHAPTER 5 - CONCLUSION

- A systematic and easy way of depositing a cheque is much needed application in the present area. If the small difficulties and problems can be eliminated, the software will be of great help in the real life scenario.
- We gained a good knowledge about the industrial proceeding while executing any project. The project work was very fruitful for us and will definitely help in the long run. As it provides an experience about the technical world and also an exposure to the competitive environment. I consider this project work as a process of formulation for young engineers to stand confidently in the technical world.

CHAPTER 5: FUTURE WORK

5.1 Further improvements

Any shortcomings or technical failure will be dealt with after the version 2.1 release of the app to make improvements in future release. More Banking features will also be added further.

5.2 Further areas of working

Website portal with other banking features

In future we will be developing web application for cheque deposition.

APPENDIX

A. Project Proposal

Currently the cheque deposit process starts by manually depositing the cheque in any branch of the same bank, then it is sent for authorization and finally it gets submitted. Our system will improve upon the present scenario. At the doorstep, the user will be facilitated with the services by digitally submitting the cheque through internet and it will be verified manually by the bank. For this purpose the cheque will contain payee details and there will be a unique QRcode which will have the information of the registered payer. Authentication and validation will be done by QR Codes. The user will also be able to check the current status and history of deposited cheques.

Expected outcome at the end of 7th Semester:

- Android app for smart phone users
- Cheque Depository Regulation System

REFERENCES

Links referred

- [https://www.quora.com/What-is-the-process-for-clearing-a-cheque-in India](https://www.quora.com/What-is-the-process-for-clearing-a-cheque-in-India) [Last Accessed Date: 8/9/2017]
- https://en.wikipedia.org/wiki/QR_code
[Last Accessed Date: 11/11/2017]
- <http://www.javatpoint.com/android-tutorial>
[Last Accessed Date: 15/11/2017].
- <http://www.tutorialspoint.com/android/>
[Last Accessed Date: 20/11/2017].

STUDENT DETAILS

Name: - Divya Shah

Enrollment No.: - 141214

Branch: - Computer Science and Engineering

Course: - B.Tech

Batch: - 2014-2018

Contact No.: 9473585479

Email Id: divya1997shah@gmail.com



Name: - Khyati Kachhawah

Enrollment No.: - 141239

Branch: - Computer Science and Engineering

Course: - B.Tech

Batch: - 2014-2018

Contact No.: -9929708670

Email Id: khyatikachhawah@gmail.com



Name: - Sakshi Saxena

Enrollment No.: - 141311

Branch: - Computer Science and Engineering

Course: - B.Tech

Batch: - 2014-2018

Contact No.: -9479962193

Email Id: - saxena.sakshigwl06@gmail.com

