CHAPTER 1

1. INTRODUCTION

1.1 Motivation

Food waste is considered as one of the predominant issue now a day. One third of the food produced gets wasted around the globe. Our streets, garbage bins and landfills are the best example to prove it. We see that millions of lives are starving for food around us. Hunger kills more people every year than any other deadly diseases.

In highly populated countries like India, food wastage is a disturbing issue. The streets, garbage bins and landfills have ample proof to prove it. Marriages, canteens, restaurants, social and family get-togethers and functions expel out so much food. Food wastage is not only an indication of hunger or pollution, but also of many economic problems. The high standard of living has resulted in the wastage of food because of quick changes in habits and lifestyle. Instead of wasting these things we can put them in use by donating them to various organizations such as orphanages, old age homes, etc. The product is a web application which aims to establish a link between donors and the charity homes/needy households to enable excess food donation.

Food donation portal is internet-based application that provides a platform for donating leftover food to all needy people/organizations. The portal is shown to be an effective means of donating food to organizations, etc. over the internet. In highly populated countries like India, food wastage is a big issue. The garbage bins, streets and landfills have sample proof to prove it. Canteens, marriages, restaurants, social and family get-togethers and functions expel out so much food. Food wastage is not only an indication of pollution or hunger, but also of many economic problems. The high standard of living has resulted in the wastage of food. Because of quick changes in habits and lifestyle instead of wasting food, the system can put them in use by donating them to various organizations such as old age homes, orphanages, etc. The product is an internet-based application that basically aims at charity through donations.

Many people, institutes wish to donate things to needy organizations. Also, many organizations wish to ask for various things required by them such as clothes, food grains, utensils, etc. But there is no source available through which they can satisfy their requirements. Thereby, an application has been developed through which people can donate food items as per their capacity and the application also allows organizations to put up their requests, that is items required by them, if any.

1.2 Problem Definition

The problem definition of the project is to bridge the gap between the hunger and the food waste through donation. This mobile based application allow excess food to be donated to the charities and orphans and feed the people starving for food.

1.3 Objective of Project

The objective of the project is to bridge the gap between the hungry and the food waste through donation. This mobile based application allows excess food to be donated to the charities and orphans and feed the people starving for food. The application would store and display basic information about the inventory contents and alerts the user of the food products which are due to expire the next day. Consequentially, users may take actions to avoid the concerned products get wasted or spoiled. It is believed that a considerable amount of food waste would be avoided in households if the occupants are well-informed of the timeline of their food stocks. Provisions have also been made to allow for the multi-device use.

Most food management applications available are mainly concerned with helping users watch their weight and food in-take and generally requires lots of information from user. The advantage of this project is the use of the simplest information of food products to monitor the inventory. With an eye on the future, a demo solution was integrated to show compatibility with future advancement in food packaging.

1.4 Limitations of Project

Sometimes before delivery food can be wasted because of weather conditions.

CHAPTER 2
LITERATURE SURVEY

2.1 Existing System

Currently, people donate stuff manually by visiting each organization number of times. In order to reduce the problems of food wastage, some websites like rescuing leftovercuisine and annakshetra have taken efforts to help people donate their surplus food to shelters through their official website, wherein people can donate food, funds and also volunteer for various activities.

- Most people don't realize how much food they throw away every day — from uneaten leftovers to spoiled produce [1].
- About 95 percent of the food we throw away ends up in landfills or combustion facilities. In 2013, we disposed more than 35 million tons of food waste [2].
- The majority of the population today uses smartphones with active internet connection, which is the basic requirement for this product to function properly [3] .

2.2 Disadvantages of Existing Systems

- Over purchasing of food items.
- Donor lack of knowledge about organizations who collects waste food.
- Waste of money.
- Impacts in environment.
- Needy people starving for food for survival.

2.3 Proposed System

The proposed application is android-based, developed on Android Studio using java and xml requires internet connection and will provide a platform for donors and seekers after they successfully register into the system. If a user wishes to donate something, he/she can share the details in application. This message will be shown as notification in donations tab to the agents.

This detail will be stored in back end in the database. Once a notification is sent, the orphanages who wish to claim the donations can reply to the donor and contact the donor.

CHAPTER 3
ANALYSIS

3.1. The study of the system:

To conduct studies and analysis of an operation and technical nature.To promote the exchange and development of methods and tools for operational analysis as applied to defence problems.

3.1.1 Logical design:

The logical design of a system pertains to an abstract representation of the data flows, inputs and outputs of the system. This is often conducted via modelling, using an over-abstract (and sometimes graphical) model of the actual system. In the context of systems design are included. Logical design includes ER diagrams i.e. Entity Relation Diagrams.

3.1.2 Physical design:

The physical design relates to the actual input and output processes of the system. This is laid down in terms of how data is input into a system, how it is verified / authenticated, how it is processed, and how it is displayed as output. In physical design, following requirements about the system are decided.

1. Input requirements

2. Output requirements

3. Storage requirements

4. Processing requirements

5. System control and backup or recovery.

The physical portion of systems designs can generally be broken down into three sub-tasks:

1. User interface design

2. Data design

3. Process design

User Interface design is concerned with how users add information to the system and with how the system presents information back to them. Data design is concerned with how the data is represented and stored within the system. Finally, Process design is concerned with how data moves through the system, and how and where it is validated, secured and/or transformed as it flows into, through and out of the system. At the end of the system design phase, documentation describing the three sub-tasks is produced and made available for use in the next phase.

Physical design, in this context does not refer to the tangible physical design of an information system. To use an analogy, a personal computer’s physical design involves input via a keyboard, processing within the CPU, and output via a monitor, printer, etc. It would not concern the actual layout of the tangible hardware, which for a PC would be a monitor, CPU, motherboard, hard drive, modems, videos/graphic cards, USB slots, etc. It involves a detailed design of a user and a product database structure processor and a control processor. The H/S personal specification is developed for the proposed system.

3.2 Software Requirement Specification

Requirements gives idea about what are necessary things that are needed for proposed system, which plays very important role in development of any system. This chapter deals with what are hardware components that are needed for system, application software that are required for the development of the system.

Software Requirements Specification is the starting point of software development activity. As system grew more complexity evident goal of the entire system cannot be easily comprehended. Hence the need for the requirement phase arose. The SRS is the means of translating the ideas in the minds of clients into a formal document. The purpose of the software requirement specification is to reduce the communication gap between clients and developers. SRS is the medium through which the client and user needs are accurately specified.

The requirement specification for any system can be broadly stated as given below:

The system should be able to interface with the existing system.

The system should be accurate.

The system should be better than the existing system.

3.2.1 Software Requirement

- IDE: Android Studio 2.1 or higher versions
- Languages: Java(JDK 1.8.0), XML(1.0)
- Database: SQLite
- UML Diagrams: Visual Paradigm(version 12.1)
- Operating System: Windows8 and above

3.2.2 Hardware Requirement

- Processor: core i5 or i3 processor
- RAM: 8GB or 16 GB
- Space on hard disk: Minimum 500GB
- Device: Android emulator , Mobile

3.3 Android installation Procedure

Setting up Android development environment takes some time at first. It helps to make sure you don't do anything wrong to save yourself from the agony of doing the whole process again.

You're required to have Windows XP or later, or Mac OS X 10.5.8 or a later version to start Android application development process. Then, there are four tools that you will need and they are available on the Internet for free:

1.Java JDK5 or JDK6

2.Android SDK

Step 1: Setup Java Development Kit (JDK)

You can download the JDK and install it, which is pretty easy. After that, you just have to set PATH and JAVA_HOME variables to the folder where you have **java** and **javac.**

Note for Windows Users: If you installed the JDK in C:\jdk1.6.0_15 then you will have to add the following two lines in your command prompt

C:\autoexec.bat file.

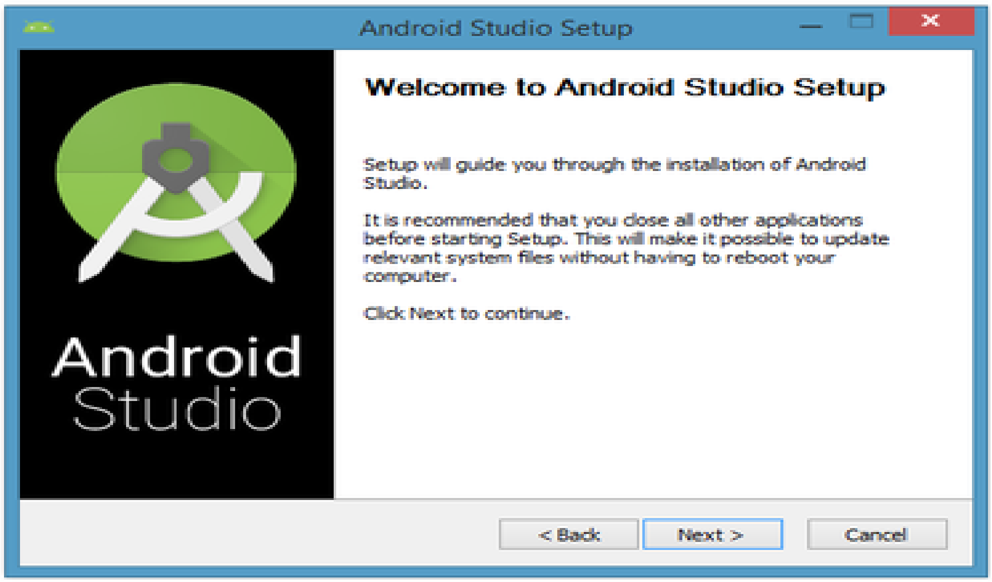
```
SetPATH=C:\jdk1.6.0_15\bin;%PATH%
set JAVA_HOME=C:\jdk1.6.0_15
```

Step 2: Downloading and setting up Android Studio

Google provides Android Studio for the Windows, Mac OS X, and Linux platforms. You can download this software from the Android Studio homepage. (You'll also find the traditional SDKs, with Android Studio's command-line tools, available from the Downloads page.)

Installing Android studio:

I launched android-studio-bundle-143.2821654-windows.exe to start the installation process. The installer responded by presenting the Android Studio Setup dialog box shown below.



Welcome to Android Studio Setup

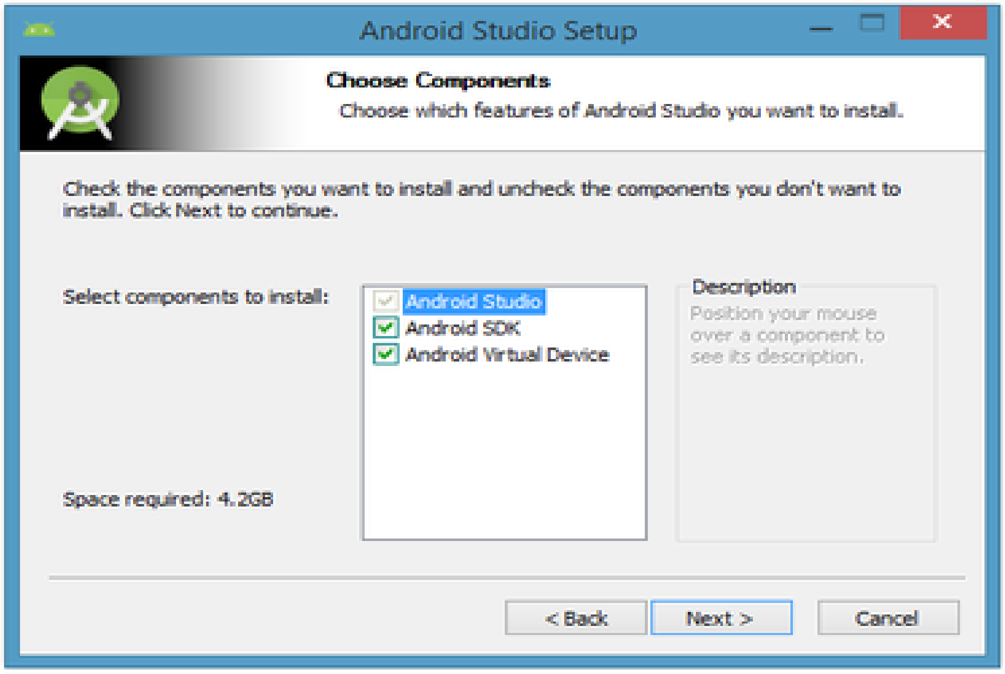
Setup will guide you through the installation of Android Studio.

It is recommended that you close all other applications before starting Setup. This will make it possible to update relevant system files without having to reboot your computer.

Click Next to continue.

Fig.3.3.1. Set up Android Studio

Clicking Next took me to the following dialog box, which gives you the option to decline installing the Android SDK (included with the installer) and an Android Virtual Device



Choose Components

Choose which features of Android Studio you want to install.

Check the components you want to install and uncheck the components you don't want to install. Click Next to continue.

Select components to install:

- [✓] Android Studio
- [✓] Android SDK
- [✓] Android Virtual Device

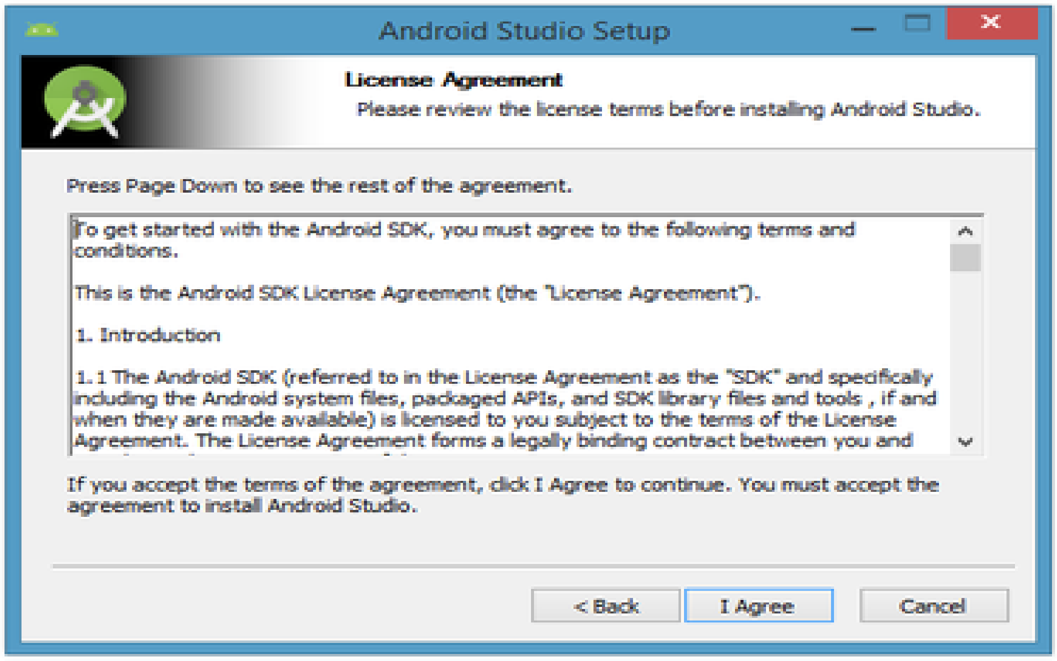
Description

Position your mouse over a component to see its description.

Space required: 4.2GB

Fig.3.3.2. Do you want to install the Android SDK and AVD

I choose to keep the default settings. After clicking Next, you'll be taken to the license agreement dialog box. Accept the license to continue the installation.



License Agreement

Please review the license terms before installing Android Studio.

Press Page Down to see the rest of the agreement.

To get started with the Android SDK, you must agree to the following terms and conditions.

This is the Android SDK License Agreement (the "License Agreement").

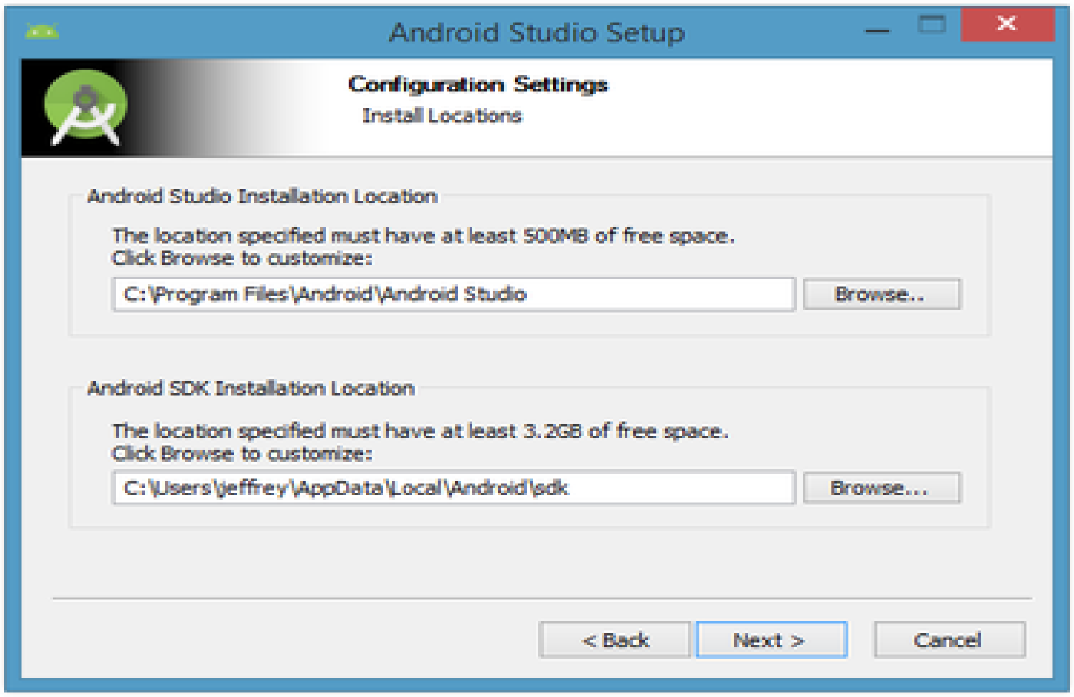
1. Introduction

1.1 The Android SDK (referred to in the License Agreement as the "SDK" and specifically including the Android system files, packaged APIs, and SDK library files and tools , if and when they are made available) is licensed to you subject to the terms of the License Agreement. The License Agreement forms a legally binding contract between you and

If you accept the terms of the agreement, click I Agree to continue. You must accept the agreement to install Android Studio.

Fig.3.3.3 Accept the license agreement to continue installation

The next dialog box invites you to change the installation locations for Android Studio and the Android SDK.



Android Studio Installation Location

The location specified must have at least 500MB of free space.
Click Browse to customize:

C:\Program Files\Android\Android Studio

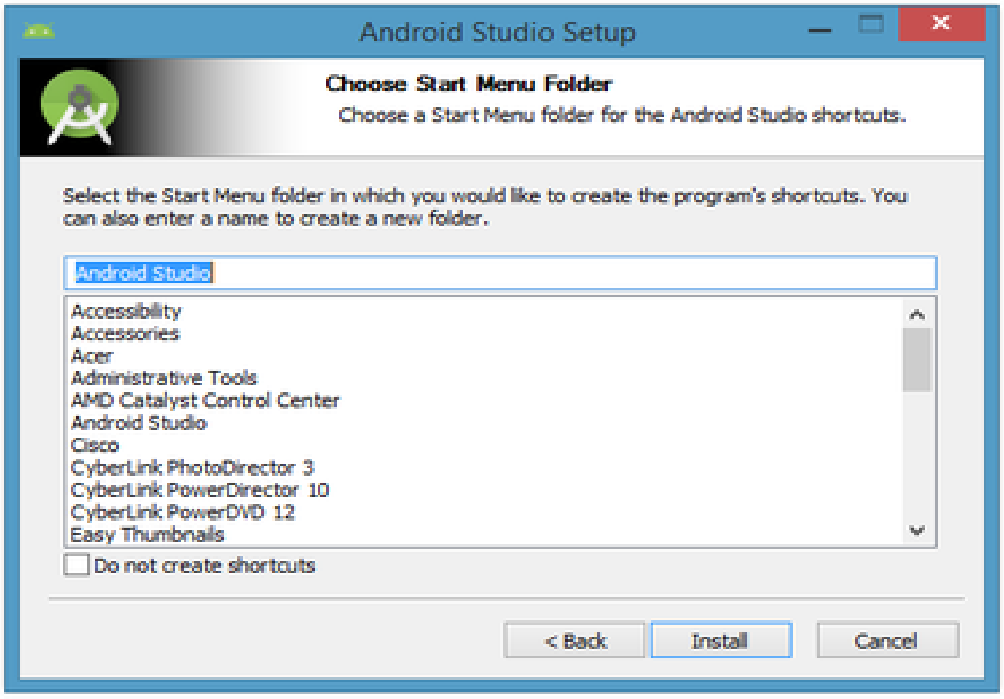
Android SDK Installation Location

The location specified must have at least 3.2GB of free space.
Click Browse to customize:

C:\Users\jeffrey\AppData\Local\Android\sdk

Fig.3.3.4. Set the Android Studio and Android SDK installation locations

Change the location or accept the default locations and click Next. The installer defaults to creating a shortcut for launching this program, or you can choose to decline. I recommend that you create the shortcut, and then click the Install button to begin installation



Select the Start Menu folder in which you would like to create the program's shortcuts. You can also enter a name to create a new folder.

Android Studio

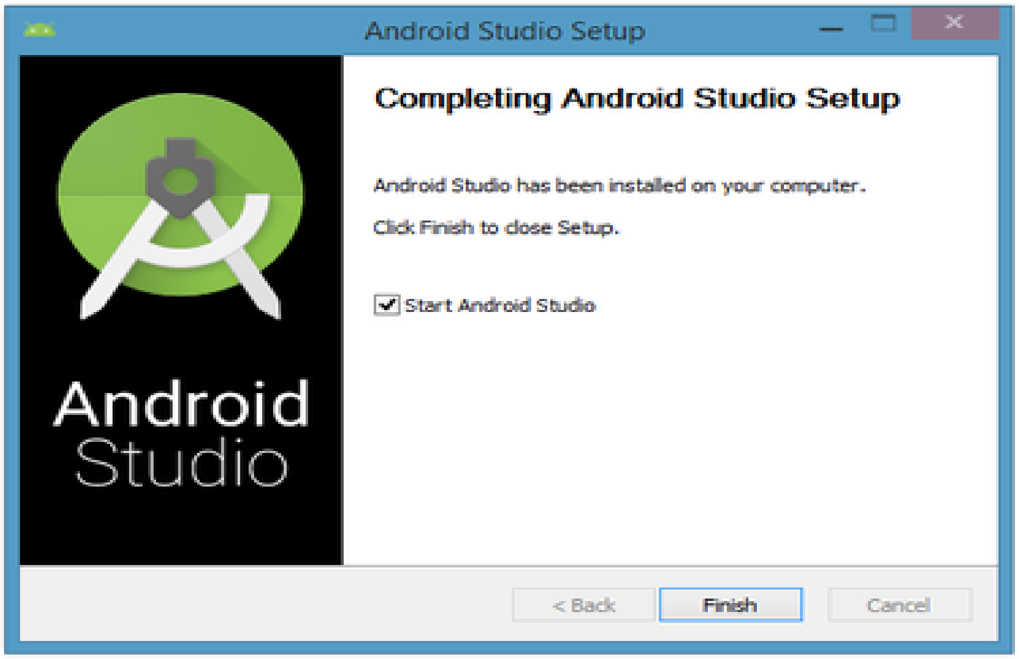
Accessibility
Accessories
Acer
Administrative Tools
AMD Catalyst Control Center
Android Studio
Cisco
CyberLink PhotoDirector 3
CyberLink PowerDirector 10
CyberLink PowerDVD 12
Easy Thumbnails

Do not create shortcuts

Fig.3.3.5. Create a new shortcut for Android Studio

The resulting dialog box shows the progress of installing Android Studio and the Android SDK. Clicking the Show Details button will let you view detailed information about the installation progress.

The dialog box will inform you when installation has finished. When you click Next, you should see the following:



Completing Android Studio Setup

Android Studio has been installed on your computer.

Click Finish to close Setup.

☑ Start Android Studio

Fig.3.3.6. Leave the Start Android Studio check box checked to run this software.

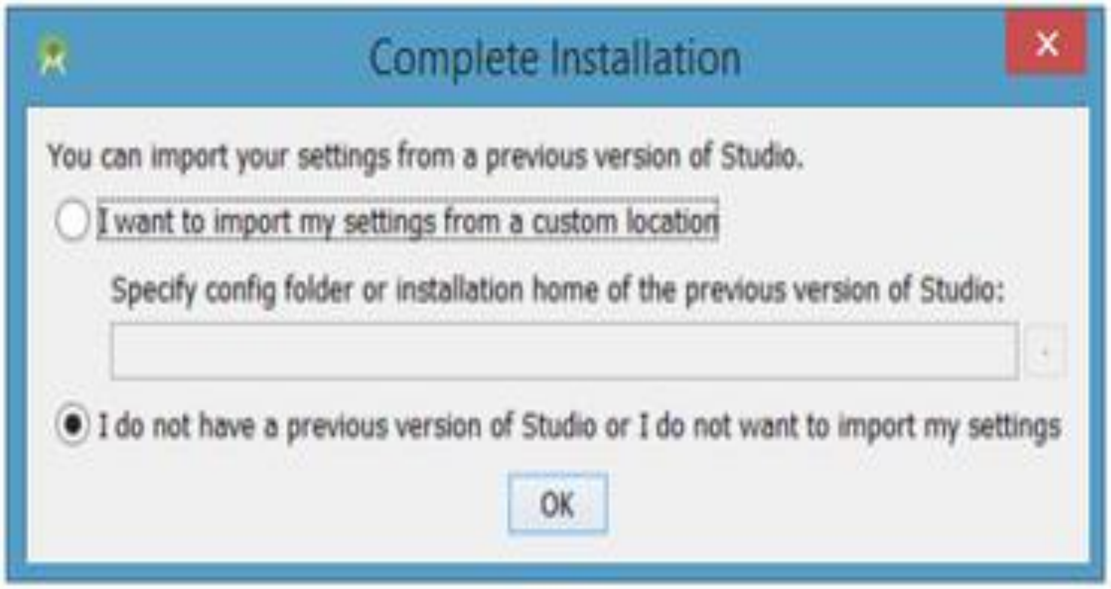
To complete your installation, leave the Start Android Studio box checked and click Finish.

Running Android Studio

Android Studio presents a splash screen when it starts running:

Fig.3.3.7. Android Studio's start screen

On your first run, you'll be asked to respond to several configuration-oriented dialog boxes. The first dialog box focuses on importing settings from any previously installed version of Android Studio.


Complete Installation

You can import your settings from a previous version of Studio.

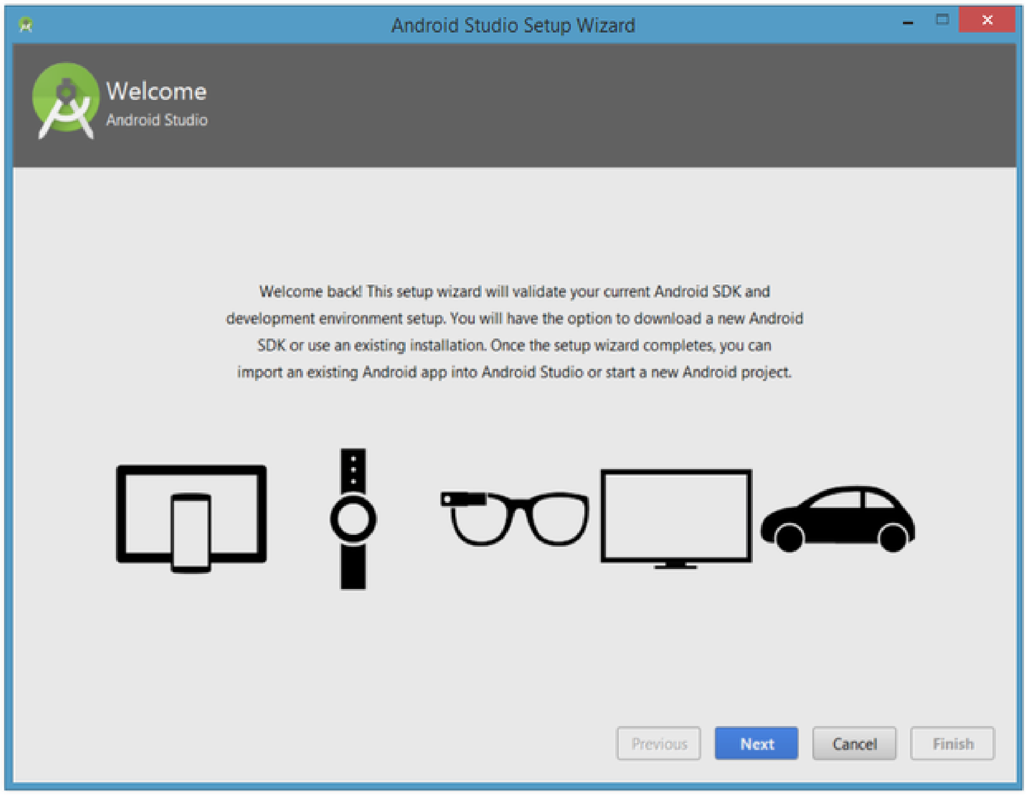
○ I want to import my settings from a custom location

Specify config folder or installation home of the previous version of Studio:

● I do not have a previous version of Studio or I do not want to import my settings

Fig.3.3.8. Import settings

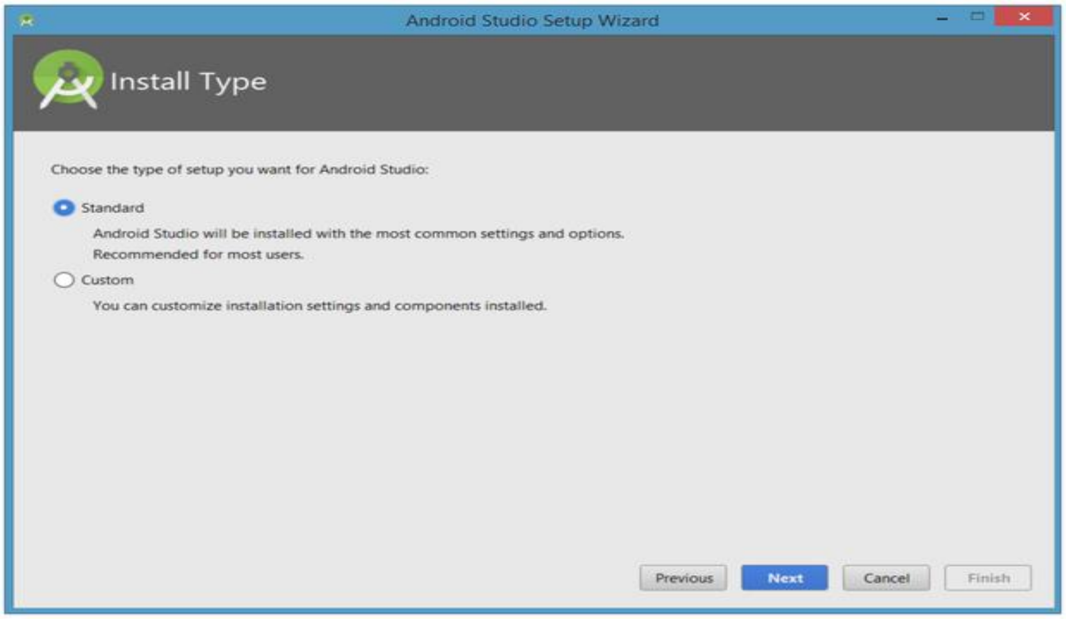
If you're like me, and don't have a previously installed version, you can just keep the default setting and click OK. Android Studio will respond with a slightly enhanced version of the splash screen, followed by the Android Studio Setup Wizard dialog box:



Welcome back! This setup wizard will validate your current Android SDK and development environment setup. You will have the option to download a new Android SDK or use an existing installation. Once the setup wizard completes, you can import an existing Android app into Android Studio or start a new Android project.

Fig.3.3.9. Validate your Android SDK and development environment setup

When you click Next, the setup wizard invites you to select an installation type for your SDK components. For now I recommend you keep the default standard setting.



Choose the type of setup you want for Android Studio:

● Standard

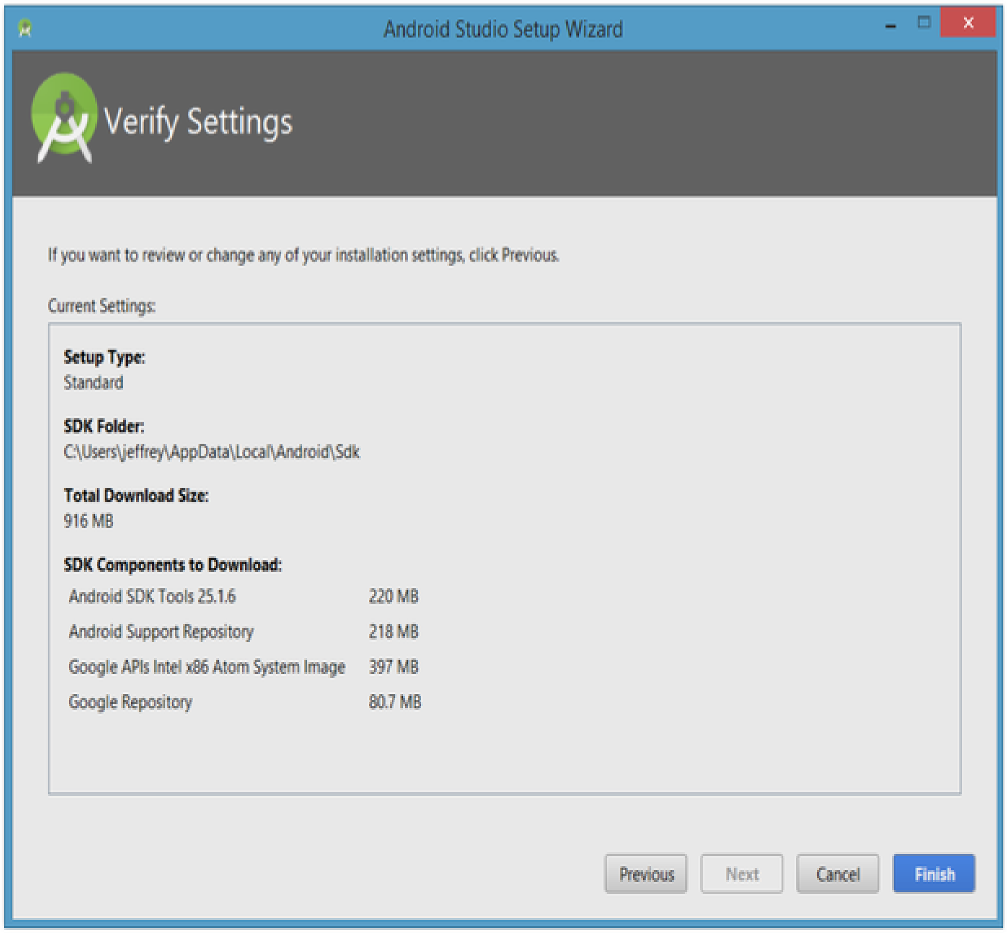
Android Studio will be installed with the most common settings and options.
Recommended for most users.

○ Custom

You can customize installation settings and components installed.

Fig.3.3.10. Choose an installation type

Click Next and verify your settings, then click Finish to continue.



If you want to review or change any of your installation settings, click Previous.

Current Settings:

Setup Type:
Standard

SDK Folder:
C:\Users\jeffrey\AppData\Local\Android\Sdk

Total Download Size:
916 MB

SDK Components to Download:

Android SDK Tools 25.1.6	220 MB
Android Support Repository	218 MB
Google APIs Intel x86 Atom System Image	397 MB
Google Repository	80.7 MB

Fig.3.3.11. Review settings

The wizard will download and unzip various components. Click Show Details if you want to see more information about the archives being downloaded and their contents.

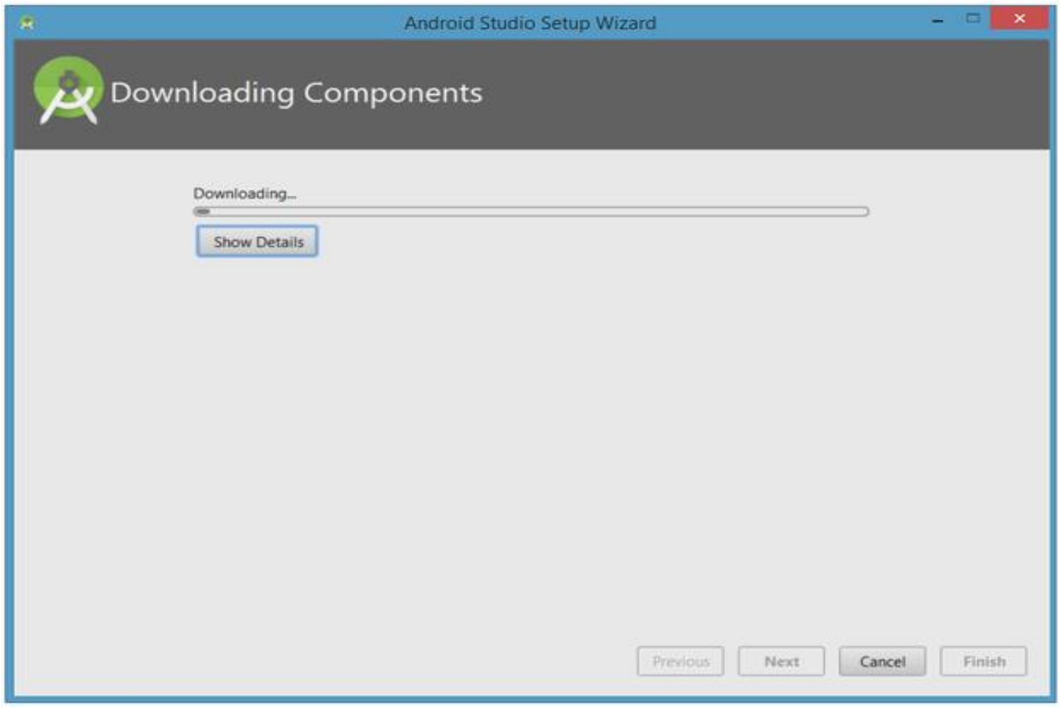


Fig.3.3.12. The wizard downloads and unzips Android Studio components

Finally, click Finish to complete the wizard. You should see the Welcome to Android Studio dialog box:

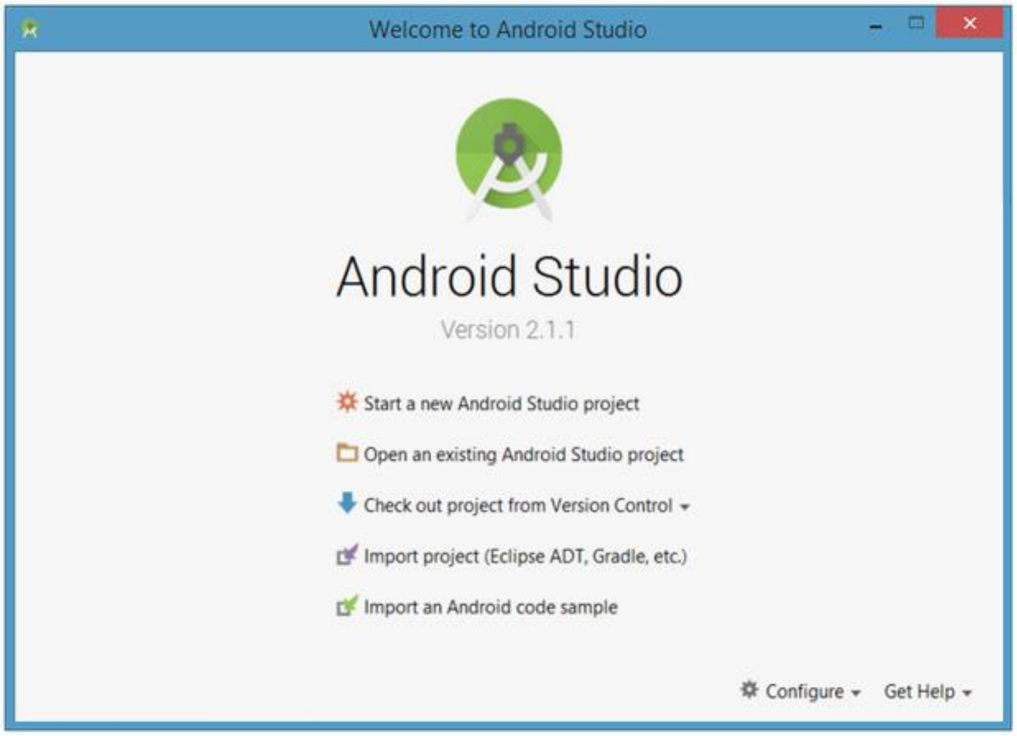
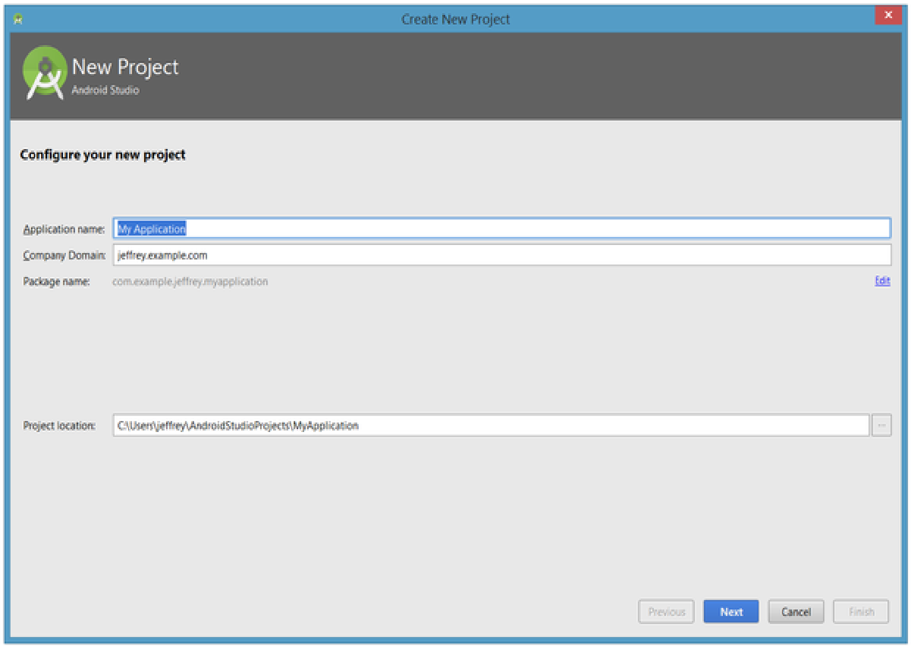

Welcome to Android Studio
Android Studio
Version 2.1.1
Start a new Android Studio project
Open an existing Android Studio project
Check out project from Version Control
Import project (Eclipse ADT, Gradle, etc.)
Import an Android code sample
Configure
Get Help

Fig.3.3.13. Welcome to Android Studio

You'll use this dialog to start up a new Android Studio project, work with an existing project, and more. You can access it anytime by double-clicking the Android Studio shortcut on your desktop.

Starting a new project

From our setup so far, you should still have Android Studio running with the Welcome to Android Studio dialog box. From here, click Start a new Android Studio project. Android Studio will respond with the Create New Project dialog box as shown below.



Configure your new project

Application name: My Application

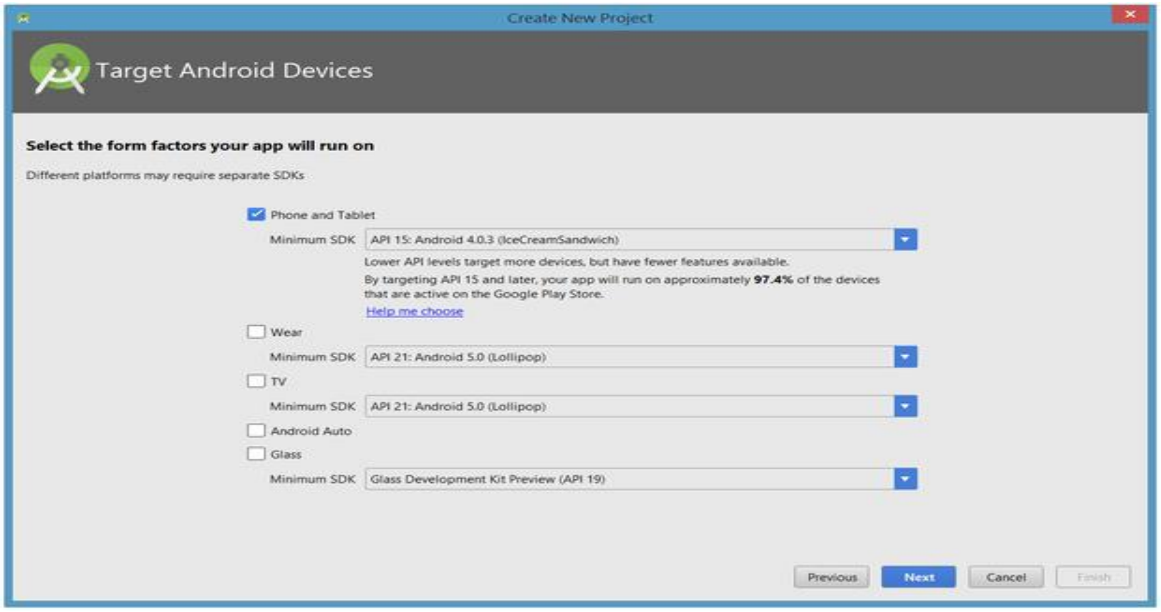
Company Domain: jeffrey.example.com

Package name: com.example.jeffrey.myapplication Edit

Project location: C:\Users\jeffrey\AndroidStudioProjects\MyApplication

Fig.3.3.14. Create a new project

Enter *W2A* (Welcome to Android) as the application name and *javajeff.ca* as the company domain name. You should then see C:\Users\jeffrey\AndroidStudioProjects\W2A as the project location. Click Next to select your target devices.



Select the form factors your app will run on

Different platforms may require separate SDKs

[✓] Phone and Tablet

Minimum SDK API 15: Android 4.0.3 (IceCreamSandwich)

Lower API levels target more devices, but have fewer features available.
By targeting API 15 and later, your app will run on approximately **97.4%** of the devices that are active on the Google Play Store.
Help me choose

[ ] Wear

Minimum SDK API 21: Android 5.0 (Lollipop)

[ ] TV

Minimum SDK API 21: Android 5.0 (Lollipop)

[ ] Android Auto

[ ] Glass

Minimum SDK Glass Development Kit Preview (API 19)

Fig.3.3.15. Select your target device categories

Android Studio lets you select form factors, or categories of target devices, for every app you create. I would have preferred to keep the default API 15: Android 4.0.3 (IceCreamSandwich) minimum SDK setting (under Phone and Tablet), which is supported by my Amazon Kindle Fire HD tablet. Because Android Studio doesn't currently support this API level (even when you add the 4.0.3 system image via the SDK Manager), I changed this setting to API 14: Android 4.0 (IceCreamSandwich), which is also supported by my tablet.

Click Next, and you will be given the opportunity to choose a template for your app's main activity. For now we'll stick with Empty Activity. Select this template and click Next.

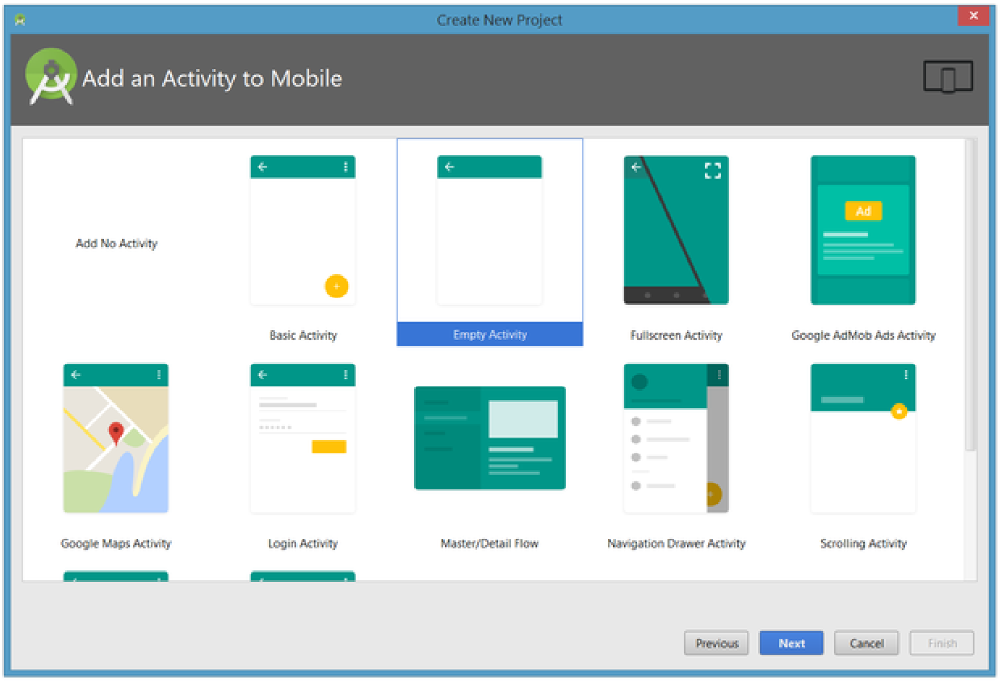
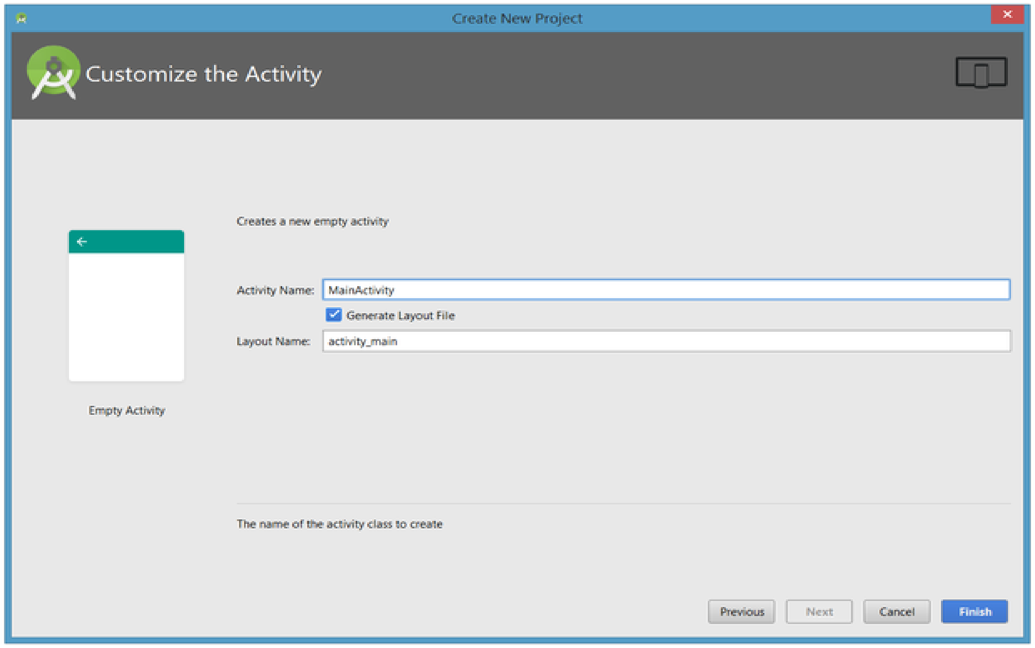


Fig.3.3.16. Specify an activity template

Next you'll customize the activity:



Creates a new empty activity

Activity Name: MainActivity

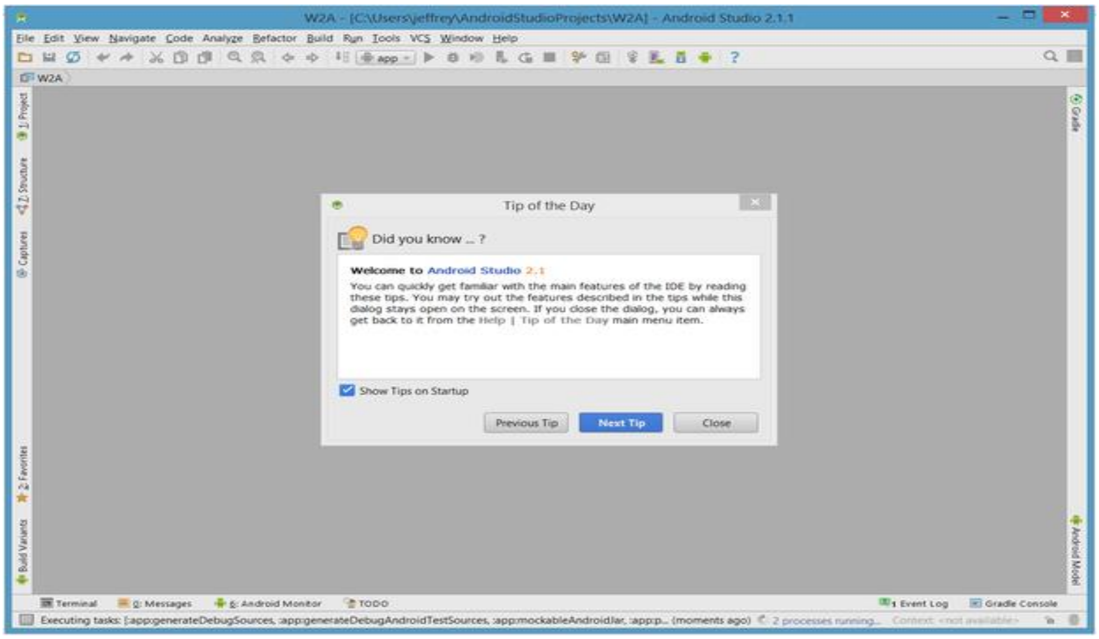
[✓] Generate Layout File

Layout Name: activity_main

The name of the activity class to create

Fig.3.3.17. Customize your activity

Enter W2A as the activity name and main as the layout name, and click Finish to complete this step. Android Studio will respond that it is creating the project, and then take you to the project workspace.



W2A - [C:\Users\jeffrey\AndroidStudioProjects\W2A] - Android Studio 2.1.1

Welcome to Android Studio 2.1

You can quickly get familiar with the main features of the IDE by reading these tips. You may try out the features described in the tips while this dialog stays open on the screen. If you close the dialog, you can always get back to it from the Help | Tip of the Day main menu item.

[✓] Show Tips on Startup

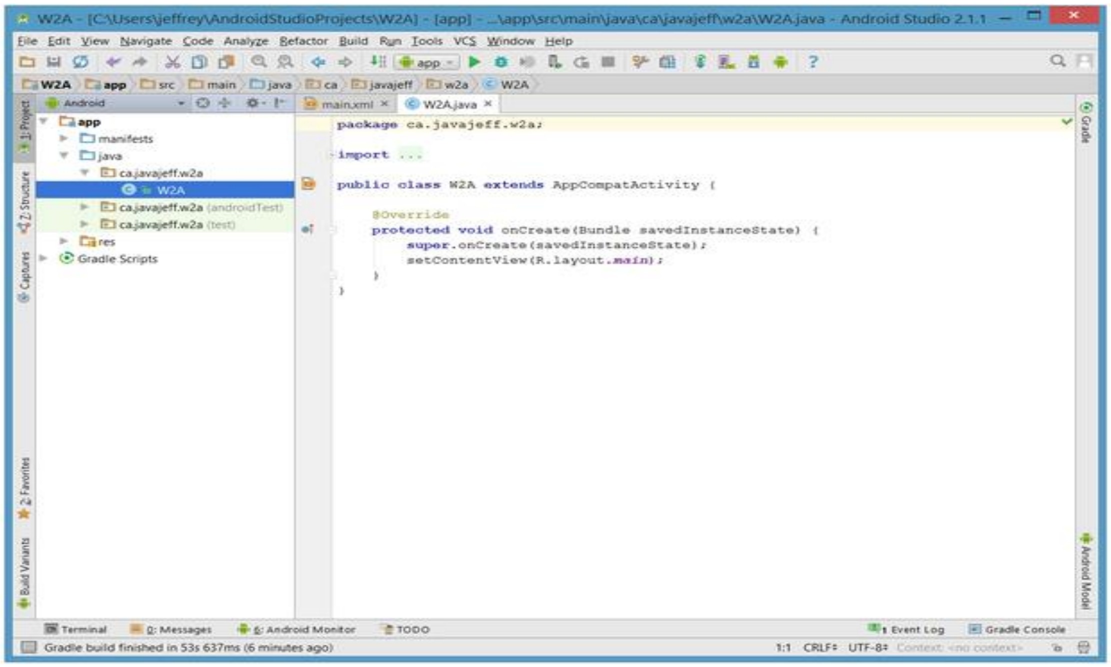
Executing tasks: [:app:generateDebugSources, :app:generateDebugAndroidTestSources, :app:mockableAndroidJar, :app:p... (moments ago) 2 processes running... Context: <not available>

Fig.3.3.18. Android Studio workspace

The project workspace is organized around a menu bar, a tool bar, a work area, additional components that lead to more windows (such as a Gradle Console window), and a status bar. Also note the Tip of the Day dialog box, which you can disable if you like.

3.3.19 The project and editor windows

When you enter the project workspace, W2A is identified as the current project, but you won't immediately see the project details. After a few moments, these details will be appearing in two new windows.


W2A - [C:\Users\jeffrey\AndroidStudioProjects\W2A] - [app] - ...\app\src\main\java\ca\javajeff\w2a\W2A.java - Android Studio 2.1.1
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
W2A app src main java ca javajeff w2a W2A
Android
app
manifests
java
ca.javajeff.w2a
W2A
ca.javajeff.w2a (androidTest)
ca.javajeff.w2a (test)
res
Gradle Scripts
main.xml
W2A.java
1: Project
7: Structure
Captures
2: Favorites
Build Variants
Gradle
Android Model
Terminal
0: Messages
6: Android Monitor
TODO
Event Log
Gradle Console
1:1 CRLF UTF-8 Context: <no context>

```
package ca.javajeff.w2a;

import ...

public class W2A extends AppCompatActivity {

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

```
Gradle build finished in 53s 637ms (6 minutes ago)
```

Fig.3.3.19 The project and editor windows

CHAPTER 4
DESIGN

4.1 Introduction

System design is the solution to the creation of a new system. This phase is composed of several systems. This phase focuses on the detailed implementation of the feasible system. It emphasis on translating design specifications to performance specification is system design. System design has two phases of development logical and physical design.

During logical design phase the analyst describes inputs (sources), outputs (destinations), databases (data stores) and procedures (data flows) all in a format that meats the uses requirements. The analyst also specifies the user needs and at a level that virtually determines the information flow into and out of the system and the data resources. Here the logical design is done through data flow diagrams and database design.

The physical design is followed by physical design or coding. Physical design produces the working system by defining the design specifications, which tell the programmers exactly what the candidate system must do.

The programmers write the necessary programs that accept input from the user, perform necessary processing on accepted data through call and produce the required report on a hard copy or display it on the screen.

4.2 SYSTEM ARCHITECTURE

4.2.1 Architectural Design:

3-Tier architecture is also called layered architecture. Some people called it n-tier architecture. Layer architectures are essentially objects and work in object oriented environment. 3-tier architecture is a very well-known architecture in the world of software development, it doesn'tmatter whether you are developing web based application or desktop based, it is the best architecture to use.

3-Tier architecture consists of

1) UI or Presentation Layer

2) Business Access Layer or Business Logic Layer

3) Data Access Layer

Presentation Layer:

Presentation layer consists of pages like .java or desktop based form where data is presented to users or getting input from users.

Business Logic layer or Business Access Layer

Business logic layer contains all of the business logic. Its responsibility is to validate the business rules of the component and communicating with the Data Access Layer.

Business Logic Layer is the class in which we write functions that get data from Presentation Layer and send that data to database through Data Access Layer.

Data Access Layer:

Data Access Layer is also the class that contains methods to enable business logic layer to connect the data and perform desired actions. These desired actions can be selecting, inserting, updating and deleting the data. DAL accepts the data from BAL and sends it to the database or DAL gets the data from the database and sends it to the business layer. In short, its responsibility is to communicate with the backend structure.

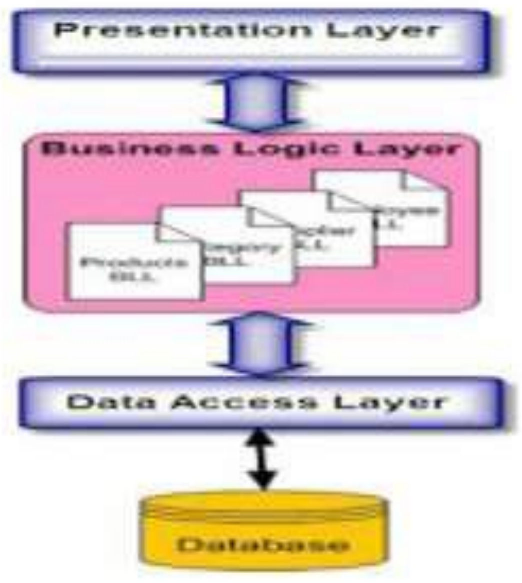


Fig 4.2.1: Illustration of3-Tier Architecture with Diagram.

4.3 UML DIAGRAMS

UML stands for Unified Modeling Language.UML is a standardized general purpose modeling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group.

The goal is for UML to become a common language for creating models of object oriented computer software. In its current form UML is comprised of two major components: a Meta- model and a notation. In the future, some form of method or process may also be added to; or associated with, UML.

The Unified Modeling Language is a standard language for specifying , visualization, constructing and documenting the artifacts of software system, as well as for business modeling and other non-software systems.

The Unified Modeling Language represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems.

The Unified Modeling Language is a very important part of developing objects oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

Goals:

The primary goals in the design of the UML are as follows:

- Provide users a ready-to-use, expressive visual modeling Language so that they can develop and exchange meaningful models.
- Provide extendibility and specialization mechanisms to extend the core concepts.
- Be independent of particular programming languages and development process.
- Provide a formal basis for understanding the modeling language.
- Encourage the growth of the Object Oriented tools market.

- Support higher level development concepts such as collaborations, frameworks, patterns and components.

4.3.1 Class Diagram:

A class diagram is a type of static structural diagram which describes the structure of a system by showing the system's classes , their attributes , operations and the relationships among objects.

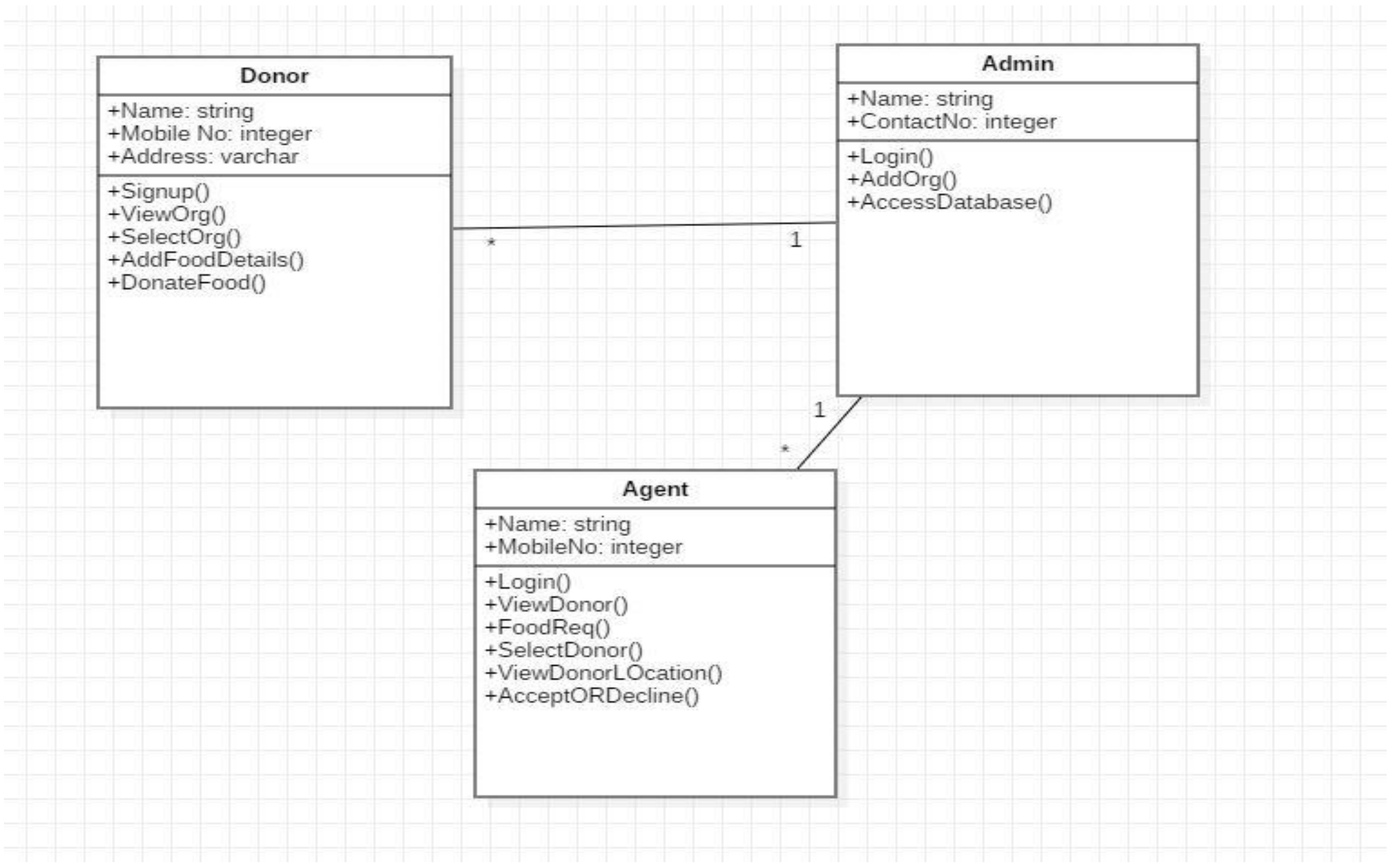


Fig Class Diagram

4.3.2 Use Case Diagram:

Use cases are used during the requirements elicitation and analysis to represent the functionality of the system . Use cases focus on the behaviour of the system from the external point of view. A use case describes a function provided by the system that yields a visible result for an actor. Use case diagrams are important in organizing and modelling the behaviour of the system. A use case is a graph of actors, a set of use cases enclosed by the system boundary, communication association between actors and the use cases, and generalization among the use cases.

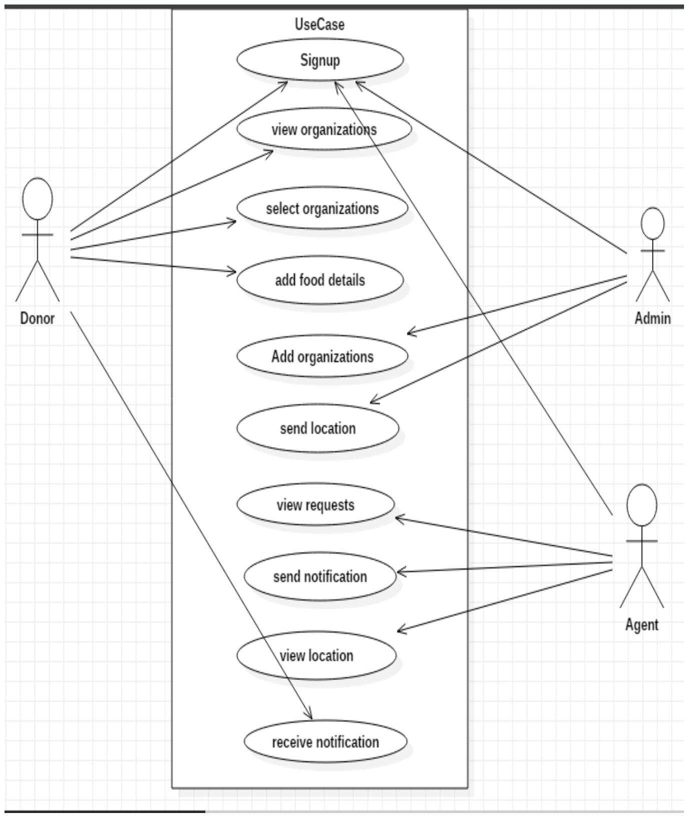


Fig Use Case Diagram

4.3.3 Sequence Diagram:

A sequence diagram is an interaction diagram that shows how processes operate with one another and in what order. It is a construct of message sequence chart. A sequence diagram shows object interactions arranged in time sequence. It has two dimensions, Vertical dimension represent time and Horizontal dimension represent different objects. The vertical line called Object Life Line that represent the objects existing during the interaction. The order in which messages appear is shown top to bottom. Each message is labelled with message name.

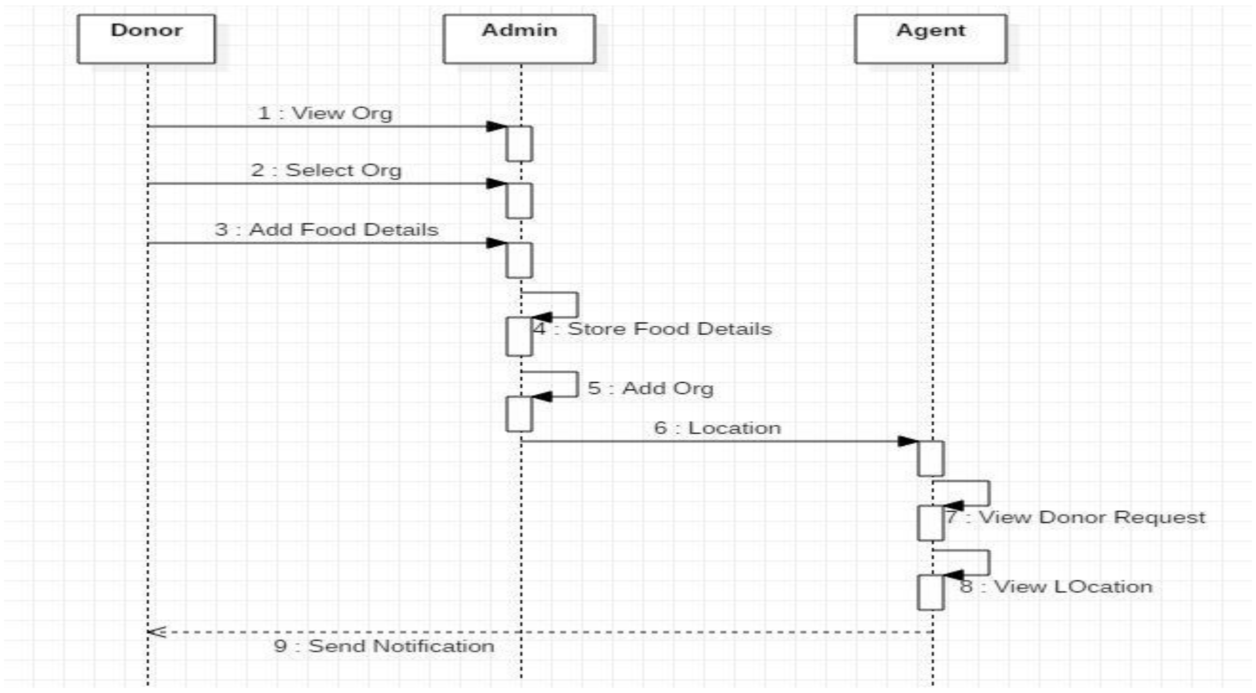


Fig Sequence Diagram

4.3.4 Activity Diagram:

Activity diagram is another important diagram in UML to describe the dynamic aspects of the system. Activity diagram is basically a flowchart to represent the flow from one activity to another activity. The activity can be described as an operation of system. The Control flow is drawn from one operation to another. This flow can be sequential, branched, or concurrent. Activity diagram deal with al type of flow control by using different elements such as fork, join, etc.

The basic Purpose of activity diagram is similar to other four diagrams. It captures the dynamic behaviour of system. Other four diagrams are used to show the message flow from one object to another but activity diagram is used to show message flow from one activity another. Activity is a particular operation of the system. Activity diagram are not only used for visualizing the dynamic nature of a system, but they are also used to construct the executable system by using forward and reverse engineering techniques. The only missing thing in the activity diagram is the message part. It does not show any message flow from one activity to another.

Activity diagram is sometimes considered as the flowchart. Although the diagram look like a flowchart, they are not. It shows different flows such as parallel, branched, concurrent, and single. The purpose of an activity diagram can be described as Draw the activity flow of a system.

- Describe the sequence from one activity to another.
- Describe the parallel, branched and concurrent flow of the system.

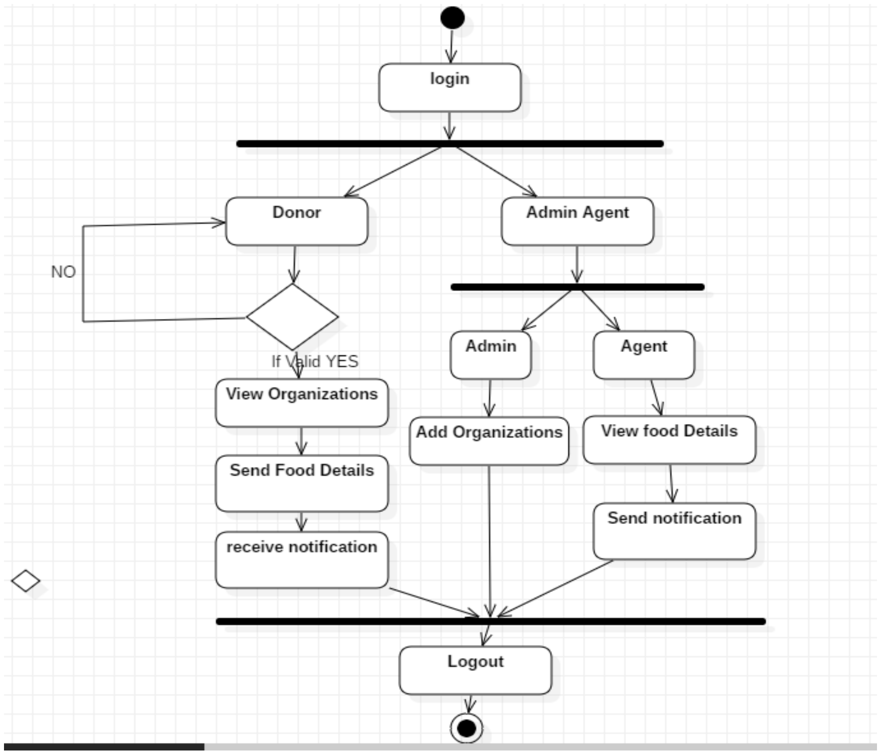


Fig Activity Diagram

4.3.5 Deployment Diagram:

Deployment diagram is a structure diagram which shows architecture of the system as deployment of software artifacts to deployment targets. Artifacts represent concrete elements in the physical world that are the results of a development process.

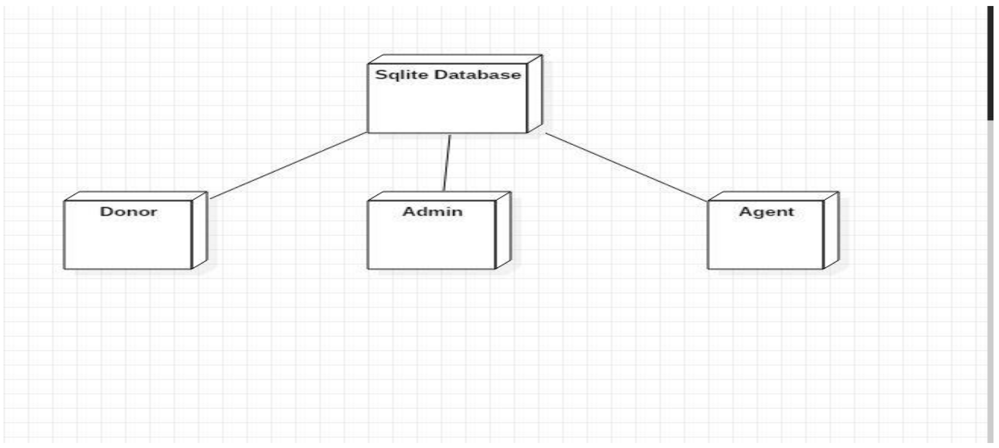


Fig Deployment Diagram

4.3.6 State diagram

A **state diagram** is used to represent the condition of the system or part of the system at finite instances of time. It’s a **behavioral** diagram and it represents the behavior using finite state transitions. State diagrams are also referred to as **State machines** and **State-chart Diagrams**. These terms are often used interchangeably. So simply, a state diagram is used to model the dynamic behavior of a class in response to time and changing external stimuli. We can say that each and every class has a state but we don’t model every class using State diagrams. We prefer to model the states with three or more states.

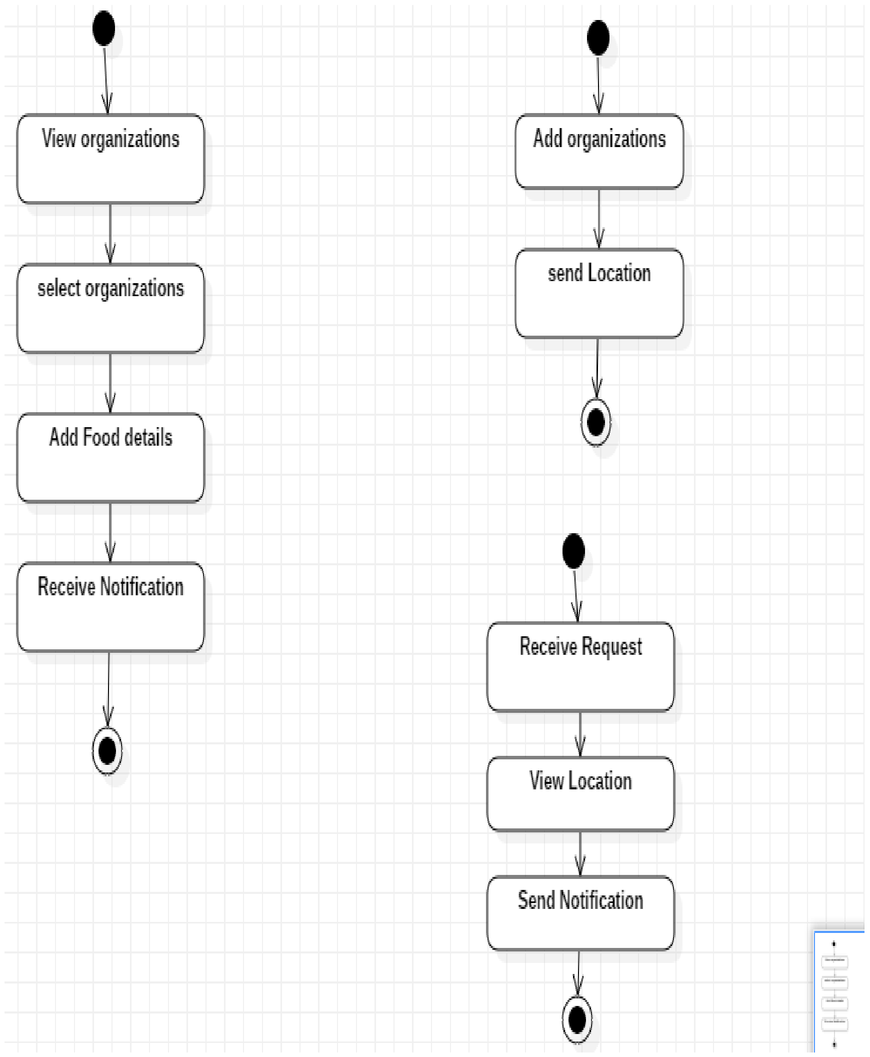


Fig State Diagram

4.3.7 Collaboration Diagram

A Collaboration is a collection of named objects and actors with links connecting them. They collaborate in performing some task.A Collaboration defines a set of participants and relationships that are meaningful for a given set of purposes.A Collaboration between objects working together provides emergent desirable functionalities in Object-Oriented systems. Each object (responsibility) partially supports emergent functionalities. Objects are able to produce (usable) high-level functionalities by working together. Objects collaborate by communicating (passing messages) with one another in order to work together.

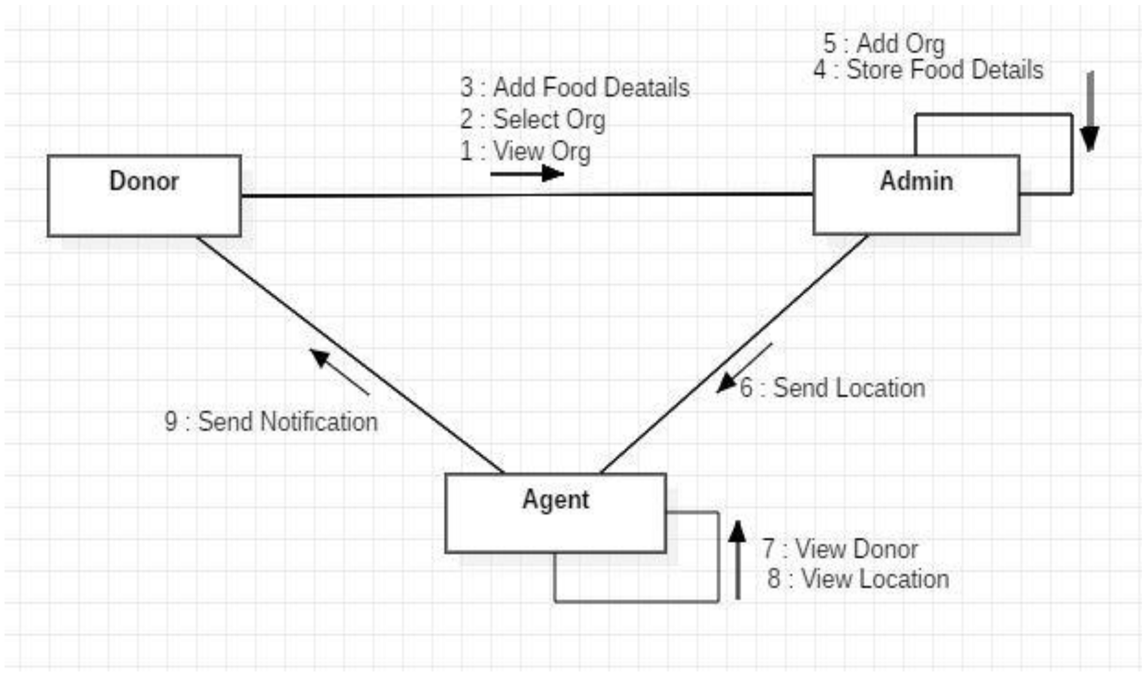


Fig Collaboration Diagram

4.3.8 Component diagram

The purpose of a component diagram is to show the relationship between different components in a**system**. For the purpose of UML 2.0, the term "component" refers to a module of classes that represent independent systems or subsystems with the ability to **interface** with the rest of the **system**.

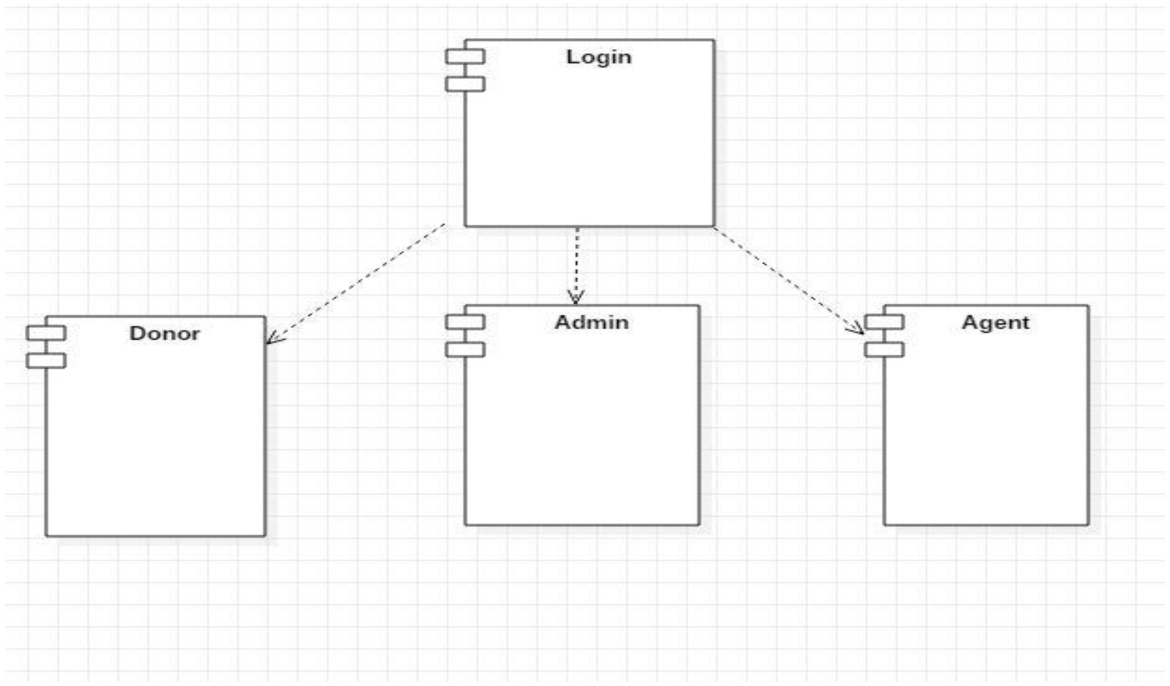


Fig Component Diagram

4.4 Conclusion

Design of this application provides clear explanation of all modules and their functionalities with their UML diagrams.

CHAPTER 5

IMPLEMENTATION

Implementation is the stage of the project when the theoretical design is turned out into working system. Thus it can be considered to be the most critical stage in achieving a successful new system and in giving the user, confidence that the new system will work and be effective.

5.1 MODULES DESCRIPTION:

5.1.1 MODULES

This project consists of four modules. They are

- Donor
- Admin
- Agent

Donor:

Here the guest is nothing but the donor who donates food. The donor can hold an account in this application & whenever there is food wastage he can login and enter the details of food and location. Details of food include amount of food available and type of food.

Admin:

The android mobile admin add the all organizations details and agent details.

Agent:

This module helps the donor to insert all the necessary details that is personal information .Select the donors and required food donors only accept the agent.

5.2 SQLITE DATABASE:

Saving data to a database is ideal for repeating or structured data, such as contact information. This page assumes that you are familiar with SQL databases in general and helps you get started with SQLite databases on Android. The APIs you'll need to use a database on Android are available in the android.database.sqlite package.

One of the main principles of SQL databases is the schema: a formal declaration of how the database is organized. The schema is reflected in the SQL statements that you use to create your database. You may find it helpful to create a companion class, known as a *contract* class, which explicitly specifies the layout of your schema in a systematic and self-documenting way.

A contract class is a container for constants that define names for URIs, tables, and columns. The contract class allows you to use the same constants across all the other classes in the same package. This lets you change a column name in one place and have it propagate throughout your code.

A good way to organize a contract class is to put definitions that are global to your whole database in the root level of the class. Then create an inner class for each table. Each inner class enumerates the corresponding table's columns.

5.2.1 Read from Database:

To read from a database, use the query() method, passing it your selection criteria and desired columns. The method combines elements of insert() and update(), except the column list defines the data you want to fetch (the "projection"), rather than the data to insert. The results of the query are returned to you in a Cursor object.

```
SQLiteDatabase db = dbHelper.getReadableDatabase();

// Define a projection that specifies which columns from the database
// you will actually use after this query.
String[] projection = {
    BaseColumns._ID,
    FeedEntry.COLUMN_NAME_TITLE,
    FeedEntry.COLUMN_NAME_SUBTITLE
    };

// Filter results WHERE "title" = 'My Title'
String selection = FeedEntry.COLUMN_NAME_TITLE + " = ?";
String[] selectionArgs = { "My Title" };

// How you want the results sorted in the resulting Cursor
String sortOrder =
    FeedEntry.COLUMN_NAME_SUBTITLE + " DESC";

Cursor cursor = db.query(
    FeedEntry.TABLE_NAME,   // The table to query
    projection,             // The array of columns to return (pass null to get all)
    selection,              // The columns for the WHERE clause
    selectionArgs,          // The values for the WHERE clause
    null,                   // don't group the rows
    null,                   // don't filter by row groups
    sortOrder               // The sort order
    );
```

The third and fourth arguments (selection and selectionArgs) are combined to create a WHERE clause. Because the arguments are provided separately from the selection query, they are escaped before being combined. This makes your selection statements immune to SQL injection. For more detail about all arguments, see the query() reference.

To look at a row in the cursor, use one of the Cursor move methods, which you must always call before you begin reading values. Since the cursor starts at position -1, calling moveToNext() places the "read position" on the first entry in the results and returns whether or not the cursor is already past the last entry in the result set. For each row, you can read a column's value by calling one of the Cursor get methods, such as getString() or getLong(). For each of the get methods, you must pass the index position of the column you desire, which you can get by callinggetColumnIndex() or getColumnIndexOrThrow(). When finished iterating through results, call close() on the cursor to release its resources. For example, the following shows how to get all the item IDs stored in a cursor and add them to a list:

```
List itemIds = new ArrayList<>();
while(cursor.moveToNext()) {
  long itemId = cursor.getLong(
      cursor.getColumnIndexOrThrow(FeedEntry._ID));
  itemIds.add(itemId);
}
cursor.close();
```

CHAPTER 6

TESTING

6.1 Software Testing

6.1.1 Testing

Software testing is a critical element of software quality assurance and represents the ultimate review of specification, design and code generation**.**

6.1.2 Testing Objectives

- To ensure that during operation the system will perform as per specification.
- TO make sure that system meets the user requirements during operation
- To make sure that during the operation, incorrect input, processing and output will be detected
- To see that when correct inputs are fed to the system the outputs are correct
- To verify that the controls incorporated in the same system as intended
- Testing is a process of executing a program with the intent of finding an error
- A good test case is one that has a high probability of finding an as yet undiscovered error

The software developed has been tested successfully using the following testing strategies and any errors that are encountered are corrected and again the part of the program or the procedure or function is put to testing until all the errors are removed. A successful test is one that uncovers an as yet undiscovered error.

Note that the result of the system testing will prove that the system is working correctly. It will give confidence to system designer, users of the system, prevent frustration during implementation process etc.,

6.2 TEST CASE DESIGN:

6.2.1 White box testing:

White box testing is a testing case design method that uses the control structure of the procedure design to derive test cases. All independents path in a module are exercised at least once, all logical decisions are exercised at once, execute all loops at boundaries and within their operational bounds exercise internal data structure to ensure their validity. Here the customer is given three chances to enter a valid choice out of the given menu. After which the control exits the current menu.

6.2.2 Black Box Testing

Black Box Testing attempts to find errors in following areas or categories, incorrect or missing functions, interface error, errors in data structures, performance error and initialization and termination error. Here all the input data must match the data type to become a valid entry.

The following are the different tests at various levels:

6.2.3 Unit Testing:

Unit testing is essentially for the verification of the code produced during the coding phase and the goal is test the internal logic of the module/program. In the Generic code project, the unit testing is done during coding phase of data entry forms whether the functions are working properly or not. In this phase all the drivers are tested they are rightly connected or not.

6.2.4 Integration Testing:

All the tested modules are combined into sub systems, which are then tested. The goal is to see if the modules are properly integrated, and the emphasis being on the testing interfaces between the modules. In the generic code integration testing is done mainly on table creation module and insertion module.

6.2.5 Validation Testing:

This testing concentrates on confirming that the software is error-free in all respects. All the specified validations are verified and the software is subjected to hard-core testing. It also aims at determining the degree of deviation that exists in the software designed from the specification; they are listed out and are corrected.

6.2.6 System Testing

This testing is a series of different tests whose primary is to fully exercise the computer-based system. This involves:

- Implementing the system in a simulated production environment and testing it.
- Introducing errors and testing for error handling.

CHAPTER 7

SOURCE CODE

7.1 XML Code

7.1.1 activity_donate_food:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout 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"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="com.example.madhav.thinkeatsave.DonateFood">


    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textAppearance="?android:attr/textAppearanceMedium"
        android:text="Donate Food"
        android:id="@+id/textView"
        android:layout_gravity="center_horizontal"
        android:layout_centerHorizontal="true"
        android:layout_alignParentTop="true" />


    <ScrollView android:layout_marginTop="30dp"
```

```
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:id="@+id/scrollView">



<LinearLayout
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" >

    <EditText
        android:id="@+id/editText4"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="55dp"
        android:ems="10"
        android:hint="Enter Name"
        android:inputType="textPersonName" />

    <EditText
        android:id="@+id/editText5"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignLeft="@+id/editText3"
        android:layout_alignStart="@+id/editText3"
        android:layout_below="@+id/editText3"
        android:layout_marginTop="28dp"
        android:ems="10"
        android:hint="Enter Email"
        android:inputType="textEmailAddress" />

    <EditText
        android:id="@+id/editText6"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignEnd="@+id/editText5"
        android:layout_alignRight="@+id/editText5"
        android:layout_below="@+id/editText5"
        android:layout_marginTop="28dp"
        android:ems="10"
        android:hint="Event Name"
        android:inputType="textPersonName" />




    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textAppearance="?android:attr/textAppearanceMedium"
        android:text="Type Of Food"
        android:id="@+id/textView1"
        android:layout_gravity="center_horizontal"
        android:layout_centerHorizontal="true"
        android:layout_alignParentTop="true" />
    <CheckBox
        android:id="@+id/checkBox1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/editText9"
        android:layout_marginTop="13dp"
        android:layout_toEndOf="@+id/textView3"
        android:layout_toRightOf="@+id/textView3"
        android:text="Veg" />
    <CheckBox
        android:id="@+id/checkBox2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/editText9"
        android:layout_marginTop="13dp"
        android:layout_toEndOf="@+id/textView3"
        android:layout_toRightOf="@+id/textView3"
        android:text="Nonveg" />

    <CheckBox
        android:id="@+id/checkBox3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/editText9"
        android:layout_marginTop="13dp"
        android:layout_toEndOf="@+id/textView3"
        android:layout_toRightOf="@+id/textView3"
        android:text="BreakFast" />
    <CheckBox
        android:id="@+id/checkBox4"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/editText9"
        android:layout_marginTop="13dp"
        android:layout_toEndOf="@+id/textView3"
        android:layout_toRightOf="@+id/textView3"
        android:text="Lunch" />

    <CheckBox
        android:id="@+id/checkBox5"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/editText9"
        android:layout_marginTop="13dp"
        android:layout_toEndOf="@+id/textView3"
        android:layout_toRightOf="@+id/textView3"
        android:text="Dinner" />
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textAppearance="?android:attr/textAppearanceMedium"
        android:text="Time Of Expire"
        android:id="@+id/textView2"
        android:layout_gravity="center_horizontal"
        android:layout_centerHorizontal="true"
        android:layout_alignParentTop="true" />

    <EditText
        android:id="@+id/editText7"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignEnd="@+id/editText6"
        android:layout_alignRight="@+id/editText6"
        android:layout_below="@+id/editText6"
```

```
        android:layout_marginTop="21dp"
        android:ems="10"
        android:hint="Time Of Expire"
        android:inputType="textPersonName" />

    <EditText
        android:id="@+id/editText8"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignEnd="@+id/editText6"
        android:layout_alignRight="@+id/editText6"
        android:layout_below="@+id/editText6"
        android:layout_marginTop="21dp"
        android:ems="10"
        android:hint="No Of Plates"
        android:inputType="number" />

    <EditText
        android:id="@+id/editText9"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignEnd="@+id/editText6"
        android:layout_alignRight="@+id/editText6"
        android:layout_below="@+id/editText6"
        android:layout_marginTop="21dp"
        android:ems="10"
        android:hint="Contact No"
        android:inputType="phone" />

    <EditText
```

```
            android:id="@+id/editText10"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_alignEnd="@+id/editText6"
            android:layout_alignRight="@+id/editText6"
            android:layout_below="@+id/editText6"
            android:layout_marginTop="21dp"
            android:ems="10"
            android:hint="Address"
            android:inputType="text" />




          <Button
            android:id="@+id/button10"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:text="Donate Food" />

        </LinearLayout>

    </ScrollView>

</RelativeLayout>
```

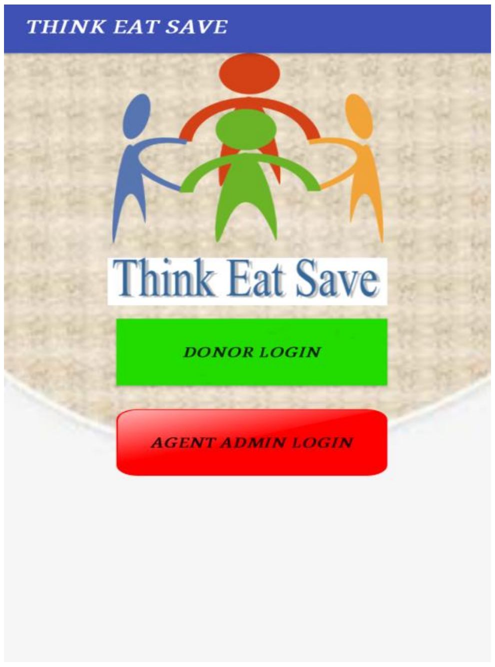
Java Code:

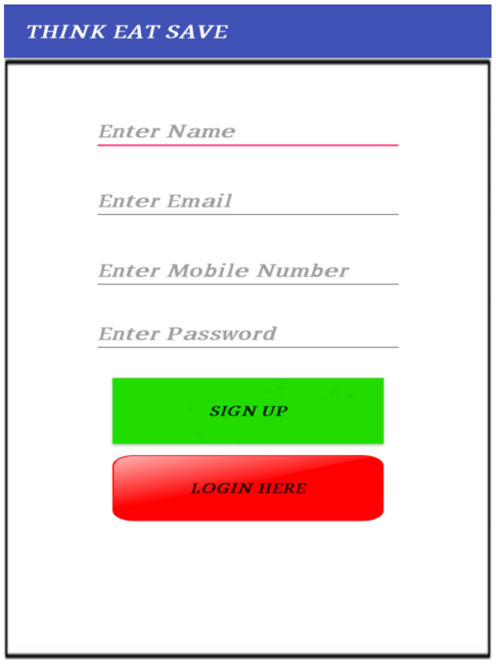
```
package com.example.madhav.thinkeatsave;


import android.content.Intent;
import android.database.sqlite.SQLiteDatabase;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.CheckBox;
import android.widget.EditText;
import android.widget.Toast;


public class DonateFood extends AppCompatActivity {
SQLiteDatabase db;
String agentname,ownername;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_donate_food);
        Intent it=getIntent();
         agentname=it.getStringExtra("aname");
         ownername=it.getStringExtra("oname");


        final EditText et1=(EditText)findViewById(R.id.editText4);
        final EditText et2=(EditText)findViewById(R.id.editText5);
        final EditText et3=(EditText)findViewById(R.id.editText6);
        final EditText et4=(EditText)findViewById(R.id.editText7);
        final EditText et5=(EditText)findViewById(R.id.editText8);
        final EditText et6=(EditText)findViewById(R.id.editText9);
        final EditText et7=(EditText)findViewById(R.id.editText10);
        final CheckBox c1=(CheckBox)findViewById(R.id.checkBox1);
        final CheckBox c2=(CheckBox)findViewById(R.id.checkBox2);
        final CheckBox c3=(CheckBox)findViewById(R.id.checkBox3);
        final CheckBox c4=(CheckBox)findViewById(R.id.checkBox4);
        final CheckBox c5=(CheckBox)findViewById(R.id.checkBox5);

        final String emailPattern = "[a-zA-Z0-9._-]+@[a-z]+\\.+[a-z]+";
        Button bt=(Button)findViewById(R.id.button10);
        bt.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                String name=et1.getText().toString();
                String mailid=et2.getText().toString();
                String eventname=et3.getText().toString();
                String fexpire=et4.getText().toString();
                String nofoplates=et5.getText().toString();
                String mobile=et6.getText().toString();
                String adr=et7.getText().toString();

                StringBuilder result=new StringBuilder();
                result.append("Selected Items:");
                if(c1.isChecked()){
                    result.append("\n veg");

                }
                if(c2.isChecked()){
                    result.append("\n NonVeg");
                }
                if(c3.isChecked()){
                    result.append("\n BreakFast");
                }
                if(c4.isChecked()){
                    result.append("\n Lunch");
                }
                if(c5.isChecked()){
                    result.append("\n Dinner");
                }
                String food=result.toString();

                if(name.trim().length()>0) {

                    if (mailid.matches(emailPattern)) {
                        if (eventname.trim().length() > 0) {
                            if (fexpire.trim().length() > 0) {
                                if (nofoplates.trim().length() > 0) {


                                    if ((mobile.trim().length() > 0) & (mobile.length() >= 10)) {
                                        if (adr.trim().length() > 0) {



                                            db = openOrCreateDatabase("tes", MODE_PRIVATE, null);
                                            db.execSQL("create table if not exists donatefood(dname varchar,mailid varchar,ename varchar,typeoffood varchar,expire varchar,plates varchar,mobile varchar,addr varchar,status varchar,aname varchar)");
```

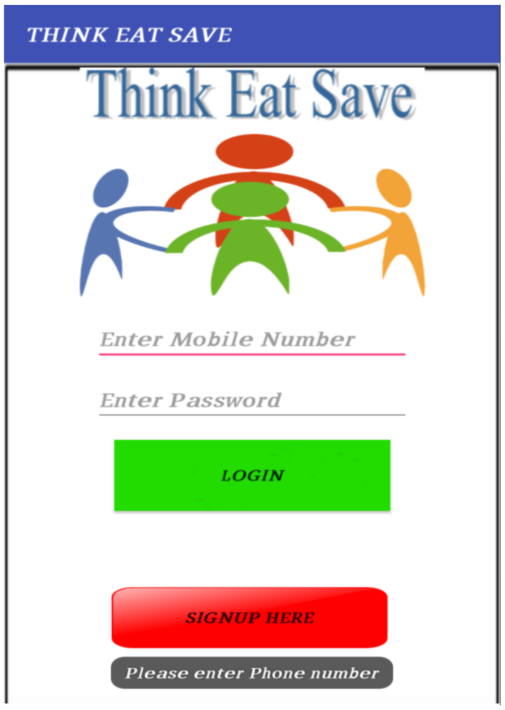
```
                                        db.execSQL("insert into donatefood values('" + name + "','" + mailid
+ "','" + eventname + "','" + food + "','" + fexpire + "','" + nofoplates + "','" + mobile + "','" + adr
+ "','no','" + agentname + "')");
                                        Toast.makeText(getApplicationContext(),        "Food        Donated
Successfuly", Toast.LENGTH_LONG).show();
                                       Intent it=new Intent(DonateFood.this,Donor.class);
                                       startActivity(it);
                                    } else {
                                        Toast.makeText(getApplicationContext(),   "Please   enter   address",
Toast.LENGTH_LONG).show();
                                    }
                                } else {
                                    Toast.makeText(getApplicationContext(),      "Please      enter      mobile
number", Toast.LENGTH_LONG).show();
                                }
                            } else {
                                Toast.makeText(getApplicationContext(), "Please enter number of plates",
Toast.LENGTH_LONG).show();
                            }
                            } else {
                                Toast.makeText(getApplicationContext(), "Please enter food expire time",
Toast.LENGTH_LONG).show();
                            }
                        } else {
                            Toast.makeText(getApplicationContext(),    "Please    enter    event    name",
Toast.LENGTH_LONG).show();
                        }
                    } else {
                        Toast.makeText(getApplicationContext(),        "Please        enter        mailid",
Toast.LENGTH_LONG).show();
```

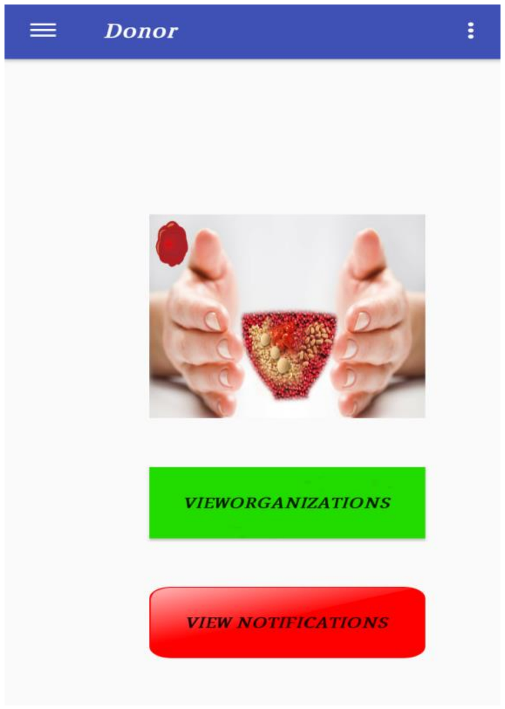
```
                    }
                } else {
                    Toast.makeText(getApplicationContext(), "Please enter donor name",
Toast.LENGTH_LONG).show();
                }

            }

        });


    }
}
```

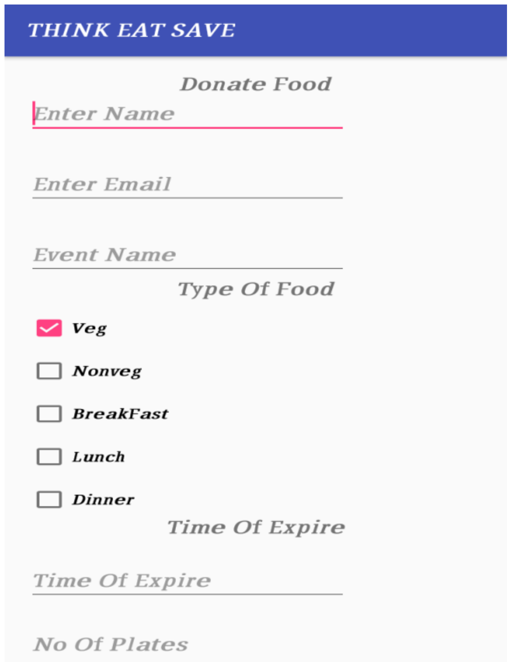
RESULT

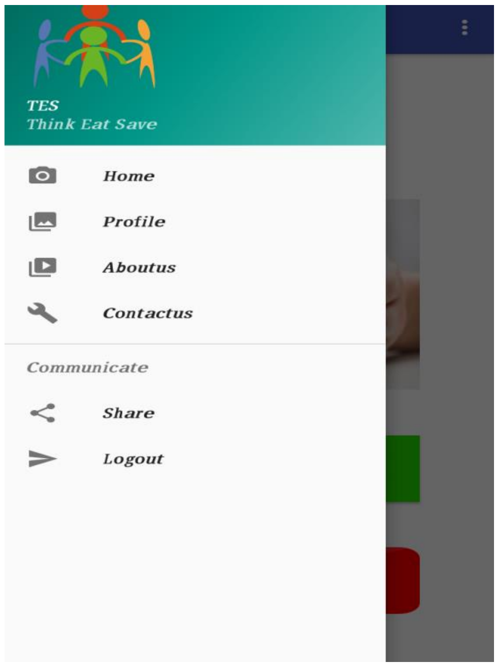

THINK EAT SAVE
Think Eat Save
DONOR LOGIN
AGENT ADMIN LOGIN

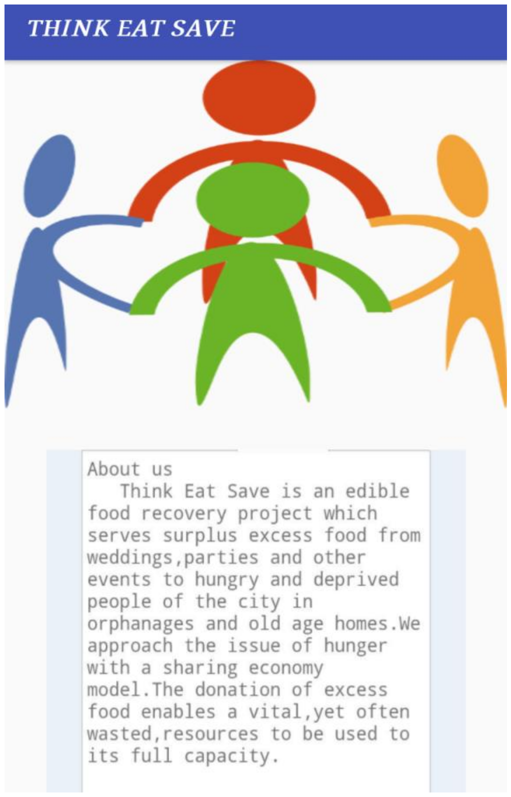

THINK EAT SAVE
Enter Name
Enter Email
Enter Mobile Number
Enter Password
SIGN UP
LOGIN HERE


THINK EAT SAVE
Think Eat Save
Enter Mobile Number
Enter Password
LOGIN
SIGNUP HERE
Please enter Phone number

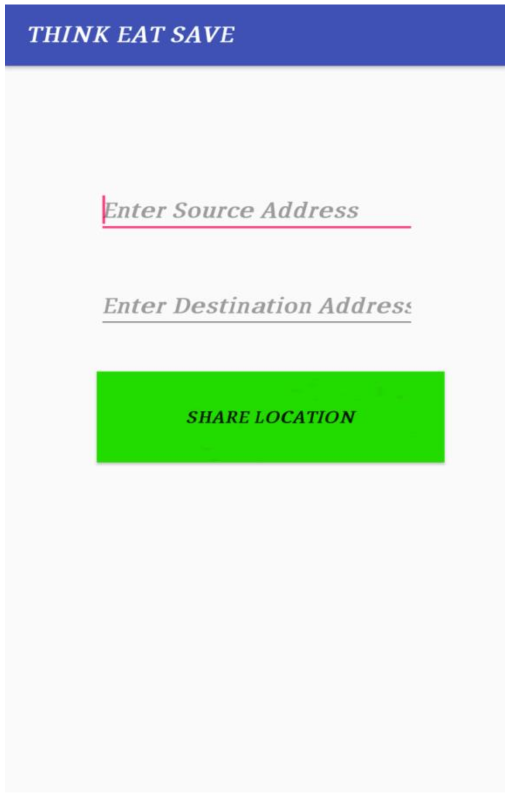

Donor
VIEWORGANIZATIONS
VIEW NOTIFICATIONS

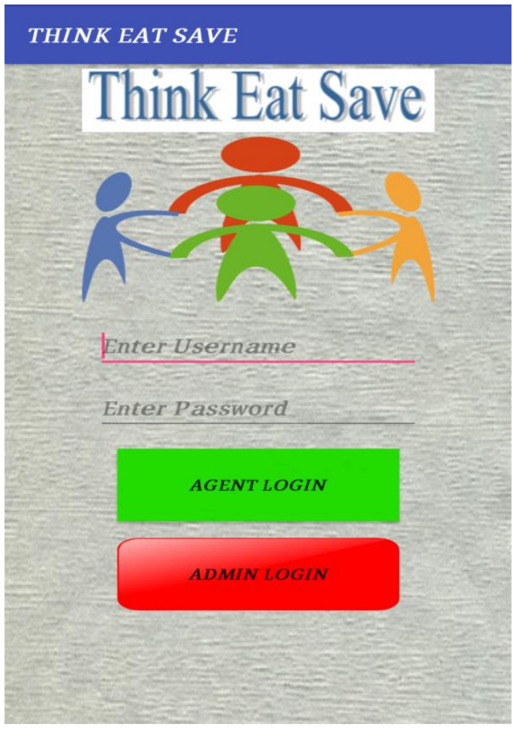

THINK EAT SAVE
Donate Food
Enter Name
Enter Email
Event Name
Type Of Food
Veg
Nonveg
BreakFast
Lunch
Dinner
Time Of Expire
Time Of Expire
No Of Plates

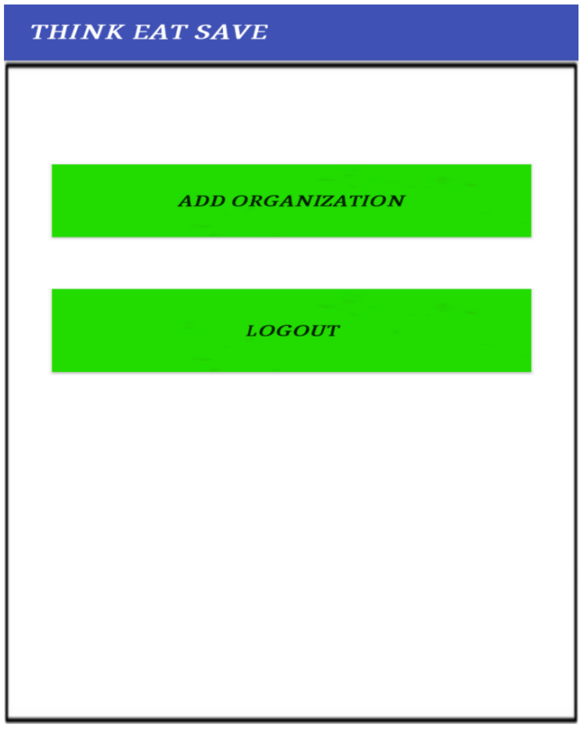

TES
Think Eat Save
Home
Profile
Aboutus
Contactus
Communicate
Share
Logout

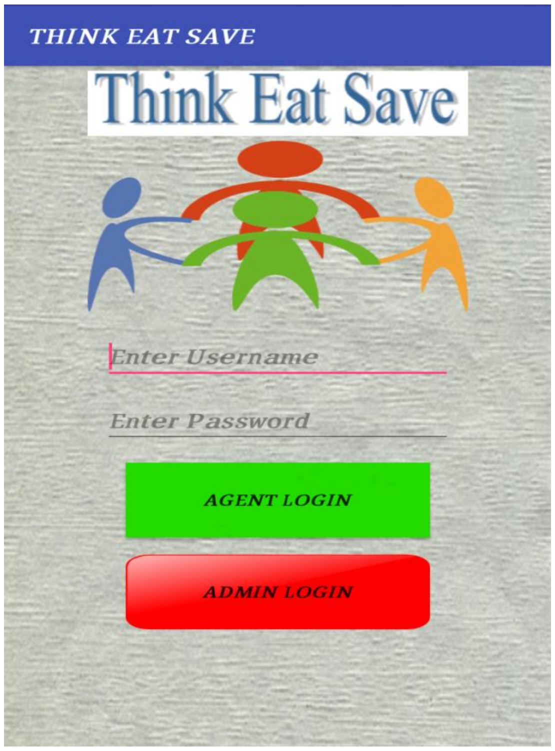


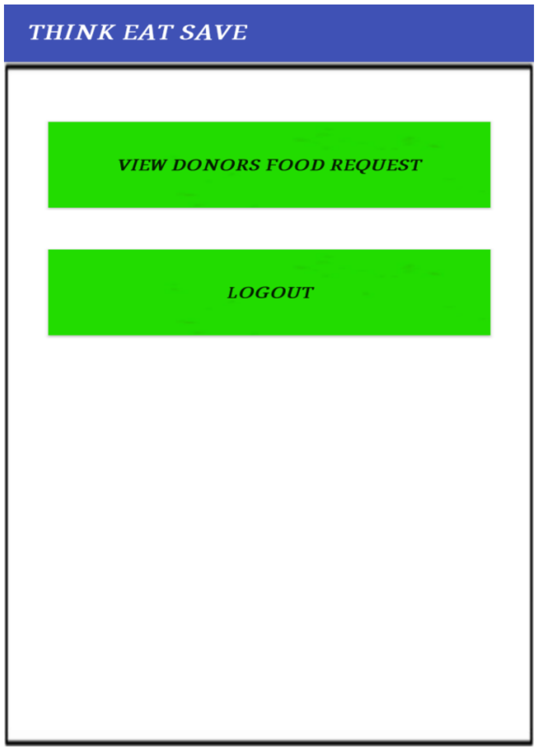
About us
Think Eat Save is an edible food recovery project which serves surplus excess food from weddings,parties and other events to hungry and deprived people of the city in orphanages and old age homes.We approach the issue of hunger with a sharing economy model.The donation of excess food enables a vital,yet often wasted,resources to be used to its full capacity.

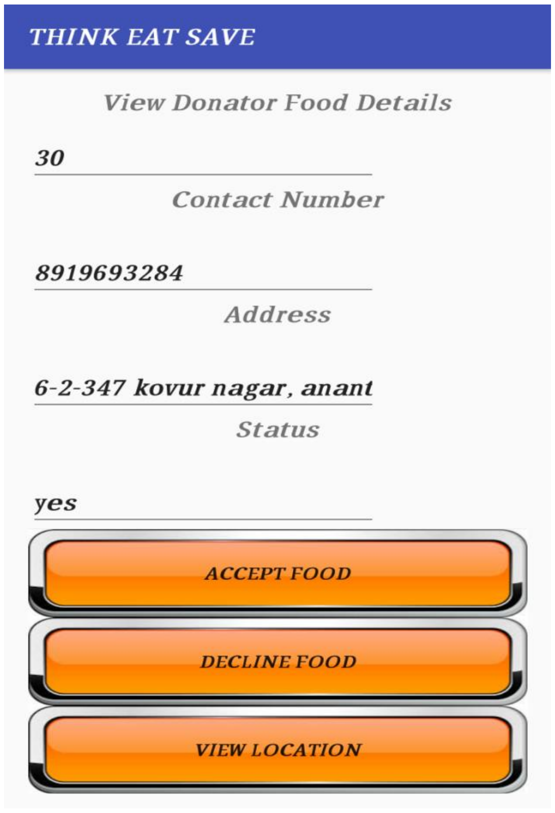












View Donator Food Details

30

Contact Number

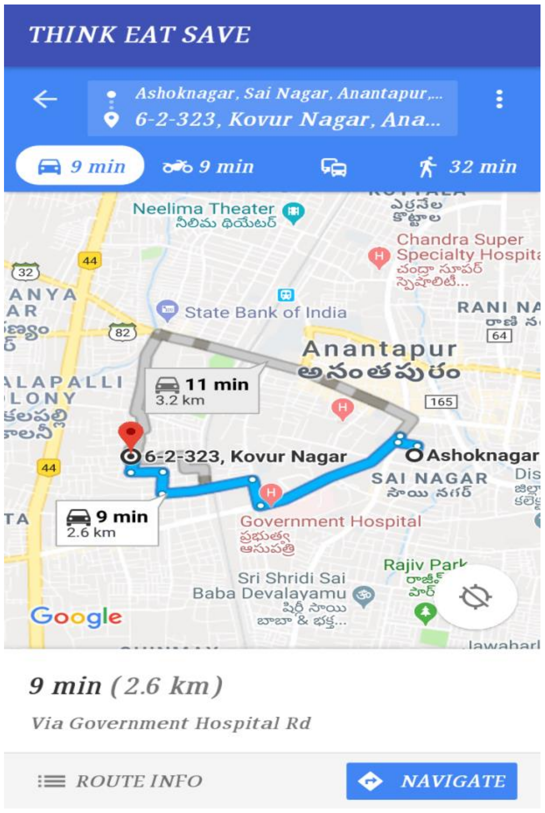
8919693284

Address

6-2-347 kovur nagar, anant

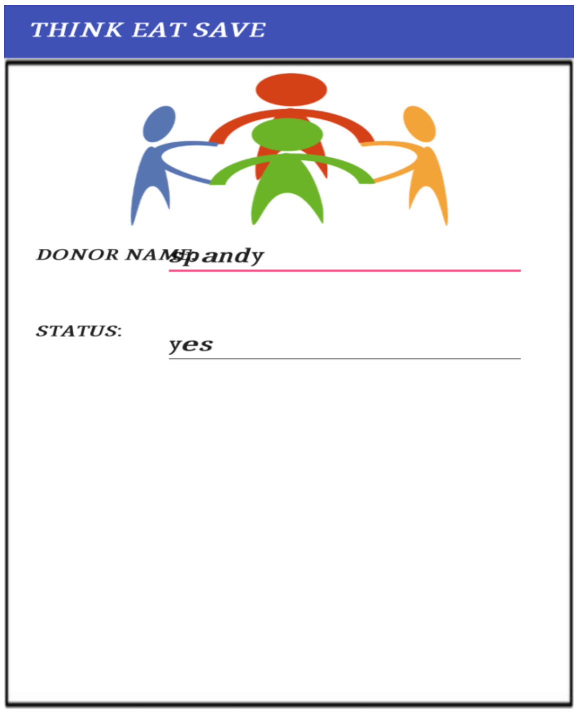
Status

yes



9 min (2.6 km)

Via Government Hospital Rd



THINK EAT SAVE

DONOR NAME	spandy
STATUS:	yes

CONCLUSION

We all know food is a primary necessity of life. There are lots of scenarios where immediate availability of food can save human lives. Our project makes one step in this direction. This project helps the people who are in need of a food by giving them all details of food availability or regarding the donors.

REFERENCES

I. Bagherzadeh, Morvarid, Mitsuhiro Inamura, and Hyunchul Jeong. "Food waste along the food chain." (2014).

II. www.https://thecsrjournal.in/food-wastage-in-india-a-serious-concern/

III. https://www.epa.gov/recycle/reducing-wasted-food-

IV. https://developer.android.com/training/data-storage/sqlite#java