**STEGANOGRAPHY TOOL**

****

Report Submitted towards partial fulfillment

Of the degree of **Bachelor of Technology**

YEAR 2019/20

Department of Computer Science & Engineering

Guided By: Submitted By:

1. Mr. Chetan Chouhan 1.Shivanshi Paliwal (16010BTCS00374)

2. Mr. Anand S Rajavat 2. Saloni Trivedi (16010BTCS00393)

3. Aradhya Jain (16010BTCS00398)

4. Chinmay Rangnekar (16010DMTCS00832)

**SHRI VAISHNAV INSTITUTE OF TECHNOLOGY & SCIENCE, INDORE**

**CERTIFICATE**

This is to certify that Ms. Shivanshi Paliwal, Ms. Saloni Trivedi, Mr.Aradhya Jain, and Mr. Chinmay Rangnekar, working in a group have satisfactorily completed the minor project titled “**Steganography Tools**” towards the partial fulfillment of the degree in Bachelor of Technology(Computer Science Engineering) Awarded by Shri Vaishnav Vidyapeeth Vishwavidyalaya, Indore for the academic year 2019-20.

Project Guide Dr. Anand Rajawat

Head of Department

Internal External

# ACKNOWLEDGEMENT

It has been great honour and privilege to complete our Minor project on Steganography Tools the guidance of Mr. Chetan Chouhan and Mr. Anand Singh Rajavat.

We are very much thankful to Dr.Anand Rajawat(Head of the Department, CSE) for providing all facilities and support to meet our project requirements.

We would like to take this opportunity to express our humble gratitude towards under whom we executed this project. Their constant guidance and willingness to share their vast knowledge made us understand this project and its manifestations in great depths and helped us complete its assigned tasks.

We are thankful to our project internal guide, whose invaluable guidance helped us understand this project better. Although there may be many who remain unacknowledged in this humble note of gratitude, there are none who remain unappreciated.

Shivanshi Paliwal

Saloni Trivedi

Aradhya Jain

Chinmay Rangnekar

Contents

1.[INTRODUCTION: 6](#_Toc448873183)

[1.1 PROBLEM STATEMENT: 7](#_Toc448873184)

[1.2 OBJECTIVE: 8](#_Toc448873185)

[1.3 SCOPE: 8](#_Toc448873186)

[1.4 PLATFORM SPECIFICATION: 8](#_Toc448873187)

[**2.** **SYSTEM ANALYSIS:** 10](#_Toc448873188)

[2.1 IDENTIFICATION OF NEED: 10](#_Toc448873189)

[2.2 PRELIMINARY INVESTIGATION: 10](#_Toc448873190)

[**3.** **FEASIBILITY STUDY:** 13](#_Toc448873191)

[3.1 TECHNICAL FEASIBILITY: 13](#_Toc448873192)

[3.2 ECONOMICAL FEASIBILITY: 13](#_Toc448873193)

[3.3 OPERATIONAL FEASIBILITY: 13](#_Toc448873194)

[**4.** **LITRATURE SURVEY:** 14](#_Toc448873195)

[4.1 WORK DONE BY OTHERS: 14](#_Toc448873196)

[4.2 BENEFITS: 14](#_Toc448873197)

[4.3 PROPOSED SOLUTION: 15](#_Toc448873198)

[4.4 TECHNOLOGY USED: 15](#_Toc448873199)

[**5.** **TECHNICAL PART**: 22](#_Toc448873200)

[**6.** **SOFTWARE ENGINEERING APPROACH:** 22](#_Toc448873201)

[6.1 SOFTWARE ENGINEERING PARADIGM APPLIED: 22](#_Toc448873202)

[6.2 REQUIREMENT ANALYSIS: 24](#_Toc448873204)

[6.3 PLANNING MANAGERIAL ISSUES 30](#_Toc448873212)

6.4[DESIGN 38](#_Toc448873221)

6.5 IMPLEMENTATION PHASE 39

6.6.[**TESTING:** 48](#_Toc448873231)

**7**.[**CONCLUSION:** 52](#_Toc448873235)

**8.BIBLIOGRAPHY &REFERENCES:**  53

**CHAPTER 1**

**INTRODUCTION**

**Smart Traffic Light System**

# INTRODUCTION:

## 1.1 PROBLEM STATEMENT:

Data is an important asset for any individual or organization and must be protected from intruders or hackers. Globalization has led to the rapid growth of the internet through which consumers can send and receive large amounts of data (e.g., text, audio, video, and images). In modern communication systems, securing data is of utmost importance. Yet sending and receiving secret files over the internet is still insecure, and therefore hiding data in an effective way protects this secret information.

## 1.2 OBJECTIVE:

The need to hide data from hackers has existed since ancient times, and nowadays, there are developments in digital media, such as audio, video, images, and so on. To secure secret information, different media methods are used and steganography is one. Steganography hides the data under other data without any differentiable changes. Many individual steganography tools can be used to transfer data securely and, in this paper, a new tool is proposed that decreases time and effort. Using this tool, we hide the text in audio, video, or images in one place, so there was no need to have access to multiple tools.

## 1.3 SCOPE:

By our application, all the emergency services will be able to cross the traffic signal without waiting for the traffic congestion to get released. The driver of the emergency vehicle would register and login to our application. Then after setting up the route, he will get all the signals green from where he passes. Thus, he could reach his destination without any extra traffic delay.

## 1.4 PLATFORM SPECIFICATION:

## HARDWARE REQUIREMENT:

* ESP8266-01
* IC 4017
* IC 555 timer
* IC 7408
* IC 7432

## SOFTWARE REQUIREMENT:

* Arduino
* Database-MYSQL
* JAVA

## IMPLEMENTATION LANGUAGE:

* HTML
* CSS
* JQuery
* Angular JS
* Android

**CHAPTER 2**

**SYSTEM ANALYSIS**

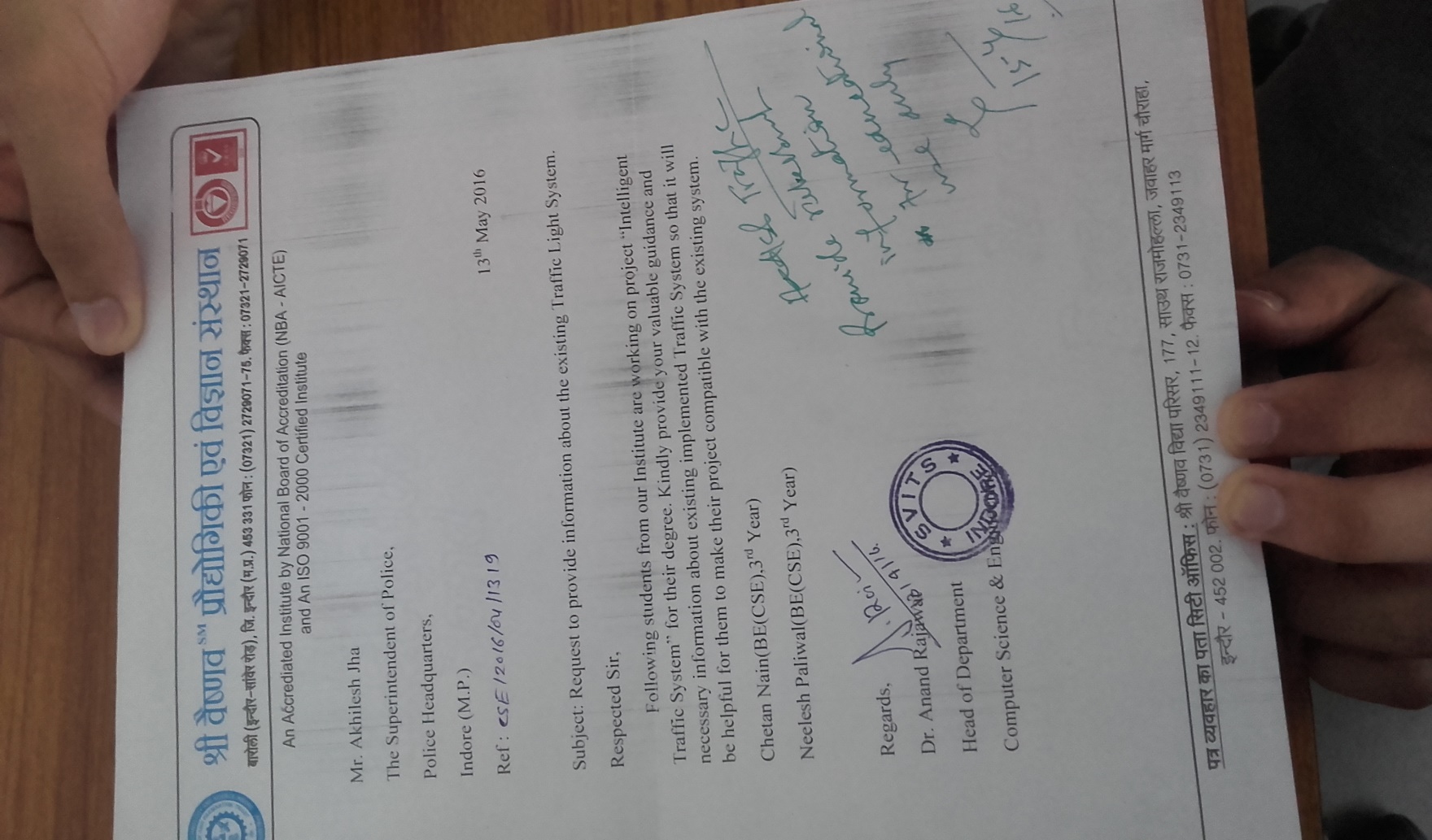
# SYSTEM ANALYSIS:

## 2.1 IDENTIFICATION OF NEED:

Traffic in India is one of the major issues in the country. Despite of having fine roads and multilanes, traffic is always at its peak. From a bicycle to a truck, every person is busy and is in a hurry to go to his destination. In this scenario, the major problem is faced by the emergency services of the country that is ambulance, fire brigade, etc. They are responsible for saving people’s life and require to reach to their place in minimum time. But due to traffic congestion they face on roads, they are unable to reach on time due to which we may suffer lives loss.

## 2.2 PRELIMINARY INVESTIGATION:

Investigation has revealed that the current traffic system is quite static and it is required to be dynamic .When searched for techniques to make the system dynamic it was found that there are many ways but the best out of it was to use a wifi module microcontroller.



**CHAPTER 3**

**FEASIBILITY STUDY**

# FEASIBILITY STUDY:

The feasibility study is a major factor which contributes to the analysis and development of the system. The decision of the system analyst whether to design a particular system or not depends on its feasibility study.

Study of requirement analysis is done through different feasibility study. Feasibility study is undertaken whenever a possibility of probability of improving the existing system or designing new system. Feasibility study helps to meet user requirements.

It enables us to determine the potential of existing system and improving it. It helps to develop a technically and economically feasible system. It helps to know what should be embedded in the system. It also helps to develop a cost-effective system. We can make better utilization of available resources.

The project concept is feasible because of the following:

3.1 Technical Feasibility

3.2 Economical Feasibility

3.3 Operational Feasibility

## 3.1 TECHNICAL FEASIBILITY:

It is a measure of the how practical solutions are and whether the technology is already available within the organization. If the technology is not available to the firm, technical feasibility also looks at whether it can be acquired.

Technical feasibility centers around the existing system and to what extent its support can be extended to the proposed system. This project is technically feasible and to maximum extent the existing systems support the proposed system.

## **3.2 ECONOMICAL FEASIBILITY**:

It is a measure of the cost-effective of a project or solutions. It is a measure of whether a solution will pay for itself or how profitable a solution will be, this is often called a cost-benefit analysis.

## 3.3 OPERATIONAL FEASIBILITY:

It is a measure of how well the system will work in the organization. It is also a measure of how people feel about the system/project. In this project the user feels that the system is very user friendly. This project developed is worth and solutions to the problem will work successfully.

# 

**CHAPTER 4**

**LITERATURE SURVEY**

# LITRATURE SURVEY:

## 4.1 WORK DONE BY OTHERS:

**Literature aspects:**

1. **RFID Technology**

Radio frequency identification (RFID) is a generic term that is used to describe a system that transmits the identity (in the form of a unique serial number) of an object or person wirelessly, using radio waves. The waves coming from the RFID reader are not dangerous and are similar to those waves coming from car radio. RFID is an automated data-capture technology that can be used to identify, track, and store information contained on a tag. A radio frequency reader scans the tag for data and sends the information to a database, which stores the data contained on the tag. A complete RFID system consists of a transponder (tag), reader/writer, antenna, and database. RFID Tag: An RFID tag, or transponder, consists of a chip and an antenna. A chip can store a unique serial number or other information based on the tag’s type of memory, which can be read-only, read-write, or write once read-many (WORM). The antenna, which is attached to the microchip, transmits information from the chip to the reader. These tags are classified as either active or passive tags. Active tags have internal batteries that allow a longer reading range, while Passive tags are powered by the signal from its reader and thus have shorter reading range. RFID Reader: In order for an RFID system to function, it needs a reader. RFID reader is a scanning device that is capable of reliably reading the tags and communicating the results to a database. A reader uses its own antenna to communicate with the tag. When a reader broadcasts radio waves, all tags designated to that frequency and within range will respond. A reader also has the capability to communicate with the tag without a direct line of sight, depending on the radio frequency and the type of tag used.Whenever any person buys a vehicle, one first needs to get his or her vehicle registered at the RTO office. RTO officials will not only assign a number plate to it but also will give a RFID enabled tag. This card will have a unique ID feasible to use with that vehicle only. They will also create an account for the use of that particular smart card and maintain transaction history in database. User needs to deposit some minimum amount to this account. Every time a registered vehicle approaches the toll booth, first the Infrared sensors will detect the presence of the vehicle. It will in turn activate the RFID circuit to read the RFID enable tag fixed on the rear-windshield of the vehicle. Transaction will begin; depending upon the balance available toll will be deducted directly from the user’s account. The software further updates the details in the Centralized database server. It also triggers mechanism to generate the bill and will be sent to user as a text message.

1. **Barcode Technology**

A barcode is a series of parallel black bars and white spaces, both of varying widths. The barcode simply provides a reference number that tells a computer to access information. A barcode reader is required to read a barcode. Barcode readers may be fixed, portable batch, or portable RF. Fixed readers are attached to a host computer and terminal, and transmit one item at a time as the data is scanned. Barcodes are simple to use, accurate, and quick. Barcode Tag can be installed on the front number plate of vehicle. This system depends on four devices. ¬ Tag Barcode ¬BarcodeReader : Laser ¬ Traffic controller system ¬ Central Server Traffic controller system is computer system which manages the traffic in single row or line by using traffic signals. The Central server stores the data which comes from different toll plaza. A local computer of every toll plaza is connected to central server through Internet. Every barcode contains the following details of owner RC (Registered Challan) which includes vehicle number, vehicle type, owners name, owners address, date of purchase, Account number and Mobile number. Working of Barcode Technology When the vehicle enters the toll plaza, the embedded laser in the barrier reads the barcode and it retrieves information and deducts the toll amount from owners account. If the balance is not enough in the owners account, the barrier will still be lifted, but a warning email or an SMS will be sent to the owner otherwise the deducted amount is sent to the owner via email or SMS along with location and the next toll booth number. Before lifting the barrier, the server checks whether the vehicle is registered as well as valid or not. The registered owners have the barcode embedded in their vehicles and any complaint of the stolen vehicle is sent to the police using toll server database which makes the vehicle invalid. Thus a valid vehicle is the one which does not have any complaint against them. If the vehicle is both registered and valid, then the barrier is lifted otherwise alarm is generated to make the police alert and police will contact.

1. **QR code technology**

QR code (abbreviated from Quick Response Code) is the trademark for a type of matrix barcode (or twodimensional barcode) first designed for the automotive industry in Japan. A barcode is a machine-readable optical label that contains information about the item to which it is attached. A QR code uses four standardized encoding modes (numeric, alphanumeric, byte/binary, and kanji) to efficiently store data; A QR code consists of black modules (square dots) arranged in a square grid on a white background, which can be read by an imaging device (such as a camera, scanner, etc.) and processed using Reed–Solomon error correction until the image can be appropriately interpreted. The required data are then extracted from patterns that are present in both horizontal and vertical components of the image. Fig : QR Code image The above figure shows QR code image as we see the QR code is a two dimensional image and it is a trademark of a matrix type. The system has become popular due to fast readability and large storage capacity. Working of QR code technology The format information records two things: the error correction level and the mask pattern used for the symbol. Masking is used to break up patterns in the data area that might confuse a scanner, such as large blank areas or misleading features that look like the locator marks. The mask patterns are defined on a 6×6 grid that is repeated as necessary to cover the whole symbol. Modules corresponding to the dark areas of the mask are inverted. The Automatic Number Plate Recognition system using QR code consists of two major blocks as shown a. Lane level b. Software Level At the lane level, the CCTV (Closed Circuit Television) cameras take a snapshot of the car in such a way that the QR code is also included in the image. Otherwise a small optical QR code scanner can be installed for on spot recognition. As vehicles pass under the camera/scanner their QR codes are digitally recorded. The image/video/QR code embedded information is sent along with the date and time stamp to the remote computer. The QR code recognition technique is performed at the software level and consists of the following step. If direct information is sent from lane level it is forwarded for searching the record in database. Else if image/video feed is received then focus the QR code recognition software component on to the QR code received to retrieve the information and then search for the record in database.

## 4.2 BENEFITS:

There are different benefits of this project like

* Human control over traffic signal change.
* No wait to emergency vehicles.

## 4.3 PROPOSED SOLUTION:

* The existing traffic system is efficient but is dependent on a timer circuit that is static so by connecting these signals to the internet we can increase there functionalities.
* In our project, we have focused upon the release of congestion that leads to delay in life saving services. Here, we will track the emergency vehicle will use time division signaling, so that as soon as the vehicle arrives at the traffic signal , we will release that lane by giving a green signal. This way, the emergency vehicle will not have to wait and thus they can reach the place within time.

## 4.4 TECHNOLOGY USED:

***Front End***:

* + HTML
  + CSS
  + JQuery
  + Angular JS
  + Android

***Back End :***

* Arduino
* Database-MYSQL
* JAVA

***ICs:***

* ESP8266-01
* 4017 & 555 timer
* 7404
* 7408
* 7432

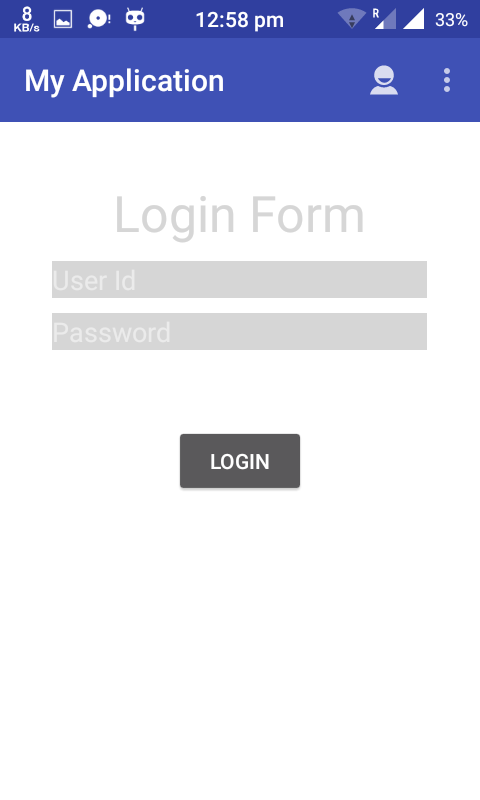
**CHAPTER 5**

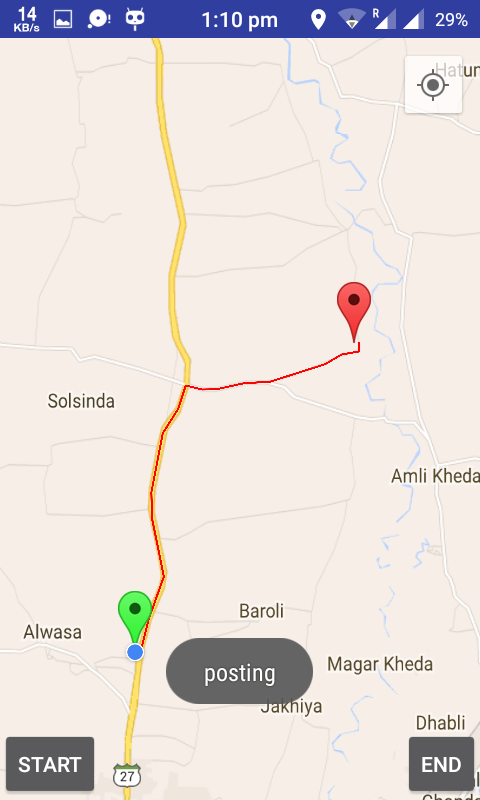
**TECHNICAL PART**

# TECHNICAL PART:

1. The user(driver) will login with its user id and password.This data will be

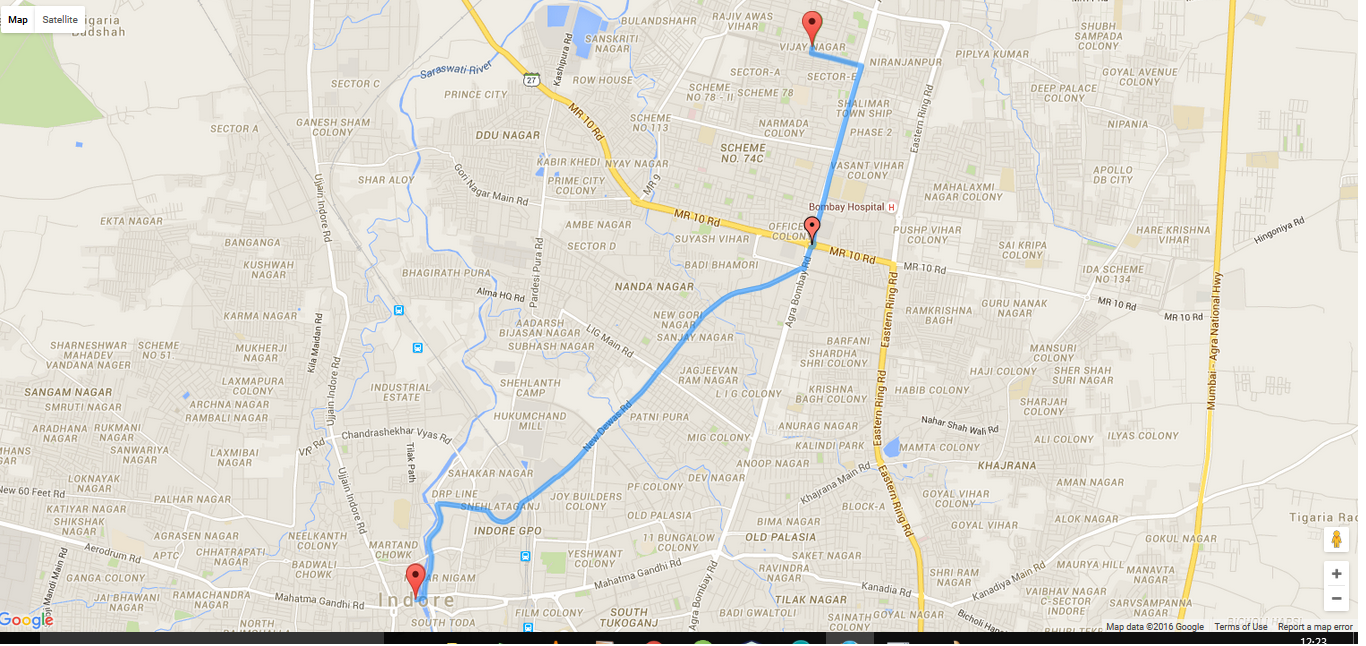
authenticated at the server end.



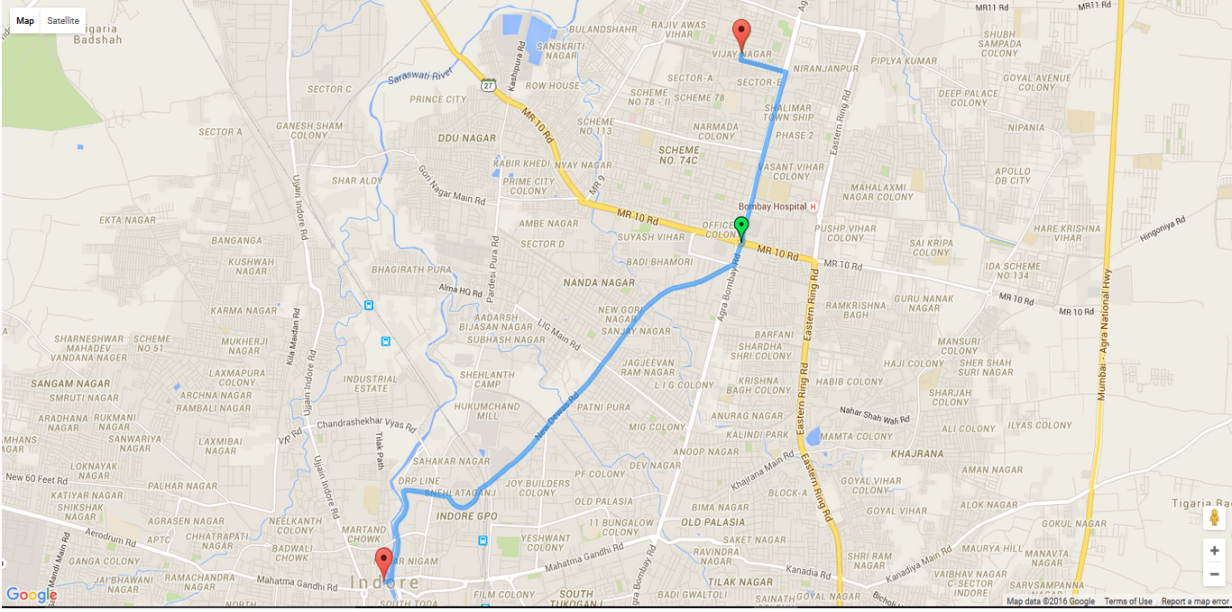


1. Then user will specify the destination and start the journey.The current changing location of the user will be tracked at the server end**.**

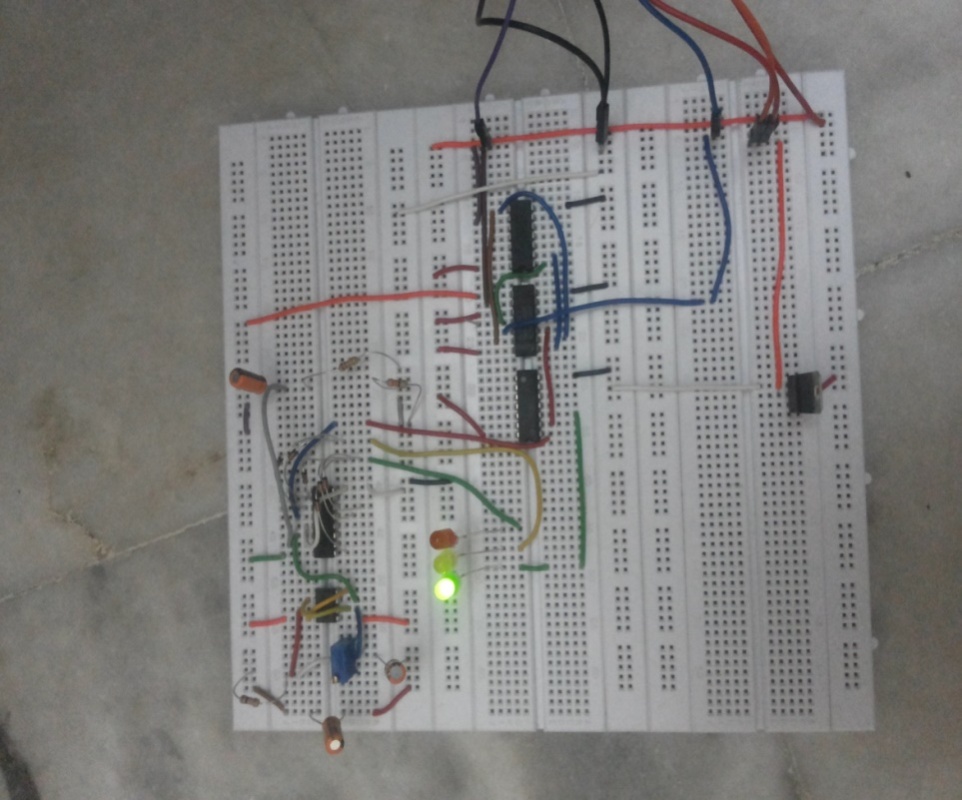
**Traffic off:**

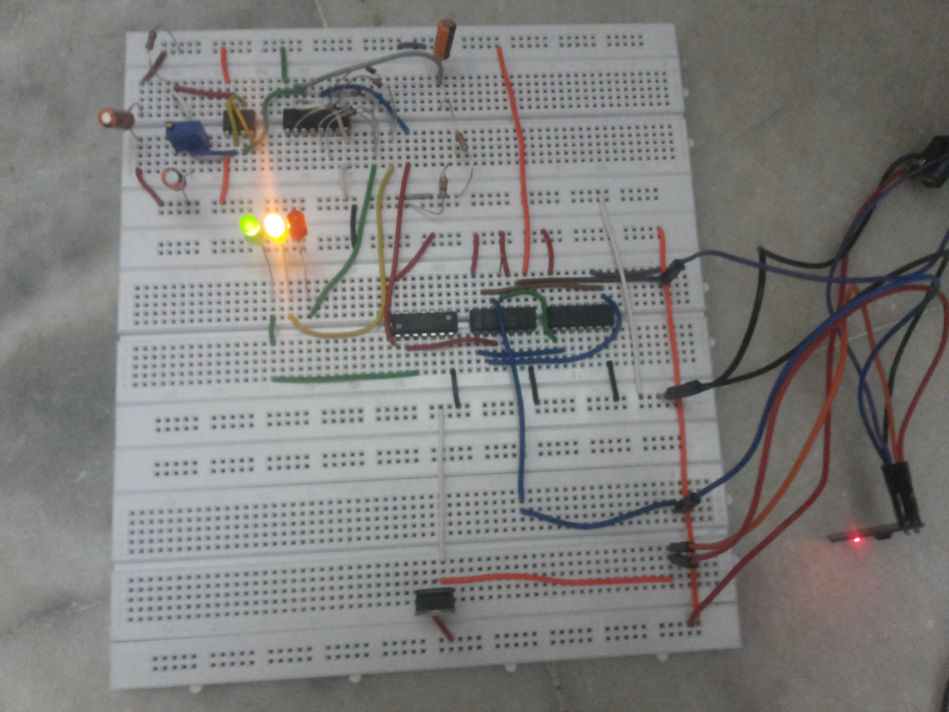


**Traffic on:**



1. As soon as the user reaches in the range of a traffic signal and the lane in which the user arrives will be shown a green signal.Thus the user need not to wait for the static signals to turn green**.**





SAMPLE CODE:

**LOGIN**

**package** a.myapplication;  
  
**import** android.content.Context;  
**import** android.content.Intent;  
**import** android.net.ConnectivityManager;  
**import** android.net.NetworkInfo;  
**import** android.os.Bundle;  
**import** android.support.v7.app.ActionBarActivity;  
**import** android.support.v7.widget.Toolbar;  
**import** android.view.Menu;  
**import** android.view.MenuItem;  
**import** android.view.View;  
**import** android.widget.Button;  
**import** android.widget.EditText;  
**import** android.widget.TextView;  
**import** android.widget.Toast;  
  
**import** java.util.concurrent.ExecutionException;  
  
  
**public class** MainActivity **extends** ActionBarActivity {  
 EditText **et\_username**, **et\_password**;  
 **public static** TextView *textView*;  
 EditText **IP**;  
 Button **b1**, **b2**;  
 String **username**, **password**,**ipString**;  
  
 **private** Toolbar **toolbar**;  
  
  
 @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.***activity\_main***);  
  
 **toolbar** = (Toolbar) findViewById(R.id.***app\_bar***);  
 setSupportActionBar(**toolbar**);  
  
 **et\_username** = (EditText) findViewById(R.id.***et\_login\_userName***);  
 **et\_password** = (EditText) findViewById(R.id.***et\_login\_userPassword***);  
 *textView* = (TextView) findViewById(R.id.***tv\_noInternet***);  
 **IP** = (EditText) findViewById(R.id.***ipEditText***);  
 **b1** = (Button) findViewById(R.id.***bt\_login***);  
 **b2** = (Button) findViewById(R.id.***bt\_register***);  
  
 *textView*.setVisibility(View.***INVISIBLE***);  
}  
  
 @Override  
 **public boolean** onCreateOptionsMenu(Menu menu) {  
 getMenuInflater().inflate(R.menu.***menu\_main***,menu);  
 **return true**;  
 }  
  
 @Override  
 **public boolean** onOptionsItemSelected(MenuItem item) {  
  
 **int** id=item.getItemId();  
  
 **if**(id==R.id.***action\_settings***)  
 {  
 **return true**;  
 }  
 **if**(id==R.id.***navigate***)  
 {  
 startActivity(**new** Intent(**this**,Register.**class**));  
 }  
  
 **return super**.onOptionsItemSelected(item);  
  
 }  
  
 **public void** userLogin(View view) {  
  
 **username** = **et\_username**.getText().toString();  
 **password** = **et\_password**.getText().toString();  
 **ipString**=**IP**.getText().toString();  
 String method = **"login"**;  
 BackgroundTask bg = **new** BackgroundTask(**this**);  
 String result;  
  
 **try** {  
 result = bg.execute(method, **username**, **password**,**ipString**).get();  
 Toast.*makeText*(MainActivity.**this**, **""**+result, Toast.***LENGTH\_SHORT***).show();  
 *//Intent i = new Intent(this, DisplayListView.class);* Toast.*makeText*(MainActivity.**this**, **""**+result, Toast.***LENGTH\_SHORT***).show();  
 String success=result.split(**","**)[0].toString();  
 String userid=result.split(**","**)[1].toString();  
  
 **if**(success.equals(**"true"**))  
 {  
  
 Intent i = **new** Intent(**this**, Map.**class**);  
 *//i.putExtra("json\_data", result);* i.putExtra(**"userid"**, userid);  
  
 i.putExtra(**"ipString"**,**ipString**);  
 startActivity(i);  
 *//System.out.println("login wala"+result);* }  
 **else** {  
  
 Toast.*makeText*(MainActivity.**this**, **"Invalid username or password"**, Toast.***LENGTH\_LONG***).show();  
 }  
  
  
 } **catch** (InterruptedException e) {  
 e.printStackTrace();  
 } **catch** (ExecutionException e) {  
 e.printStackTrace();  
 }  
  
  
 }  
  
 **public void** userReg(View view) {  
 startActivity(**new** Intent(**this**, Register.**class**));  
  
 }  
  
}

**Map(Android)**

**package** a.myapplication;  
  
**import** android.content.Context;  
**import** android.content.Intent;  
**import** android.content.pm.PackageManager;  
**import** android.graphics.Color;  
**import** android.location.Location;  
**import** android.location.LocationListener;  
**import** android.location.LocationManager;  
**import** android.os.AsyncTask;  
**import** android.support.v4.app.ActivityCompat;  
**import** android.support.v4.app.FragmentActivity;  
**import** android.os.Bundle;  
**import** android.util.Log;  
**import** android.view.Menu;  
**import** android.view.View;  
**import** android.widget.Button;  
**import** android.widget.Toast;  
  
**import** com.google.android.gms.maps.CameraUpdateFactory;  
**import** com.google.android.gms.maps.GoogleMap;  
**import** com.google.android.gms.maps.OnMapReadyCallback;  
**import** com.google.android.gms.maps.SupportMapFragment;  
**import** com.google.android.gms.maps.model.BitmapDescriptor;  
**import** com.google.android.gms.maps.model.BitmapDescriptorFactory;  
**import** com.google.android.gms.maps.model.LatLng;  
**import** com.google.android.gms.maps.model.MarkerOptions;  
**import** com.google.android.gms.maps.model.PolylineOptions;  
  
**import** org.apache.http.client.HttpClient;  
**import** org.json.JSONObject;  
  
**import** java.io.BufferedReader;  
**import** java.io.IOException;  
**import** java.io.InputStream;  
**import** java.io.InputStreamReader;  
**import** java.net.HttpURLConnection;  
**import** java.net.URL;  
**import** java.util.ArrayList;  
**import** java.util.HashMap;  
**import** java.util.List;  
  
**import** org.apache.http.HttpResponse;  
**import** org.apache.http.client.ClientProtocolException;  
**import** org.apache.http.client.HttpClient;  
**import** org.apache.http.client.methods.HttpGet;  
**import** org.apache.http.impl.client.DefaultHttpClient;  
**import** org.json.JSONObject;  
  
  
  
**public class** Map **extends** FragmentActivity{  
 GoogleMap **map**;  
 ArrayList<LatLng> **markerPoints**,**directionPoints**,**trafficSignalMarkerPoints**;  
 **int userid**;  
 Button **start**,**end**;  
 **boolean isTravelling**=**false**;  
 String **ipString**;  
 BitmapDescriptor **trafficIcon**;  
 **boolean doCheck**=**false**;  
  
 @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.***activity\_map***);  
  
 **trafficSignalMarkerPoints** =**new** ArrayList<LatLng>();  
 **trafficSignalMarkerPoints**.add(**new** LatLng(22.7493,75.9036));  
  
  
 **start**=(Button)findViewById(R.id.***startButton***);  
 **end**=(Button)findViewById(R.id.***endButton***);  
  
 **start**.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View v) {  
  
 **isTravelling**=**true**;  
 }  
 });  
  
 **end**.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View v) {  
 **isTravelling**=**false**;  
 }  
 });  
 *// Initializing* **markerPoints** = **new** ArrayList<LatLng>();  
  
  
 **userid**=Integer.*parseInt*(getIntent().getExtras().getString(**"userid"**));  
 *// Getting reference to SupportMapFragment of the activity\_main* SupportMapFragment fm = (SupportMapFragment) getSupportFragmentManager().findFragmentById(R.id.***map***);  
  
 *// Getting Map for the SupportMapFragment* **map** = fm.getMap();  
  
  
  
 **if** (**map** != **null**) {  
  
 *// Enable MyLocation Button in the Map* **if** (ActivityCompat.*checkSelfPermission*(**this**, android.Manifest.permission.***ACCESS\_FINE\_LOCATION***) != PackageManager.***PERMISSION\_GRANTED*** && ActivityCompat.*checkSelfPermission*(**this**, android.Manifest.permission.***ACCESS\_COARSE\_LOCATION***) != PackageManager.***PERMISSION\_GRANTED***) {  
**return**;  
 }  
 **map**.setMyLocationEnabled(**true**);  
  
  
  
  
  
  
  
  
  
  
  
  
  
 **final** MarkerOptions trafficMarkerOptions=**new** MarkerOptions();  
 trafficMarkerOptions.position(**trafficSignalMarkerPoints**.get(0));  
 **trafficIcon** = BitmapDescriptorFactory.*defaultMarker*(BitmapDescriptorFactory.***HUE\_BLUE***);  
  
  
  
 *// Acquire a reference to the system Location Manager* LocationManager locationManager = (LocationManager) Map.**this**.getSystemService(Context.***LOCATION\_SERVICE***);  
 *// Define a listener that responds to location updates* LocationListener locationListener = **new** LocationListener() {  
 **public void** onLocationChanged(Location location) {  
  
  
 *//Convert Location to LatLng* LatLng gpsPoint = **new** LatLng(location.getLatitude(), location.getLongitude());  
  
 *// Creating MarkerOptions* MarkerOptions optionsG = **new** MarkerOptions();  
  
  
**if**(**markerPoints**.size()<1)  
 {  
 **markerPoints**.add(0,gpsPoint);  
 }  
 **else** {  
 **markerPoints**.set(0, gpsPoint);  
 }  
  
  
 *// Setting the position of the marker* optionsG.position(gpsPoint);  
  
  
  
 */\*\*  
 \* For the start location, the color of marker is GREEN and  
 \* for the end location, the color of marker is RED.  
 \*/* optionsG.icon(BitmapDescriptorFactory.*defaultMarker*(BitmapDescriptorFactory.***HUE\_GREEN***));  
  
  
 *// Creating MarkerOptions* MarkerOptions optionsR = **new** MarkerOptions();  
  
 **if**(**markerPoints**.size()>1) {  
 *// Setting the position of the marker* optionsR.position(**markerPoints**.get(1));  
 *//Toast.makeText(Map.this, "hello"+markerPoints.size(), Toast.LENGTH\_SHORT).show();  
  
  
  
 /\*\*  
 \* For the start location, the color of marker is GREEN and  
 \* for the end location, the color of marker is RED.  
 \*/* optionsR.icon(BitmapDescriptorFactory.*defaultMarker*(BitmapDescriptorFactory.***HUE\_RED***));  
  
 **map**.clear();  
 **map**.addMarker(optionsG);  
 **map**.addMarker(optionsR);  
 trafficMarkerOptions.icon(**trafficIcon**);  
 **map**.addMarker(trafficMarkerOptions);  
}  
**else** {  
 **map**.clear();  
 trafficMarkerOptions.icon(**trafficIcon**);  
 **map**.addMarker(optionsG);  
 }  
  
  
  
 **if**(**markerPoints**.size()>1 && **isTravelling**==**true**) {  
 *//Toast.makeText(Map.this, "posting", Toast.LENGTH\_SHORT).show();  
 //checkStatus();* **doCheck**=**true**;  
 postData(**userid**, **markerPoints**.get(0).**latitude**, **markerPoints**.get(0).**longitude**, **markerPoints**.get(1).**latitude**, **markerPoints**.get(1).**longitude**);  
 }  
  
  
  
  
 *// Checks, whether start and end locations are captured* **if**(**markerPoints**.size() >= 2){  
 LatLng origin = **markerPoints**.get(0);  
 LatLng dest = **markerPoints**.get(1);  
  
 *// Getting URL to the Google Directions API* String url = getDirectionsUrl(origin, dest);  
  
 DownloadTask downloadTask = **new** DownloadTask();  
  
 *// Start downloading json data from Google Directions API* downloadTask.execute(url);  
  
 }  
  
 **ipString**=getIntent().getExtras().getString(**"ipString"**);  
  
 checkStatus();  
  
 }  
  
 **public void** onStatusChanged(String provider, **int** status, Bundle extras) {  
 }  
  
 **public void** onProviderEnabled(String provider) {  
 }  
  
 **public void** onProviderDisabled(String provider) {  
 }  
  
  
  
  
  
  
  
  
  
  
 };  
 *// Register the listener with the Location Manager to receive location updates* **if** (ActivityCompat.*checkSelfPermission*(Map.**this**, android.Manifest.permission.***ACCESS\_FINE\_LOCATION***) != PackageManager.***PERMISSION\_GRANTED*** && ActivityCompat.*checkSelfPermission*(Map.**this**, android.Manifest.permission.***ACCESS\_COARSE\_LOCATION***) != PackageManager.***PERMISSION\_GRANTED***) {  
 *//* ***TODO: Consider calling*** *// ActivityCompat#requestPermissions  
 // here to request the missing permissions, and then overriding  
 // public void onRequestPermissionsResult(int requestCode, String[] permissions,  
 // int[] grantResults)  
 // to handle the case where the user grants the permission. See the documentation  
 // for ActivityCompat#requestPermissions for more details.* **return**;  
 }  
 **if** (ActivityCompat.*checkSelfPermission*(Map.**this**, android.Manifest.permission.***ACCESS\_FINE\_LOCATION***) != PackageManager.***PERMISSION\_GRANTED*** && ActivityCompat.*checkSelfPermission*(Map.**this**, android.Manifest.permission.***ACCESS\_COARSE\_LOCATION***) != PackageManager.***PERMISSION\_GRANTED***) {  
 *//* ***TODO: Consider calling*** *// ActivityCompat#requestPermissions  
 // here to request the missing permissions, and then overriding  
 // public void onRequestPermissionsResult(int requestCode, String[] permissions,  
 // int[] grantResults)  
 // to handle the case where the user grants the permission. See the documentation  
 // for ActivityCompat#requestPermissions for more details.* **return**;  
 }  
 locationManager.requestLocationUpdates(LocationManager.***NETWORK\_PROVIDER***, 0, 0, locationListener);  
  
  
  
  
 *// Setting onclick event listener for the map* **map**.setOnMapClickListener(**new** GoogleMap.OnMapClickListener() {  
  
 @Override  
 **public void** onMapClick(LatLng point) {  
  
 *// Already two locations  
  
 // Adding new item to the ArrayList* **if**(**markerPoints**.size()<2) {  
 **markerPoints**.add(1, point);  
 }  
 **else**{  
 **markerPoints**.set(1,point);  
 }  
 *// Creating MarkerOptions* MarkerOptions options1 = **new** MarkerOptions();  
  
 *// Setting the position of the marker* options1.position(point);  
  
 */\*\*  
 \* For the start location, the color of marker is GREEN and  
 \* for the end location, the color of marker is RED.  
 \*/* options1.icon(BitmapDescriptorFactory.*defaultMarker*(BitmapDescriptorFactory.***HUE\_RED***));  
 *// Creating MarkerOptions* MarkerOptions options0 = **new** MarkerOptions();  
  
 *// Setting the position of the marker* options0.position(**markerPoints**.get(0));  
  
 */\*\*  
 \* For the start location, the color of marker is GREEN and  
 \* for the end location, the color of marker is RED.  
 \*/* options0.icon(BitmapDescriptorFactory.*defaultMarker*(BitmapDescriptorFactory.***HUE\_GREEN***));  
 *// Add new marker to the Google Map Android API V2* **map**.clear();  
 **map**.addMarker(options0);  
 **map**.addMarker(options1);  
 trafficMarkerOptions.icon(**trafficIcon**);  
  
  
  
 *// Checks, whether start and end locations are captured* **if**(**markerPoints**.size() >= 2){  
 LatLng origin = **markerPoints**.get(0);  
 LatLng dest = **markerPoints**.get(1);  
  
 *// Getting URL to the Google Directions API* String url = getDirectionsUrl(origin, dest);  
  
 DownloadTask downloadTask = **new** DownloadTask();  
  
 *// Start downloading json data from Google Directions API* downloadTask.execute(url);  
  
 }  
  
  
  
 *//myDemoLocation();* }  
 });  
 }  
 }  
**private** String getDirectionsUrl(LatLng origin,LatLng dest){  
  
 *// Origin of route* String str\_origin = **"origin="**+origin.**latitude**+**","**+origin.**longitude**;  
  
 *// Destination of route* String str\_dest = **"destination="**+dest.**latitude**+**","**+dest.**longitude**;  
  
  
 *// Sensor enabled* String sensor = **"sensor=false"**;  
  
 *// Building the parameters to the web service* String parameters = str\_origin+**"&"**+str\_dest+**"&"**+sensor;  
  
 *// Output format* String output = **"json"**;  
  
 *// Building the url to the web service* String url = **"https://maps.googleapis.com/maps/api/directions/"**+output+**"?"**+parameters;  
  
  
 **return** url;  
 }  
  
 */\*\* A method to download json data from url \*/* **private** String downloadUrl(String strUrl) **throws** IOException {  
 String data = **""**;  
 InputStream iStream = **null**;  
 HttpURLConnection urlConnection = **null**;  
 **try**{  
 URL url = **new** URL(strUrl);  
  
 *// Creating an http connection to communicate with url* urlConnection = (HttpURLConnection) url.openConnection();  
  
 *// Connecting to url* urlConnection.connect();  
  
 *// Reading data from url* iStream = urlConnection.getInputStream();  
  
 BufferedReader br = **new** BufferedReader(**new** InputStreamReader(iStream));  
  
 StringBuffer sb = **new** StringBuffer();  
  
 String line = **""**;  
 **while**( ( line = br.readLine()) != **null**){  
 sb.append(line);  
 }  
  
 data = sb.toString();  
  
 br.close();  
  
 }**catch**(Exception e){  
 Log.*d*(**"Exception ding url"**, e.toString());  
 }**finally**{  
 iStream.close();  
 urlConnection.disconnect();  
 }  
 **return** data;  
 }  
  
  
  
 *// Fetches data from url passed* **private class** DownloadTask **extends** AsyncTask<String, Void, String> {  
  
 *// Downloading data in non-ui thread* @Override  
 **protected** String doInBackground(String... url) {  
  
 *// For storing data from web service* String data = **""**;  
  
 **try**{  
 *// Fetching the data from web service* data = downloadUrl(url[0]);  
 }**catch**(Exception e){  
 Log.*d*(**"Background Task"**,e.toString());  
 }  
 **return** data;  
 }  
  
 *// Executes in UI thread, after the execution of  
 // doInBackground()* @Override  
 **protected void** onPostExecute(String result) {  
 **super**.onPostExecute(result);  
  
 ParserTask parserTask = **new** ParserTask();  
  
 *// Invokes the thread for parsing the JSON data* parserTask.execute(result);  
  
 }  
 }  
  
 */\*\* A class to parse the Google Places in JSON format \*/* **private class** ParserTask **extends** AsyncTask<String, Integer, List<List<HashMap<String,String>>> >{  
  
 *// Parsing the data in non-ui thread* @Override  
 **protected** List<List<HashMap<String, String>>> doInBackground(String... jsonData) {  
  
 JSONObject jObject;  
 List<List<HashMap<String, String>>> routes = **null**;  
  
 **try**{  
 jObject = **new** JSONObject(jsonData[0]);  
 DirectionsJSONParser parser = **new** DirectionsJSONParser();  
  
 *// Starts parsing data* routes = parser.parse(jObject);  
 }**catch**(Exception e){  
 e.printStackTrace();  
 }  
 **return** routes;  
 }  
  
 *// Executes in UI thread, after the parsing process* @Override  
 **protected void** onPostExecute(List<List<HashMap<String, String>>> result) {  
 ArrayList<LatLng> points = **null**;  
 PolylineOptions lineOptions = **null**;  
 MarkerOptions markerOptions = **new** MarkerOptions();  
  
 *// Traversing through all the routes* **for**(**int** i=0;i<result.size();i++){  
 points = **new** ArrayList<LatLng>();  
  
 lineOptions = **new** PolylineOptions();  
  
 *// Fetching i-th route* List<HashMap<String, String>> path = result.get(i);  
  
 *// Fetching all the points in i-th route* **for**(**int** j=0;j<path.size();j++){  
 HashMap<String,String> point = path.get(j);  
  
 **double** lat = Double.*parseDouble*(point.get(**"lat"**));  
 **double** lng = Double.*parseDouble*(point.get(**"lng"**));  
 LatLng position = **new** LatLng(lat, lng);  
  
 points.add(position);  
 *//directionPoints.add(position);* }  
  
 *// Adding all the points in the route to LineOptions* lineOptions.addAll(points);  
  
 **public void** postData(**final int** userid,**final double** las,**final double** los,**final double** lad,**final double** lod) {  
  
Thread thread = **new** Thread(**new** Runnable(){  
 @Override  
 **public void** run() {  
 **try** {  
 *// Create a new HttpClient and Post Header* HttpClient httpclient = **new** DefaultHttpClient();  
String url=**"http://192.168.0.101:8080/GPSDemo/AddData?userid="**+userid+**"&las="**+las+**"&los="**+los+**"&lad="**+lad+**"&lod="**+lod;  
HttpGet htget = **new** HttpGet(url);  
 **try** {  
 *// Execute HTTP Post Request* HttpResponse response = httpclient.execute(htget);  
 String resp = response.getStatusLine().toString();  
 *//Toast.makeText(Map.this, resp, Toast.LENGTH\_LONG).show();* } **catch** (ClientProtocolException e) {  
 *//Toast.makeText(Map.this, "Error", Toast.LENGTH\_LONG).show();* } **catch** (IOException e) {  
 *//Toast.makeText(Map.this, "Error", Toast.LENGTH\_LONG).show();* }  
 } **catch** (Exception e) {  
 *//Toast.makeText(Map.this, "catch", Toast.LENGTH\_SHORT).show();* e.printStackTrace();  
 }  
 }  
 });  
  
 thread.start();  
  
  
  
 }  
  
 @Override  
 **public boolean** onCreateOptionsMenu(Menu menu) {  
 *// Inflate the menu; this adds items to the action bar if it is present.* getMenuInflater().inflate(R.menu.***main***, menu);  
 **return true**;  
 }  
  
  
  
 **public void** checkStatus()  
 {  
 Thread checkStatusThread=**null**;  
 **if**(**doCheck**==**true**) {  
 checkStatusThread = **new** Thread(**new** Runnable() {  
 @Override  
 **public void** run() {  
  
 String method = **"accepted"**;  
 BackgroundTask bg = **new** BackgroundTask(Map.**this**);  
 String status;  
 **while** (**true**) {  
 **try** {  
  
 status = bg.execute(method, **"192.168.0.101:8080"**, **userid** + **""**).get();  
 *//Intent i = new Intent(this, DisplayListView.class);* **if** (status.equals(**"1"**)) {  
 **trafficIcon** = BitmapDescriptorFactory.*defaultMarker*(BitmapDescriptorFactory.***HUE\_CYAN***);  
  
 *//System.out.println("login wala"+result);* } **else** {  
 **trafficIcon** = BitmapDescriptorFactory.*defaultMarker*(BitmapDescriptorFactory.***HUE\_CYAN***);  
  
 }  
 } **catch** (Exception e) {  
 e.printStackTrace();  
 }  
 **try** {  
 Thread.*sleep*(1000);  
 } **catch** (InterruptedException e) {  
 e.printStackTrace();  
 }  
 }  
 }  
 });  
  
 checkStatusThread.start();  
 }  
 **else if**(**doCheck**==**false** && checkStatusThread!=**null**)  
 {  
 checkStatusThread.stop();  
  
 }  
 }  
}

**DIRECTIONJSONPARSER.JAVA**

**package** a.myapplication;  
  
**import** com.google.android.gms.maps.model.LatLng;  
  
**import** org.json.JSONArray;  
**import** org.json.JSONException;  
**import** org.json.JSONObject;  
  
**import** java.util.ArrayList;  
**import** java.util.HashMap;  
**import** java.util.List;  
  
**public class** DirectionsJSONParser {  
   
 */\*\* Receives a JSONObject and returns a list of lists containing latitude and longitude \*/* **public** List<List<HashMap<String,String>>> parse(JSONObject jObject){  
   
 List<List<HashMap<String, String>>> routes = **new** ArrayList<List<HashMap<String,String>>>() ;  
 JSONArray jRoutes = **null**;  
 JSONArray jLegs = **null**;  
 JSONArray jSteps = **null**;   
   
 **try** {   
   
 jRoutes = jObject.getJSONArray(**"routes"**);  
   
 */\*\* Traversing all routes \*/* **for**(**int** i=0;i<jRoutes.length();i++){   
 jLegs = ( (JSONObject)jRoutes.get(i)).getJSONArray(**"legs"**);  
 List path = **new** ArrayList<HashMap<String, String>>();  
   
 */\*\* Traversing all legs \*/* **for**(**int** j=0;j<jLegs.length();j++){  
 jSteps = ( (JSONObject)jLegs.get(j)).getJSONArray(**"steps"**);  
   
 */\*\* Traversing all steps \*/* **for**(**int** k=0;k<jSteps.length();k++){  
 String polyline = **""**;  
 polyline = (String)((JSONObject)((JSONObject)jSteps.get(k)).get(**"polyline"**)).get(**"points"**);  
 List<LatLng> list = decodePoly(polyline);  
   
 */\*\* Traversing all points \*/* **for**(**int** l=0;l<list.size();l++){  
 HashMap<String, String> hm = **new** HashMap<String, String>();  
 hm.put(**"lat"**, Double.*toString*(((LatLng)list.get(l)).**latitude**) );  
 hm.put(**"lng"**, Double.*toString*(((LatLng)list.get(l)).**longitude**) );  
 path.add(hm);   
 }   
 }  
 routes.add(path);  
 }  
 }  
   
 } **catch** (JSONException e) {   
 e.printStackTrace();  
 }**catch** (Exception e){   
 }  
   
   
 **return** routes;  
 }   
   
   
 */\*\*  
 \* Method to decode polyline points   
 \* Courtesy : http://jeffreysambells.com/2010/05/27/decoding-polylines-from-google-maps-direction-api-with-java   
 \* \*/* **private** List<LatLng> decodePoly(String encoded) {  
  
 List<LatLng> poly = **new** ArrayList<LatLng>();  
 **int** index = 0, len = encoded.length();  
 **int** lat = 0, lng = 0;  
  
 **while** (index < len) {  
 **int** b, shift = 0, result = 0;  
 **do** {  
 b = encoded.charAt(index++) - 63;  
 result |= (b & 0x1f) << shift;  
 shift += 5;  
 } **while** (b >= 0x20);  
 **int** dlat = ((result & 1) != 0 ? ~(result >> 1) : (result >> 1));  
 lat += dlat;  
  
 shift = 0;  
 result = 0;  
 **do** {  
 b = encoded.charAt(index++) - 63;  
 result |= (b & 0x1f) << shift;  
 shift += 5;  
 } **while** (b >= 0x20);  
 **int** dlng = ((result & 1) != 0 ? ~(result >> 1) : (result >> 1));  
 lng += dlng;  
  
 LatLng p = **new** LatLng((((**double**) lat / 1E5)),  
 (((**double**) lng / 1E5)));  
 poly.add(p);  
 }  
  
 **return** poly;  
 }  
}

**MAP(WEB)**

<%@ page import="java.io.\*,java.util.\*" %>

<%@page import="java.io.PrintWriter"%>

<%@page import="java.sql.Connection" %>

<%@page import="java.sql.DriverManager"%>

<%@page import="java.sql.ResultSet"%>

<%@page import="java.sql.PreparedStatement"%>

<%@page contentType="text/html" pageEncoding="UTF-8"%>

<!DOCTYPE html>

<html>

<head>

<meta http-equiv="content-type" content="text/html; charset=UTF-8"/>

<title> Maps Direction</title>

<script src="http://maps.google.com/maps?file=api&amp;v=2&amp;sensor=false"

type="text/javascript"></script>

<script> </script>

</head>

<body onunload="GUnload()">

<%

// Set refresh, autoload time as 5 seconds

//response.setIntHeader("Refresh", 2);

double las=0,los=0;

ResultSet rs;

//int userid=Integer.parseInt(request.getParameter("userid"));

Class.forName("com.mysql.jdbc.Driver");

Connection con=DriverManager.getConnection("jdbc:mysql://localhost/test","root","root");

String query="SELECT \* FROM location where userid =?";

PreparedStatement pst=con.prepareStatement(query);

pst.setInt(1, 1);

rs =pst.executeQuery();

rs.next();

las =rs.getDouble("las");

los =rs.getDouble("las");

double lad =rs.getDouble("lad");

double lod =rs.getDouble("lod");

%>

<div id="map" style="width: 1340px; height: 610px"></div>

<script type="text/javascript">

var las=<%=rs.getDouble("las")%>;

var los=<%=rs.getDouble("los")%>;

var lad=<%=rs.getDouble("lad")%>;

var lod=<%=rs.getDouble("lod")%>;

var myLatLng = {lat: las, lng: los };

var desLatLng= {lat: lad, lng: lod };

var map = new GMap2(document.getElementById("map"));

var directions = new GDirections(map);

var headMarker,tailMarker,demo;

map.setCenter(new GLatLng(las,los), 16);

headMarker = new GMarker(myLatLng);

//map.addOverlay(headMarker);

demo=new GMarker()

tailMarker = new GMarker(desLatLng);

//map.addOverlay(tailMarker);

directions.load("from:" + headMarker.getPoint().lat()+ ", " +

headMarker.getPoint().lng() +

" to:" + tailMarker.getPoint().lat() + "," +

tailMarker.getPoint().lng(),

{ getPolyline: true, getSteps: true });

google.maps.event.addListener(tailMarker, 'click', function () {

window.alert("Click Received") ;

/\*var xmlHttp = new XMLHttpRequest();

xmlHttp.open( "GET","/GPSDemo/AddData?las=22.7684&los=75.8957&lad=22.7196&lod=75.8577&userid=12", false ); // false for synchronous request

xmlHttp.send( null );

return xmlHttp.responseText;\*/

});

</script>

</body>

</html>

**CHAPTER 6**

**SOFTWARE ENGINEERING APPROACH**

SOFTWARE ENGINEERING APPROACH:

## 6.1 SOFTWARE ENGINEERING PARADIGM APPLIED DESCRIPTION:

# AGILE SOFTWARE DEVELOPMENT

Agile Software Development 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 software development is a set of principles for software development in which requirements and solutions evolve through collaboration between self-organizing cross-functional teams. It promotes adaptive planning, evolutionary development, early delivery, and continuous improvement, and it encourages rapid and flexible response to change.

Agile Methods break the product into small incremental builds. These builds are provided in iterations. Each iteration typically lasts from about one to three weeks. Iteration involves cross functional teams working simultaneously on various areas like planning, requirements analysis, design, coding, unit testing, and acceptance testing. At the end of the iteration a working product is displayed to the customer and important stakeholders.

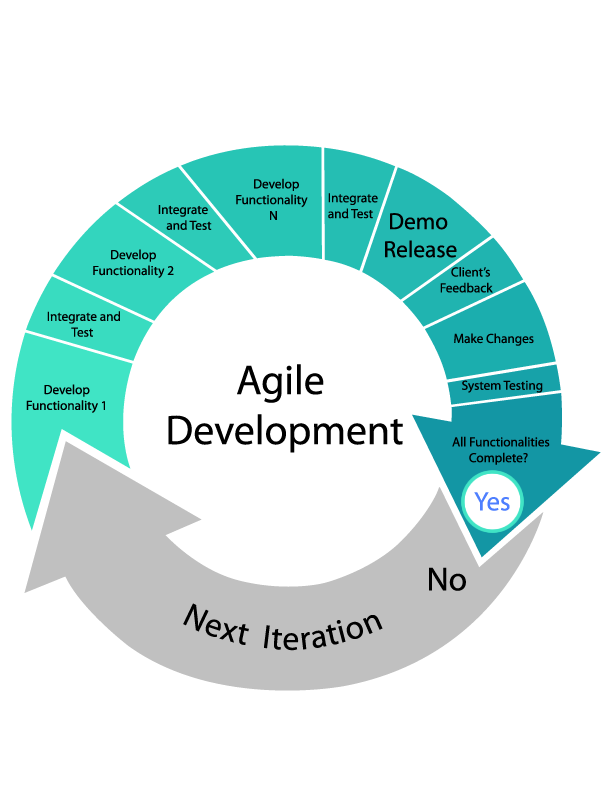
In agile the tasks are divided to time boxes (small time frames) to deliver specific features for a release. Iterative approach is taken and working software build is delivered after each iteration. Each build is incremental in terms of features; the final build holds all the features required by the customer. Agile thought process had started early in the software development and started becoming popular with time due to its flexibility and adaptability.

The most popular agile methods include Rational Unified Process (1994), Scrum (1995), Crystal Clear, Extreme Programming (1996), Adaptive Software Development, Feature Driven Development, and Dynamic Systems Development Method (DSDM) (1995). These are now collectively referred to as agile methodologies, after the Agile Manifesto was published in 2001.

#### Agile principles

The Agile Manifesto is based on twelve principles:

1. Customer satisfaction by early and continuous delivery of valuable software
2. Welcome changing requirements, even in late development
3. Working software is delivered frequently (weeks rather than months)
4. Close, daily cooperation between business people and developers
5. Projects are built around motivated individuals, who should be trusted
6. Face-to-face conversation is the best form of communication (co-location)
7. Working software is the principal measure of progress
8. Sustainable development, able to maintain a constant pace
9. Continuous attention to technical excellence and good design
10. Simplicity—the art of maximizing the amount of work not done—is essential
11. Best architectures, requirements, and designs emerge from self-organizing teams
12. Regularly, the team reflects on how to become more effective, and adjusts accordingly.



ADVANTAGES OF AGILE APPROACH:

* Is a very realistic approach to software development
* Promotes teamwork and cross training.
* Functionality can be developed rapidly and demonstrated.
* Resource requirements are minimum.
* Suitable for fixed or changing requirements
* Delivers early partial working solutions.
* Good model for environments that change steadily.
* Minimal rules, documentation easily employed.
* Enables concurrent development and delivery within an overall planned context.
* Little or no planning required
* Easy to manage
* Gives flexibility to developers

DISADVANTAGES OF AGILE APPROACH:

* Not suitable for handling complex dependencies.
* More risk of sustainability, maintainability and extensibility.
* An overall plan, an agile leader and agile PM practice is a must without which it will not work.
* Strict delivery management dictates the scope, functionality to be delivered, and adjustments to meet the deadlines.
* Depends heavily on customer interaction, so if customer is not clear, team can be driven in the wrong direction.
* There is very high individual dependency, since there is minimum documentation generated.
* Transfer of technology to new team members may be quite challenging due to lack of documentation.

## 6.2 REQUIREMENT ANALYSIS:

## 6.2.1 SOFTWARE REQUIREMENT SPECIFICATION:

Software requirement specification (SRS) is a technical specification of requirements for the software product. SRS represents an overview of products, features and summaries the processing environments for development operation and maintenance of the product. The goal of the requirement specification phase is to produce the software specification document also called requirement document.

**Requirement Specification**

This requirement specification must have the system properties. Conceptually every SRS should have the components:

* Functionality
* Performance
* Design constraints imposed on an implementation
* External interfaces

## 6.2.1.1 GLOSSARY:

* **JAVA Servlet** is a server-side scripting language designed for web development but also used as a general-purpose programming language.
* **JSON** stands for JavaScript Object Notation. **JSON** is a lightweight data-interchange format. **JSON** is language independent \* **JSON** is "self-describing" and easy to understand.
* Extensible Markup Language (**XML**) is a markup language that defines a set of rules for encoding documents in a format which is both human-readable and machine-readable. It is defined by the W3C's **XML** 1.0 Specification and by several other related specifications, all of which are free open standards.

## **6.2.1.2 SUPPLEMENTARY SPECIFICATION**:

**End user specification:**

* When user opens the application, login page is visible first.
* User has to login to avail the service.
* If he enters the valid username and password the he reaches the map page
* Then he can set the destination by clicking on the map
* After selecting the destination he can start the journey.

## 6.2.1.3 USE CASE MODEL:

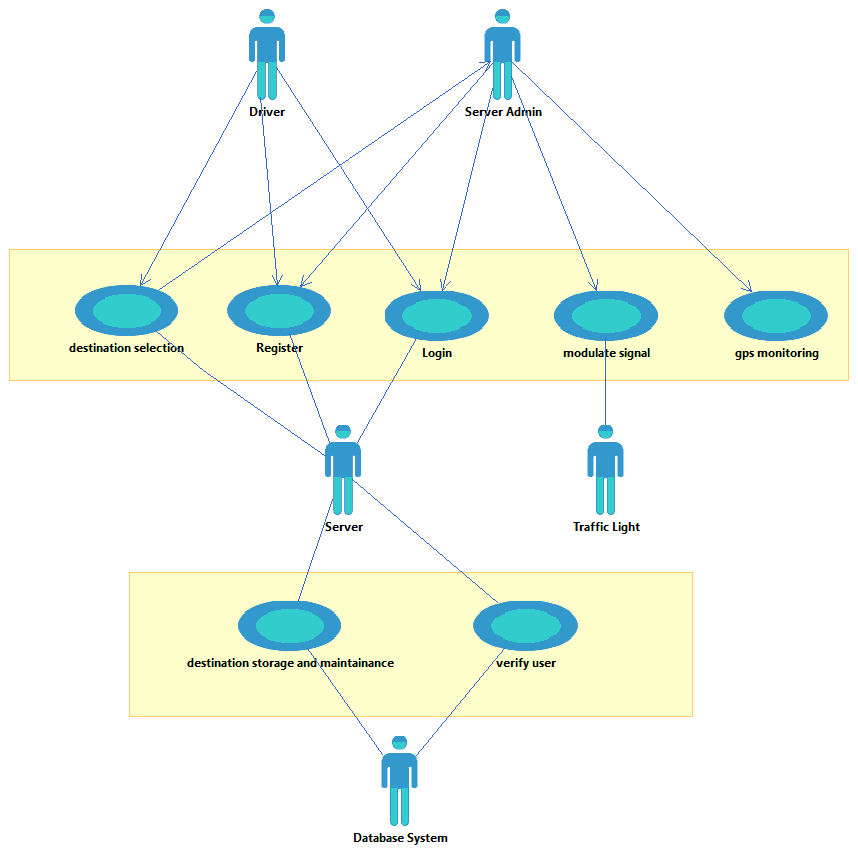
In software and systems engineering, a use case is a list of actions or event steps, typically defining the interactions between a role (known in the Unified Modelling Language as an actor) and a system, to achieve a goal. The actor can be a human or other external system.

Use case analysis is an important and valuable requirement analysis technique that has been widely used in modern software engineering since its formal introduction by Ivar Jacobson in 1992. Use case driven development is a key characteristic of many process models and frameworks such as ICONIX, the Unified Process (UP), the IBM Rational Unified Process (RUP), and the Oracle Unified Method (OUM). With its inherent iterative, incremental and evolutionary nature, use case also fits well for agile development. Use cases are not only texts, but also diagrams, if needed. The purpose of use case diagram is to capture the dynamic aspect of a system. But this definition is too generic to describe the purpose.

Because other four diagrams (activity, sequence, collaboration and State chart) are also having the same purpose. So we will look into some specific purpose which will distinguish it from other four diagrams. Use case diagrams are used to gather the requirements of a system including internal and external influences. These requirements are mostly design requirements. So when a system is analysed to gather its functionalities use cases are prepared and actors are identified. Now when the initial task is complete use case diagrams are modelled to present the outside view.

So in brief, the purposes of use case diagrams can be as follows:

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



## Conceptual Level class diagram:

In software engineering, a class diagram in the Unified Modelling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among objects. The class diagram is the main building block of object-oriented modelling. It is used both for general conceptual modelling of the systematics of the application, and for detailed modelling translating the models into programming code. Class diagrams can also be used for data modelling. The classes in a class diagram represent both the main elements, interactions in the application, and the classes to be programmed.

In the diagram, classes are represented with boxes that contain three compartments:

* The top compartment contains the name of the class. It is printed in bold and centred, and the first letter is capitalized.
* The middle compartment contains the attributes of the class. They are left-aligned and the first letter is lowercase.
* The bottom compartment contains the operations the class can execute. They are also left-aligned and the first letter is lowercase.

In the design of a system, a number of classes are identified and grouped together in a class diagram that helps to determine the static relations between them. With detailed modelling, the classes of the conceptual design are often split into a number of subclasses.

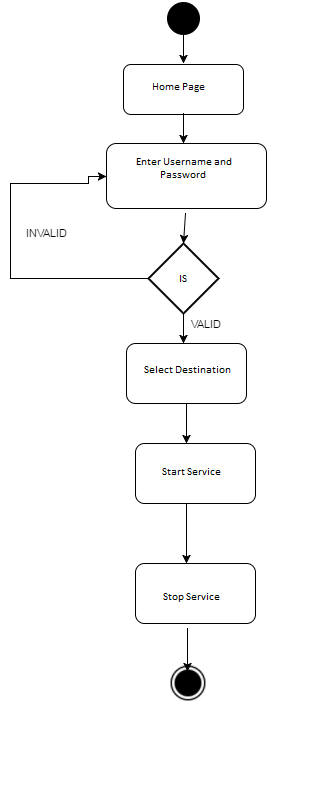
## Conceptual Level Sequence diagram:

A Sequence diagram is an interaction diagram that shows how processes operate with one another and in what order. It is a construct of 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. A sequence diagram shows, as parallel vertical lines (lifelines), different processes or objects that live simultaneously, and, as horizontal arrows, the messages exchanged between them, in the order in which they occur. This allows the specification of simple runtime scenarios in a graphical manner.

## Conceptual Level Activity diagram:

**An activity diagram** is a UML diagram that models the dynamic aspects of a system. It is a simplification of the UML state chart diagram for modeling control flows in computational and organizational processes. It allows you to represent a functional decomposition of a system behavior. An activity diagram provides a complete specification of a behavior and not, like the interaction diagrams, a single possible scenario.

The activity diagram gives a simplified representation of a process, showing control flows (called transitions) between actions performed in the system (called activities). These flows represent the internal behavior of a model element (use case, package, classifier or operation) from a start point to several potential end points.



## 6.3 PLANNING MANAGERIAL ISSUES

## 6.3.1 PLANNING SCOPE

The first activity in software project planning is the determination of software scope.

Software scope describes the data and control to be processed, function, performance, constraints, interfaces, and reliability. Functions described in the statement of scope are evaluated and in some cases refined to provide more detail prior to the beginning of estimation. Because both cost and schedule estimates are functionally oriented, some degree of decomposition is often useful. Performance considerations encompass processing and response time requirements. Constraints identify limits placed on the software by external hardware, available memory, or other existing systems.

## 6.3.2 PROJECT RESOURCES

The second software planning task is estimation of the resources required to accomplish the software development effort.

The development environment—hardware and software tools—sits at the foundation of the resources pyramid and provides the infrastructure to support the development effort. At a higher level, we encounter reusable software components—software building blocks that can dramatically reduce development costs and accelerate delivery. At the top of the pyramid is the primary resource—people.

Each resource is specified with four characteristics: description of the resource, a statement of availability, time when the resource will be required; duration of time that resource will be applied. The last two characteristics can be viewed as a time window. Availability of the resource for a specified window must be established at the earliest practical time.

* **Human Resources**

The planner begins by evaluating scope and selecting the skills required to complete development. Both organizational positions (e.g., manager, senior software engineer) and specialty (e.g., telecommunications, database, and client/server) are specified. For relatively small projects (one person-year or less), a single individual may perform all software engineering tasks, consulting with specialists as required. The number of people required for a software project can be determined only after an estimate of development effort (e.g., person-months) is made.

* **Reusable Software Resources**

Component-based software engineering (CBSE) emphasizes reusability—that is, the creation and reuse of software building blocks. Such building blocks, often called components, must be cataloged for easy reference, standardized for easy application, and validated for easy integration.

* Off-the-shelf components: Existing software that can be acquired from a third party or that has been developed internally for a past project.
* Full-experience components: Existing specifications, designs, code, or test data developed for past projects that are similar to the software to be built for the current project. Members of the current software team have had full experience in the application area represented by these components. Therefore, modifications required for full-experience components will be relatively low-risk.
* Partial-experience components: Existing specifications, designs, code, or test data developed for past projects that are related to the software to be built for the current project but will require substantial modification. Members of the current software team have only limited experience in the application area represented by these components. Therefore, modifications required for partial-experience components have a fair degree of risk.
* New components: Software components that must be built by the software team specifically for the needs of the current project.
* **Environmental Resources**

The environment that supports the software project, often called the software engineering environment (SEE), incorporates hardware and software. Hardware provides a platform that supports the tools (software) required to produce the work products that are an outcome of good software engineering practice. Because most software organizations have multiple constituencies that require access to the SEE, a project planner must prescribe the time window required for hardware and software and verify that these resources will be available.

When a computer-based system (incorporating specialized hardware and software) is to be engineered, the software team may require access to hardware elements being developed by other engineering teams. For example, software for a numerical control (NC) used on a class of machine tools may require a specific machine tool (e.g., an NC lathe) as part of the validation test step; a software project for advanced page-layout may need a digital-typesetting system at some point during development. Each hardware element must be specified by the software project planner.

## 6.3.3 TEAM ORGANIZATION:

The following options are available for applying human resources to a project that will require n people working for k years:

* 1. N individuals are assigned to M different functional tasks, relatively little combined work occurs; coordination is the responsibility of a software manager who may have six other projects to be concerned with.
  2. N individuals are assigned to m different functional tasks ( m< n) so that informal "teams" are established; an ad hoc team leader may be appointed; coordination among teams is the responsibility of a software manager.
  3. N individuals are organized into t teams; each team is assigned one or more functional tasks; each team has a specific structure that is defined for all teams working on a project; coordination is controlled by both the team and a software project manager.

Although it is possible to voice arguments for and against each of these approaches, a growing body of evidence indicates that a formal team organization (option 3) is most productive.

The “best” team structure depends on the management style of your organization, the number of people who will populate the team and their skill levels, and the overall problem difficulty.

How should a software team be organized?

Democratic decentralized (DD): This software engineering team has no permanent leader. Rather, "task coordinators are appointed for short durations and then replaced by others who may coordinate different tasks." Decisions on problems and approach are made by group consensus. Communication among team members is horizontal.

Controlled decentralized (CD): This software engineering team has a defined leader who coordinates specific tasks and secondary leaders that have responsibility for subtasks. Problem solving remains a group activity, but implementation of solutions is partitioned among subgroups by the team leader. Communication among subgroups and individuals is horizontal. Vertical communication along the control hierarchy also occurs.

Controlled Centralized (CC):Top-level problem solving and internal team coordination are managed by a team leader. Communication between the leader and team members is vertical.

Seven project factors that should be considered when planning the structure of software engineering teams:

• The difficulty of the problem to be solved.

• The size of the resultant program(s) in lines of code or function points

• The time that the team will stay together (team lifetime).

• The degree to which the problem can be modularized.

• The required quality and reliability of the system to be built.

• The rigidity of the delivery date.

• The degree of sociability (communication) required for the project.

Because a centralized structure completes tasks faster, it is the most adept at handling simple problems. Decentralized teams generate more and better solutions than individuals. Therefore such teams have a greater probability of success when working on difficult problems. Since the CD team is centralized for problem solving, either a CD or CC team structure can be successfully applied to simple problems. A DD structure is best for difficult problems.

Because the performance of a team is inversely proportional to the amount of communication that must be conducted, very large projects are best addressed by team with a CC or CD structures when subgrouping can be easily accommodated. It has been found that DD team structures result in high morale and job satisfaction and are therefore good for teams that will be together for a long time. The DD team structure is best applied to problems with relatively low modularity, because of the higher volume of communication needed. When high modularity is possible (and people can do their own thing), the CC or CD structure will work well.

It’s often better to have a few small, well-focused teams than a single large team.

CC and CD teams have been found to produce fewer defects than DD teams, but these data have much to do with the specific quality assurance activities that are applied by the team. Decentralized teams generally require more time to complete a project than a centralized structure and at the same time are best when high sociability is required.

## 

## 6.3.4 PROJECT SCHEDULING

Software project scheduling is an activity that distributes estimated effort across the planned project duration by allocating the effort to specific software engineering tasks.

It is important to note, however, that the schedule evolves over time. During early stages of project planning, a macroscopic schedule is developed. This type of schedule identifies all major software engineering activities and the product functions to which they are applied. As the project gets under way, each entry on the macroscopic schedule is refined into a detailed schedule. Here, specific software tasks (required to accomplish an activity) are identified and scheduled.

Scheduling for software engineering projects can be viewed from two rather different perspectives. In the first, an end-date for release of a computer-based system has already (and irrevocably) been established. The software organization is constrained to distribute effort within the prescribed time frame. The second view of software scheduling assumes that rough chronological bounds have been discussed but that the end-date is set by the software engineering organization. Effort is distributed to make best use of resources and an end-date is defined after careful analysis of the software. Unfortunately, the first situation is encountered far more frequently than the second.

Basic principles for software project scheduling:

* Compartmentalization: The project must be compartmentalized into a number of manageable activities and tasks. To accomplish compartmentalization, both the product and the process are decomposed
* Interdependency: The interdependency of each compartmentalized activity or task must be determined. Some tasks must occur in sequence while others can occur in parallel. Some activities cannot commence until the work product produced by another is available. Other activities can occur independently.
* Time allocation: Each task to be scheduled must be allocated some number of work units (e.g., person-days of effort). In addition, each task must be assigned a start date and a completion date that are a function of the interdependencies and whether work will be conducted on a full-time or part-time basis.
* Effort validation: Every project has a defined number of staff members. As time allocation occurs, the project manager must ensure that no more than the allocated number of people has been scheduled at any given time.
* Defined responsibilities: Every task that is scheduled should be assigned to a specific team member.
* Defined outcomes: Every task that is scheduled should have a defined outcome. For software projects, the outcome is normally a work product (e.g., the design of a module) or a part of a work product. Work products are often combined in deliverables.
* Defined milestones: Every task or group of tasks should be associated with a project milestone. A milestone is accomplished when one or more work products has been reviewed for quality and has been approved.

## 6.3.5 ESTIMATION

* The accuracy of a software project estimate is predicated on:
  + The degree to which the planner has properly estimated the size (e.g., KLOC) of the product to be built
  + The ability to translate the size estimate into human effort, calendar time, and money
  + The degree to which the project plan reflects the abilities of the software team
  + The stability of both the product requirements and the environment that supports the software engineering effort

PROJECT ESTIMATION OPTIONS

* Options for achieving reliable cost and effort estimates
  + Delay estimation until late in the project (we should be able to achieve 100% accurate estimates after the project is complete)
  + Base estimates on similar projects that have already been completed
  + Use relatively simple decomposition techniques to generate project cost and effort estimates
  + Use one or more empirical estimation models for software cost and effort estimation.
* Option #1 is not practical, but results in good numbers.
* Option #2 can work reasonably well, but it also relies on other project influences being roughly equivalent
* Options #3 and #4 can be done in tandem to cross check each other.

PROJECT ESTIMATION APPROACHES

* Decomposition techniques
  + These take a "divide and conquer" approach
  + Cost and effort estimation are performed in a stepwise fashion by breaking down a project into major functions and related software engineering activities
* Empirical estimation models
  + Offer a potentially valuable estimation approach if the historical data used to seed the estimate is good

​

## 6.3.6 RISK ANALYSIS

Risk analysis and management are a series of steps that help a software team to understand and manage uncertainty. Many problems can plague a software project. There is general agreement that risk always involves two characteristics

Uncertainty—the risk may or may not happen; that is, there are no 100% probable risks.

Loss—if the risk becomes a reality, unwanted consequences or losses will occur.

When risks are analyzed, it is important to quantify the level of uncertainty and the degree of loss associated with each risk. To accomplish this, different categories of risks are considered.

* Project risks: if project risks become real, it is likely that project schedule will slip and that costs will increase. Project risks identify potential budgetary, schedule, personnel (staffing and organization), resource, customer, and requirements problems and their impact on a software project.
* Technical risks: It threatens the quality and timeliness of the software to be produced. If a technical risk becomes a reality, implementation may become difficult or impossible. Technical risks identify potential design, implementation, interface, verification, and maintenance problems. Technical risks occur because the problem is harder to solve than we thought it would be.
* Business risks: It threatens the viability of the software to be built. Business risks often jeopardize the project or the product
* Known risks are those that can be uncovered after careful evaluation of the project plan, the business and technical environment in which the project is being developed, and other reliable information sources (e.g., unrealistic delivery date, lack of documented requirements or software scope, poor development environment).
* Predictable risks are extrapolated from past project experience (e.g., staff turnover, poor communication with the customer, dilution of staff effort as ongoing maintenance requests are serviced).
* Unpredictable risks are the joker in the deck. They can and do occur, but they are extremely difficult to identify in advance.

There are a few well-known types of risk analysis that can be used [21]. In software engineering, risk analysis is used to identify the high-risk elements of a project. It provides ways of documenting the impact of risk mitigation strategies. Risk analysis has also been shown to be important in the software design phase to evaluate criticality of the system, where risks are analyzed and necessary countermeasures are introduced . The purpose of risk analysis is to understand risk better and to verify and correct attributes. A successful analysis includes essential elements like problem definition, problem formulation, data collection.

Risk Tree Analysis and Assessment Method:

In risk tree analysis method, software risks are classified at first. Then risks are identified in each group. Afterwards, primary or basic risk events, intermediate events, top event, and the necessary sub-tree are found. All these require that managers have a complete knowledge about the projects. Then the risk tree can be constructed. Likelihood and impact must be assigned to each event and failure. Then probabilities starting from primary events to the top event are calculated. The events are ordered according to their probabilities. Maximum probability indicates the importance of those events; therefore, it is necessary to attend more to them. Managers should use solutions to prevent risks from occurring or reduce undesirable incidents.

The presented classifications and risk tree structures can apply with some software tools. Fault Tree Creation and Analysis Program, Fault Tree Tool or Relax Fault Tree can be used for this analysis. These tools have facilities that help users to create tree symbols and construct the risk tree structures.

## 

## 6.3.7 SECURITY PLAN

Security plan include these steps:

1. Identify the assets you want to protect and the value of these assets.
2. Identify the risks to each asset.
3. Determine the category of the cause of the risk (natural disaster risk, intentional risk, or unintentional risk).
4. Identify the methods, tools, or techniques the threats use.

After assessing your risk, the next step is proactive planning. Proactive planning involves developing security policies and controls and implementing tools and techniques to aid in security.

 The various types of policies that could be included are:

* Password policies
  + Administrative Responsibilities
  + User Responsibilities
* E-mail policies
* Internet policies
* Backup and restore policies

## 6.3.8 CONFIGURATION MANAGEMENT PLAN:

## 6.4 DESIGN

## 6.4.1. DESIGN CONCEPT

The purpose of the design phase is to plan a solution of the problem specified by the requirement of the problem specified by the requirement document. This phase is the first step in moving from the problem domain to the solution domain. In other words, starting with what is needed, design takes us towards how to satisfy the needs. The design of system is the most critical factor affecting the quality of the software and has major impact on testing and maintenance. The output of this phase is the design document.

## 6.4.2. DESIGN TECHNIQUE

**System Design**

System design provides the understandings and procedural details necessary for implementing the system recommended in the system study. Emphasis is on the translating the performance requirements into design specifications. The design phase is a transition from a user-oriented document (System proposal) to a document oriented to the programmers or database personnel.

System Design goes through two phases of development:

* Logical design
* Physical Design

A data flow diagram shows the logical flow of the system. For a system it describes the input (source), output (destination), database (data stores) and procedures (data flows) all in a format that meets the user’s requirement. When analysis prepares the logical system design, they specify the user needs at a level of detail that virtually determines the information flow into an out of the system and the required data resources. The logical design also specifies input forms and screen layouts.

The activities following logical design are the procedure followed in the physical design e.g., producing programs, software, file and a working system.

The logical design of an information system is analogous to an engineering blue print of an automobile. It shows the major features and how they are related to one another. The detailed specification for the new system was drawn on the basis of user’s requirement data. The outputs inputs and databases are designed in this phase. Output design is one of the most important features of the information system. When the output is not of good quality the user will be averse to use the newly designed system and may not use the system. There are many types of output, all of which can be either highly useful or can be critical to the users, depending on the manner and degree to which they are used. Outputs from computer system are required primarily to communicate the results of processing to users, They are also used to provide a permanent hard copy of these results for later consultation. Various types of outputs required can be listed as below:

* External Outputs, whose destination is outside the organization
* Internal outputs, whose destination is with the organization
* Operational outputs, whose use is purely with in the computer department e.g., program-listing etc.
* Interactive outputs, which involve the user is communicating directly with the computer, it is particularly important to consider human factor when designing computer outputs.

End user must find outputs easy to use and useful to their jobs, without quality output, user may find the entire system unnecessary and avoid using it. The term “Output” in any information system may apply to either printer or displayed information. During the designing the output for this system, it was taken into consideration, whether the information to be presented in the form of query of report or to create documents etc.

Other important factors that were taken into consideration are:

* The End user, who will use the output.
* The actual usage of the planned information
* The information that is necessary for presentation when and how often output and their format is needed. While designing output for project based Attendance Compilation System, the following aspects of outputs designing were taken into consideration.

**Detailed Design**

During detailed design the internal logic of each of the modules specified in the system design is decided. In system design the focus is on identifying the modules, where as during detailed design the focus is on designing the logic for each of modules. In other words, in system design the attention is on what components are needed, while in detailed design how the components can be implemented in the software. During this phase further details of the data structures and algorithmic design of each of the module is usually specified in a high – level design description language, which is independent of the target language in which the software will eventually be implemented. Thus a design methodology is a systematic approach to creating a design by application of a set of techniques and guidelines.

# 6.4.3. MODELING

## 6.4.3.1. DETAILED CLASS DIAGRAM:

The class diagram describes the attributes and operations of a class and also the constraints imposed on the system. The class diagrams are widely used in the modelling of object oriented systems because they are the only UML diagrams which can be mapped directly with object oriented languages. The class diagram shows a collection of classes, interfaces, associations, collaborations and constraints. It is also known as a structural diagram. The purpose of the class diagram is to model the static view of an application.

The following points should be remembered while drawing a class diagram:

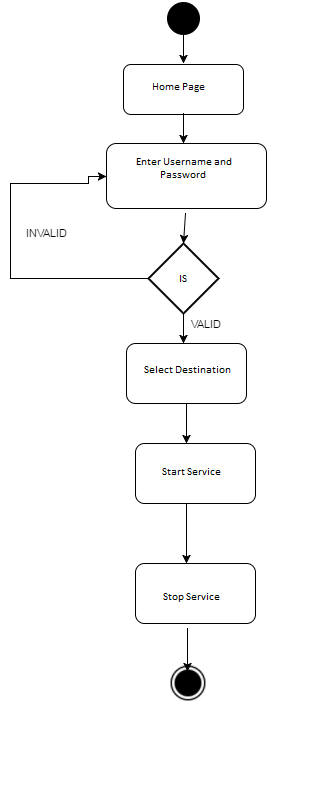
* The name of the class diagram should be meaningful to describe the aspect of the system.
* Each element and their relationships should be identified in advance.
* Responsibility (attributes and methods) of each class should be clearly identified.
* For each class minimum number of properties should be specified. Because unnecessary properties will make the diagram complicated.
* Use notes whenever required to describe some aspect of the diagram. Because at the end of the drawing it should be understandable to the developer/coder.
* Finally, before making the final version, the diagram should be drawn on plain paper and rework as many times as possible to make it correct.

**6.4.3.2.1. SEQUENCE DIAGRAM**

## 6.4.3.4. ACTIVITY DIAGRAM:

Activity diagrams illustrate the dynamic nature of a system by modelling the flow of control from activity to activity. An activity represents an operation on some class in the system that results in a change in the state of the system. Typically, activity diagrams are used to model workflow or business processes and internal operation.

Activity diagrams are used to model workflow or business processes and internal operation. The figure shows the work flow the system. The client logs in, gives the requirements, Admin views the requirements, contacts the Dealers and then finally generates the final product.



## 6.4.3.6. DEPLOYMENT DIAGRAM:

## 6.4.3.7. COMPONENT DIAGRAM:

## 6.5 IMPLEMENTATION PHASE:

## 6.5.1. LANGUAGE USED CHARACTERISTICS:

ANDROID

Android is a powerful operating system competing with Apple 4GS and supports great features. Few of them are listed below:

* Android OS basic screen provides a beautiful and intuitive user interface.
* Simple and powerful RAD tool for developing native Android applications
* Complete IDE and programming language 100% focused on Android development
* Compiles to native bytecode. No runtime libraries are required. APK files are exactly the same as APK files created with Java / Eclipse
* Performance is similar to applications written with Java
* Object oriented programming language
* Share the code with B4J - a development tool for desktop applications New!
* No need for XML programming
* Rapid debugger - supports quick deployments, hot code swapping and expressions watches New! No other native Android tool supports these features!
* Highly extensible with support for custom Java libraries
* WYSIWYG visual editor for Android. The visual editor supports multiple screens and resolutions
* Powerful designer scripts feature. Let’s you easily create sophisticated layouts
* Basic4android UI Cloud service. Test your layouts on a cloud of real phones and tablets
* Supports all Android phones and tablets from Android 1.6 and up to Android 4.x
* Modern IDE with autocomplete, built-in documentation, internal index and other advanced features
* Powerful step-by-step debugger
* Large set of documentation
* Built-in code obfuscation.

Servlets:

* Servlets dont have graphical user interface.
* Servlets call Enterprise Java Beans to perform business logic functions.
* Servlets call Java Server Pages to perform page layout functions.
* Servlets can also be considered as url.
* The buttons or any other keys to be pressed(imagine) on your application will always be pointing to servlets.
* Servlets call other servlets at times when there will case of dependencies for the task to be accomplished.
* While the server is running, Server's can be dynamically reloaded.

JavaScript:

JavaScript resembles Java and supports most of Java’s expression syntax. Some other characteristics of JavaScript are:

* JavaScript is interpreted, not compiled.
* JavaScript is case sensitive.
* Single-line comments start with a double-slash (//).
* Multi-line comments begin with a slash-asterisk (/\*) and end with an asterisk-slash (\*/).
* JavaScript uses the same rules as C where variable name can use letters, digits, or underscores.
* Variable can change type dynamically.
* JavaScript can perform actions on various objects in an HTML document, such as frames, buttons, links, and other objects.

XML:

XML stands for Extensible Markup Language. It is a software and hardware-independent tool for storing and transporting data.

* XML does not use predefined tags. With XML, the author must define both the tags and the document structure.
* XML is extensible. Most XML applications will work as expected even if new data is added (or removed). The way XML is constructed, older version of the application can still work.
* XML simplifies data sharing, data transport, platform changed and data availability.
* XML is a W3C recommendation.

## 6.5.2. CODING:

## 6.5.3. CODE EFFICIENCY:

Code efficiency is an important aspect for smooth working of code. There are various factors which focus on code efficiency. Basically we need to answer three questions for estimation code efficiency:

* Have functions been optimized for speed?
* Have repeatedly used block of code been formed into subroutines?
* Are there memory leaks or overflow errors?

For optimization of speed we have not imported the complete library rather we have imported individual classes of that library for fulfilling our requirements. For sort, we haven’t used math library for arithmetic calculations. Also we have taken a low quality of icons and .png images so that our project can run smoothly.

There were many occasions where we need to use same piece of code repeatedly(like JSONParser.class file). But instead of using the same code again and again, we used its object wherever required.

For preventing overflow errors and memory leaks, we have restricted the user to enter data of specific size. In database , the size of each datatype is predefined. This will not let memory leaks and overflow errors to occur.

## 6.5.4. OPTIMIZATION OF CODE:

Code optimization is one of the most important aspect for efficiency measurement. Optimization of code is defined as how efficiently a code can run with fewest possible resources. Here are some of the optimization practices that we have involved in our project:

* Avoid\_constant\_expressions\_in\_loops
* Avoid\_duplication\_of\_code
* Do\_not\_declare\_members\_accessed\_by\_inner\_class\_private
* Avoid\_synchronized\_modifier\_in\_method
* Avoid\_empty\_if
* Avoid\_unnecessary\_if
* Avoid\_unnecessary\_parentheses
* Avoid\_unnecessary\_implementing\_Clonable\_interface
* Remove\_unnecessary\_if\_then\_else\_statement
* Avoid\_instantiation\_of\_class\_with\_only\_static\_members
* Close\_jdbc\_connections
* Avoid\_boolean\_array
* Avoid\_string\_concatenation\_in\_loop
* Place\_try\_catch\_out\_of\_loop
* Avoid\_empty\_try\_blocks
* Avoid\_empty\_loops
* Avoid\_unnecessary\_substring
* Avoid\_unnecessary\_exception\_throwing
* Use\_PreparedStatement\_instead\_of\_Statement
* Avoid\_Extending\_java\_lang\_Object
* Avoid\_empty\_catch\_blocks
* Avoid\_synchronized\_methods\_in\_loop
* Avoid\_synchronized\_blocks\_in\_loop

## 6.5.5. VALIDATION CHECK:

**Login:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Sr. no. | Input values | Test cases | Condition Being checked | Result |
| 1 | User id | Empty | Please enter valid email id | Successful |
| 2 | Password | Empty | Password cannot be empty | Successful |

## 6.6TESTING:

Testing is the major quality control that can be used during software development. Its basic function is to detect the errors in the software. During requirement analysis and design, the output is a document that is usually textual and non-executable. After the coding phase, computer program is available that can be executed for testing purposes. This implies that testing not only has to uncover errors introduced during coding, but also errors introduced during previous phases. Thus the goal of the testing is to uncover requirement, design and coding errors in the program.

An elaborate testing of data is prepared and the system is tested using that test date. Errors noted and corrections made during the testing. The corrections are also noted for future use. The users are trained to operate the developed system. Both hardware and software securities are made to run the developed system successfully in future. System testing is the stage of implementation, which is aimed at ensuring that the system works accurately before live operation commences. Testing is vital to the success of any system. System testing makes a logical assumption that if all the parts of the system are correct, the goal will be successfully achieved.

## 6.6.1 TESTING OBJECTIVES

* 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 undiscovered error
* A successful test is one that uncovers an as-yet undiscovered error

**Testing Principles**

* All tests should be traceable to customer requirements
* Tests should be planned long before testing begins
* Testing should begin “in the small” and progress toward testing “in the large”
* Exhaustive testing is not completely possible
* To be most effective, testing should be conducted by an independent third party

## 6.6.2TESTING METHODS

Software Testing Strategies

A strategy for software testing integrates software test case design methods into a well-planned series of steps that result in the successful construction of software. As important, a software testing strategy provides a road map. Testing is a set of activities that can be planned in advance and conducted systematically.

Various strategies are given below:

* Unit Testing
* Integration Testing
* Validation Testing
* User Acceptance Testing
* System Testing

Unit Testing

Unit testing focuses verification efforts on the smallest unit of software design of module. This is also known as “Module Testing”. Acceptance of package is used for computerization of module. Machine Utilization was prepared and approved by the project leader.

In this testing step, each module is found to be working satisfactory as regards to the expected output from the module. The suggested changes were incorporated into the system. Here each module in the Machine Utilization has been tested.

Integration Testing

After the package is integrated, the user test version of the software was released. This testing consists of testing with live data and various stress tests and result were noted down. Then the corrections were made based on the users feedback. Integration testing is systematic testing for constructing the program structure, while at the same time conducting tests to uncover errors associated within the interface. The objective is to take unit tested modules and build a program structure. All the modules are combined and tested as a whole. Here correction is difficult because the vast expenses of the entire program complicate the isolation of causes. Thus the integration testing step, all the errors uncovered are corrected for the next steps.

Validation Testing

At the culmination of integration testing, software is completely assembled as a package; interfacing errors have been uncovered and corrected, and a final series of software tests - Validation testing - may begin.

User Acceptance Testing

User acceptance of a system is the key factor for the success of any system. The system under consideration is tested for user acceptance by constantly keeping in touch with prospective system users at time of development and making changes wherever required.

This is done in regard to the following points:

* Input Screen Design
* On-line Messages to guide the user
* Format of reports and other outputs

After performing all the above tests the system was found to be running successfully according to the user requirements i.e., (constraints).

System Testing

Software is only one element of a larger computer-based system.

Ultimately, software is incorporated with other system elements and a series of system integration and validation tests are conducted. The various types of system testing are:

* Recovery Testing: Many computer-based systems must recover from faults and resume processing within a pre specified time.
* Security Testing: Security testing attempts to verify that protection mechanisms built into a system will in fact protect it from improper penetration.
* Stress Testing: Stress tests are designed to confront programs with abnormal situations.
* Performance Testing: Performance testing is designed to test run-time performance of software within the context of an integrated system.

Black Box Testing

Black box testing is carried out to check the functionality of the various modules. Although they are designed to uncover errors, black-box tests are used to demonstrate that software functions are operational; that input is properly accepted and output is correctly produced; and that the integrity of external information is maintained. A black-box test examines some fundamental aspect of the system with little regard for the internal logical structure of the software.

White Box Testing

White-box testing of software is predicated on close examination of procedural detail providing the test cases that exercise specific sets of conditions and, loops tests logical paths through the software. White-box testing, sometimes called glass-box testing is a test case design method that uses the control structure of the procedural design to derive test cases. Using white-box testing methods, following test cases can be derived.

* Guarantee that all independent paths within a module have been exercised at least once.
* Exercise all logical decisions on their true and false sides.
* Execute all loops at their boundaries and within their operational bounds.
* Exercise internal data structures to assure their validity.
* The errors that can be encountered while conducting white-box testing are Logic errors and incorrect assumptions.
* Typographical errors

**CHAPTER 7**

**Conclusion & Discussion**

# CONCLUSION:

We have been given the problem of automating the material of store in a collage. Earlier the materials of store have not been automated. As the result of this automation, manual work load is reduced and data retrieval becomes easy .This project can be helpful for centralization of information regarding the stores of the collage.

## 7.1 LIMITATIONS OF PROJECT:

* Requires WiFi connectivity.

## 7.2 DIFFICULTIES ENCOUNTERED:

* Poor requirements-if the requirements are not clear, unfinished, too common, and not testable, then there will be problems.
* Unrealistic Schedule-if too much work is given in too little time, problems are inevitable.
* Inadequate testing-no one will know whether or not the program is any good until the customer complain or system collide.
* Futurities-request to pile on new features after development is underway; extremely common.
* Miscommunication-if developers do not know what’s needed or customers have wrong expectations, problem are assured

## 7.3 FUTURE ENHANCEMENT SUGGESTIONS:

The adoption of RFID and other similar technologies is spurring innovation and the development of internet of things.Ip networks are common place throughout households,offices,warehouses,parks and many other places.Industry and government mandates are regulating technologies leading to accepted standards across industries allowing for interoperability among devices.sAdditionally ,the cost and size of devices continues

**CHAPTER 8**

**Bibliography & References**

# BIBLOGRAPHY AND REFERENCES:

**Books:**

1. Jiangsu Engineering Center of Network Monitoring, Nanjing University of Information Science & Technology, Nanjing 210044
2. School of Computer & Software, Nanjing University of Information Science & Technology, Nanjing 210044
3. Key Laboratory of Meteorological Disaster of Ministry of Education Nanjing University of Information Science & Technology, Nanjing 210044
4. School Of engineering and Science, Lovely Professional University 2, G.T. Road, Near Chehru Railway Bridge, Phagwara (Punjab)-144401, India
5. Mona Institute of Applied Science, University of West Indies, Kingston, Jamaica salboglepro gmer@gmail.com 2Computing & Information Systems, Institute Technology Brunei, Brunei 3Department of Computing, University of West Indies, Kingston, Jamaica
6. Review on Android and Smartphone Security Tiwari Mohini, Srivastava Ashish Kumar and Gupta Nitesh NRI Institute of Information Science and Technology, Bhopal,Madhya Pradesh, INDIA
7. International Journal of Engineering Technology & Management Research Journal homepage: www.ijetmr.org

**Websites:**

1.  “Programming Android Java Programming for the New Generation of Mobile Devices” by Zigurd Mennieks , Laird Dornin , G. Blake Meike ,& Mausmi Nakamura

2. “Hello, Android Introducing Google’s Mobile Development Platform” by Ed Burnette

3.  “Learning Android Building Applications for the Android Market” by Marko Gargenta

4.**[The Busy Coder's Guideto Android Development](http://www.e-booksdirectory.com/details.php?ebook=7885) by Mark L. Murphy - CommonsWare , 2009**

**5.** <http://www.appdevelopmentcompany.com.au/android-application-development>**.**

**6.**  <http://androidgifts.com/5-tools-every-android-developer-must-use/>﻿

7. :<http://groups.google.com/group/android-developers?lnk=>

8. Android Application Development (O'Reilly)

9. Android Programming: The Big Nerd Ranch Guide at [Big Nerd Ranch](http://www.bignerdranch.com/book/android_the_big_nerd_ranch_guide)

10. *Beginning Android Tablet Application Development* - Wei-Meng Lee.

11.http://androidhive.com