An internship report submitted in partial fulfilment of the requirements for the award of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE

prepared by

CH. MOHAN TEJA (315126510194)

B. VINAY (315126510189)

M.G.K. RAJU (315126510211)

at

**VISAKHAPATNAM PORT TRUST**

Submitted to



Department of Computer Science Engineering

ANIL NEERUKONDA INSTITUTE OF TECHNOLOGY & SCIENCES

Autonomous status accorded by UGC and Andhra University

Approved by AICTE, Permanently Affiliated to Andhra University,

Accredited and reaccredited by NBA& accredited by NAAC

Sangivalasa-531162, Bheemunipatnam Mandal, Visakhapatnam District.

Phone: 08933- 225084,22639

**CERTIFICATE**

This is to certify that Mr. B.Vinay (3151265610189), Mr. Ch. Mohan Teja (315126510194), Mr.M.G.K Raju (315126510211),bonafide Students of B.Tech(CSE) from **Anil Neerukonda Institute of Technology and Sciences, Sangivalasa** for the Session may 1st to may 21th as obtained my guidance and supervision from time to time training project entitled “**MACHINERY MONITORING SYSTEM**” of Visakhapatnam Port Trust and have been submitted to the esteemed institute in partial fulfilment of the requirement for the award of B.TECH.

Mr.K.RAJENDRA KUMAR Mr.G.KISHORE

JOINT DIRECTOR ASSISTANT DIRECTOR

I.T. Centre I.T. Centre

Visakhapatnam Port Trust Visakhapatnam Port Trust

APPLICATION WORKING:

The application MACHINERY MONITORING SYSTEM provides data to the workers working in different shifts, about the availability of machinery, their working condition, type of machinery available, their quantity available etc..,. The data at real time is uploaded by shift in-charge PO’s either through altering database through the main website or editing feature which is provided in the app itself through further development of this app. We provided necessary interfaces to the design of code so that it can be helpful for future modifications of the app, for now the app only shows the data available in the database.

The working of this application is pretty simple and it is very parallelized using AsyncTask threading feature available in library. It is resource-less, consumable, making it run on old android devices with low hardware specs. The app in main activity, on clicking any button of available seven options, directs to their respective activity where the data is downloaded by the GetRawData class and then the data is converted from json format to their respective objects by GetJson which then is passed to a RecyclerView for displaying.

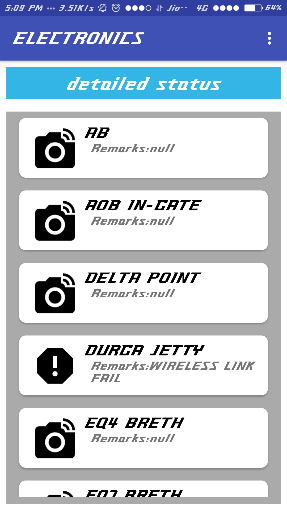
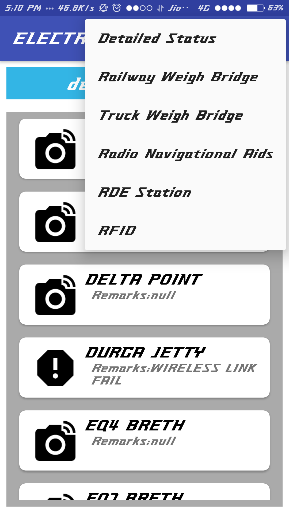
We also used the recyclerview so that the views created can be reused without creating any new views, this reduces the memory consumption.

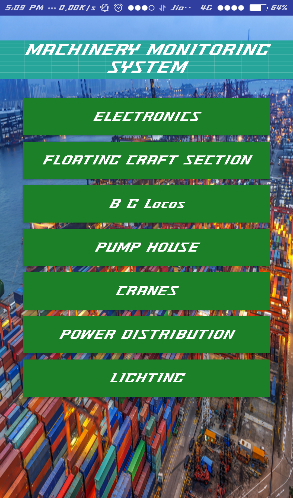
The json object array is output by php script available in webhosting service where the database also resides. The php script on request echoes a json array of the table requested.

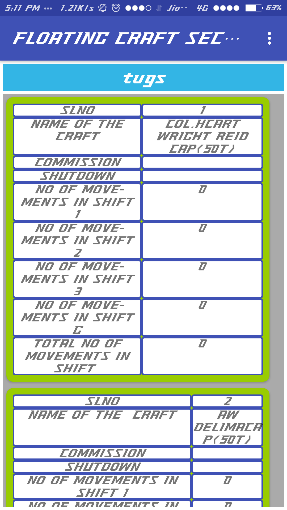
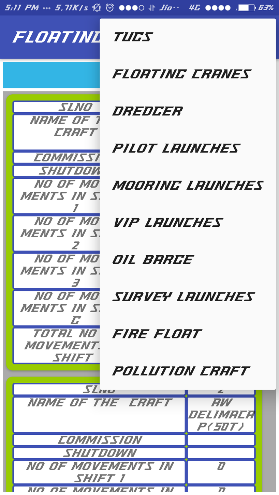
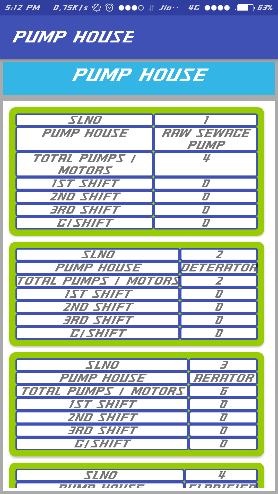
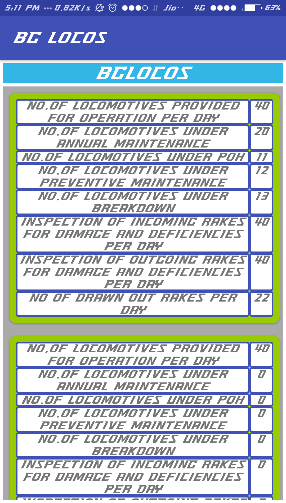
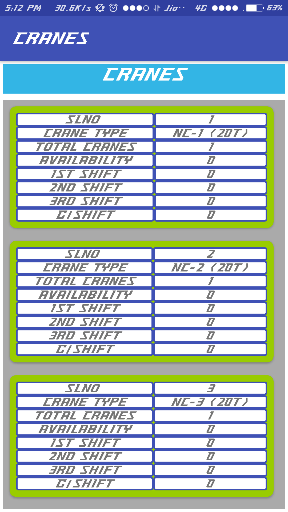
GetRawData: this is called through GetJson This takes php script usl as parameter and downloads the json data upon download it invoke download complete interface which then invokes the GetJson class using call-back interface we implemented.

GetJson: upon receiving data from the GetRawData it converts the json data to respective item arrays i.e., for example for electronics section we display cameras in the list so GetJson creates a list of cameras objects, then it passes it to the section activity requested.

SectionActivity: the section activity sends the object array to RecyclerView through a Viewholder for displaying the items.

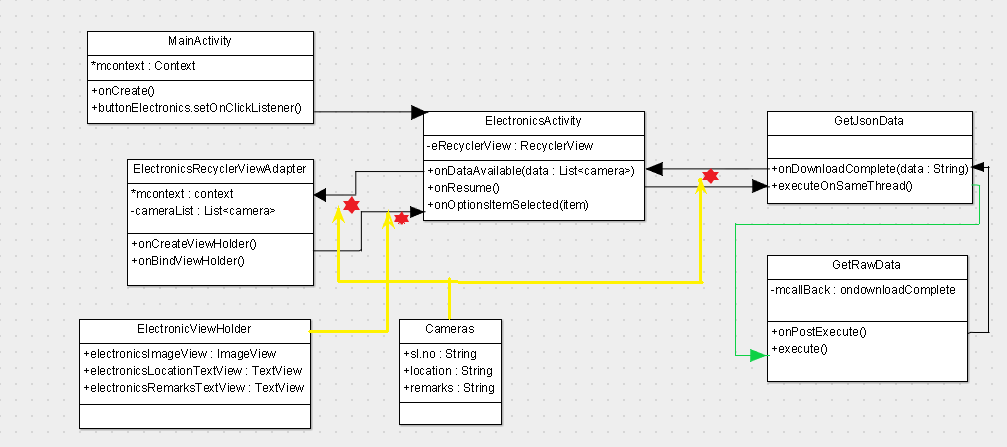




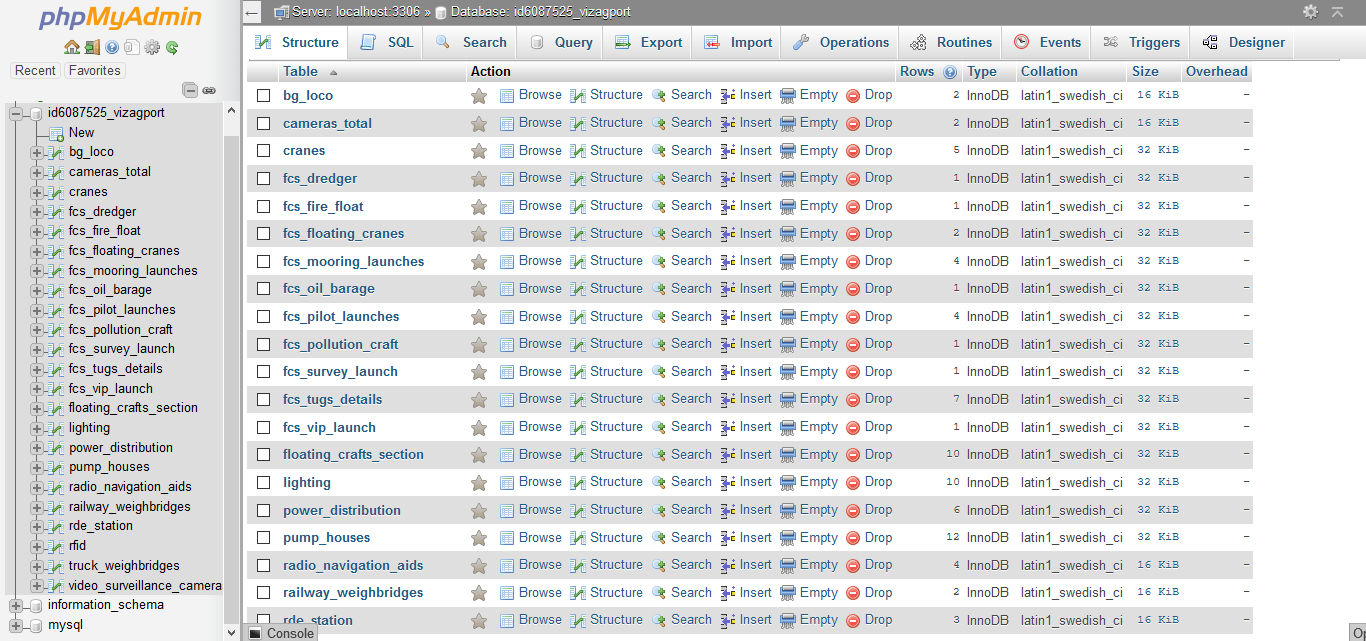


TOTAL APPLICATION LAYOUT

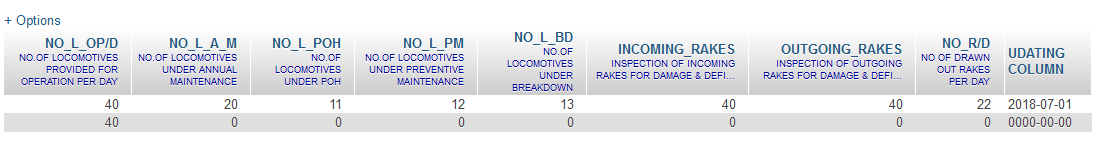
**CLASS DIAGRAM OF MACHINERY MONITERING SYSTEM:**



**DATABASE:**

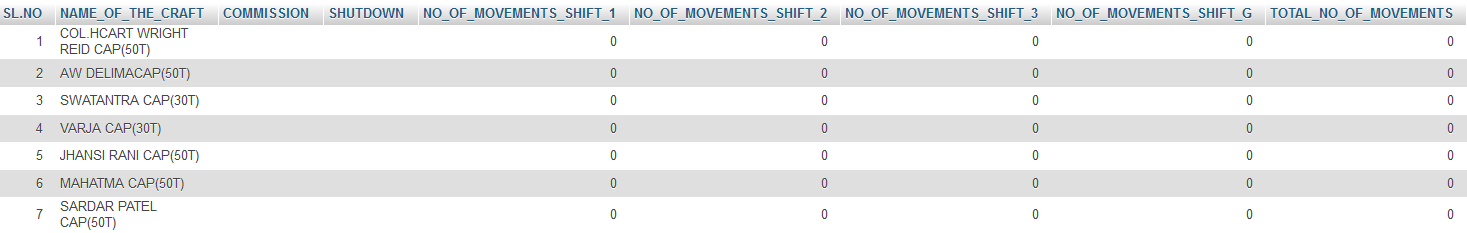


**TABLES**:

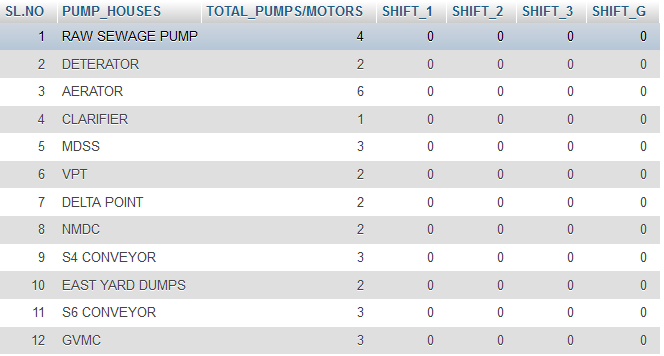


*Table 1 :BG LOCOS*

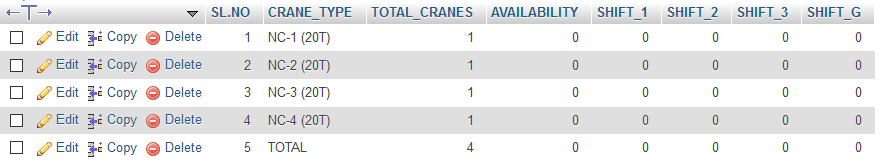
*Table 2: FCS*



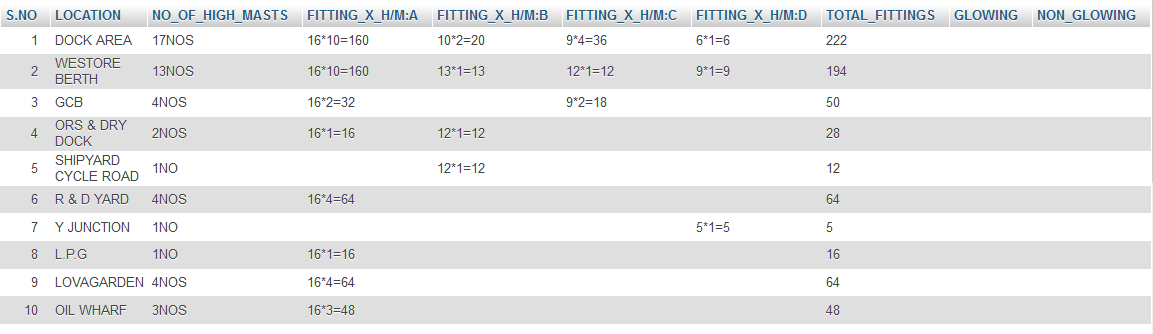
*Table 3:PUMP HOUSE*



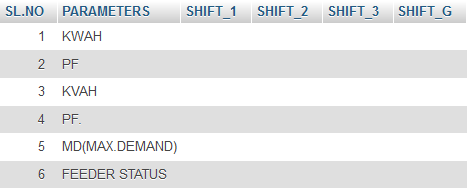
*Table 4:CRANES*



*Table 5:LIGHTING*



*Table 6:POWER HOUSE*



**CODE**

Gradle:-

<?xml version="1.0" encoding="UTF-8"?>

<project version="4">

<component name="GradleSettings">

<option name="linkedExternalProjectsSettings">

<GradleProjectSettings>

<option name="distributionType" value="DEFAULT\_WRAPPED" />

<option name="externalProjectPath" value="$PROJECT\_DIR$" />

<option name="modules">

<set>

<option value="$PROJECT\_DIR$" />

<option value="$PROJECT\_DIR$/app" />

</set>

</option>

<option name="resolveModulePerSourceSet" value="false" />

</GradleProjectSettings>

</option>

</component>

</project>

**JAVA CODE:-**

Main Activity:-

package com.example.mohanteja.vizagporttrust;

import android.app.Activity;

import android.content.Intent;

import android.support.v7.app.AppCompatActivity;

import android.os.Bundle;

import android.view.View;

import android.widget.Button;

public class MainActivity extends AppCompatActivity {

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

Button btnElectronics= (Button) findViewById(R.id.btnElectronics);

Button btnCranes= (Button) findViewById(R.id.btnCranes);

Button btnFloatingCraftSection= (Button) findViewById(R.id.btnFloatingCraftsSection);

Button btnBGLocos= (Button) findViewById(R.id.btnBGLocos);

Button btnPowerDistribution= (Button) findViewById(R.id.btnPowerDistribution);

Button btnPumpHouse= (Button) findViewById(R.id.btnPumpHouse);

Button btnLighting= (Button) findViewById(R.id.btnlighting);

btnElectronics.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

Intent intent =new Intent(getApplicationContext(),ElectronicsActivity.class);

startActivity(intent);

}

});

btnBGLocos.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

Intent intent =new Intent(getApplicationContext(),BGLocosActivity.class);

startActivity(intent);

}

});

btnCranes.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

Intent intent =new Intent(getApplicationContext(),CraneActivity.class);

startActivity(intent);

}

});

btnFloatingCraftSection.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

Intent intent =new Intent(getApplicationContext(),FCSActivity.class);

startActivity(intent);

}

});

btnLighting.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

Intent intent =new Intent(getApplicationContext(),LightingActivity.class);

startActivity(intent);

}

});

btnPowerDistribution.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

Intent intent =new Intent(getApplicationContext(),PowerHouseActivity.class);

startActivity(intent);

}

});

btnPumpHouse.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

Intent intent =new Intent(getApplicationContext(),PumpHouseActivity.class);

startActivity(intent);

}

});

}

}

Cameras:-

package com.example.mohanteja.vizagporttrust;  
public class Cameras {  
 private String location;  
 private String status;  
 private String remarks;  
 public Cameras(String location, String status, String remarks) {  
 this.location = location;  
 this.status = status;   
 this.remarks = remarks;  
 }  
 public String getLocation() {  
 return location;  
 }  
 public String getStatus() {  
 return status;

}  
 public String getRemarks() {  
 return remarks;  
 }

@Override  
 public String toString() {  
 return "Cameras{" +  
 "location='" + location + '\'' +  
 ", status='" + status + '\'' +  
 ", remarks='" + remarks + '\'' +  
 '}';  
 }  
}

Electronic Activity:-

package com.example.mohanteja.vizagporttrust;  
import android.content.Intent;  
import android.content.SharedPreferences;  
import android.preference.PreferenceManager;  
import android.support.v7.app.AppCompatActivity;  
import android.os.Bundle;  
import android.support.v7.widget.DefaultItemAnimator;  
import android.support.v7.widget.LinearLayoutManager;  
import android.support.v7.widget.RecyclerView;  
import android.util.Log;  
import android.view.Menu;  
import android.view.MenuItem;  
import android.view.View;  
import android.widget.TextView;  
import android.widget.Toast;  
import java.util.ArrayList;  
import java.util.List;  
public class ElectronicsActivity extends AppCompatActivity{  
 private static final String TAG = "ElectronicsActivity";  
 private List<Cameras> mPhotoList = new ArrayList<Cameras>();|  
 private RecyclerView mRecyclerView;  
 private ElectronicsAdapter electronicsAdapter;  
 private TextView menuOpt ;  
 public String jsonPage ="video\_surveillance\_cameras.php"; //default page  
 private int pageIndex =2;  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {|  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.activity\_electronics);  
 menuOpt = findViewById(R.id.electronicsMnuOpt);  
 menuOpt.setText("detailed status");  
// activateToolbar();  
 mRecyclerView = (RecyclerView) findViewById(R.id.electronicsRecyclerView1);  
 mRecyclerView.setLayoutManager(new LinearLayoutManager(this));  
 electronicsAdapter = new ElectronicsAdapter(ElectronicsActivity.this,  
 new ArrayList<Cameras>());  
 mRecyclerView.setAdapter(electronicsAdapter);  
 RecyclerView.ItemAnimator itemAnimator = new DefaultItemAnimator();  
 itemAnimator.setAddDuration(1000);  
 itemAnimator.setRemoveDuration(1000);  
 mRecyclerView.setItemAnimator(itemAnimator);  
 }  
 @Override  
 public boolean onCreateOptionsMenu(Menu menu) {  
 getMenuInflater().inflate(R.menu.electronic\_menu, menu);  
 return super.onCreateOptionsMenu(menu);

}  
 @Override  
 public boolean onOptionsItemSelected(MenuItem item) {  
 int id= item.getItemId();  
 switch (id){  
 case R.id.detailedStatusMenuItem:  
 jsonPage="video\_surveillance\_cameras.php";  
 pageIndex=2;  
 menuOpt.setText("detailed status");  
 onResume();  
 break;  
 case R.id.railwayMenuItem:  
 jsonPage="railway\_weighbridges.php";  
 pageIndex=3;  
 menuOpt.setText("railway weighbridge");  
 onResume();  
 break;  
 case R.id.truckMenuItem:  
 jsonPage="truck\_weighbridges.php";  
 pageIndex=4;  
 menuOpt.setText("truck\_weighbridges");  
 onResume();  
 break;  
 case R.id.radioNavigationMenuItem:  
 jsonPage="radio\_navigation\_aids.php";  
 menuOpt.setText("radio\_navigation\_aids");  
 pageIndex=5;  
 onResume();  
 break;  
 case R.id.rdeStationMenuItem:  
 jsonPage="rde\_station.php";  
 pageIndex=6;  
 menuOpt.setText("rde\_station");  
 onResume();  
 break;  
 case R.id.rfidMenuItem:  
 jsonPage="rfid.php";  
 pageIndex=7;  
 menuOpt.setText("rfid");  
 onResume();  
 break;  
 default:  
 jsonPage="video\_surveillance\_cameras.php";  
 pageIndex=2;  
 menuOpt.setText("detailed status");  
 onResume();  
 }  
 return super.onOptionsItemSelected(item);  
 }  
 @Override  
 protected void onResume() {  
 super.onResume();  
 ProcessPhotos processPhotos = new ProcessPhotos(jsonPage, pageIndex);  
 processPhotos.execute();  
 }  
 private String getSavedPreferenceData(String key) {  
 SharedPreferences sharedPref = PreferenceManager.getDefaultSharedPreferences(getApplicationContext());  
 return sharedPref.getString(key, "");

}  
 public class ProcessPhotos extends GetElectronicsJson  
 public ProcessPhotos(String searchCriteria, int pageIndex) {  
 super(searchCriteria, pageIndex);

}  
 public void execute() {  
 super.execute();  
 ProcessData processData = new ProcessData();  
 processData.execute();  
 }

public class ProcessData extends DownloadJsonData {  
 protected void onPostExecute(String webData) {  
 super.onPostExecute(webData);  
 electronicsAdapter.loadNewData(getmCameras());  
 }  
 }  
 }  
}

GetRawData.java:-  
package com.example.mohanteja.vizagporttrust;  
import android.os.AsyncTask;  
import android.util.Log;  
import java.io.BufferedReader;  
import java.io.IOException;  
import java.io.InputStream;  
import java.io.InputStreamReader;  
import java.net.HttpURLConnection;  
import java.net.MalformedURLException;  
import java.net.URL;  
enum DownloadStatus { IDLE, PROCESSING, NOT\_INITIALISED, FAILED\_OR\_EMPTY, OK }  
public class GetRawData {  
 private String LOG\_TAG = GetRawData.class.getSimpleName();  
 private String mRawUrl;  
 private String mData;  
 private DownloadStatus mDownloadStatus;  
 public GetRawData(String mRawUrl) {  
 this.mRawUrl = mRawUrl;  
 this.mDownloadStatus = DownloadStatus.IDLE;  
 }  
 public void reset() {  
 this.mDownloadStatus = DownloadStatus.IDLE;  
 this.mRawUrl = null;  
 this.mData = null;  
 }  
 public void setmRawUrl(String mRawUrl) {  
 this.mRawUrl = mRawUrl;  
 }  
  
 public String getmData() {  
 return mData;  
 }  
 public DownloadStatus getmDownloadStatus() {  
 return mDownloadStatus;  
 }

public void execute() {  
 this.mDownloadStatus = DownloadStatus.PROCESSING;  
 DownloadRawData downloadRawData = new DownloadRawData();  
 downloadRawData.execute(mRawUrl);  
 }  
public class DownloadRawData extends AsyncTask<String, Void, String> {  
 protected void onPostExecute(String webData) {  
 mData = webData;  
 Log.v(LOG\_TAG, "Data returned was: " + mData);  
 if(mData == null) {  
 if(mRawUrl == null) {  
 mDownloadStatus = DownloadStatus.NOT\_INITIALISED;  
 } else {  
 mDownloadStatus = DownloadStatus.FAILED\_OR\_EMPTY;  
 }

} else {  
 // success  
 mDownloadStatus = DownloadStatus.OK;  
 }  
 }  
 @Override  
 protected String doInBackground(String... params) {  
 HttpURLConnection urlConnection = null;  
 BufferedReader reader = null;  
 if(params == null) {  
 return null;  
 }  
 try {  
 URL url = new URL(params[0]);  
 urlConnection = (HttpURLConnection) url.openConnection();  
 urlConnection.setRequestMethod("GET");  
 urlConnection.connect();  
 InputStream inputStream = urlConnection.getInputStream();  
 if(inputStream == null) {  
 return null;  
 }  
 StringBuffer buffer = new StringBuffer();  
 reader = new BufferedReader(new InputStreamReader(inputStream));  
 String line;  
 while ((line = reader.readLine()) != null) {  
 buffer.append(line + "\n");  
 }  
 return buffer.toString();  
  
 } catch (IOException e ) {  
 Log.e(LOG\_TAG, "Error", e);  
 return null;  
 } finally {  
 if(urlConnection != null) {  
 urlConnection.disconnect();  
 }  
 if(reader != null) {  
 try {  
 reader.close();  
 } catch (final IOException e) {  
 Log.e(LOG\_TAG, "Error closing stream", e);  
 }  
 }  
 }  
 }  
 }  
}

GetElectronicsJson.java:-  
package com.example.mohanteja.vizagporttrust;  
import android.net.Uri;  
import android.os.AsyncTask;  
import android.util.Log;  
import android.widget.Switch;  
import org.json.JSONArray;  
import org.json.JSONException;  
import org.json.JSONObject;  
import java.util.ArrayList;  
import java.util.List;  
public class GetElectronicsJson extends GetRawData {  
 private static final String TAG = "GetElectronicsJson";  
 private String LOG\_TAG = GetElectronicsJson.class.getSimpleName();  
 //total declaration of json types used in the app  
 private List<TotalCameraItem> totalCameraItemList;  
 private List<Cameras> mCameras;  
 private List<BGLocoItem> bgLocoItemList;  
 private List<CraneItem> craneItemList;  
 private List<LightingItem> lightingItemList;  
 private List<FcsDeploymentItems> fcsDeploymentItemsList;  
 private List<FcsEachCraftItem> fcsEachCraftItemList;  
 private List<PowerDistributionItem> powerDistributionItemList;  
 private List<PumpHouseItem> pumpHouseItemList;  
 private int pageIndex;  
 /\*--------------list of pages in database that send json items and their pageindex  
 pageIndex page  
 1. cameras\_total  
 2 video\_surveillance\_cameras  
 3 railway\_weighbridges  
 4 truck\_weighbridges  
 5 radio\_navigation\_aids  
 6 rde\_station  
 7 rfid  
 8 floating\_crafts\_section  
 9 fcs\_tugs\_details  
 10 fcs\_floating\_cranes  
 11 fcs\_dredger  
 12 fcs\_pilot\_launches  
 13 fcs\_mooring\_launches  
 14 fcs\_vip\_launch  
 15 fcs\_oil\_barage  
 16 fcs\_survey\_launch  
 17 fcs\_fire\_float  
 18 fcs\_pollution\_craft  
 19 bg\_loco  
 20 pump\_houses  
 21 cranes  
 22 power\_distribution  
 23 lighting  
 \* \*/  
 //url  
 private Uri mDestinationUri;  
 //---------constructor-----------------------------just initializes the respective array  
 public GetElectronicsJson(String jsonpage, int pageIndex) {  
 super(null);  
 createAndUpdateUri(jsonpage);  
 this.pageIndex=pageIndex;  
 switch (pageIndex) {  
 case 1:  
 totalCameraItemList = new ArrayList<TotalCameraItem>();  
 break;  
 //similar objects in database  
 case 2: case 3: case 4: case 5: case 6:  
 case 7:  
 mCameras= new ArrayList<Cameras>();  
 break;  
 case 8:  
 fcsDeploymentItemsList=new ArrayList<FcsDeploymentItems>();  
 break;  
 //similar objects  
 case 9: case 10: case 11: case 12: case 13: case 14: case 15: case 16: case 17:  
 case 18:  
 fcsEachCraftItemList =new ArrayList<FcsEachCraftItem>();  
 break;  
 case 19:  
 bgLocoItemList=new ArrayList<BGLocoItem>();  
 break;  
 case 20:  
 pumpHouseItemList=new ArrayList<PumpHouseItem>();  
 break;  
 case 21:  
 craneItemList=new ArrayList<CraneItem>();  
 break;  
 case 22:  
 powerDistributionItemList=new ArrayList<PowerDistributionItem>();  
 break;  
 case 23:  
 lightingItemList=new ArrayList<LightingItem>();  
 default:  
 Log.e(TAG, "GetElectronicsJson: illegal page index refer GetJsonData.java file for commentSection" );  
 break;  
 }  
 }  
 public void execute() {  
 super.setmRawUrl(mDestinationUri.toString());  
 DownloadJsonData downloadJsonData = new DownloadJsonData();  
 Log.v(LOG\_TAG, "Built URI = " + mDestinationUri.toString());  
 downloadJsonData.execute(mDestinationUri.toString());  
 }  
 public boolean createAndUpdateUri(String jsonPage) {  
 final String DATABASE\_BASE\_URL = "http://mohantejachitturi1.000webhostapp.com";  
 mDestinationUri = Uri.parse(DATABASE\_BASE\_URL).buildUpon()  
 .appendPath(jsonPage)  
 .build();  
 return mDestinationUri != null;  
 }  
// ----------------getter functions--------------------------------------  
 public List<TotalCameraItem> getTotalCameraItemList() {  
 return totalCameraItemList;  
 }  
 public List<Cameras> getmCameras() {  
 return mCameras; }  
 public List<BGLocoItem> getBgLocoItemList() {  
 return bgLocoItemList;  
 }  
 public List<CraneItem> getCraneItemList() {  
 return craneItemList;  
 }  
 public List<LightingItem> getLightingItemList() {  
 return lightingItemList;   
}  
 public List<FcsDeploymentItems> getFcsDeploymentItemsList() {  
 return fcsDeploymentItemsList;  
 }  
 public List<FcsEachCraftItem> getFcsEachCraftItemList() {  
 return fcsEachCraftItemList;  
 }  
 public List<PowerDistributionItem> getPowerDistributionItemList() {  
 return powerDistributionItemList;  
 }  
 public List<PumpHouseItem> getPumpHouseItemList() {  
 return pumpHouseItemList;  
 }  
 public void clear(){  
 }  
//------------------json to object conversion--------------------------------------------

public void processResult() {  
 if(getmDownloadStatus() != DownloadStatus.OK) {  
 Log.e(LOG\_TAG, "Error downloading raw file");  
 return;  
 }  
 try {  
 //switch case for downloading only for the parsing required item---  
 //difficult to understand but eliminates code duplication  
 JSONArray itemArray = new JSONArray(getmData());  
 switch (pageIndex) {  
 case 1:  
 if(!totalCameraItemList.isEmpty())  
 totalCameraItemList.clear();  
 for(int i=0; i<itemArray.length(); i++) {  
// total cameras table  
 JSONObject jsonObject1= itemArray.getJSONObject(i);  
 String type =jsonObject1.getString("type");  
 String working =jsonObject1.getString("WORKING");  
 String not\_working=jsonObject1.getString("NOT\_WORKING");  
 String remarks=jsonObject1.getString("REMARKS");  
 TotalCameraItem totalCameraItem =new TotalCameraItem(type,working,not\_working,remarks);  
 this.totalCameraItemList.add(totalCameraItem);  
 }  
 for(TotalCameraItem singleTotalCameraItemList: totalCameraItemList) {  
 Log.v(LOG\_TAG, singleTotalCameraItemList.toString());  
 }  
 break;  
 case 2:  
 case 3:  
 case 4:  
 case 5:  
 case 6:  
 case 7:

// cameras  
 if(!mCameras.isEmpty())  
 mCameras.clear();  
 for(int i=0; i<itemArray.length(); i++) {  
 JSONObject jsonObject1= itemArray.getJSONObject(i);  
 String location =jsonObject1.getString("LOCATION");  
 String status=jsonObject1.getString("STATUS");  
 String remarks=jsonObject1.getString("REMARKS");  
 Cameras cameras= new Cameras(location,status,remarks);  
 this.mCameras.add(cameras);  
 }  
 for(Cameras singlePhoto: mCameras) {  
 Log.v(LOG\_TAG, singlePhoto.toString());  
 }  
 break;  
 case 8:  
 if(!fcsDeploymentItemsList.isEmpty())

fcsDeploymentItemsList.clear();

//fcs craft items types

for(int i=0; i<itemArray.length(); i++) {

JSONObject jsonObject1= itemArray.getJSONObject(i);

String slno =jsonObject1.getString("SLNO");

String craftType=jsonObject1.getString("CRAFT\_TYPE");

String totalCrafts=jsonObject1.getString("TOTAL\_CRAFTS");

String availableCrafts=jsonObject1.getString("AVAILABLE\_CRAFTS");

String shift1=jsonObject1.getString("1\_SHIFT");

String shift2=jsonObject1.getString("2\_SHIFT");

String shift3=jsonObject1.getString("3\_SHIFT");

String shiftg=jsonObject1.getString("G/SHIFT");

FcsDeploymentItems fcsDeploymentItem= new FcsDeploymentItems(slno,craftType,totalCrafts,availableCrafts,shift1,shift2,shift3,shiftg);

this.fcsDeploymentItemsList.add(fcsDeploymentItem);

}

for(FcsDeploymentItems fcsDeploymentItem: fcsDeploymentItemsList) {

Log.v(LOG\_TAG, fcsDeploymentItem.toString());

}

break;

case 9:

case 10:

case 11:

case 12:

case 13:

case 14:

case 15:

case 16:

case 17:

case 18:

//fcs items of each type of craft

if(!fcsEachCraftItemList.isEmpty())

fcsEachCraftItemList.clear();

for(int i=0; i<itemArray.length(); i++) {

JSONObject jsonObject1= itemArray.getJSONObject(i);

String slno =jsonObject1.getString("SL.NO");

String name\_of\_the\_craft =jsonObject1.getString("NAME\_OF\_THE\_CRAFT");

String commission =jsonObject1.getString("COMMISSION");

String shutdown =jsonObject1.getString("SHUTDOWN");

String no\_of\_movements\_shift\_1 =jsonObject1.getString("NO\_OF\_MOVEMENTS\_SHIFT\_1");

String no\_of\_movements\_shift\_2 =jsonObject1.getString("NO\_OF\_MOVEMENTS\_SHIFT\_2");

String no\_of\_movements\_shift\_3 =jsonObject1.getString("NO\_OF\_MOVEMENTS\_SHIFT\_3");

String no\_of\_movements\_shift\_g=jsonObject1.getString("NO\_OF\_MOVEMENTS\_SHIFT\_G");

String total\_no\_of\_movements=jsonObject1.getString("TOTAL\_NO\_OF\_MOVEMENTS");

FcsEachCraftItem fcsEachCraftItem =new FcsEachCraftItem(slno,name\_of\_the\_craft,commission,shutdown,no\_of\_movements\_shift\_1,no\_of\_movements\_shift\_2,no\_of\_movements\_shift\_3,no\_of\_movements\_shift\_g,total\_no\_of\_movements);

this.fcsEachCraftItemList.add(fcsEachCraftItem);

}

for(FcsEachCraftItem singlePhoto: fcsEachCraftItemList) {

Log.v(LOG\_TAG, singlePhoto.toString());

} break;

case 19:

//bg locos table

if(!bgLocoItemList.isEmpty())

bgLocoItemList.clear();

for(int i=0; i<itemArray.length(); i++) {

JSONObject jsonObject1= itemArray.getJSONObject(i);

String no\_l\_oPorD =jsonObject1.getString("NO\_L\_OP/D");

String no\_l\_a\_m =jsonObject1.getString("NO\_L\_A\_M");

String no\_l\_poh =jsonObject1.getString("NO\_L\_POH");

String no\_l\_pm =jsonObject1.getString("NO\_L\_PM");

String no\_l\_bd =jsonObject1.getString("NO\_L\_BD");

String incoming\_rakes =jsonObject1.getString("INCOMING\_RAKES");

String outgoing\_rakes=jsonObject1.getString("OUTGOING\_RAKES");

String no\_rorD=jsonObject1.getString("NO\_R/D");

BGLocoItem bgLocoItem = new BGLocoItem(no\_l\_oPorD,no\_l\_a\_m,no\_l\_poh,no\_l\_pm,no\_l\_bd,incoming\_rakes,outgoing\_rakes,no\_rorD);

this.bgLocoItemList.add(bgLocoItem);

}

for(BGLocoItem bgLocoItem: bgLocoItemList) {

Log.v(LOG\_TAG, bgLocoItem.toString());

}

break;

// pump house

case 20:

if(!pumpHouseItemList.isEmpty())

pumpHouseItemList.clear();

for(int i=0; i<itemArray.length(); i++) {

JSONObject jsonObject1= itemArray.getJSONObject(i);

String slno =jsonObject1.getString("SL.NO");

String pump\_houses =jsonObject1.getString("PUMP\_HOUSES");

String total\_pumps\_motors =jsonObject1.getString("TOTAL\_PUMPS/MOTORS");

String shift\_1 =jsonObject1.getString("SHIFT\_1");

String shift\_2 =jsonObject1.getString("SHIFT\_2");

String shift\_3=jsonObject1.getString("SHIFT\_3");

String shift\_g=jsonObject1.getString("SHIFT\_G");

PumpHouseItem pumpHouseItem = new PumpHouseItem(slno,pump\_houses,total\_pumps\_motors,shift\_1,shift\_2,shift\_3,shift\_g);

this.pumpHouseItemList.add(pumpHouseItem);

}

for(PumpHouseItem pumpHouseItem: pumpHouseItemList) { Log.v(LOG\_TAG, pumpHouseItem.toString());  
 }  
 break;

// cranes  
 case 21:  
 if(!craneItemList.isEmpty())  
 craneItemList.clear();

for(int i=0; i<itemArray.length(); i++) {  
 JSONObject jsonObject1= itemArray.getJSONObject(i);  
 String slno =jsonObject1.getString("SL.NO");

String crane\_type =jsonObject1.getString("CRANE\_TYPE");

String total\_cranes =jsonObject1.getString("TOTAL\_CRANES");

String availability =jsonObject1.getString("AVAILABILITY");

String shift\_1 =jsonObject1.getString("SHIFT\_1");

String shift\_2 =jsonObject1.getString("SHIFT\_2");

String shift\_3=jsonObject1.getString("SHIFT\_3");

String shift\_g=jsonObject1.getString("SHIFT\_G");

CraneItem craneItem = new CraneItem(slno,crane\_type,total\_cranes,availability,shift\_1,shift\_2,shift\_3,shift\_g);

this.craneItemList.add(craneItem);

}

for(CraneItem singlePhoto: craneItemList) {

Log.v(LOG\_TAG, singlePhoto.toString());

}

break;

// power distribution

case 22:

if(!powerDistributionItemList.isEmpty())

powerDistributionItemList.clear();

for(int i=0; i<itemArray.length(); i++) {

JSONObject jsonObject1= itemArray.getJSONObject(i);

String slno =jsonObject1.getString("SL.NO");

String parameters =jsonObject1.getString("PARAMETERS");

String shift\_1 =jsonObject1.getString("SHIFT\_1");

String shift\_2 =jsonObject1.getString("SHIFT\_2");

String shift\_3=jsonObject1.getString("SHIFT\_3");

String shift\_g=jsonObject1.getString("SHIFT\_G");

PowerDistributionItem powerDistributionItem = new PowerDistributionItem(slno,parameters,shift\_1,shift\_2,shift\_3,shift\_g);

this.powerDistributionItemList.add(powerDistributionItem);

}

for(PowerDistributionItem powerDistributionItem: powerDistributionItemList) {

Log.v(LOG\_TAG, powerDistributionItem.toString());

}

break;

// lighting

case 23:

if(!lightingItemList.isEmpty())

lightingItemList.clear();

for(int i=0; i<itemArray.length(); i++) {

JSONObject jsonObject1= itemArray.getJSONObject(i);

String slno =jsonObject1.getString("S.NO");

String location =jsonObject1.getString("LOCATION");

String no\_of\_high\_masts =jsonObject1.getString("NO\_OF\_HIGH\_MASTS");

String fitting\_x\_h\_m\_a =jsonObject1.getString("FITTING\_X\_H/M:A");

String fitting\_x\_h\_m\_b =jsonObject1.getString("FITTING\_X\_H/M:B");

String fitting\_x\_h\_m\_c=jsonObject1.getString("FITTING\_X\_H/M:C");

String fitting\_x\_h\_m\_d=jsonObject1.getString("FITTING\_X\_H/M:D");

String total\_fittings=jsonObject1.getString("TOTAL\_FITTINGS");

String glowing=jsonObject1.getString("GLOWING");

String non\_glowing=jsonObject1.getString("NON\_GLOWING");

LightingItem lightingItem = new LightingItem(slno,location,no\_of\_high\_masts,fitting\_x\_h\_m\_a,fitting\_x\_h\_m\_b,fitting\_x\_h\_m\_c,fitting\_x\_h\_m\_d,total\_fittings,glowing,non\_glowing);

this.lightingItemList.add(lightingItem);

}

for(LightingItem lightingItem: lightingItemList) {

Log.v(LOG\_TAG, lightingItem.toString());

}

break;

default:

Log.e(TAG, "GetElectronicsJson: illegal page index refer GetJsonData.java file for commentSection" );

break;

}

} catch (JSONException jsone) {

jsone.printStackTrace();

Log.e(LOG\_TAG, "Error process Json data");

}

}

public class DownloadJsonData extends DownloadRawData {

protected void onPostExecute(String webData) {

super.onPostExecute(webData);

processResult();

}

protected String doInBackground(String... params) {

String[] par = { mDestinationUri.toString()};

return super.doInBackground(par);

}

}

}

ElectronicViewHolder:-

package com.example.mohanteja.vizagporttrust;

import android.support.v7.widget.RecyclerView;

import android.view.View;

import android.widget.ImageView;

import android.widget.Switch;

import android.widget.TextView;

public class ElectronicsViewHolder extends RecyclerView.ViewHolder {

public ImageView electronicsImageView;

public TextView electronicsLocationTextView;

public TextView electronicsRemarksTextView;

// public Switch statusSwitch;

public ElectronicsViewHolder(View itemView) {

super(itemView);

electronicsLocationTextView = itemView.findViewById(R.id.location);

electronicsImageView = itemView.findViewById(R.id.elecCameraIcon);

electronicsRemarksTextView = itemView.findViewById(R.id.remarks);

// statusSwitch = itemView.findViewById(R.id.statusSwitch);

}

}

ElectronicAdapter.java:-

package com.example.mohanteja.vizagporttrust;

import android.content.Context;

import android.media.Image;

import android.support.annotation.NonNull;

import android.support.v7.widget.RecyclerView;

import android.util.Log;

import android.view.LayoutInflater;

import android.view.View;

import android.view.ViewGroup;

import android.widget.ImageView;

import android.widget.Switch;

import android.widget.TextView;

import java.util.ArrayList;

import java.util.List;

public class ElectronicsAdapter extends RecyclerView.Adapter<ElectronicsViewHolder> {

private static final String TAG = "ElectronicsAdapter";

private List<Cameras> mCameraList;

private Context mContext;

private final String LOG\_TAG = ElectronicsAdapter.class.getSimpleName();

public ElectronicsAdapter(Context context, List<Cameras> camerasList) {

mContext = context;

this.mCameraList = camerasList;

}

@Override

public ElectronicsViewHolder onCreateViewHolder(ViewGroup viewGroup, int i) {

View view = LayoutInflater.from(viewGroup.getContext()).inflate(R.layout.electronics\_card, null);

ElectronicsViewHolder flickrImageViewHolder = new ElectronicsViewHolder(view);  
 return flickrImageViewHolder;  
 }  
 @Override  
 public void onBindViewHolder(ElectronicsViewHolder electronicsViewHolder, int i) {  
 Cameras camerasItem = mCameraList.get(i);  
 Log.d(LOG\_TAG,"Processing: " + camerasItem.getLocation() + "==> " + Integer.toString(i)); electronicsViewHolder.electronicsLocationTextView.setText(camerasItem.getLocation()); electronicsViewHolder.electronicsRemarksTextView.setText("Remarks:" + camerasItem.getRemarks()); if(Integer.parseInt(camerasItem.getStatus())== 1) {

electronicsViewHolder.electronicsImageView.setImageResource(R.drawable.baseline\_linked\_camera\_black\_36);

}

else { electronicsViewHolder.electronicsImageView.setImageResource(R.drawable.baseline\_report\_black\_36);

}

}

@Override

public int getItemCount() {

return (null != mCameraList ? mCameraList.size() : 0);

}

public void loadNewData(List<Cameras> newPhotos) {

// clear(newPhotos);

mCameraList = newPhotos;

notifyDataSetChanged();

}

public void clear(List<Cameras> newPhotos){

Log.d(TAG, "clear: clearing list");

final int size =mCameraList.size();

if(size > 0){

for (int i =0; i<size;i++)

{

mCameraList.remove(0);

}

}

notifyItemRangeRemoved(0, size);

Log.d(TAG, "clear: removed =" + mCameraList.isEmpty());

}

public Cameras getPhoto(int position) {

return (null != mCameraList ? mCameraList.get(position) : null);

}

@Override

public int getItemViewType(int position) {

return position;

}

@Override

public long getItemId(int position) {

return position;

}

}

CONCLUSION:

The primary goal of these app is to provide an efficient way for accessing,retrieving,monitoring the data and also to provide an User friendly environment for the user so that he/she will not feel any burden while doing his work.As we know now-a-days,smart phones have been playing an important role in present daily living and everyone will love to work through devices which are portable ,hand held and can be used anywhere. There are no restrictions on usage for this type of devices .So,every person loves to work on such type of devices. The Main advatage of using such type of devices is that it saves more time than working on paper .Coming to the situation of employees ,every employee in an organisation shows more interest on working such kind of devices than working on paper,

In Visakhapatnam Port Trust (VPT) , employees depend on various categories of machinery in order to meat their daily duties.Any repair or failure of machinery or unavailability may result in the delay of work and wastage of employees time if they don’t know the information prior and attend the work.Our main objective behind designing these app is to help employees of VPT in knowing the information regarding the machinery in working area in their respective shifts and so that they can attend their duties effectively and thereby increasing the productivity for the port.