|  |  |
| --- | --- |
| Diploma in Infocomm & Network Engineering  Mobile Device Applications Development (MDAD) |  |

Project Report (save report as PE0xTy.doc where x is class no. and y is team no.)

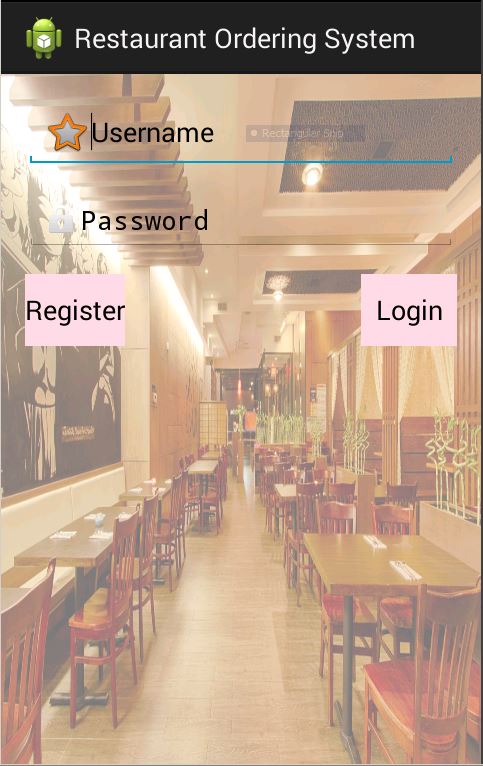
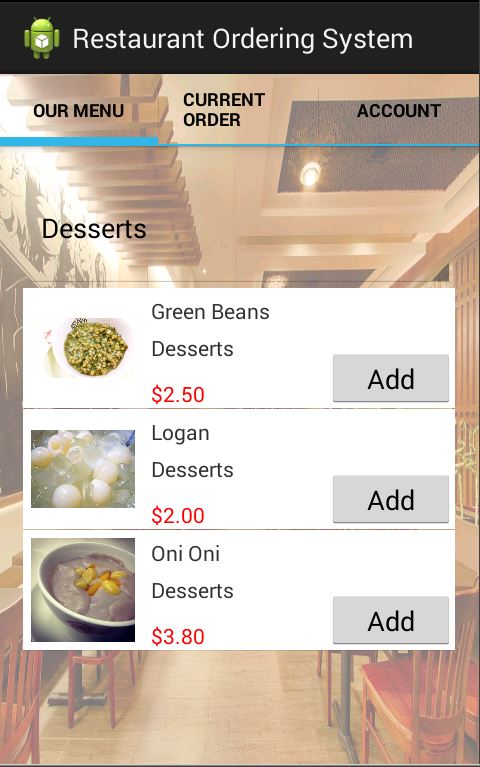
*(eg. PE01T3.doc)*

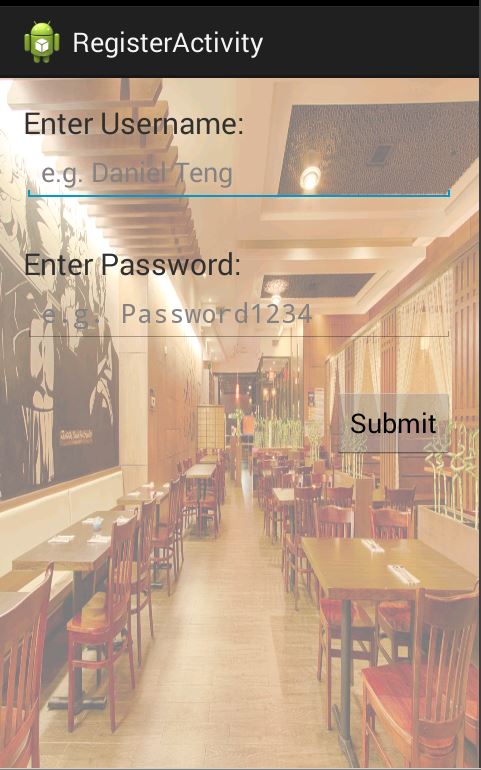
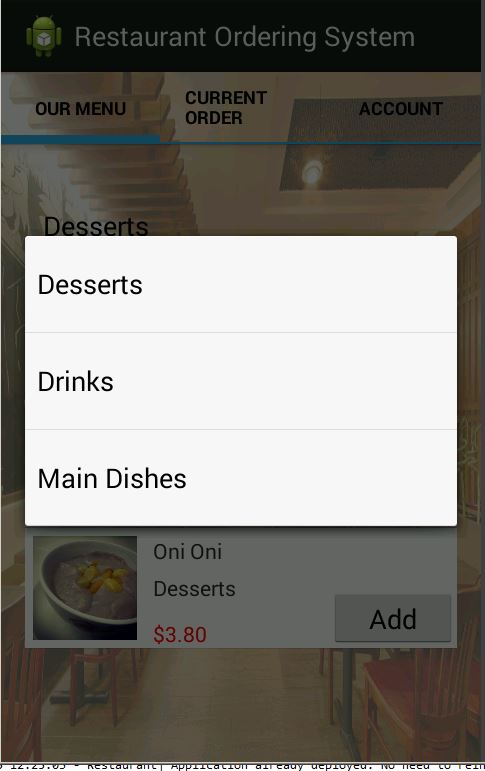
|  |  |
| --- | --- |
| **Class (lab):** | **PE01** |
| **Team Number:** | **4** |
| **Name of Mobile App** | **Restaurant** |
| **Category** | Business |
| **Matric No** | **Name** |
| **1501545F** | **Yong Jian Ming** |
| **1506600D** | **Tan Ying Lin** |

**About the App** (**brief** description about the mobile application)

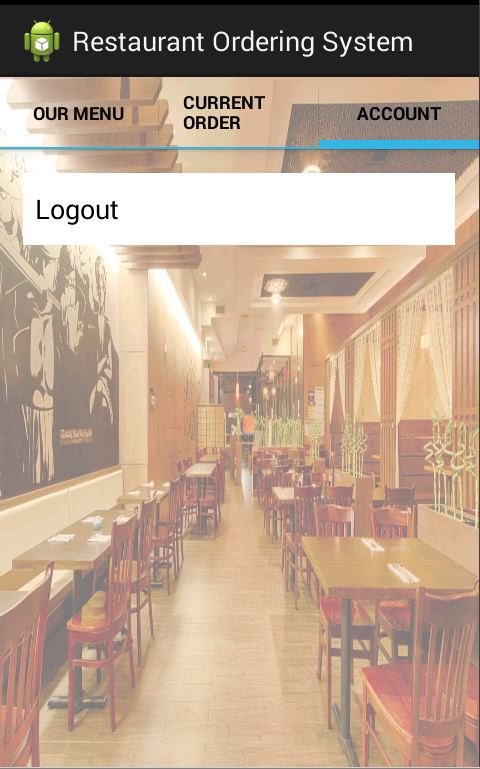
Our main focus of our mobile application is meant for customers and staff in a restaurant for friendly uses purposes. The customer is required to register for first time customer to login and to select the menu to order their food. As for the staff, they are known as the admin who has access to edit, insert new category and food, as well as removing some food that they want to take down.

**How to use the App**(with screenshots)

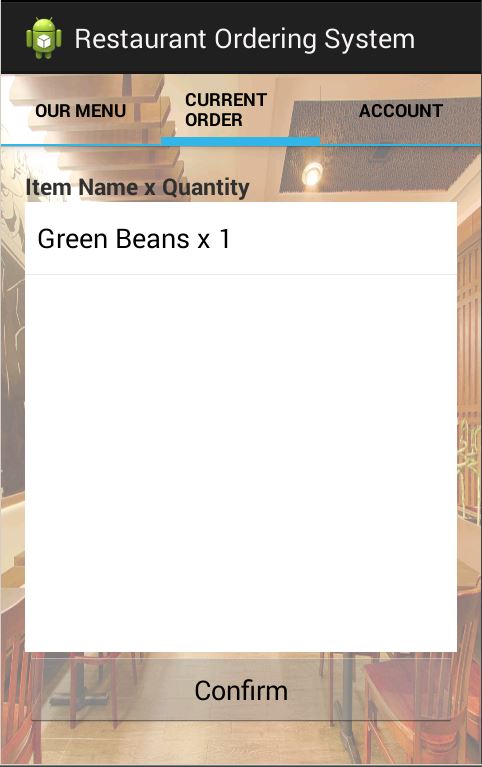
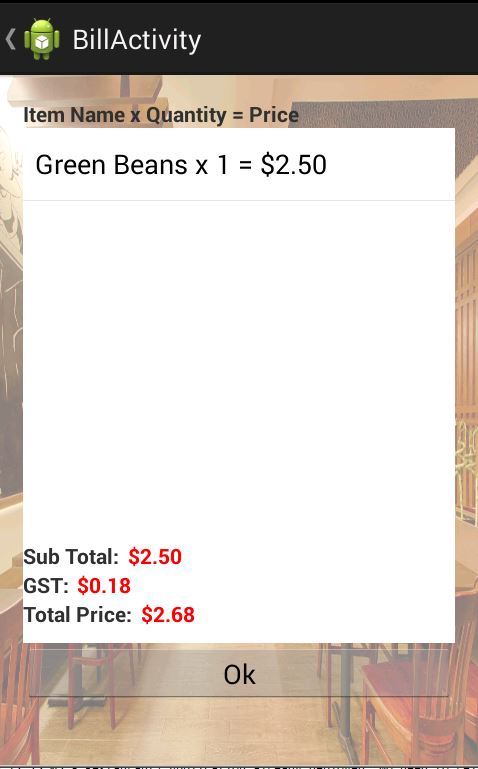
 

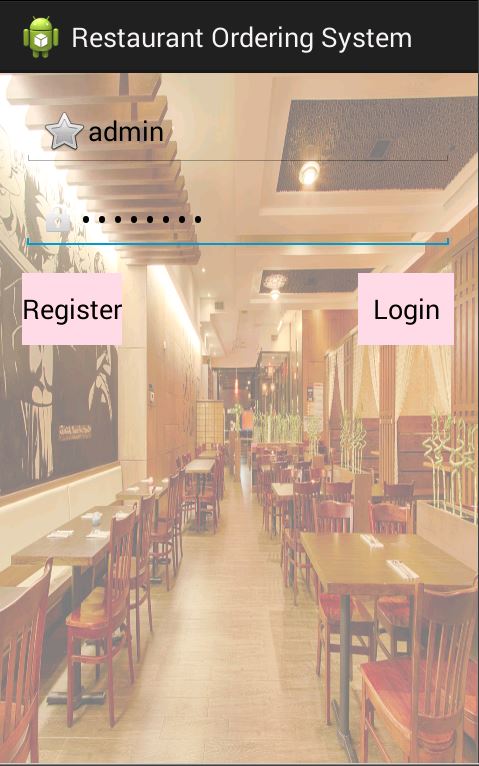
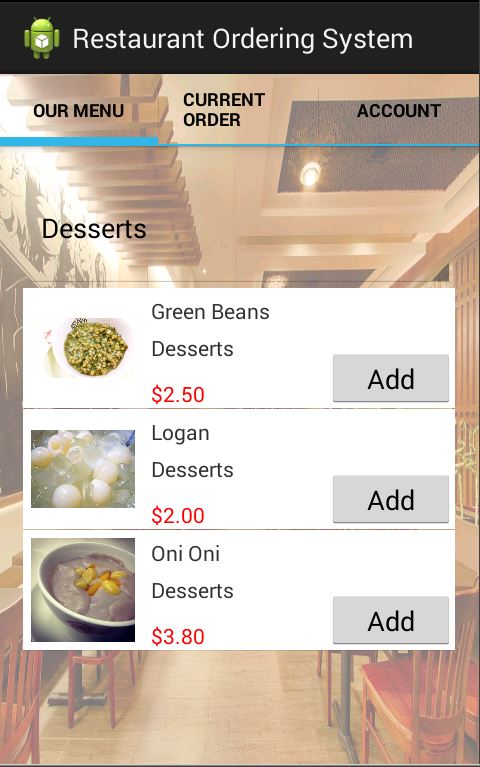
It will store in database and the page will be sent back to the login page

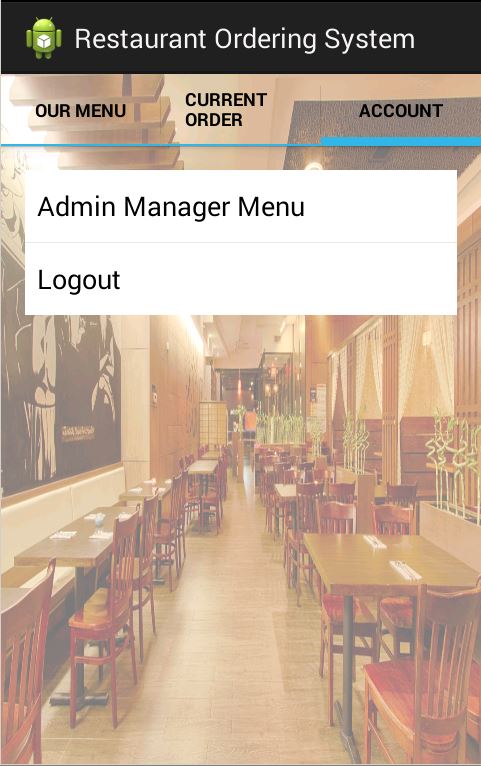
 

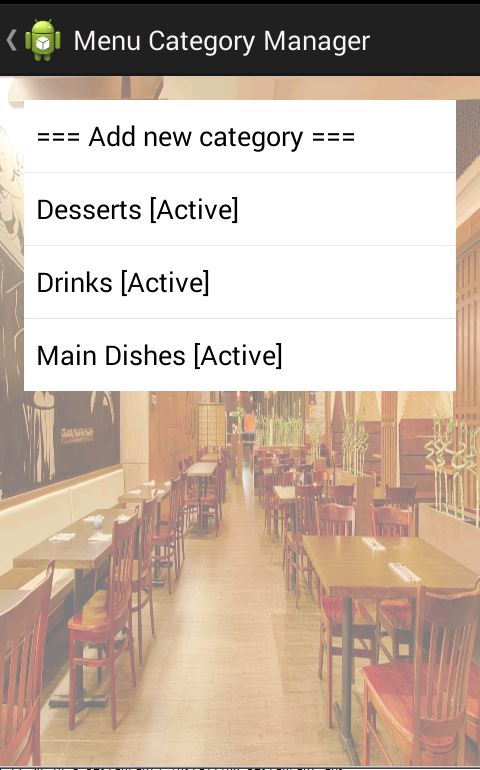
It will go back to the login page.

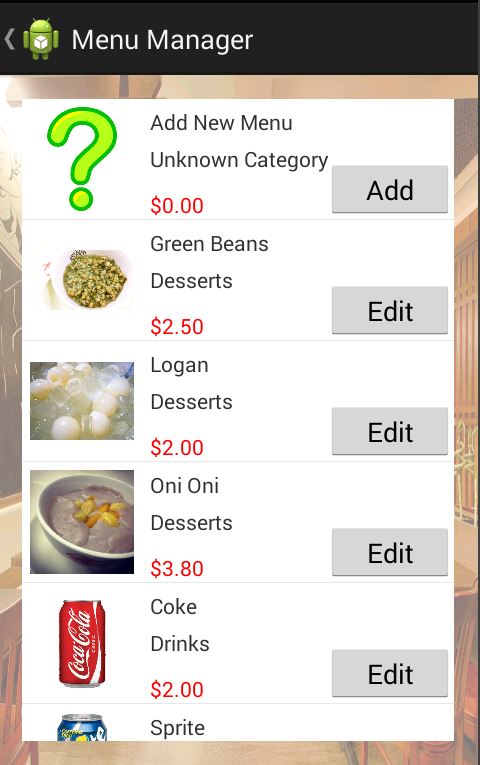
It will go back to main page.

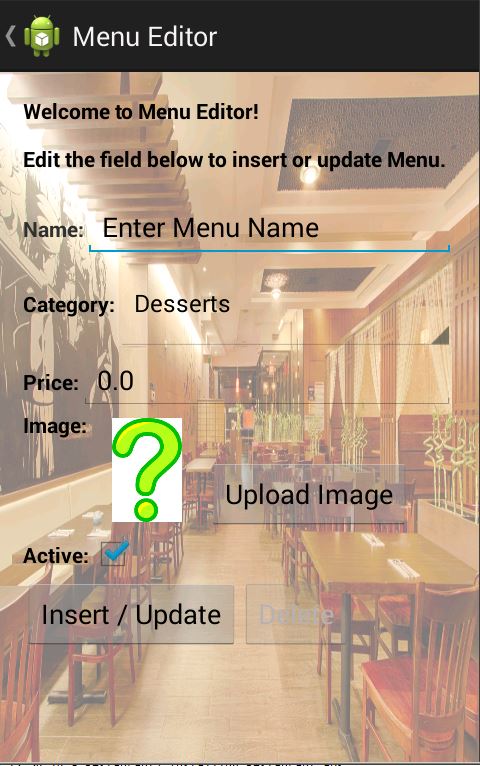
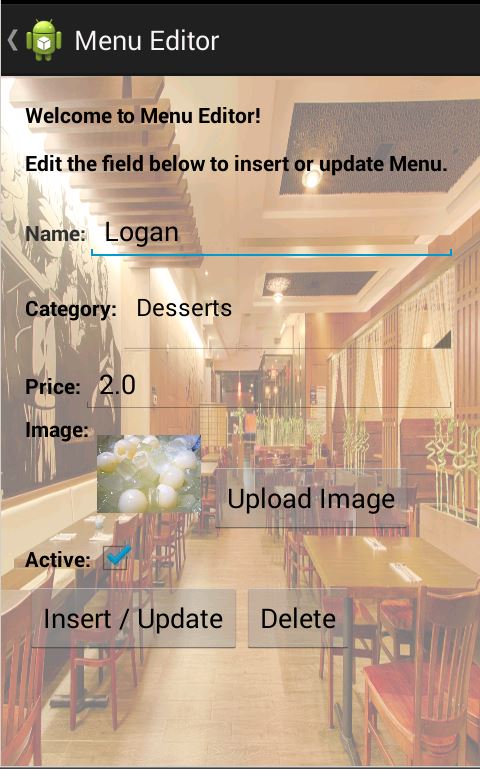
**** 

** **

** **

It will appear here

** **

** **

After clicking this button, it will be deleted from database and the screen will return to Menu Manager activity.

Both insert button will go into the Menu Manager activity in the list of all the items.

**Technical Info about the App** (**brief** **description** on **each** of the Java classes written – ***Do NOT copy the whole program codes here*)**

LoginActivity.java

Get the account from the Account table in mysql to validate the account:

NetworkHelper.execute(LoginActivity.this, new NetworkTaskDelegate() {

@Override

public HttpURLConnection initConnection(URL url, Map<String, String> postData) throws Exception {

postData.put("action", "getAccount");

postData.put("name", username);

postData.put("password", password);

return NetworkHelper.doPostRequest(url, postData);

}

@Override

public void onSuccess(List<String> result) throws Exception {

SharedPreferences sp = getSharedPreferences("Account", 0);

Editor edit = sp.edit();

edit.putInt("AccountID", Integer.valueOf(result.get(1)));

edit.putBoolean("IsAdmin", result.get(2).contentEquals("1") ? true : false);

edit.commit();

if (result.get(3).contentEquals("1"))

Toast.makeText(getApplicationContext(), "Sorry you are not allowed to login.", Toast.LENGTH\_LONG).show();

else {

Intent msg = new Intent(LoginActivity.this, MainActivity.class);

startActivity(msg);

}

}

});

RegisterActivity.java

Register an account by inserting into the Account table in mysql.

NetworkHelper.execute(getApplicationContext(), new NetworkTaskDelegate() {

@Override

public HttpURLConnection initConnection(URL url, Map<String, String> postData) throws Exception {

postData.put("action", "setAccount");

postData.put("username", username);

postData.put("password", pwd);

return NetworkHelper.doPostRequest(url, postData);

}

@Override

public void onSuccess(List<String> result) throws Exception {

startActivity(new Intent(RegisterActivity.this, LoginActivity.class));

Toast.makeText(getApplicationContext(), "You are registered!", Toast.LENGTH\_LONG).show();

System.out.println("You are registered!");

}

});

MainActivity.java

This is to setup the TabHost with three options. “Our Menu”, “Current Order” and “Account”.

private final void initThMain() {

final TabHost thMain = (TabHost) findViewById(R.id.thMain);

thMain.setup(getLocalActivityManager());

thMain.addTab(thMain.newTabSpec("menu").setIndicator("Our Menu", null).setContent(new Intent(MainActivity.this, MenuActivity.class)));

thMain.addTab(thMain.newTabSpec("order").setIndicator("Current Order", null).setContent(new Intent(MainActivity.this, OrderActivity.class)));

thMain.addTab(thMain.newTabSpec("account").setIndicator("Account", null).setContent(new Intent(MainActivity.this, AccountActivity.class)));

thMain.setCurrentTab(index);

}

MenuActivity.java

Display the category:

NetworkHelper.execute(this, new NetworkTaskDelegate() {

@Override

public final HttpURLConnection initConnection(URL url, Map<String, String> postData) throws Exception {

postData.put("action", "getMenuCategoryList");

return NetworkHelper.doPostRequest(url, postData);

}

@Override

public final void onSuccess(List<String> result) throws Exception {

final List<String> items = new ArrayList<String>();

for (int i = 1; i < result.size(); i += 3) {

final int id = Integer.parseInt(result.get(i));

final String name = result.get(i + 1);

final Boolean isActive = result.get(i + 2).contentEquals("1") ? true : false;

if (isActive) {

items.add(name);

menuCategoryItems.add(new MenuCategoryItem(id, name, isActive));

}

}

}

});

Display the menu:

NetworkHelper.execute(this, new NetworkTaskDelegate() {

@Override

public final HttpURLConnection initConnection(URL url, Map<String, String> postData) throws Exception {

postData.put("action", "getMenuList");

return NetworkHelper.doPostRequest(url, postData);

}

@Override

public final void onSuccess(List<String> result) throws Exception {

for (int i = 1; i < result.size(); i += 7) {

final int id = Integer.parseInt(result.get(i));

final String name = result.get(i + 1);

final int categoryID = Integer.parseInt(result.get(i + 2));

final String category = result.get(i + 3);

final double price = Double.parseDouble(result.get(i + 4));

final String image = result.get(i + 5);

final Boolean isActive = result.get(i + 6).contentEquals("1") ? true : false;

if (isActive)

menuItems.add(new mdad.restaurant.menu.MenuItem(id, name, categoryID, category, price, image.contentEquals("NULL") ? null : image, isActive));

}

final List<mdad.restaurant.menu.MenuItem> items = new ArrayList<mdad.restaurant.menu.MenuItem>();

for (mdad.restaurant.menu.MenuItem item : menuItems) {

if (item.getCategoryID() == menuCategoryItems.get(0).getId())

items.add(item);

}

}

});

MenuSelectionActivity.java

To add an order:

public final void onClick(View v) {

final String additionalRequest = etMenuSelectionRequest.getText().toString();

SQLiteHelper.setOrder(menuID, menuQuantity, additionalRequest);

final Intent intent = new Intent(MenuSelectionActivity.this, MainActivity.class);

intent.putExtra("index", index);

startActivity(intent);

Toast.makeText(MenuSelectionActivity.this, "Menu added/updated into the order list.", Toast.LENGTH\_LONG).show();

}

public static final void setOrder(int menuID, int menuQuantity, String menuAdditionalRequest) {

db.beginTransaction();

if (getMenuExists(menuID))

db.execSQL("update orderList set menuQuantity = ?, menuAdditionalRequest = ? where menuID = ?", new Object[] { String.valueOf(menuQuantity), menuAdditionalRequest, String.valueOf(menuID) });

else

db.execSQL("insert into orderList(menuID, menuQuantity, menuAdditionalRequest) values (?, ?, ?)", new Object[] { String.valueOf(menuID), String.valueOf(menuQuantity), menuAdditionalRequest });

db.setTransactionSuccessful();

db.endTransaction();

}

To delete the order:

public final void onClick(View v) {

SQLiteHelper.deleteOrder(menuID);

final Intent intent = new Intent(MenuSelectionActivity.this, MainActivity.class);

intent.putExtra("index", index);

startActivity(intent);

Toast.makeText(MenuSelectionActivity.this, "Menu removed from the order list.", Toast.LENGTH\_LONG).show();

}

public static final void deleteOrder(int menuID) {

db.beginTransaction();

db.execSQL("delete from orderList where menuID = ?", new Object[] { String.valueOf(menuID) });

db.setTransactionSuccessful();

db.endTransaction();

}

OrderActivity.java

Get all the selected order:

NetworkHelper.execute(this, new NetworkTaskDelegate() {

@Override

public HttpURLConnection initConnection(URL url, Map<String, String> postData) throws Exception {

postData.put("action", "getMenuList");

return NetworkHelper.doPostRequest(url, postData);

}

@Override

public void onSuccess(List<String> result) throws Exception {

for (int i = 1; i < result.size(); i += 7) {

int id = Integer.parseInt(result.get(i));

String name = result.get(i + 1);

int categoryID = Integer.parseInt(result.get(i + 2));

String category = result.get(i + 3);

double price = Double.parseDouble(result.get(i + 4));

String image = result.get(i + 5);

Boolean isActive = result.get(i + 6).contentEquals("1") ? true : false;

menuItems.add(new mdad.restaurant.menu.MenuItem(id, name, categoryID, category, price, image.contentEquals("NULL") ? null : image, isActive));

}

List<String> items = new ArrayList<String>();

Cursor cursor = SQLiteHelper.getOrderItem();

while(cursor.moveToNext()) {

String name;

int qty;

for(mdad.restaurant.menu.MenuItem item : menuItems) {

if(item.getId() == cursor.getInt(1)) {

name = item.getName();

qty = cursor.getInt(2);

items.add(name + " x " + String.valueOf(qty));

menuOrderItems.add(new MenuOrderItem(item, qty, cursor.getString(3)));

}

}

}

}

});

BillActivity.java

Get the order from mysql database and show:

NetworkHelper.execute(this, new NetworkTaskDelegate() {

@Override

public HttpURLConnection initConnection(URL url, Map<String, String> postData) throws Exception {

postData.put("action", "getMenuList");

return NetworkHelper.doPostRequest(url, postData);

}

@Override

public void onSuccess(List<String> result) throws Exception {

for (int i = 1; i < result.size(); i += 7) {

int id = Integer.parseInt(result.get(i));

String name = result.get(i + 1);

int categoryID = Integer.parseInt(result.get(i + 2));

String category = result.get(i + 3);

double price = Double.parseDouble(result.get(i + 4));

String image = result.get(i + 5);

Boolean isActive = result.get(i + 6).contentEquals("1") ? true : false;

menuItems.add(new mdad.restaurant.menu.MenuItem(id, name, categoryID, category, price, image.contentEquals("NULL") ? null : image, isActive));

}

List<String> items = new ArrayList<String>();

Cursor cursor = SQLiteHelper.getOrderItem();

while(cursor.moveToNext()) {

String name;

int qty;

for(mdad.restaurant.menu.MenuItem item : menuItems) {

if(item.getId() == cursor.getInt(1)) {

name = item.getName();

qty = cursor.getInt(2);

items.add(name + " x " + String.valueOf(qty) + " = " + String.format("$%.2f", item.getPrice() \* qty));

menuOrderItems.add(new MenuOrderItem(item, qty, cursor.getString(3)));

totalPrice += item.getPrice() \* qty;

}

}

}

ArrayAdapter<String> adapter = new ArrayAdapter<String>(BillActivity.this, android.R.layout.simple\_list\_item\_1, items);

lvViewList.setAdapter(adapter);

tvBillSubTotal.setText(String.format("$%.2f", totalPrice));

tvBillGst.setText(String.format("$%.2f", totalPrice \* 0.07));

tvBillTotalPrice.setText(String.format("$%.2f", totalPrice + (totalPrice \* 0.07)));

}

});

Submit the order and store into database:

NetworkHelper.execute(this, new NetworkTaskDelegate() {

@Override

public HttpURLConnection initConnection(URL url, Map<String, String> postData) throws Exception {

postData.put("action", "setOrder");

postData.put("accountID", String.valueOf(accountID));

postData.put("menuID", String.valueOf(menuID));

postData.put("quantity", String.valueOf(quantity));

postData.put("additionalRequest", StringHelper.isNullOrEmpty(additionalRequest) ? "NULL" : Base64.encodeToString(additionalRequest.getBytes(), Base64.NO\_WRAP | Base64.URL\_SAFE));

postData.put("dateTime", String.valueOf(submitDateTime));

return NetworkHelper.doPostRequest(url, postData);

}

@Override

public void onSuccess(List<String> result) throws Exception {

submitIndex++;

if (submitIndex >= menuOrderItems.size()) {

SQLiteHelper.resetOrder();

startActivity(new Intent(BillActivity.this, MainActivity.class));

} else

submitOrder();

}

});

NetworkHelper.java

A helper class that execute all NetworkTask related jobs:

public static final void execute(Context context, NetworkTaskDelegate doInBackground) {

new NetworkTask(context, doInBackground).execute();

}

public static final HttpURLConnection doPostRequest(URL url, Map<String, String> postData) throws IOException {

final HttpURLConnection connection;

final StringBuilder postDataString = new StringBuilder();

final byte[] postDataBytes;

for (Map.Entry<String, String> param : postData.entrySet()) {

if (postDataString.length() != 0)

postDataString.append('&');

postDataString.append(URLEncoder.encode(param.getKey(), "UTF-8"));

postDataString.append('=');

postDataString.append(URLEncoder.encode(param.getValue(), "UTF-8"));

}

postDataBytes = postDataString.toString().getBytes("UTF-8");

connection = (HttpURLConnection) url.openConnection();

connection.setRequestMethod("POST");

connection.setConnectTimeout(5000);

connection.setDoInput(true);

connection.setDoOutput(true);

connection.setRequestProperty("Content-Type", "application/x-www-form-urlencoded");

connection.setRequestProperty("Content-Length", String.valueOf(postDataBytes.length));

connection.getOutputStream().write(postDataBytes);

return connection;

}

NetworkTask.java

The actual network task:

protected final List<String> doInBackground(String... params) {

try {

final URL url = new URL("http://10.0.2.2:8080/MDAD/servlet/server.DatabaseManagerServlet");

if (networkTaskDelegate != null) {

final HttpURLConnection connection = networkTaskDelegate.initConnection(url, postData);

final BufferedReader reader = new BufferedReader(new InputStreamReader(connection.getInputStream()));

String temp;

while ((temp = reader.readLine()) != null)

result.add(temp);

reader.close();

connection.disconnect();

} else {

result.add("false");

result.add("Unknown action defined.");

}

} catch (Exception ex) {

result.add("false");

result.add(ex.getMessage());

}

return result;

}

protected final void onPostExecute(List<String> result) {

try {

if (result.get(0).contentEquals("false"))

catchError(result.get(1));

else if (result.get(0).contentEquals("true"))

networkTaskDelegate.onSuccess(result);

else

catchError("Unknown result");

} catch (Exception ex) {

catchError(ex.getMessage(), ex.getStackTrace());

}

}

Admin Related task:

Add new category:

NetworkHelper.execute(getApplicationContext(), new NetworkTaskDelegate() {

@Override

public final HttpURLConnection initConnection(URL url, Map<String, String> postData) throws Exception {

postData.put("action", "setMenuCategoryList");

postData.put("id", String.valueOf(id));

postData.put("name", name);

postData.put("isActive", isActive.toString());

return NetworkHelper.doPostRequest(url, postData);

}

@Override

public final void onSuccess(List<String> result) throws Exception {

startActivity(new Intent(AdminCategoryEditorActivity.this, AdminCategoryActivity.class));

Toast.makeText(getApplicationContext(), "Menu category updated!", Toast.LENGTH\_LONG).show();

System.out.println("Menu category updated!");

}

});

Delete Category:

NetworkHelper.execute(getApplicationContext(), new NetworkTaskDelegate() {

@Override

public final HttpURLConnection initConnection(URL url, Map<String, String> postData) throws Exception {

postData.put("action", "deleteMenuCategoryList");

postData.put("id", String.valueOf(id));

return NetworkHelper.doPostRequest(url, postData);

}

@Override

public final void onSuccess(List<String> result) throws Exception {

startActivity(new Intent(AdminCategoryEditorActivity.this, AdminCategoryActivity.class));

Toast.makeText(getApplicationContext(), "Menu category deleted!", Toast.LENGTH\_LONG).show();

System.out.println("Menu category deleted!");

}

});

Upload Image from mobile (gallery):

@Override

public final void onClick(View v) {

startActivityForResult(new Intent(Intent.ACTION\_PICK, android.provider.MediaStore.Images.Media.EXTERNAL\_CONTENT\_URI), 1);

}

@Override

protected final void onActivityResult(int requestCode, int resultCode, Intent data) {

super.onActivityResult(requestCode, resultCode, data);

if (requestCode == 1 && resultCode == RESULT\_OK && data != null) {

final Uri selectedImage = data.getData();

final String[] filePathColumn = { MediaColumns.DATA };

final Cursor cursor = getContentResolver().query(selectedImage, filePathColumn, null, null, null);

cursor.moveToNext();

final int columnIndex = cursor.getColumnIndex(filePathColumn[0]);

final String picturePath = cursor.getString(columnIndex);

cursor.close();

final Bitmap imageBitmap = BitmapFactory.decodeFile(picturePath);

image = BitmapHelper.convert(imageBitmap);

ivAdminMenuEditorImage.setImageBitmap(BitmapHelper.convert(image));

}

}

Add New Menu:

NetworkHelper.execute(getApplicationContext(), new NetworkTaskDelegate() {

@Override

public final HttpURLConnection initConnection(URL url, Map<String, String> postData) throws Exception {

postData.put("action", "setMenuList");

postData.put("id", String.valueOf(id));

postData.put("name", name);

postData.put("categoryID", String.valueOf(categoryID));

postData.put("price", String.valueOf(price));

postData.put("image", StringHelper.isNullOrEmpty(image) ? "NULL" : image);

postData.put("isActive", isActive.toString());

return NetworkHelper.doPostRequest(url, postData);

}

@Override

public final void onSuccess(List<String> result) throws Exception {

startActivity(new Intent(AdminMenuEditorActivity.this, AdminMenuActivity.class));

Toast.makeText(getApplicationContext(), "Menu updated!", Toast.LENGTH\_LONG).show();

System.out.println("Menu updated!");

}

});

Delete Menu:

NetworkHelper.execute(getApplicationContext(), new NetworkTaskDelegate() {

@Override

public final HttpURLConnection initConnection(URL url, Map<String, String> postData) throws Exception {

postData.put("action", "deleteMenuList");

postData.put("id", String.valueOf(id));

return NetworkHelper.doPostRequest(url, postData);

}

@Override

public final void onSuccess(List<String> result) throws Exception {

startActivity(new Intent(AdminMenuEditorActivity.this, AdminMenuActivity.class));

Toast.makeText(getApplicationContext(), "Menu deleted!", Toast.LENGTH\_LONG).show();

System.out.println("Menu deleted!");

}

});

BitmapHelper.java

Convert Bitmap to String (or the reverse)

public static final Bitmap convert(String base64Str) throws IllegalArgumentException

{

final byte[] decodedBytes = Base64.decode(base64Str, Base64.URL\_SAFE | Base64.NO\_WRAP);

return BitmapFactory.decodeByteArray(decodedBytes, 0, decodedBytes.length);

}

public static final String convert(Bitmap bitmap)

{

final ByteArrayOutputStream outputStream = new ByteArrayOutputStream();

bitmap.compress(Bitmap.CompressFormat.PNG, 100, outputStream);

return Base64.encodeToString(outputStream.toByteArray(), Base64.URL\_SAFE | Base64.NO\_WRAP);

}

Part 2: Servlet (Tomcat)

Initialize MySql Database:

@Override

public final void init(ServletConfig config) {

try {

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

connection = DriverManager.getConnection("jdbc:mysql://localhost:3306/restaurant", "root", "");

isActive = true;

} catch (Exception ex) {

System.out.println(ex.getMessage());

}

}

Read Data from Android network task:

@Override

public final void doPost(HttpServletRequest request, HttpServletResponse response) {

try {

final PrintWriter writer = response.getWriter();

if (!isActive) {

writer.println("false");

writer.println("Database connection is unavailable. Please try again later.");

System.out.println("Database connection is unavailable. Please try again later.");

} else {

final String action = request.getParameter("action");

if (action == null || action.isEmpty()) {

writer.println("false");

writer.println("No action is defined.");

System.out.println("No action is defined.");

} else {

try {

final Method method = DatabaseManagerServlet.class.getMethod(action, HttpServletRequest.class, PrintWriter.class);

try {

method.invoke(null, request, writer);

} catch (Exception ex) {

writer.println("false");

writer.println(ex.getMessage());

System.out.println(ex.getMessage() == null ? "NullPoinerException" : ex.getMessage());

for (StackTraceElement item : ex.getStackTrace()) {

System.out.println(item);

}

}

} catch (Exception ex) {

writer.println("false");

writer.println("Unknown action is defined.");

System.out.println("Unknown action is defined.");

}

}

}

writer.close();

} catch (Exception ex) {

System.out.println(ex.getMessage());

}

}

Logic for all the MySQL Queries:

Get Category:

public static final void getMenuCategoryList(HttpServletRequest request, PrintWriter writer) throws Exception {

final ResultSet result = query("select \* from menu\_category order by Name");

writer.println("true");

while (result.next()) {

writer.println(result.getString(1));

writer.println(result.getString(2));

writer.println(result.getString(3));

}

result.close();

}

Insert Category:

public static final void setMenuCategoryList(HttpServletRequest request, PrintWriter writer) throws Exception {

final int id = Integer.valueOf(request.getParameter("id"));

final String name = request.getParameter("name");

final Boolean isActive = Boolean.valueOf(request.getParameter("isActive"));

final ResultSet result = prepare("select \* from menu\_category where Name = ?", name);

int fetchCount = 0;

while (result.next()) {

fetchCount++;

}

result.close();

if (fetchCount > 0) {

writer.println("false");

writer.println("Category name exists. Please try again.");

} else {

if (id == 0)

prepare("insert into menu\_category (Name, IsActive) values (?, ?)", name, isActive);

else

prepare("update menu\_category set Name = ?, IsActive = ? where ID = ?", name, isActive, id);

writer.println("true");

}

}

Delete Category:

public static final void deleteMenuCategoryList(HttpServletRequest request, PrintWriter writer) throws Exception {

final int id = Integer.valueOf(request.getParameter("id"));

prepare("delete from menu\_category where ID = ?", id);

writer.println("true");

}

Get Menu:

public static final void getMenuList(HttpServletRequest request, PrintWriter writer) throws Exception {

final ResultSet result = query("select menu.ID, menu.Name, menu\_category.ID, menu\_category.Name, menu.Price, menu.Image, menu.IsActive from menu, menu\_category where menu.CategoryID = menu\_category.ID order by menu\_category.Name, menu.Name");

writer.println("true");

while (result.next()) {

writer.println(result.getString(1));

writer.println(result.getString(2));

writer.println(result.getString(3));

writer.println(result.getString(4));

writer.println(result.getString(5));

writer.println(result.getString(6) == null ? "NULL" : result.getString(6));

writer.println(result.getString(7));

}

result.close();

}

Insert Menu:

public static final void setMenuList(HttpServletRequest request, PrintWriter writer) throws Exception {

final int id = Integer.valueOf(request.getParameter("id"));

final String name = request.getParameter("name");

final int categoryID = Integer.valueOf(request.getParameter("categoryID"));

final double price = Double.valueOf(request.getParameter("price"));

final String image = request.getParameter("image");

final Boolean isActive = Boolean.valueOf(request.getParameter("isActive"));

final ResultSet result = prepare("select \* from menu where Name = ?", name);

int fetchCount = 0;

while (result.next()) {

fetchCount++;

}

result.close();

if (fetchCount > 0) {

writer.println("false");

writer.println("Menu name exists. Please try again.");

} else {

if (id == 0)

prepare("insert into menu (Name, CategoryID, Price, Image, IsActive) values (?, ?, ?, ?, ?)", name, categoryID, price, image.contentEquals("NULL") ? null : image, isActive);

else

prepare("update menu set Name = ?, CategoryID = ?, Price = ?, Image = ?, IsActive = ? where ID = ?", name, categoryID, price, image, isActive, id);

writer.println("true");

}

}

Delete Menu:

public static final void deleteMenuList(HttpServletRequest request, PrintWriter writer) throws Exception {

final int id = Integer.valueOf(request.getParameter("id"));

prepare("delete from menu where ID = ?", id);

writer.println("true");

}

Get Account:

public static final void getAccount(HttpServletRequest request, PrintWriter writer) throws Exception {

final String name = request.getParameter("name");

final String password = request.getParameter("password");

final ResultSet result = prepare("select \* from account where Username = ? and Password = ?", name, password);

int fetchCount = 0;

while (result.next()) {

fetchCount++;

}

if (fetchCount > 0) {

result.first();

writer.println("true");

writer.println(result.getInt(1));

writer.println(result.getInt(4));

writer.println(result.getInt(5));

} else {

writer.println("false");

writer.println("Invalid username or password. Please try again!");

}

}

Register Account:

public static final void setAccount(HttpServletRequest request, PrintWriter writer) throws Exception {

final String username = request.getParameter("username");

final String password = request.getParameter("password");

final ResultSet result = prepare("select \* from account where Username = ?", username);

int fetchCount = 0;

while (result.next()) {

fetchCount++;

}

result.close();

if (fetchCount == 0) {

prepare("insert into account (Username, Password, IsAdmin, IsBanned) values (?, ?, ?, ?)", username, password, 0, 0);

writer.println("true");

} else {

writer.println("false");

writer.println("There is an existing username. Please try another name.");

}

}

Store Order:

public static final void setOrder(HttpServletRequest request, PrintWriter writer) throws Exception {

final int accountID = Integer.valueOf(request.getParameter("accountID"));

final int menuID = Integer.valueOf(request.getParameter("menuID"));

final int quantity = Integer.valueOf(request.getParameter("quantity"));

final String additionalRequest = request.getParameter("additionalRequest");

final long dateTime = Long.valueOf(request.getParameter("dateTime"));

prepare("insert into `order` (AccountID, MenuID, Quantity, AdditionalRequest, DateTime) values (?, ?, ?, ?, ?)", accountID, menuID, quantity, additionalRequest.contentEquals("NULL") ? null : additionalRequest, dateTime);

writer.println("true");

}