

# In-Store Click & Collect Shopping System for Asda's Home Shopping Department

## Table of Contents

Analysis .....	4
Project Background .....	4
Problems of the Existing System .....	4
Project Outline .....	5
Client.....	5
Research.....	6
Detailed Description of Existing System .....	6
Research of Requirements from the Client – Interview .....	7
Research of Requirements from the Client – Evaluation.....	8
Research of Programming Requirements .....	8
Breakdown of New Fastest Path Algorithm .....	11
Data Required for the System .....	12
Detailed Description of Proposed New System .....	13
Possible Problems Encountered.....	14
Sources and References Used in Research .....	14
Programming Objectives.....	15
Design .....	16
Overall System Design .....	16
Modular Design .....	17
Colleague Application .....	17
Customer Application .....	17
Data Requirements and Data Dictionary (Database Only) .....	17
Validation.....	19
Entity-Relationship Diagram .....	20
Planned SQL Queries .....	20
Planned Algorithms .....	21
Dijkstra's Fastest Path Algorithm .....	21
Create Small Graph Algorithm.....	22
UI Design .....	22
Class Definition Diagrams .....	23
Customer Application .....	23
Employee Application .....	23
Technical Solution .....	24

Customer Application .....	24
MainActivity .....	24
HomePage .....	27
BackgroundWorker .....	29
LoginFragment .....	32
RegisterFragment .....	33
ProductsFragment .....	35
BasketFragment .....	40
Employee Application .....	45
MainActivity .....	45
HomePage .....	47
BackgroundWorker .....	51
UncompOrdersFragment .....	55
ViewUncompOrdersFragment .....	58
CompOrdersFragment .....	63
ViewCompOrdersFragment .....	66
AccountFragment .....	67
NewAccountFragment .....	68
WeightedGraph .....	69
Layouts and UI Design – Customer Application .....	73
fragment_login.xml .....	73
fragment_register.xml .....	73
fragment_products.xml .....	73
fragment_basket.xml .....	74
drawer_menu.xml .....	74
Layouts and UI Design – Employee Application .....	74
fragment_login.xml .....	74
fragment_account.xml .....	75
fragment_new_account.xml .....	75
fragment_uncomp_orders.xml .....	75
fragment_view_uncomp_orders.xml .....	76
fragment_comp_orders.xml .....	76
drawer_menu.xml .....	76
PHP and SQL Database Queries .....	77
conn.php .....	77
custLogin.php .....	77
custRegister.php .....	77
empLogin.php .....	78

empRegister.php .....	78
retrieveBasket.php .....	78
addToBasket.php .....	79
deleteFromBasket.php.....	79
addOrder.php .....	79
checkOrder.php .....	79
createOrder.php .....	80
pickItem.php .....	80
retrieveUncompOrders.php.....	80
updateOrder.php .....	81
retrieveOrderProducts.php.....	81
updateName.php.....	81
retrieveCompOrders.php .....	82
retrieveProducts.php .....	82
Testing .....	83
Customer Application .....	83

# Analysis

## Project Background

Asda currently has a Click & Collect system that is implemented at most of its stores across the country. The Click & Collect system is used by customers who may not want or have the time to do a full shop themselves but would still like the guarantee of fresh food from their local store. Whilst the system proves very successful for the customers who use it, most employees who work in the home shopping department at Asda find it slow and inefficient, often causing a hold-up of orders that are behind schedule. My project is focused around this system, aiming to improve the entire home shopping system so that the efficiency of the system can be vastly improved, so that the list of orders from customers can be worked through and finished faster. The current system does not include a clear and easy to understand user interface, but rather a messy and unaesthetically pleasing interface with limited controls and functions. The devices use a server to communicate with the incoming orders from the online Click & Collect system. This often causes the devices to feel unresponsive to some commands from the user when server is under a lot of demand from multiple users, stopping employees from carrying out their job.

The server communicates to each device by transmitting a list of products that require 'picking' from the shop-floor. The list of products from each order only include a name and a UPC (Universal Product Code). The barcode scanner on the devices scans the product and compares the given UPC and the returned string of digits from the barcode to confirm that the correct product has been picked. No specific details about the product is given, such as its location.

Newer colleagues to the store would find it difficult to find specific products when given them within an order. This is obviously very inefficient for the home shopping department, as the time taken for each order should be reduced rather than increased.

## Problems of the Existing System

- Data retrieval is slow
- The system is inefficient for employees to use
- The UI is difficult to understand
- New colleagues find it difficult to learn
- No details about products are shown
- Completed orders cannot be traced back to colleagues who finished them
- The program often crashes on the devices

## Project Outline

The aim of this project is to vastly improve the system used at Asda, boosting morale, reducing stress and improving the efficiency and usability of it. The new system would include a better user interface for the colleagues using it, allowing new employees to understand and learn how to use the system much faster than previously possible. It should use a similar method of communicating over a server, but instead providing more resources for the application to use, rather than transmitting plain text lists of orders to each device with little functionality for the users.

It should offer two applications at the end of development, one side of the system for employees to access and use to work the orders from customers, and another side that allows customers to create orders for the employees to work on behalf of them.

The most important change to the system should include an implementation of a fastest path algorithm, giving the employees an advantage of speed when picking the items from the shop-floor. This added efficiency should reduce the number of delayed orders in the home shopping department, therefore increasing productivity.

The system should also provide the colleagues with clear, helpful information about the product, such as its location on the shop-floor. This would allow newer employees who may not know the layout of the store, or perhaps employees who are unsure of a general product's location, to find and pick specific products.

## Client

As mentioned previously, my intended clients are those working in the home shopping department, and the customers of Asda. Not all the colleagues and users are not all technologically adept to deal with issues when they occur. This means that I should design my system considering the needs of all the colleagues who work in the department, including features that all users can understand.

To receive feedback for the different aspects of this project, a colleague from that department has agreed to help me, providing me with invaluable research for this new system. I will create a series of questions to use to help interview the colleague.

# Research

## Detailed Description of Existing System

The existing system for colleagues to use is implemented on the Zebra TC70 and TC70x Series Touch Computer, which uses an Android Operating System on a 4.7" Display. The devices use different versions of Android, one running Lollipop 5.1, and one running Marshmallow 6.0. I can use these key details to program my new system specifically to these devices. The devices use a combined barcode and QR code scanner to verify that the product being searched for is the correct one.

At the beginning of the day, each colleague in the hope shopping department is given one of these devices. The current system does not use a log-in system, so supervisors and managers cannot view who specifically is picking each order.

The system uses a server to send lists of products, stored as key-value pairs for the employee to collect, containing a UPC as the key and product names as data, imported directly from their online Click & Collect system.

The lists do not contain any information about the product or where its stored. This is not very effective for the users since the employees often do not know where a product is located, so a lot of time is wasted whilst searching for a specific product.

Once a product is found, it is removed from the list. Once the list is empty, it retrieves a new list from the server, until all users have finished the collection of all the orders on the server.

However, the system for the users is implemented as a standard mobile application featuring their online Click & Collect system. This system allows users to create new orders for in-store colleagues to pick on behalf of them so that the customer can pick the shopping up.

Each user signs into the system with their unique logins, storing some of their private data such as name, age and address. This data is used then when the users wish to collect their order afterwards.



*Zebra TC70x Series Touch Computer*

## Research of Requirements from the Client – Interview

As part of my research into the user requirements, I interviewed a section leader in the home shopping department at Asda, who has had a lot of experience using the system. I asked Jasmine P, shortened to JP, about the UI, functionality and efficiency of the old system, and what she would want to have in a new and improved system.

*Me: How do you feel about the current picking system used in the home shopping department?*

*JP: At the moment, I feel like there is a lot to improve. The entire system is stressful and messy.*

*Me: What do you mean by 'messy' exactly?*

*JP: Well right now, when new orders drop<sup>1</sup>, the products aren't organised in any order, making it difficult to locate when you're going around the store. For example, an order could send you to get eggs, then send you to the other side of the shop-floor for candles, only to go back again to pick up tea and biscuits.*

*Me: So how would you like this to be improved? Would you like the order to come in pre-ordered with the products in the most efficient order to collect?*

*JP: I think that would be a very good idea, since a lot of pressure is already placed on us by the general store manager to get orders done quickly.*

*Me: How would you like the user interface to be improved?*

*JP: I think maybe adding a nicer theme to the app would be preferable - I think the current one looks a bit old fashioned and out-dated.*

*Me: How would you feel about extra functionality to view important details about the product, such as seeing its location within the store?*

*JP: It would be very useful to see at least what aisle the product is on, since sometimes we are often confused which aisle something is on if we are unfamiliar with it. However, in the other departments, they have access to a full locating app on the guns<sup>2</sup> to find any product they would like. I think this would be helpful to include on our new system also.*

*Me: Is there anything else you think I should try including in the new system?*

*JP: I think a better log-in system could be beneficial, so that the managers could identify which colleague finished which orders, so they can be referred to if needed. There have often been times when an order has been finished incorrectly, but we cannot trace which colleague picked it, so we cannot offer them feedback or help them improve for the future.*

*Me: So, would having different levels of access through the application also be a desirable feature?*

*JP: Absolutely. I think it should be included so that managers, other section leaders and supervisors would have access to change or overview the system and the information stored about the product if needed.*

*(References):* 1. Drop – when new orders come in through the network  
2. Gun – a name for the devices used to scan products and pick orders

## Research of Requirements from the Client – Evaluation

From this interview with Jasmine, it is clear that the current system is slow and not user-friendly. The important user requirements I can extract from this is that efficiency will be the forefront of the new system, which would allow the colleagues in that department be able to work in a better and generally less-stress environment. Including an authorised log-in wall would add security to the program, protecting the system and the data stored in the large database. It would also allow extra functionality to be given to specific or generalised groups of users such as managers and department leaders to be assigned higher levels of access to the system. As well as allowing users to log-in, I believe it would be beneficial for administrators to add or delete colleague accounts.

However, if I am to include log-in functionality, then I need to carefully consider encryption or hashing algorithms that would sufficiently protect the data from being intercepted during communication with the server. This is very important to consider based on the amount of private data that could be stored in the database about the users and orders.

## Research of Programming Requirements

I must program my system in Java, using a database back-end to access and change details and information about certain products and users using an online web-hosting server with MySQL to implement database functions within the system. Java will be used so that the application will be compatible with the Android devices that will be used in the end-system. Android Studio is the best option to begin programming for an Android device. I will need to ensure that the developed solution for colleagues will be compatible with both Android 5.1 and 6.0, as both versions are used on different devices, as well as a multi-platform support for many Android devices for the customer solution. I will also need to consider SQL and PHP to implement by database.

A cross-table database could be used so that all the data about a product, including its name, UPC and location within the store can all be accessed easily within one large file. I will use different levels of access within the program so that only authorised users can access or make changes to the large database, including changes to product or user details.

I will be using a server as a central storage location for the large database so that multiple devices can access the data at once. This database will be accessible by all colleagues however access will be limited based on their level of authorisation. The server will add an advantage to the system so that the data can be accessed wirelessly as long as the devices have a network connection. Using a central server to store the data would allow for a reduced weighting on the amount of local storage space required on each device, therefore vastly reducing the amount of storage space required.

As well as including functionality to have multiple users to have unique log-ins to the server, I need to ensure that the program can securely authenticate these users, ensuring that any private data that is communicated is sufficiently encrypted. I should include a database that includes all the users stored in one place, stored securely with encryption on the passwords, so that even if the server was to be attacked and accessed without authorisation, then the user's data would still be stored securely. The authorisation method should include a hashing algorithm that creates a key for each user that can be compared with the hashed value stored within the database. As well as including a secure hashing algorithm, I should consider how data is going to be manipulated when communicating over the server. To avoid private data being accessed, no plaintext sensitive data should be stored on a semi-permanent basis or transmitted at any point during the applications run time.

There are a number of secure hashing algorithms. Some hashing algorithms are more secure than others, however with both advantages and disadvantages. MD5 is a very popular hashing algorithm that produces a 128-bit hash, however there have been instances where this hash has been attacked and failed. Some algorithms produce larger hash outputs which is often more secure, although for



some algorithms, data has still been attacked, such as SHA-0 and SHA-1. SHA-2 and SHA-3 are a couple of the most secure functions. However, since my database will be storing large amounts of data for all the items within the large database, I need to ensure that my hash output is small enough to efficiently store in a table without using unnecessary space.

The Data Protection Act 2018 has strict regulations on what is allowed to be stored about a user, and who is allowed to access it. My user database should store information about the user, including their UserID, employee number if applicable, date of birth, their full name, their email address (if applicable) and their username and password. The table would store information for both customers and employees, so all customer information must be hidden to all users. Only management position users will be able to access limited information about the employees' accounts. The table will store Boolean values to determine whether the user is management, standard colleague or customer.

I will need to create a graph to store all the data about the product with its location. The graph will be stored as a multi-table database, storing all data about each product, using SQL queries to search for and locate product within the database from products from the orders stored on the server. The database will also be stored on the server, so that the device does not have to store the data locally. This should improve the efficiency of the product since the devices used have limited storage space.

By using the server to store both the orders and the large database, it allows for multiple devices to use and query the system at once, without devices storing identical copies of the database in internal storage.

The graph will need to include data about each product, including its ProductID, name, UPC and location. As mentioned previously, the different levels of access will allow fully authorised users to edit the names, orders and locations, whereas lower levels of access will only be authorised to view orders and details about the product.

I need to ensure that the functionality of the application for the user is clear, and that all parts of the application should be easy to understand. I will most likely use a simple selection system for the users to create their order, structured with a drop-down list of every product, along with a text box for the quantity they require. For every product they want, customers can click a small 'New' button that will add another product to the current order. After the user clicks 'Submit', their order is added to the database, and their OrderID and OrderDate from the Orders table will be returned.

Since this new system is orientated around efficiency, it is crucial to include some shortest path algorithms so that the system can be used to improve the effectiveness of the system. Some shortest path algorithm solutions include Dijkstra's, Bellman-Ford, A\* Search, Floyd-Warshall and Johnson's. These algorithms work in different ways to find the shortest path between pairs of nodes. Since my efficiency evaluation will be focused around time taken, and not distance travelled, I will not be needing to consider algorithms specifically designed for negative weightings or directed graphs.

Because my system will be including a graph where various nodes would be required to be passed, such as specific aisles on the shop-floor, any of the shortest path algorithms I choose needs to be altered to allow for the specific requirements of the returned shortest path. If alterations weren't made, then the shortest route through the shop-floor would be an example of an intractable problem. This means that there is no algorithm that can solve the problem efficiently. Therefore, a solution to this could either be a brute-force method to find the fastest path or use a combination of other various fastest path algorithms and searches and take shortcuts whilst programming the solution so that although it may not return the perfect path every single time, it should still reduce the processing time taken for my solution.

Rather than designing an algorithm that tests every single possibility of path combinations of a large graph, I have decided to design a new algorithm that can find the fastest path through a sub-graph of products from an order. It will use a combination of Dijkstra's fastest path algorithm and a

breadth-first search. I have decided to use Dijkstra's algorithm rather than a brute force method to reduce the time-complexity of the problem. A brute-force method could cause the program to become unstable and slow, caused by the exponential time-complexity of attempting every single possibility through a 10-30 node graph.

Instead, Dijkstra's shortest path algorithm has a worst-case performance of  $O(|E| + |V|\log|V|)$  and the breadth-first search has a worst-case performance of  $O(|E| + |V|)$ , where  $|E|$  is the number of edges between nodes and  $|V|$  is the total number of nodes within the graph.

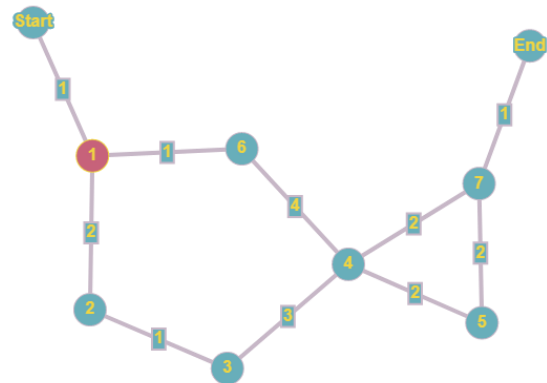
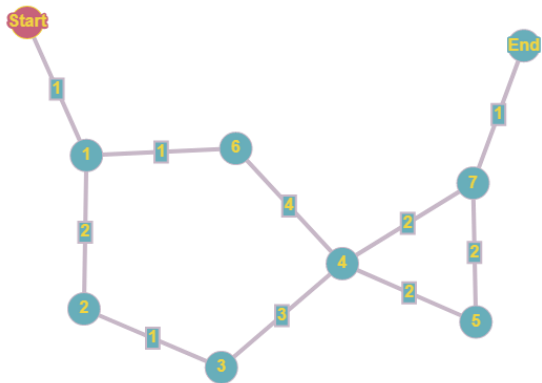
My algorithm for this solution will implement a combination of both algorithms, in order to provide a shortlist of shortest paths between nodes. This shortlist can then be narrowed down to a single path using a simple method to compare each value in the shortlist to produce a single path.

When a new order drops in the system, each product within the order will be compared against the large graph database of products. From this database, another sub-graph will be extracted of all the products in the order. In order for the system to function correctly, two more nodes will be added to the graph, which is the start and the end of each path, which directly represents how the shop-floor is laid out within the Asda store. These two nodes will not hold any information except a name of "Start" and "End", just so that when the program is being managed or adjusted in the future, the code and its features are easy to understand.

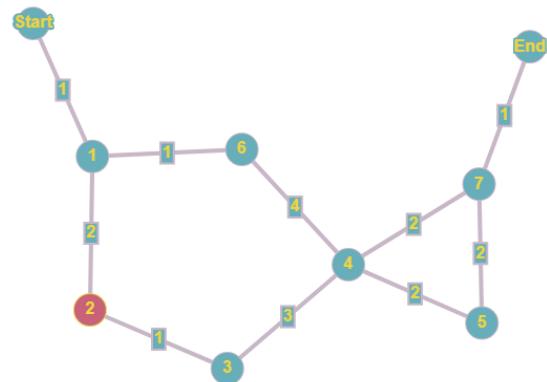
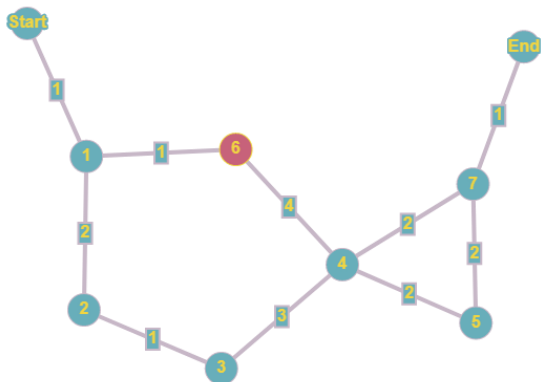
All of the orders will start from the same "Start" node, using a breadth-first search to locate all of the start node's unvisited connected edges. The weightings on each edge, representing the time between each location on the graph, can be compared against each other. The smallest node will be added to the first stack in a dynamic list, so that the path can be traceable afterwards. If there are multiple edges with the same smallest value, then the current stack will be duplicated into another index in the list, and the algorithm can carry on with the process for each stack. If the algorithm reaches a point where the only unvisited edge is the "End" node, but not all the products in the order have been picked, then Dijkstra's algorithm will be used for each unvisited node in-turn, returning the smallest distance between the current node and any other node. This then repeats until all of the nodes have been visited, and Dijkstra's algorithm has been used from the last node to the end node.

The reason I think that the orders that come in should be represented as sub-graphs is to abstract all the unnecessary detail from the rest of the graph database, which may cause the program to work through the problem slower, or even crash if the main memory of the device is not large enough to hold all the necessary data about the graph and execute the program simultaneously.

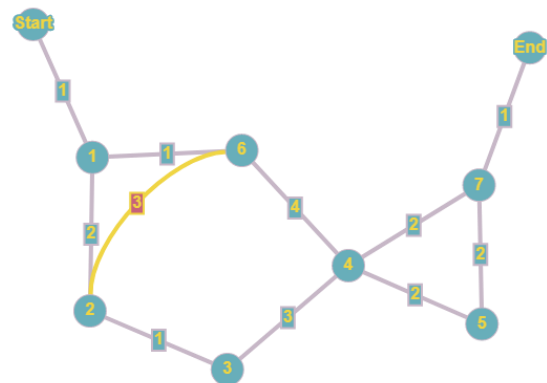
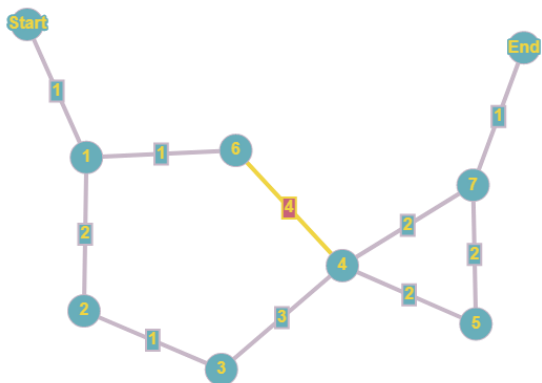
### Breakdown of New Fastest Path Algorithm



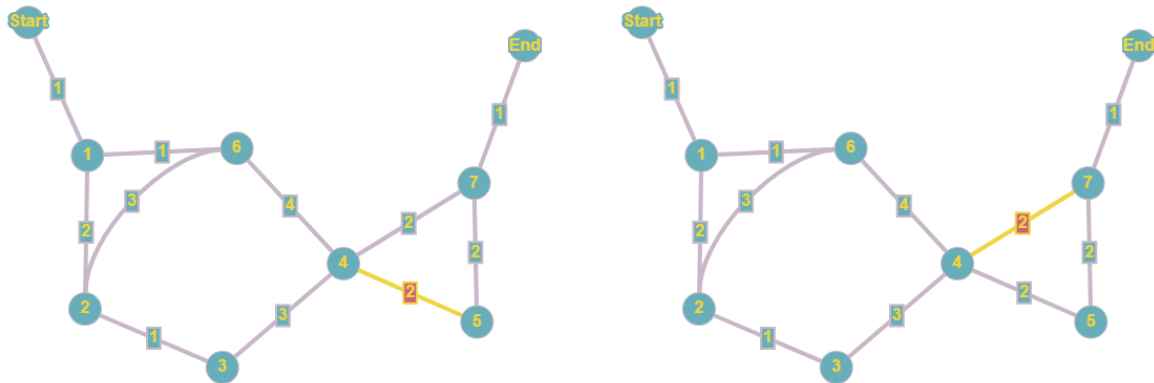
The start node is always selected first. A breadth-first search is then used to identify and locate all surrounding connected nodes. In this case, there is only a single connected node, "1". Therefore the name of this node is added to a stack of visited locations.



Next, the searching algorithm will be re-executed, comparing the weightings of the next two connected edges on the graph, returning the values of 1 and 2. Because "6" has the smaller distance, it will be traversed first, and the stack of visited nodes will contain "Start", "1" and "6".



After, "6" has been visited, it the algorithm will now execute another breadth-first search to locate surrounding nodes that have not been visited. When a breadth first-search is used from "6", the algorithm discovers that the path from "6" to "2" (total edge weighting of 3) is shorter than the path from "6" to "4" (total edge weighting of 4). Therefore, the graph is traversed backwards along the graph to "2". It is then added to the stack.

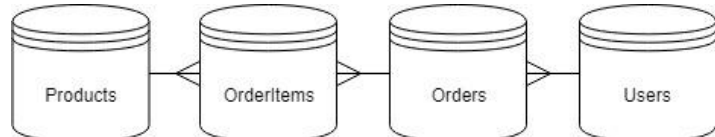


If the traversal of the graph reaches a point where there are two equal weightings on edges, then the current stack will be duplicated and added to the dynamic list. Each stack is then added to, beginning from alternate vertices. In this example, the current stack would return {"Start", "1", "6", "2", "3", "4"}. The two vertices "7" and "5" cause an issue, so the stack would copy itself and execute the rest of the algorithm, producing two stacks of: {"Start", "1", "6", "2", "3", "4", "7", "5", "End"} and {"Start", "1", "6", "2", "3", "4", "5", "7", "End"}. These two results return total weightings of 16 and 14 respectively. Therefore, the second path is more efficient.

If a node reaches a point where all surrounding nodes have been visited, then a breadth first search is used to locate each unvisited node. All the unvisited nodes are input into Dijkstra's fastest path algorithm, returning which node is the fastest to visit. This repeats until all nodes have been visited. Dijkstra's algorithm is then used between the node at the top of the stack and "end". All total distances are stored in a parallel dynamic list as the dynamic list of stacks.

## Data Required for the System

For this system, I will be using 4 tables within one large database to store and manipulate the data required from the users, orders and products. Each table will store relevant information about the product, allowing tables to be easily cross-referenced.



For the Products database, each record will contain information about one specific item, including its unique ProductID, unique Name, unique UPC, and Location. Only storing relevant information should reduce the total storage space required to store the database on the server. The OrderItems table would contain a large list of products on all the orders, indexed with an OrderItemID and OrderID so that the products can be cross-referenced back to a user's order. The Orders table will be used to store abstracted information about an order, such as its OrderID, UserID, EmployeeID, OrderDate and an OrderCompleted check. This information can be used to order the uncompleted orders by date to prioritise the oldest orders. The Users table will contain all the users on the system, including both employees and customers. Data such as UserID, EmployeeNumber (if applicable), FullName, DateOfBirth, Email (If Applicable), Username, Password, and two UserAdmin and UserCustomer checks.

## Detailed Description of Proposed New System

The system I am aiming to successfully develop should be 100% oriented around efficiency. As well as designing efficient functionality, I should also ensure that the structure of my code is also efficient so that the devices that they will be running on should execute the program quickly without any issues. This means that I should use object-oriented programming to allow my program to be efficiently and appropriately structured so that as well as the program having a more efficient execution process, it will also be easier to understand and manage in the future.

The system will include a collection of devices running Android 5.1 and 6.0, connected to a network that can access resources such as graph databases and an input of orders. I will not be aiming to any functionality of creating new orders, since this is managed by another system that runs of the Asda website.

The devices will run a small program, implementing a user-friendly interface allowing new employees to learn and understand the system much easier. This interface should allow colleagues to log-in with their own personal, unique credentials, so that any orders that they work can be reviewed by management. Some colleagues, such as section leaders and managers should also have a higher level of access that can give them full access to the total list of orders and the product database, including abilities to change details about their location.

The authentication method I am going to use will ensure that any data is encrypted before any communication. The log-in service should include sufficient securing methods to ensure that no private data is accessible on the device. Only the higher level of authorisation can manage other colleague's data, however this will be limited to viewing existing user's username and employee number, with permissions to create new and delete existing accounts, and reset existing account's passwords to a temporary default. Employees are expected to complete the rest of the data on the first log-in to the system. The hashing algorithm that my system will use will be an MD5 hash, since it is easy to develop yet difficult to crack. The hashed output is only 128 bits, therefore reducing the total amount of storage space used by the database when storing the passwords.

When employees are picking products from an order, they will be provided with the full name of the product including its UPC, location and the number of products needed for each pick. Including this functionality is essential for this new system and should allow all employees to complete orders faster. Each product will be stored with an aisle number, ranging from 1-20. Each product will have a specific position number, ranging from 1000+ or 2000+, depending whether the location is located on the left or right of the aisle. For example, a product could be located on aisle 6, in the 17<sup>th</sup> position on the left, represented by [06-1017].

When a new order drops on the system, then a device on the network is automatically assigned it, removing it from a list of 'available' orders. When the device receives the order, each aisle number from each product in the order is extracted and queried against the graph database. From this, a sub-graph is created of 'Start', 'End', and the rest of products represented in graphical form. Each node on the graph will represent one aisle, so that the algorithm described above can generate a shortest path between all of the aisles required. The shortest path will then be used to arrange all the products in a list, pushing all the products in the first node to the top of the list to be picked, and the last nodes to the bottom. The devices will access the graph database over the network to avoid the devices having to store the entirety of the database locally. This would also allow changes to be made universally without having to modify every instance of the database on each device. This would be inefficient since there are over 50 of these devices in the home shopping department.

I will be using the online WAMP web-hosting server to store all the database information, so that the devices can be completely wireless during use. When each order has been finished, an appropriate message will be presented, and the next order from the system will be fetched automatically if possible. If there are no more orders on the system, the system should display an appropriate

message, periodically checking the server for any new orders. Communication between the devices and the server should avoid any two devices working the same order. This may be achieved by assigning each order to a single active device. A device would be assigned 'Active' status if a user is logged in. A user will be logged out after 5 minutes of inactivity and any unfinished orders would be re-added to the orders list on the server.

## Possible Problems Encountered

I currently don't have access to the system, devices, or the large database that Asda has, which stores all the information about the products in-store. Since I don't have access to these devices, I will test my application on an emulator and any other Android 5.1 and 6.0 I can get access to.

I do not have access to the thousands of products that Asda have in stock. For this project, I will include a limited number of products, with UPC codes generated with a hashing algorithm that uses the product name to create an almost unique 12-digit number. Although this number may not be entirely accurate, it should provide evidence that the system works with any number generated.

I will also not have access to the barcode scanner that is used in the devices used at Asda, so for this project I will just implement a "collected" button that would replace the barcode scanner, since the scanner is not essential for either system. This shouldn't detract from the functionality of the application since the button could still be used as a secondary function if the final system was to be used in the target devices.

Since I don't have access to the large database of product that Asda has, I will have to limit my project to a simplified version of what would be the final system. To achieve this, I will create a smaller database of products that I can still test successfully without needing access to all the data that Asda has.

## Sources and References Used in Research

<https://www.zebra.com/gb/en/products/mobile-computers/handheld/tc7x-touch-computer-series.html> - This allowed me to research the exact specifications of the target device so that my developed solution will work efficiently, ensuring that it will be developed for the correct operating system version.

[https://en.wikipedia.org/wiki/Graph\\_database](https://en.wikipedia.org/wiki/Graph_database) - This allowed me to research the various uses of graph databases and how to implement them.

<https://graphonline.ru> - This allowed me to create and visualise graphs to test different algorithms with, including Dijkstra's fastest path, breadth-first and depth-first searches, allowing me to find the most efficient algorithms for my problem.

[https://en.wikipedia.org/wiki/Secure\\_Hash\\_Algorithms](https://en.wikipedia.org/wiki/Secure_Hash_Algorithms) - This allowed me to research different hashing algorithms that are used to hide sensitive data and authenticate and validate data.

<https://www.wampserver.com> - This is the free online web-hosting platform I will be using to store the database.

## Programming Objectives

1. User-Friendly Interface
  - a. Attractive Colour Scheme
  - b. Good Functionality for Colleague Application
    - i. Log-In
    - ii. Exit
    - iii. View Uncompleted Orders
    - iv. View Completed Orders with Details (Administrator Access Only)
  - c. Good Functionality for Customer Application
    - i. Log-In
    - ii. Exit
    - iii. Create New Order
    - iv. View Current Basket
  - d. Clearly Displays Product Information for Colleagues
    - i. Name
    - ii. Universal Product Code
    - iii. Location
  - e. Clearly Displays Most Efficient Order of The Picks for Colleagues
2. Efficient Use of Searching and Fastest Path Algorithms
  - a. Creates A Sub-Graph of Required Products for A Specific Order
  - b. Uses Proposed Algorithm to Traverse the Graph and Create Shop-Floor Path
  - c. Orders the Products in The Pick List for The Colleagues
3. Uses A Server to Store the Database for The System
  - a. All Data Is Stored Within A Single Database
  - b. The Database Can Be Wirelessly Accessed and Modified with Sufficient Permissions
4. Sufficient Security Measures
  - a. No User Will Have Full Access to Any Other User's Data
  - b. No Private Data Should Be Stored or Transmitted in Plaintext
  - c. All Users Will Have to Log in Before They Are Permitted to Access the Program
    - i. Colleague Accounts Can Be Accessed and Edited by Administrator Accounts
    - ii. Customer Accounts May Not Be Accessed or Edited by Any Other User
  - d. Passwords Must Be Encrypted
5. The Program Must Be Robust
  - a. The Program Should Remain Equally Functional with All Android Operating Systems
  - b. The Program Should Respond Well to Errors
    - i. The System Should Not Crash or Behave Unexpectedly
    - ii. Users Should Be Prompted with Solutions When Errors Occur

# Design

## Overall System Design

Input	Processes
<ul style="list-style-type: none"> <li>• Add New Colleague Accounts</li> <li>• Add New Customer Accounts</li> <li>• Add New Orders to the Database</li> <li>• Submit Login Information</li> <li>• Change Username and Password</li> <li>• View Uncompleted Orders</li> <li>• View Completed Orders</li> <li>• View Product Information</li> </ul>	<ul style="list-style-type: none"> <li>• Add New User to Database</li> <li>• Add New Order to Database</li> <li>• Determine Fastest Path Using Algorithm</li> <li>• Hash Private Data in the Database</li> <li>• Validate User Information</li> <li>• Retrieve Uncompleted Orders from Database</li> <li>• Retrieve Completed Orders from Database</li> <li>• Retrieve Product Information from Database</li> <li>• Retrieve Users from Database</li> </ul>
Storage	Output
<ul style="list-style-type: none"> <li>• Database Tables             <ul style="list-style-type: none"> <li>• Products</li> <li>• OrdersItems</li> <li>• Orders</li> <li>• Users</li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li>• Ordered List of Products on Each Order Determined by Fastest Path</li> <li>• Display of Product Details</li> <li>• Display of Order Details</li> </ul>



## Modular Design

I need to design the way in which my mobile applications will function, ensuring the way my applications are designed is easy to navigate through by all users.

### Colleague Application

- Authentication
  - Login
    - Username
    - Password
- Edit Account
  - Change Username
  - Change Password
- Create New Colleague Account
  - Employee Number
  - Full Name
  - Username
  - AccessLevel
- View Uncompleted Orders
  - Start New Order
- View Completed Orders (Administrator Only)
  - View OrderID
  - View OrderDate
  - View UserID and Name
  - View EmployeeID
- Log Out

### Customer Application

- Authentication
  - Login
    - Username
    - Password
  - Registration
    - Full Name
    - Date of Birth
    - Email Address
    - Username
    - Password
- Edit Account
  - Change Username
  - Change Password
- Create New Order
  - Product Selection
  - Submit Order
- Log Out

## Data Requirements and Data Dictionary (Database Only)

Field Name	Data Type	Field Purpose	Field Size	Example Data
UserID	Integer	Uniquely Identifies Each User	5	275
EmployeeID	Integer	Uniquely Identifies Each Employee	10	1016847392
Full Name	String	Stores the Full Name of Each User	255	Adam Rutherford-Shaw
Email Address	String	Stores the Email Address Used by Each User	254	adam@email.com
Date of Birth	Date/Time	Stores the Date of Birth of Each Customer	DD/MM/YYYY (8 Figures)	19/11/1976
Username	String	Used to Validate Each User	20	adamrutherfordshaw
Password	String	Used to Validate Each User (Hashed)	32	482c811da5d5b4bc6d497ffa98491e38
Customer	Boolean	Used to Identify System Customers	True/False	1

Administrator	Boolean	Used to Identify System Administrators	True/False	True
OrderID	Integer	Used to Uniquely Identify Each Order, and to Reference OrderItems to Orders	5	106
Submitted	Boolean	Used to Identify Whether the Order has Been Sent	True/False	True
OrderDate	Date/Time	Used to Prioritise Each Order	YYYY/MM/DD (8 Figures)	02/07/19
CustomerID	Integer	References UserID in Users Table	5	275
ColleagueID	Integer	References EmployeeID in Users Table	10	1016847392
Completed	Boolean	Used to Identify Whether the Order has Been Picked	True/False	False
ProductID	Integer	Used to Identify Each Product	5	802
UPC	Integer	Used to Uniquely Identify Each Product from the Manufacturer, and to Reference OrderItems to Products	10	5056291048
ProductName	String	Used to Generally Identify the Product	30	Beans
Aisle	Integer	Stores Where the Product is on the Shop-Floor	2	16
Location	Integer	Stores Where the Product is on the Aisle	4	1065
OrderItemsID	Integer	Uniquely Identifies Each Product in the OrderItems Table	5	338
Quantity	Integer	Stores How Many Items Are Required	2	1
Picked	Boolean	Stores Whether the Specific Item Has Been Picked from the Order	True/False	False

## Validation

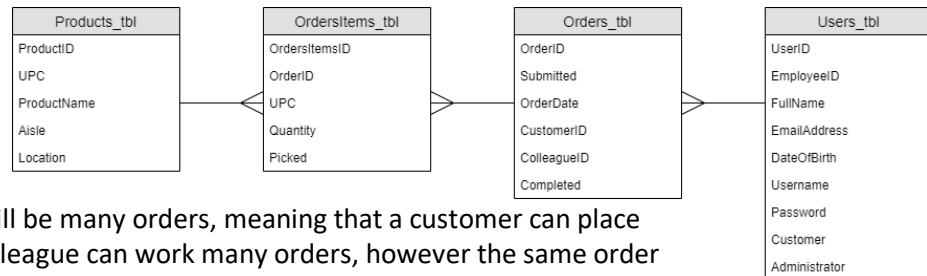
I need to ensure that the data entered by the user will be valid and will not cause any problems when entered into the database. I need to implement validation checks to ensure that the data entered is valid. The table below represents all of the data that could possibly be input by the user.

Field Name	Validation Checks	Description	Error Message
EmployeeID	Type - Integer Format - 10 Characters Uniqueness	Invalidates an EmployeeID if it is More or Less Than 10 Characters and Not Unique	"Please Enter a Valid EmployeeID"
Full Name	Range - $2 \leq \text{Full Name} \leq 255$ Format - No Integers	Invalidates Name if it Contains Integers	"Please Enter a Valid Name"
Date of Birth	Type - Date/Time Format – YYYY/MM/DD	Invalidates Date of Birth if it is Not in Correct Format	"Please Enter a Valid Date of Birth"
Email Address	Format - (name@domain.com)	Invalidates Email if Not in Correct Format	"Please Enter a Valid Email Address"
Username	Range - $8 \leq \text{Username} \leq 20$ Uniqueness	Checks Username is Between 8 and 20 Characters and Unique	"Please Enter a Valid Username" "Username Already Taken - Please Choose Another"
Password	Range - $8 \leq \text{UnhashedPassword} \leq 20$	Checks Plaintext is Between 8 and 20 Characters	"Please Enter a Valid Password"

## Entity-Relationship Diagram

Each table in the database will be linked to one another using foreign keys to relate the records from each table to each other. Separating the data into different tables allow the system to be managed more easily and allows for tables to be better organised with fewer fields per record.

In my system, the tables should relate to each other in the way this diagram represents.



For each user, there will be many orders, meaning that a customer can place many orders, and a colleague can work many orders, however the same order cannot be split between multiple colleagues or customers.

The Orders table stores information about each order, including the date ordered, the customer who ordered it, and various other key details about the order. Each order will contain many products; however, this has been simplified to having many OrderItems per order. Each OrderItems record will have a single product, but a single product could be on multiple OrdersItems records.

## Planned SQL Queries

I am planning on using SQL to query my database through my application. I will be using SQL to SELECT, INSERT, UPDATE and DELETE a number of records repeatedly from the four tables in my database.

Here are a few planned SQL statements that I will use within my program:

```
SELECT userid FROM users_tbl WHERE username LIKE '$username'
AND password LIKE '$password' AND customer LIKE '1';
```

```
INSERT INTO users_tbl (fullname, emailaddress, dateofbirth,
username, password, customer) VALUES ('$fullname', '$email',
'$dateofbirth', '$username', '$password', '1');
```

```
SELECT products_tbl.productname, ordersitems_tbl.upc, quantity
FROM products_tbl RIGHT JOIN ordersitems_tbl ON
products_tbl.upc = ordersitems_tbl.upc RIGHT JOIN orders_tbl
ON ordersitems_tbl.orderid = orders_tbl.orderid WHERE
orders_tbl.orderid = '$orderid' AND submitted LIKE '0';
```

```
UPDATE orders_tbl SET submitted = '1', orderdate = '$date'
WHERE orderid LIKE '$orderid';
```

```
DELETE FROM ordersitems_tbl WHERE orderid LIKE '$orderid' AND
upc LIKE '$upc';
```

## Planned Algorithms

### Dijkstra's Fastest Path Algorithm

This algorithm will define how the fastest path between all the points will be calculated.

```
FUNCTION fastPath (v1)
  FOR i ← 1 0 to 35
    dijkstraFastPath[i][0] ← ((v1 + i) MOD 36)
    dijkstraFastPath[i][1] ← 2147483647
    unvisitedList.add(dijkstraFastPath[i][0])
  ENDFOR
  dijkstraFastPath[0][1] ← 0
  WHILE NOT (unvisitedList.isEmpty)
    closestV ← 0
    smallestW ← 2147483647
    FOR I ← 0 to 35
      IF (dijkstraFastPath[i][1] < smallestW) AND
        (unvisitedList.contains(dijkstraFastPath[i][0]))
        closestV ← dijkstraFastPath[i][0]
        smallestW ← dijkstraFastPath[i][1]
      ENDIF
    ENDFOR
    unvisitedList.remove(closestV)
    FOREACH x in adjacencyList[closestV]
      IF unvisitedList.contains(x.vName2)
        alt ← smallestW + x.weight
        IF v1 > x.vName2
          IF dijkstraFastPath ((x.vName2) +
            36) - v1][1] > alt
            dijkstraFastPath[((x.vName2) + 36) v1][1]
              ← alt
            dijkstraFastPath[((x.vName2) + 36) -
              v1][2] ← x.vName1
          ENDIF
        ELSE IF v1 < x.vName2
          IF dijkstraFastPath[(x.vName2) - v1][1] > alt
            dijkstraFastPath[(x.vName2) - v1][1] ← alt
            dijkstraFastPath[(x.vName2) - v1][2] ←
              x.vName1
          ENDIF
        ENDIF
      ENDIF
    ENDFOR
  ENDWHILE
```

## Create Small Graph Algorithm

This algorithm creates a smaller sub-graph from the large graph of the store.

```
FUNCTION createSmallGraph (graph)
  A ← 0
  FOREACH v in requiredVList
    fastPathArray = graph.FastPath(x)
    FOR I ← 1 to 35
      IF requiredVList.contains(fastPathArray[i][0])
        newGraph.addEdge(fastPathArray[i][1],
          fastPathArray[0][0], fastPathArray[1][0],
          false
        a ← a + 1
      ENDIF
    ENDFOR
  ENDFOR
  requiredVList.remove(35)
  tempFastestPath ←
    NewGraph.calculateFastPathAroundStore(requiredVList)
```

## UI Design

For both of my applications, I plan on having a modern, and easy-to-use user interface. With my application being designed for Android, this design should fit in with other Android designs. This includes light colours, mainly white, blue and green.

Many newer Android programs feature a sliding navigation bar, allowing users to access pages more fluidly than having to go to a main menu or selection page when switching between fragments. I can achieve this modernistic UI by using instances of `FragmentManager`, `Drawer` and `NavigationView`, which will allow me to integrate different parts of my application into one small navigation drawer. My navigation drawer should include navigation to the different fragments I have in my program.

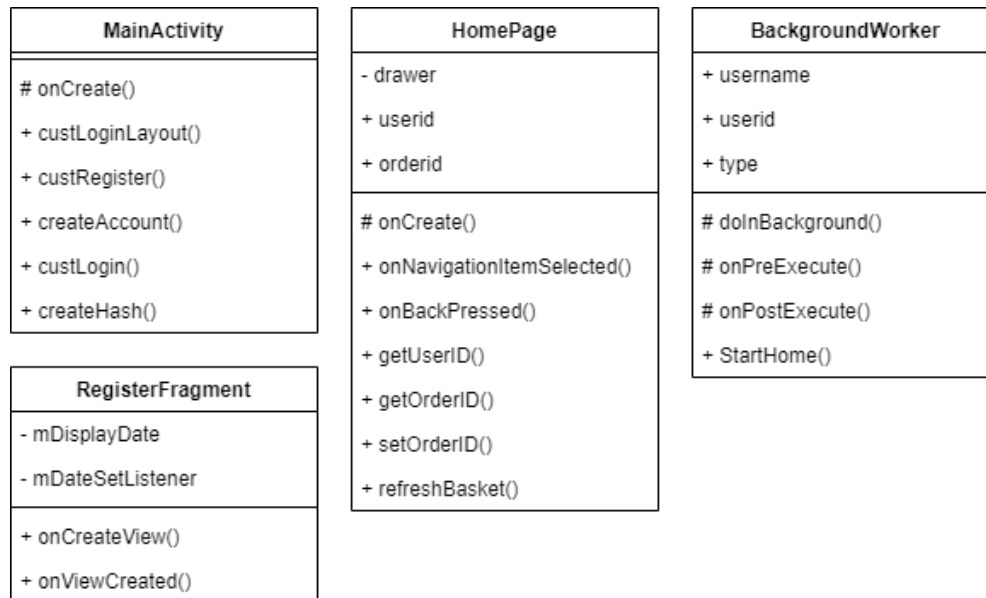
For the customer application, this should include a way to view the full list of products they can order, a way to view their current basket, as well as a way to log out of the application and return to the log-in page.

For the employee application, my navigation menu should include uncompleted orders, completed orders, and account settings.

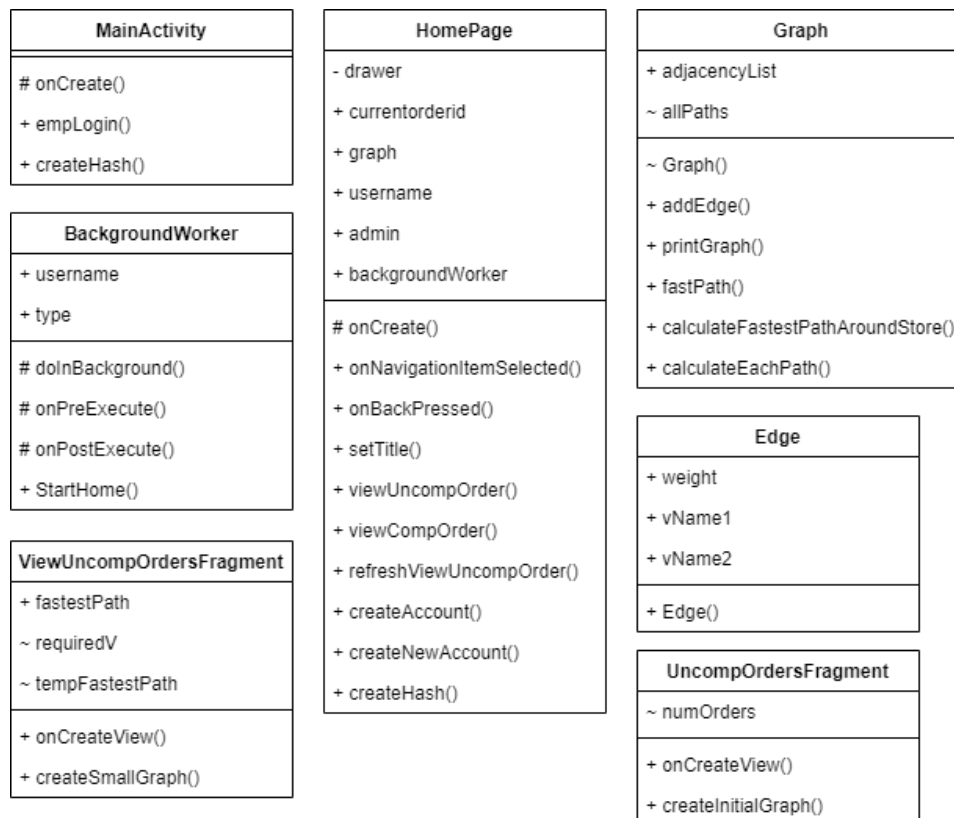
## Class Definition Diagrams

I have briefly outlined some of the key classes that will be included in the final solution. These are not all the classes that will be included in the final program, though these are the classes containing the most important methods and variables

### Customer Application



### Employee Application



# Technical Solution

## Customer Application

### MainActivity

```
package com.example.asdaclickcollectcustomersystem;

import android.net.ConnectivityManager;
import android.net.NetworkInfo;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.view.View;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;

import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;

public class MainActivity extends AppCompatActivity {

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

        getSupportFragmentManager().beginTransaction().replace(R.id.mainfragment_container,
            new LoginFragment()).commit(); // sets default layout to the login
        fragment
    }

    public void custLoginLayout (View view) {

        getSupportFragmentManager().beginTransaction().replace(R.id.mainfragment_container,
            new LoginFragment()).commit(); // procedure to set layout to the
        login fragment
    }

    public void custRegister(View view) {

        getSupportFragmentManager().beginTransaction().replace(R.id.mainfragment_container,
            new RegisterFragment()).commit(); // procedure to set layout to the
        register fragment
    }

    public void createAccount(View view) {

        String fullname = ((EditText)
            (findViewById(R.id.et_fullname))).getText().toString(); // stores full name from
        register fragment
        String email = ((EditText)
            (findViewById(R.id.et_email))).getText().toString(); // stores email from register
        fragment
        String username = ((EditText)
            (findViewById(R.id.et_username))).getText().toString(); // stores username from
        register fragment
        String dateofbirth = String.valueOf(((EditText)
            findViewById(R.id.et_dateofbirth))).getText(); // stores date of birth from
        register fragment

        try { // sets layout of date of birth so it is compatible for the database
            dateofbirth = dateofbirth.split("/") [2] + "-" +
            dateofbirth.split("/") [0] + "-" + dateofbirth.split("/") [1];
        } catch (Exception e) {
            dateofbirth = "";
        }
    }
}
```



```

        String password = createHash((EditText) findViewById(R.id.et_password)); //
creates hash for password
        String passwordConfirmation = createHash((EditText)
findViewById(R.id.et_passwordconfirmation)); // creates hash for password
confirmation

        TextView pw = findViewById(R.id.et_password);
        pw.setText(""); // sets password text back to null
        TextView pwc = findViewById(R.id.et_passwordconfirmation);
        pwc.setText(""); // sets password confirmation text back to null

        BackgroundWorker backgroundWorker = new BackgroundWorker(this);
        ConnectivityManager connectivityManager = (ConnectivityManager)
getSystemService(this.CONNECTIVITY_SERVICE);
        if
(connectivityManager.getNetworkInfo(ConnectivityManager.TYPE_MOBILE).getState() ==
NetworkInfo.State.CONNECTED ||

connectivityManager.getNetworkInfo(ConnectivityManager.TYPE_WIFI).getState() ==
NetworkInfo.State.CONNECTED ||

connectivityManager.getNetworkInfo(ConnectivityManager.TYPE_ETHERNET).getState() ==
NetworkInfo.State.CONNECTED) {
            backgroundWorker.execute("custRegister", fullname, email, dateofbirth,
username, password, passwordConfirmation); // checks integrity of network
connection before executing
        } else {
            Toast.makeText(this, "Connection failed. Please check connection",
Toast.LENGTH_SHORT);
        }

        fullname = null; // sets data to null
        email = null; // sets data to null
        dateofbirth = null; // sets data to null
        System.gc(); // attempts to clear unused variables
    }

    public void custLogin(View view) {
        String username = ((EditText)
(findViewById(R.id.et_username))).getText().toString(); // stores username from
login fragment
        String password = createHash((EditText) findViewById(R.id.et_password)); //
creates hash for password

        TextView tv = findViewById(R.id.et_password);
        tv.setText(""); // sets password text back to null

        BackgroundWorker backgroundWorker = new BackgroundWorker(this);

        ConnectivityManager connectivityManager = (ConnectivityManager)
getSystemService(this.CONNECTIVITY_SERVICE);
        if
(connectivityManager.getNetworkInfo(ConnectivityManager.TYPE_MOBILE).getState() ==
NetworkInfo.State.CONNECTED ||

connectivityManager.getNetworkInfo(ConnectivityManager.TYPE_WIFI).getState() ==
NetworkInfo.State.CONNECTED ||

connectivityManager.getNetworkInfo(ConnectivityManager.TYPE_ETHERNET).getState() ==
NetworkInfo.State.CONNECTED) {
            backgroundWorker.execute("custLogin", username, password); // checks
integrity of network connection before executing
        } else {
            Toast.makeText(this, "Connection failed. Please check connection",
Toast.LENGTH_SHORT);
        }
    }
}

```

```
    public String createHash(EditText password) { // converts input text to a
secure MD5 hash code
        try {
            MessageDigest digest = MessageDigest.getInstance("MD5");
            digest.update(password.getText().toString().getBytes());
            byte messageDigest[] = digest.digest();

            StringBuffer hexString = new StringBuffer();
            for (int i = 0; i < messageDigest.length; i++) {
                hexString.append(Integer.toHexString(0xFF & messageDigest[i]));
            }
            return hexString.toString();
        } catch (NoSuchAlgorithmException e) {
            e.printStackTrace();
        }
        return "";
    }
}
```

## HomePage

```
package com.example.asdaclickcollectcustomersystem;

import android.os.Bundle;
import android.support.annotation.NonNull;
import android.support.design.widget.NavigationView;
import android.support.v4.view.GravityCompat;
import android.support.v4.widget.DrawerLayout;
import android.support.v7.app.ActionBarDrawerToggle;
import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.Toolbar;
import android.view.MenuItem;
import android.view.View;
import android.widget.ImageButton;
import android.widget.TextView;

public class HomePage extends AppCompatActivity implements
NavigationView.OnNavigationItemSelectedListener {

    private DrawerLayout drawer;
    public String userid;
    public String orderid = "";

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        String username = getIntent().getExtras().getString("username"); // gets
bundle extra from mainactivity
        userid = getIntent().getExtras().getString("userid"); // gets bundle extra
from mainactivity

        setContentView(R.layout.activity_home_page); // assigns relevant layout

        final NavigationView navigationView = findViewById(R.id.nav_view);
        View headerView = navigationView.getHeaderView(0);
        TextView navUsername = headerView.findViewById(R.id.username_text);
        navUsername.setText(username); // sets username in layout to stored
username

        Toolbar toolbar = findViewById(R.id.toolbar);
        setSupportActionBar(toolbar); // sets toolbar for navigation

        drawer = findViewById(R.id.drawer_layout); // sets drawer layout for
navigation
        navigationView.setNavigationItemSelectedListener(this);

        ImageButton img = (ImageButton) findViewById(R.id.image_button); // toolbar
button for viewbasket
        img.setOnClickListener(new View.OnClickListener() { // defines
onclicklistener for viewbasket button
            public void onClick(View v) {
                navigationView.setCheckedItem(R.id.nav_basket);
                getSupportActionBar().setTitle("Your Basket");

                getSupportFragmentManager().beginTransaction().replace(R.id.fragment_container,
                    new BasketFragment()).commit(); // replaces current view
with new layout
            }
        });

        ActionBarDrawerToggle toggle = new ActionBarDrawerToggle(this, drawer,
toolbar,
            R.string.nav_drawer_open, R.string.nav_drawer_close);
        drawer.addDrawerListener(toggle);
        toggle.syncState();

        if (savedInstanceState == null) {
```

```

        getSupportActionBar().setTitle("Products");

getSupportFragmentManager().beginTransaction().replace(R.id.fragment_container,
        new ProductsFragment()).commit(); // replaces current view with
new layout
        navigationView.setCheckedItem(R.id.nav_products);
    }
}

@Override
public boolean onNavigationItemSelected(@NonNull MenuItem menuItem) {
    switch (menuItem.getItemId()) {
        case R.id.nav_products:
            getSupportActionBar().setTitle("View Products");

getSupportFragmentManager().beginTransaction().replace(R.id.fragment_container,
        new ProductsFragment()).commit(); // replaces current view
with new layout
            break;
        case R.id.nav_basket:
            getSupportActionBar().setTitle("Your Basket");

getSupportFragmentManager().beginTransaction().replace(R.id.fragment_container,
        new BasketFragment()).commit(); // replaces current view
with new layout
            break;
        case R.id.nav_logout:
            this.finish(); // ends current activity if user chooses to log out
            break;
    }

    drawer.closeDrawer(GravityCompat.START);
    return true;
}

@Override
public void onBackPressed() { // used stack to store previous fragments so that
navigation can be used properly
    if (drawer.isDrawerOpen(GravityCompat.START)) {
        drawer.closeDrawer(GravityCompat.START);
    } else {
        super.onBackPressed();
    }
}

public String getUserId() {
    return userid;
} // returns userid

public String getOrderid() {
    return orderid;
} // returns orderid

public void setOrderid(String Orderid) {
    orderid = Orderid;
} // sets orderid

public void refreshBasket() { // refreshes basket to show updated layout after
item is deleted or order is submitted

getSupportFragmentManager().beginTransaction().replace(R.id.fragment_container, new
BasketFragment()).commit();
    }
}

```

## BackgroundWorker

```
package com.example.asdaclickcollectcustomersystem;

import android.app.AlertDialog;
import android.content.Context;
import android.content.Intent;
import android.os.AsyncTask;
import android.os.Bundle;
import android.util.Log;

import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.OutputStream;
import java.io.OutputStreamWriter;
import java.net.HttpURLConnection;
import java.net.URL;
import java.net.URLEncoder;

public class BackgroundWorker extends AsyncTask<String, Void, String> {
    Context context;
    AlertDialog alertDialog;
    public String username = "";
    public String userid = "";
    String type = "";
    BackgroundWorker(Context ctx) {
        context = ctx;
    }

    @Override
    protected String doInBackground(String... voids) {
        type = voids[0];
        String password;
        String line;
        String custLogin = "http://10.0.2.2/custLogin.php";
        String custRegister = "http://10.0.2.2/custRegister.php";

        if (type.equals("custLogin")) { // queries database for user details, uses
custlogin.php
            try {
                username = voids[1];
                password = voids[2];
                if (!(username != "" || username.isEmpty()) && !(password != "" ||
password.isEmpty())) { // logs customer in, uses custlogin.php
                    URL url = new URL(custLogin);
                    HttpURLConnection httpURLConnection = (HttpURLConnection)
url.openConnection();
                    httpURLConnection.setRequestMethod("POST");
                    httpURLConnection.setDoOutput(true);
                    httpURLConnection.setDoInput(true);
                    OutputStream outputStream =
httpURLConnection.getOutputStream();
                    BufferedWriter bufferedWriter = new BufferedWriter(new
OutputStreamWriter(outputStream, "UTF-8"));
                    String post_data = URLEncoder.encode("username", "UTF-8") + "="
+ URLEncoder.encode(username, "UTF-8") + "&"
                        + URLEncoder.encode("password", "UTF-8") + "=" +
URLEncoder.encode(password, "UTF-8");
                    bufferedWriter.write(post_data);
                    bufferedWriter.flush();
                    bufferedWriter.close();
                    InputStream inputStream = httpURLConnection.getInputStream();
                    BufferedReader bufferedReader = new BufferedReader(new
InputStreamReader(inputStream, "iso-8859-1"));
                    String result = "";
                    while ((line = bufferedReader.readLine()) != null) {
```

```

        result += line; // reads in echo message
    }
    bufferedReader.close();
    inputStream.close();
    httpURLConnection.disconnect();
    String[] parts = result.split("\\|"); // stores userid and
returns result

    userid = parts[1];
    result = parts[0];
    return result;
}
} catch (Exception e) {
    e.printStackTrace();
}
} else if (type.equals("custRegister")) { // creates new customer account,
uses custregister.php
    try {
        String fullname = voids[1];
        String email = voids[2];
        String dateofbirth = voids[3];
        username = voids[4];
        password = voids[5];
        String passwordconfirmation = voids[6];

        if (password.equals(passwordconfirmation)) {
            URL url = new URL(custRegister);
            HttpURLConnection httpURLConnection = (HttpURLConnection)
url.openConnection();
            httpURLConnection.setRequestMethod("POST");
            httpURLConnection.setDoOutput(true);
            httpURLConnection.setDoInput(true);
            OutputStream outputStream =
httpURLConnection.getOutputStream();
            BufferedWriter bufferedWriter = new BufferedWriter(new
OutputStreamWriter(outputStream, "UTF-8"));
            String post_data = URLEncoder.encode("username", "UTF-8") + "="
+ URLEncoder.encode(username, "UTF-8") + "&"
                + URLEncoder.encode("password", "UTF-8") + "=" +
URLEncoder.encode(password, "UTF-8") + "&"
                + URLEncoder.encode("email", "UTF-8") + "=" +
URLEncoder.encode(email, "UTF-8") + "&"
                + URLEncoder.encode("fullname", "UTF-8") + "=" +
URLEncoder.encode(fullname, "UTF-8") + "&"
                + URLEncoder.encode("dateofbirth", "UTF-8") + "=" +
URLEncoder.encode(dateofbirth, "UTF-8");
            bufferedWriter.write(post_data);
            bufferedWriter.flush();
            bufferedWriter.close();
            InputStream inputStream = httpURLConnection.getInputStream();
            BufferedReader bufferedReader = new BufferedReader(new
InputStreamReader(inputStream, "iso-8859-1"));
            String result = "";
            while ((line = bufferedReader.readLine()) != null) {
                try {
                    result += line; // reads in echo message
                } catch (Exception e) {
                    Log.e("Error 2", e.toString());
                }
            }
            bufferedReader.close();
            inputStream.close();
            httpURLConnection.disconnect();
            String[] parts = result.split("\\|"); // stores userid and
returns result

            result = parts[0];
            return result;
        } else {
            alertDialog = new AlertDialog.Builder(context).create();

```

```

        alertDialog.setTitle("The passwords do not match.\nPlease try
again.");
        alertDialog.show();
    }
    } catch (Exception e) { // catches any errors
        Log.e("Error", e.toString()); // logs error
        e.printStackTrace();
    }
}

return "";
}

@Override
protected void onPreExecute() {
    alertDialog = new AlertDialog.Builder(context).create();
    alertDialog.setTitle("Connection Status");
}

@Override
protected void onPostExecute(String result) { // determines whether the query
was successful
    if (result.contains("Login not successful")) {
        alertDialog.setMessage("Incorrect credentials");
        alertDialog.show();
    } else if (result.contains("Register not successful")) {
        alertDialog.setMessage("Register not successful" );
        alertDialog.show();
    } else if ((result == "") || result.isEmpty()) {
        alertDialog.setMessage("Something went wrong. Please try again" );
        alertDialog.show();
    } else {
        StartHome();
    }
}

public void StartHome() { // starts new homepage activity
    Intent intent = new Intent(context, HomePage.class);
    Bundle extras = new Bundle(); // adds bundle to store username and userid
    extras.putString("username", username);
    extras.putString("userid", userid);
    intent.putExtras(extras);
    context.startActivity(intent);
}

@Override
protected void onProgressUpdate(Void... values) {
    super.onProgressUpdate(values);
}
}

```

## LoginFragment

```
package com.example.asdaclickcollectcustomersystem;

import android.os.Bundle;
import android.support.annotation.NonNull;
import android.support.annotation.Nullable;
import android.support.v4.app.Fragment;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;

public class LoginFragment extends Fragment {

    @Nullable
    @Override
    public View onCreateView(@NonNull LayoutInflater inflater, @Nullable ViewGroup
container, @Nullable Bundle savedInstanceState) {
        return inflater.inflate(R.layout.fragment_login, container, false);
    }
}
```



## RegisterFragment

```
package com.example.asdaclickcollectcustomersystem;

import android.app.DatePickerDialog;
import android.graphics.Color;
import android.graphics.drawable.ColorDrawable;
import android.os.Bundle;
import android.support.annotation.NonNull;
import android.support.annotation.Nullable;
import android.support.v4.app.Fragment;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.DatePicker;
import android.widget.EditText;
import android.widget.TextView;

import java.util.Calendar;

public class RegisterFragment extends Fragment {

    private static final String TAG = "RegisterFragment";
    private TextView mDisplayDate;
    private DatePickerDialog.OnDateSetListener mDateSetListener;

    @Nullable
    @Override
    public View onCreateView(@NonNull LayoutInflater inflater, @Nullable ViewGroup container, @Nullable Bundle savedInstanceState) {
        return inflater.inflate(R.layout.fragment_register, container, false);
    }

    @Override
    public void onViewCreated(View view, @Nullable Bundle savedInstanceState) { //
        used dialog box to create a date picker for the register fragment

        mDisplayDate = (EditText) getView().findViewById(R.id.et_dateofbirth);

        mDisplayDate.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Calendar cal = Calendar.getInstance();
                int year = cal.get(Calendar.YEAR);
                int month = cal.get(Calendar.MONTH);
                int day = cal.get(Calendar.DAY_OF_MONTH);

                DatePickerDialog dialog = new DatePickerDialog(
                    getContext(),
                    android.R.style.Theme_Holo_Light_Dialog_MinWidth,
                    mDateSetListener,
                    year, month, day);
                dialog.getWindow().setBackgroundDrawable(new
                    ColorDrawable(Color.TRANSPARENT));
                dialog.show();
            }
        });

        mDateSetListener = new DatePickerDialog.OnDateSetListener() {
            @Override
            public void onDateSet(DatePicker datePicker, int year, int month, int
            day) {
                month = month + 1;
                Log.d(TAG, "onDateSet: mm/dd/yyyy: " + month + "/" + day + "/" +
                year);

                String date = month + "/" + day + "/" + year;
            }
        };
    }
}
```

```
        mDisplayDate.setText (date) ;  
    }  
};  
}
```

## ProductsFragment

```
package com.example.asdaclickcollectcustomersystem;

import android.app.AlertDialog;
import android.support.v4.app.Fragment;
import android.os.Bundle;
import android.os.StrictMode;
import android.support.annotation.NonNull;
import android.support.annotation.Nullable;
import android.support.v7.widget.CardView;
import android.view.Gravity;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.LinearLayout;
import android.widget.TextView;
import android.widget.Toast;

import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.OutputStream;
import java.io.OutputStreamWriter;
import java.io.UnsupportedEncodingException;
import java.net.HttpURLConnection;
import java.net.MalformedURLException;
import java.net.ProtocolException;
import java.net.URL;
import java.net.URLEncoder;

import static android.widget.LinearLayout.HORIZONTAL;
import static android.widget.LinearLayout.VERTICAL;

public class ProductsFragment extends Fragment {

    @Nullable
    @Override
    public View onCreateView(@NonNull LayoutInflater inflater, @Nullable ViewGroup
container, @Nullable Bundle savedInstanceState) {
        View view = inflater.inflate(R.layout.fragment_products, container, false);

        LinearLayout layout = view.findViewById(R.id.fragment_container); //
establishes layouts and layoutparams
        LinearLayout.LayoutParams cvParams = new LinearLayout.LayoutParams(
            LinearLayout.LayoutParams.MATCH_PARENT,
            LinearLayout.LayoutParams.WRAP_CONTENT);
        cvParams.setMargins(10, 10, 10, 10);
        LinearLayout.LayoutParams tvParams = new LinearLayout.LayoutParams(
            LinearLayout.LayoutParams.MATCH_PARENT,
            LinearLayout.LayoutParams.WRAP_CONTENT);
        tvParams.setMargins(5, 5, 5, 5);
        LinearLayout.LayoutParams btnParams = new LinearLayout.LayoutParams(
            LinearLayout.LayoutParams.WRAP_CONTENT,
            LinearLayout.LayoutParams.WRAP_CONTENT);
        btnParams.setMargins(25, 5, 5, 5);
        LinearLayout.LayoutParams layout1Params = new LinearLayout.LayoutParams(
            LinearLayout.LayoutParams.MATCH_PARENT,
            LinearLayout.LayoutParams.MATCH_PARENT);
        LinearLayout.LayoutParams layout2Params = new LinearLayout.LayoutParams(
            LinearLayout.LayoutParams.WRAP_CONTENT,
            LinearLayout.LayoutParams.WRAP_CONTENT);
        layout2Params.setMargins(10, 10, 100, 10);
        LinearLayout.LayoutParams layout3Params = new LinearLayout.LayoutParams(
            LinearLayout.LayoutParams.MATCH_PARENT,
```

```

        LinearLayout.LayoutParams.WRAP_CONTENT);

String line = "";
try {
    StrictMode.ThreadPolicy policy = new
StrictMode.ThreadPolicy.Builder().permitAll().build();
    StrictMode.setThreadPolicy(policy);
    String retrieveProductsURL = "http://10.0.2.2/retrieveProducts.php";
    final String addToBasketURL = "http://10.0.2.2/addToBasket.php";
    final String checkOrderURL = "http://10.0.2.2/checkOrder.php";
    final String createOrderURL = "http://10.0.2.2/createOrder.php";
    URL url = new URL(retrieveProductsURL); // queries database for
relevant products, uses retrieveproducts.php
    HttpURLConnection httpURLConnection = (HttpURLConnection)
url.openConnection();
    httpURLConnection.setDoOutput(true);
    httpURLConnection.setDoInput(true);
    InputStream inputStream = httpURLConnection.getInputStream();
    BufferedReader bufferedReader = new BufferedReader(new
InputStreamReader(inputStream, "iso-8859-1"));
    while ((line = bufferedReader.readLine()) != null) { // uses layouts
and cardview to present each item
        String[] parts = line.split("\\|");
        CardView cv = new CardView(getContext());
        LinearLayout layout1 = new LinearLayout(getContext());
        LinearLayout layout2 = new LinearLayout(getContext());
        LinearLayout layout3 = new LinearLayout(getContext());
        layout1.setOrientation(HORIZONTAL);
        layout2.setOrientation(VERTICAL);
        layout3.setOrientation(HORIZONTAL);

        TextView productName = new TextView(getContext());
        productName.setText(parts[1]);
        final String productid = parts[0];
        final String quantity = "1"; // sets default quantity to 1

        Button addToBasket = new Button(getContext());
        addToBasket.setText("Add to Basket");
        addToBasket.setOnClickListener(new Button.OnClickListener() {
            public void onClick(View v) { // sends product to basket
                HomePage homePage = (HomePage) getActivity();
                String userid = homePage.getUserid();
                String orderid = homePage.getOrderid();
                String result = "";
                if (orderid == "") {
                    try {
                        URL url = new URL(checkOrderURL); // checks for
existing order, uses checkorder.php
                        HttpURLConnection httpURLConnection =
(HttpURLConnection) url.openConnection();
                        httpURLConnection.setRequestMethod("POST");
                        httpURLConnection.setDoOutput(true);
                        httpURLConnection.setDoInput(true);
                        OutputStream outputStream =
httpURLConnection.getOutputStream();
                        BufferedWriter bufferedWriter = new
BufferedWriter(new OutputStreamWriter(outputStream, "UTF-8"));
                        String post_data = URLEncoder.encode("userid",
"UTF-8") + "=" + URLEncoder.encode(userid, "UTF-8");
                        bufferedWriter.write(post_data);
                        bufferedWriter.flush();
                        bufferedWriter.close();
                        InputStream inputStream =
httpURLConnection.getInputStream();
                        BufferedReader bufferedReader = new
BufferedReader(new InputStreamReader(inputStream, "iso-8859-1"));
                        result = "";
                        String line;

```

```

        while ((line = bufferedReader.readLine()) != null)
        {
            result += line;
        }
        bufferedReader.close();
        inputStream.close();
        httpURLConnection.disconnect();

        if (result.contains("No orders found")) {
            try {
                url = new URL(createOrderURL); // creates
new order in database, uses createorder.php
                httpURLConnection = (HttpURLConnection)
url.openConnection();

                httpURLConnection.setRequestMethod("POST");
                httpURLConnection.setDoOutput(true);
                httpURLConnection.setDoInput(true);
                outputStream =
httpURLConnection.getOutputStream();
                bufferedWriter = new BufferedWriter(new
OutputStreamWriter(outputStream, "UTF-8"));
                post_data = URLEncoder.encode("userid",
"UTF-8") + "=" + URLEncoder.encode(userid, "UTF-8");
                bufferedWriter.write(post_data);
                bufferedWriter.flush();
                bufferedWriter.close();
                inputStream =
httpURLConnection.getInputStream();
                bufferedReader = new BufferedReader(new
InputStreamReader(inputStream, "iso-8859-1"));
                result = "";
                while ((line = bufferedReader.readLine())
!= null) {

                    result += line;
                }
                bufferedReader.close();
                inputStream.close();
                httpURLConnection.disconnect();
            } catch (MalformedURLException e) {
                e.printStackTrace();
            } catch (UnsupportedEncodingException e) {
                e.printStackTrace();
            } catch (ProtocolException e) {
                e.printStackTrace();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    } catch (MalformedURLException e) {
        e.printStackTrace();
    } catch (UnsupportedEncodingException e) {
        e.printStackTrace();
    } catch (ProtocolException e) {
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    }
    }

    homepage.setOrderid(result); // returns result to
homepage

    orderid = homepage.getOrderid();
}
try {
    URL url = new URL(addToBasketURL); // adds item to
basket, uses addtobasket.php
    HttpURLConnection httpURLConnection =
(HttpURLConnection) url.openConnection();
    httpURLConnection.setRequestMethod("POST");
    httpURLConnection.setDoOutput(true);

```

```

        httpURLConnection.setDoInput(true);
        OutputStream outputStream =
httpURLConnection.getOutputStream();
        BufferedWriter bufferedWriter = new BufferedWriter(new
OutputStreamWriter(outputStream, "UTF-8"));
        String post_data = URLEncoder.encode("orderid", "UTF-
8") + "=" + URLEncoder.encode(orderid, "UTF-8") + "&"
+ URLEncoder.encode("productid", "UTF-8") + "="
+ URLEncoder.encode(productid, "UTF-8");
        //+ URLEncoder.encode("quantity", "UTF-8") + "=" +
URLEncoder.encode(quantity, "UTF-8");
        bufferedWriter.write(post_data);
        bufferedWriter.flush();
        bufferedWriter.close();
        InputStream inputStream =
httpURLConnection.getInputStream();
        BufferedReader bufferedReader = new BufferedReader(new
InputStreamReader(inputStream, "iso-8859-1"));
        result = "";
        String line;
        while ((line = bufferedReader.readLine()) != null) {
            result += line;
        }
        if (result != "") {
            Toast toast = Toast.makeText(getApplicationContext(), result,
Toast.LENGTH_LONG);
            toast.show();
        }
        bufferedReader.close();
        inputStream.close();
        httpURLConnection.disconnect();
    } catch (MalformedURLException e) {
        e.printStackTrace();
    } catch (UnsupportedEncodingException e) {
        e.printStackTrace();
    } catch (ProtocolException e) {
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    }
}

layout1.setLayoutParams(layout1Params);
layout2.setLayoutParams(layout2Params);
layout3.setLayoutParams(layout3Params);
layout3.setGravity(Gravity.RIGHT);
addToBasket.setLayoutParams(btnParams);
productName.setLayoutParams(tvParams);

layout3.addView(addToBasket);
layout2.addView(productName);
layout1.addView(layout2);
layout1.addView(layout3);
cv.addView(layout1);
cv.setElevation(15);
cv.setRadius(15);
cv.setLayoutParams(cvParams);
layout.addView(cv);
}
bufferedReader.close();
inputStream.close();
httpURLConnection.disconnect();
} catch (Exception e) {
    AlertDialog alertDialog = new
AlertDialog.Builder(getApplicationContext()).create();
    alertDialog.setMessage(e.getMessage());
    alertDialog.show();
}

```

```
        e.printStackTrace();
    }
    return view;
}
```

## BasketFragment

```
package com.example.asdaclickcollectcustomersystem;

import android.app.AlertDialog;
import android.support.v4.app.Fragment;
import android.content.Context;
import android.os.Bundle;
import android.os.StrictMode;
import android.support.annotation.NonNull;
import android.support.annotation.Nullable;
import android.support.v7.widget.CardView;
import android.view.Gravity;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.LinearLayout;
import android.widget.TextView;
import android.widget.Toast;

import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.OutputStream;
import java.io.OutputStreamWriter;
import java.io.UnsupportedEncodingException;
import java.net.HttpURLConnection;
import java.net.MalformedURLException;
import java.net.ProtocolException;
import java.net.URL;
import java.net.URLEncoder;

import static android.widget.LinearLayout.HORIZONTAL;
import static android.widget.LinearLayout.VERTICAL;

public class BasketFragment extends Fragment {

    @Override
    public Context getContext() {
        return super.getContext();
    }

    @Nullable
    @Override
    public View onCreateView(@NonNull LayoutInflater inflater, @Nullable ViewGroup
container, @Nullable Bundle savedInstanceState) {
        View view = inflater.inflate(R.layout.fragment_basket, container, false);

        LinearLayout layout = view.findViewById(R.id.fragment_container); //
establishes layouts and layoutparams for the basket fragment
        LinearLayout.LayoutParams cvParams = new LinearLayout.LayoutParams(
            LinearLayout.LayoutParams.MATCH_PARENT,
            LinearLayout.LayoutParams.WRAP_CONTENT);
        cvParams.setMargins(10, 10, 10, 10);
        LinearLayout.LayoutParams tvParams = new LinearLayout.LayoutParams(
            LinearLayout.LayoutParams.MATCH_PARENT,
            LinearLayout.LayoutParams.WRAP_CONTENT);
        tvParams.setMargins(5, 5, 5, 5);
        LinearLayout.LayoutParams btnParams = new LinearLayout.LayoutParams(
            LinearLayout.LayoutParams.WRAP_CONTENT,
            LinearLayout.LayoutParams.WRAP_CONTENT);
        btnParams.setMargins(25, 5, 5, 5);
        LinearLayout.LayoutParams layoutlParams = new LinearLayout.LayoutParams(
            LinearLayout.LayoutParams.MATCH_PARENT,
            LinearLayout.LayoutParams.MATCH_PARENT);
```



```

        LinearLayout.LayoutParams layout2Params = new LinearLayout.LayoutParams (
            LinearLayout.LayoutParams.WRAP_CONTENT,
            LinearLayout.LayoutParams.WRAP_CONTENT);
        layout2Params.setMargins(10, 10, 100, 10);
        LinearLayout.LayoutParams layout3Params = new LinearLayout.LayoutParams (
            LinearLayout.LayoutParams.MATCH_PARENT,
            LinearLayout.LayoutParams.WRAP_CONTENT);

        String line = "";
        final HomePage homePage = (HomePage) getActivity();
        final String userid = homePage.getUserid();
        final String orderid = homePage.getOrderid();
        try {
            StrictMode.ThreadPolicy policy = new
StrictMode.ThreadPolicy.Builder().permitAll().build();
            StrictMode.setThreadPolicy(policy);
            String retrieveBasketURL = "http://10.0.2.2/retrieveBasket.php";
            final String addOrderURL = "http://10.0.2.2/addOrder.php";
            final String deleteFromBasketURL =
"http://10.0.2.2/deleteFromBasket.php";

            URL url = new URL(retrieveBasketURL); // queries database for relevant
order, uses retrievebasket.php
            HttpURLConnection httpURLConnection = (HttpURLConnection)
url.openConnection();
            httpURLConnection.setRequestMethod("POST");
            httpURLConnection.setDoOutput(true);
            httpURLConnection.setDoInput(true);
            OutputStream outputStream = httpURLConnection.getOutputStream();
            BufferedWriter bufferedWriter = new BufferedWriter(new
OutputStreamWriter(outputStream, "UTF-8"));
            String post_data = URLEncoder.encode("orderid", "UTF-8") + "=" +
URLEncoder.encode(orderid, "UTF-8");
            bufferedWriter.write(post_data);
            bufferedWriter.flush();
            bufferedWriter.close();
            InputStream inputStream = httpURLConnection.getInputStream();
            BufferedReader bufferedReader = new BufferedReader(new
InputStreamReader(inputStream, "iso-8859-1"));
            String result = "";
            while ((line = bufferedReader.readLine()) != null) { // uses layouts
and cardview to present each item
                if (line.length() < 2) {
                    break;
                }
                result += line;
                String[] parts = line.split("\\|");
                CardView cv = new CardView(getContext());
                LinearLayout layout1 = new LinearLayout(getContext());
                LinearLayout layout2 = new LinearLayout(getContext());
                LinearLayout layout3 = new LinearLayout(getContext());
                layout1.setOrientation(HORIZONTAL);
                layout2.setOrientation(VERTICAL);
                layout3.setOrientation(HORIZONTAL);
                TextView productName = new TextView(getContext());
                productName.setText(parts[1]);
                final String productid = parts[0];
                final String quantity = parts[2];

                Button deleteFromBasket = new Button(getContext());
                deleteFromBasket.setText("Delete from Basket");
                deleteFromBasket.setOnClickListener(new Button.OnClickListener() {
                    public void onClick(View v) { // removes product from basket,
uses deletefrombasket.php
                        String result;
                        try {
                            URL url = new URL(deleteFromBasketURL); // deletes
selected item

```

```

        HttpURLConnection httpURLConnection =
(HttpURLConnection) url.openConnection();
        httpURLConnection.setRequestMethod("POST");
        httpURLConnection.setDoOutput(true);
        httpURLConnection.setDoInput(true);
        OutputStream outputStream =
httpURLConnection.getOutputStream();
        BufferedWriter bufferedWriter = new BufferedWriter(new
OutputStreamWriter(outputStream, "UTF-8"));
        String post_data = URLEncoder.encode("orderid", "UTF-
8") + "=" + URLEncoder.encode(orderid, "UTF-8") + "&"
+ URLEncoder.encode("productid", "UTF-8") + "="
+ URLEncoder.encode(productid, "UTF-8");
        bufferedWriter.write(post_data);
        bufferedWriter.flush();
        bufferedWriter.close();
        InputStream inputStream =
httpURLConnection.getInputStream();
        BufferedReader bufferedReader = new BufferedReader(new
InputStreamReader(inputStream, "iso-8859-1"));
        result = "";
        String line;
        while ((line = bufferedReader.readLine()) != null) {
            result += line;
        }
        bufferedReader.close();
        inputStream.close();
        httpURLConnection.disconnect();

        homePage.refreshBasket(); // refreshes the basket
fragment

        if (result.contains("Error")) {
            Toast toast = Toast.makeText(getContext(), result,
Toast.LENGTH_SHORT);
            toast.show();
        }
    } catch (MalformedURLException e) { // catches errors when
querying the database
        e.printStackTrace();
    } catch (UnsupportedEncodingException e) {
        e.printStackTrace();
    } catch (ProtocolException e) {
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    }
}

});
layout1.setLayoutParams(layout1Params);
layout2.setLayoutParams(layout2Params);
layout3.setLayoutParams(layout3Params);
layout3.setGravity(Gravity.RIGHT);
deleteFromBasket.setLayoutParams(btnParams);
productName.setLayoutParams(tvParams);

layout3.addView(deleteFromBasket);
layout2.addView(productName);
layout1.addView(layout2);
layout1.addView(layout3);
cv.addView(layout1);
cv.setElevation(15);
cv.setRadius(15);
cv.setLayoutParams(cvParams);
layout.addView(cv);
}
if (result == "") { // output when basket is empty
    TextView tv = new TextView(getContext());

```

```

        tv.setTextSize(20);
        tv.setText("You haven't added anything to your basket yet. Add
something and come back later.");
        tv.setGravity(Gravity.CENTER);
        tv.setLayoutParams(layout1Params);
        layout.addView(tv);
    } else { // output when basket contains items
        Button sendBtn = new Button(getContext()); // creates a new button
to submit the order
        sendBtn.setText("Send Order");
        sendBtn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) { // submits order for picking,
uses addorder.php
                String result;
                try {
                    URL url = new URL(addOrderURL); // checks for existing
order, uses checkorder.php
                    HttpURLConnection httpURLConnection =
(HttpURLConnection) url.openConnection();
                    httpURLConnection.setRequestMethod("POST");
                    httpURLConnection.setDoOutput(true);
                    httpURLConnection.setDoInput(true);
                    OutputStream outputStream =
httpURLConnection.getOutputStream();
                    BufferedWriter bufferedWriter = new BufferedWriter(new
OutputStreamWriter(outputStream, "UTF-8"));
                    String post_data = URLEncoder.encode("orderid", "UTF-
8") + "=" + URLEncoder.encode(orderid, "UTF-8");
                    bufferedWriter.write(post_data);
                    bufferedWriter.flush();
                    bufferedWriter.close();
                    InputStream inputStream =
httpURLConnection.getInputStream();
                    BufferedReader bufferedReader = new BufferedReader(new
InputStreamReader(inputStream, "iso-8859-1"));
                    result = "";
                    String line;
                    while ((line = bufferedReader.readLine()) != null) {
                        result += line;
                    }
                    bufferedReader.close();
                    inputStream.close();
                    httpURLConnection.disconnect();

                    if (result.contains("Error")) {
                        Toast toast = Toast.makeText(getContext(), result,
Toast.LENGTH_SHORT);
                        toast.show();
                    }

                    homePage.setOrderid("");
                    homePage.refreshBasket(); // refreshes the
basketfragment

                } catch (MalformedURLException e) {
                    e.printStackTrace();
                } catch (UnsupportedEncodingException e) {
                    e.printStackTrace();
                } catch (ProtocolException e) {
                    e.printStackTrace();
                } catch (IOException e) {
                    e.printStackTrace();
                }
            }
        });
        sendBtn.setLayoutParams(btnParams);
        sendBtn.setGravity(Gravity.RIGHT);

```

```
        layout.addView(sendBtn);
        bufferedReader.close();
        inputStream.close();
        httpURLConnection.disconnect();
    }
} catch (Exception e) {
    AlertDialog alertDialog = new
AlertDialog.Builder(getContext()).create();
    alertDialog.setMessage(e.getMessage());
    alertDialog.show();
    e.printStackTrace();
}
return view;
}
```

# Employee Application

## MainActivity

```
package com.example.asdaclickcollectemployeesystem;

import android.net.ConnectivityManager;
import android.net.NetworkInfo;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;

public class MainActivity extends AppCompatActivity {

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

    public void empLogin(View view) throws NoSuchAlgorithmException {
        String username = ((EditText)
(findViewById(R.id.et_username))).getText().toString(); // gets username
        String password = createHash(); // creates hash of password

        TextView tv = findViewById(R.id.et_password);
        tv.setText(""); // sets password text back to null

        BackgroundWorker backgroundWorker = new BackgroundWorker(this);

        ConnectivityManager connectivityManager = (ConnectivityManager)
getSystemService(this.CONNECTIVITY_SERVICE);
        if
(connectivityManager.getNetworkInfo(ConnectivityManager.TYPE_MOBILE).getState() ==
NetworkInfo.State.CONNECTED ||

connectivityManager.getNetworkInfo(ConnectivityManager.TYPE_WIFI).getState() ==
NetworkInfo.State.CONNECTED ||

connectivityManager.getNetworkInfo(ConnectivityManager.TYPE_ETHERNET).getState() ==
NetworkInfo.State.CONNECTED) {
            backgroundWorker.execute("empLogin", username, password); // checks
integrity of network connection before executing
        } else {
            Toast.makeText(this, "Connection failed. Please check connection",
Toast.LENGTH_SHORT);
        }
    }

    public String createHash() { // converts password text to a secure MD5 hash
code
        try {
            MessageDigest digest = MessageDigest.getInstance("MD5");
            digest.update(((EditText)
(findViewById(R.id.et_password))).getText().toString()).getBytes());
            byte messageDigest[] = digest.digest();

            StringBuffer hexString = new StringBuffer();
            for (int i = 0; i < messageDigest.length; i++) {
                hexString.append(Integer.toHexString(0xFF & messageDigest[i]));
            }
            return hexString.toString();
        }
    }
}
```

```
    } catch (NoSuchAlgorithmException e) {  
        e.printStackTrace();  
    }  
    return "";  
}  
}
```

## HomePage

```
package com.example.asdaclickcollectemployeesystem;

import android.os.Bundle;
import android.support.annotation.NonNull;
import android.support.design.widget.NavigationView;
import android.support.v4.view.GravityCompat;
import android.support.v4.widget.DrawerLayout;
import android.support.v7.app.ActionBarDrawerToggle;
import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.Toolbar;
import android.view.MenuItem;
import android.view.View;
import android.widget.EditText;
import android.widget.Switch;
import android.widget.TextView;
import android.widget.Toast;

import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;

public class HomePage extends AppCompatActivity implements
NavigationView.OnNavigationItemSelectedListener {

    private DrawerLayout drawer;
    public String currentorderid; // stores current orderid for picking
    public WeightedGraph.Graph graph; // stores the large graph of the products in
the store
    public String username; // stores username
    public boolean admin; // stores access level
    public BackgroundWorker backgroundWorker = new BackgroundWorker(this);

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        username = getIntent().getStringExtra("username"); // gets username from
bundle extras
        if (getIntent().getStringExtra("admin").contains("0")) { // gets admin from
bundle extras
            admin = false;
        } else {
            admin = true;
        }

        setContentView(R.layout.activity_home_page);

        NavigationView navigationView = findViewById(R.id.nav_view);
        View headerView = navigationView.getHeaderView(0);
        TextView navUsername = headerView.findViewById(R.id.username_text);
        navUsername.setText(username);

        Toolbar toolbar = findViewById(R.id.toolbar);
        setSupportActionBar(toolbar); // sets toolbar for navigation

        drawer = findViewById(R.id.drawer_layout); // sets drawer for navigation
        navigationView.setNavigationItemSelectedListener(this);

        ActionBarDrawerToggle toggle = new ActionBarDrawerToggle(this, drawer,
toolbar,
        R.string.nav_drawer_open, R.string.nav_drawer_close);
        drawer.addDrawerListener(toggle);
        toggle.syncState();

        if (savedInstanceState == null) {
            getSupportActionBar().setTitle("Uncompleted Orders");
        }

        getSupportFragmentManager().beginTransaction().replace(R.id.fragment_container,
```

```

        new UncompOrdersFragment().addToBackStack(null).commit(); //
starts uncomporders fragment
        navigationView.setCheckedItem(R.id.nav_uncompOrders);
    }

    @Override
    public boolean onNavigationItemSelected(@NonNull MenuItem menuItem) {
        switch (menuItem.getItemId()) {
            case R.id.nav_uncompOrders:
                getSupportActionBar().setTitle("Uncompleted Orders");

                getSupportFragmentManager().beginTransaction().replace(R.id.fragment_container,
                    new UncompOrdersFragment().addToBackStack(null).commit());
                // starts uncomporders fragment
                break;
            case R.id.nav_compOrders:
                if (admin == true) {
                    getSupportActionBar().setTitle("Completed Orders");

                    getSupportFragmentManager().beginTransaction().replace(R.id.fragment_container,
                        new
                        CompOrdersFragment().addToBackStack(null).commit()); // starts comporders fragment
                } else { // verifies access level
                    Toast.makeText(this, "This page is only accessible by
administrators.\nTalk to your manager or section leader for more information.",
                        Toast.LENGTH_LONG).show();
                }
                break;
            case R.id.nav_account:
                getSupportActionBar().setTitle("Account");

                getSupportFragmentManager().beginTransaction().replace(R.id.fragment_container,
                    new AccountFragment().addToBackStack(null).commit()); //
starts account fragment
                break;
            case R.id.nav_logOut:
                this.finish(); // ends activity when user logs out
                break;
        }

        drawer.closeDrawer(GravityCompat.START);
        return true;
    }

    @Override
    public void onBackPressed() { // stores previous fragments for navigation
        if (drawer.isDrawerOpen(GravityCompat.START)) {
            drawer.closeDrawer(GravityCompat.START);
        } else if (getFragmentManager().getBackStackEntryCount() == 0) {
            super.onBackPressed();
        }
        else {
            getFragmentManager().popBackStack();
        }
    }

    public void setTitle(String title) {
        getSupportActionBar().setTitle(title); // sets current title for toolbar
    }

    public void viewUncompOrder() {

        getSupportFragmentManager().beginTransaction().replace(R.id.fragment_container,
            new ViewUncompOrdersFragment().addToBackStack(null).commit()); //
starts uncomporder fragment
    }

```



```

        public void viewCompOrder() {

getSupportFragmentManager().beginTransaction().replace(R.id.fragment_container,
                new ViewCompOrdersFragment()).addToBackStack(null).commit(); //
starts comporder fragment
        }

        public void refreshViewUncompOrder() { // refreshes uncomporder fragment after
completed order

getSupportFragmentManager().beginTransaction().replace(R.id.fragment_container, new
ViewUncompOrdersFragment()).commit();
        }

        public void updateAccount (View view) { // updates username
            if (((EditText)
findViewById(R.id.et_password)).getText().toString().isEmpty()) || (((EditText)
findViewById(R.id.et_password)).getText().toString() == "") ||
                (((EditText)
findViewById(R.id.et_password)).getText().toString().isEmpty()) || (((EditText)
findViewById(R.id.et_password)).getText().toString() == "")) {
                Toast toast = Toast.makeText(this, "Enter a valid username or
password", Toast.LENGTH_LONG);
                toast.show();
            } else {
                String password = createHash(); // creates hash of password
confirmation
                String newUsername = ((EditText)
findViewById(R.id.et_username)).getText().toString(); // stores new username
                backgroundWorker.execute("updateUsername", username, newUsername,
password); // executes background worker
            }
        }

        public void createAccount (View view) { // starts new account fragment
            if (admin == true) {

getSupportFragmentManager().beginTransaction().replace(R.id.fragment_container, new
NewAccountFragment()).addToBackStack(null).commit();
            } else { // verifies access level
                Toast.makeText(this, "This page is only accessible by
administrators.\nTalk to your manager or section leader for more information.",
Toast.LENGTH_LONG).show();
            }
        }

        public void createNewAccount (View view) { // creates new account
            String newEmployeeID = ((EditText)
(findViewById(R.id.et_employeeID))).getText().toString(); // stores new employeeid
            String newFullName = ((EditText)
(findViewById(R.id.et_fullname))).getText().toString(); // stores new full name
            String newUsername = ((EditText)
(findViewById(R.id.et_username))).getText().toString(); // stores new username
            String newAdmin; // stores new admin
            if (((Switch) findViewById(R.id.switch_admin)).isChecked()) {
                newAdmin = "1";
            } else {
                newAdmin = "0";
            }
            String newPassword = createHash();

            backgroundWorker.execute("empNewAccount", newUsername, newPassword,
newEmployeeID, newFullName, newAdmin); // executes background worker
        }

        public String createHash() { // converts password text to a secure MD5 hash
code
            try {

```

```
        MessageDigest digest = MessageDigest.getInstance("MD5");
        digest.update(((EditText)
(findViewById(R.id.et_password))).getText().toString().getBytes());
        byte messageDigest[] = digest.digest();

        StringBuffer hexString = new StringBuffer();
        for (int i = 0; i < messageDigest.length; i++) {
            hexString.append(Integer.toHexString(0xFF & messageDigest[i]));
        }
        return hexString.toString();
    } catch (NoSuchAlgorithmException e) {
        e.printStackTrace();
    }
    return "";
}
}
```

## BackgroundWorker

```
package com.example.asdaclickcollectemployeesystem;

import android.app.AlertDialog;
import android.content.Context;
import android.content.Intent;
import android.os.AsyncTask;
import android.util.Log;
import android.widget.Toast;

import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.OutputStream;
import java.io.OutputStreamWriter;
import java.net.HttpURLConnection;
import java.net.URL;
import java.net.URLEncoder;

public class BackgroundWorker extends AsyncTask<String, Void, String> {
    Context context;
    AlertDialog alertDialog;
    public String username = "";
    String type = "";

    BackgroundWorker(Context ctx) {
        context = ctx;
    }

    @Override
    protected String doInBackground(String... voids) {
        type = voids[0];
        String password;
        String line;
        String empLoginURL = "http://10.0.2.2/empLogin.php";
        String empRegisterURL = "http://10.0.2.2/empRegister.php";
        String updateNameURL = "http://10.0.2.2/updateName.php";
        if (type.equals("empLogin")) { // employee log in, uses emplogin.php
            try {
                username = voids[1];
                password = voids[2];
                URL url = new URL(empLoginURL);
                HttpURLConnection httpURLConnection = (HttpURLConnection)
url.openConnection();
                httpURLConnection.setRequestMethod("POST");
                httpURLConnection.setDoOutput(true);
                httpURLConnection.setDoInput(true);
                OutputStream outputStream = httpURLConnection.getOutputStream();
                BufferedWriter bufferedWriter = new BufferedWriter(new
OutputStreamWriter(outputStream, "UTF-8"));
                String post_data = URLEncoder.encode("username", "UTF-8") + "=" +
URLEncoder.encode(username, "UTF-8") + "&"
+ URLEncoder.encode("password", "UTF-8") + "=" +
URLEncoder.encode(password, "UTF-8");
                bufferedWriter.write(post_data);
                bufferedWriter.flush();
                bufferedWriter.close();
                InputStream inputStream = httpURLConnection.getInputStream();
                BufferedReader bufferedReader = new BufferedReader(new
InputStreamReader(inputStream, "iso-8859-1"));
                String result = "";
                while ((line = bufferedReader.readLine()) != null) {
                    result += line;
                }
                bufferedReader.close();
                inputStream.close();
            }
        }
    }
}
```

```

        httpURLConnection.disconnect();
        return result;
    } catch (Exception e) {
        e.printStackTrace();
    }
} else if (type.equals("empNewAccount")) { // employee register, uses
empregister.php
    try {
        username = voids[1];
        password = voids[2];
        String employeeid = voids[3];
        String fullname = voids[4];
        String admin = voids[5];
        URL url = new URL(empRegisterURL);
        HttpURLConnection httpURLConnection = (HttpURLConnection)
url.openConnection();
        httpURLConnection.setRequestMethod("POST");
        httpURLConnection.setDoOutput(true);
        httpURLConnection.setDoInput(true);
        OutputStream outputStream = httpURLConnection.getOutputStream();
        BufferedWriter bufferedWriter = new BufferedWriter(new
OutputStreamWriter(outputStream, "UTF-8"));
        String post_data = URLEncoder.encode("username", "UTF-8") + "=" +
URLEncoder.encode(username, "UTF-8") + "&"
            + URLEncoder.encode("password", "UTF-8") + "=" +
URLEncoder.encode(password, "UTF-8") + "&"
            + URLEncoder.encode("employeeid", "UTF-8") + "=" +
URLEncoder.encode(employeeid, "UTF-8") + "&"
            + URLEncoder.encode("fullname", "UTF-8") + "=" +
URLEncoder.encode(fullname, "UTF-8") + "&"
            + URLEncoder.encode("admin", "UTF-8") + "=" +
URLEncoder.encode(admin, "UTF-8");
        bufferedWriter.write(post_data);
        bufferedWriter.flush();
        bufferedWriter.close();
        InputStream inputStream = httpURLConnection.getInputStream();
        BufferedReader bufferedReader = new BufferedReader(new
InputStreamReader(inputStream, "iso-8859-1"));
        String result = "";
        while ((line = bufferedReader.readLine()) != null) {
            try {
                result += line; // reads in echo message
            } catch (Exception e) {
                Log.e("Error", e.toString());
            }
        }
        bufferedReader.close();
        inputStream.close();
        httpURLConnection.disconnect();
        return result;
    } catch (Exception e) {
        e.printStackTrace();
    }
} else if (type.equals("updateUsername")) { // update username, uses
updatename.php
    try {
        username = voids[1];
        String newusername = voids[2];
        password = voids[3];
        URL url = new URL(updateNameURL);
        HttpURLConnection httpURLConnection = (HttpURLConnection)
url.openConnection();
        httpURLConnection.setRequestMethod("POST");
        httpURLConnection.setDoOutput(true);
        httpURLConnection.setDoInput(true);
        OutputStream outputStream = httpURLConnection.getOutputStream();
        BufferedWriter bufferedWriter = new BufferedWriter(new
OutputStreamWriter(outputStream, "UTF-8"));

```

```

        String post_data = URLEncoder.encode("username", "UTF-8") + "=" +
URLEncoder.encode(username, "UTF-8") + "&"
        + URLEncoder.encode("newusername", "UTF-8") + "=" +
URLEncoder.encode(newusername, "UTF-8") + "&"
        + URLEncoder.encode("password", "UTF-8") + "=" +
URLEncoder.encode(password, "UTF-8");
        bufferedWriter.write(post_data);
        bufferedWriter.flush();
        bufferedWriter.close();
        InputStream inputStream = httpURLConnection.getInputStream();
        BufferedReader bufferedReader = new BufferedReader(new
InputStreamReader(inputStream, "iso-8859-1"));
        String result = "";
        while ((line = bufferedReader.readLine()) != null) {
            try {
                result += line; // reads in echo message
            } catch (Exception e) {
                Log.e("Error", e.toString());
            }
        }
        bufferedReader.close();
        inputStream.close();
        httpURLConnection.disconnect();
        return result;
    } catch (Exception e) {
        e.printStackTrace();
    }
}
return null;
}

@Override
protected void onPreExecute() {
    alertDialog = new AlertDialog.Builder(context).create();
    alertDialog.setTitle("Connection Status");
}

@Override
protected void onPostExecute(String result) { // determines whether querying
was successful
    if (type.equals("empLogin")) {
        if (result.contains("Login not successful")) { // responds with
appropriate error message
            alertDialog.setMessage("Incorrect credentials");
            alertDialog.show();
        } else {
            StartHome(result);
        }
    } else if (type.equals("empNewAccount")){
        if (result.contains("Register not successful")) { // responds with
appropriate error message
            alertDialog.setMessage("Register not successful" );
            alertDialog.show();
        } else if ((result == "") || result.isEmpty()) { // responds with
appropriate error message
            alertDialog.setMessage("Something went wrong. Please try again");
            alertDialog.show();
        }
        else {
            Toast toast = Toast.makeText(context, "Account Created
Successfully", Toast.LENGTH_LONG);
            toast.show();
        }
    } else {
        if (result.contains("An error occurred") || (result == "") ||
result.isEmpty()) { // responds with appropriate error message
            alertDialog.setMessage("Something went wrong. Please try again");
            alertDialog.show();
        }
    }
}

```

```

        }
        else {
            Toast toast = Toast.makeText(context, "Account Updated
Successfully", Toast.LENGTH_LONG);
            toast.show();
        }
    }

    public void StartHome(String result) {
        String admin = result.split("\\|")[1];
        Intent intent = new Intent(context, HomePage.class); // adds bundle to
store username and admin access
        intent.putExtra("username", username);
        intent.putExtra("admin", admin);
        context.startActivity(intent);
    }

    @Override
    protected void onProgressUpdate(Void... values) {
        super.onProgressUpdate(values);
    }
}

```

## UncompOrdersFragment

```
package com.example.asdaclickcollectemployeesystem;

import android.app.AlertDialog;
import android.os.Bundle;
import android.os.StrictMode;
import android.support.annotation.NonNull;
import android.support.annotation.Nullable;
import android.support.v4.app.Fragment;
import android.support.v7.widget.CardView;
import android.view.Gravity;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.LinearLayout;
import android.widget.TextView;

import java.io.BufferedReader;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
import java.net.URL;

import static android.widget.LinearLayout.HORIZONTAL;
import static android.widget.LinearLayout.VERTICAL;

public class UncompOrdersFragment extends Fragment {

    int numOrders = 0;
    AlertDialog alertDialog;

    @Nullable
    @Override
    public View onCreateView(@NonNull LayoutInflater inflater, @Nullable ViewGroup container, @Nullable Bundle savedInstanceState) {

        final HomePage homePage = (HomePage) getActivity();
        homePage.setTitle("Uncompleted Orders");

        View view = inflater.inflate(R.layout.fragment_uncomp_orders, container, false);
        LinearLayout layout = view.findViewById(R.id.fragment_container); // sets layout and layoutparams

        LinearLayout.LayoutParams cvParams = new LinearLayout.LayoutParams(
            LinearLayout.LayoutParams.MATCH_PARENT,
            LinearLayout.LayoutParams.WRAP_CONTENT);
        cvParams.setMargins(10, 10, 10, 10);
        LinearLayout.LayoutParams tvParams = new LinearLayout.LayoutParams(
            LinearLayout.LayoutParams.MATCH_PARENT,
            LinearLayout.LayoutParams.WRAP_CONTENT);
        tvParams.setMargins(5, 5, 5, 5);
        LinearLayout.LayoutParams btnParams = new LinearLayout.LayoutParams(
            LinearLayout.LayoutParams.WRAP_CONTENT,
            LinearLayout.LayoutParams.WRAP_CONTENT);
        btnParams.setMargins(25, 5, 5, 10);
        LinearLayout.LayoutParams layout1Params = new LinearLayout.LayoutParams(
            LinearLayout.LayoutParams.MATCH_PARENT,
            LinearLayout.LayoutParams.MATCH_PARENT);
        LinearLayout.LayoutParams layout2Params = new LinearLayout.LayoutParams(
            LinearLayout.LayoutParams.WRAP_CONTENT,
            LinearLayout.LayoutParams.WRAP_CONTENT);
        layout2Params.setMargins(10, 10, 100, 10);
        LinearLayout.LayoutParams layout3Params = new LinearLayout.LayoutParams(
            LinearLayout.LayoutParams.WRAP_CONTENT,
            LinearLayout.LayoutParams.MATCH_PARENT);
```

```

String line = "";
try {
    StrictMode.ThreadPolicy policy = new
StrictMode.ThreadPolicy.Builder().permitAll().build();
    StrictMode.setThreadPolicy(policy);
    String retrieveUncompOrdersURL =
"http://10.0.2.2/retrieveUncompOrders.php";
    URL url = new URL(retrieveUncompOrdersURL);
    HttpURLConnection httpURLConnection = (HttpURLConnection)
url.openConnection();
    httpURLConnection.setDoOutput(true);
    httpURLConnection.setDoInput(true);
    InputStream inputStream = httpURLConnection.getInputStream();
    BufferedReader bufferedReader = new BufferedReader(new
InputStreamReader(inputStream, "iso-8859-1"));
    while ((line = bufferedReader.readLine()) != null) { // sets layout and
cardview for each order
        String[] parts = line.split("\\|");

        if (Integer.parseInt(parts[2]) == 0) {
            break;
        }

        CardView cv = new CardView(getContext());
        LinearLayout layout1 = new LinearLayout(getContext());
        LinearLayout layout2 = new LinearLayout(getContext());
        LinearLayout layout3 = new LinearLayout(getContext());
        layout1.setOrientation(HORIZONTAL);
        layout2.setOrientation(VERTICAL);
        layout3.setOrientation(HORIZONTAL);

        final String orderid = parts[0];
        TextView orderID = new TextView(getContext());
        orderID.setText("Order ID: " + parts[0]);
        TextView orderDate = new TextView(getContext());
        orderDate.setText("Order Date: " + parts[1]);
        TextView numItems = new TextView(getContext());
        numItems.setText("Items Remaining: " + parts[2]);

        Button viewOrder = new Button(getContext());
        viewOrder.setText("View Order");
        viewOrder.setOnClickListener(new Button.OnClickListener() {
            public void onClick(View v) {
                homePage.currentorderid = orderid;
                homePage.viewUncompOrder();
            }
        });

        layout1.setLayoutParams(layout1Params);
        layout2.setLayoutParams(layout2Params);
        layout3.setLayoutParams(layout3Params);
        layout3.setGravity(Gravity.BOTTOM);
        viewOrder.setLayoutParams(btnParams);
        orderID.setLayoutParams(tvParams);
        orderDate.setLayoutParams(tvParams);
        numItems.setLayoutParams(tvParams);

        layout3.addView(viewOrder);
        layout2.addView(orderID);
        layout2.addView(orderDate);
        layout2.addView(numItems);
        layout1.addView(layout2);
        layout1.addView(layout3);
        cv.addView(layout1);
        cv.setElevation(15);
        cv.setRadius(15);
        cv.setLayoutParams(cvParams);
        layout.addView(cv);

```



```

        numOrders++;
    }
    bufferedReader.close();
    inputStream.close();
    httpURLConnection.disconnect();
} catch (Exception e) {
    alertDialog = new AlertDialog.Builder(getContext()).create();
    alertDialog.setMessage(e.getMessage());
    alertDialog.show();
    e.printStackTrace();
}

if (numOrders == 0) {
    alertDialog = new AlertDialog.Builder(getContext()).create();
    alertDialog.setMessage("Nothing to see here.\nThere aren't any orders right now.");
    alertDialog.show();
}

homePage.graph = new WeightedGraph.Graph();
createInitialGraph(homePage.graph);

return view;
}

public void createInitialGraph(WeightedGraph.Graph graph) { // creates base
graph of store
    for (int i = 1; i <= 14; i++) {
        graph.addEdge(2, i, 29 - i, true);
    }
    for (int i = 23; i <= 28; i++) {
        graph.addEdge(2, i, 57 - i, true);
    }
    for (int i = 1; i < 14; i++) {
        graph.addEdge(1, i, i + 1, true);
    }
    for (int i = 15; i < 28; i++) {
        graph.addEdge(1, i, i + 1, true);
    }
    for (int i = 29; i < 34; i++) {
        graph.addEdge(1, i, i + 1, true);
    }
    graph.addEdge(3, 0, 34, true);
    graph.addEdge(2, 0, 20, true);
    graph.addEdge(2, 35, 17, true);
}
}

```

## ViewUncompOrdersFragment

```
package com.example.asdaclickcollectemployeesystem;

import android.app.AlertDialog;
import android.os.Bundle;
import android.os.StrictMode;
import android.support.annotation.NonNull;
import android.support.annotation.Nullable;
import android.support.v4.app.Fragment;
import android.support.v7.widget.CardView;
import android.util.Pair;
import android.view.Gravity;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.LinearLayout;
import android.widget.TextView;
import android.widget.Toast;

import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.OutputStream;
import java.io.OutputStreamWriter;
import java.net.HttpURLConnection;
import java.net.URL;
import java.net.URLEncoder;
import java.util.ArrayList;
import java.util.Collections;
import java.util.List;
import java.util.Stack;

import static android.widget.LinearLayout.HORIZONTAL;
import static android.widget.LinearLayout.VERTICAL;

public class ViewUncompOrdersFragment extends Fragment {

    public Stack<Integer> fastestPath = new Stack<>();
    List<Integer> requiredV = new ArrayList<>();
    Stack<Integer> tempFastestPath;
    AlertDialog alertDialog;

    @Nullable
    @Override
    public View onCreateView(@NonNull LayoutInflater inflater, @Nullable ViewGroup
container, @Nullable Bundle savedInstanceState) {

        final HomePage homePage = (HomePage) getActivity();
        homePage.setTitle("Picking Order: " + homePage.currentorderid);

        View view = inflater.inflate(R.layout.fragment_view_uncomp_orders,
container, false);
        LinearLayout layout = view.findViewById(R.id.vuo_fragment_container); //
sets layout and layoutparams

        LinearLayout.LayoutParams cvParams = new LinearLayout.LayoutParams(
            LinearLayout.LayoutParams.MATCH_PARENT,
            LinearLayout.LayoutParams.WRAP_CONTENT);
        cvParams.setMargins(10, 10, 10, 10);
        LinearLayout.LayoutParams tvParams = new LinearLayout.LayoutParams(
            LinearLayout.LayoutParams.MATCH_PARENT,
            LinearLayout.LayoutParams.WRAP_CONTENT);
        tvParams.setMargins(5, 5, 5, 5);
        LinearLayout.LayoutParams btnParams = new LinearLayout.LayoutParams(
            LinearLayout.LayoutParams.WRAP_CONTENT,
```

```

        LinearLayout.LayoutParams.WRAP_CONTENT);
btnParams.setMargins(25, 5, 5, 10);
LinearLayout.LayoutParams layout1Params = new LinearLayout.LayoutParams(
    LinearLayout.LayoutParams.MATCH_PARENT,
    LinearLayout.LayoutParams.MATCH_PARENT);
LinearLayout.LayoutParams layout2Params = new LinearLayout.LayoutParams(
    LinearLayout.LayoutParams.WRAP_CONTENT,
    LinearLayout.LayoutParams.WRAP_CONTENT);
layout2Params.setMargins(10, 10, 100, 10);
LinearLayout.LayoutParams layout3Params = new LinearLayout.LayoutParams(
    LinearLayout.LayoutParams.WRAP_CONTENT,
    LinearLayout.LayoutParams.MATCH_PARENT);

String line = "";

final String orderid = homePage.currentorderid;

List<String[]> productList = new ArrayList<>();

try {
    StrictMode.ThreadPolicy policy = new
StrictMode.ThreadPolicy.Builder().permitAll().build();
    StrictMode.setThreadPolicy(policy);
    String retrieveOrderProductsURL =
"http://10.0.2.2/retrieveOrderProducts.php";
    String updateOrderURL = "http://10.0.2.2/updateOrder.php";
    final String pickItemURL = "http://10.0.2.2/pickItem.php";
    URL url = new URL(retrieveOrderProductsURL);
    HttpURLConnection httpURLConnection = (HttpURLConnection)
url.openConnection();
    httpURLConnection.setRequestMethod("POST");
    httpURLConnection.setDoOutput(true);
    httpURLConnection.setDoInput(true);
    OutputStream outputStream = httpURLConnection.getOutputStream();
    BufferedWriter bufferedWriter = new BufferedWriter(new
OutputStreamWriter(outputStream, "UTF-8"));
    String post_data = URLEncoder.encode("orderid", "UTF-8") + "=" +
URLEncoder.encode(orderid, "UTF-8");
    bufferedWriter.write(post_data);
    bufferedWriter.flush();
    bufferedWriter.close();
    InputStream inputStream = httpURLConnection.getInputStream();
    BufferedReader bufferedReader = new BufferedReader(new
InputStreamReader(inputStream, "iso-8859-1"));
    while ((line = bufferedReader.readLine()) != null) { // uses layouts
and cardview to present each item
        if (line.length() < 2) {
            break;
        }

        String[] parts = line.split("\\|");
        String[] tempArray = {(parts[0]), (parts[1]), (parts[2]),
(parts[3]), (parts[4])};
        productList.add(tempArray); // stores information about each
product
    }

    requiredV.add(0); // adds a 0 'start' node
    requiredV.add(35); // adds a 35 'end' node

    for (String[] s : productList) {
        if (!requiredV.contains(Integer.parseInt(s[3]))) {
            requiredV.add(Integer.parseInt(s[3])); // adds each node in
productlist
        }
    }

    createSmallGraph(homePage.graph); // creates smaller graph of required

```

```

products
    List<String[]> orderedList = new ArrayList<>(); // stores the fastest
order of traversal

    if (productList.size() > 1) {
        while (!tempFastestPath.isEmpty()) {
            int currentAisle = tempFastestPath.pop(); // stores current
aisle to visit

            for (String[] s : productList) {
                if (Integer.parseInt(s[3]) == currentAisle) {
                    orderedList.add(s); // adds current aisle to fastest
path
                }
            }
        }
    }
    else if (productList.size() == 1) {
        orderedList.add(productList.get(0));
    } else {
        alertDialog = new AlertDialog.Builder(getContext()).create();
        alertDialog.setMessage("There are no more products in this order");
        alertDialog.show();

        url = new URL(updateOrderURL);
        httpURLConnection = (HttpURLConnection) url.openConnection();
        httpURLConnection.setRequestMethod("POST");
        httpURLConnection.setDoOutput(true);
        httpURLConnection.setDoInput(true);
        outputStream = httpURLConnection.getOutputStream();
        bufferedWriter = new BufferedWriter(new
OutputStreamWriter(outputStream, "UTF-8"));
        post_data = URLEncoder.encode("orderid", "UTF-8") + "=" +
URLEncoder.encode(orderid, "UTF-8") + "&"
            + URLEncoder.encode("username", "UTF-8") + "=" +
URLEncoder.encode(homePage.username, "UTF-8");
        bufferedWriter.write(post_data);
        bufferedWriter.flush();
        bufferedWriter.close();
        inputStream = httpURLConnection.getInputStream();
        bufferedReader = new BufferedReader(new
InputStreamReader(inputStream, "iso-8859-1"));
        if (bufferedReader.readLine() != null) {
            Toast toast = Toast.makeText(getContext(),
bufferedReader.readLine(), Toast.LENGTH_SHORT);
            toast.show();
        }
    }
    Collections.reverse(orderedList);

    for (String[] s : orderedList) {
        CardView cv = new CardView(getContext());
        LinearLayout layout1 = new LinearLayout(getContext());
        LinearLayout layout2 = new LinearLayout(getContext());
        LinearLayout layout3 = new LinearLayout(getContext());
        layout1.setOrientation(HORIZONTAL);
        layout2.setOrientation(VERTICAL);
        layout3.setOrientation(HORIZONTAL);
        final String upc = s[1];
        TextView orderID = new TextView(getContext());
        TextView orderDate = new TextView(getContext());
        TextView itemQuantity = new TextView(getContext());
        TextView itemAisle = new TextView(getContext());
        TextView itemLocation = new TextView(getContext());
        try { // sets layout for each item in order
            orderID.setText("Name: " + s[0]);
            orderDate.setText("UPC: " + s[1]);
            itemQuantity.setText("Quantity: " + s[2]);

```

```

        itemAisle.setText("Aisle: " + s[3]);
        itemLocation.setText("Location: " + s[4]);
        Pair<String, Integer> thisItem = new Pair<>(s[1],
Integer.parseInt(s[3]));
        Button pickItem = new Button(getContext());
        pickItem.setText("Pick Item");
        pickItem.setOnClickListener(new Button.OnClickListener() {
            public void onClick(View v) {
                try {
                    URL url = new URL(pickItemURL);
                    HttpURLConnection httpURLConnection =
(HttpURLConnection) url.openConnection();
                    httpURLConnection.setRequestMethod("POST");
                    httpURLConnection.setDoOutput(true);
                    httpURLConnection.setDoInput(true);
                    OutputStream outputStream =
httpURLConnection.getOutputStream();
                    BufferedWriter bufferedWriter = new
BufferedWriter(new OutputStreamWriter(outputStream, "UTF-8"));
                    String post_data = URLEncoder.encode("orderid",
"UTF-8") + "=" + URLEncoder.encode(orderid, "UTF-8") + "&"
+ URLEncoder.encode("upc", "UTF-8") + "=" +
URLEncoder.encode(upc, "UTF-8");
                    bufferedWriter.write(post_data);
                    bufferedWriter.flush();
                    bufferedWriter.close();
                    InputStream inputStream =
httpURLConnection.getInputStream();
                    BufferedReader bufferedReader = new
BufferedReader(new InputStreamReader(inputStream, "iso-8859-1"));
                    if (bufferedReader.readLine() != null) {
                        Toast toast = Toast.makeText(getContext(),
bufferedReader.readLine(), Toast.LENGTH_SHORT);
                        toast.show();
                    } else {
                        homePage.refreshViewUncompOrder();
                    }
                } catch (Exception e) {
                    AlertDialog alertDialog = new
AlertDialog.Builder(getContext()).create();
                    alertDialog.setMessage(e.getMessage());
                    alertDialog.show();
                    e.printStackTrace();
                }
            }
        });

        layout1.setLayoutParams(layout1Params);
        layout2.setLayoutParams(layout2Params);
        layout3.setLayoutParams(layout3Params);
        layout3.setGravity(Gravity.RIGHT);
        pickItem.setLayoutParams(btnParams);
        pickItem.setGravity(Gravity.RIGHT);
        orderID.setLayoutParams(tvParams);
        orderDate.setLayoutParams(tvParams);
        itemQuantity.setLayoutParams(tvParams);
        itemAisle.setLayoutParams(tvParams);
        itemLocation.setLayoutParams(tvParams);

        layout3.addView(pickItem);
        layout2.addView(orderID);
        layout2.addView(orderDate);
        layout2.addView(itemQuantity);
        layout2.addView(itemAisle);
        layout2.addView(itemLocation);
        layout1.addView(layout2);
        layout1.addView(layout3);
        cv.addView(layout1);

```

```

        cv.setElevation(15);
        cv.setRadius(15);
        cv.setLayoutParams(cvParams);
        layout.addView(cv);
    } catch (Exception e) {
        AlertDialog alertDialog = new
AlertDialog.Builder(getContext()).create();
        alertDialog.setMessage("Some products were not located on the
system");

        alertDialog.show();
        e.printStackTrace();
        layout.removeView(cv);
    }
    }
    bufferedReader.close();
    inputStream.close();
    httpURLConnection.disconnect();
} catch (Exception e) {
    alertDialog = new AlertDialog.Builder(getContext()).create();
    alertDialog.setMessage(e.getMessage());
    alertDialog.show();
    e.printStackTrace();
}

return view;
}

public void createSmallGraph(WeightedGraph.Graph graph) { // creates smaller
subgraph of required nodes

    WeightedGraph.Graph newGraph = new WeightedGraph.Graph();
    int[][] fastPathArray;
    int a = 0;

    for (int x : requiredV) { // iterates through each required vertex and adds
path
        fastPathArray = graph.fastPath(x);
        for (int i = 1; i <= 35; i++) {
            if (requiredV.contains(fastPathArray[i][0])) {
                newGraph.addEdge(fastPathArray[i][1], fastPathArray[0][0],
fastPathArray[i][0], false);
                a++;
            }
        }
    }

    requiredV.remove(Integer.valueOf(35)); // removes 35 'end' node
    tempFastestPath = newGraph.calculateFastestPathAroundStore(requiredV); //
calculates fastest path for each required vertex
    }
}

```

## CompOrdersFragment

```
package com.example.asdaclickcollectemployeesystem;

import android.app.AlertDialog;
import android.os.Bundle;
import android.os.StrictMode;
import android.support.annotation.NonNull;
import android.support.annotation.Nullable;
import android.support.v4.app.Fragment;
import android.support.v7.widget.CardView;
import android.view.Gravity;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.LinearLayout;
import android.widget.TextView;

import java.io.BufferedReader;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
import java.net.URL;

import static android.widget.LinearLayout.HORIZONTAL;
import static android.widget.LinearLayout.VERTICAL;

public class CompOrdersFragment extends Fragment {

    int numOrders = 0; // stores the total number of orders
    AlertDialog alertDialog;

    @Nullable
    @Override
    public View onCreateView(@NonNull LayoutInflater inflater, @Nullable ViewGroup container, @Nullable Bundle savedInstanceState) {

        final HomePage homePage = (HomePage) getActivity();
        homePage.setTitle("Completed Orders");

        View view = inflater.inflate(R.layout.fragment_comp_orders, container, false);
        LinearLayout layout = view.findViewById(R.id.fragment_container); // sets layout and layoutparams

        LinearLayout.LayoutParams cvParams = new LinearLayout.LayoutParams(
            LinearLayout.LayoutParams.MATCH_PARENT,
            LinearLayout.LayoutParams.WRAP_CONTENT);
        cvParams.setMargins(10, 10, 10, 10);
        LinearLayout.LayoutParams tvParams = new LinearLayout.LayoutParams(
            LinearLayout.LayoutParams.MATCH_PARENT,
            LinearLayout.LayoutParams.WRAP_CONTENT);
        tvParams.setMargins(5, 5, 5, 5);
        LinearLayout.LayoutParams btnParams = new LinearLayout.LayoutParams(
            LinearLayout.LayoutParams.WRAP_CONTENT,
            LinearLayout.LayoutParams.WRAP_CONTENT);
        btnParams.setMargins(25, 5, 5, 10);
        LinearLayout.LayoutParams layout1Params = new LinearLayout.LayoutParams(
            LinearLayout.LayoutParams.MATCH_PARENT,
            LinearLayout.LayoutParams.MATCH_PARENT);
        LinearLayout.LayoutParams layout2Params = new LinearLayout.LayoutParams(
            LinearLayout.LayoutParams.WRAP_CONTENT,
            LinearLayout.LayoutParams.WRAP_CONTENT);
        layout2Params.setMargins(10, 10, 100, 10);
        LinearLayout.LayoutParams layout3Params = new LinearLayout.LayoutParams(
            LinearLayout.LayoutParams.WRAP_CONTENT,
            LinearLayout.LayoutParams.MATCH_PARENT);
```

```

        String line = "";
        try {
            StrictMode.ThreadPolicy policy = new
StrictMode.ThreadPolicy.Builder().permitAll().build();
            StrictMode.setThreadPolicy(policy);
            String retrieveUncompOrdersURL =
"http://10.0.2.2/retrieveCompOrders.php";
            URL url = new URL(retrieveUncompOrdersURL);
            HttpURLConnection httpURLConnection = (HttpURLConnection)
url.openConnection();
            httpURLConnection.setDoOutput(true);
            httpURLConnection.setDoInput(true);
            InputStream inputStream = httpURLConnection.getInputStream();
            BufferedReader bufferedReader = new BufferedReader(new
InputStreamReader(inputStream, "iso-8859-1"));
            while ((line = bufferedReader.readLine()) != null) { // sets layouts
and cardview for each order
                String[] parts = line.split("\\|");

                CardView cv = new CardView(getContext());
                LinearLayout layout1 = new LinearLayout(getContext());
                LinearLayout layout2 = new LinearLayout(getContext());
                LinearLayout layout3 = new LinearLayout(getContext());
                layout1.setOrientation(HORIZONTAL);
                layout2.setOrientation(VERTICAL);
                layout3.setOrientation(HORIZONTAL);

                final String orderid = parts[0];
                TextView orderID = new TextView(getContext());
                orderID.setText("Order ID: " + parts[0]);
                TextView orderDate = new TextView(getContext());
                orderDate.setText("Order Date: " + parts[1]);
                TextView numItems = new TextView(getContext());
                numItems.setText("Number of Items: " + parts[2]);

                Button viewOrder = new Button(getContext());
                viewOrder.setText("View Order");
                viewOrder.setOnClickListener(new Button.OnClickListener() {
                    public void onClick(View v) {
                        homePage.currentorderid = orderid;
                        homePage.viewCompOrder();
                    }
                });

                layout1.setLayoutParams(layout1Params);
                layout2.setLayoutParams(layout2Params);
                layout3.setLayoutParams(layout3Params);
                layout3.setGravity(Gravity.BOTTOM);
                viewOrder.setLayoutParams(btnParams);
                orderID.setLayoutParams(tvParams);
                orderDate.setLayoutParams(tvParams);
                numItems.setLayoutParams(tvParams);

                layout3.addView(viewOrder);
                layout2.addView(orderID);
                layout2.addView(orderDate);
                layout2.addView(numItems);
                layout1.addView(layout2);
                layout1.addView(layout3);
                cv.addView(layout1);
                cv.setElevation(15);
                cv.setRadius(15);
                cv.setLayoutParams(cvParams);
                layout.addView(cv);

                numOrders++;
            }
            bufferedReader.close();

```



```

        inputStream.close();
        httpURLConnection.disconnect();
    } catch (Exception e) {
        alertDialog = new AlertDialog.Builder(getContext()).create();
        alertDialog.setMessage(e.getMessage());
        alertDialog.show();
        e.printStackTrace();
    }

    if (numOrders == 0) { // outputs appropriate error message when no
completed orders exist
        alertDialog = new AlertDialog.Builder(getContext()).create();
        alertDialog.setMessage("Nothing to see here.\nYou haven't completed any
orders yet.");
        alertDialog.show();
    }

    return view;
}
}

```

## ViewCompOrdersFragment

```
package com.example.asdaclickcollectemployeesystem;

import android.os.Bundle;
import android.support.annotation.NonNull;
import android.support.annotation.Nullable;
import android.support.v4.app.Fragment;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;

public class ViewCompOrdersFragment extends Fragment {

    @Nullable
    @Override
    public View onCreateView(@NonNull LayoutInflater inflater, @Nullable ViewGroup
container, @Nullable Bundle savedInstanceState) {

        HomePage homePage = (HomePage) getActivity();
        homePage.setTitle("Picked Order: " + homePage.currentorderid); // sets
title for toolbar

        return inflater.inflate(R.layout.fragment_view_comp_orders, container,
false);
    }
}
```

## AccountFragment

```
package com.example.asdaclickcollectemployeesystem;

import android.os.Bundle;
import android.support.annotation.NonNull;
import android.support.annotation.Nullable;
import android.support.v4.app.Fragment;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;

public class AccountFragment extends Fragment {

    @Nullable
    @Override
    public View onCreateView(@NonNull LayoutInflater inflater, @Nullable ViewGroup
container, @Nullable Bundle savedInstanceState) {

        HomePage homePage = (HomePage) getActivity();
        homePage.setTitle("Account Settings"); // sets title for toolbar

        return inflater.inflate(R.layout.fragment_account, container, false);
    }
}
```

## NewAccountFragment

```
package com.example.asdaclickcollectemployeesystem;

import android.os.Bundle;
import android.support.annotation.NonNull;
import android.support.annotation.Nullable;
import android.support.v4.app.Fragment;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;

public class NewAccountFragment extends Fragment {

    @Nullable
    @Override
    public View onCreateView(@NonNull LayoutInflater inflater, @Nullable ViewGroup
container, @Nullable Bundle savedInstanceState) {

        HomePage homePage = (HomePage) getActivity();
        homePage.setTitle("Create New Account"); // sets title for toolbar

        return inflater.inflate(R.layout.fragment_new_account, container, false);
    }
}
```

## WeightedGraph

```
package com.example.asdaclickcollectemployeesystem;

import android.util.Pair;

import java.util.ArrayList;
import java.util.LinkedList;
import java.util.List;
import java.util.Stack;

public class WeightedGraph {

    public static class Graph {

        public LinkedList<Edge>[] adjacencyList; // stores a linked list of edges
        List<Pair<Stack<Integer>, Integer>> allPaths = new ArrayList<>();

        Graph() {
            adjacencyList = new LinkedList[36]; //initialize adjacency lists for
all the vertices
            for (int i = 0; i <= 35; i++) {
                adjacencyList[i] = new LinkedList<>();
            }
        }

        public void addEdge(int w, int v1, int v2, boolean both) {
            Edge edge1 = new Edge(w, v1, v2);
            Edge edge2 = new Edge(w, v2, v1);

            if (both) {
                adjacencyList[v1].add(edge1);
                adjacencyList[v2].add(edge2);
            } else {
                adjacencyList[v1].add(edge1);
            }
        }

        public Stack<String> printGraph() {
            List<Edge> list;
            Stack<String> adjList = new Stack<>();
            for (int i = 0; i < 36; i++) { // iterates through each vertex in the
graph
                list = adjacencyList[i]; // creates a linked list of each edge
connected to the selected vertex
                for (int j = 0; j < list.size(); j++) {
                    adjList.push(" Vertex-" + i + " is connected to " +
list.get(j).vName2 + " with weight " + list.get(j).weight + "\n");
                }
            }
            return adjList;
        }

        public int[][] fastPath(int v1) { // finds the fastest path between the
specified vertex and all the other vertices
            List<Integer> unvisited = new ArrayList<>();
            int[][] dijkstraFastPath = new int[36][3]; // 36 rows of 3 columns - |
vertex to be visited | total distance | previous vertex |
            for (int i = 0; i <= 35; i++) {
                dijkstraFastPath[i][0] = ((v1 + i) % 36); // stores which vertex is
being measured
                dijkstraFastPath[i][1] = 2147483647; // initiates path lengths
unvisited.add(dijkstraFastPath[i][0]); // adds all vertices to
unvisited list
            }
            dijkstraFastPath[0][1] = 0; // sets distance of source to source to 0

            while (!unvisited.isEmpty()) {
```

```

        Integer closestV = 0; // stores closest vertex
        int smallestW = 2147483647;
        for (int i = 0; i < 36; i++) {
            if ((dijkstraFastPath[i][1] < smallestW) &&
(unvisited.contains(dijkstraFastPath[i][0]))) {
                closestV = dijkstraFastPath[i][0];
                smallestW = dijkstraFastPath[i][1];
            }
        }

        unvisited.remove(Integer.valueOf(closestV)); // remove vertex from
unvisited

        for (Edge x : adjacencyList[closestV]) {
            if (unvisited.contains(x.vName2)) {
                int alt = smallestW + x.weight;

                if (v1 > x.vName2) {
                    if (dijkstraFastPath[(x.vName2) + 36] - v1][1] > alt)
{
                        dijkstraFastPath[(x.vName2) + 36] - v1][1] = alt;
                        dijkstraFastPath[(x.vName2) + 36] - v1][2] =
x.vName1;
                    }

                } else if (v1 < x.vName2) {
                    if (dijkstraFastPath[(x.vName2) - v1][1] > alt) {
                        dijkstraFastPath[(x.vName2) - v1][1] = alt;
                        dijkstraFastPath[(x.vName2) - v1][2] = x.vName1;
                    }
                }
            }
        }
        return dijkstraFastPath;
    }

    public Stack<Integer> calculateFastestPathAroundStore (List<Integer>
requiredV) {
        int currentCounter = 0; // initiates the counter through the allPaths
list

        Stack<Integer> stack = new Stack<>(); // initiates stack and weight
values for new path
        stack.push(0);
        Integer weight = 0;
        Pair<Stack<Integer>, Integer> pair = new Pair<>(stack, weight);
        allPaths.add(pair);

        calculateEachPath(requiredV, currentCounter);

        int shortestPathPointer = 0;

        for (int i = 0; i < allPaths.size(); i++) {
            if (allPaths.get(i).second <
allPaths.get(shortestPathPointer).second) {
                shortestPathPointer = i;
            }
        }

        return allPaths.get(shortestPathPointer).first;
    }

    public void calculateEachPath(List<Integer> requiredV, int currentCounter)
{

```

```

        boolean pathsCalculated = false; // boolean to represent whether all
paths in allPaths are requiredV size long

        List<Pair<Integer, Integer>> weightsList = new ArrayList<>(); // stores
all connected edges to current v on current path

        while (!pathsCalculated) {

            weightsList.clear();
            Pair<Integer, Integer> tempPair;
            int currentV = allPaths.get(currentCounter).first.peek();

            for (Edge e : adjacencyList[currentV]) {
                Pair<Integer, Integer> pair = new Pair<>(e.vName2, e.weight);
                weightsList.add(pair);
            }

            if (allPaths.get(currentCounter).first.size() == requiredV.size())
{

                weightsList.clear();

                for (Edge e : adjacencyList[35]) {
                    if (e.vName2 == currentV) {

                        tempPair = new Pair<>(e.vName1, e.weight);
                        weightsList.add(tempPair);
                        break;
                    }
                }

                } else {

                    List<Pair<Integer, Integer>> pairsToRemove = new ArrayList<>();
                    for (Pair<Integer, Integer> p : weightsList) { // iterates
through each connected vertex
                        if (allPaths.get(currentCounter).first.contains(p.first)) {
// removes edge from list if vertex already visited
                            pairsToRemove.add(p);
                        } else if (p.first == 35) {
                            pairsToRemove.add(p);
                        }
                    }

                    for (Pair<Integer, Integer> p : pairsToRemove) {
                        weightsList.remove(p);
                    }

                }

                boolean swapped = true; // boolean to represent whether the
weightsList is sorted, bubble sort

                while (swapped) { // orders connected edges by weight, bubble sort
                    swapped = false;

                    for (int i = 1; i < weightsList.size(); i++) {
                        if (weightsList.get(i - 1).second >
weightsList.get(i).second) {

                            tempPair = weightsList.get(i - 1);
                            weightsList.set(i - 1, weightsList.get(i));
                            weightsList.set(i, tempPair);
                            swapped = true;
                        }
                    }
                }
            }
        }

```

```

    }

    if (weightsList.size() == 1) {
        Stack<Integer> stack = allPaths.get(currentCounter).first;
        stack.push(weightsList.get(0).first);
        Integer weight = allPaths.get(currentCounter).second +
weightsList.get(0).second;
        Pair<Stack<Integer>, Integer> pair = new Pair<>(stack, weight);
        allPaths.set(currentCounter, pair);
    } else {
        for (int i = 1; i < weightsList.size(); i++) {
            if (weightsList.get(i).second == weightsList.get(0).second)
{
                Stack<Integer> stack = (Stack<Integer>)
allPaths.get(currentCounter).first.clone();
                Integer weight = allPaths.get(currentCounter).second +
weightsList.get(i).second;
                stack.push(weightsList.get(i).first);
                Pair<Stack<Integer>, Integer> pair = new Pair<>(stack,
weight);
                allPaths.add(pair);
            } else {
                break;
            }
        }

        Stack<Integer> stack = (Stack<Integer>)
allPaths.get(currentCounter).first.clone();
        stack.push(weightsList.get(0).first);
        Integer weight = allPaths.get(currentCounter).second +
weightsList.get(0).second;
        Pair<Stack<Integer>, Integer> pair = new Pair<>(stack, weight);
        allPaths.set(currentCounter, pair);
    }

    if (allPaths.get(currentCounter).first.size() == requiredV.size() +
1) {
        if (allPaths.size() == currentCounter + 1) {
            pathsCalculated = true;
        } else {
            currentCounter++;
        }
    }
}

}

}

public static class Edge {
    public int weight, vName1, vName2;

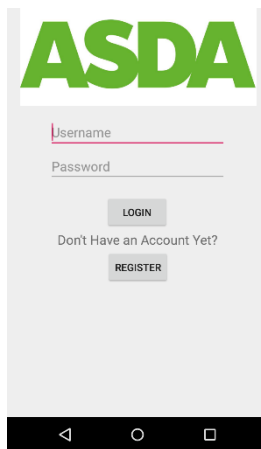
    public Edge(int w, int v1, int v2) {
        weight = w;
        vName1 = v1;
        vName2 = v2;
    }
}
}

```

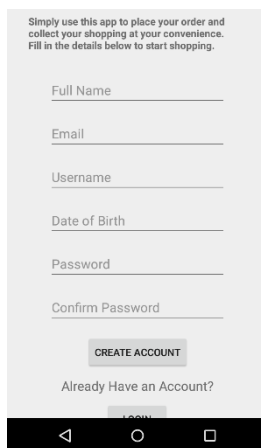


## Layouts and UI Design – Customer Application

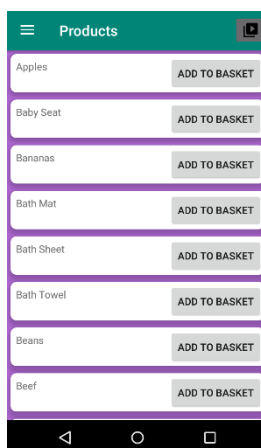
fragment\_login.xml



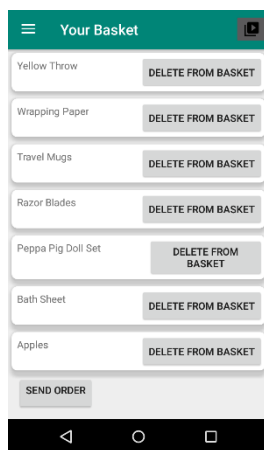
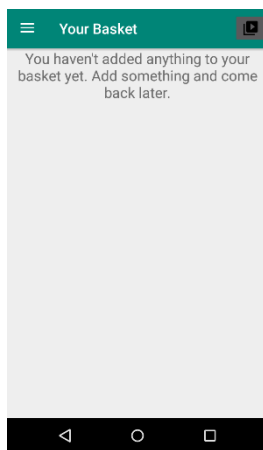
fragment\_register.xml



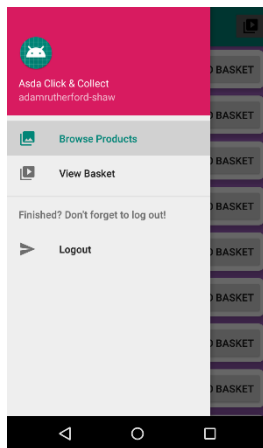
fragment\_products.xml



## fragment\_basket.xml

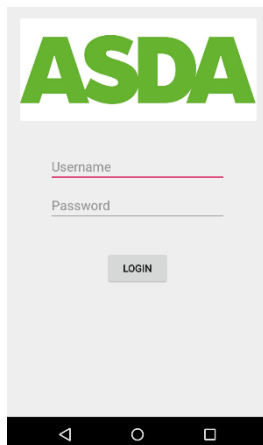


## drawer\_menu.xml

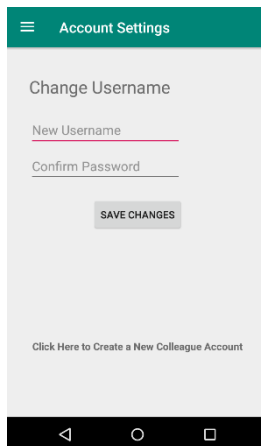


## Layouts and UI Design – Employee Application

## fragment\_login.xml

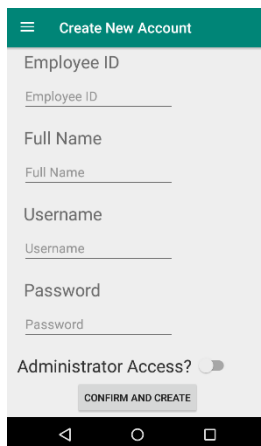


## fragment\_account.xml



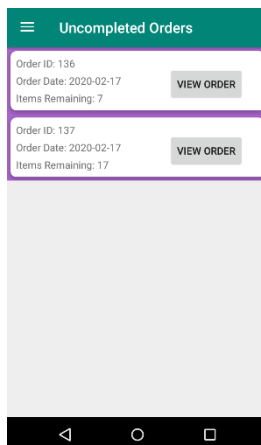
The screenshot shows the 'Account Settings' screen. At the top is a green header with a hamburger menu icon and the text 'Account Settings'. Below the header, the title 'Change Username' is displayed. There are two input fields: 'New Username' and 'Confirm Password'. A 'SAVE CHANGES' button is positioned below the input fields. At the bottom of the screen, there is a link that says 'Click Here to Create a New Colleague Account'. The Android navigation bar is visible at the very bottom.

## fragment\_new\_account.xml



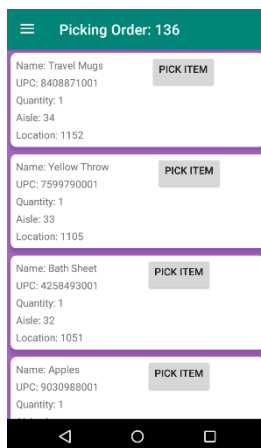
The screenshot shows the 'Create New Account' screen. It has a green header with a hamburger menu icon and the text 'Create New Account'. The form contains several fields: 'Employee ID', 'Full Name', 'Username', and 'Password'. There is also a toggle switch for 'Administrator Access?'. A 'CONFIRM AND CREATE' button is located at the bottom of the form. The Android navigation bar is visible at the bottom.

## fragment\_uncomp\_orders.xml

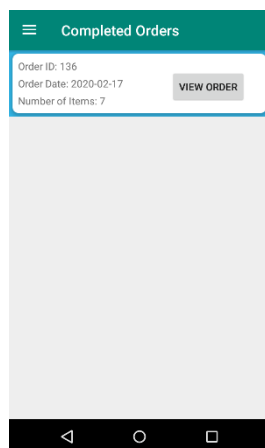
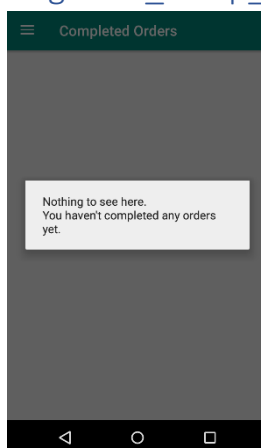


The screenshot shows the 'Uncompleted Orders' screen. It has a green header with a hamburger menu icon and the text 'Uncompleted Orders'. The screen displays a list of two orders, each enclosed in a purple border. The first order has an ID of 136, a date of 2020-02-17, and 7 items remaining. The second order has an ID of 137, a date of 2020-02-17, and 17 items remaining. Each order entry has a 'VIEW ORDER' button. The Android navigation bar is visible at the bottom.

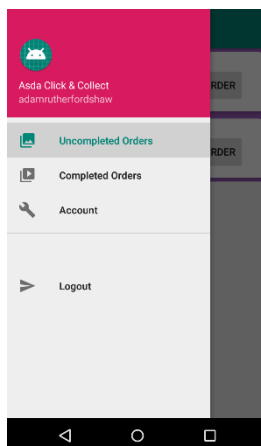
## fragment\_view\_uncomp\_orders.xml



## fragment\_comp\_orders.xml



## drawer\_menu.xml



## PHP and SQL Database Queries

### conn.php

```
<?php
$db_name = "asdac&c";
$mysql_username = "root";
$mysql_password = "@XHfELFEsGZG7c6";
$server_name = "localhost";
$conn = mysqli_connect($server_name, $mysql_username, $mysql_password, $db_name);
if (!$conn) {
    echo "Connection failed";
}
?>
```

### custLogin.php

```
<?php
require "conn.php";
$username = $_POST["username"];
$password = $_POST["password"];
$mysql_qry = "select userid from users_tbl where username like '$username' and
password like '$password' and customer like '1'";
$result = mysqli_query($conn, $mysql_qry);
if(mysqli_num_rows($result) == 1) {
    echo "Login successful" . "|";
    $row = ($result->fetch_assoc());
    $custid = $row["userid"];
    echo $custid . "|";
    $POSTorderid_qry = "select orders_tbl.orderid from orders_tbl left join
ordersitems_tbl on orders_tbl.orderid = ordersitems_tbl.orderid where customerid
like '$custid' and submitted like '0' and ordersitems_tbl.orderid is null;";
    $orderidresult = mysqli_query($conn, $POSTorderid_qry);
    if (mysqli_num_rows($orderidresult) > 0) {
        $row = $orderidresult->fetch_assoc();
        $orderid = $row["orderid"];
        $deleteorder_qry = "delete from orders_tbl where orderid like
'$orderid' and submitted like '0'";
        if($conn->query($deleteorder_qry) === TRUE) {
            echo "Completed";
        }
    }
} else {
    echo "Login not successful. Error: " . $mysql_qry . "<br>" . $conn->error;
}
$conn->close();
?>
```

### custRegister.php

```
<?php
require "conn.php";
$username = $_POST["username"];
$password = $_POST["password"];
$email = $_POST["email"];
$fullname = $_POST["fullname"];
$dateofbirth = $_POST["dateofbirth"];
$mysql_qry = "insert into users_tbl (fullname, emailaddress, dateofbirth, username,
password, customer) values ('$fullname', '$email', '$dateofbirth', '$username',
'$password', '1')";
if($conn->query($mysql_qry) === TRUE) {
    echo "Register successful";
}
else {
    echo "Register not successful: " . $mysql_qry . "<br>" . $conn->error;
}
$conn->close();
?>
```

## empLogin.php

```
<?php
require "conn.php";
$username = $_POST["username"];
$password = $_POST["password"];
$mysql_gry = "select administrator from users_tbl where username like '$username'
and password like '$password' and customer like '0'";
$result = mysqli_query($conn, $mysql_gry);
if(mysqli_num_rows($result) == 1) {
    $row = $result->fetch_assoc();
    echo "Login successful|" . $row["administrator"];
}
else {
    echo "Login not successful|Error: " . $mysql_gry . "<br>" . $conn->error;
}
$conn->close();
?>
```

## empRegister.php

```
<?php
require "conn.php";
$username = $_POST["username"];
$password = $_POST["password"];
$employeeid = $_POST["employeeid"];
$fullname = $_POST["fullname"];
$admin = $_POST["admin"];
$mysql_gry = "insert into users_tbl (username, password, employeeid, fullname,
administrator) values ('$username', '$password', '$employeeid', '$fullname',
'$admin')";
if($conn->query($mysql_gry) === TRUE) {
    echo "Register successful";
}
else {
    echo "Register not successful. Error: " . $mysql_gry . "<br>" . $conn->error;
}
$conn->close();
?>
```

## retrieveBasket.php

```
<?php
require "conn.php";
$orderid = $_POST["orderid"];
$mysql_gry = "select products_tbl.productname, ordersitems_tbl.upc, quantity from
products_tbl right join ordersitems_tbl on products_tbl.upc = ordersitems_tbl.upc
right join orders_tbl on ordersitems_tbl.orderid = orders_tbl.orderid where
orders_tbl.orderid = '$orderid' and submitted like '0'";
$result = mysqli_query($conn, $mysql_gry);
if ($conn->query($mysql_gry) == TRUE) {
    while($row = $result->fetch_assoc()) {
        if (($row["upc"] != "") and ($row["productname"] != "") and
($row["quantity"] != "")) {
            echo $row["upc"] . "|" . $row["productname"] . "|" .
$row["quantity"] . "\n";
        }
    }
} else {
    echo "An error occurred. Please try again. Error: " . $mysql_gry . "<br>" .
$conn->error;
}
$conn->close();
?>
```

## addToBasket.php

```
<?php
require "conn.php";
$orderid = $_POST["orderid"];
$upc = $_POST["productid"];
$sql_gry = "insert into ordersitems_tbl (orderid, upc, quantity) values
('$orderid', '$upc', '1')";
if($conn->query($sql_gry) === FALSE) {
    echo "An error occurred. Please try again. Error: " . $sql_gry . "<br>" .
    $conn->error;
}
$conn->close();
?>
```

## deleteFromBasket.php

```
<?php
require "conn.php";
$orderid = $_POST["orderid"];
$upc = $_POST["productid"];
$sql_gry = "delete from ordersitems_tbl where orderid like '$orderid' and upc
like '$upc'";
if($conn->query($sql_gry) === FALSE) {
    echo "An error occurred. Please try again. Error: " . $sql_gry . "<br>" .
    $conn->error;
}
?>
```

## addOrder.php

```
<?php
require "conn.php";
$orderid = $_POST["orderid"];
$date = date('Y-m-d');
$sql_gry = "update orders_tbl set submitted = '1', orderdate = '$date' where
orderid like '$orderid'";
if($conn->query($sql_gry) === FALSE) {
    echo "An error occurred. Please try again. Error: " . $sql_gry . "<br>" .
    $conn->error;
}
$conn->close();
?>
```

## checkOrder.php

```
<?php
require "conn.php";
$customerid = $_POST["userid"];
$sql_gry = "select orderid from orders_tbl where submitted like '0' and
customerid like '$customerid'";
$result = mysqli_query($conn, $sql_gry);
if(mysqli_num_rows($result) == 1) {
    $row = ($result->fetch_assoc());
    echo $row["orderid"];
} else if(mysqli_num_rows($result) == 0) {
    echo "No orders found";
}
$conn->close();
?>
```

## createOrder.php

```
<?php
require "conn.php";
$customerid = $_POST["userid"];
$mysql_qry = "insert into orders_tbl (customerid) values ('$customerid')";
if($conn->query($mysql_qry) == TRUE) {
    $mysql_qry = "select orderid from orders_tbl where customerid like
'$customerid' and submitted like '0'";
    $result = mysqli_query($conn, $mysql_qry);
    if (mysqli_num_rows($result) == 1) {
        $row = $result->fetch_assoc();
        echo $row["orderid"];
    } else {
        echo "An error occurred. Please try again. Error: " . $mysql_qry .
"<br>" . $conn->error;
    }
} else {
    echo "An error occurred. Please try again. Error: " . $mysql_qry . "<br>" .
$conn->error;
}
$conn->close();
?>
```

## pickItem.php

```
<?php
require "conn.php";
$orderid = $_POST["orderid"];
$upc = $_POST["upc"];
$mysql_qry = "update ordersitems_tbl set picked = '1' where orderid like '$orderid'
and upc like '$upc'";
if($conn->query($mysql_qry) === FALSE) {
    echo "An error occurred. Please try again. Error: " . $mysql_qry . "<br>" .
$conn->error;
}
?>
```

## retrieveUncompOrders.php

```
<?php
require "conn.php";
$mysql_qry = "select orderid, orderdate from orders_tbl where completed like '0'
and submitted like '1' order by orderdate";
$result = mysqli_query($conn, $mysql_qry);
if ($conn->query($mysql_qry) == TRUE) {
    while($row = $result->fetch_assoc()) {
        $orderid = $row["orderid"];
        $orderdate = $row["orderdate"];
        $count_qry = "select count(*) as count from ordersitems_tbl where
orderid like '$orderid' and picked like '0'";
        $countresult = mysqli_query($conn, $count_qry);
        if ($conn->query($count_qry) == TRUE) {
            $countrow = $countresult->fetch_assoc();
            $count = $countrow["count"];
            echo $orderid . "|" . $orderdate . "|" . $count . "\n";
        } else {
            echo "An error occurred. Please try again. Error: " .
$mysql_qry . "<br>" . $conn->error;
        }
    }
} else {
    echo "An error occurred. Please try again. Error: " . $mysql_qry . "<br>" .
$conn->error;
}
?>
```



## updateOrder.php

```
<?php
require "conn.php";
$orderid = $_POST["orderid"];
$username = $_POST["username"];
$mysql_gry = "select employeeid from users_tbl where username like '$username'";
$result = mysqli_query($conn, $mysql_gry);
$row = mysqli_fetch_array($result);
$colleagueid = $row['employeeid'];
$mysql_gry = "update orders_tbl set completed = '1', colleagueid = '$colleagueid'
where orderid like '$orderid'";
if($conn->query($mysql_gry) === FALSE) {
    echo "An error occurred. Please try again. Error: " . $mysql_gry . "<br>" .
$conn->error;
}
$conn->close();
?>
```

## retrieveOrderProducts.php

```
<?php
require "conn.php";
$orderid = $_POST["orderid"];
$mysql_gry = "select productname, products_tbl.upc, quantity, aisle, location from
products_tbl right join ordersitems_tbl on products_tbl.upc = ordersitems_tbl.upc
where orderid like '$orderid' and picked like '0'";
$result = mysqli_query($conn, $mysql_gry);
if ($conn->query($mysql_gry) == TRUE) {
    while($row = $result->fetch_assoc()) {
        $productname = $row["productname"];
        $upc = $row["upc"];
        $quantity = $row["quantity"];
        $aisle = $row["aisle"];
        $location = $row["location"];
        echo $productname . "|" . $upc . "|" . $quantity . "|" . $aisle . "|"
. $location . "<br>";
    }
} else {
    echo "An error occurred. Please try again. Error: " . $mysql_gry . "<br>" .
$conn->error;
}
$conn->close();
?>
```

## updateName.php

```
<?php
require "conn.php";
$oldusername = $_POST["username"];
$newusername = $_POST["newusername"];
$password = $_POST["password"];
$mysql_gry = "update users_tbl set username = '$newusername' where username like
'$oldusername' and password like '$password'";
if($conn->query($mysql_gry) === FALSE) {
    echo "An error occurred. Please try again. Error: " . $mysql_gry . "<br>" .
$conn->error;
}
else {
    echo "Updated name successfully";
}
$conn->close();
```

## retrieveCompOrders.php

```
<?php
require "conn.php";
$mysql_gry = "select orderid, orderdate from orders_tbl where completed like '1'
and submitted like '1' order by orderdate;";
$result = mysqli_query($conn, $mysql_gry);
if ($conn->query($mysql_gry) == TRUE) {
    while($row = $result->fetch_assoc()) {
        $orderid = $row["orderid"];
        $orderdate = $row["orderdate"];
        $count_gry = "select count(*) as count from ordersitems_tbl where
orderid like '$orderid' and picked like '1';";
        $countresult = mysqli_query($conn, $count_gry);
        if ($conn->query($count_gry) == TRUE) {
            $countrow = $countresult->fetch_assoc();
            $count = $countrow["count"];
            echo $orderid . "|" . $orderdate . "|" . $count . "\n";
        } else {
            echo "An error occurred. Please try again. Error: " .
$mysql_gry . "<br>" . $conn->error;
        }
    }
} else {
    echo "An error occurred. Please try again. Error: " . $mysql_gry . "<br>" .
$conn->error;
}
?>
```

## retrieveProducts.php

```
<?php
require "conn.php";
$mysql_gry = "select upc, productname, aisle from products_tbl order by
productname";
$result = mysqli_query($conn, $mysql_gry);
if ($conn->query($mysql_gry) == TRUE) {
    while($row = $result->fetch_assoc()) {
        echo $row["upc"] . "|" . $row["productname"] . "|" . $row["aisle"] .
"\n";
    }
} else {
    echo "An error occurred. Please try again. Error: " . $mysql_gry . "<br>" .
$conn->error;
}
$conn->close();
?>
```

## Testing Customer Application

[illegible]