|  |  |
| --- | --- |
|  | Object-Oriented Implementation of a Smart Vending Machine Complete with GUI |
|  |  |
|  | Jonathan Phipps  Student Number: 14729831 |

Table of Contents

[Statement of Originality 2](#_Toc4705500)

[Abstract 3](#_Toc4705501)

[Introduction 4](#_Toc4705502)

[Background Research 5](#_Toc4705503)

[Smart Vending Machines 5](#_Toc4705504)

[Recommendation Algorithms 7](#_Toc4705505)

[Collecting Customer Data 10](#_Toc4705506)

[Design 11](#_Toc4705507)

[GUI Design 14](#_Toc4705508)

[Implementation 22](#_Toc4705509)

[First Prototype 22](#_Toc4705510)

[Second Prototype 30](#_Toc4705511)

[Third Prototype 33](#_Toc4705512)

[Fourth Prototype 36](#_Toc4705513)

[Android Application 40](#_Toc4705514)

[Testing & Evaluation 49](#_Toc4705515)

[Conclusion & Future Work 64](#_Toc4705516)

[References 65](#_Toc4705517)

[Appendix 66](#_Toc4705518)

[Appendix 1 - Ethics Checklist 66](#_Toc4705519)

[Appendix 2 - Prototype 1 State Diagram 68](#_Toc4705520)

[Appendix 3 - Prototype 1 UML Diagram 69](#_Toc4705521)

[Appendix 4 - Prototype 2 State Diagram 70](#_Toc4705522)

[Appendix 5 - Prototype 2 UML Diagram 71](#_Toc4705523)

[Appendix 6 - Prototype 3 State Diagram 72](#_Toc4705524)

[Appendix 7 - Prototype 3 UML Diagram 73](#_Toc4705525)

[Appendix 8 - Prototype 3 Database Diagram 74](#_Toc4705526)

[Appendix 9 - Prototype 4 UML Diagram 75](#_Toc4705527)

[Appendix 10 - Prototype 4 Database Diagram 76](#_Toc4705528)

[Appendix 11 - Android Application State Diagram 77](#_Toc4705529)

[Appendix 12 - Android Application Main Classes UML Diagram 78](#_Toc4705530)

[Appendix 13 - Android Application GUI Classes UML Diagram 79](#_Toc4705531)

[Appendix 14 - Android Application Database Diagram 80](#_Toc4705532)

[Appendix 15 - Gantt Chart 81](#_Toc4705533)

# Statement of Originality

CS3D661 Individual Project

This is to certify that, except where specific reference is made, the work described within this project is the result of the investigation carried out by myself, and that neither this project, nor any part of it, has been submitted in candidature for any other award other than this being presently studied.

Any material taken from published texts or computerized sources have been fully referenced, and I fully realize the consequences of plagiarizing any of these sources.

Student Name (Printed): Jonathan Phipps

Student Signature: 

Registered Course of Study: MComp Computer Science

Date of Signing: 19/03/2019

# Abstract

This project shows that object-oriented programming techniques and design patterns can be used to develop a working smart vending machine with an interactive digital interface. The smart vending machine program is capable of dispensing and providing customised items that meet the customer’s specification as well as provide recommendations on products that they may like based on similar items that other customers have bought. In addition to the item dispensing side of the project, the software also provides the means to configure the machine and track statistics of purchases. The software will be programmed using different design patterns and will make uses of the Android mobile platform as the basis of the user interface. The project was completed with an android application that met all the objectives laid out in this report, including incorporating a recommendation system and control panel into the application with different levels of user permissions. This demonstrates that vending machines could possibly start to include more modern technology in their design to not only provide a more personalised experience for the customer but to also make the machine more manageable for the suppliers especially if multiple machines are in operation at the same time.

# Introduction

The aim of this project is to create a new type of smart vending machine using object-oriented programming complete with a graphical user interface that the customers can use to interact with the machine. This vending machine will be different from the existing machines as it will have smart features such as the ability to collect and record data for the supplier to use in order to more efficiently and effectively manage the machine. The vending machine will also have a recommendation engine so that the data it has collected from customers can be used to find and suggest products that the customer may like based on their previous orders and what other customers have ordered. The objectives for this project are to research and evaluate existing products, implement a recommendation system that suggests products based on previous sales, implement features to track machine inventory and produce sales data, and research data protection due to the potential risks of collecting user information.

Existing research and products featured on the internet and in academic journals will form the basis of background research for this project as they hold a great deal of information about existing smart vending machines and recommendation systems that can be used when designing the solution to this problem. The research would also give an understanding of how a recommendation system works and the methods or algorithms used in such systems.

The final solution will feature a recommendation engine so that the system will be able to pick out products for the customer depending on their preferences. In order to fulfil this objective, there would need to be a database in the background of the application that would store the purchase history for each customer.

The final solution should also include features for the suppliers to track the inventory in the machine and produce sales data. These features would show statistics such as how many products have sold, which items need to be restocked and which items should probably be replaced with different items. The suppliers should also have the ability to edit stock through the interface so that they can add new products or edit existing ones.

If this project was a commercial operation then it would have to abide by data protection due to the collection and storage of the user’s personal details such as their name, password and purchase history. This report will also have a small section of research on data protection to show an outline of what would have to be done for the final application to be used commercially.

# Background Research

## Smart Vending Machines

Vending machines are usually placed in convenient locations and allow people to purchase cold snacks and drinks. However, more modern vending machines can also dispense other items. For example, a University in the Netherlands has developed a machine that can cook and dispense french fries with a variety of sauces (Fuhrmeister, 2015). This kind of approach to vending machines can make obtain even more items more convenient and could lead to vending machines for all kinds of items being used daily.

Intel has a solution that uses cloud-based analytics that help to improve customer experiences and management of the vending machine (Intel, 2016). The machine does this by determining which of the products in the machine are the most popular and ensuring that these products are always stocked. This would lead to a better customer experiences as not only would the customer’s favourite products always be in stock, but the machine suppliers could swap out less frequently purchased products for brand new products for customers to try. Although this is an upgrade for existing machines, it does add more functionality for older systems and turns the machine into a smart vending machine. With this upgrade, the machine would be able to accept other payment methods like contactless cards, payment via NFC and mobile payments which would be more convenient for customers who don’t always carry cash.

A smart coffee vending machine that would allow its users to interact with the machine via a smartphone application was proposed by Kim et al. (2014). The smartphone application allows the customer to place and customize their orders as well as see the status of the machine such as the date that the machine was last cleaned and the date that all the ingredients in the machine were replaced last. The application would be openly available from the android market and communicates with the machine via Bluetooth. The application has a graphical user interface that the customer can use to customise their orders such as adding and changing the amount of milk and sugar in their coffee. The data from the order is also used to estimate the calories that the user consumed in that purchase. This approach to the problem allows the customer to easily customize their order and provides an easy to use interface to place an order. This method could be combined with other systems such as mobile payments to provide a truly convenient and mobile experience. The collection of orders and calorie data could be useful where people must monitor what they consume i.e. limit the amount of sugar or caffeine that they consume. This type of vending machine which offers customisable products that the customer can personalise to their own preferences means that there is more chance of the machine have a product for every customer to enjoy.

The smart vending machine mentioned in the article by Shoot (2014, p. 49) also uses cloud computing to allow more efficient and convenient management of the machine to be carried out remotely. The suppliers would also get notified when items go out of stock and when the internal temperature of the machine changes. This is a similar solution to the one developed by Intel where the machine uses cloud computing to provide an efficient way to manage the vending machine however, the machine mentioned in the article by Shoot is a concept for a brand-new machine whereas the solution offered by Intel is an upgrade to existing machines.

One of the manufacturers of smart vending machines, Silkron, produce systems that have touch screen panels to provide an easy to use interface for the customers to use as well as incorporating multiple payment methods including cash, credit cards and contactless payments (http://www.silkron.com/, no date). The vending machines produced also make recommendations to the customers based on what they have previous purchased. The ability to recommend products could help customers choose which product to purchase based on records of previously ordered products and what other people have ordered as it suggests a product that the customer is likely to enjoy. Another feature of the machines is the ability to produce sales data to show what products are selling well and what products could possibly be replaced. Sales data from the machine could help identify which products sell well in a specific location and replace the products that aren’t purchased that often with new products that the local population could enjoy. By using data collection to aid the decision of which stock to include in the machine could maximise sales of the products in the machine as the suppliers could change the stock based on what sells well and fill the machine will similar items.

All the solutions mentioned so far have focused on producing a more convenient machine for customers to use with the addition of more efficient management of the machine for the suppliers. A machine that can collect data to use in its daily operations, such as to suggest similar products for the customers to purchase based on previous orders as well as using the data to show what products are selling well and notifying the suppliers when an item goes out of stock provides convenient and efficient operations for both customer and supplier. The idea of a mobile application to interact with the machine should also make it much more convenient for users to order and pay for their item. The solutions by Intel, Silkron and the machine proposed by Shoot (2014) all use cloud-based systems to collect and store data but only the solution by Kim (2014) makes use of a mobile application to interact with the machine.

Although the various solutions offered may update vending machine technology to a modern-day standard, there is still the issue of technical problems and customers not understanding how to use the new technology. Technical problems could arise from any number of situations whether it’s a problem with the touchscreen on the device which would lead to people being unable to use the machine, or if the machine loses connection to the network so it could not be managed remotely. These technical problems would mean that even though the machine is supposed to be more efficient to manage, the suppliers would still have to personnel to service and repair the machines when something goes wrong as well as to restock the machine when the inventory in the machine runs low.

The new technology should come with plenty of documentation and instructions so that both the supplies and the customers know how to operate the new machines. Whether this comes in the form of onscreen assistance or some form of printed material, it allows consumers to learn how to operate the new technology so that it becomes a part of everyday life and not something that is too hard for someone to operate.

## Recommendation Algorithms

With more products and content being available online, Websites have started to make use of recommendation algorithms to filter which of their items they think would be the most suitable for the customer. Websites such as Amazon, Netflix and YouTube are just a few examples of companies who collect data such as what users have watched or ordered and what the user has rated an item. The rating of an item could be a product review, a rating or just a purchase of an item. These recommendation systems then use this data to provide a personalised list of items that the user would probably enjoy (Isinkaye et al, 2015).

One example of a recommendation system is the one used by YouTube, a system that is used by the website to filter videos depending on the user’s liked videos and watch history (Covington et al, 2016). Covington et al goes on state that the recommendation system used on YouTube can balance the new daily content uploaded with the older content as this is used to give the user the newest content that they might enjoy according to their watch history, but it is not limited to the newer content as it will recommend older content too. This is a good approach for a recommendation system because without this method, newer items with no rating may end up not being recommended to the consumer and the older content will always remain on top of the rated items with the newer content being ignored.

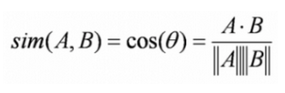
Another example of a recommendation system is one used by Amazon, this system is discussed by Schafer et al (1999) and is specifically about the recommendation system in the book section of the website. The system has 2 features of interest for this project, ‘The customers who bought this like’ feature and the Book matcher feature. ‘The customers who bought this like’ feature displays recommendations based on other books purchased by other customers who also purchased the current book that the consumer is looking at, it also recommends other authors that have written books that other customers have ordered who also purchased the current book (Schafer et al, 1999). This feature has been highlighted for this project because it can be used to display products that other customers have bought who also purchase the items that the customer regularly orders. The 2nd feature of interest, the book matcher, lets the customer give a rating for the books that they have read on a 5-point rating system, these ratings are then used to gather recommendations of books that the consumer has not yet read based on their preferences given in the feedback process (Schafer et al, 1999). This feature would also be useful for this project as it would allow users to rate the orders that they have previously ordered and let the machine suggest something new.

There are 2 types of recommendation systems featured in this background research, these are Content-based filtering and Collaborative-based filtering (Isinkaye, 2015; Sharma, 2018). Content-based filtering focuses more on the products previous bought by a customer and finds similar products to recommend whereas Collaborative-based filtering looks at what other customers have purchased and tries to find similarities between the current customer and other customers based on their order history and uses this information to find products.

Content-based filtering finds products based on what the customer has ordered before. The algorithm does this most commonly by cosine similarity (Sharma, 2018). With the cosine similarity, all the ratings for items are expressed as vectors and the similarities between the items are calculated by finding the cosine of the angle between the 2 vectors (Zhang, 2009). These values are then placed into descending order and the Top-n items are recommended to this user where the Top-n items is the N number of items at the top of the list. This approach all depends on the ratings left by the user and can only calculate the similarities of the same type of products, so it cannot recommend any product of a product type that the customer hasn’t purchased or rated (Sharma, 2018). Another problem with this approach is that if a new item is added with no ratings, then that new item would not get recommended by this type of algorithm.

Collaborative-based filtering finds products based on what other customers have purchased. This type of filtering can either be focused on the items or the users (Sharma, 2018; Schafer *et al*, 1999). If this approach is focused on the items, then the algorithm would find other products based on what other customers have ordered when there is at least one common item in each order. However, if the approach is focused on users, then the similarities between users is calculated and the customer is recommended products that other customer that has the most in common has bought. User-based filtering should only be used when there is a small number of users because the computations would take too long if there is loads of users (Sharma, 2018).

According to Sharma (2018), The main 2 formulas for calculating the similarity between items / users are the cosine similarity and Euclidean distance. Each of these formulas work for content-based filtering and collaborative-based filtering.



The cosine similarity models the data as mathematical vectors and the cosine of the angle between the 2 vectors is the similarity score. The score can range from -1 to 1 with 1 being the most recommended item and -1 being the least recommended item.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Item A | Item B | Item C |
| User 1 | Purchased | Purchased | Purchased |
| User 2 | Purchased | Purchased |  |
| User 3 | Purchased | Purchased |  |
| User 4 | Purchased |  | Purchased |

If the value assigned to purchased is 1 and the value assigned for no purchase is 0 and the user selects Item C, the similarity scores for Item C to Item A and Item C to Item B is calculated.

= 0.707107 (6dp)

= 0.408248 (6dp)

The item with the highest similarity score is Item A because there are more users who bought Item C also bought Item A than those who bought Item B. These were calculated by representing the Item columns as vectors i.e. Item A vector would be (1,1,1,1), Item B vector would be (1,1,1,0) and Item C vector would be (1,0,0,1). When comparing one item to another, the dot product of the 2 chosen item vectors are calculated and divided by the product of the 2 vector lengths.



Euclidean distance is another formula that can be used to calculate the similarity score between 2 items or users. The shortest distance between any 2 items would be the most recommended item. Using the same data from the previous examples, the Euclidean distance can be calculated when the user selected Item C.

This example shows that Item A would still be recommended when using Euclidean distance because the distance between Item C and Item A is lower than the distance between Item C and Item B. These scores are calculated by using the item columns as co-ordinates in N-dimensional space with N being the number of items i.e. Item A would have the coordinates (1,1,1,1) etc.

Both content-based and collaborative-based filtering have their advantages and disadvantages but for this project, combining the 2 approaches and taking the average similarity scores would work better because the products with no ratings would still have a possibility of being recommended when other users purchase those items. The collaborative-based filtering should be focused on items instead of users as a vending machine could have numerous users depending on its location.

## Collecting Customer Data

The recommendation system and the vending machine in general would have to collect user information for use in daily operations. Because of this collection of data, Data protection measures must be considered. The UK Law states that any data collected must be done with the full consent of the individual and it must be stored in a secure manner (Great Britain. Data Protection Act 2018). This means that to collect any data from the customer, they must first consent to the collection and storage of their data. This could be done with a notice when the user registers to use the vending machine which could say that their information will be stored in order to use it for the purpose of recommending items to them and using other data such as their name solely for the purpose of their profile on the system.

All the data collected on the machine must be kept private where not even the supplier should have easy access to the user’s data. The suppliers should only have access to the number of items that they have sold and possibly the basic demographics of the customers such as their age and gender so that they can establish who generally buys each product.

The solution to this project should make sure that it uses secure programming techniques to make sure that the customers data is secure. These techniques could be data encryption, authorisation and authentication, and general secureness of the software so making sure basic security issues such as buffer overflows and unsecure operations cannot occur.

Failure to abide to data protection laws could lead to the unlawful collection and storage of user’s data. A recent example of a data protection breach would be where Microsoft collected user’s data without their consents through the means of their Office suite of software (Field, 2018). This could lead to severe consequences for Microsoft such as being fined for collecting data without the user’s consents and for unnecessary collecting large amounts of data.

In conclusion, the smart vending machine should only collect the data that it needs from the customers with no unnecessary data being stored. This ensures that it has enough data for its daily operations. The user should also give their consent for their data being used and stored in a way that is fully explained to them in the registration process.

# Design

The design of the proposed solution is to be easily accessible to anyone who wants to use the vending machine, it should be easy to maintain and restock and it should also have features that help both the customer and the supplier. For these reasons, the solution discussed in this report is going to be designed as an android application that would be available on a publicly accessible resource such as a company website or app store if this was a commercial project. The reason that the android platform was chosen is because android is the most popular mobile operating system with three quarters of mobile phone running a version of android (StatCounter, 2018). The Smart Vending Machine Application will be targeted at Android version 4.0.3 or API 15, otherwise known as Ice Cream Sandwich. By targeting an older version of android ensures close to 100% compatibility with all android devices running today. Another advantage of using the android platform is that it fully supports the Java programming language. Using Java for this project means that small prototypes can be easily be built in Windows before moving on to the more time-consuming development of the android application, this makes sure that the logic of the program is stable before moving on to the android development. All these prototypes will then form the underlying engine that will run the smart vending machine application. Having an underlying Java program underneath the mobile interface makes it easier for future development on other platforms as the underlying code will still run independently with little change.

The smart vending machine itself will be designed to work with customisable products which it is up to the supplier what kind of food or drink that they want to serve. For example, a supplier could choose to have a create your own pizza vending machine set up in a busy shopping centre or a just a simple coffee machine in an office complex that offers a wide variety of coffee and condiments. The application is programmed to be dynamically programmed with inventory items and details of those items such as cost and stock. The customer would then only need to get into proximity to a machine, log into the application on an android device, create their order, pay and their customized order would be dispensed after being prepared specifically to their order. The vending machine being dynamically programmed means that the software isn’t hard coded to any one product or type of product so the supplier could put any product in the machine as long as the hardware supported the product.

The application will be programmed with 2 levels of users, these would be administrators and users. The table below shows the privileges each level would have:

|  |  |  |
| --- | --- | --- |
| Function | Users | Administrators |
| Access the dispenser to purchase a fully customisable product | Yes | Yes |
| Edit their own profile to change their own password | Yes | Yes |
| Add, Edit or Remove current inventory in the machine | No | Yes |
| Edit other user’s roles or remove other users from the system | No | Yes |

The User role is the level of permissions that most customers would have just to access the vending machine and to manage their own account. The Admin role is designed for suppliers to program the machine with details about inventory, setting up other administrators so that employees can carry out maintenance on the machine and restock when needed. The reason that there are 2 user roles in the software is to make sure that the person that logs on to the machine only has access to the part of the machine that they are authorised to access. This is important as the suppliers wouldn’t want customers to be able to do tasks that only administrators can do such as alter the prices of products or change the roles of other users as this would be disruptive to the supplier and their business.

The current order of operations of the machine are as follows:

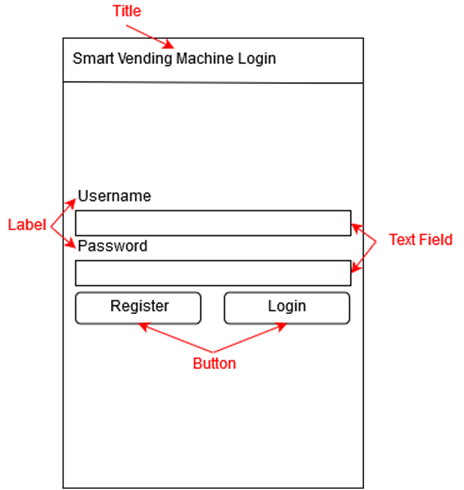
1. The supplier sets up the machine with their products, Enter all the required details about the inventory. If this step is not carried out, then the dispense will have no products to dispense. The dispenser will now have stock and is ready to take orders.
2. Users can now register or log in to the machine, to either order a product or change their own password.
3. If a user chooses to order a product, they first choose which base product they want. The base products are set by the supplier and depends on what type of food or drink the machine is serving e.g. A base product could be a pizza base in a create your own pizza machine, or the type of bread in a sandwich dispenser or even just the type of coffee.
4. Once a base product has been chosen, the user can then pick as many topping products as they want. A Topping product could be condiments, ingredients or extras to be added to the base product.
5. If there are recommendations available, they will be displayed after the user has selected toppings. The user will then be able to add recommended toppings to their product.
6. The user is then asked to confirm if they want to continue with the purchase of their product. If the user chooses to restart at this point, they will have to reselect their product again
7. Once the product has been made, the machine can then move to one of 2 states, either the user has enough credit on their account and the item cost is then deducted from their credit and the item is dispensed, or the user must insert credit first before their item is dispensed. Any remaining credit from any order is stored on the customer’s account and can be used to pay part or all the cost of an item.
8. The user is then taken back to the main menu where they can either choose to purchase another product, change their password or log out.
9. Admins can log in and either access the dispenser in the same way as the customers, or they can access their admin control panel which has all the functions needed to manage the inventory in the machine and manage the users accounts registered on a machine.

The design is currently only focused on a single vending machine being in operation at any one time with one user accessing the machine at a time. For this reason, all the data will be stored locally making use of the SQLite Database functionality that is built into android. This provides a persistent database where all the inventory and user profiles are stored. If this was a commercial project with multiple of vending machines in operation, then a MySQL or similar database would be more suitable on a server so that all the user accounts can be stored remotely, and users can access their profile on any machine in operation. However, the inventory of the machine can remain in local database on each machine unless the suppliers want to edit inventory remotely. An SQLite database is a small file that is stored directly on to the phone’s storage which is perfect for demonstration purposes of this project as long as the number of products or number of users do not become so large.

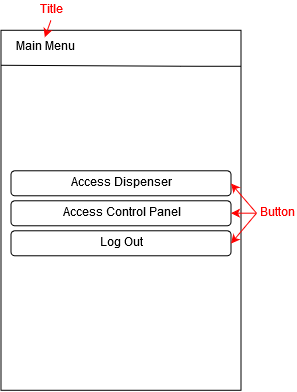
The management of the smart vending machine can all be done using the admin control panel however there must always be one admin registered with the machine at any time as without the admin, it is not possible to restock the machine when the stock runs out. To always make sure there is at least one admin, an admin cannot delete themselves or change their own role. If an admin wants their role changed or have their account deleted, it must be done by someone else with an admin role. It is important to maintain one admin in the system because if that admin was deleted then the machine could not be maintained by the suppliers without resetting the entire machine.

## GUI Design

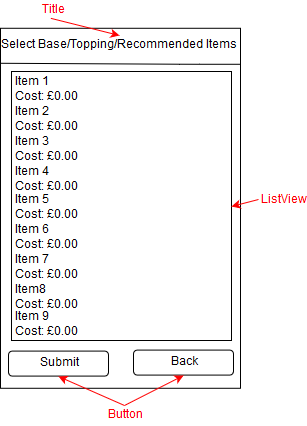
This section will discuss the designs for each screen of the GUI and how the application will look on an android device.



The screen shown above is the login screen for the application. The user can use this screen to login to an existing account on the machine or register a new account to start using the machine. All the text fields on this screen are required and the username field is validated to make sure that when a new user registers, their username is unique and doesn’t not exist in the database already. The login screen is the first screen that appears when the user opens the application and there is no way to get through the login screen without either a valid login or registering as a new user. The first user to register through the application is assigned an admin role, this is so when the supplier is setting up the machine, their account will automatically be given the admin role.



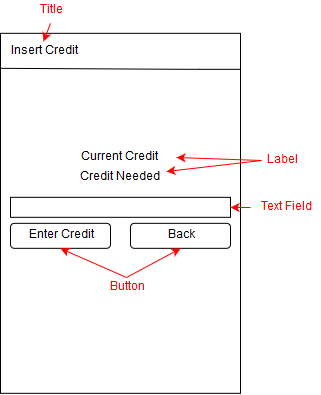
After a successful login, the user is then taken to a main menu screen where they can select what they want to do next in the app. The user can either purchase a product by choosing to access the dispenser, they can access the control panel which will allow them to change their account settings or machine settings depending on their user role, or the user can simply log out.



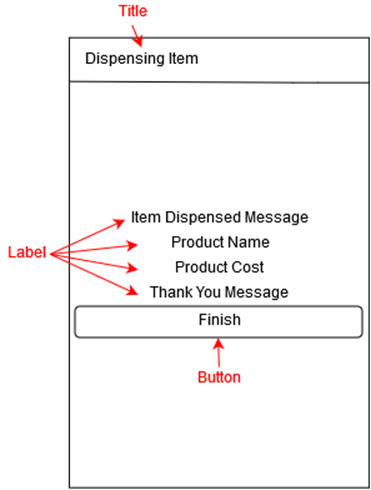
When the user chooses to purchase a product, they will start to configure the item they want to order by selecting the base product and then any toppings that they want to add. The user must first select a base product in order to continue. If the user tries to continue without choosing a base product or tries to select multiple bases, an error message will be displayed saying that one base must be selected. After the user has selected their base, the app then moves to a different screen for the user to select their toppings. The user can either select as many toppings as they want, or they can continue without choosing any toppings. If the user has selected toppings and the application can find some recommendations based on the toppings they have selected and the toppings other users previously purchased, then a third selection screen will appear allowing the user to choose any of the recommended products if they wanted to. If there are no recommendations available, the app will skip the third selection select and go straight to the product summary screen.



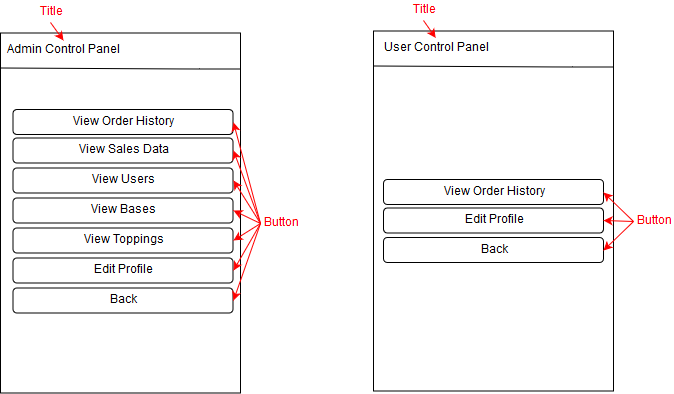
Once the user has configured the item that they want to purchase, a product summary screen will appear to confirm the name and cost of the item for the user to confirm before continuing. If the user is happy with the product, they can confirm the product is correct and continue with their purchase. Otherwise, the user can choose to restart and start configuring their product from scratch.



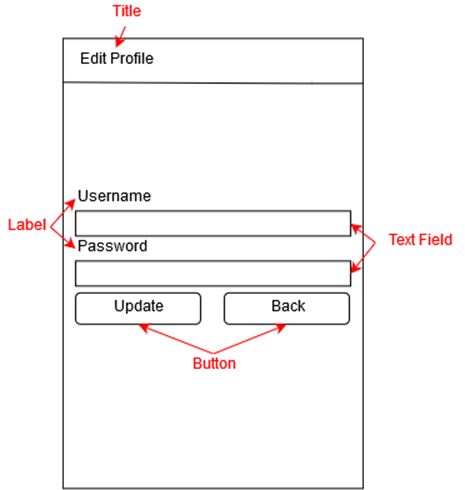
After the user has confirm their product, they are prompted to insert credit into the machine if there is not enough credit currently on their account. Any excess credit that was entered into the machine will remain on their account for the next time they purchase a product. If there is enough credit on the user’s account to cover the cost of the product, then the insert credit screen is skipped, and the app goes straight to dispense the item.



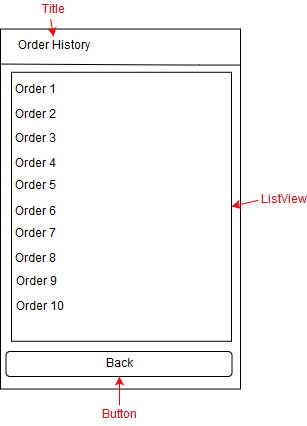
Once the user has enough credit in their account, a dispensing item screen is displayed which informs the user that their item has been dispensed. This screen will also display the product cost and the product name. Pressing the finish button will take the user back to the main menu.



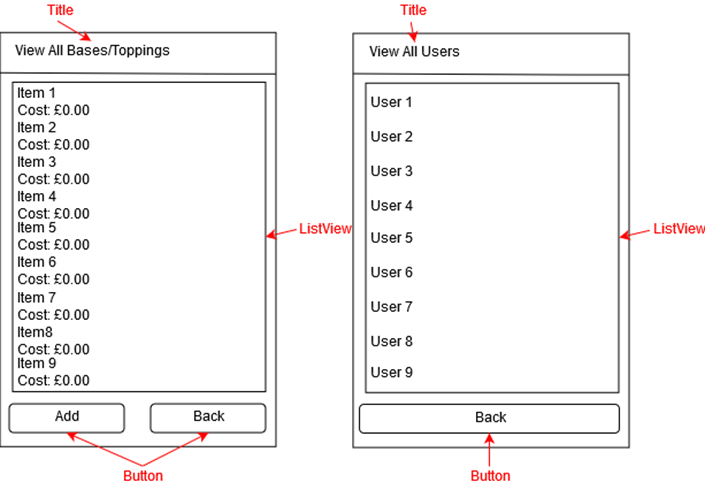
If the user chooses to go to the control panel, either an admin control panel or a user control panel will be displayed depending on the user’s role. This is done so that the user can only access settings that they have permission to change. The user control panel gives the option to edit the user’s profile which lets the user change their password. The user control panel also gives the option to view the user’s order history. The admin control panel can perform all the tasks that the user control panel can in addition to having access to the sales data that the machine has collected and access to the inventory and user management sections.



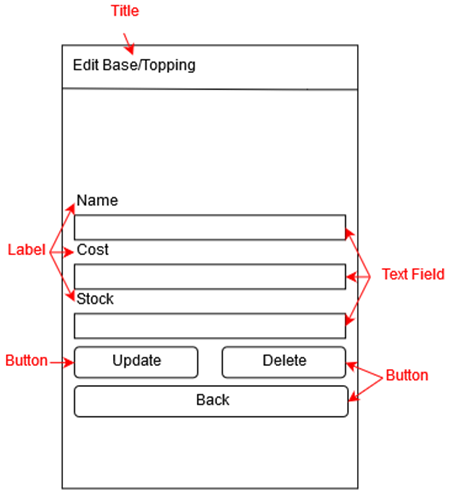
If the user chooses to edit their profile, a screen will appear that lets them change their own password. The user will not be allowed to edit their own username or delete their account. If the password has been changes, pressing update will save the change to the database. Otherwise leaving this screen will leave the profile unedited.



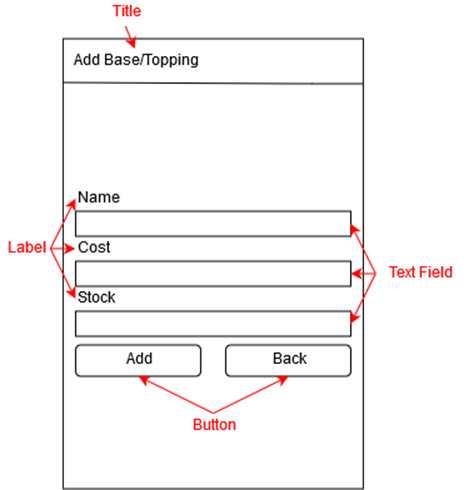
The order history option in the control panel will display the current users order history along with the name of the product they ordered, the cost of that product and the date that the product was purchased on.



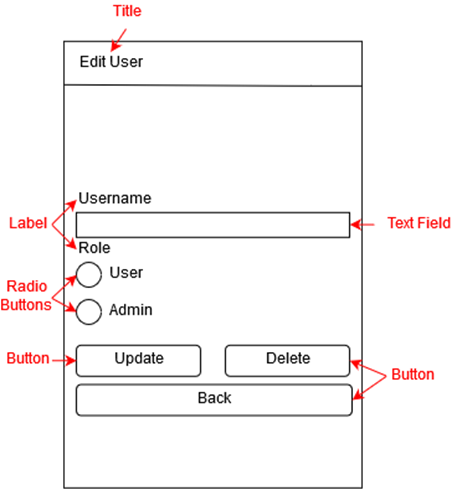
The options in the admin control panel to view the inventory or to view users will take the users to one of these screens. These screens will display all the entries in the relevant table i.e. if the user clicked view bases, it will display all bases stored in the database. Clicking on an entry in the list view will let the user edit the entry. The only difference between the view bases/toppings screen and the view user screen is that the add button has been removed from view users because users register themselves at the login screen and do not need admins to register them. Clicking the add button on the view bases/toppings screen will take the user to a screen that lets them a new inventory item.



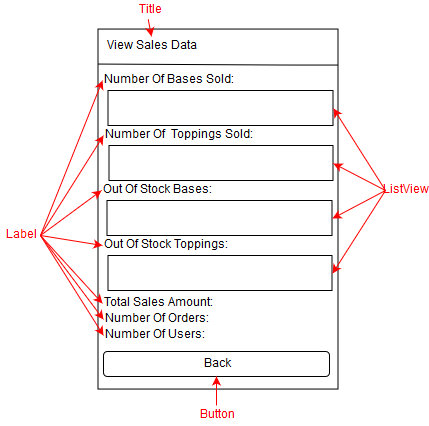
When the user selects a base or a topping from the View All Bases/Topping screen, it will take them to this screen where all the fields will be prepopulated with the details of the selected item. The user will then be allowed to change the cost and stock level of the selected item, but they will be unable to edit the name. Any values that have been changed will be validated before they can be changed in the database. The user can also delete the item using the delete button.



When the user chooses to add a base or topping, this screen will appear with fields for them to fill in asking for a product name, cost and stock level. All the fields are required for a product to be added and the product name must be unique in its category i.e. 2 bases cannot have the same name. All the text inputs are also validated to ensure the correct values are entered.



The edit user screen appears when the user selects a list entry in the View All Users screen. The selected user’s name and role will be prepopulated, but the username cannot be edited. The user’s role can be switched using the radio buttons or the selected user can be deleted however if a user selects themselves from the list, they will not be able to edit their own role or delete their own account, this is to make sure that there is always an admin present in the system.



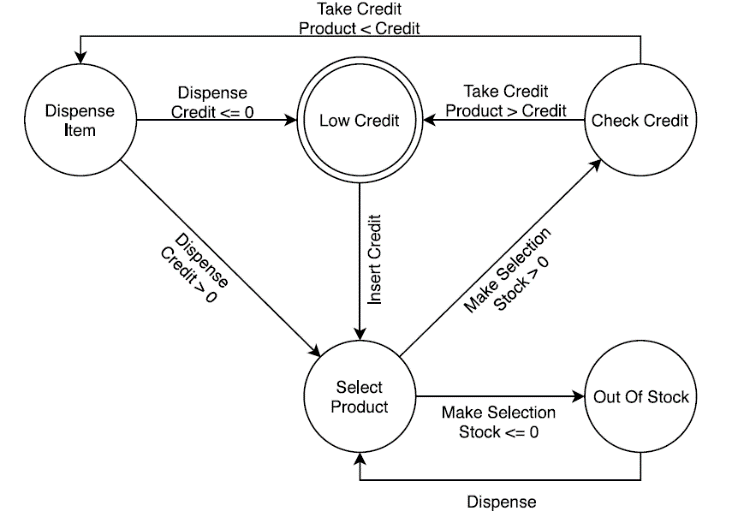
The view sales data screen will allow the administrators to see statistics about the machine such as how many products have been purchased, which products are out of stock, the total amount of money made so far, the total number of orders made, and the total number of users registered on the machine.

# Implementation

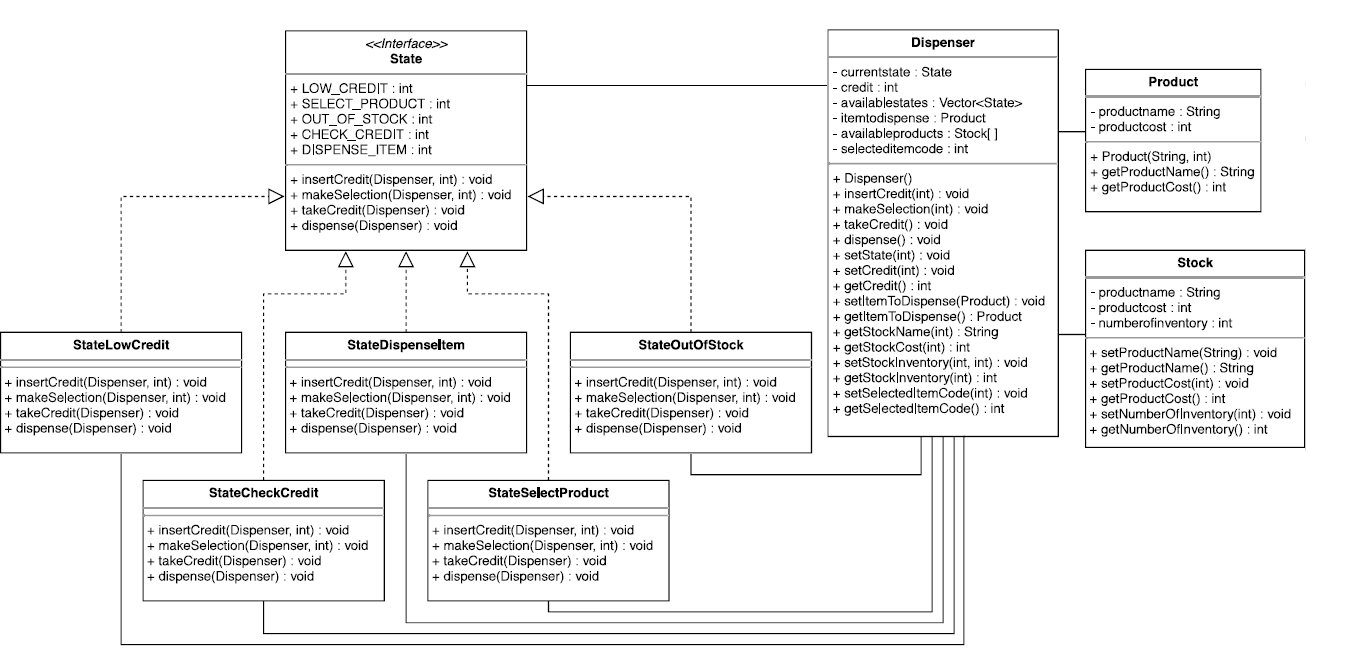
The project is carried out in stages with 4 prototypes being developed first followed by the android application. This leads to the application be developed in small chunks at a time rather than the entire program being done all together. It also gives the chance to test the program at the end of each stage to make sure it is stable before adding more features.

## First Prototype

The first prototype of this project sets up a finite state machine to use as the basis of the dispenser for the smart vending machine. For now, the finite state machine will be used to model a basic vending machine with addition functionality being added in later prototypes. A finite state machine has been used for the dispenser because it allows more flexibility when it comes to adding more states to the dispenser. When using a finite state machine, new states can be plugged into the existing model.

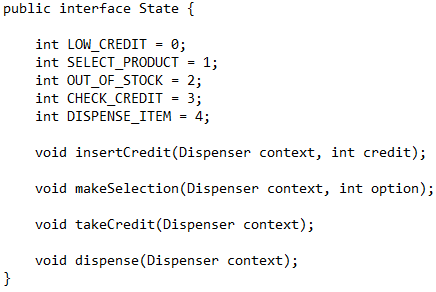


The basic dispenser state machine has 5 states, these are shown in the state diagram above. The transitions between states and the conditions that need to be true for the transition to take place are shown as arrows on the diagram and represent the functions that will be used in the prototype to switch between states.



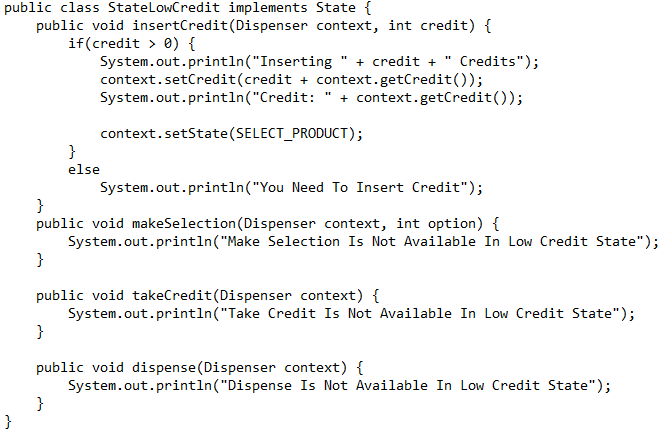
The UML diagram above shows the class layout for this prototype. The program will have a state interface class which all the state classes such as StateLowCredit and StateOutOfStock must implement. This means that they must all have the same functions as the interface. Each state shown on the state diagram has its own class with every possible transition being represented as a function. For states that do not support a certain transition i.e. inserting money when dispensing the item, an error message will be displayed, and the machine will not switch states.

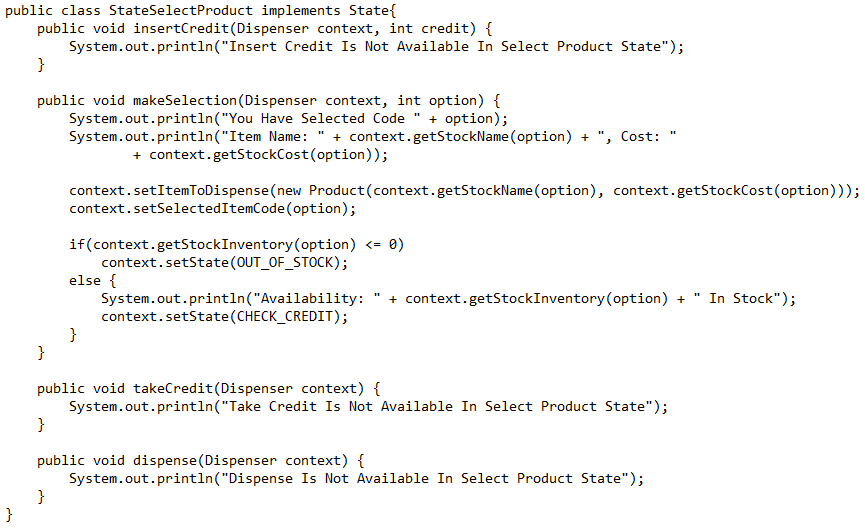
The programming language that will be used is Java. Java has been chosen for this project because it is the main programming language for android applications so by using Java most of the code developed for earlier prototypes can be incorporated into the final android application without the need to rewrite those sections from scratch.



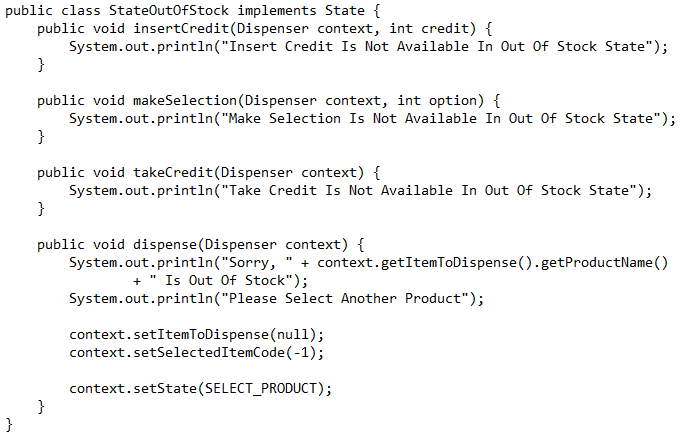
The state interface holds some static variables that all the states can access and functions that represent the transitions on the state diagram. The variables shown in the interface are used to keep track of the position of each state in a vector of available states that is created and used within the dispenser class e.g. LOW\_CREDIT has a value of 0, so the object created of StateLowCredit will be in position 0 of the available states vector. Each of the state classes must also have these 4 functions when implementing this interface. However, this leads to a state having functions for transitions that it cannot support such as the low credit state having a dispense function.

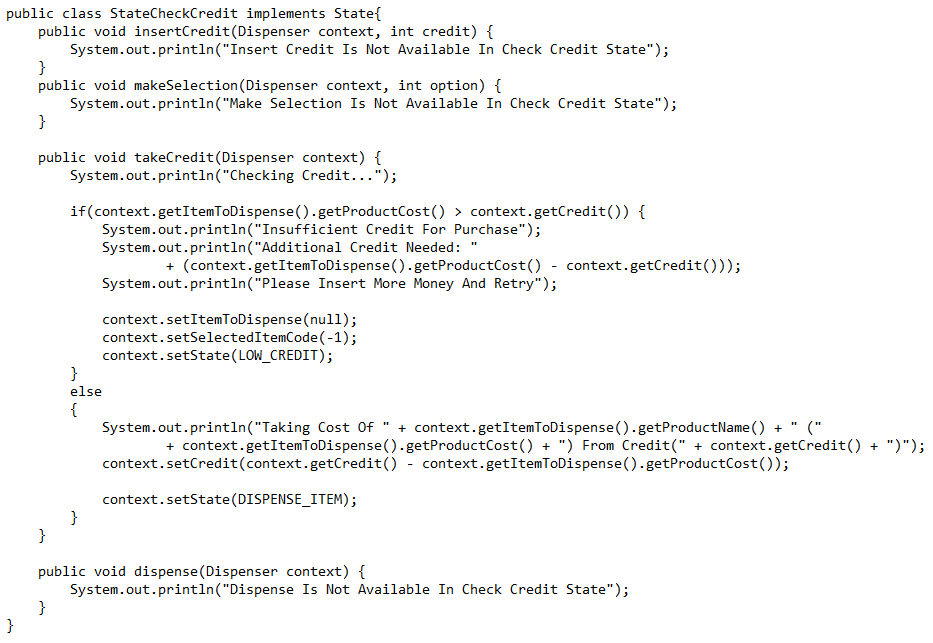
The transitions that the state does not support simply displays an error message so that the class can still implement the interface but does not include any transitions that the state does not support. Each of the states in the state diagram has its own class each with their own versions of these 4 functions shown above.

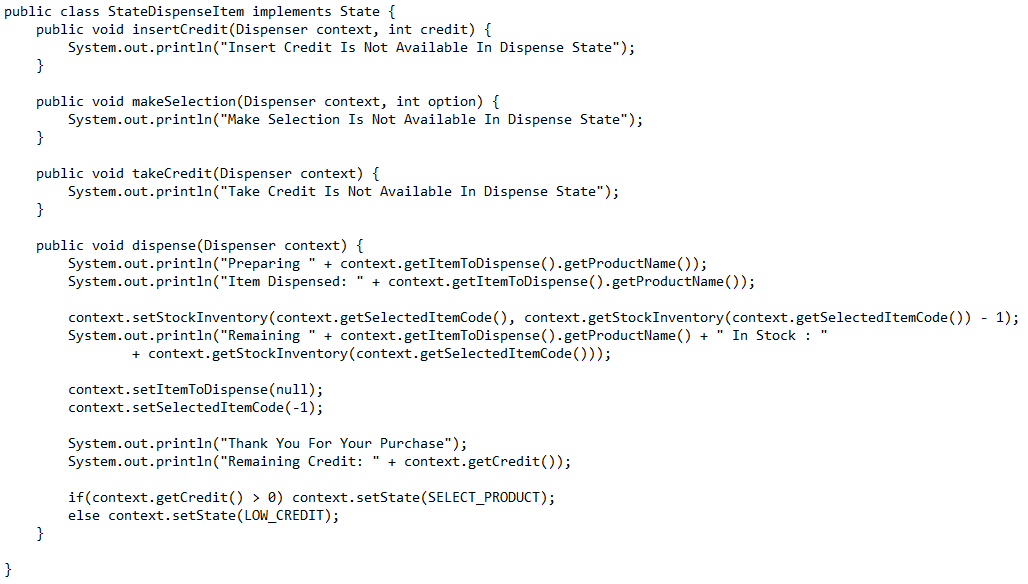
Each of the functions take a parameter of type dispenser called context, this represents the vending machine and what context it is in at the time of running the function. The dispenser class represents a vending machine and keeps track of its own context.

The state machine starts off in the low credit state. The code for the low credit state is shown above. The insertCredit function in this class allows the user to add credit to their account. Once credit has been added to the account, the state of the machine is moved to the select product state. All the other functions in this class are used to display error messages for the transitions that are not supported for this class.

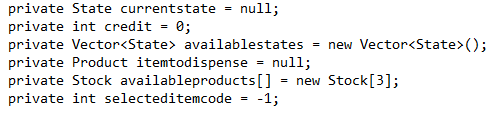
The select product state then allows the user to call the makeSelection function that takes a product code and sets it as the item is dispense. If the product is out of stock, the machine moves to an out of stock state. However, if the product is in stock, the machine moves to the check credit state.



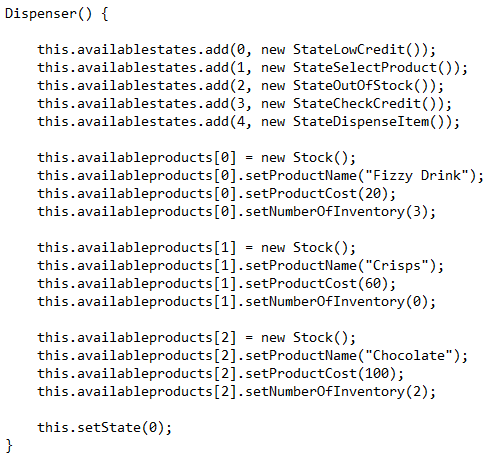
If the machine moves to the out of stock state, when the machine tries to dispense the item, an error message is displayed and the product to dispense is reset. The machine is then returned to the select product state for the user to select another product.

If the selected product is in stock, then the amount of credit entered into the machine is checked. If there isn’t enough credit in the machine, then the machine is reset and the changed to the low credit state. However, if there is enough credit in the machine, the machine moves to the dispense item state.

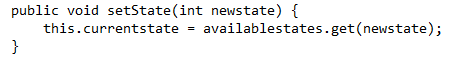
The final stage of the machine is the dispense item state. Here the product selected is dispensed and the cost of that item is removed from the credit in the machine. The stock of that item is reduced by 1 to reflect one of the items being purchased. The machine is then reset and changed back to the low credit state for the next order.



The context of the dispenser is kept track of using these variables within the dispenser class. The dispenser class has parameters to hold the current state that it is currently in, the user’s credit, a list of available states that the machine can transition to, an object to represent the item select by the user that needs to be dispensed, an array of available products and a record of the selected item code. All these variables are used as the context of the machine.



The constructor for the dispense is used to create an instance of each state class and add each instance to a vector of available states and set the initial state to the out of credit state. For this prototype the constructor is also where all the stock is added to the machine.



To change between the states, the dispenser has a set state function that takes an integer value which represents the index of the desired state within the available states vector. This function then gets the object at the index in the vector and sets it equal to the current state.

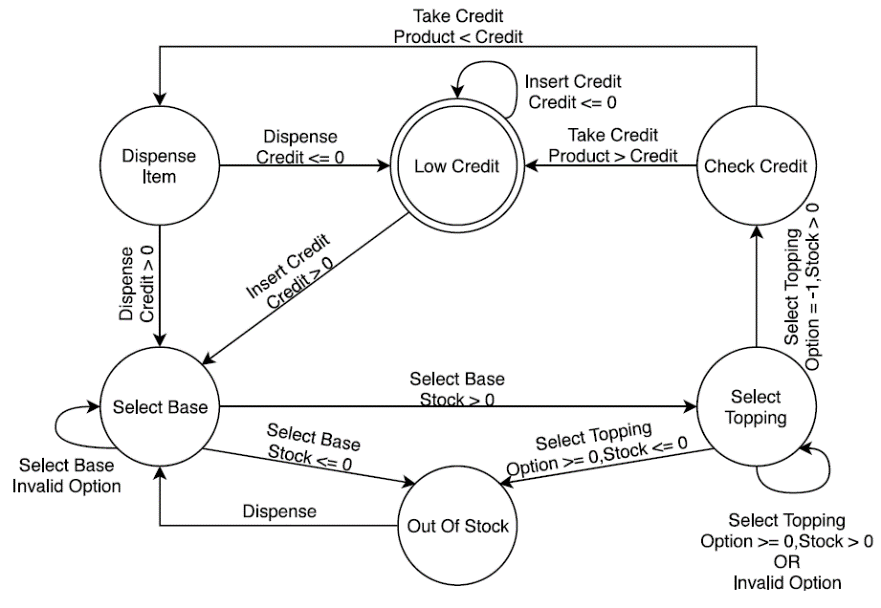


The dispenser class also has its own versions of the functions shown in the interface but without the context variable, these functions call their counterparts within the class of the current state passing in the current instance of the dispenser class as the context and any other parameters the functions require. This is also why the transition functions with error messages are needed, because the user does not immediately know which state the dispenser is in, it is possible to call an invalid transition in a given state. In this scenario, the relevant error message is displayed, and the state does not change letting the user try another transition to get to the next state.

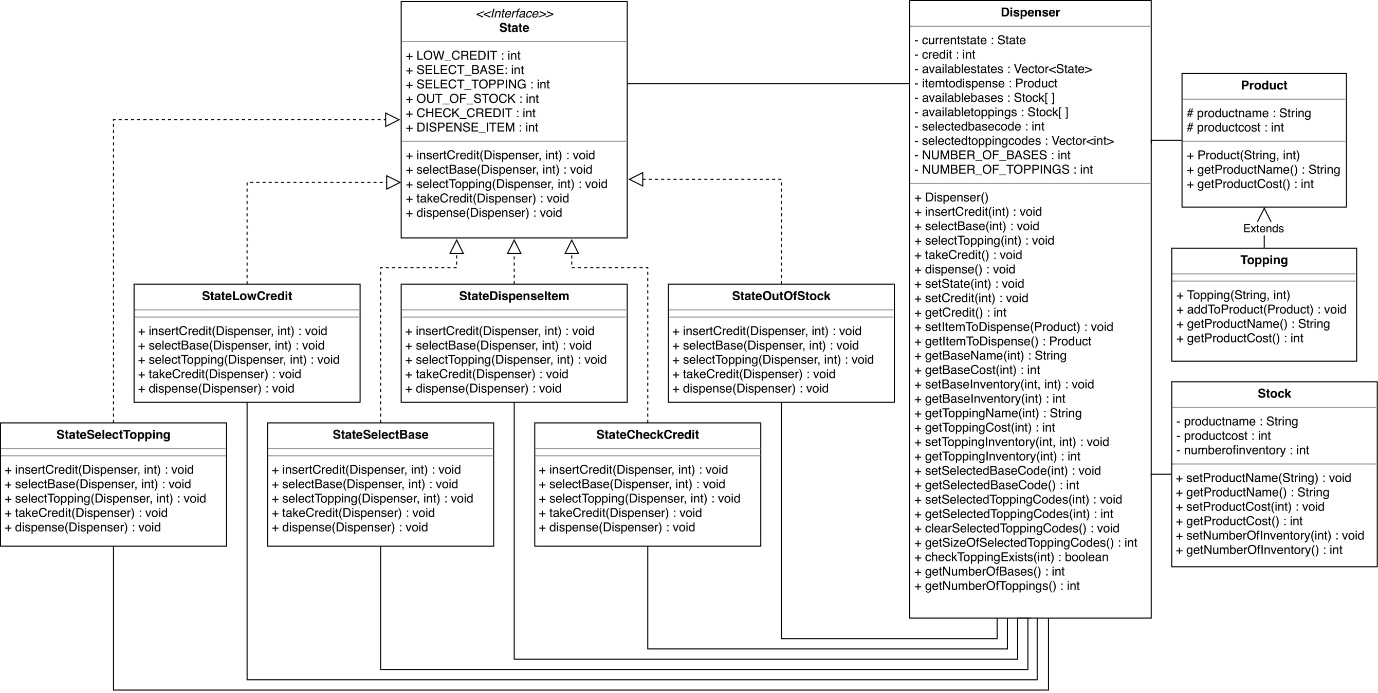
All the other functions in the dispenser class are getters and setters for the variables in that class, these are used so that the values of the variables can be fetched and modified from elsewhere in the program but the variables themselves are private to the class.

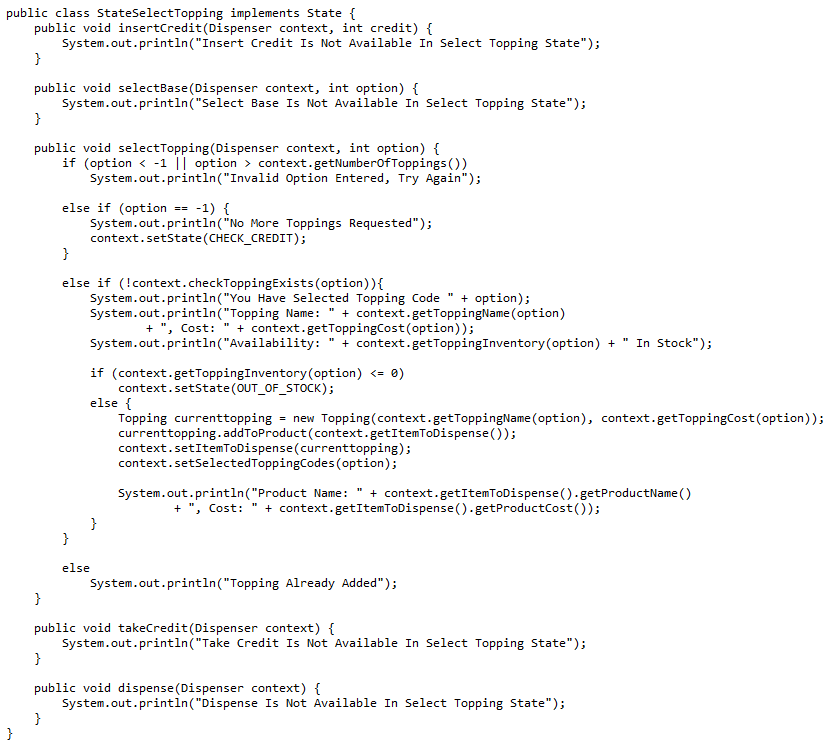
The product class is used to represent the products within the dispenser and hold attributes such as name and cost. The stock class is just used as a placeholder at this point to keep track of the inventory of the machine. This will be replaced by a database in a later iteration.

## Second Prototype

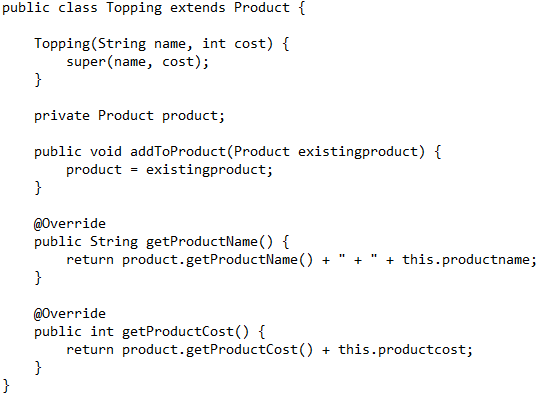


The second prototype builds on the state machine featured in the first prototype with the addition of a decorator pattern being used so that the products in the machine can be customisable. An addition state is added to the state machine called add toppings which allows the user to decorate their base product with topping items to get a customized item. The second prototype also adds user validations so that the user does not cause the program to crash when an unknown product code is entered.



The UML of the second prototype is very similar to the first prototype except for an extra state class, the topping class, additional variables in the dispenser class with setters and getters, and additional transitions in each of the state classes as shown in the state diagram.

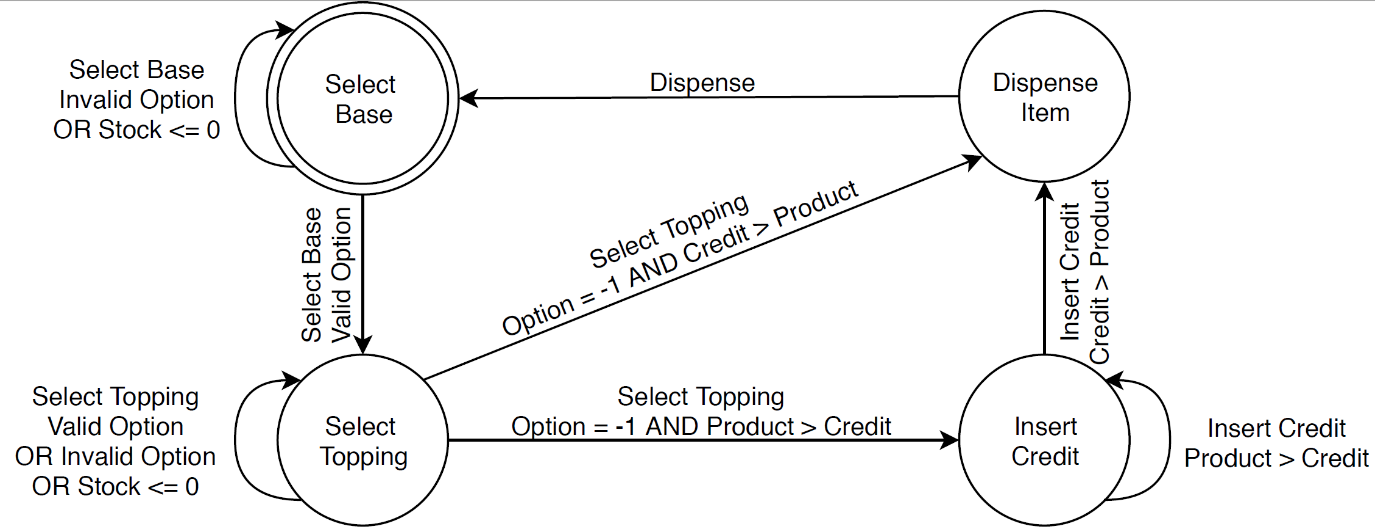
The new select topping state class implements the state interface like all the other state classes. The select topping state will appear after the select base state when a valid base has been selected. The new transition in this prototype is the select topping transition which makes use of a decorator to add toppings to the product. But firstly, the select topping function checks if the input is valid and outputs an error message if the input is invalid, this is the same in any part of the program where the user needs to input data. As with other transitions, if the dispenser is in a state that doesn’t support the select toppings transition, an error message will be displayed.

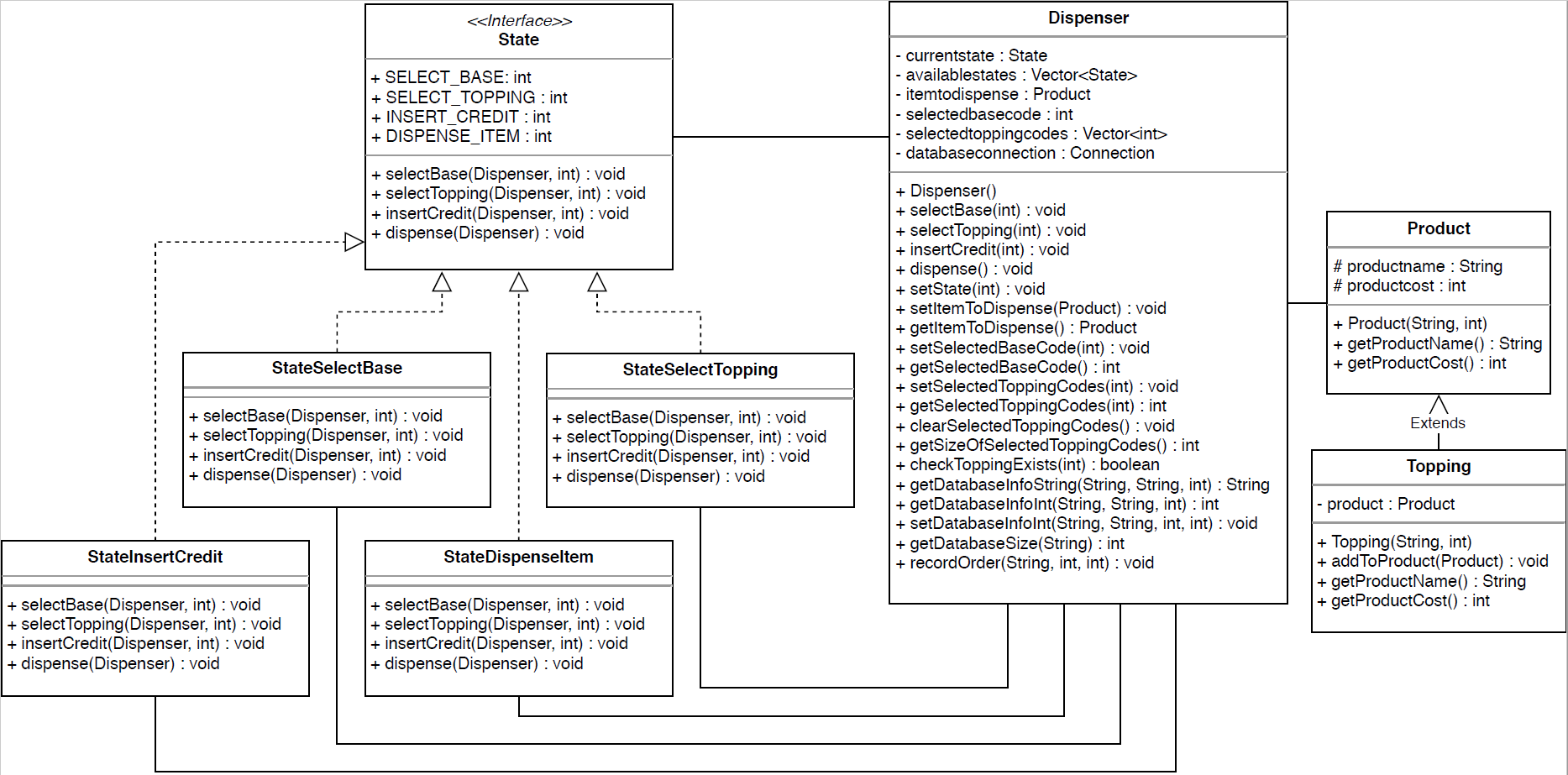


The topping class forms the basis of the decorator pattern. The class contains a variable of type product and essentially wraps a topping object around a product object so that it builds up the product with other objects just like adding toppings to food items. This is used so that users can add items to their base product items and makes the products in the machine customizable. The addToProduct function in the topping class take the existing product and sets it equal to the product which the class, the new topping object is then set as the existing product. This happens for as many toppings as the user wants to add and creates a nesting effect with objects being inside objects.

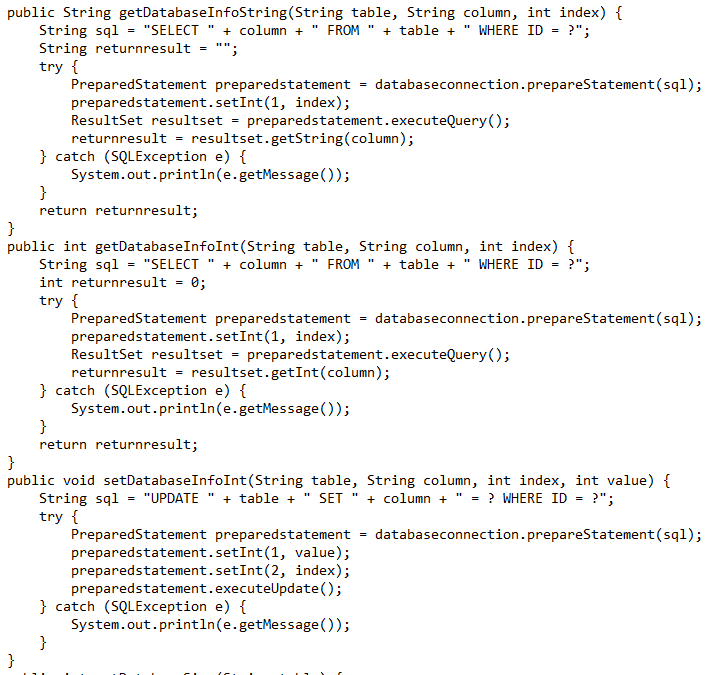
The getProductName and getProductCost functions get the name and cost of the product object instead the class and adds them to the variables of the topping class. This uses a form of recursion so that it collects the name and cost for all the nested objects within a decorated item.

## Third Prototype

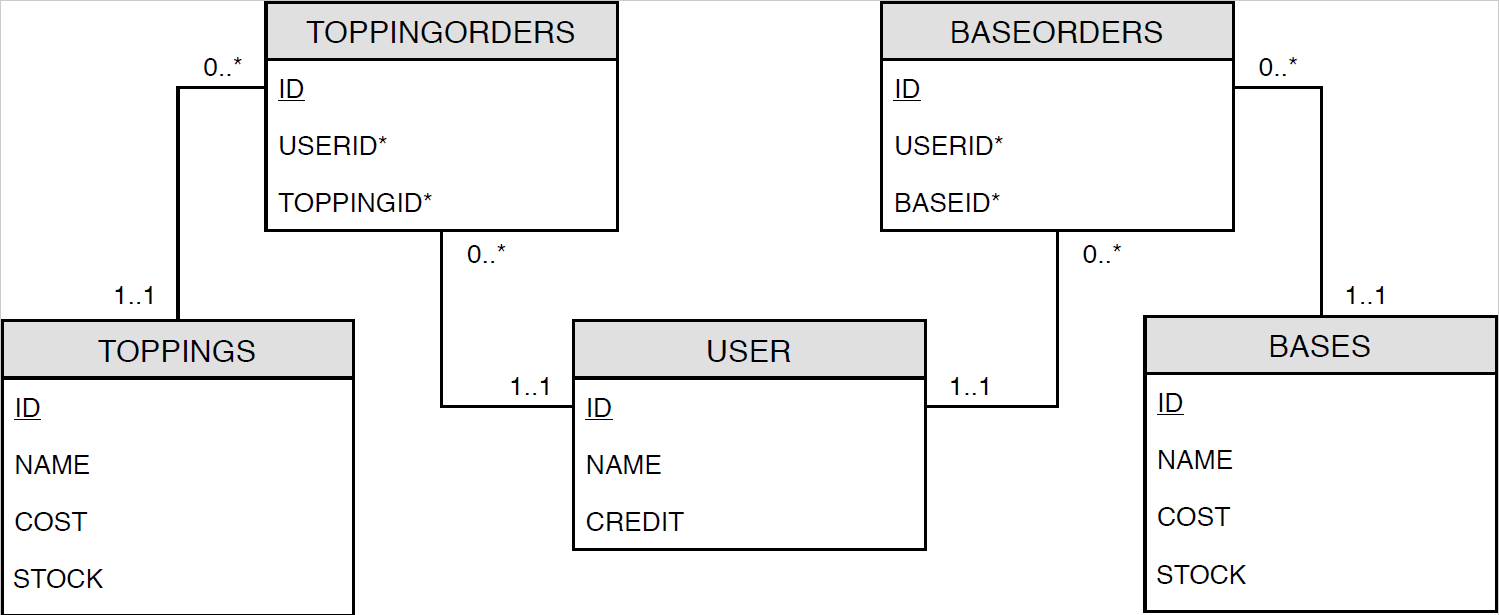


The third prototype removes the out of stock state and the check credit state from the program as they contained transitions that caused the user to re-enter their order if the they didn’t have enough credit or there was an item out of stock. In this prototype, the user enters the item that they desire first then enter credit if they need to. If an item is out of stock, then a message is displayed, and the user can enter another product code without having to enter the entire order again.

This prototype also removes the stock class and replaces it with a database, so all products, users and orders are all recorded in the database. The leads to the array of products being removed from the dispenser class as well as the number of products variables.



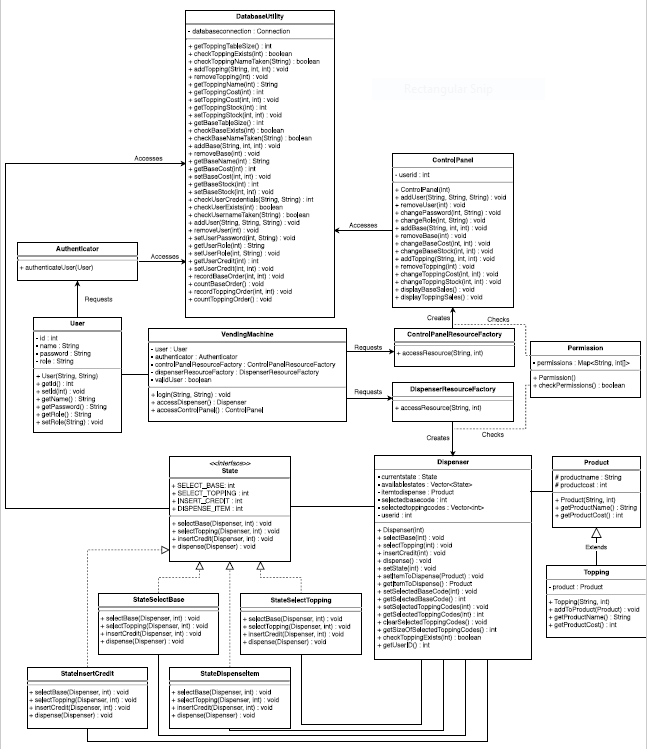
This program now uses an JDBC connection object to interact with an SQLite database and uses prepared statements to run queries to fetch information from the database. The database is currently only local to the device. But if there were multiple machines in operation then the database for users would be storage online so that users can log in to any of the machines and use one account for every machine of this type. The move to a database was made so that the machine would allow multiple user accounts to be stored on the machine and the database provides a form of persistent storage so that the details of the products and the users are saved to file in between runs.

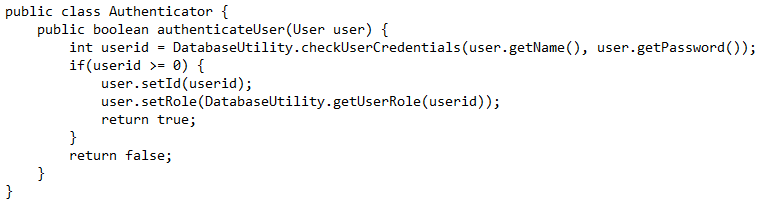


The database has five tables that are used to record and store information about each user, topping and base or are used to record which user order which bases and toppings. The structure of this database would have to change when android application is implemented but this is all that is needed for now.

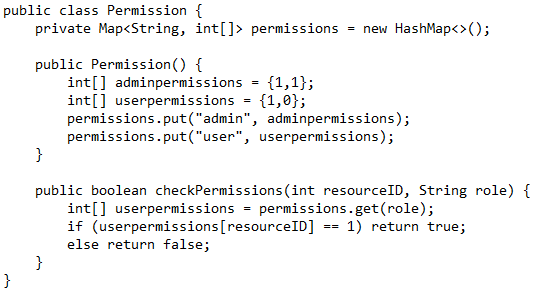
## Fourth Prototype

The fourth prototype of this project adds an authenticator and authorisation pattern to the dispenser for users to login into the machine and having different levels of access for users and administrators. This prototype also adds factories for creating resources so that some items are not created until they are needed. A control panel is also added so that the machine’s inventory and users can be edited from within the program dynamically. Both the dispenser and the control panel are held in a class called vending machine which will now become the main object of the project.

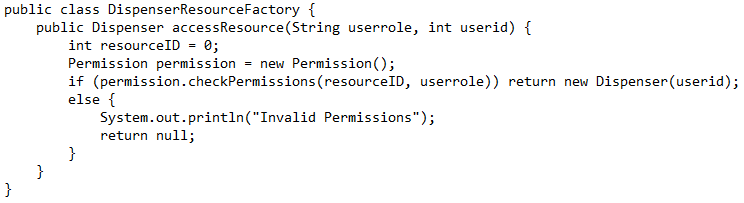
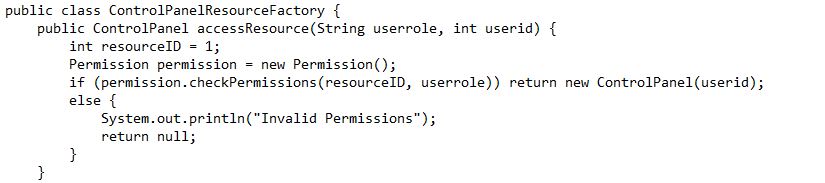


This prototype adds new classes to form the authorisations and authentication patterns as well as the factory patterns. These new classes are the Authenticator class, Permission class, ControlPanelResourceFactory class and the DispenserResourceFactory class. The control panel class has also been added to help manage the vending machine and all the database code has been taken out of the dispenser class and put into its own DatabaseUtility class.

The Authenticator class is used to take the credentials entered by the user and check if they are valid credentials. The class does this by checking the database user table to see if the credential match for the specified user. If valid credentials are entered, the user’s details are fetched from the database and set as the current user’s details on the machine then return true signalling a valid user. If there are invalid credentials entered, then the authenticator will return false. An authenticator is used to make sure that only users with valid credentials have access to their accounts on the machine.

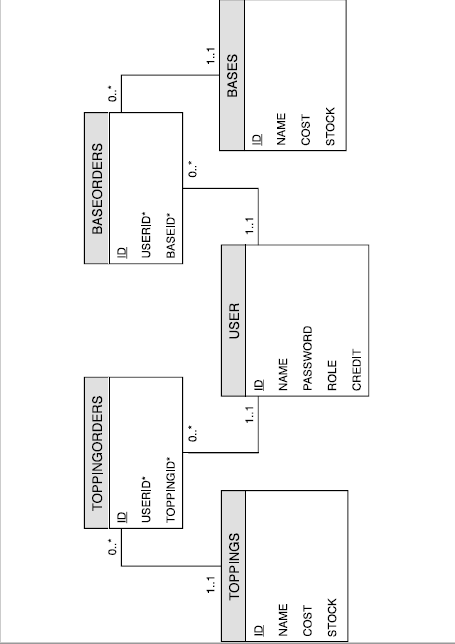


The permission class forms the authorisation pattern and holds a map of permissions with each permission haven a role associated with an integer array. The checkPermission function will take the ID of the requested resource and the user’s role to check if the user has the relevant permission to access the requested resource. The function will look up the permissions associated with the user’s role and check the element in the position of the resource ID’s value. If this value is a 1, then the user has the correct permissions to access the resource and the function returns true. If the value is a 0, the user does not have relevant permissions and the function returns false.



Both the resource factory classes have the same function except one is for a Dispenser object and the other is for a Control Panel object. The accessResource function in both classes check the user’s permissions to see if they have the right role to access the resource. If the user has permission, then the resource factory will create a new object of the relevant resource and return it i.e. DispenserResourceFactory will return a Dispenser object and ControlPanelResourceFactory will return a ControlPanel object. If the user does not have the correct permissions, then an invalid permissions message will be displayed, and the function will return null. Resource factories have been used to make sure that a valid user is trying to access the resource before the resource is created. This prevents the resource being available in memory if a person with invalid credentials tries to access it as it could still be accessed.

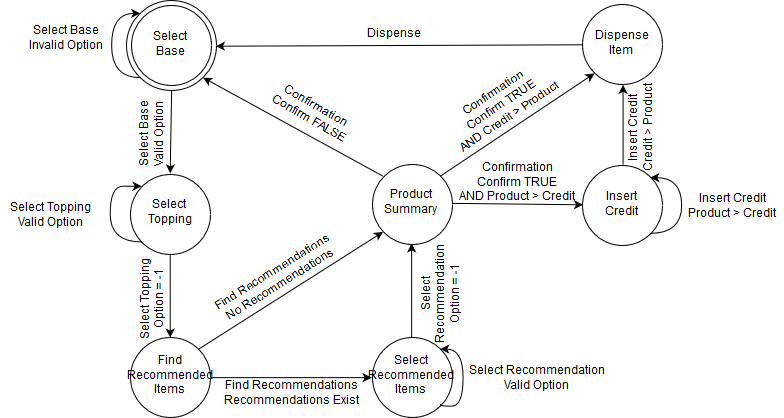
The vending machine class acts as an interface for the whole system. The vending machine class contains an authenticator, a control panel resource factory, a dispenser resource factory, a user and a Boolean to represent if there is a valid user signed in. In this prototype, the vending machine is created in main and has functions to login, access the dispense and access the control panel. However only administrators can access the control panel which is why the authorization and authenticator patterns have been used. The dispenser and control panel are only created when they are called and if the user has the correct permissions, this prevents an unauthorised user from accessing a resource that they are not supposed to access.



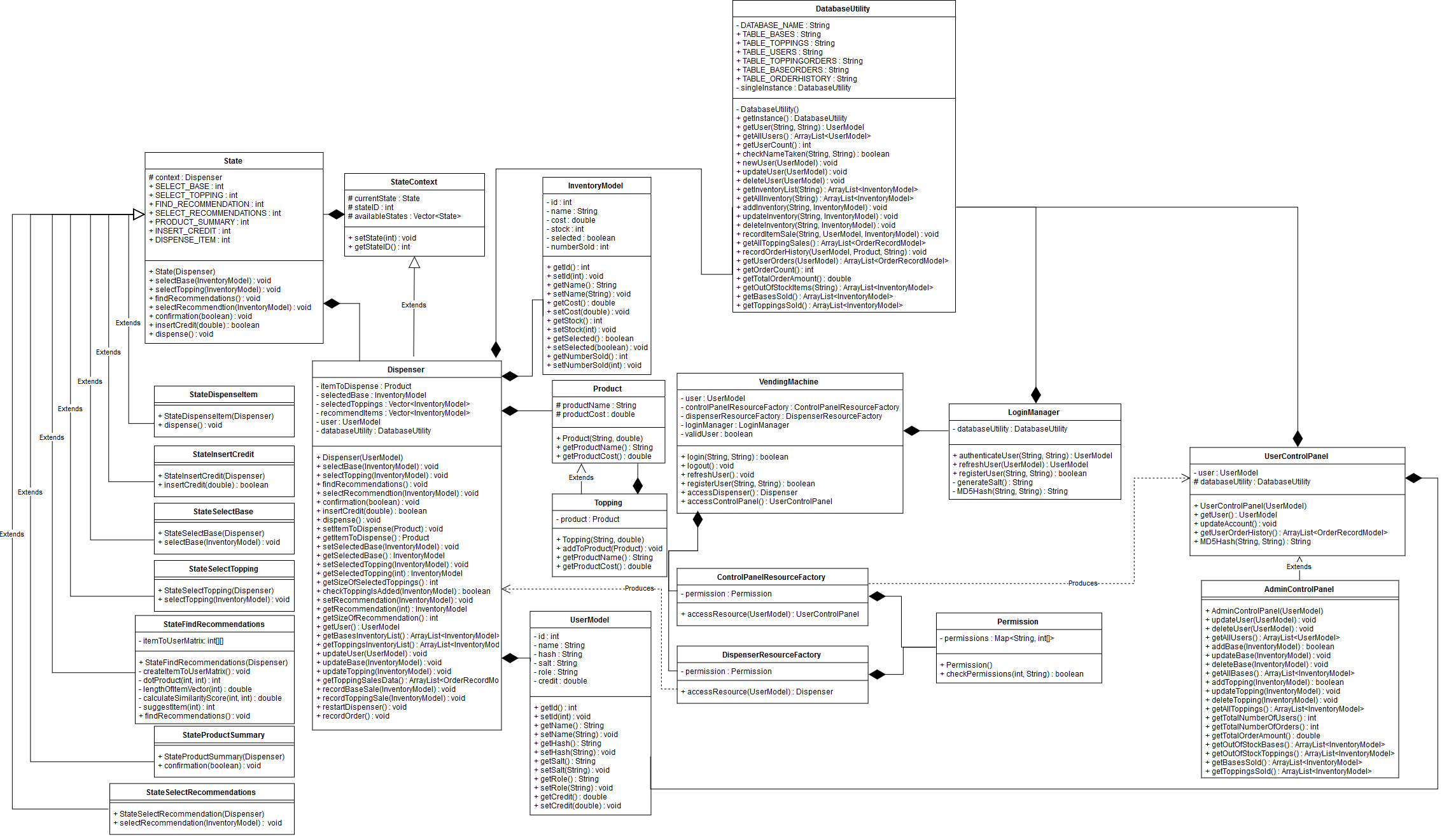
To support the use of customer profiles, a User class has also been added to store basic details about the user such as name, credit and role. The database has also had to be updated to include the user’s passwords and roles.

## Android Application

The android application is the final program of the project and incorporates all the previous 4 prototypes into a program with a GUI. In addition to the previous 4 prototypes, the application also has a recommendation system and a sales data screen along with password encryption.



This iteration of the project adds the find recommended items, select recommended items and product summary states. The recommended items states form the recommendation system with the find recommended items states running the calculations to find possible items to recommend and the select recommended items allow the user to choose items from a list of recommendations. The product summary state asks the user to confirm that the product is correct before the purchase is continued.



The UML above shows the class for the underlying smart vending machine engine without any of the GUI classes so that it only shows the more important parts for now.

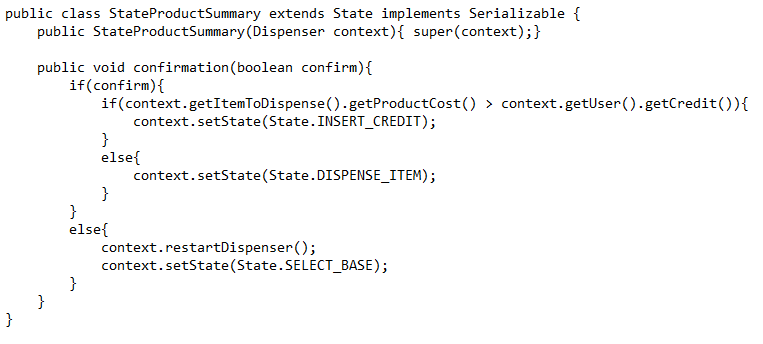
The state interface that has been used in the previous 4 prototypes has been replaced with a state class which the different states of the machine inherit from. This was done because there was no direct way to display an error message from within this code as it would have meant a lot of the underlying code would have had to been written into each of the classes for the GUI.

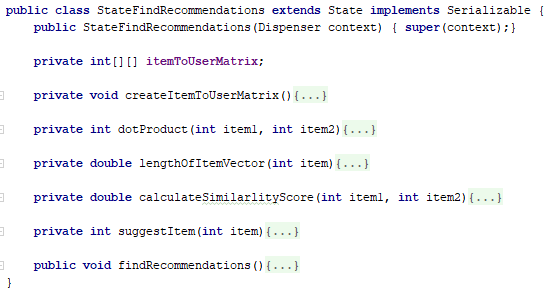
All the variables and functions relating to states have been taken out of dispenser and put in a class called StateContext. This helps make the code more reusable in future projects. The dispenser class now inherits from StateContext.

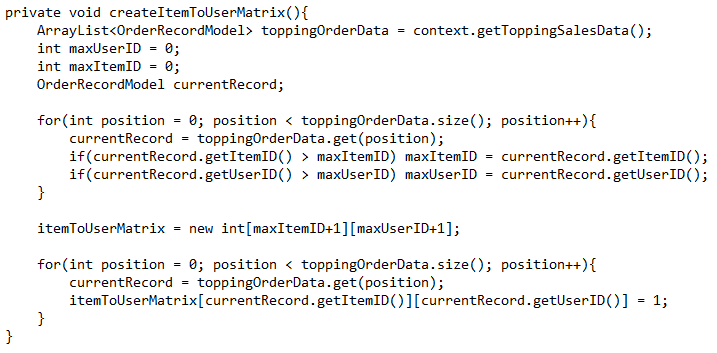
The Authenticator Class from V4 has been renamed to LoginManager and now also handles the registration of new users. The code from the authenticator has been tidied up by having the database utility return a UserModel Object instead of separate strings. This saves multiple database look ups for one login.

The LoginManager class also features the password encryption. Password encryption was added to the application to prevent passwords being stored in plain text. The application generates a salt string for each user, the user’s password is then concatenated with the salt and the entire string is turned into a hash value using the MD5 algorithm built into java security. the hash value and the salt string are then saved to the database instead of a password. When the user tries to login to the system, the password that they entered in order to attempted to login is concatenated to the user’s salt string and the entire string is turned into a hash value using the same MD5 algorithm. If the hash value from the login matches the has saved for the user, then the password is correct, and the login is successful.

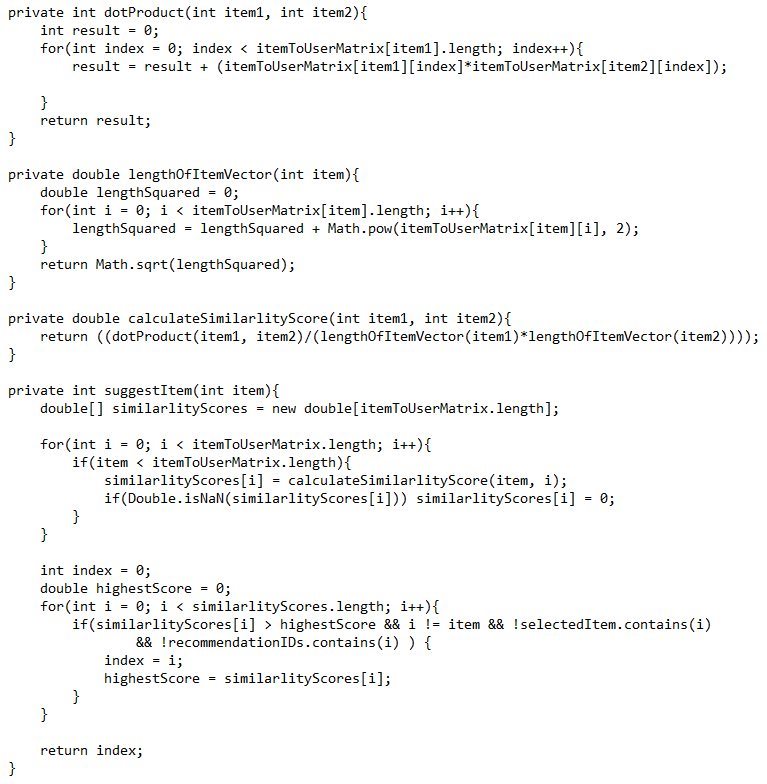
The ControlPanel Class from prototype 4 has been split up into UserControlPanel and AdminControlPanel with AdminControlPanel inheriting from UserControlPanel. This gives the users a control panel but then is extended in the admin control panel where a lot more operations can be carried out.

The DatabaseUtility class has been rewritten using the singleton pattern. This ensures that there is only one instance of DatabaseUtility at any one time. The use of a singleton pattern for a database works in this scenario as there will only ever be one database transaction at any one time. if there was going to be more than one database transaction at a time, then a singleton pattern wouldn’t be efficient.

The product summary state requires the user to either enter true if the product displayed is correct or false if the product is incorrect. If the user enters true, the state machine will continue with the purchase. However, if the user enters false, then the dispenser will reset, and the state will be set back to the select base state. This state has been added as a layer of confirmation for the user so that they can be sure that everything has been entered correctly before proceeding.

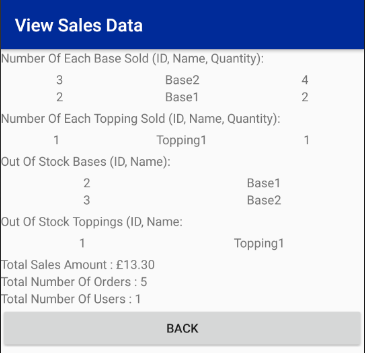
The find recommendations state is where the calculations required for generating recommendations are carried out. The application uses cosine similarity and purchase history to generate a list of products that the user might like based on what other users have purchased.

To start finding recommendations, the program must first collect the data stored in the database for topping purchases. This data is used to create a User-Item matrix that shows which users bought which topping items.

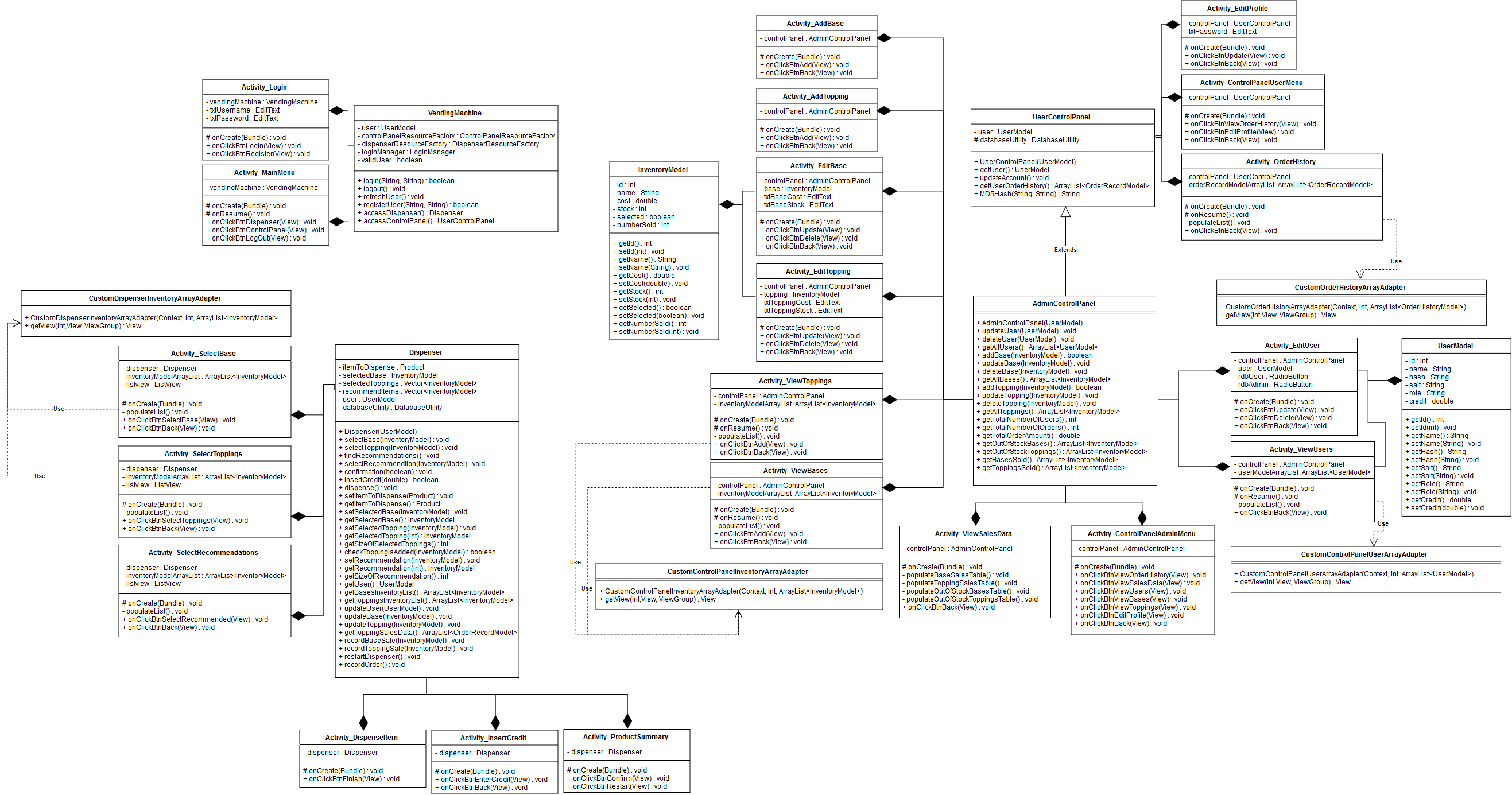
Once the User-Item matrix has been generated, the program will run through every item in the matrix and calculate its similarity score compared to the item that has been given as a parameter. The program calculates the similarity score by representing the item columns in the matrix as mathematical vectors so that the dot product between the 2 vectors and the length of the vectors can be calculated. By dividing the dot product between the vectors by the product of the 2 vector lengths, a value between 0 and 1 can be calculate which 0 being least recommended and 1 being the most recommended. The index of the product with the highest similarity score is then returned as it would be the most recommended item.

The program runs the recommendation algorithm for every topping that the user has selected. The findRecommendations function will generate a list of recommended toppings depending on if the item is in stock and if it hasn’t already been added to the product already. If there are recommendations available, the program will go to the select recommendations state otherwise the program will go to the product summary state. The select recommendations state is identical to the select toppings state except there is only the recommendations to choose from, not the entire inventory of toppings.

This approach of a recommendation system has been chosen because it requires the least computational resources considering that the algorithm is being created to run on an android device and must generate recommendations quickly on the go. The recommendation system will not recommend a product that no one else has bought so it does require more data in order to be more accurate.



Within the admin control panel of the application, the sales data for the machine can now be displayed. The data includes the number of items that have sold, the items that are out of stock, the total amount of money made in sales so far, the number of orders places and the number of users currently registered on the machine. This data should provide the suppliers with some information about which products are selling well in the machine and which ones they should take out and replace.

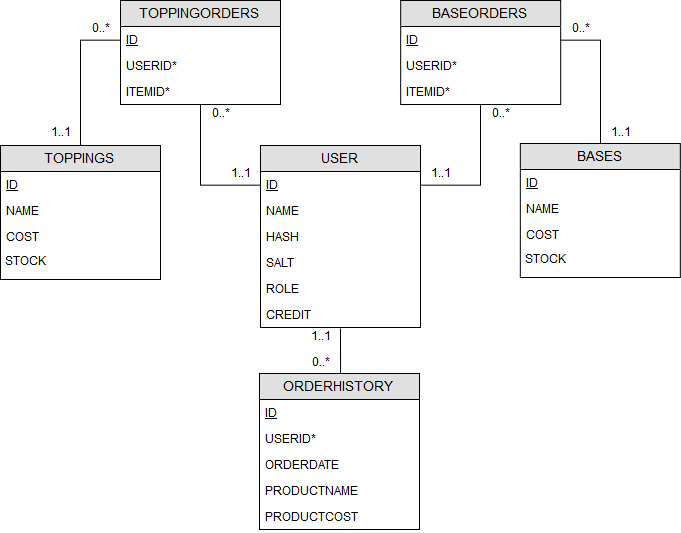


The UML above shows all the GUI classes and how they interact with the 3 main classes of the program which are the vending machine, dispenser and control panel classes.

All the classes related directly to the android application interface and event handling have the prefix “Activity\_” and each of them relates to a different screen of the application. Objects of VendingMachine, Dispenser and ControlPanel are passed along from each activity to keep everything operating in the same instance of the classes.

All the project files for the smart vending machine app have been included for this Milestone. The .XML files serve as the layout files for the app’s GUI and the .java files serve as the logic behind the app.

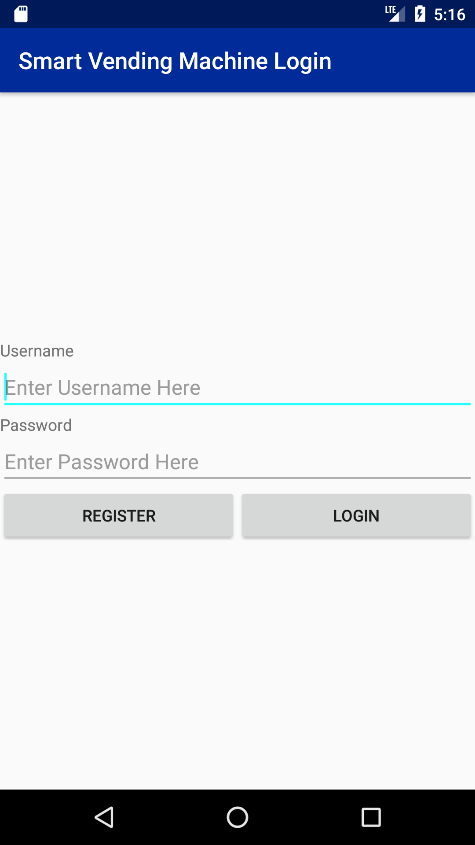
The android application only serves as a GUI to the smart vending machine and all the underlying logic for the application is built up from the first 4 prototypes discussed. This is so that other interfaces can developed for the same program if needed in the future.



For the new features in this implementation, a new table has been added to the database called order history. This new table records the name, cost and date for each order. The user table has also been changed to store a hash and salt attributes instead of a password attribute.

# Testing & Evaluation

This section of the report will demonstrate the smart vending machine application working and evaluate the application along the way. Testing of the application will focus on the 3 main parts of the application which are the login system, the dispenser and the control panel. Testing will consist of Unit tests along with GUI checks.

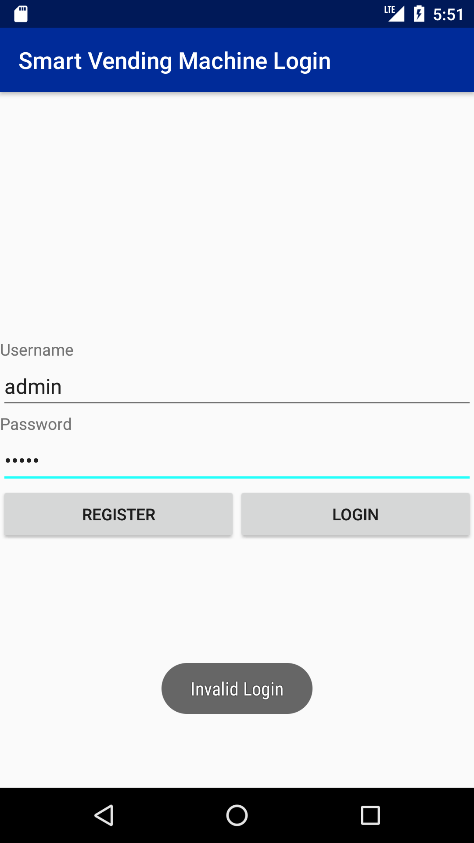


The login system is the first screen that appears to the user when they open the application. The login system needs to be tested for correct login, incorrect login, blank text fields, new user registration and attempted registration with a duplicate username.

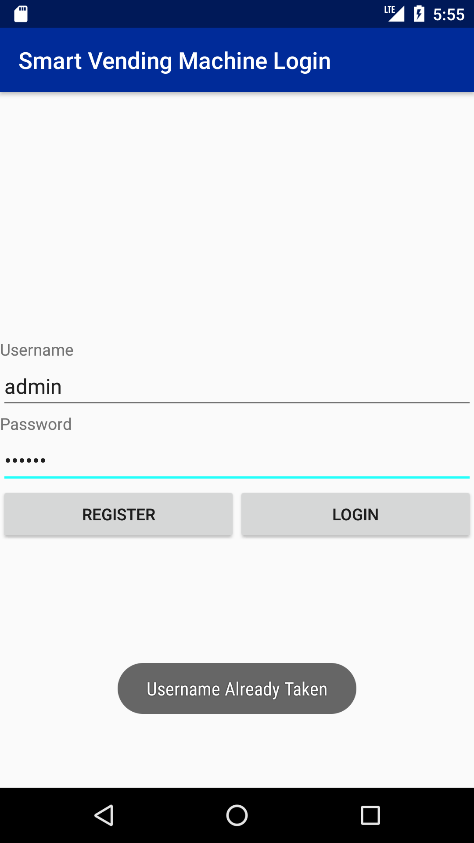
Unit tests can be used for testing the login system except for checking if the text fields are blank. the following unit tests was written to check the correct behaviour of the login system.



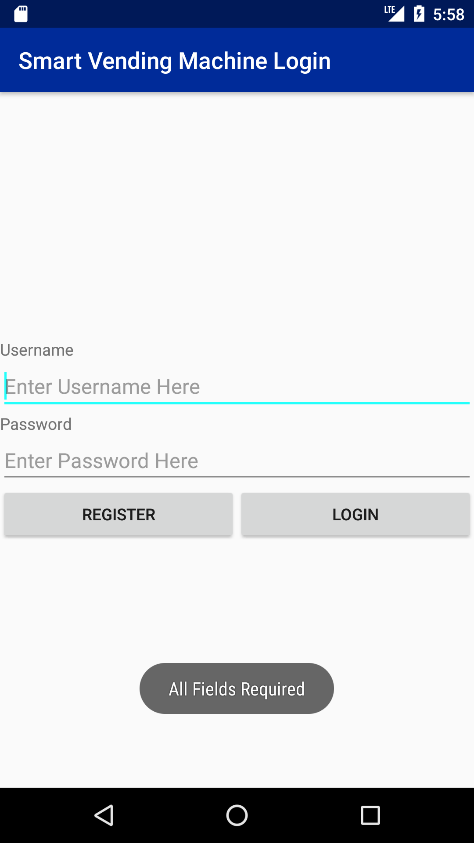
At the time of testing, only one user was registered on the system which had the username “admin” and the password of “admin2”. The correctLogin() test attempted to login and expected a UserModel object back as it would have fetched the user’s details from the database. The incorrectLogin() test attempted to login with incorrect credentials and expected null back as the login failed. The registerNewUser() test attempted to register a new user with a unique username and expected true to be returned as the register should have been successful. The registerNonUniqueName() test attempted to register a user with a name that another user already has, this test was expected to return false as the register would have been unsuccessful. These unit tests run successful showing the correct behaviour of the login system.



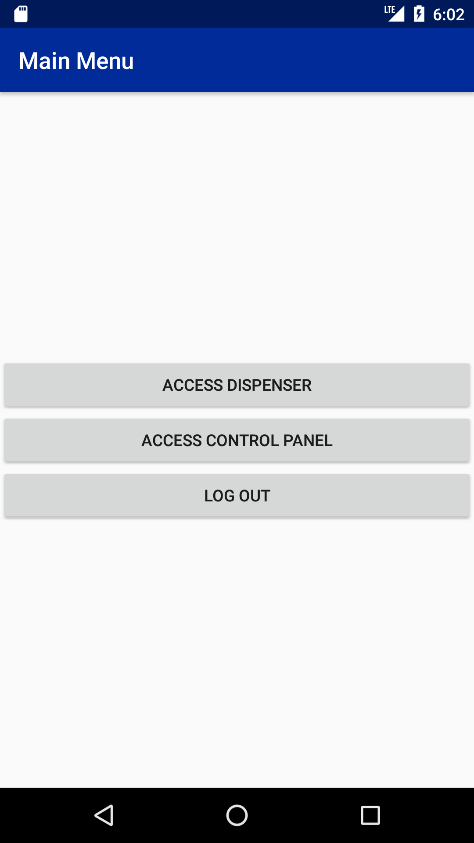
If invalid credentials are entered when the application is in use, a message will appear to inform the user that the credentials they have entered to login are invalid. A successful login will take the user to the main menu screen.



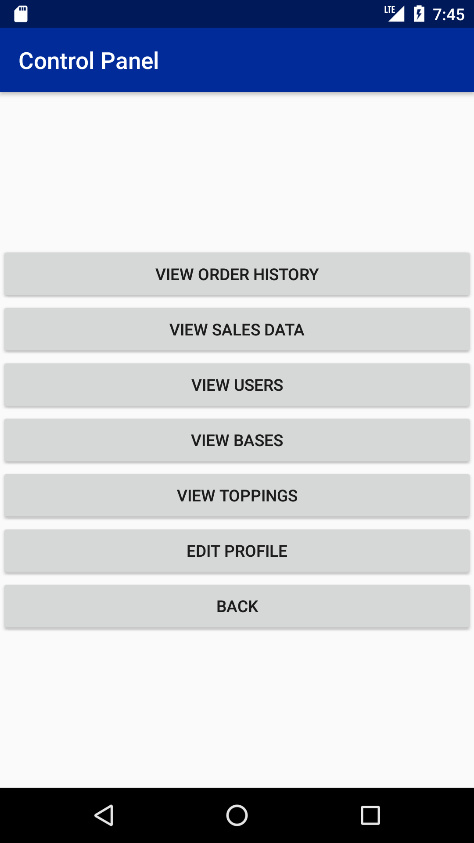
Similarly, if a user tries to register an account with the same username as an existing user, a message will appear to inform the user that the username they have entered is already taken.



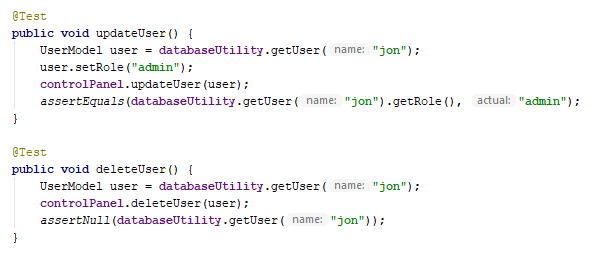
If one or both fields on the login screen are left blank, the user will be prompted to fill in all fields on the current screen. These results show that the behaviour of the login system and the GUI interface is correct and that no one can log into the machine without valid credentials and no one can register a new account if the username is not unique.



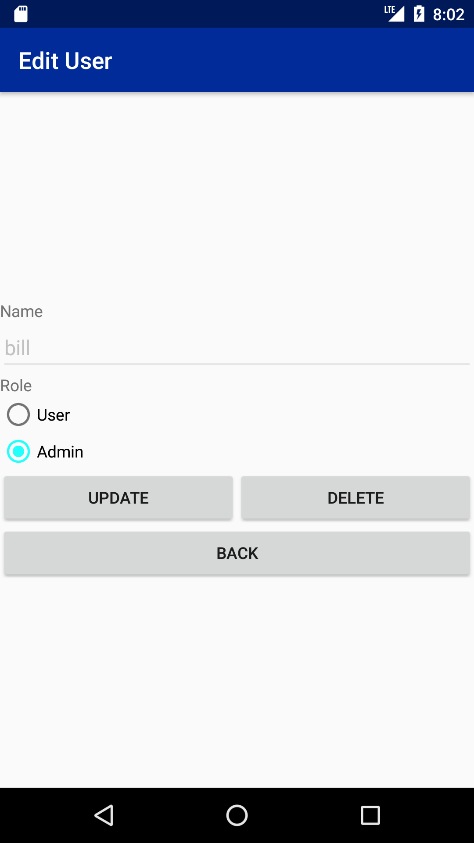
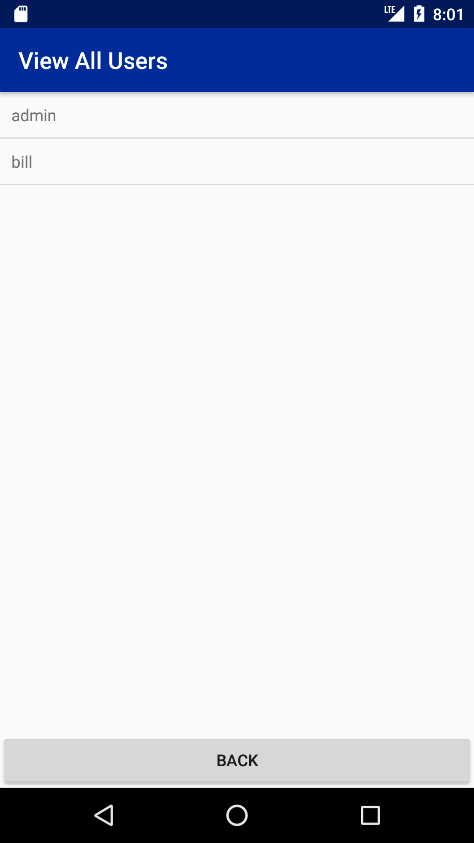
After a successful login or registration, the user is taken the main menu.



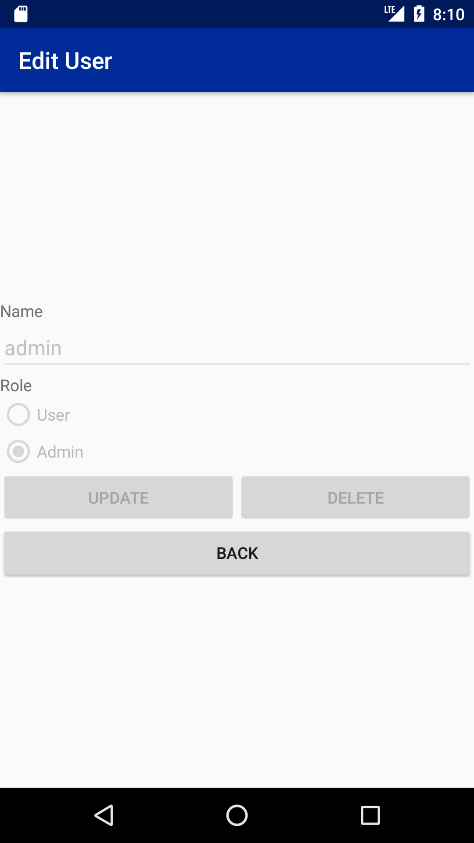
The control panel is the next component to test. Firstly, unit tests can be run on the AdminControlPanel class as by testing all the available functions inside this class, the behaviour of UserControlPanel can also be verified. The following unit tests were ran for the control panel. At the time of testing, there were 2 users in the system with the usernames “admin” and “jon”. The inventory in the machine was empty.



Firstly, the functions to edit the users were tested to ensure correct behaviour. the updateUser() test edits the users role and then checks if that role was changed in the database. The deleteUser() test then deletes the user and expects not to get an object back when trying to find the deleted user. Both tests passed.



Clicking on the view users option in the control panel shows a list of users currently in the system. Selecting one of the users allows the current user to change the role of the user or delete the user from the system. These features were tested in the unit tests on the previous page.



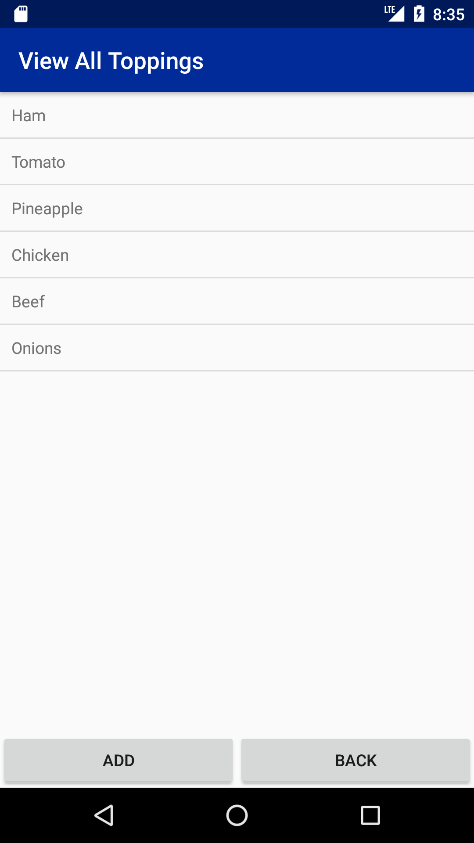
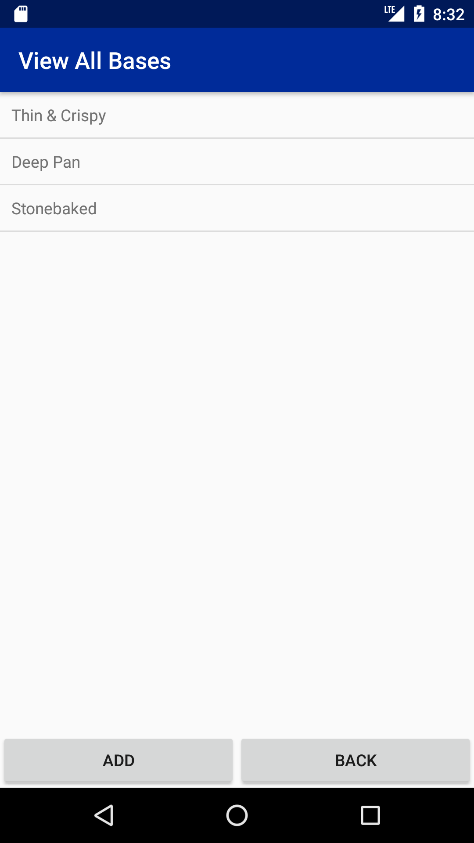
If a user selects themselves from the list, the form will be disabled as an admin is not allowed to delete themselves or change their own role. This ensures that there is always one admin in the system.



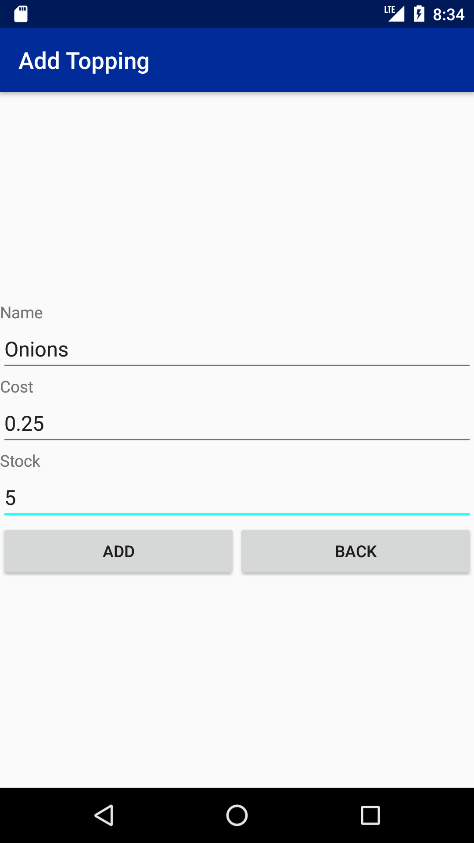
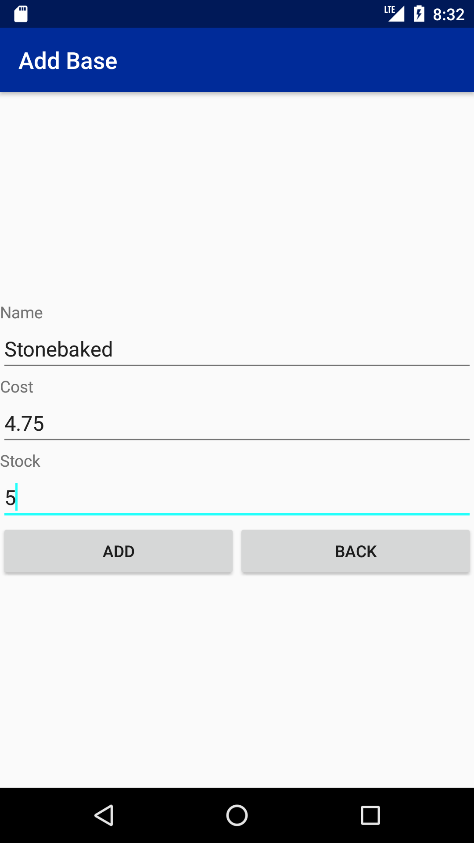
The above unit tests verify that the behaviour of the functions to edit base products is correct. The addNewBase() test attempts to add a new base to the database would should return true when that base is looked up. The addExistingBase() test tries to add the same base again to the database but should expect false to be returned as duplicate item names are not allowed. The updateBase() test tries to update the stock of a base and checks to see if the change has occurred in the database. The deleteBase() test deletes a base from the database and verifies that the item has been deleted from the database. All these tests ran successfully which ensure that the functions behave correctly.



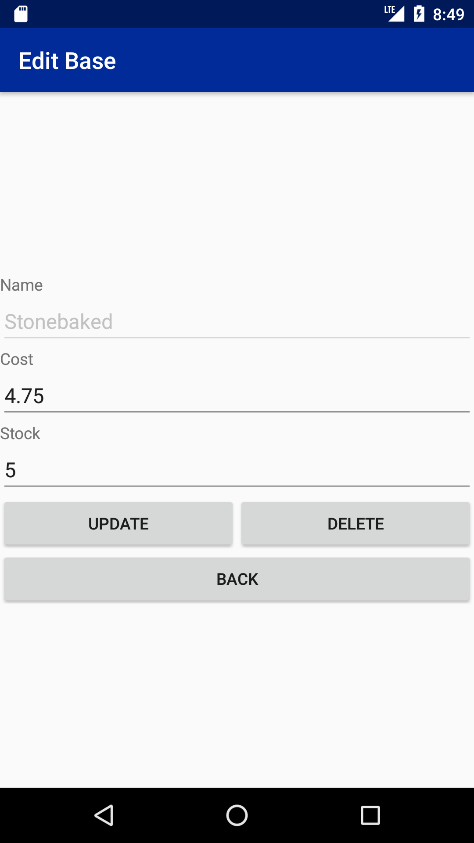
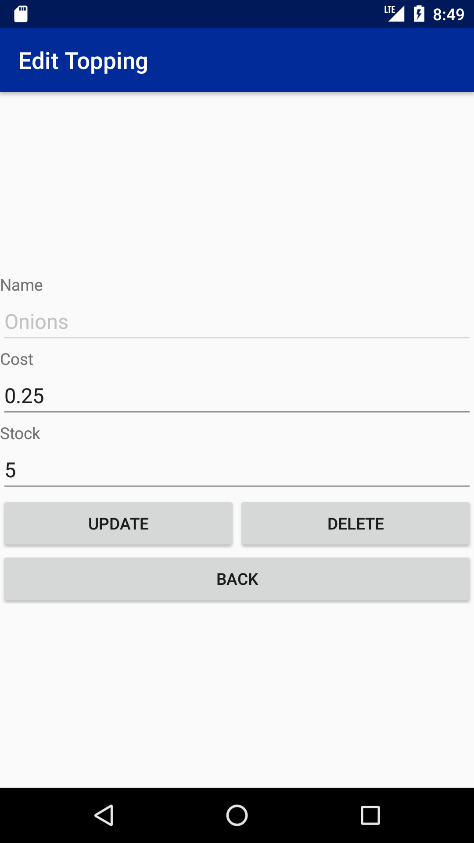
The same unit tests are run for functions to edit topping products. These tests are identical to the previous tests for editing base products except they are for editing topping products instead. All tests ran successfully.



Clicking view bases / toppings in the control panel will display a list of inventories. From this screen, the user can either add a new base / topping or they can edit an existing item.



Clicking the add button will allow the user to enter the name, stock and cost for the item that they want to add. All the fields are required and have validation to make sure all the values are in the correct format before continuing.

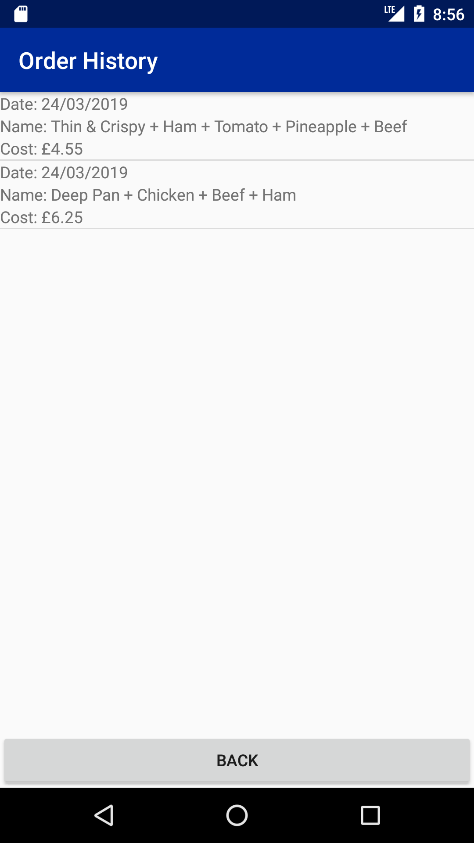
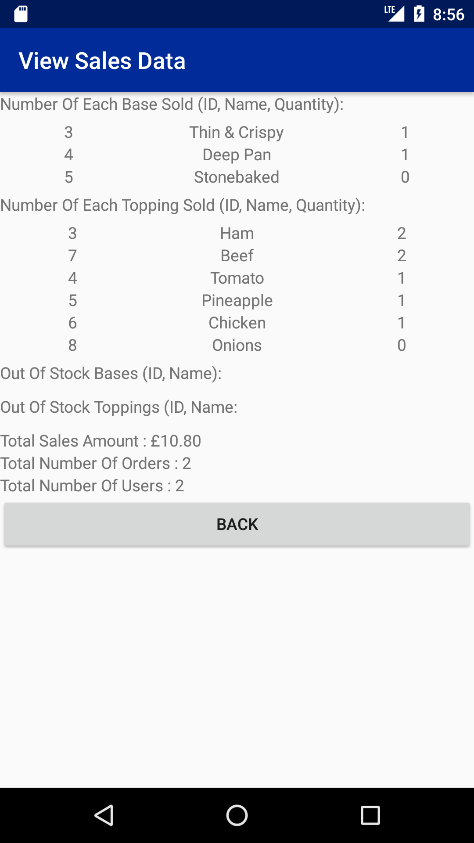


When the user edits an item, a similar screen appears except the name field is disabled to stop the user changing the product name. The user can also delete an item when editing the item. All fields are still required, and the validation is the same as the add item screens.



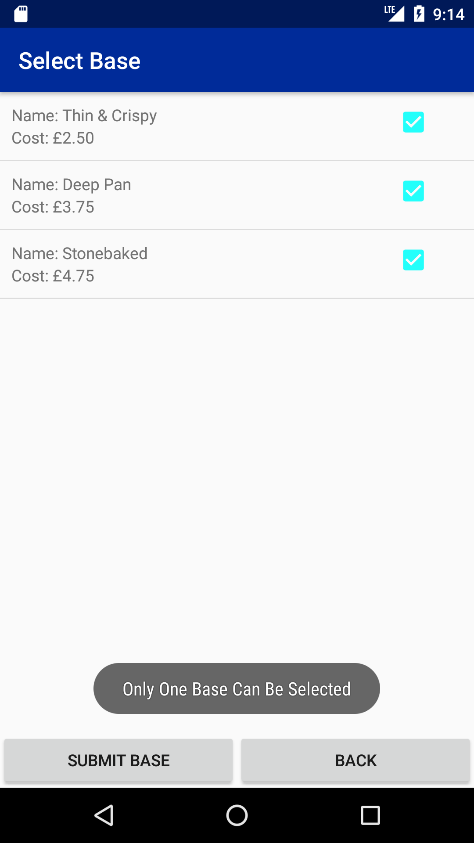
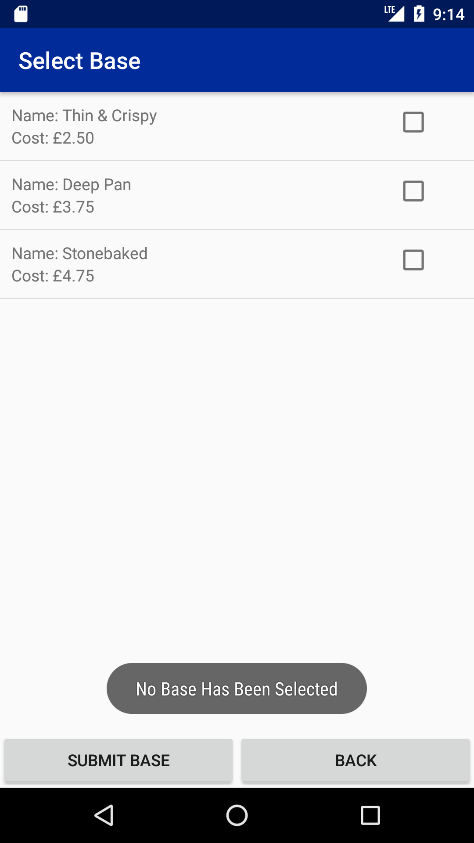
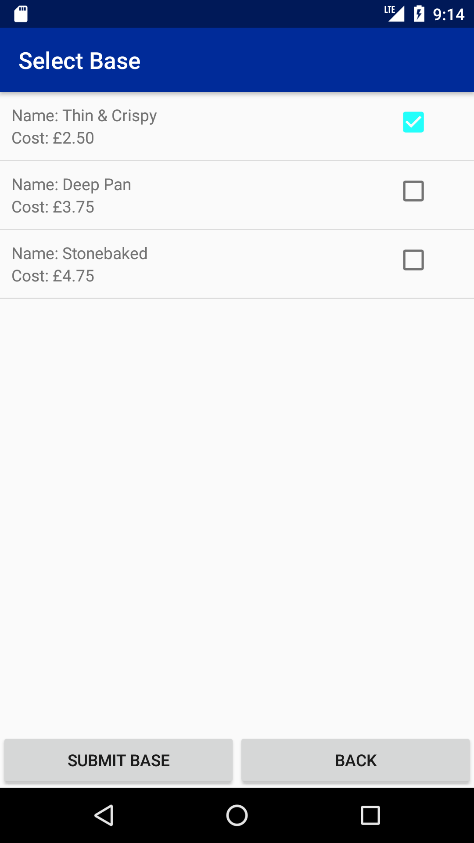
The final unit test for control panel is the test to check if the ability to edit the profile works correctly. This test changes the user’s password and sees if the changes have occurred in the database. The test ran successfully.

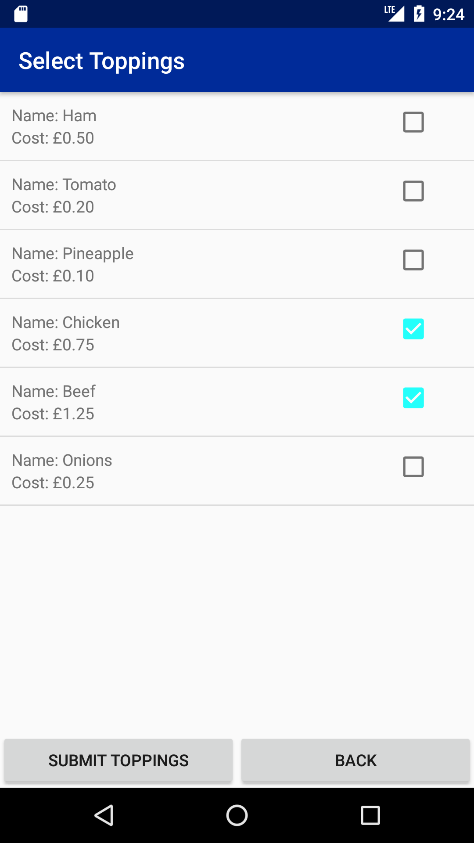
The final 2 features of the control panel are used to display data such as order history and the sales statistics of the machine. No testing is required for these as the user does not have many interactions with these screens except for reading data.



All the tests for the control panel component of the application were successful which means that both the suppliers and the users can use the control panel to manage their accounts and the machine successfully.

The last component of the vending machine to test is the dispenser. To test the dispense, screenshots will demonstrate the dispenser in use.

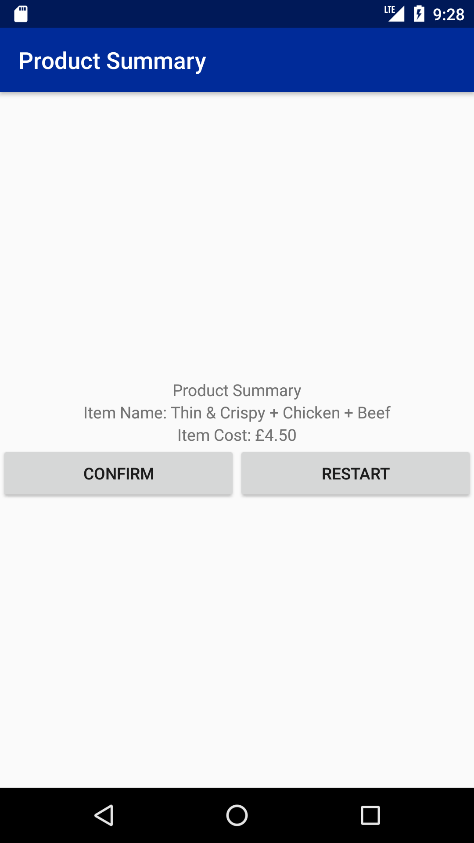


The first state of the dispenser is the select base state which can be seen in the screenshots above. It is not possible in this state to proceed without a base or with more than one base. If anything other than one base is selected, an error message will be displayed. Because of the use of a GUI, it is not possible for the user to access any invalid transitions in the state diagram. Click on submit base will proceed to the next state.

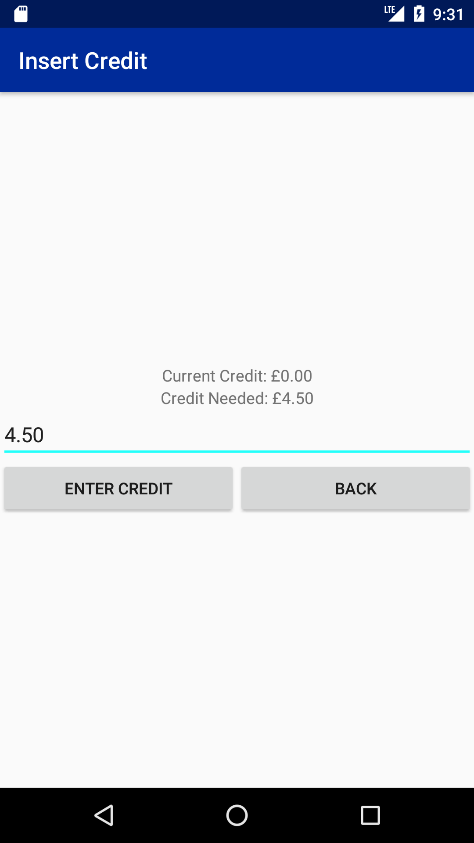
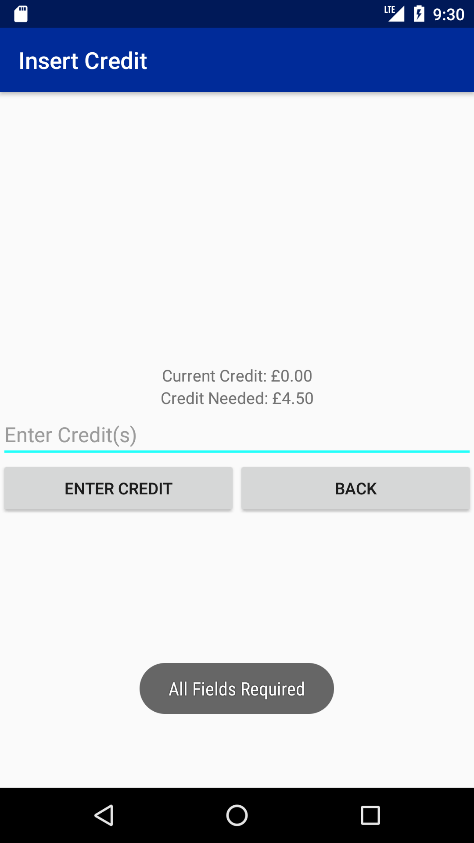
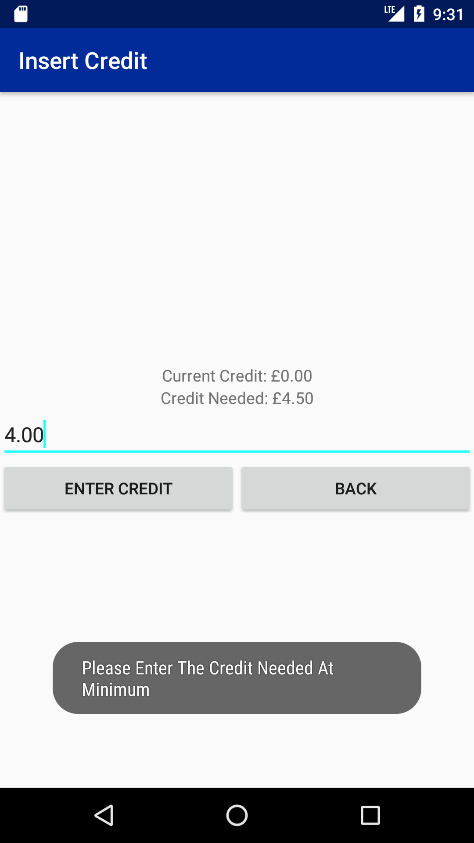
The select topping state allows the user to add toppings to their product. No input verification is needed here as the user can choose as many toppings or no toppings at all. As with the base selection, only items that are in stock will be displayed in the list. Once the user has selected their toppings the state machine then moves to the find recommendations state which is the only state not to have a GUI associated with it. If recommendations are found, the state is changed to the select recommendations state, otherwise the state is changed to the product summary state.

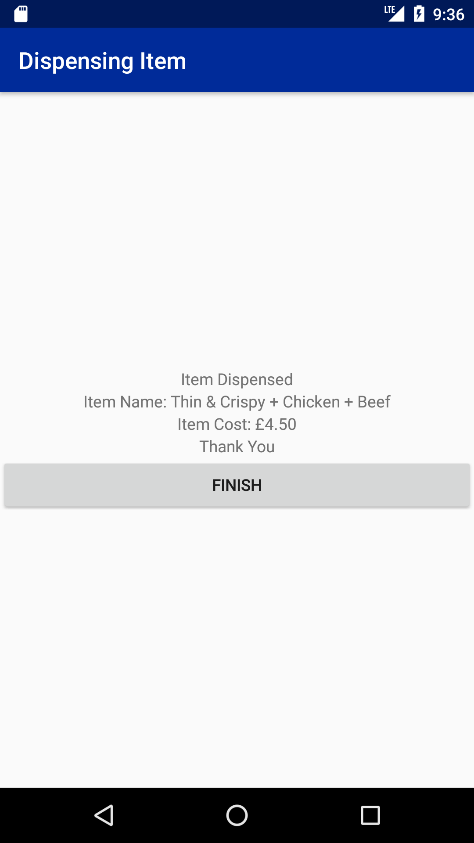


The select recommendation state is like the select topping state in the way that the user does not need to select any items to continue so no user verification is required.



At the product summary state, the user has the choice to either confirm the product that they have built so far or to start over. If the user chooses to confirm the product, they are taken to the insert credit state, otherwise the dispenser is reset, and the user is taken back to the select base state.



The insert credit state is where the user must enter the credit for their product if they don’t have enough credit on their account. The text field on this screen has user verification to make sure that they correct amount is entered and that the field is filled in as required.

The final state of the dispenser is the dispensing state. Here a message is displayed saying the item has been dispensed along with the name and cost of the item. Clicking the finish button here takes the user back to the main menu.

The 3 main components of the application all work as intended as noticed during testing. The use of unit testing mixed with GUI tests and demonstrations cover a wide range of use cases. From these tests, the behaviour of the application is stable with no major bugs. However, in the hands of the more users, even more scenarios might come up that were not thought of during testing.

# Conclusion & Future Work

The research conducted during the beginning of this project have aided the design and creation of a fully functional smart vending machine engine with an android application for a user interface and a few smart features such as a recommendation system, inventory management system and sales statistics.

The smart vending machine has been designed using various design patterns that make the program more efficient and more modular. The smart vending machine engine can run as a standalone program and is able to be used with a variety of interfaces such as a PC application, web application etc due to the engine’s ability to be plugged into another interface with very few modifications to the code base.

The smart vending machine application produced in this project brings together all the features from previously discussed solutions produced by Intel, Slikron etc, along with a collaborative based recommendation system.

The application produced could be able to use mobile payment systems if the code was implemented within the application to allow such transactions to occur. If mobile payments were integrated within the application, then it would become even more convenient for the user and the supplier.

Another feature that could be implemented into the application would be the introduction of an online based database solution which would replace the SQLite database currently used. By using an online database, the user accounts could be carried over to different machine running the same software and the sales statistics and inventory management systems could be accessed by the suppliers remotely.

If the smart vending machine was to feature online servers for storing data and providing remote access, the recommendation system could be taken online into the cloud so that a more advanced recommendation algorithm could be run that would be too computational heavy for an android device. The recommendation system could be replaced with one that allows context for each item so that the algorithm can recommend items with similar context and not just recommending items based on what other people have previously bought.

More customisation options could be added to the dispenser so that the supplier can set which toppings can be added to which bases. This would be a useful feature so that the machine can stock all kinds of toppings, but it would know which toppings can be used with each base.

In conclusion, the project demonstrates the use of new mobile technology and orient oriented programming design patterns to create a new concept of a dynamic smart vending machine that can be used with any customisable product that the supplier decides to stock in the machine providing that the machine hardware can handle the products. There would have to be steps taken such as online servers for data storage and machine management and an upgrade of the recommendation system for the application to become a more useful commercial product.

# References

Covington, P., Adams, J. and Sargin, E. (2016) ‘Deep Neural Networks for YouTube Recommendation’, 10th ACM Conference on Recommendation Systems, Boston Massachusetts, 15-19 September. Google [Online]. Available at: https://ai.google/research/pubs/pub45530 (Accessed: 14 November 2018).

Field, M. (2018) ‘Microsoft Office collects email data in breach of GDPR, regulator rules’, The Telegraph, 15 November [Online]. Available at: https://www.telegraph.co.uk/technology/2018/11/15/microsoft-office-collects-email-data-breach-gdpr-regulator-rules/ (Assessed: 17 November 2018).

Fuhrmeister, C. (2015) Finally, a Vending Machine That Dispenses Hot French Fries. Available at: https://www.eater.com/2015/9/3/9258355/french-fry-vending-machine-netherlands (Accessed: 14 November 2018).

Great Britain. Data Protection Act 2018: Elizabeth II. Chapter 12 (2018) London: The Stationary Office.

Intel (2016) IVA and Intel | Building Smart Vending Machines. Available at: https://www.intel.co.uk/content/www/uk/en/retail/documents/building-smart-vending-into-heart-of-machines-brief.html (Accessed: 7 October 2018).

Isinkaye, F., Folajimi, Y. and Ojokoh, B. (2015) ‘Recommendation systems: Principles, methods and evalution’, Egypyion Informatics Journal, 16(3), pp. 261-273. Science Direct [Online] . Available at: https://www.sciencedirect.com/science/article/pii/S1110866515000341 (Accessed: 15 November 2018).

Kim, K., Park, D., Bang, H., Hong, G. and Jin, S. (2014) ‘Smart Coffee Vending Machine Using Sensor and Actuator Networks’, IEEE International Conference on Consumer Electronics, Las Vegas USA, 10-13 January. IEEE [Online]. Available at: https://ieeexplore-ieee-org.ergo.southwales.ac.uk/stamp/stamp.jsp?tp=&arnumber=6775913 (Accessed: 7 October 2018).

Schafer, J., Konstan, J. and Riedl, J. (1999) ‘Recommender systems in e-commerce’ 1st ACM conference on Electronic commerce, Denver Colorado, 3-5 November. New York: ACM [Online]. Available at: https://dl.acm.org/citation.cfm?id=337035 (Accessed: 15 November 2018).

Sharma, P. (2018) ‘Comprehensive Guide to build a Recommendation Engine from scratch (in Python)’, Analytics Vidhya, 21 June. Available at: https://www.analyticsvidhya.com/blog/2018/06/comprehensive-guide-recommendation-engine-python/ (Accessed: 15 November 2018).

Shoot, B (2014) ‘The Smart Vending Machine’, Fortune, 170(7), pp. 49.

Silkron http://www.silkron.com/ (no date) (Accessed: 7 October 2018).

StatCounter (2018) *Mobile Operating System Market Share Worldwide*. Available at: <http://gs.statcounter.com/os-market-share/mobile/worldwide> (Accessed 30 January 2019).

Zhang, J., Lin, Z., Xiao, B. and Zhang, C. (2009) ‘An optimized item-based collaborative filtering recommendation algorithm’, 2009 IEEE International Conference on Network Infrastructure and Digital Content, Beijing China, 6-8 November. IEEE [Online]. Available at: https://ieeexplore.ieee.org/abstract/document/5360986/ (Accessed: 15 November 2018).

# Appendix

## Appendix 1 - Ethics Checklist

This form is only applicable for assessed exercises that use other people (‘participants’) for the collection of information, typically in getting comments about a system or a system design, or getting information about how a system could be used, or evaluating a working system.

If your proposed activity does not comply with any one or more of the points below then please contact your project supervisor and/or project coordinator for advice. If your evaluation does comply with all the points below, please sign this form and submit it with your assessed work.

Participants were not exposed to any risks greater than those encountered in their normal working life. Investigators have a responsibility to protect participants from physical and mental harm during the investigation. The risk of harm must be no greater than in ordinary life. Areas of potential risk that require ethical approval include, but are not limited to, investigations that occur outside usual laboratory areas, or that require participant mobility (e.g. walking, running, use of public transport), unusual or repetitive activity or movement, that use sensory deprivation (e.g. ear plugs or blindfolds), bright or flashing lights, loud or disorienting noises, smell, taste, vibration, or force feedback.

The experimental materials were paper-based, or comprised software running on standard hardware. Participants should not be exposed to any risks associated with the use of non-standard equipment: anything other than pen-and-paper, standard PCs, mobile phones and PDAs.

All participants explicitly stated that they agreed to take part, and that their data could be used in the project. If the results of the evaluation are likely to be used beyond the term of the project (for example, the software is to be deployed, or the data is to be published), then signed consent is necessary. A separate consent form should be signed by each participant. Otherwise, verbal consent is sufficient, and should be explicitly requested in the introductory script.

No incentives were offered to the participants. The payment of participants must not be used to induce them to risk harm beyond that which they risk without payment in their normal lifestyle.

No information about the evaluation or materials was intentionally withheld from the participants. Withholding information or misleading participants is unacceptable if participants are likely to object or show unease when debriefed.

No participant was under the age of 16. Parental consent is required for participants under the age of 16.

No participant has an impairment that may limit their understanding or communication. Additional consent is required for participants with impairments.

Neither I nor my supervisor is in a position of authority or influence over any of the participants. A position of authority or influence over any participant must not be allowed to pressurise participants to take part in, or remain in, any experiment.

All participants were informed that they could withdraw at any time. All participants have the right to withdraw at any time during the investigation. They should be told this in the introductory script.

All participants have been informed of my contact details. All participants must be able to contact the investigator after the investigation. They should be given the details of both student and module co-ordinator or supervisor as part of the debriefing.

The evaluation was discussed with all the participants at the end of the session, and all participants had the opportunity to ask questions. The student must provide the participants with sufficient information in the debriefing to enable them to understand the nature of the investigation.

All the data collected from the participants is stored in an anonymous form. All participant data (hard-copy and soft-copy) should be stored securely, and in anonymous form.

Student Name: Jonathan Phipps

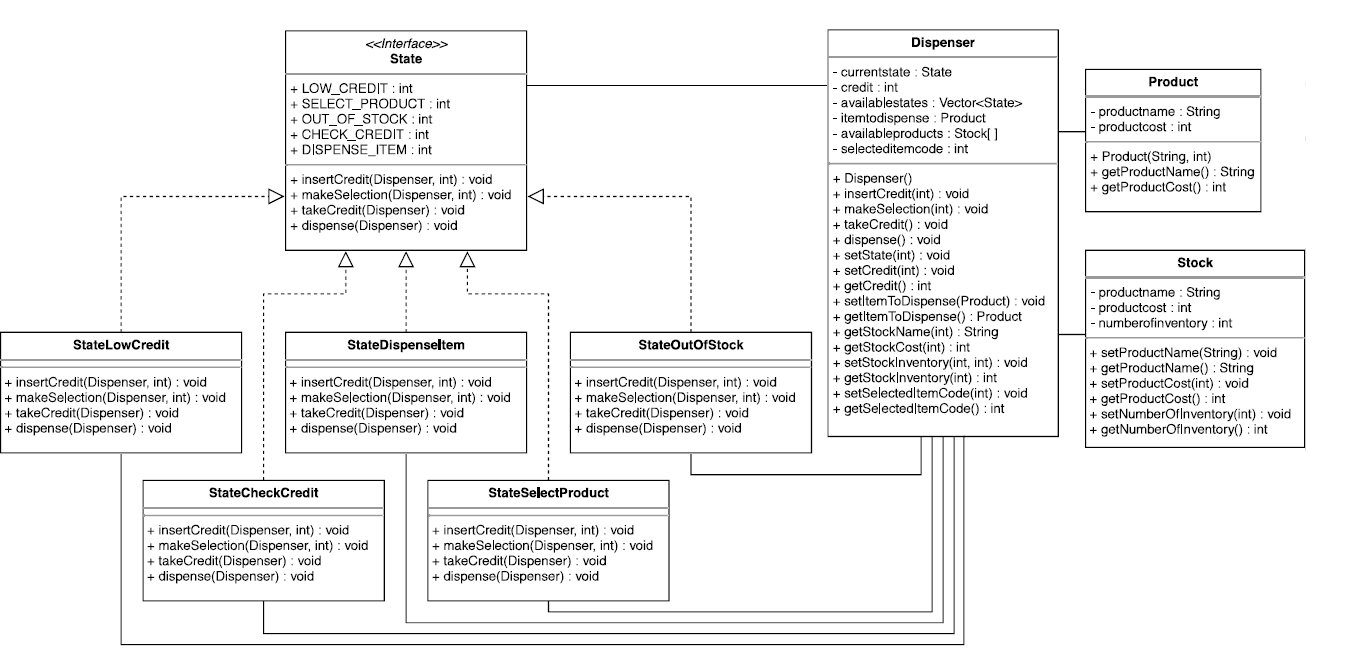
Student ID: 14729831

Student’s Signature: 

Date: 18 November 2018

## Appendix 2 - Prototype 1 State Diagram

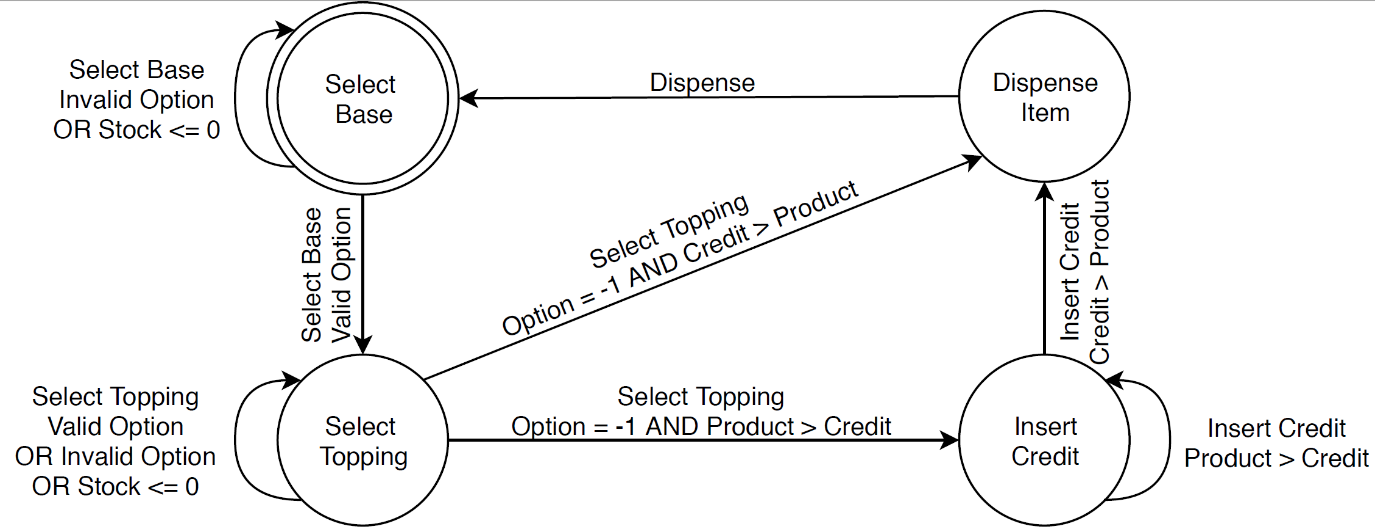
## Appendix 3 - Prototype 1 UML Diagram



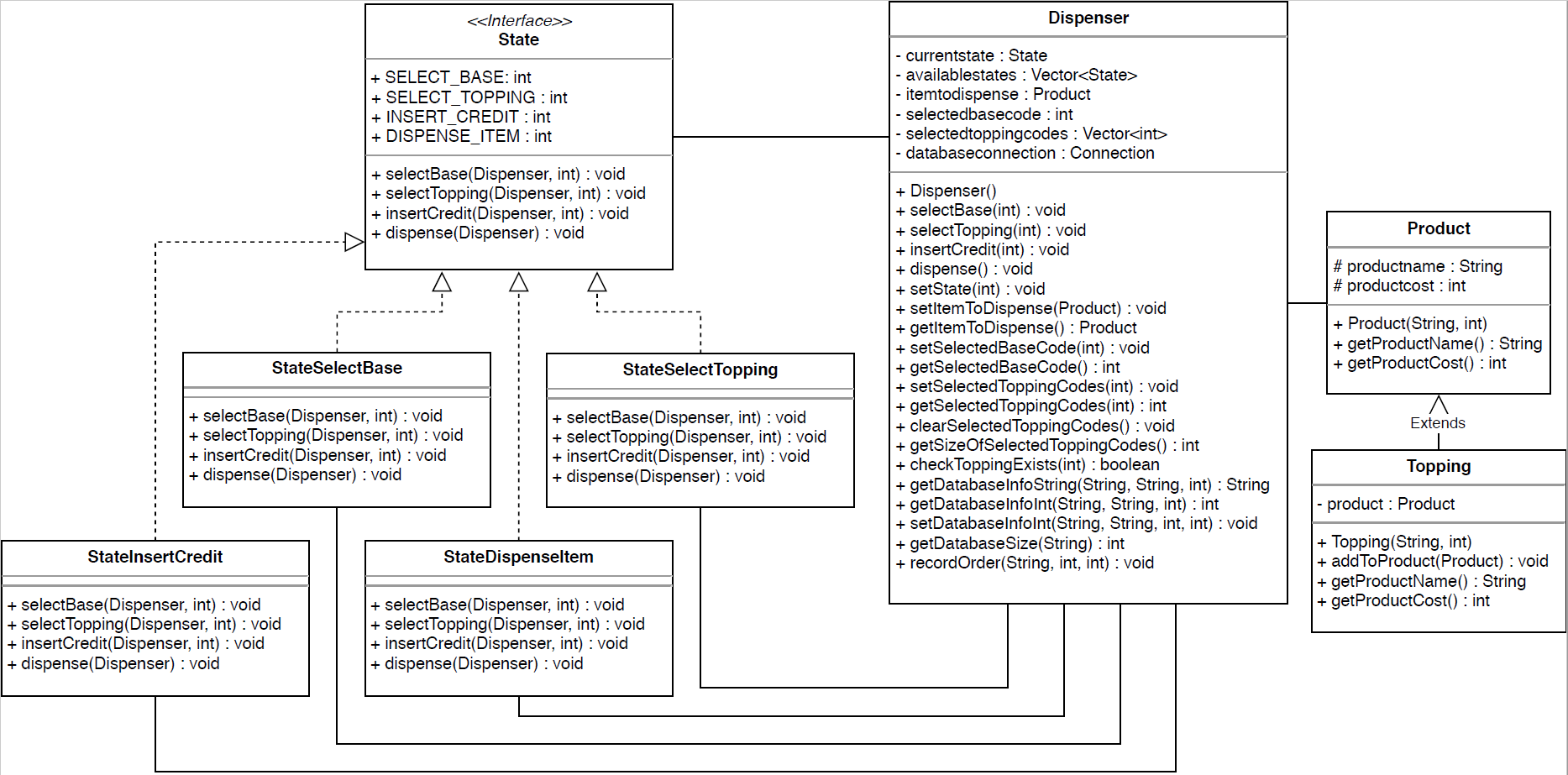
## Appendix 4 - Prototype 2 State Diagram

## Appendix 5 - Prototype 2 UML Diagram

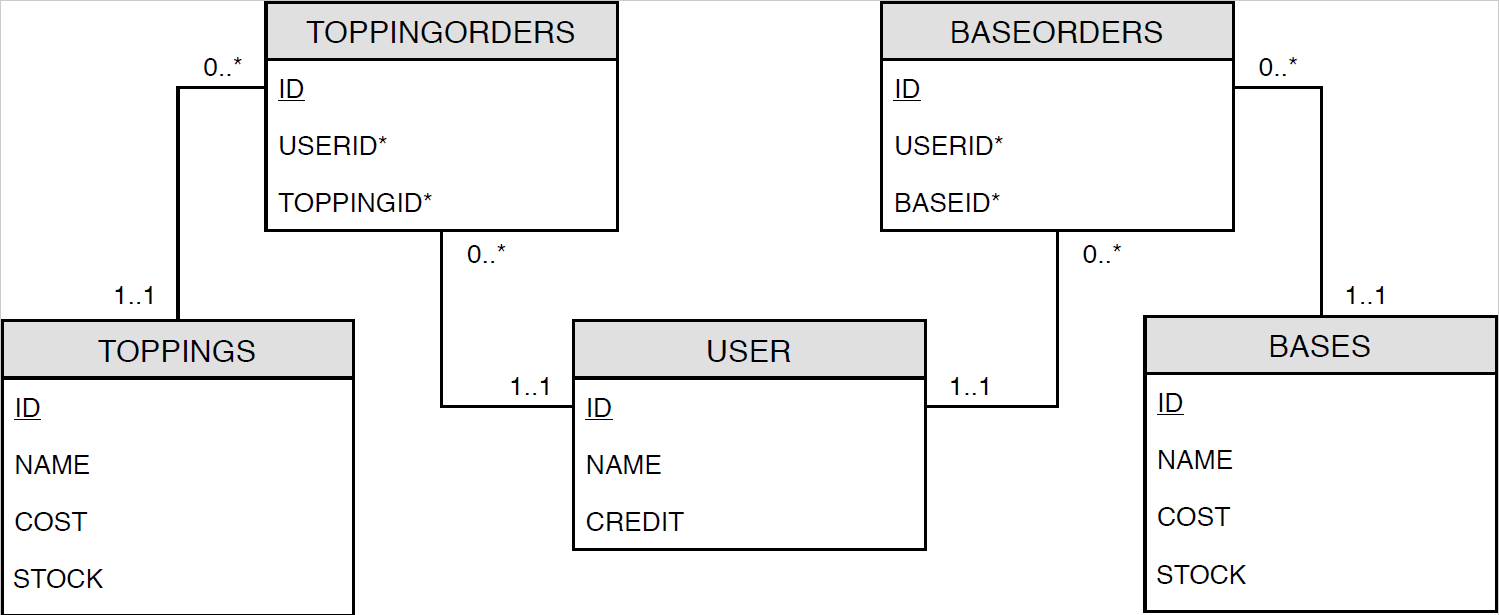
## Appendix 6 - Prototype 3 State Diagram



## Appendix 7 - Prototype 3 UML Diagram



## Appendix 8 - Prototype 3 Database Diagram



## Appendix 9 - Prototype 4 UML Diagram

## Appendix 10 - Prototype 4 Database Diagram

## Appendix 11 - Android Application State Diagram

## Appendix 12 - Android Application Main Classes UML Diagram

## Appendix 13 - Android Application GUI Classes UML Diagram

## Appendix 14 - Android Application Database Diagram

## Appendix 15 - Gantt Chart

