Cafeteria Management System

IT PAT Specs and Design Document

Name: Karina Krishnaswamy

Grade: 11

Contents Page

1.1. Problem summary

What is the purpose of the project

The purpose of my app is to create an easier system of management of pre orders for meals at the cafeteria. By allowing students to pre order both the owners/ workers of the restaurant and customers benefit. The app avoids queues at the tuckshop at break and gives workers enough time to make lots of meals. They would also not have to consume break times collecting money from students as they pay. The owners and chefs can also see the orders needed to be made throughout the day.

Description of the target user groups.

The target users of my app are the cafeteria owners as well as admin officers who work at schools. It is designed for those who run the cafeteira in order to help them increase efficiency.

Functionality of the project.

A student who would like to order goes to the admin office and places an order from the cafeteria. The admin user then goes onto the app and adds a new order to the system. This order contains the food ordered, the details of the student's details as well as a boolean of whether the student has paid or not. If the boolean paid is true the order is sent through for making at the cafeteria.

The admin office can edit the menu by deleting or removing items or changing the value of a menu item. The app also has access to a system of students. Admin workers can add students to the system so they can order.

A view of all the orders exists. Tuckshop owners can view all the orders or all the paid ones so they know when to start making them. An order state can be changed from unpaid to paid.

The manage students screen allows students to be added and removed from the system and the managed menu allows menu items to be added and removed from the system. The place order screen allows the user to place an order. The features this screen has are the ability to view the current order and price and edit it as well as. The admin user also adds the details of the order i.e the student, time and whether the order has been paid for or not on the screen.

1.2. Motivation and Research

1.2.1. Existing solution

A similar product that has been designed is the “Tap Tuck” app. This app allows parents to pre order meals for their children. The app also allows parents to pay with their phone using a QR code, top up their kids’ wallets and add children. It too has a transaction history option and the ability to contact the cafeteria owners.

A more public app is CanteenApp. Restaurant owners can download this app for their restaurant to allow customers to pre order. An option is given for the customer to eat in or order and go. Health tips from the restaurant are also given from the restaurant in control of the app. Order and menu management is also available.

1.2.2. How my project will differ from the current available programs

My app is designed for the management of the tuckshop and its owners. It differs as an admin worker controls all cash payments that are made. Parents may order for their children at the front desk in the morning or students themselves. My app allows for owners of the cafeteria to view all the orders placed and whether they have been paid for or not. The target users are students and the app helps create a cashless cafe system. Orders placed in my app do not all have to be paid for, a student can order and pay later if they do not have any money.

1.3. Specs of program functionality

1.3.1. Home specifications

- Have buttons that direct users to other screens

1.3.2. Manage menu specifications

- Let user view the menu
- Let user add items to the menu
- Let user remove items from menu

1.3.3. Place order specifications

- Let user place an order
- Search for a menu item by its type
- Add items to the current order
- Remove items from the current order
- Add student to order
- Add time to order
- Add whether student has paid or not to order
- View current total price of order
- Return back to home screen

1.3.4. View orders specifications

- View paid orders
- View unpaid orders
- Change an unpaid order to a paid order

1.3.5. Manage students specifications

- View all students
- Add student
- Delete student

1.4. Specs of data storage

Grades
- **Data**: List of the different grades
- **When are records accessed:** When the ”manage student” screen is created in order to add a grade to a student.

Menu item types
- **Data:** List of the different types of menu items
- **When are records accessed:** When the manage menu and place order screen are created in order to view different menu options

Student
- **Fields**: name and grades
- **When are records created:** When a student is added or deleted
- **When are records accessed:** When an order is to be placed the names appear in a drop down menu. When the student management screen is selected
- **When are records updated:** when a student is added or deleted

Menu Item
- **Fields**: name, type and price
- **When are records created:** When a menu item is added or deleted
- **When are records accessed:** When the manage menu screen is selected or when the place order screen is selected
- **When are records updated:** when a menu item is added or deleted
-

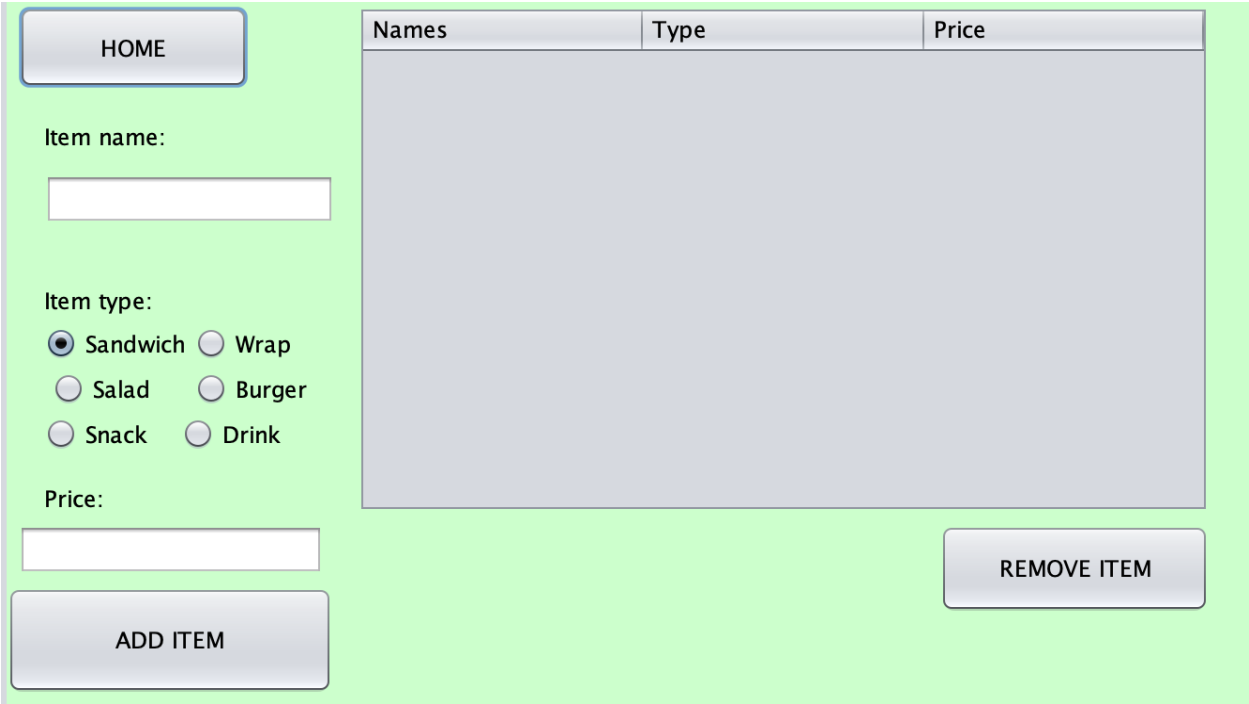
Order
- **Fields**: student name and grade, list of ordered item,
- **When are records created:** When an order is added
- **When are records accessed:** When the view order screen is selected this data populated the table
- **When are records updated:** when an order is added or when an order is changed to paid

2.1. Interface design

2.1.1. Home screen

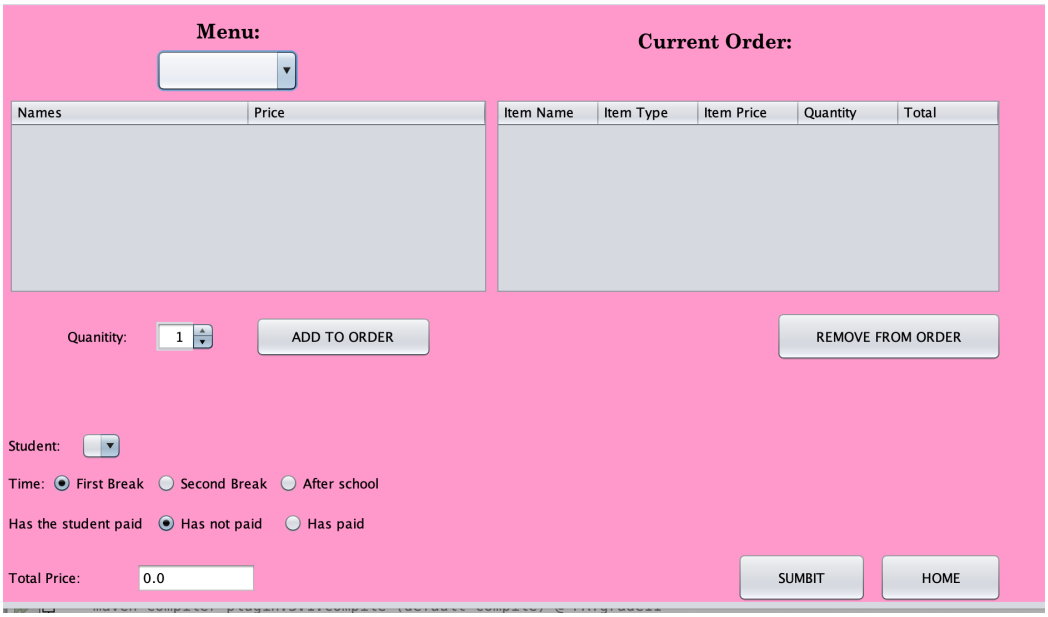
Description	The purpose of this screen is to allow the app user to choose which action for the tuckshop they would like to perform.
Data	N/A
Actions	**MANAGE MENU** The user clicks and is taken to a screen where they can make edits to the menu **CREATE ORDER** The user clicks this and is taken to the menu where they can select items made by a student **VIEW ALL ORDERS** The user clicks this and taken to a screen where they can view all the orders or just the paid ones **MANAGE STUDENTS** The user clicks this and is taken to a screen where they can add or delete a student to the system

2.1.2. Manage Menu



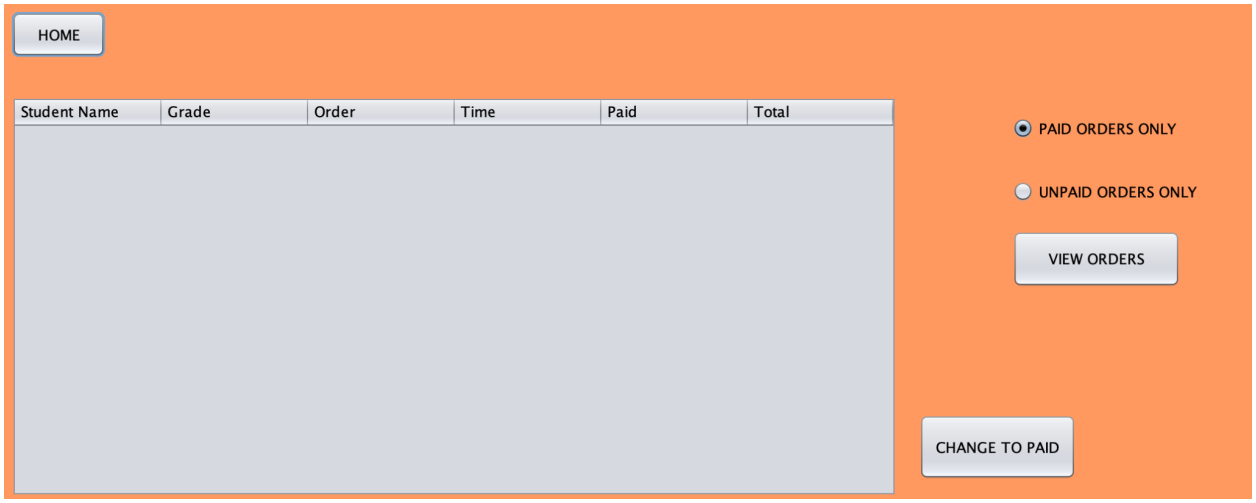
Description	The purpose of this screen is to allow admins to make edits to the menu
Data	The data in when the screen is created and when the add and remove buttons are pressed is the table data from the Menu Item text file. The data out of the screen is the fields of the new menu item to be added when the add item button is selected and the row of the selected menu item to be deleted
Actions	**ADD ITEM** Allows the user to add an item, its price and type to the menu system **REMOVE ITEM** Allows user to delete an item's data from the system **HOME** Takes the user back to the home screen

2.1.3. Place order



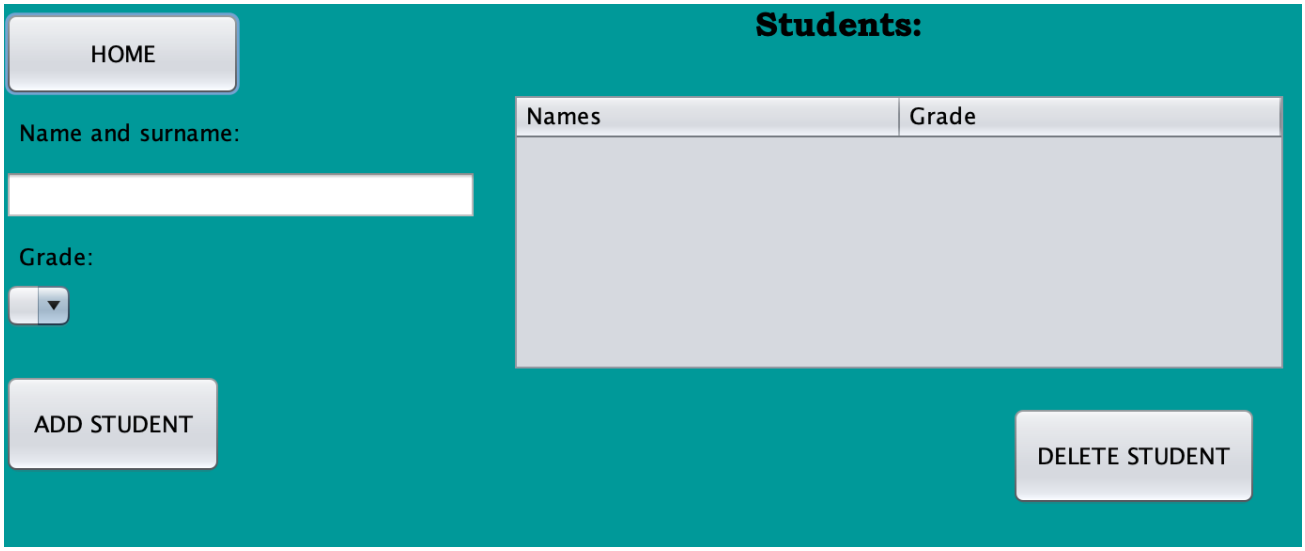
Description	The purpose of this screen is to allow users select menu items in order to create an order for a student as well as add the other fields of an order
Data	The data that is used in this screen are menu items, the current order, a student combo box to select a student, two radio groups- one for time and the other for price, the total price of the order
Actions	**ADD TO ORDER** The user is allowed to highlight items of the menu that the students would like to order and add to the current order table **REMOVE FROM ORDER** The user is allowed to highlight an item of the current order table and remove it from the current order **ADD ORDER** The user selects this button and adds the current order to the system **HOME** Takes user back to the home screen

2.1.4. View orders



Description	The purpose of this screen is to allow the user to view the orders that have been made
Data	The data in is the orders from the order text file for the table depending on whether they have been paid for or not. The data out is the selected row of the menu item that needs to be changed
Actions	**VIEW ORDERS** The user chooses an option from the radio button: either paid or unpaid order and clicks this button to view them **CHANGE TO PAID** The user highlights an unpaid order clicks on thai button to change it to paid **HOME** Takes user back to the home screen

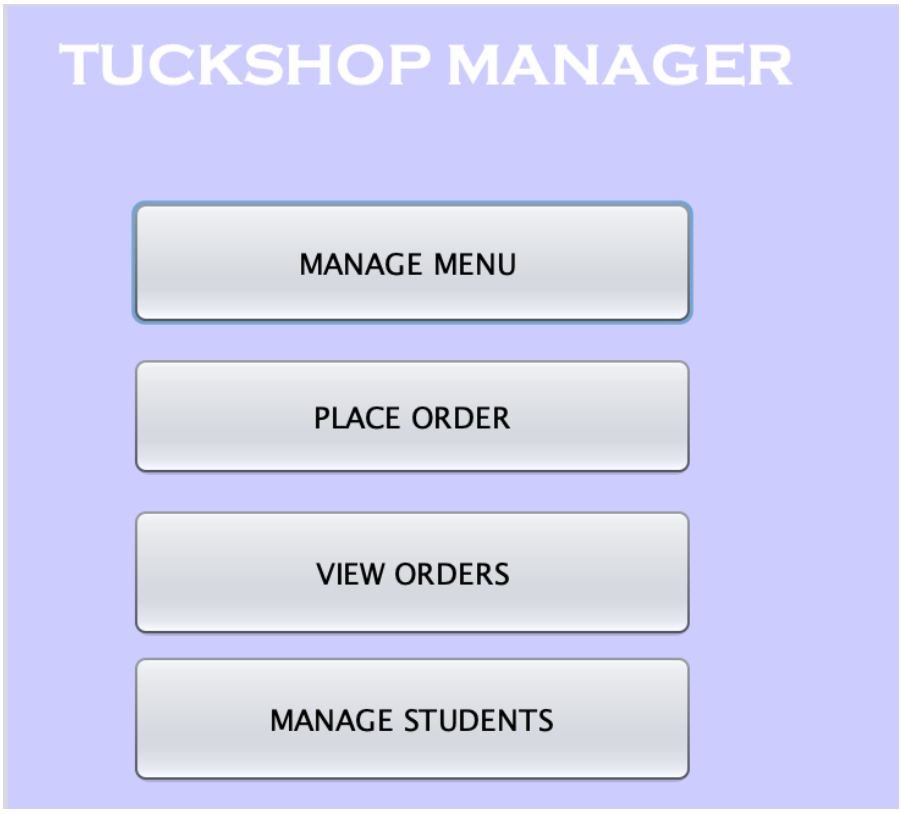
2.1.5. Manage students



Description	The purpose of this screen is to allow the user manage the students in the system
Data	The data is a table of all the students
Actions	**ADD STUDENT** The user adds the name and grade of a new student and clicks this button to add the student to the system **DELETE STUDENT** The user highlights a student in the table and clicks this button to remove them from the system **HOME** Takes user back to the home screen

2.2. Program flow

2.1.1. Home screen



- When place order is selected

  ```
  A new place order screen is created and the home
  screen is disposed
  ```

- When manage menu is selected

  ```
  A new place order screen is created and the home
  screen is disposed
  ```

- When view order is selected

  ```
  A new place order screen is created and the home
  screen is disposed
  ```

- When manage students is selected

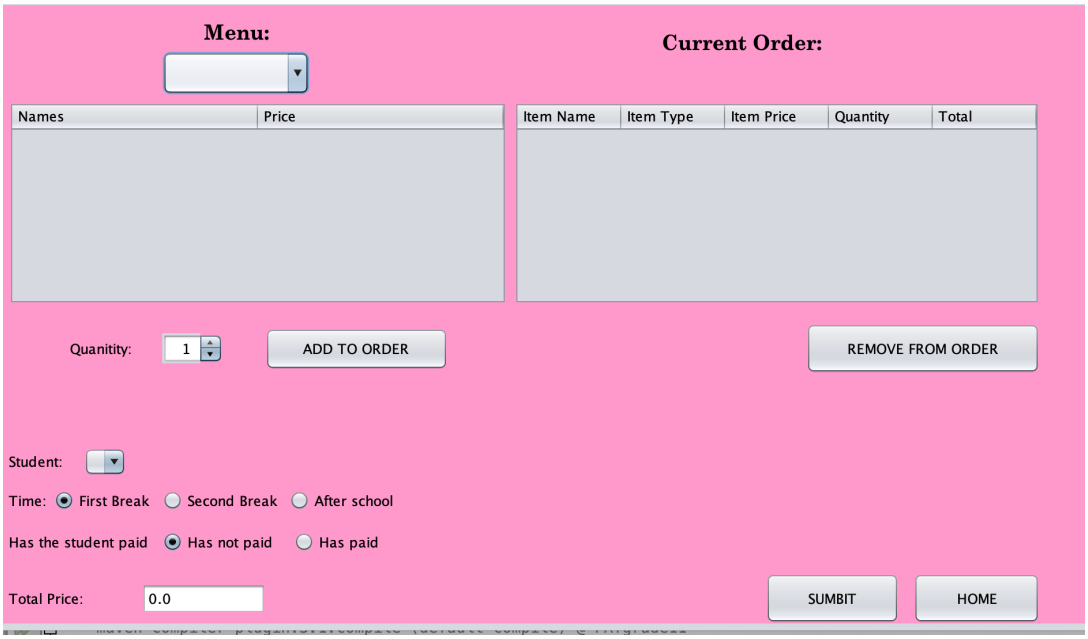
  ```
  A new manage student screen is created and the home
  screen is disposed
  ```

2.1.2. Manage menu screen

- When add item is selected
    - The text is gotten from the itemNameTextField
    - The type is selected from the action command of the radio buttons
    - The text is gotten from the priceTextField
    - The fields are used to create a new menu item object
    - A menu item array object is created
    - This new menu item is added to the menuItemArray class
    - The array is rewritten to the text file and the table is updated
- When delete item is selected
    - A new menu item is created using selected table row's fields
    - The menu item's name and type is compared to those in the array
    - The corresponding item is deleted from the array
    - The array is re written to the text file and the table is updated
- When home is selected

  A new home screen is created and the manage menu screen is disposed

2.1.2. Place order screen



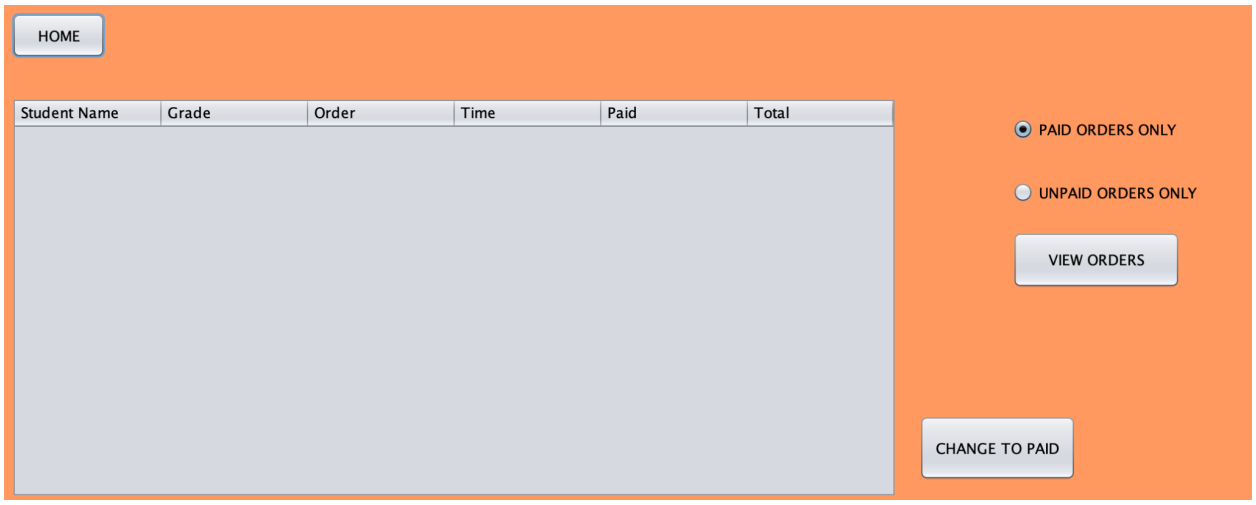
- When add to order is selected
    - Int row = menu item table's selected row
    - A new menu item object is created using the data from the row
    - An ordered item object is created and is added to the array with the quantity gotten from the quantity box and the new menu item
    - Total price = total price + new item's total price
    - Ordered table is recreated
- When remove from order is selected
    - Int row = ordered item table's selected row
    - A new menu item object is created using the data from the row
    - An ordered item object is created
    - This ordered item is compared to the array and the corresponding ordered item is removed from the array
    - Total price = total price - item's total price
    - Ordered table is recreated
- When add order is selected
    - A new order object is created
    - Order.setStudent to combobox.getStudent which compares the student name to the names of the

students in the student array object
  - Order.setOrderedItemArr to table
  - Order.set time to radio button action command
  - Order.set paid to radio button action command
  - Order.set price to text area.get text
  - An order array object is created
  - The new order is added to the order array object

- When home is selected

  A new home screen is created and the place order screen is disposed

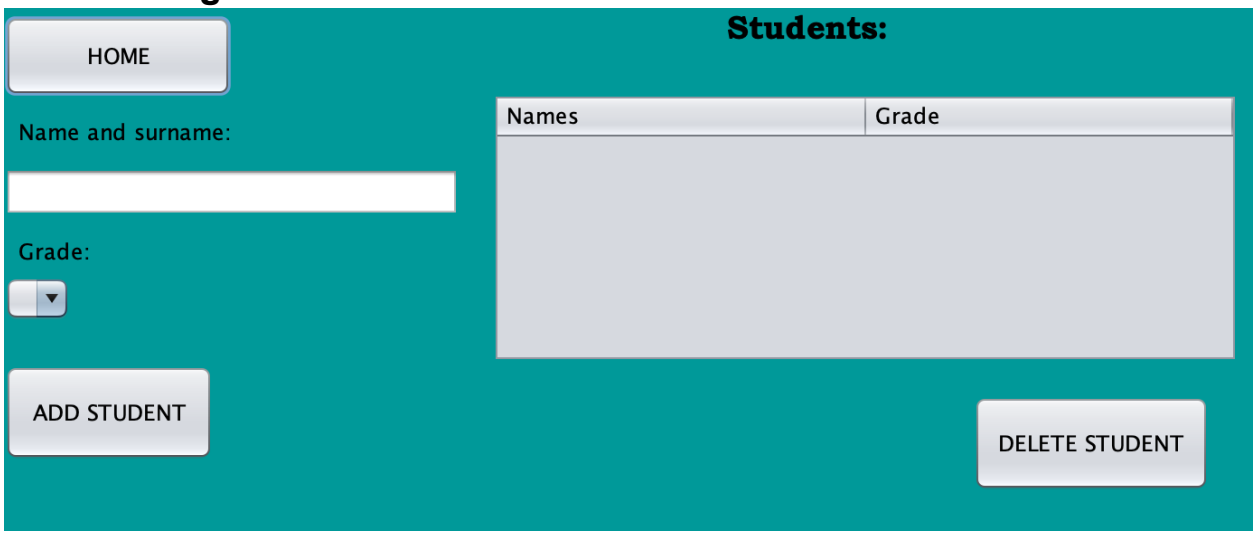
2.1.2. View orders screen



- When view orders is selected
    - Boolean paid = the action command of the radio button
    - OrderArray orders = new OrderArray
    - orders.getData(paid)
    - The selected data is populated into the table
    - Data is based on whether the boolean is true
- When change to paid is selected
    - New orderArray object is formed
    - Method compares selected highlighted order to those in the array using the ordered item array field
    - When the order is found is boolean of the order in the order is array is changed to paid
    - The table is popualted
- When home is selected

  A new home screen is created and the place order screen is disposed

2.1.2. Manage Students screen



- When add student is selected
    - `Student s = new Student object`
    - `s name = text field . get text`
    - `S grade = combo box selection`
    - `A new student array object is created`
    - `New student is added to the array`
- When delete student is selected
    - `A new student object is created using the fields of the selected row of the table`
    - `The new student is compared to those in the array`
    - `The corresponding student is deleted from the array`
    - `The table is populated`
- When home is selected

  `A new home screen is created and the place order screen is disposed`

2.3. Class Design

Student	Description
- name: string - grade: int	Stores the name and surname of student Stores the grade of the student
+ Constructor(name: string, grade: int) + getName(): string + setName(name: string) + getGrade{): int + setGrade(grade: int)	Instantiates a user object Returns Name Sets the name Returns the grade Sets the grade

StudentArray	Description
- studentArr: student[] - Size: int	Holds the array of students Holds the size of the array
+ Constructor() + addStudent(String name, int grade) - shiftLeft(int index) + deleteStudent(String name) + printArrayToFile + getStudentsnamesAsarrayForComboBox(): out + getStudentTableData(): out + getStudentForOrder(String name): student	Instantiates the student array object Adds a new student to the array given its field Shifts the array left removing student at the given index Deletes the student from array given its name Prints the current state of the current state of the array to the file Returns a string array of the name of all the students Returns a 2D object array of students Returns a student object based on the name given

GradesArray	Description
- gradesArr[]: String - size: int	Holds the grades in an array Holds the size of the array
+ Constructor() + getArr : gradesArr []	Instantiates a grades array object Returns the gradesArr as a String array

MenuItem	Description
- Name: String	Holds the name of the menu item

- Price: double - Type: String	Holds the price of the menu item Holds the type of the menu item
+ Constructor(name: String, price: double, type: String) + getName : name + getType: type + getPrice: price + fileFormat: string + setName(name: string) + setType(type: string) + setPrice(price: double)	Instantiates a menu item object Returns the name of a menu item Returns the type of a menu item Returns the price of menu item Returns the menu item in the format that it is set in the file Sets the name of an item Sets the type of an item Sets the price of an item

MenuItemArray	**Description**
- menuItemArr: menuItem[] - Size: int	Holds the array of menu item Holds the size of the array
+ Constructor() + addMenuItem(String name, double price, String type) - shiftLeft(int index) + deleteMenuItem(String name, String type) + printArrayToFile + getOutSize(String type): outSize + getMenuItemFromType(String type): out + getMenuItemForMangeMenu()	Instantiates a menu item array object Adds a new menu item to the array given its field Shifts the array left removing item at the given index Deletes the menu item from array given its name and type Prints the current state of the current state of the array to the file Returns the number of menu items of a specific type from the array Returns a 2D object array of the menu items of a specific type Returns a 2D object array of all the menu items

Order	**Description**
- S: Student - orderedItemArr: orderedItemArray - Time: int - Paid: boolean - totalPrice: double	Holds the student who the order is for Holds the array of ordered items Holds the time of the order (i.e 0 = first break, 1 = second beak and 2 = equals school) Holds whether the order has been paid for or not Holds the price of the order
+ Constructor(s: Student, orderedItemArr: ordereditemArray, time: int, paid: boolean, totalPrice: double) + fileFormat: String	Instantiates a order object Returns the order in the format it has to be

	printed to the file
+ setStudent(String s)	Sets the student
+ getStudent: s	Returns the student object
+ setOrderedItemArr(OrderedItemArray: orderedItemArr)	Sets the ordered item array
+ getOrderedItemmArr: orderedItemArr	Returns the orderedItemArray object
+ setTime(int time)	Sets the time
+ getTime: int time	Returns the time
+ isPaid: boolean paid	Returns whether the order has been paid for or not
+ setPaid(boolean paid)	Sets whether the order has been paid for or not
+ setTotalPrice(double totalPrice)	Set the price
+ getTotalPrice: totalPrice	Returns the price

OrderArray	**Description**
- ordersArr: order[]	Holds the array of orders
- Size: int	Holds the size of the array
+ Constructor()	Instantiates an order array object
+ fileFormat: String	Returns the order in the format it has to be printed to the file
+ setStudent(String s)	Sets the student
+ getStudent: s	Returns the student object
+ setOrderedItemArr(OrderedItemArray: orderedItemArr)	Sets the ordered item array
+ getOrderedItemmArr: orderedItemArr	Returns the orderedItemArray object
+ setTime(int time)	Sets the time
+ getTime: int time	Returns the time
+ isPaid: boolean paid	Returns whether the order has been paid for or not
+ setPaid(boolean paid)	Sets whether the order has been paid for or not
+ setTotalPrice(double totalPrice)	Set the price
+ getTotalPrice: totalPrice	Returns the price

OrderedItem extends **MenuItem**	**Description**
- Quantity: int	Holds the quantity of the menu item ordered
+ Constructor(quantity: int, menuItem: menuItem)	Instantiates an ordered item object
+ getQuantity: quantity	Returns the quantity of the item
+ setQuantity(int quantity)	Sets the quantity of an item
+ getTotalPrice: totalPrice	Calculates the total price spent on that item by multiplying cost and quantity

OrderedItemArray	**Description**
- orderedItemArr: orderedItem[] - Size: int	Holds array of ordered items Holds the size of the array
+ Constructor() + add(MenuItem menuItem, int quantity) - shiftLeft + delete(MenuItem menItem) + getTotalPrice: totalPrice + fileFormat: out + getSize: size + setSize(int size) + getArr: orderedItemArr + setArr(OrderedItem[] orderedItemArr) + toString: out + getOrdredItemsData: out	Instantiates an ordered array object Adds a new ordered item to the array Shifts the array left to remove an item Removes the ordered item from the array Returns a double of the total price of the ordered items in the array Returns a String formatted for a file of the items in the array Returns size of the array Sets the size of the array Returns the array of ordered items Sets the array of ordered items Returns a string of the way the array is to be displayed in a table Returns a 2D object array of the data needed for the current order tabel

2.4. Secondary storage design

Permanent storage that will be used is text files

Types of items.txt(text file)

```
Type 1
Type 2
```

```
Sandwich
Drink
Warp
```

Grades.txt(text file)

```
Grade 1
Grade 2
```

```
1
2
3
4
5
```

Students.txt (text file)

```
<name>@<grade>
```

```
Karina Krishnaswamy@11
Ronaq sayed@10
```

MenuItem.txt (text file)

```
<name>@<price>@<type>
```

```
Chicken and mayo@12.0@Sandwich
Chicken@20.0@Salad
Diet coke@10.0@Drink
```

Order.txt (text file)

```
<Student name>@<student grade>
<ordered item 1 name>@<ordered item 1 type>@<ordered item 1 price>@<ordered item 1quantity>#<ordered item 2 name>...
<Time>
<Paid>
<Total price>
```

```
Cole Smith@7
Avocado and chicken@Wrap@30.0@2
1
true
60.0
Ryan Griffiths@5
Chicken@Burger@30.0@2#Beetroot and feta@Salad@30.0@2
0
true
120.0
Karina Krishnaswamy@11
Future life energy bar@Snack@16.0@2
1
True
32.00
```