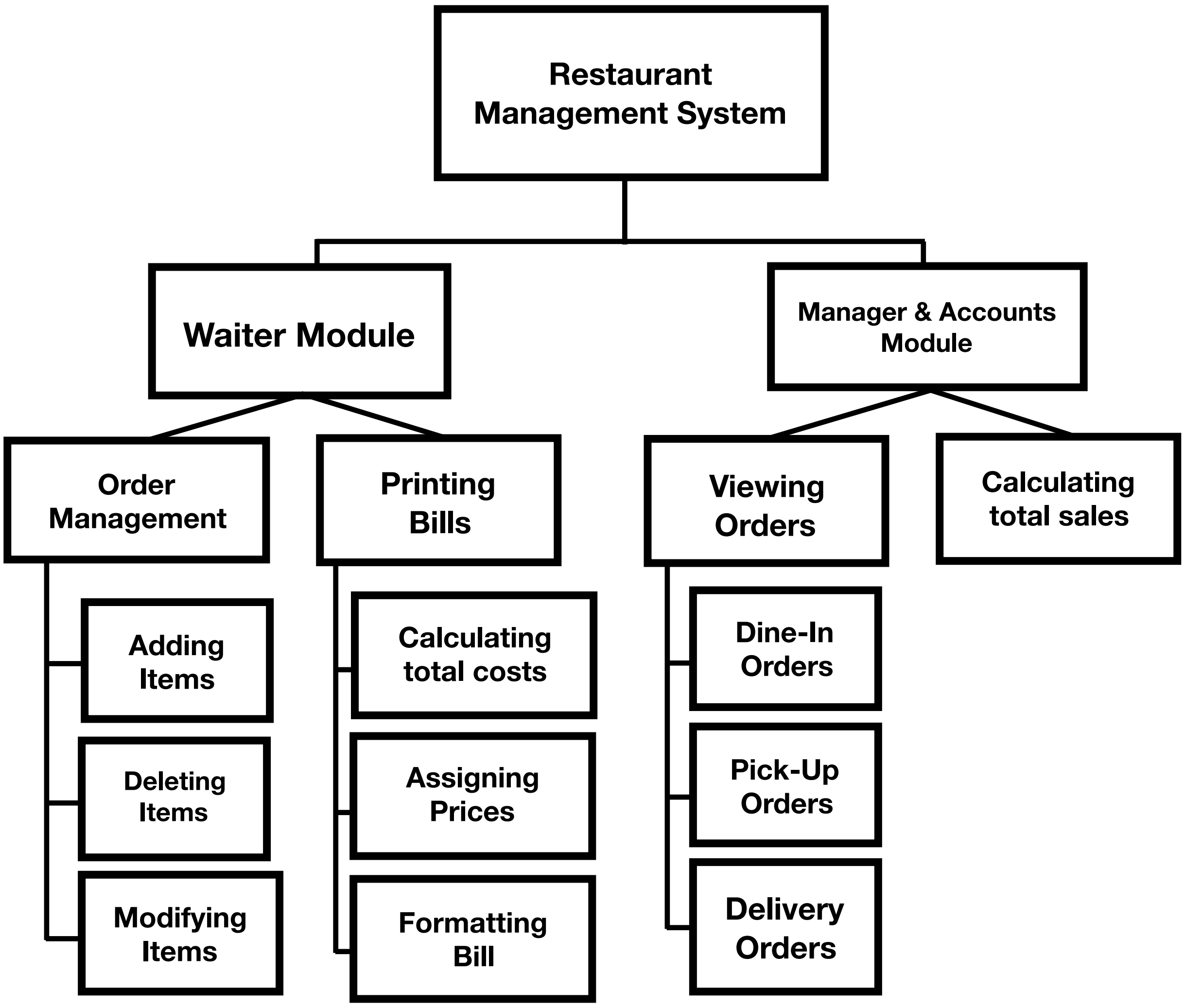
Criterion B: Design

With regards to the design of the proposed product, I designed multiple prototypes of the design and discussed the appropriateness of such with my client. After identifying a suitable prototype, and ensuring that it meets the client's requirements, I began to design the operations and overall structure of the product.

The prototype design I would be going forth with is a program that involves two sections, one for a waiter to take orders, and another for a manager to view and manage said orders. Below is given a top-down structure diagram for the proposed product that I used to explain the product to my client :



Data Input Structure

Input	**Comments**
Waiter navigates through menu to desired option. *Example* 1.Add Item 2. Delete Item 3. Check Menu	The overall main menu is divided into multiple sub-menus, and consists of an overall "tree" structure of menus. There should be fully functional menus for each case, such as delivery, dine in, or pickup orders.
The waiter should be able to enter the code of a dish, and a bill for the customer should automatically be calculated and printed at the completion of their order.	The dynamic queue or linked list should have a function that traverses the list and prints a customer's bill accordingly.
The manager may choose to add an item to the menu, or modify the price of an item already in the menu.	The menu of the restaurant will be stored in a text file that has a ordered format of records such as "<<Item Code>>|<<Item Unit Price>>", where '|' or another suitable character would act as a delimiter.
All of the previous orders should be stored and upon startup they must return to their original state, unless the manager specifies otherwise.	All of the orders will be stored in organised lists and queues of objects that will be stored into a serialised text file. Only upon the execution of a "reset list" function do the orders reset for a new day in business.

Required Outputs :

- The printing of an individual bill for every customer along with the total price (inclusive of tax).
- The printing of a list of the bills for the current session, that the manager will be able to view.

Structuring Plan for the product

As my program will be used by multiple people, I will have a sectional division of my product, and provide each classification of individual access to their own model.

Function	**Comments**
Waiter Segment	
A main menu Enter Customer Name : 1.Dine-In 2. Pick-Up 3. Delivery	Before the client proceeds, he must enter the customer's name . After choosing the appropriate option that the current order falls into, the client will be prompted to enter further information, such as a table number for Dine In orders, or a Delivery address for delivery orders.
Within each module, the available functions are : 1. Add Item 2. Cancel Item 3. Print Menu 4. Print Bill 5. Complete Order and Exit	1. Will prompt the user to enter a dish's code and its quantity to add to the bill. 2. Will allow the user to delete an item from the current order. 3. Will print out the menu along with the price to the user 4. Will print the current bill of the customer 5. Will complete the current order and print the bill so that waiters can move to the next order.
Manager Segment	
Main Menu 1. View/Modify Orders 2. Calculate Total Earnings for the day 3. Modify Menu	1. Will allow the manager to view and modify the current list of orders for the day. 2. Will calculate total sales of the restaurant so far, to make calculations easier. 3. Will branch into a submenu that allows the manager to modify items in a menu.

<u>Test Plan :</u>

Test Type	**Nature of the Test**	**Example**
Upon Starting up the program, the main menu should be displayed.	To check if the conditions of the menu work	“Hello, Welcome to AVS Restaurant, Choose one of the options below” “1. Add Order”
The menu should continuously loop until a valid option is entered	Checking if the validity checks work.	“1.Add Order” *User enters 900* “INVALID OPTION ENTER AGAIN” “1. Add Order”
Check if the print menu method works	To ensure that the file access method works	“3. Print Menu” “<<menuitem>> <<price>>”
Check if the delete order method works	To check if the code works effectively as it should	*User tells the product to delete item 1 from the order* “Item 1 has been deleted” *Prints the rest of the order after deletion.*
Checking if the orders are stored permanently even after restarting the application	Ensuring the permanent file storage works.	*After closing the application and reopening it, orders from the previous session should be saved and user should be able to modify them as they please.*