Software Design: GotoGro-MRM

Team Details

Team Name:	MSP 14
Tutorial:	Tue 2:30 ATC325
Tutor:	Dr Kaberi Naznin

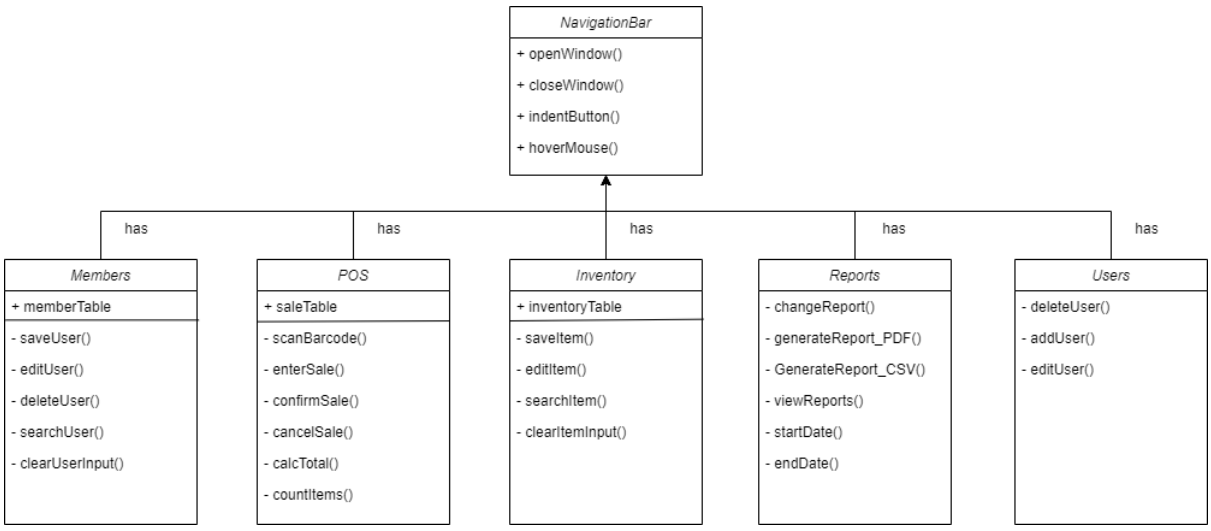
Members:	
Dylan Jarvis	102093138
Rabya Tayal	103144215
Simon Tran	103602807
Thomas Babicka	103059885
Cody Cronin-Sporys	103610020
Nicholas Dyt	101624265

Software Design

The diagram below displays how each software component interacts with another, this includes items from sprint one. The design is partially object-oriented and only shares a common navigation bar and a common MySQL database. Each function only exists within its specific page. Currently, there is no need to have further interaction as each page has a separate role in the overall operation of the software solution.

Sprint two focused majorly on the reporting module and as such the overall design has not changed majorly from task 14P. Of the six items within sprint two, four are relating to the reporting module while the other relates to the UI of the project and adding user authentication.

Diagram 1. GotoGro - UML



Software Design Principles

Table 1. Software Design Principles

Principle	Description
Modularity	The measure of how well the software is broken down into individual components. The components must be separately maintainable from other components and cannot affect others.
High Cohesion & Low Coupling	**High Cohesion** refers to how a single class's function relates to one another. **Low Coupling** refers to classes having the least number of dependencies on other classes.
DRY (Don't Repeat Yourself)	All pieces of code are not repeated in other elements. For example, a function/definition is only defined once. This helps make maintainable and scalable coding.
MVC Design Pattern	Model View Controller (MVC) is a design principle that breaks down each functional component of an application into three separate components. These three components describe an application's interaction between a user and the backend. A User uses the Controller which in turn manipulates the Model and updates the View that is seen by the User. This is essentially a feedback loop for user input and data manipulation.

Table 2. Justification of Design Principles

Principle	Justification
Modularity	The reporting module is separate from any other features and even the UI and functions listed in the UML diagram. The reporting module is a standalone reporting solution that pulls data from a variety of different sources and has the tools to present these in any desired format. This can operate without any of the UI or manual user input from the UI. The UI buttons merely call for report generation and the reports themselves can be altered without affecting any of the functions listed in the UML.
High Cohesion & Low Coupling	While the reporting module is a separate entity the backbone of the reports is the same for both the CSV and the PDFs. They use the same SQL queries, and they are the same reports just presented differently. The two reports themselves operated independently from each other and have different datasets. High cohesion is shown by how the same reports use the same backbone data acquisition. Whereas low coupling is demonstrated by how independent the reports are from other software structures.
DRY (Don't Repeat Yourself)	All new code written in sprint two is unique from the sprint one items, and the majority of the changes made were through UI and by definition cannot be repeated code.
MVC Design Pattern	Much like all sprint one elements this solution still completely follows the MVC design pattern. The 'Vision' module is the user interface (both View and Controller) to the database (Model). The reporting module is an addition to this design and functions purely as a controller.