The purpose of this program is to manage data for items stored in inventory, customers, and orders placed by customers. The main goal is to automate inventory management in response to orders: once an order is placed (i.e., finalized) the ordered quantity of each item is automatically deducted from the inventory in storage. This is accomplished through object-oriented programming and full database access. There are three main object classes: Inventory, Users, and Orders. Each object class has a full set of methods that are used to manipulate the data of an object. The class methods have direct access to a database that stores the long-term data for all inventory items, customers, and orders. Through database access, the class methods can retrieve objects for inventory items, customers, and orders. The class methods can even alter the data for the inventory items and customers in the database. Access to archived orders is read-only, since orders represent a record of past activity.

The program has a web interface using the Django framework. Our team has not made full use of the Django framework. The classes and their class methods are directly located in the views.py file of the application directory, and we are using our own database file, while directly accessing the database file using SQL in sqlite3. As part of the full application, there is a file in the main directory that can be used to create the database file. The file is a python script named “createTable.” The script merely needs to be run to create the database file, which is called “SmallShops.” There are three additional scripts in the main directory. These files can be used to delete all inventory items, users, and orders in the “SmallShops” database file. There is a separate script for each object.

In the database file, there are three tables: “Inventory,” “Users,” and “Orders.” Each file has a primary key column that has auto-incrementation enabled. There is no need to manually enter ID values when storing new rows in the database. The ID values are automatically assigned. The columns of each table represent the major attributes for each class. Inventory rows have the primary key ID column, the name of the inventory item, the department, the price, the quantity in storage, and the value of the quantity in storage. User rows have the primary key ID column, the user’s name, email, password, credit card, city, state, country, address, and phone number. Order rows have the primary key ID column, the user ID of the user that placed the order, the credit card used for the order, the shipping city of the order, the shipping state of the order, the shipping country of the order, and the shipping address of the order, the inventory item ID’s for the inventory items in the order, the price per each item in the order, the quantity per each item in the order, the cost per each item in the order, the original quantity of each item in storage, the original value of each item in storage, the altered quantity of each item in storage, the altered value of each item in storage, the total cost of the order, and the date of the order. There are no foreign key constraints in the Orders table, because the proper limitations for order data are enforced in the application’s coding.

The main interface is divided into five different pages: inventoryMenu, userMenu, orderMenu, employeeMenu, and customerMenu. inventoryMenu provides full access to all inventory management functions, userMenu provides full access to all user management functions, orderMenu provides full access to all order management functions, employeeMenu provides full access to all management functions, and customerMenu provides access to the functions that customers need for placing orders.

“Inventory List” outputs a list of all inventory items that are stored in the database.

“New Inventory Item” creates a new inventory object. The inventory object has no ID value, because the ID value can only be obtained by either adding the inventory object to the database or retrieving an inventory object from the database. The ID value is meant to represent the ID value for the row in the database. The name, department, price, and quantity can be manually entered. The value is derived by multiplying the price by quantity.

“Output Current Inventory Item” outputs the attributes of the current active inventory object.

“Add Current Inventory Item to Database” adds the current active inventory object to the database. A new database row is automatically created, which has an automatically assigned primary key value. The ID value of the active inventory object is altered to match the primary key value.

“Add New Inventory Item to Database” directly adds a new inventory object to the database using the manually entered attributes of inventory name, department, price, and quantity and the derived attribute value. The ID value of the inventory object is altered to match the automatically generated primary key value.

“Retrieve Inventory Item from Database” creates a new inventory object and sets all of its attributes to match a row in the database. A manually entered primary key value is necessary for retrieving the row in the database.

“Alter Current Inventory Item” alters all manually entered attributes for the current inventory object. Everything but the ID value can be altered.

“Alter Current Inventory Item Quantity” alters the quantity attribute for the current inventory object. The inventory value is recalculated using the new quantity.

“Commit Alterations to Current Inventory Item to Database” alters a row in the database to match the attributes of the current inventory object. For this function to work properly, the inventory object must have an ID value. The inventory object can only obtain an ID value by either getting added to the database through “Add Current Inventory Item to Database” or “Add New Inventory Item to Database” or getting retrieved from the database through “Retrieve Inventory Item from Database.”

“Delete Current Inventory Item” deletes the current inventory object from the database. As with “Commit Alterations to Current Inventory Item to Database,” the current inventory object must have an ID value for this function to work properly.

“Delete Selected Inventory Item” creates a new inventory object by retrieving a row from the database using a manually entered primary key value. The inventory object is then deleted from the database.

“User List” outputs a list of all users in the database.

“New User” creates a new user object with manually entered values for each major attribute, except for the ID value. The manually entered email cannot be reset using any other function.

“Output Current User” outputs the attributes for the current user object.

“Add Current User to Database” adds the current user object to the database. The primary key value is automatically created, and the ID value of the user object is altered to match the primary key value.

“Add New User to Database” creates a new user object using manually entered values for all attributes, except for the ID value, and the user object is immediately added to the database. The ID value is altered to match the primary key value.

“Retrieve User” creates a user object and obtains the data for the user object from the database using a manually entered primary key value.

“Update Current user Information” alters all attributes for the user object, except for the ID value and email.

“Commit Alterations to Current User to Database” alters a row in the database to match the data of the user object. This function only works properly if the user object has an ID value, which can only be obtained by adding a new user to the database through “Add Current User to Database” or “Add New User to Database” or by retrieving a user from the database through “Retrieve User.”

“Update Current User Password” alters the user object’s password with a manually entered value.

“Commit Reset Password of Current User to Database” alters the password of a row in the database to match the password of the user object. For this function to work properly, the user object must have an ID value.

“Delete Current User” deletes the current user from the database. For this function to work properly, the user object must have an ID value.

“Delete Selected User” creates a user object using data from the database and then deletes the user object from the database.

“User Login” mimics the login function of a user: an email and password are entered. A user object is created. If the email is found in the database, and if the password matches the email, then the attributes of the user object are set to the row of the database.

“Order List” outputs a list of all orders in the database.

“New Order” creates a new order object using data from the current user object. The ID value is left empty. For this function to work properly, the user object must have an ID value, meaning that it must have a row in the database.

“Output Order” outputs the attributes of the order object.

“Retrieve Order” outputs the data for a row in the database that represents a previous order object, using a manually entered primary key value.

“Set Address” alters the full address of the order object.

“Set Pay Method” alters the credit card of the order object.

“Add Current Item” adds an inventory item to the order using data from the current inventory object and a manually entered quantity. For this function to work properly, the inventory item must have an ID value.

“Add Item Through ID” adds an inventory item to the order using a row in the database through a manually entered primary key value and quantity.

“Remove Current Inventory Item” removes an inventory item from the order using data from the current inventory object. For this function to work properly, the inventory object must have an ID value.

“Remove Inventory Item With ID” removes an inventory item from the order using a manually entered ID value for the inventory item.

“Change Current Inventory Item Quantity” alters the quantity of an inventory item in the order using the data of the current inventory object and a manually entered quantity. For this function to work properly, the inventory object must have an ID value.

“Change Inventory Item Quantity with ID” alters the quantity of an inventory item in the order through a manually entered ID value and a manually entered quantity.

“Finalize Order” adds a date value to the order object and adds the order object to the database. A primary key value is automatically generated, and the order ID is altered to match the primary key value.