Files and Methods

Controller

MainController

```
public class MainController {
      FileManager fileManager;
      MainState mainState;
      FileState fileState;
      String writeToFileText;

      ArrayList<Admin> admins = new ArrayList<>();
      ArrayList<Customer> customers = new ArrayList<>();
      ArrayList<Product> products = new ArrayList<>();
      ArrayList<Order> orders = new ArrayList<>();
      ArrayList<OrderItem> orderItems = new ArrayList<>();
      ArrayList<OrderItem> tempOrderItems = new ArrayList<>();


      boolean isAdmin;
      boolean isAdminLogin = false;
      boolean isCustomerLogin = false;
      int loginCustomerId = 0;


      // get all data to ArrayList when MainController loaded
      public MainController() {
       readAllFiles();
      }


      //Read all files and save to ArrayList
      //Manage checkPersion
```

- [ ] getIsAdmin()
- [ ] getIsAdminLogin()
- [ ] getIsCustomerLogin()
- [ ] getMainState()
- [ ] getFileState()
- [ ] getWriteToFileText()

- ❑ isHasAdminPermission()
- ❑ isHasCustomerPermission()
- ❑ isHasBothPermission()
- ❑ getLoginCustomerId()
- ❑ getTempOrderItems()
- ❑ setIsAdmin(boolean newValue)
- ❑ setAdminLogin(boolean newValue)
- ❑ setCustomerLogin(boolean newValue)
- ❑ setLoginCustomerId(int customerId)
- ❑ loginForNewCustomer(int customerId)
- ❑ addNewAdmin(Admin newItem)
- ❑ addNewCustomer(Customer newItem)
- ❑ addNewProduct(Product newItem)
- ❑ addNewOrder(Order newItem)
- ❑ addNewOrderItem(OrderItem newItem)
- ❑ readAllFiles(){

```
 //Loop enum to reduce lines
 //ADMIN,PRODUCT,CUSTOMER,ORDER,ORDERITEM,
 for (MainState mainStateItem : MainState.values()) {
   mainState = mainStateItem;
   fileState = FileState.READ;
   fileManager = new FileManager(this, mainState, fileState);
   fileManager.choose();
 }
}
```

- ❑ addNewLine()

}

AuthController

```
public class AuthController {
      DefaultView view;
      int maxTry = 3;
      int nextTryInSecond = 10;
      MainController mainObject;

      public AuthController(MainController mainObject) {
       this.mainObject = mainObject;
       view = new DefaultView(mainObject);
      }
```

```
        //Manage login both backend and frontend
        //Max try 3 times
        //Next try 10 second
        //Navigate to backend menu / frontend menu
           ☐ setLogin()
           ☐ setLogout()
           ☐ customerLogin()
           ☐ isAdminLoginSuccess(String username, String password)
           ☐ delay()
           ☐ isCustomerLoginSuccess(String username, String password)
           ☐ isAdminLoginSuccess(String username, String password)

}
```

CustomerController

```
public class CustomerController {
        CustomerView view = new CustomerView();
        MainController mainObject;

        public CustomerController(MainController mainObject) {
         this.mainObject = mainObject;
        }



        //Frontend
           ☐ register()
           ☐ updateForm(int indexItem)

        //Backend
           ☐ listAll()
           ☐ showCustomerList()
           ☐ updateStatus()
           ☐ delete()

        //Both frontend and backend
           ☐ viewProfileByCustomerId(int customerId)
           ☐ viewProfileByItemIndex(int searchIndex)
           ☐ viewCustomerOrder()
           ☐ showProfile(int itemIndex)

        //Help functions
           ☐ checkIfHasOrder(int customerId)
           ☐ rewriteFile()
           ☐ getCustomerIndexItemById(int customerId)
           ☐ searchByItemIndex(int itemIndex)
           ☐ searchByInputNumber(String inputString)
           ☐ newItemInputForm(String label)
           ☐ updateItemInputForm(String oldString, String label)


}
```

OrderController

```
public class OrderController {
      OrderView view = new OrderView();
      OrderItemController orderItemObject;
      MainController mainObject;

      public OrderController(MainController mainObject) {
       this.mainObject = mainObject;
       orderItemObject = new OrderItemController(mainObject);
      }

      //Frontend
        ☐ addTempOrderItem(Product productItem, int amount)
        ☐ confirmOrder()

      //Backend
        ☐ listAll()
        ☐ search()
        ☐ updateOrderItem()
        ☐ delete()

      //Both frontend and backend
        ☐ orderHistory(int customerId)
        ☐ printOrderList(ArrayList<Order> orders)
        ☐ getOrderDetail(int orderIndexId)
```

```
           ☐ listOrderItems(ArrayList<OrderItem> orderItems)

      //Help functions
           ☐ searchOrderByIndex(int searchId)
           ☐ getCustomerName(int customerId)
           ☐ getTotalPurchase(int orderId)
           ☐ rewriteFile()
           ☐ getNextId()
           ☐ searchByInputNumber(String inputString)

}
```

OrderItemController

```
public class OrderItemController {
      OrderView view = new OrderView();
      MainController mainObject;
      public OrderItemController(MainController mainObject) {
       this.mainObject = mainObject;
      }

      //Frontend
           ☐ deleteOrderItem(int ItemIndex)
           ☐ updateOrderItem(int orderItemId, int amount)
           ☐ addOrderItem(OrderItem newItem)
      //Both frontend and backend
           ☐ getOrderItems(int orderId)
           ☐ listOrderItems(ArrayList<OrderItem> orderItems)
           ☐ getProductItemByProductId(int productId)
      //Help functions
           ☐ rewriteFile()
           ☐ getNextId()
           ☐ searchByIndex(int ItemIndex)
           ☐ searchByIndex(String inputString)
}
```

ProductController

```
public class ProductController {
      ProductView view = new ProductView();

      OrderController orderObject;
      MainController mainObject;
      public ProductController(MainController mainObject) {
       this.mainObject = mainObject;
       orderObject = new OrderController(mainObject);
      }

      //Frontend
         ☐ frontendProductList()
         ☐ addProductToCart()
         ☐ viewCart()
         ☐ deleteOrderItem()
         ☐ confirmOrder()

      //Backend
         ☐ listAll()
         ☐ search()
         ☐ newItem()
         ☐ update()
         ☐ delete()

      //Help functions
         ☐ rewriteFile()
         ☐ searchByItemIndex(int itemIndex)
         ☐ newItemInputForm(String label)
         ☐ updateItemInputForm(String oldString, String label)

}
```

Data Class

Admin

```
public class Admin {
 String username;
 String password;
}
```

Customer

```
public class Customer {
 int customerId;
 String firstname;
 String lastname;
 String username;
 String password;
 String email;
 String address;
 String zipCode;
 String country;
 boolean isActive;
}
```

Order

```
public class Order {
 int orderId;
 int customerId;
 String remark;
 boolean isPending;
 LocalDateTime orderDate;
 LocalDateTime completeDate;
}
```

OrderItem

```
public class OrderItem {
 int orderItemId;
 int orderId;
 int productId;
 int amount;
 double fullPrice;
 double salePrice;
}
```

Product

```
public class Product {
 int productId;
 String productName;
 double price;
 boolean isSoldOut;
 boolean isActive;
}
```

Manager

FileManager

```
public class FileManager {
 String filepath = "src/datafiles/";
 String filename;
 this.mainObject = mainObject;
 this.mainState = mainState;
 this.fileState = fileState;


 public void choose(){

   switch (mainObject.getMainState()) {
     case ADMIN -> filename = filepath + "Admin.txt";
     case PRODUCT -> filename = filepath + "Product.txt";
     case CUSTOMER -> filename = filepath + "Customer.txt";
     case ORDER -> filename = filepath + "Order.txt";
     case ORDER_ITEM -> filename = filepath + "OrderItem.txt";
   }

   switch (mainObject.getFileState()) {
     case READ -> readFile();
     case NEW_LINE -> addNewLine(mainObject.getWriteToFileText());
     case OVERWRITE -> overwriteFile(mainObject.getWriteToFileText());
   }

 }
```

- ☐ `readFile()`
- ☐ `readWithScanner(String filename)`
- ☐ `addNewLine(String contentLine)`
- ☐ `writeToFile(String writeContent, boolean isRewrite)`
- ☐ `addNewAdmin(String contentLine)`
- ☐ `addNewCustomer(String contentLine)`
- ☐ `addNewProduct(String contentLine)`

```
        ☐ addNewOrder(String contentLine)
        ☐ addNewOrderItem(String contentLine)

}
```

FormatManager

```
public class FormatManager {
        ☐ stringToLocalDate(String dateString)
        ☐ localDateToString(LocalDateTime localDateTime)
        ☐ stringToInt(String strNumber)
        ☐ StringToDouble(String doubleString)
        ☐ isNumber(String inputNumber)
}
```

Color

CustomerColor
DefaultColor
MainColor
OrderColor
ProductColor

State Folder

```
public enum FileState {
 READ,
 NEW_LINE,
 OVERWRITE
}

public enum MainState {
 ADMIN,
 PRODUCT,
 CUSTOMER,
 ORDER,
 ORDER_ITEM,
}

public enum OperationState {
 ADD,
 SEARCH,
 DELETE,
 UPDATE_STATUS,
 VIEW_ORDER,

}
```

View

Theme

CustomerView
DefaultView
OrderView
ProductView