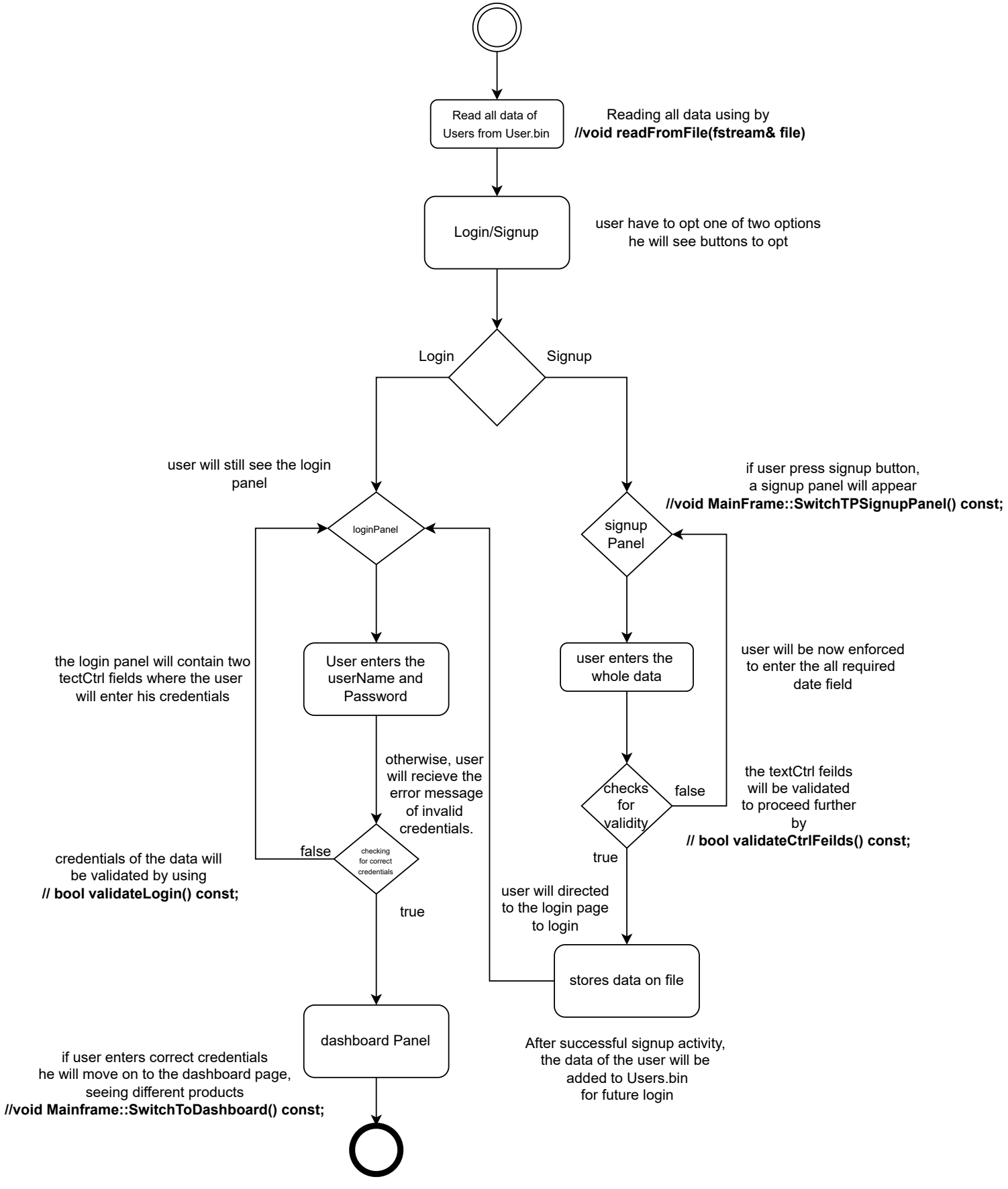
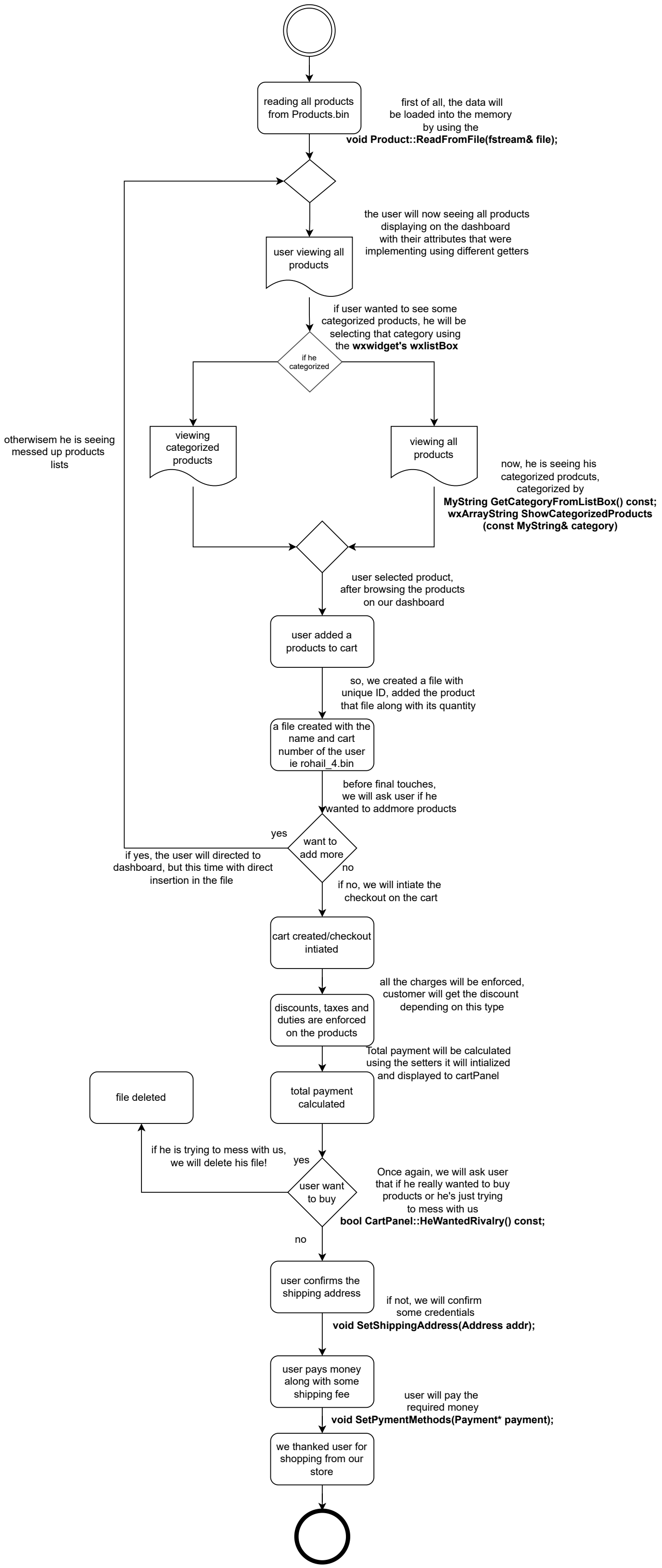
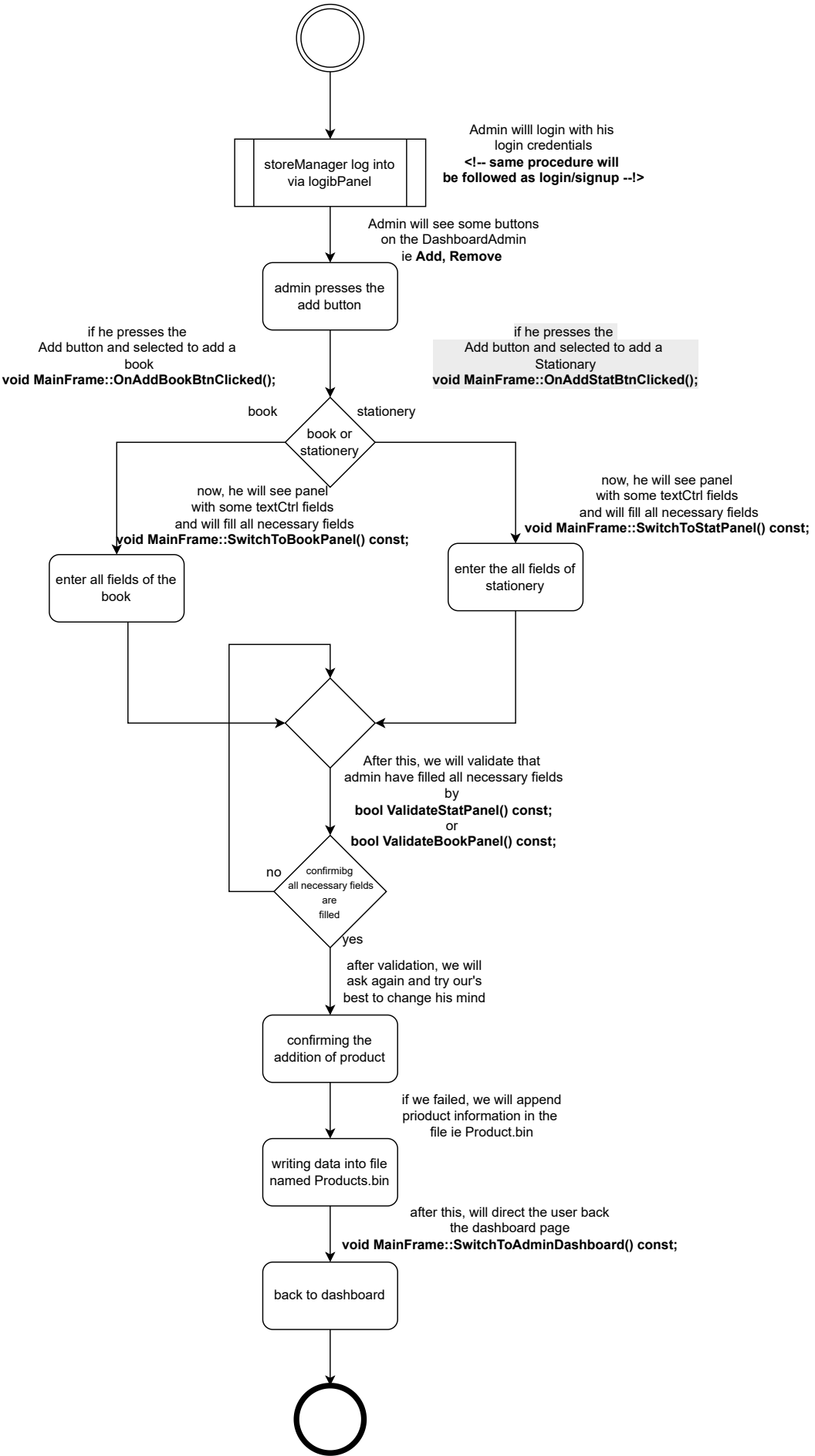
Book
MyString bookId;
MyString bookName;
MyString authorName;
MyString publisherName;
MyString bookType;
MyString bookLanguage;
MyString bookEdition;
int pages;
Book(const Book& other);
Book(const MyString& id, const MyString& name, const MyString& author, const MyString& publisher, const MyString& type, const MyString& language, const MyString& edition, int pages);
Book& operator=(const Book& other);
void writeToFile(fstream& fout) const override;
void readFromFile(fstream& fin) const override;
//setter
void SetPrice(int price) override;
void SetProductDescription(const MyString& desc) override;
//getter
MyString getProductID() const override;
MyString getProductName() const override;
MyString getProductCategory() const override;
MyString getDescription() const override;
int getPrice() const override;
MyString GetBookId() const;
MyString GetBookName() const;
MyString GetAuthorName() const;
MyString GetPublisherName() const;
MyString GetBookType() const;
MyString GetBookLanguage() const;
MyString GetBookEdition() const;
int GetPages() const;
//display
void DisplayProduct() const override;
Stationery
MyString StationeryId;
MyString StationeryName;
MyString StationeryType;
MyString StationeryBrand;
MyString StationeryColor;
Stationery(const Book& other);
Stationery(const MyString& id, const MyString& name, const MyString& type, const MyString& brand, const MyString color);
Stationery& operator=(const Stationery& other);
void writeToFile(fstream& fout) const override;
void readFromFile(fstream& fin) const override;
//setter
void SetPrice(int price) override;
void SetProductDescription(const MyString& desc) override;
//getter
MyString getProductID() const override;
MyString getProductName() const override;
MyString getProductCategory() const override;
MyString getDescription() const override;
int getPrice() const override;
MyString GetStationeryId() const;
MyString GetStationeryName() const;
MyString GetStationeryBrand() const;
MyString GetStationeryType() const;
MyString GetStationeryColor() const;
//display
void DisplayProduct() const override;
Patreon
static int discountRate;
MyString PatreonId;
bool isActivePatreon;
Patreon();
Patreon(const Patreon& other);
Patreon(const MyString email, const Date& dob, const MyString& name, const Address& address);
Patreon& operator=(const Patreon& other);
void writeToFile(fstream& fout) const override;
void readFromFile(fstream& fin) const override;
//setter
void AddOrder(Order*& order) override;
void SetLogin(const Login& login) override;
void SetName(const Name& name) override;
void SetEmail(const MyString& email) override;
void SetAddress(const Address& add) override;
void SetPerson(const Person& person) override;
void SetPremiumMember();
void RemovePremiumMember();
//getters
MyString GetPersonType() const override;
MyString GetPerson() const override;
MyString getLoginName() const override;
MyString getPassword() const override;
Name GetName() const override;
bool isActivePremiumMember()const;
//display
void DisplayCustomer() const override;
Premium
static int discountRate;
bool isActivePremiumMember;
Premium();
Premium(const Premium& other);
Premium(const Login& login, const MyString email, const Date& dob, const Name& name, const Address& add);
Premium& operator=(const Premium& other);
void writeToFile(fstream& fout) const override;
void readFromFile(fstream& fin) const override;
//setters
void AddOrder(Order*& order) override;
void SetLogin(const Login& login) override;
void SetName(const Name& name) override;
void SetEmail(const MyString& email) override;
void SetAddress(const Address& add) override;
void SetPerson(const Person& person) override;
void SetPremiumMember();
void RemovePremiumMember();
//getters
MyString GetPersonType() const override;
MyString GetPerson() const override;
MyString GetLoginName() const override;
MyString getPassword() const override;
Name GetName() const override;
bool isActivePremiumMember()const;
//display
void DisplayCustomer() const override;
Classic
static int discountRate;
Classic();
Classic(const Classic& other);
Classic(const Login& login, const MyString email, const Date& dob, const Name& name, const Address& add);
Classic& operator=(const Classic& other);
void writeToFile(fstream& fout) const override;
void readFromFile(fstream& fin) const override;
//setters
void AddOrder(Order*& order) override;
void SetLogin(const Login& login) override;
void SetName(const Name& name) override;
void SetEmail(const MyString& email) override;
void SetAddress(const Address& add) override;
void SetPerson(const Person& person) override;
void SetPremiumMember();
void RemovePremiumMember();
//getters
MyString GetPersonType() const override;
MyString GetPerson() const override;
MyString GetLoginName() const override;
MyString getPassword() const override;
MyString GetName() const override;
//display
void DisplayCustomer() const override;
Orders
Cart* cart;
Address shippingAddress;
MyString status;
Date orderDate;
Date deliveryDate;
int totalPrice;
Order();
Order(const Order& other);
Order(const Address& add, const MyString& shpAddr, const Date& orderDate, const int& price);
Order& operator=(const Order& other);
//setters
void SetShippingAddress(const Address& addr);
void SetStatus(const MyString& status);
void SetDeliveryDate(const Date& date);
void SetCart(const Cart& cart);
void AddProductToCart(Product* prod);
//getters
MyString GetOrderStatus() const;
Date GetOrderDate() const;
int GetTotalPrice() const;
Date GetDeliveryDate() const;
Address GetShippingAddress() const;
//display
void DisplayCartProduct() const;
void CartInfo() const;
Cart
Payment* payment;
MyVector<Product**> products;
int totalPrice;
Cart();
Cart(const Cart& other);
Cart& operator=(const Cart& other);
void writeToFile(fstream& fout) const;
void readFromFile(fstream& fin) const;
//setters
void SetPayment(Payment* payment);
void AddProductToCart(Product* product);
void ClearCart();
void RemoveProduct(Product* product);
//getters
MyVector<Product**> GetProducts() const;
int GetProductCount() const;
MyString GetPraductList() const;
Payment* GetPaymentInfo() const;
int GetTotalPrice() const;
Payments
MyString cardNumber;
MyString cardHolderName;
Date expirationDate;
MyString cvv;
Payment();
Payment(const Payment& other);
Payment(const MyString& cardNumber, const MyString& holderName, const Date& date, const MyString& cvv);
Payment& operator=(const Payment& other);
//setters
void selCardNumber(const MyString& cardNumber);
void selCardHolderName(const MyString& holderName);
void selExpirationDate(const Date& date);
void selCvv(const MyString& cvv);
//validators
bool ValidateCvv(const MyString& cvv) const;
bool ValidateCardNumber(const MyString& cardNumber) const;
bool ValidateCardHolderName(const MyString& cardHolderName) const;
bool ValidateExpirationDate(const Date& date) const;
Login
MyString username;
MyString password;
Login();
Login(const MyString& username, const MyString& password);
Login(const Login& other);
Login& operator=(const Login& other);
bool operator==(const Login& other) const;
bool operator!=(const Login& other) const;
void setUsername(const MyString& _username);
void setPassword(const MyString& _password);
MyString getLoginName() const;
MyString getPassword() const;
Person
MyString email;
Date birthDate;
Name fullName;
Address address;
Login login;
Person();
Person(const Person& other);
Person(const Login& _login, const MyString email, const Date& dob, const Name& name, const Address& add);
Person& operator=(const Person& other);
bool operator==(const Person& other) const;
bool operator!=(const Person& other) const;
virtual void writeToFile(fstream& fout) const;
virtual void readFromFile(fstream& fin) const;
//setter
virtual void SetLogin(const Login& _login);
virtual void SetName(const Name& _name);
virtual void SetEmail(const MyString& _email);
virtual void SetAddress(const Address& add);
virtual void SetPerson(const Person& _person);
// display
virtual void display() const;
virtual MyString GetPersonType() const = 0;
virtual MyString getLoginName() const;
virtual MyString getPassword() const;
virtual Name GetName() const;
virtual Login GetLogin() const;
virtual Name GetName() const;
virtual Address GetAddress() const;
virtual MyString GetEmail() const;
virtual Date GetDate() const;
Admin
static int AdminCount;
MyVector<Store*>* stores;
Admin();
Admin(const Admin& other) = delete;
Admin& operator=(const Admin& other) = delete;
Admin(const Login& _login, const MyString email, const Date& dob, const Name& name, const Address& add);
static Admin* GetInstance();
void writeToFile(fstream& fout) const;
void readFromFile(fstream& fin) const override;
//setters
void AddStore(Store* store);
void RemoveStore(Store* store);
void ChangeStoreManager(Person* person, Store* store);
void AddProductToStore(Product* product, Store* store);
void RemoveProductFromStore(Product* product, Store* store);
void AddCustomerToStore(Customer* customer, Store* store);
void RemoveCustomerFromStore(Customer* customer, Store* store);
//getters
MyVector<Store*>* GetStores() const;
MyVector<Product*>* GetProducts() const;
MyVector<Customer*>* GetCustomers() const;
MyVector<Order*>* GetOrders() const;
MyString GetProductListByStore(Store* store) const;
// display
void DisplayProducts() const;
void DisplayStores() const;
void DisplayStoreManagers() const;
void DisplayCustomers() const override;
Customer
MyVector<Order*> orders;
Customer();
Customer(const Customer& other);
Customer(const Login& _login, const MyString email, const Date& dob, const Name& name, const Address& addr);
Customer& operator=(const Customer& other);
void writeToFile(fstream& fout) const override;
void readFromFile(fstream& fin) const override;
//setters
void AddOrder(Order*& order) override;
void SetLogin(const Login& _login) override;
void SetName(const Name& _name) override;
void SetEmail(const MyString& _email) override;
void SetAddress(const Address& add) override;
void SetPerson(const Person& _person) override;
void SetPremiumMember();
void RemovePremiumMember();
//getters
MyString GetPersonType() const override;
MyString GetPerson() const override;
MyString GetLoginName() const override;
MyString getPassword() const override;
Name GetName() const override;
bool isActivePremiumMember()const;
//display
void DisplayCustomer() const override;
Product
MyString productID;
wxImage productImage;
MyString productName;
MyString description;
int price;
Product();
Product(const Product& other);
Product(const MyString& id, const MyString& name, const MyString& category, const MyString& desc, int price, const wxImage& image);
Product& operator=(const Product& other);
virtual void writeToFile(fstream& fout) const;
virtual void readFromFile(fstream& fin) const;
//setter
virtual void SetPrice(int price);
virtual void SetProductDescription(const MyString& desc);
//getter
virtual MyString getProductID() const;
virtual MyString getProductName() const;
virtual MyString getProductCategory() const;
virtual MyString getDescription() const;
virtual int getPrice() const;
//display
virtual void display() const;
Store
MyString storeId;
MyString storeName;
Address storeAddress;
MyVector<Product*>* products;
MyVector<Customer*>* customers;
Store();
Store(const Store& other);
Store(const MyString& storeId, const MyString& storeName, const Address& storeAddress);
Store& operator=(const Store& other);
// displays
void DisplayStore() const;
void DisplayProducts() const;
void DisplayCustomers() const;
void DisplayProduct(const MyString& productId) const;
void DisplayProductByCategory(const MyString& category) const;
void DisplayCustomer(const MyString& customerId) const;
MyVector<Product*>* GetProducts() const;
MyVector<Customer*>* GetCustomers() const;
void GetStoreInfo() const;
//adders
void AddCustomer(Classic*& customer);
// setters
void SetStore(Store* store);
// virtual
MyString GetPersonType() const override;
MyString getUserName() const override;
MyString getPassword() const override;
StoreManager
Store* store;
StoreManager();
StoreManager(const StoreManager& other);
StoreManager(const StoreManager& operator=(const StoreManager& other);
//display
wxArrayString DisplayCustomer() const;
void DisplayStore() const;
// getters
Store* GetStore() const;
MyString GetLists();
// setters
void SetStore(Store* store);
// virtual
MyString GetPerson() const override;
MyString getUserName() const override;
MyString getPassword() const override;
Address
MyString houseNumber;
MyString city;
MyString province;
MyString country;
Address();
Address(MyString houseNumber,const MyString& address(const MyString& province, const MyString& co);
Address(const Address& other);
Address& operator=(const Address& other);
friend ostream& operator<<(ostream& os, const Address& a);
bool operator==(const Address& other) const;
void writeToFile(fstream& fout) const;
void readFromFile(fstream& fin) const;

SIGNUP/LOGIN ACTIVITY DIAGRAM

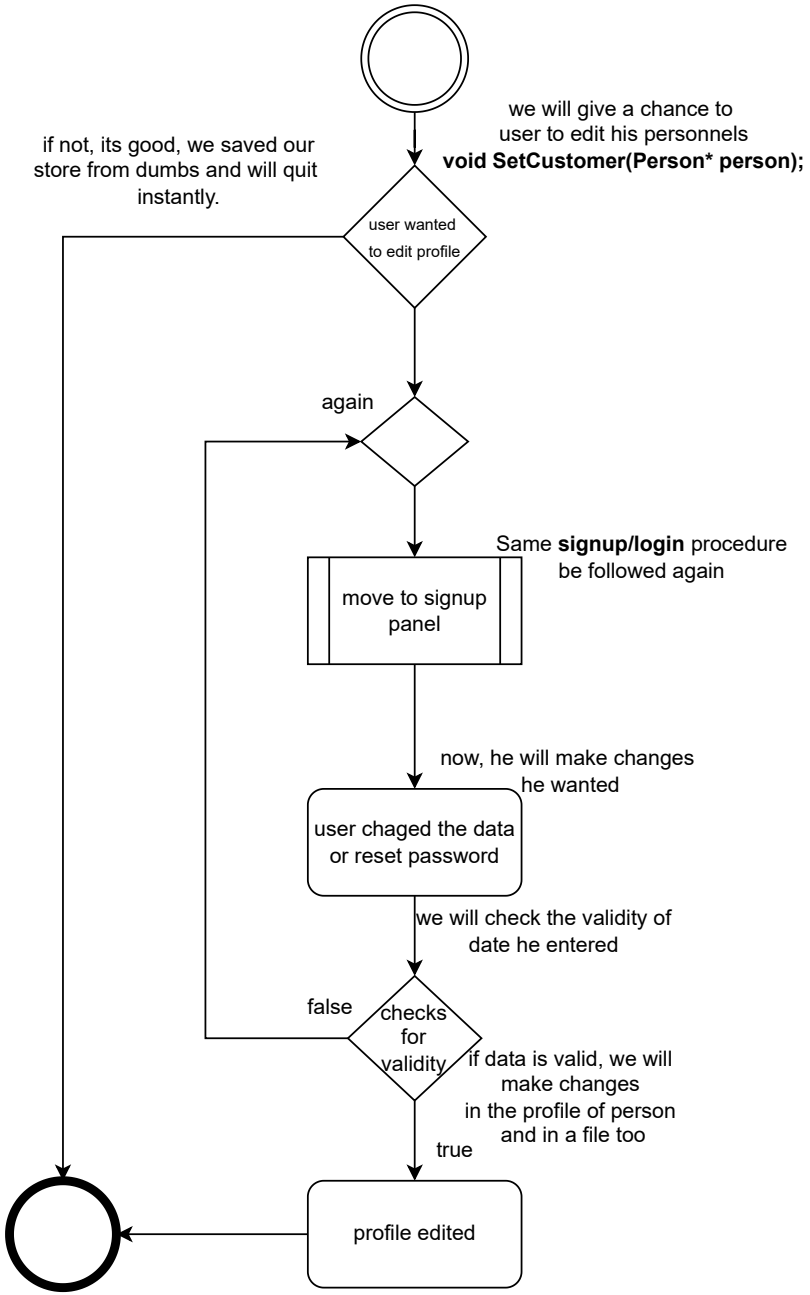



reading all products from Products.bin
first of all, the data will be loaded into the memory by using the
void Product::ReadFromFile(fstream& file);
the user will now seeing all products displaying on the dashboard with their attributes that were implementing using different getters
user viewing all products
if user wanted to see some categorized products, he will be selecting that category using the wxwidget's wxlistBox
if he categorized
viewing categorized products
viewing all products
otherwisem he is seeing messed up products lists
now, he is seeing his categorized prodcuts, categorized by
MyString GetCategoryFromListBox() const;
wxArrayString ShowCategorizedProducts (const MyString& category)
user selected product, after browsing the products on our dashboard
user added a products to cart
so, we created a file with unique ID, added the product that file along with its quantity
a file created with the name and cart number of the user ie rohail_4.bin
before final touches, we will ask user if he wanted to addmore products
yes
want to add more
no
if yes, the user will directed to dashboard, but this time with direct insertion in the file
if no, we will intiate the checkout on the cart
cart created/checkout intiated
all the charges will be enforced, customer will get the discount depending on this type
discounts, taxes and duties are enforced on the products
Total payment will be calculated using the setters it will intialized and displayed to cartPanel
file deleted
total payment calculated
if he is trying to mess with us, we will delete his file!
yes
user want to buy
Once again, we will ask user that if he really wanted to buy products or he's just trying to mess with us
bool CartPanel::HeWantedRivalry() const;
no
user confirms the shipping address
if not, we will confirm some credentials
void SetShippingAddress(Address addr);
user pays money along with some shipping fee
user will pay the required money
void SetPymentMethods(Payment* payment);
we thanked user for shopping from our store

Adding a Product



Edting the User Profile



<!-- this activity diagram will be validated for all kind of the person, ie if admin wanted to change some personals or the store manager--!>