Cafe Go

The mobile app that serves coffee from a cafe

Daniel Ontiveros, Bilal Qureshi, & Samuel Puchinskiy

12.13.2019

Team 4

Software Engineering CS 319-02

Professor Sotak

Table of Contents

Contents Page

Vision & Scope

Background: Business Opportunity/Customer Needs

Currently, there are long wait lines at the school's coffee shop, the congestion may cause students to be late to class and employees to be late to work. The Café Go app will make it simple and convenient to get food or a beverage before, during or after class. This mobile application will help individuals customize their coffee order to their liking. Instantly, order and pay on your phone and your food or beverage will be waiting for you when your arrive to the coffee shop. The food/beverages will be stored in small secured lockers in which users can simply scan their barcodes to get access to their items. The app will contain menu options from the cafe, where a person can choose and easily customized. There is a login where all history, payments, and favorites will be saved for your records and saving you time for future orders.

Vision Statement

The old way of ordering your coffee is waiting through a long line during rush hour. Occasionally, individuals will wait up to thirty minutes for their coffee to be ready.The benefits of creating this mobile app is to decrease wait time and increase satisfaction. Order on your phone, scan your barcode, and your coffee is ready!

As well with the brand new app, we add a reward programs in the app. For every dollar spent, one point will be added to the person's account. When they reach 100 points, they will earn a $5 dollar credit towards their next coffee order! This will increase people to migrate towards our app and will eventually decrease the wait line.

Project Selection

Currently, Northeastern Illinois University has no application to order from the Coffee Shop. There are long lines and students and staff members are always late because of the long line.

To fix this issue, we need an easy and convenient application that will allow customers to create orders, pay for their orders, and pick up their order at the Coffee Shop. This application will save time and be less hassle for the customer to order, pay and pickup their order.

Major Features

- Be able to order food/ drink on the mobile app
- Be able to customize on what to add or delete on the order
- Be able to view favorites and past orders that future orders are quickly made.
- Have a barcode to be scanned when picking up order that way it is the correct order for the correct customer.

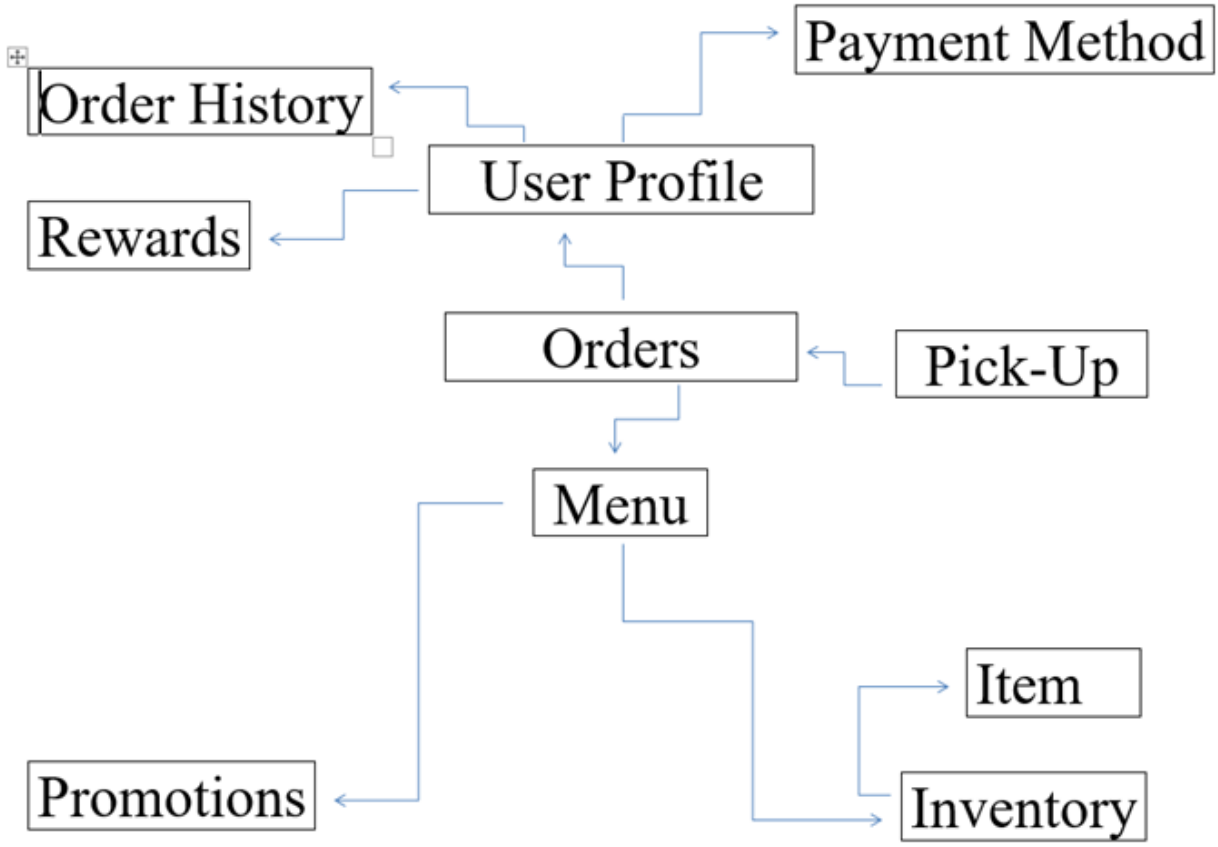
Problem Definition

We live in a face paced environment, where there is no time waiting in lines to get your hot or cold beverage in the rush of morning sunset. Long lines will make any student or staff member late to class or work. There should be an easier and safer way to order from the school's Cafe shop. Students and Staff members should pay from their phone and grab their coffee and go to their class. More time studying and less time waiting.

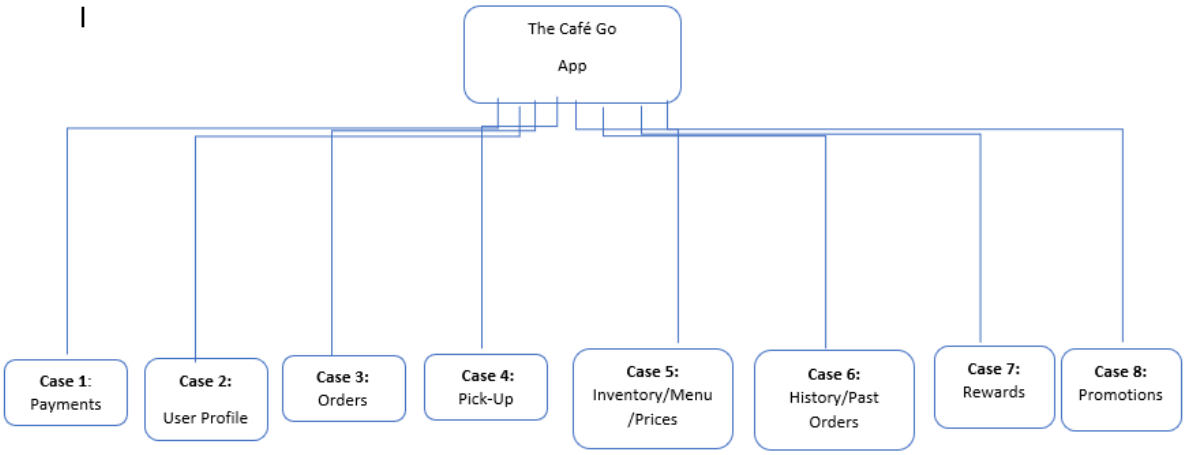
Problem Solution

The solution is an easy and convenient web application that any student or staff member can use. The Cafe Go app will contain the Menu for students or staff members to choose from and pay for them using the app. The individual can order from the app and pick their order at the Coffee Shop without standing in lines. Students can pay on their phone, and grab their coffee at the Coffee Shop, and go on to their studies. There is no waiting in line because the coffee will be made when they order and pay from the app. The solution is to keep a database of the menu and each individual account information without the need to always to reinstate their payment and order. The individual can order using their previous payment option and pervious order recite.

Activity Relationship Diagram



Topic: NEIU Cafe Ordering Site



Use Case 1: Payments - view a variety of payment options for orders. (Sam)

Use Case 2: User Profile - users will be able to create a profile to save their information.(Sam)

Use Case 3: Orders - what the user is currently ordering. (Bilal)

Use Case 4: Pick-Up - lets the user know that their order is ready for pickup at the cafe.

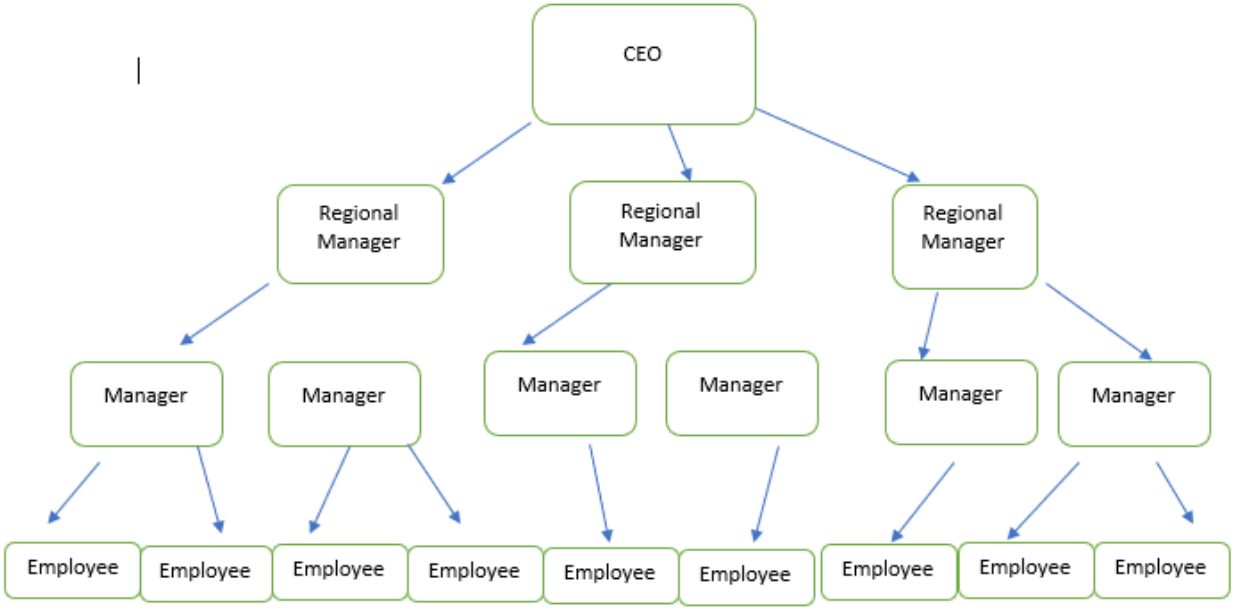
Use Case 5: Inventory/Menu/Prices - what is available to order at the cafe.(Bilal)

Use Case 6: History/Past Orders - saved orders, in case user would like to order again.

Use Case 7: Rewards - point accumulation to claim rewards for orders.(Danny)

Use Case 8: Promotions - lets users know what weekly item is available for sale price.(Danny)

Organization Chart



Case 1 Payment Description

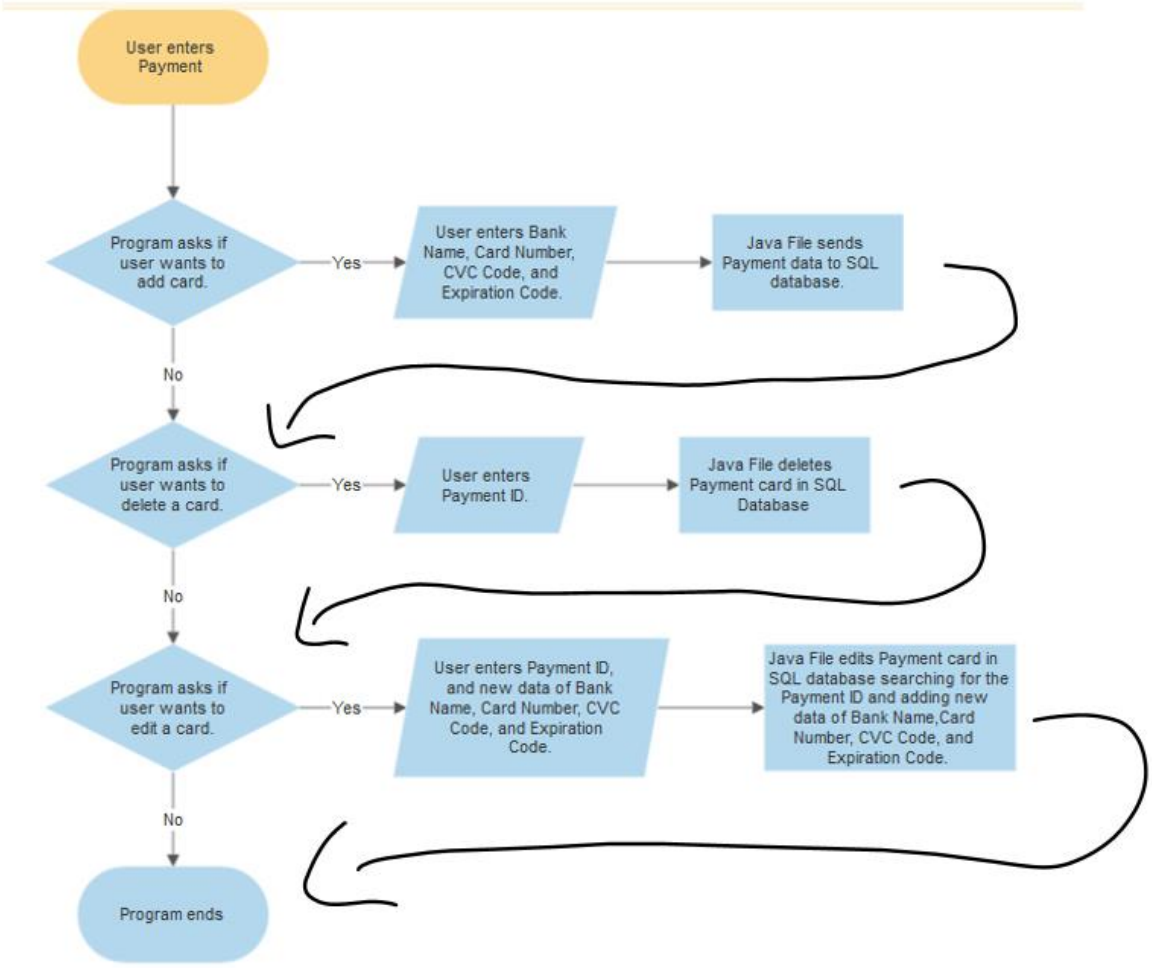
The payment case contains two classes. The PaymentCard java file and the PaymentTest java file.

The PaymentCard class is an object that creates a payment option for the user. This class contains Paymentid, BankName, CardNum, CVCCode, and expirationDate. All of these variables are integers, except BankName is a string variable. There constructor that assigns values for all the variables. In addition there are setter and getter methods for all of the variables.

The PaymentTest class is meant to be used with the Payment SQL database. This class creates, edits and deletes an object of Payment. When the customer wants to create a Payment, then the program will ask for Bank Name, Card Number, CVC Code, and Expiration Date. These values are then sent to the SQL database, with the Payment ID being auto incremented. The Edit Payment method will ask the user to enter the Payment ID, and the new values for Bank Name, Bank Number, CVC code, and Expiration Date. The delete method will delete the Payment Object in the SQL database by asking the user for the Payment ID. In addition, the program contains a Connection method that creates a connection from the Java file with the SQL database. The main method prompts the customer if they wish to Add, Edit, or Delete a Payment object.

Case 1 Payment Table	
ID	Payment_Case_1
Created by	Samuel Puchinskiy
Date Created	October 4th, 2019
Description	Selecting a payment option, either card, paypal, or even cash
Primary Actors	Users (Instructors, Students, factory members) who will be paying for their coffee order
Triggers	The users clicking on payment
Precondition	1. User's profile been fully created 2. Item(s) from Menu has been picked
Postconditions	1. Online payment has been successfully approved from bank 2. Or user gave cash/check to cashier 3. Code for order has been sent via email or phone message 4. User is granted the order with paper recite or emailed receipt
Normal Flow	1. The user chooses how they are going to pay. Either they at the cafe shop with cash or check OR they pay using the app via card or paypal 2. If the person wants to pay with a credit card or paypal, then they can pay online 3. If the person wants to pay with cash or check, then they will pay at the cashier register 4. Once the payment is successfully approved, the user will get a code for their order via email or phone number.
Alternative Flow	1. The alternative flow is described in the normal flow. The user can pay with cash, instead of card
Exceptions	❖ If user left payment blank, display error ❖ If user's payment has been declined, display error
Priority	Very high, user will not receive order without proper payment
Frequency of use	Every time the user is ordering from the app
Business Rules	The cafe shop
Other Information	None
Assumptions	None

Case 1 Payment Flowchart



Case 1 Payment UML

PaymentCard
Paymentid: int; BankName String; CardNum: Int; CVCCode int; expirationDate int;

PaymentCard(int Paymentid, String BankName, int CardNum, int CVCCode, int expirationDate): void;
setPaymentid(int Paymentid): void;
getBankName(): String BankName;
setBankName(String BankName): void;
getCardNum(): Int CardNum;
setCardNum(int CardNum): void;
getCVCCode(): Int CVCCode;
setCVCCode(int CVCCode): void;
getexpirationDate(): int exirationDate;
setexpirationDate(int expirationDate); void;

PaymentTest
Input: Scanner; Card: Payment;
Main(): void; getConnection(): Connection; AddPayment(PaymentCard ID): int status; EditPayment(int ID): int status; DeletePayment(int ID): int status;

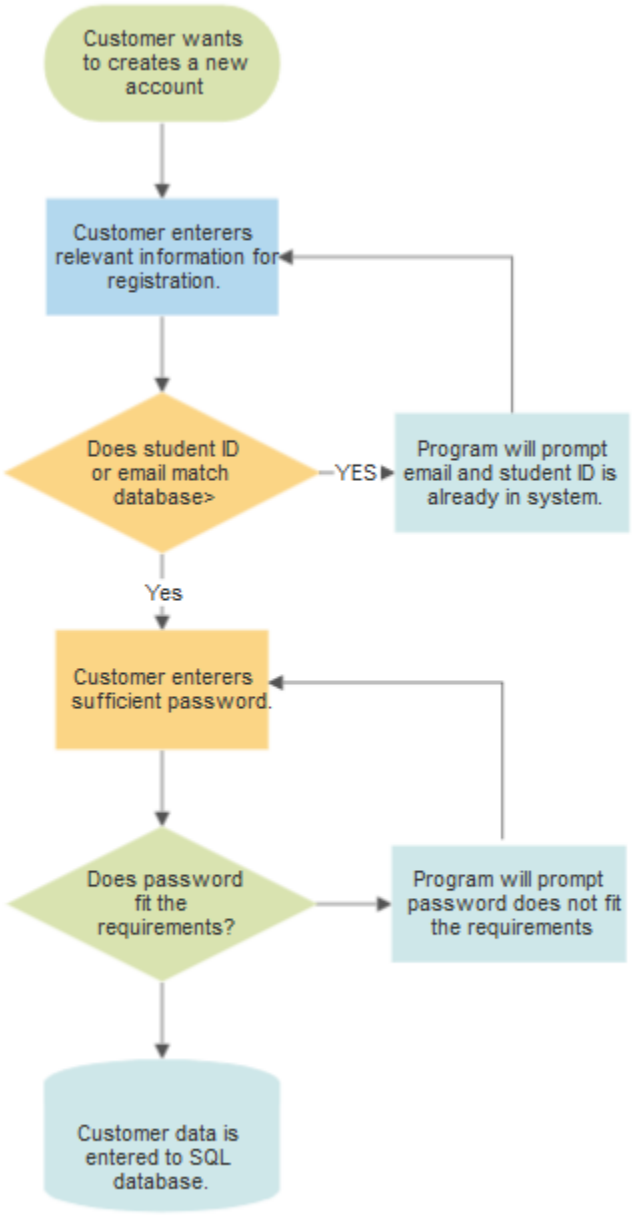
Case 2 Profile Description

The customer's profile will ask the user to enter their first name, last name, username, email, phone number, password, age, and possibly their student ID as a primary key. The customer must provide a unique email and student ID number. If the email and student ID number matches from the database, then the program will prompt the customer already has an account and if they wish to change their password. If they wish to change their password, then they must provide their email and student ID and/or birthday and/or username for verification purposes. In addition, the customer must provide a sufficient password, that includes eight strings of characters, one capital letter, one numeric value, and a special character.

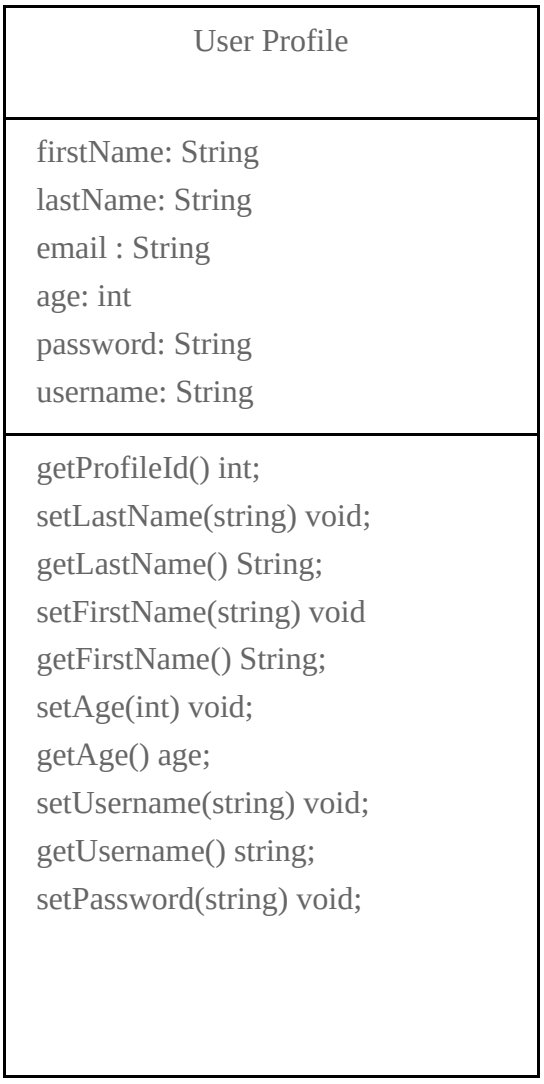
An object will be created each time the customer creates their profile. The object is then sent to the SQL database. Once the customer officially creates their profile, then they are granted more permissions. The customer can add a payment option and create an order using the menu. The customer can view what's in stock and not stock from the inventory. The customer can edit their profile name, phone number, password but cannot change their student ID number or username without administrator rights. All these edits will be updated in the SQL database.

Case 2 Profile Table	
ID	Profile_Case_2
Created by	Samuel Puchinskiy
Date Created	October 4th, 2019
Description	The user will create a profile
Primary Actors	Users (Instructors, Students, factory members, etc.) who will be paying for their coffee order
Triggers	The users clicks on Create a Profile
Precondition	User downloaded the app or on the cafe's web app User has all the information necessary for creating a profile
Postconditions	The user's profile has been created or updated and approved by the cafe shop
Normal Flow	1. User clicks on create profile 2. The application will provide text fields for the user to enter proper information 3. The user will be asked to provide an email address, phone number, student ID, student full name, address, payment options 4. The user will be asked to create a unique username and a unique password with compliances to the password rules 5. The user can also add information of their favorite coffee/tea orders and other miscellaneous customization 6. Once the user had provided the proper information, there will be a system check and will either approve the user's provided information or prompt an error and how to fix the error
Alternative Flow	1. A user that is returning to the app, with previous user login
Exceptions	❖ If user left anything blank in the necessary text fields, display error ❖ If user password is not good, display error ❖ Duplicate profiles with same login or user id, display error
Priority	Very high, user will not receive order without proper profile
Frequency of use	Every time the user is ordering from the app
Business Rules	The cafe shop
Other Information	None
Assumptions	None

Case 2 Profile Flowchart



Case 2 Profile UML



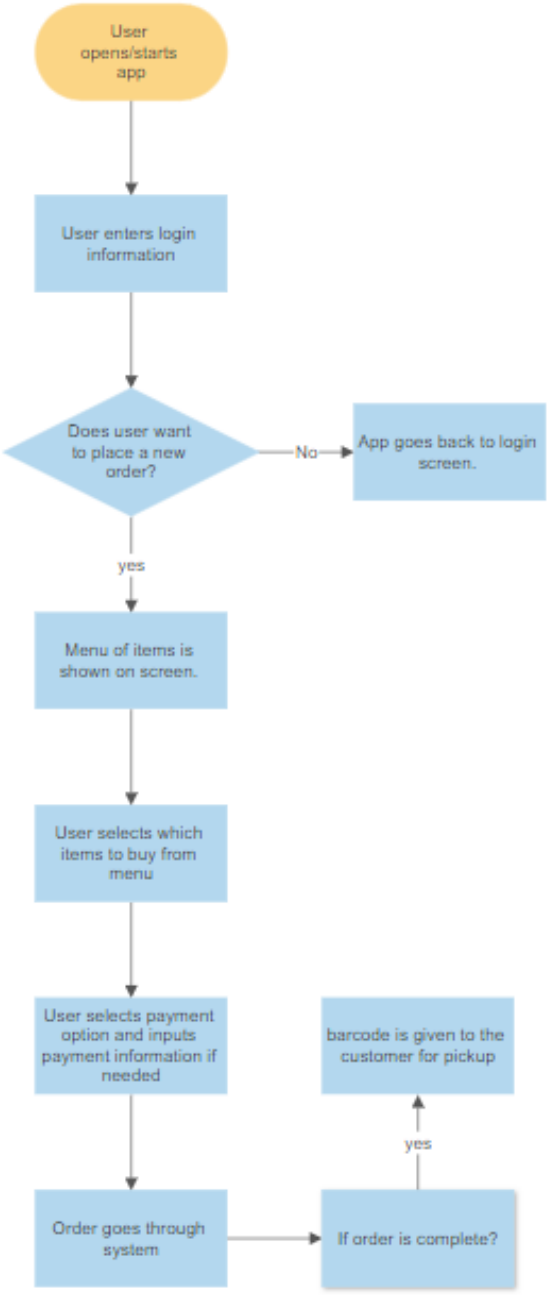
Case 3 Order Description

After the user is logged in with their profile, or if they don't have a profile, they will make one and then they can proceed to make an order. When they click on new order button, then a list of items will appear on the screen, where a customer can select which items they want to order. After the user selects which item or items to order, then it will proceed to the payment method, asking users for their payment of their order. After the payment has been made, the order goes through system, and the workers are alerted of the order. After the order is made, and ready for the customer to be picked from them. A barcode is sent to the customer. That barcode is used to for pickup of the order.

An object will be created each time the customer creates an order. The object is then sent to the SQL database.

Case 3 Order Table	
ID	Orders_Case_3
Created by	Bilal Qureshi
Date Created	October 4th, 2019
Description	When a customer wants to order items from the menu, they can select which items to buy and then proceed with payment.
Primary Actors	Customer, workers
Triggers	Customer clicks on the order button
Precondition	1. Items are available to order. 2. Cafe is open to complete and fulfill order 3. Customer profile is made
Postconditions	1. Payment is made for order 2. Order is then sent to cafe for it to be made
Normal Flow	1. Customer signs in with their user profile 2. Customer clicks on orders button 3. Program shows list of items that customers can order 4. Customer selects item to buy. 5. Customer selects payment option and inputs payment information 6. Order goes through system 7. Workers are alerted for order. 8. If order is ready, barcode is given to the customer for them to scan to get their items.
Alternative Flow	User can order from history
Exceptions	1. If there is no menu is available. 2. If there's an item that is not available to order. 3. Customer has incorrect payment information
Priority	High
Frequency of use	Everytime a customer orders an item, on average between 1-2 items per order.
Business Use	None
Other Information	Menu is ready to be used by customers
Assumption	Cafe is ready to take orders User profile is made

Case 3 Order Flowchart



Case 3 Order UML

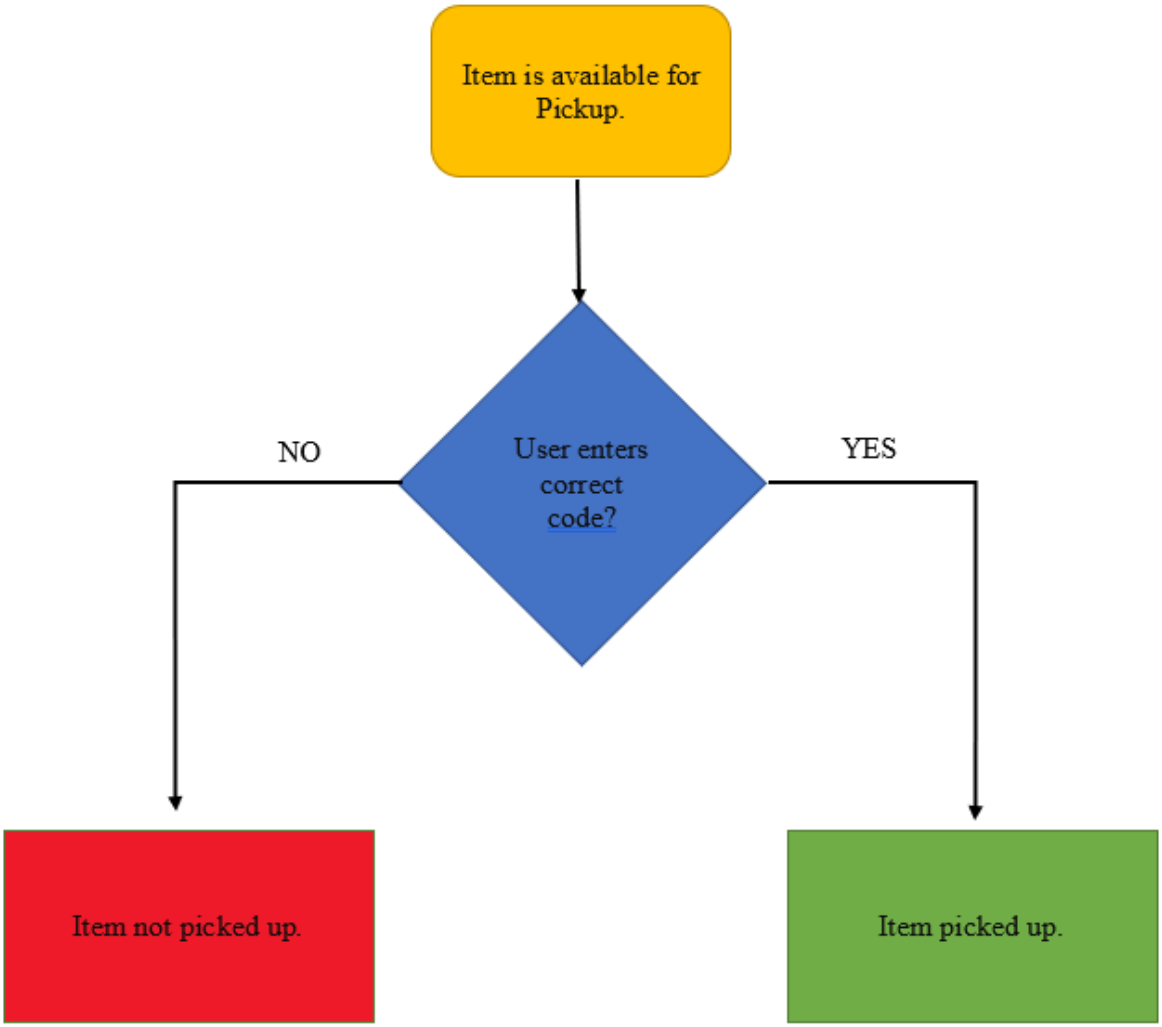
Order
paymentID: int orderNum: int
setOrderNum: int setOrderReady(): Boolean setPickUpID(): int setLockNum: int

Case 4 Pickup Description

After purchase or order is made, users will receive a notification on their phone that their order is ready for pickup. From there they will get an order number. The order number will be their key that is used for opening their locker for pickup. When user is at locker use the kiosk to scan order QR code or punch in code. After correctly putting in order key at kiosk, locker number is displayed on screen and locker will be unlocked.

Case 4 Pickup Table	
ID	Pickup_Case_4
Created by	Daniel Ontiveros, Bilal Qureshi, Samuel Puchinskiy
Date Created	10/08/2019
Description	Lets users know when their order is ready for pickup at the cafe/locker.
Primary Actors	All users
Triggers	When system prompts order is ready
Precondition	Order must have been already paid for.
Postconditions	None
Normal Flow	- Log in. - Select item(s) - Place item(s) in cart - Pay for order. - Pick up at locker with code when order is ready.
Alternative Flow	- Same as above but without login.
Exceptions	Site is down or payment cannot go through
Priority	High
Frequency of use	Daily
Business Use	Daily
Other Information	None
Assumptions	None

Case 4 Pickup Flowchart



Case 4 Pickup UML

Pickup
pickupID: int Pickup: Boolean
set(): int setOrderReady(): Boolean getOrderReady(): Boolean setpickupID(): int getpickupID(): int

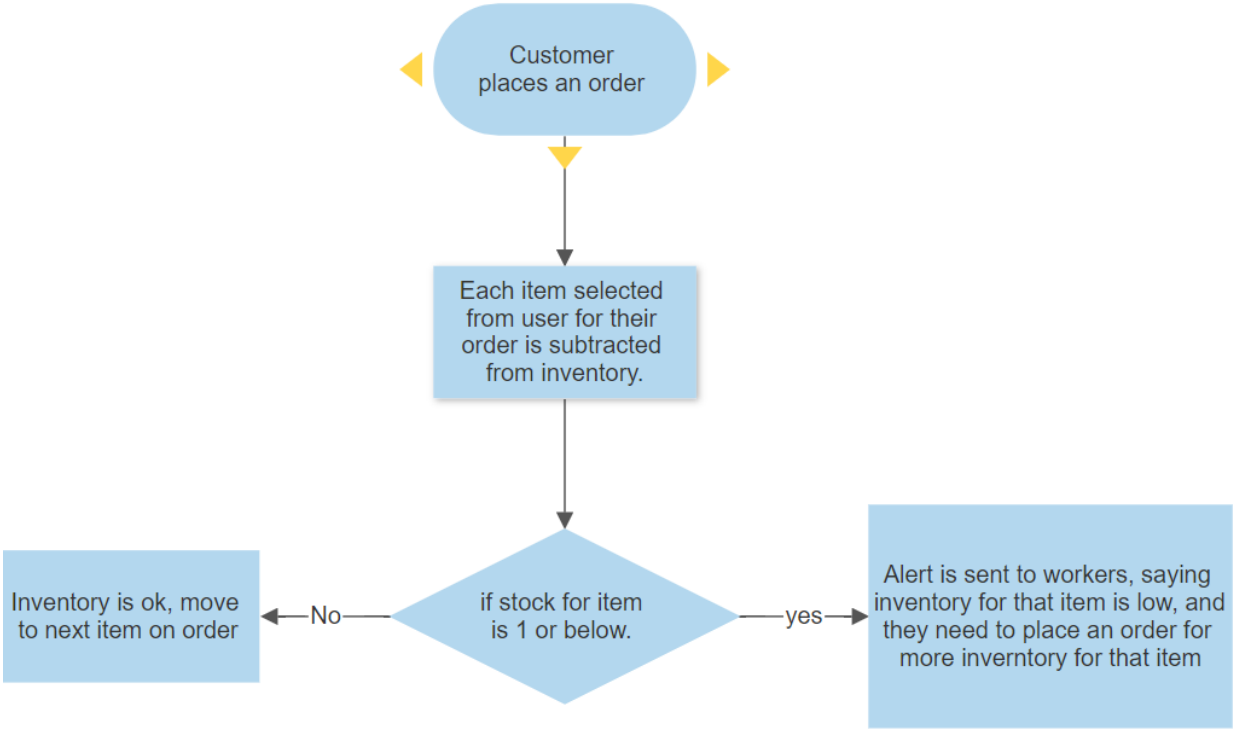
\

Case 5 Inventory Description

The inventory contains the item and the quantity of each item. When a customer places an order, each item on that order is subtracted from inventory. If stock for an item is 1 or below, then an alert is sent to the workers, saying inventory is low, and they need to place an order for inventory for that item, so that way inventory for that item does not go down to zero. If it goes to zero, and a customer wants to order that item, then customer will not be able to order it because they is nothing to order. But if the stock for that item is two or above, then inventory for that item is ok. Then it checks the next item and the inventory quantity for that item.

Case 5 Inventory Table	
ID	Inventory_Case_5
Created by	Bilal Qureshi
Date Created	October 8, 2019
Description	When a customer ordered an item, then after the quantity of that item is subtracted by one, if it is below a certain number it should send an alert to the workers that stock is low on a certain item
Primary Actors	Cafe workers, customer
Triggers	When an item is below a certain quantity number
Precondition	1. Stock of an item is at a very low level.
Postconditions	1. Alert is sent to workers 2. An order of items needed is placed
Normal Flow	1. A customer places an order 2. Inventory stock is then subtracted because item is used for customer's order. 3. If stock is at 2 or more than inventory is ok at the moment. 4. If stock for item is 1 or below than, alert is sent to workers saying inventory for that item is low, need to place an order for item
Alternative Flow	None
Exceptions	1. If inventory is already at 0 or negative for an item.
Priority	Medium
Frequency of use	Every time a customer orders an item, on average between 1-2 items per order.
Business Rules	None

Case 5 Inventory Flowchart



Case 5 Inventory UML

Inventory
getItemID Item: quantity: int Description: string
setItemQuantity: void getItemQuantity: Int

Item
itemID: int itemName: String itemDescription: String itemPrice: Double(4,2)
getItemId(): int setItemId(): void getItemDescription(): String setItemDescription(): void

Case 6 History Description

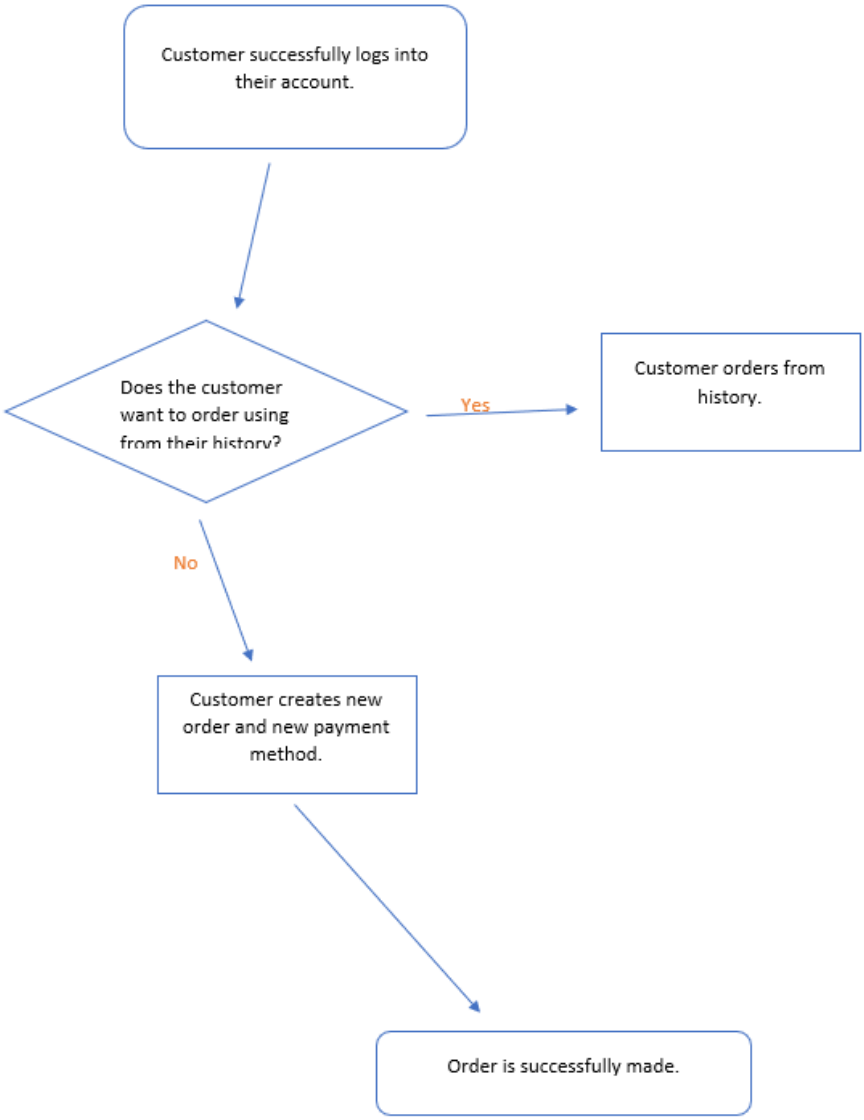
The history will allow the student or staff member to use their previous payment option to use again for their new order. It will always take time registering a new card to the app, so to make things efficient, the customer may purchase their new order with the same payment card. This will same time and will be less of a hassle than to always add a new card.

In addition to saving previous payment methods, the history case will also save previous orders. If a customer orders the same coffee, then they can simply choose from the history. Instead of always creating a new order, the customer can order from the history. This will also save time and hassle to always reorder the same order.

The History database will extract data from the payment database and the order database to create the History database. This database will be the left and right join of the payment and order database. By implementing the database with two joins, then we will save memory and resources

Case 6 History Table	
ID	History_Case_7
Created by	Daniel Ontiveros, Bilal Qureshi, Samuel Puchinskiy
Date Created	10/08/2019
Description	Saved orders, in case users would like to order the same items again
Primary Actors	Users (Instructors, Students, factory members, etc.) who want to view their past orders and want to reorder the same items again.
Triggers	When user click history button.
Precondition	User downloaded the app or on the cafe's web app User's profile been fully created Users has previous orders saved in history.
Postconditions	Everytime a user orders, then the order is saved in history User is able to view history User is able to order from history
Normal Flow	1. User logins in account successfully 2. User clicks on history 3. User views history 4. User reorders from history
Alternative Flow	If there are no previous orders in history. For instance, if user is using the app for the first time or deleted their order history.
Exceptions	If there history is empty.
Priority	Low
Frequency of use	Everytime the user is on the app.
Business Rules	None
Other Information	None
Assumptions	None

Case 6 History Flowchart



Case 6 History UML

Item
userHistoryID: int; userHistory: String; userPaymentObject:Object; userOrderObject: Object;

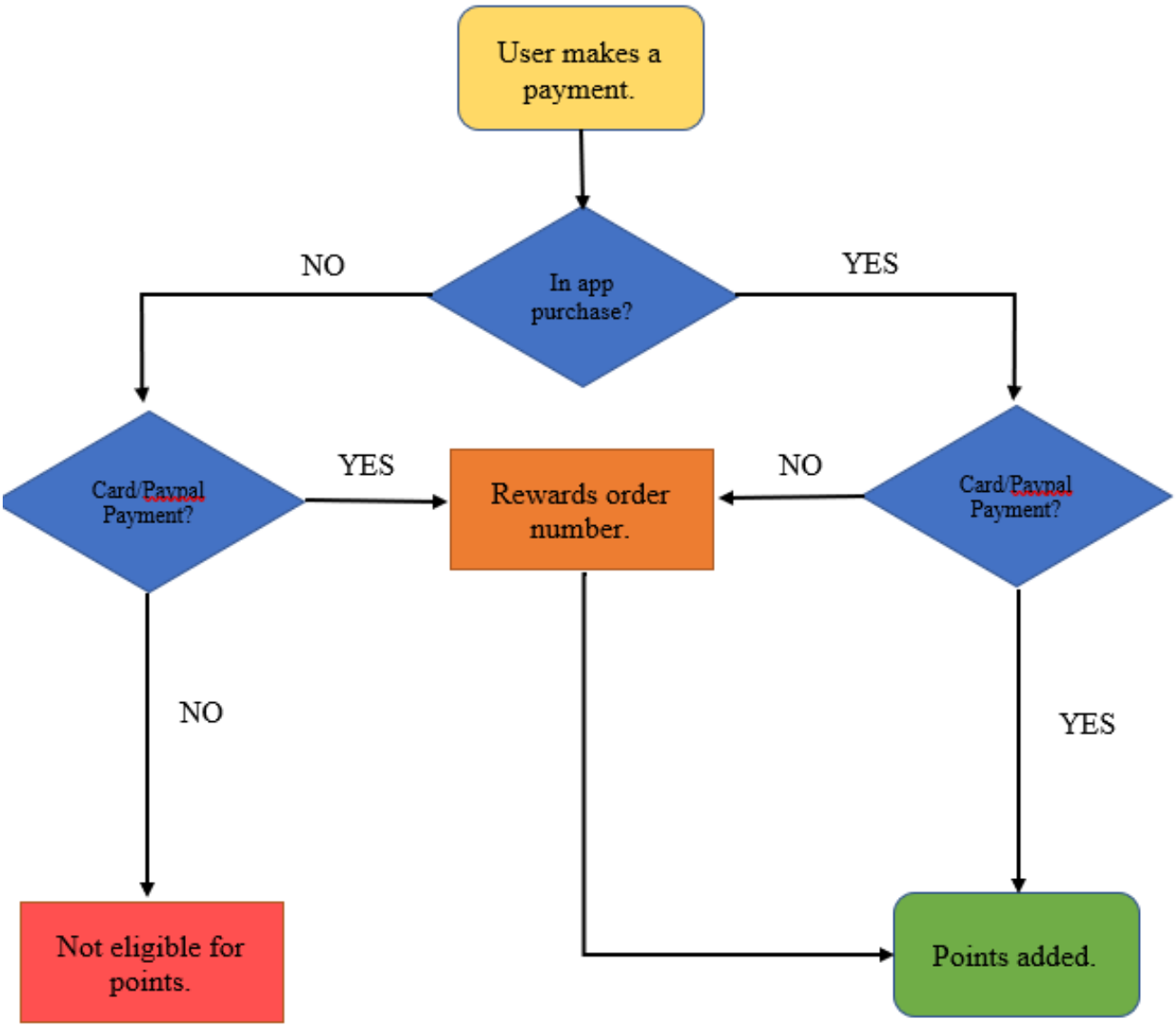
getuserHistoryID(): int
setuserHistoryID(): void
getuserHistory(): String
setuserHistory(String): void
getuserPaymentObject(): Object
setuserPaymentObject(Object): void
getuserOrderObject(): Object;
setuserOrderObject(Object) void;

Case 7 Rewards Description

Users who have already created a login will automatically be enrolled into the CafeGo Rewards program. This will let the user accumulate points through purchases, which can later be redeemed for select items from the menu. Points must accumulate to a certain limit to be eligible for redemption. Points are equivalent to every dollar spent, any decimal value will be rounded to the nearest dollar value point. For example, 1 point awarded if purchase total is less than or equal to $1.49, 2 points awarded if purchase total is greater than or equal to $1.50. Points can only reach a maximum value of 25 which will add a $5 value will be credited to your account. Any purchases made outside of app can be claimed within 5 days of purchase date if the card or paypal that paid for items on receipt matches card or paypal account on file in the app. Any purchases made with cash are eligible for redemption if order was made from within the app. Any purchases made with rewards credits will not accumulate points.

Case 7 Rewards Table	
ID	Rewards_Case_7
Created by	Daniel Ontiveros
Date Created	October 6th, 2019
Description	Point accumulation to claim rewards for orders.
Primary Actors	Users with a login account
Triggers	Completion of purchase from cart. Receipt redemptions for valid purchases.
Precondition	Must have a profile created.
Postconditions	None
Normal Flow	-Log in to site -Place item in cart. -Pay for item. -Points will be added.
Alternative Flow	None
Exceptions	Payment declined, Cash payment made outside app.
Priority	Normal
Frequency of use	Unlimited
Business Use	Daily
Other Information	None
Assumptions	None

Case 7 Rewards Flowchart



Case 7 Rewards UML

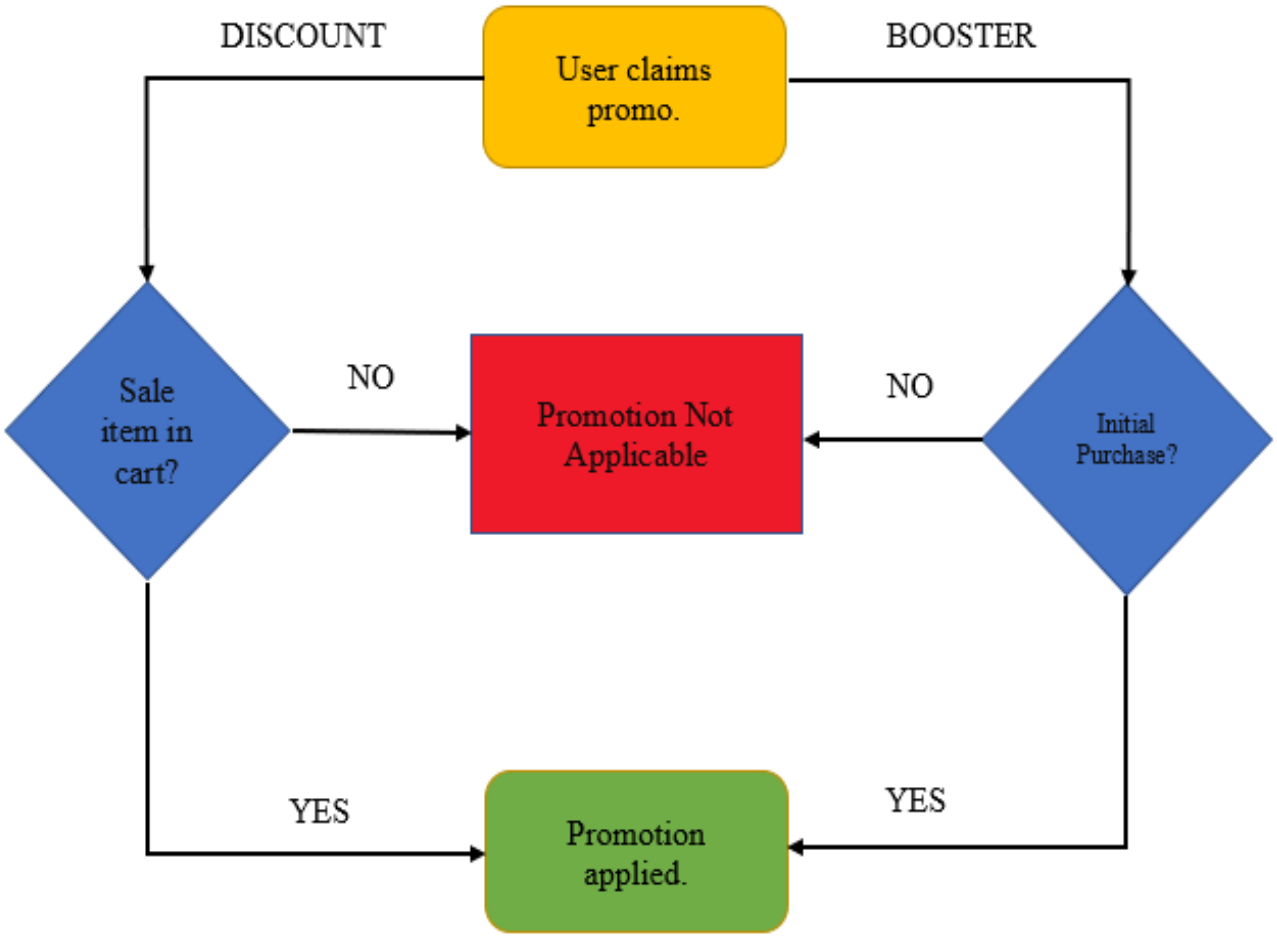
Rewards
userRewardsID: int amount: int hasRewards: boolean
getuserRewardsID(): int setuserRewardsID(int): void getamount(): int setamount(int): void gethasRewards() int sethasRewards(int): void

Case 8 Promotion Description

Users with accounts will get access to exclusive weekly promotions that will give items at a discounted price or occasionally a points boost. Promotions will be given from app and will only be allowed for a one time use per day of promotion. Point boost promotion will increase rate by 0.5% and can only be applied to initial purchase. Once used cannot be applied to any purchases made after. All promotions have an expiration time and can only be used during the specific week they were distributed. Promotions do not stack or combine.

Case 8 Promotion Description Table	
ID	Promotions_Case_8
Created by	Daniel Ontiveros
Date Created	October 6th, 2019
Description	Lets users know what weekly item is available for promotions.
Primary Actors	Users who have already established a profile and are logged in.
Triggers	Logging in to site with profile.
Precondition	User must have a profile already created to have access to promotions during activity hour.
Postconditions	Must have ordered promotion item.
Normal Flow	-Log in to site. -Select promotion item -Discount will appear in cart.
Alternative Flow	None
Exceptions	App errors or item out of stock.
Priority	Normal
Frequency of use	1 promotion item per day during the promotion week.
Business Use	Weekly
Other Information	Some promotion items may come back in circulation.
Assumptions	None

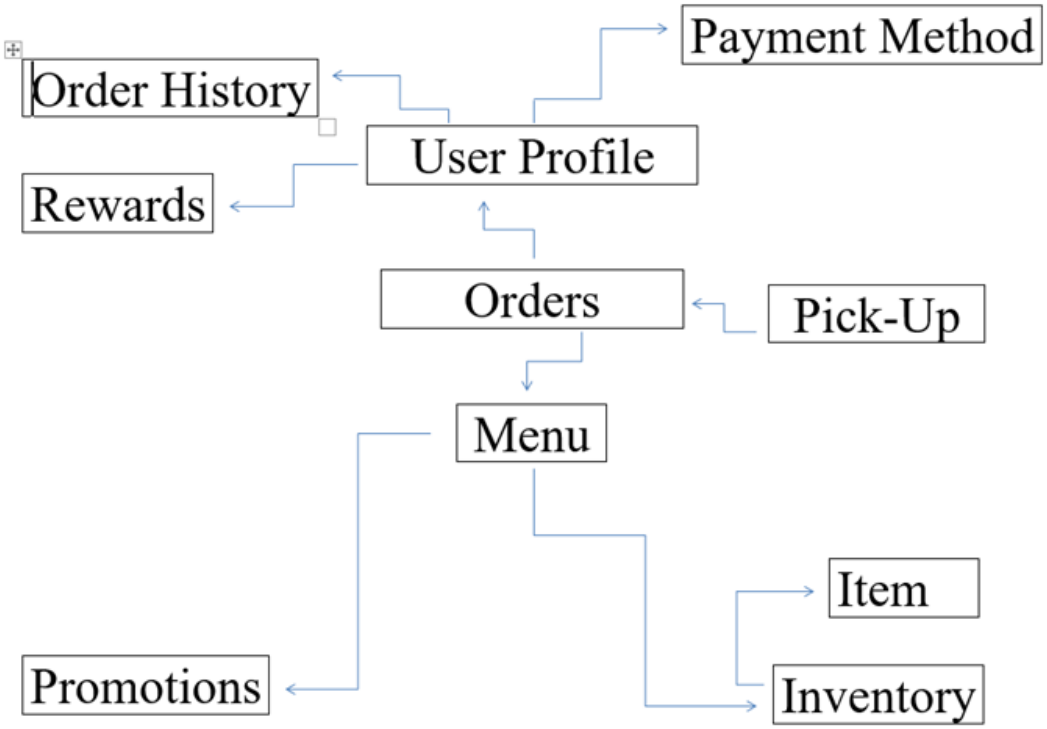
Case 8 Promotion Flowchart



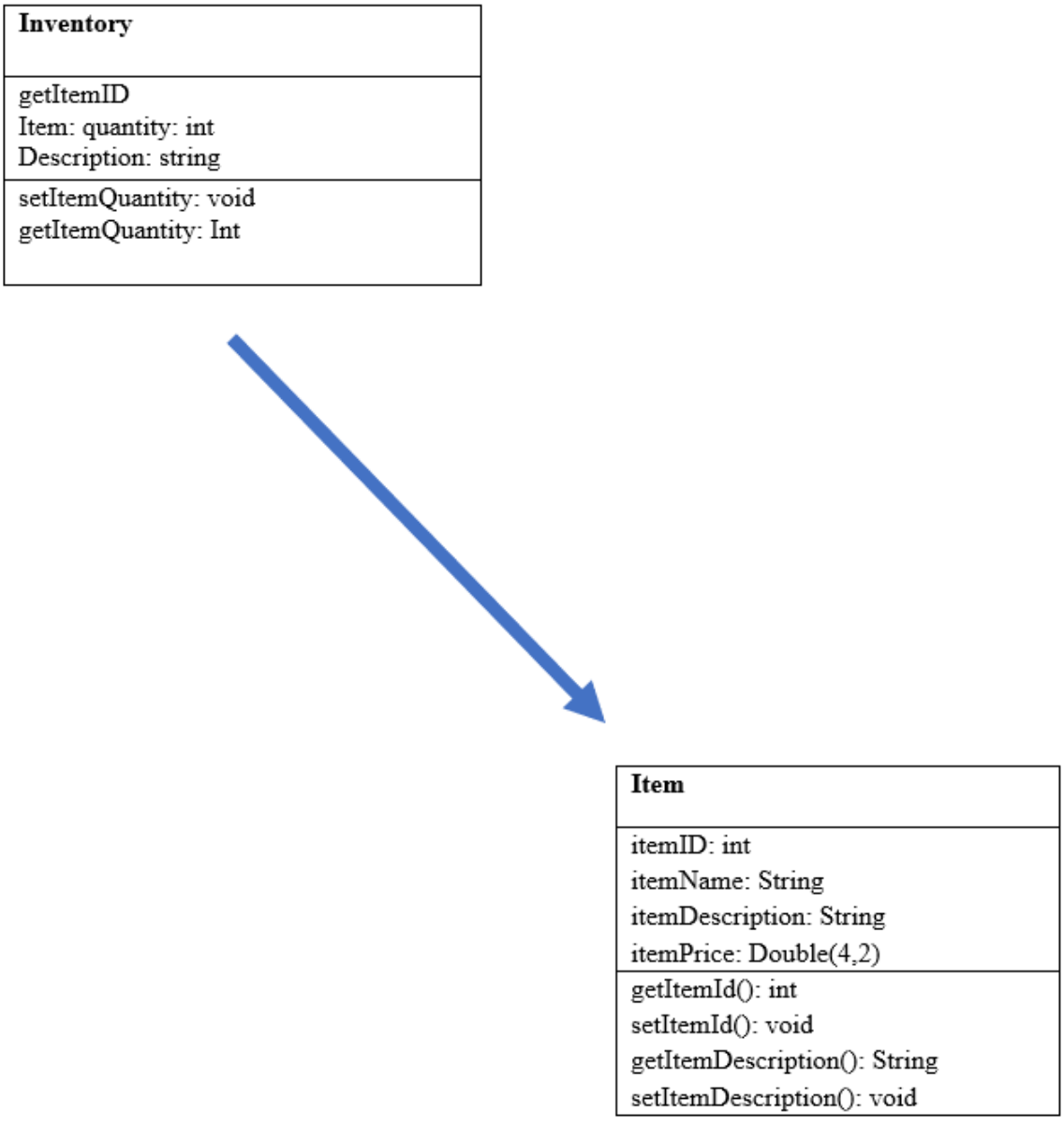
Case 8 Promotion UML

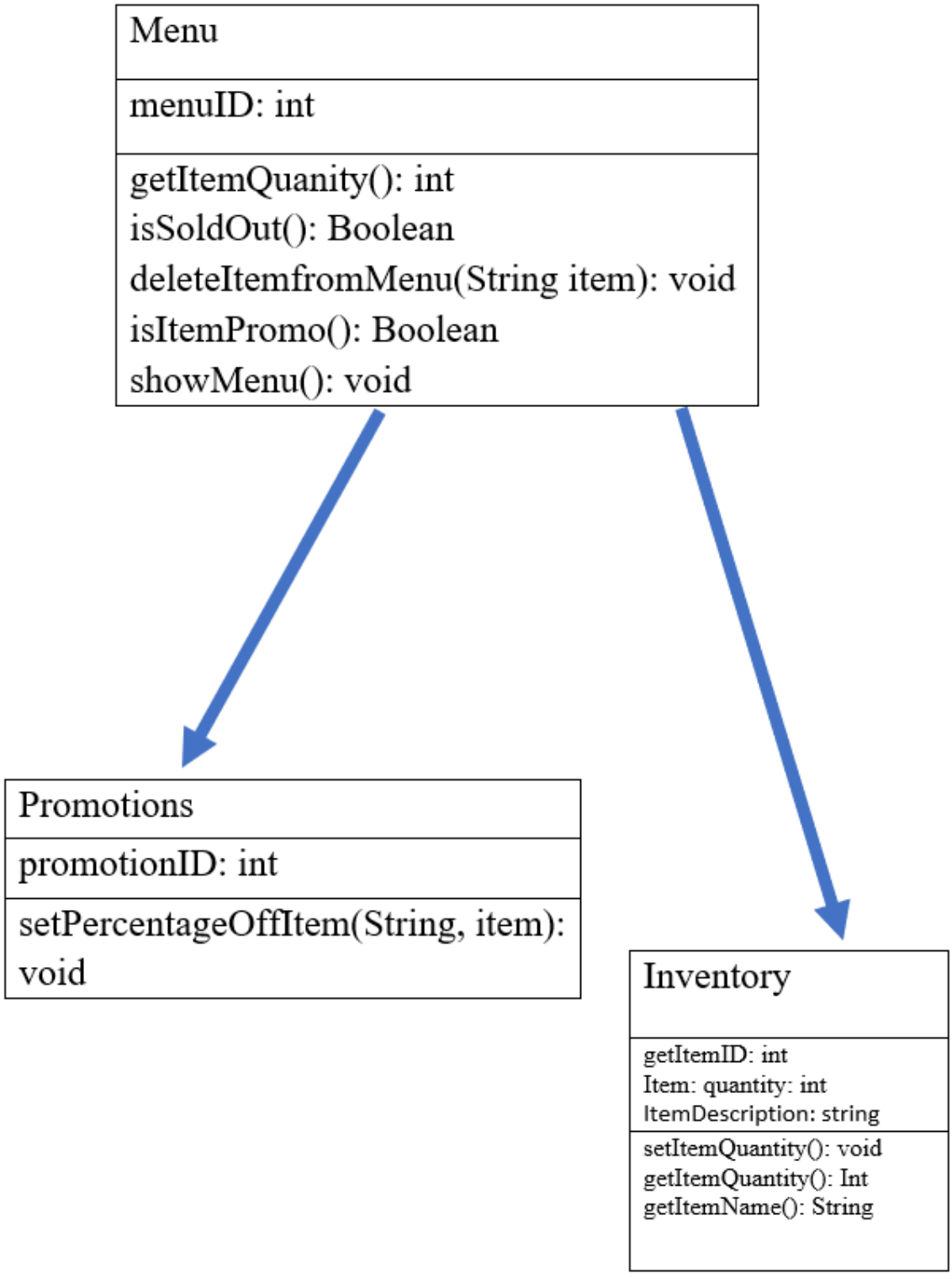
Promotion
userPromotionID: int amount: int hasPromo: boolean
getuserPromotionID(): int userPromotionID(): void getamount(): int setamount(int): void gethasPromo(): boolean sethasPromo(boolean): void

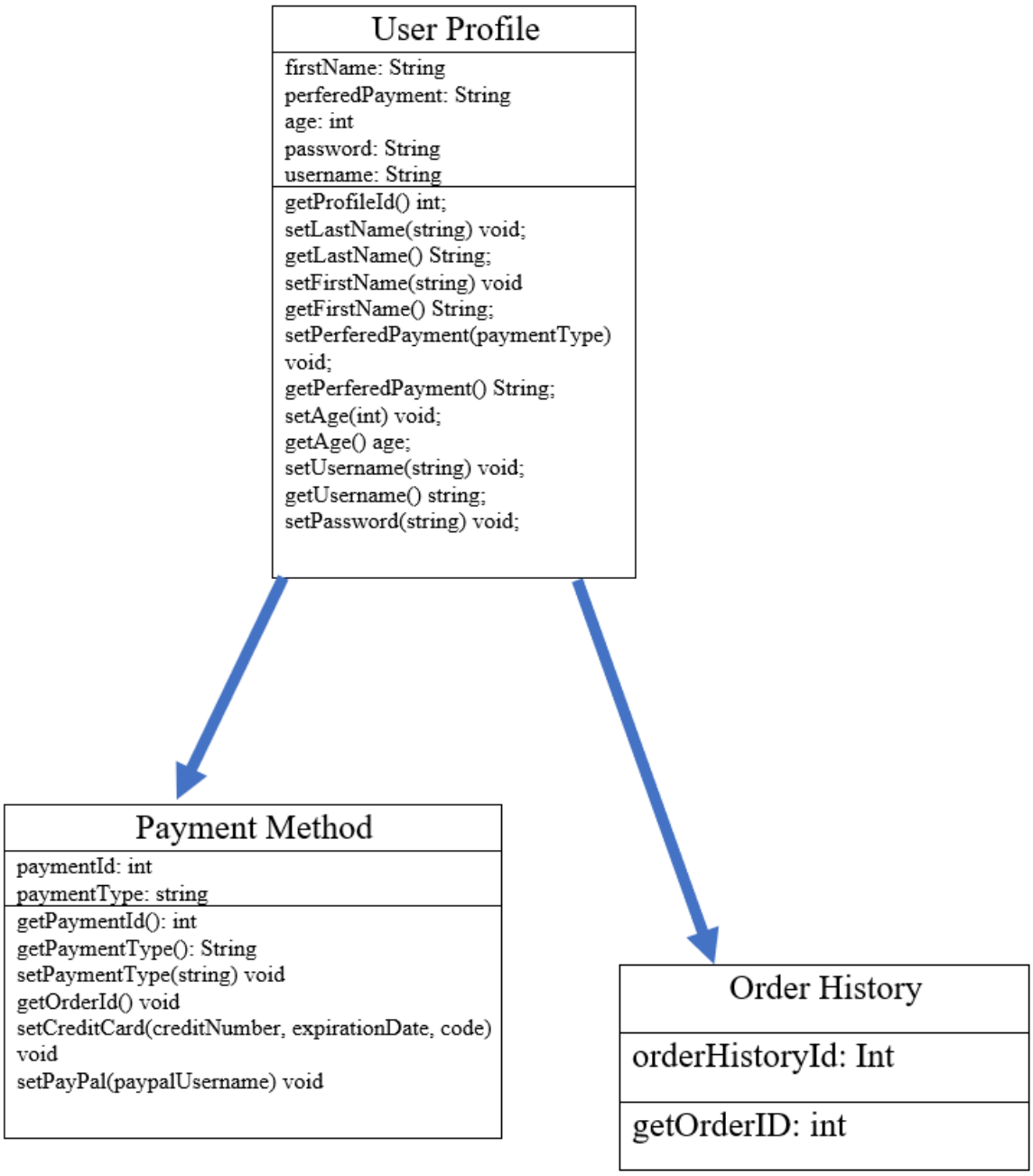
Data Model Diagram

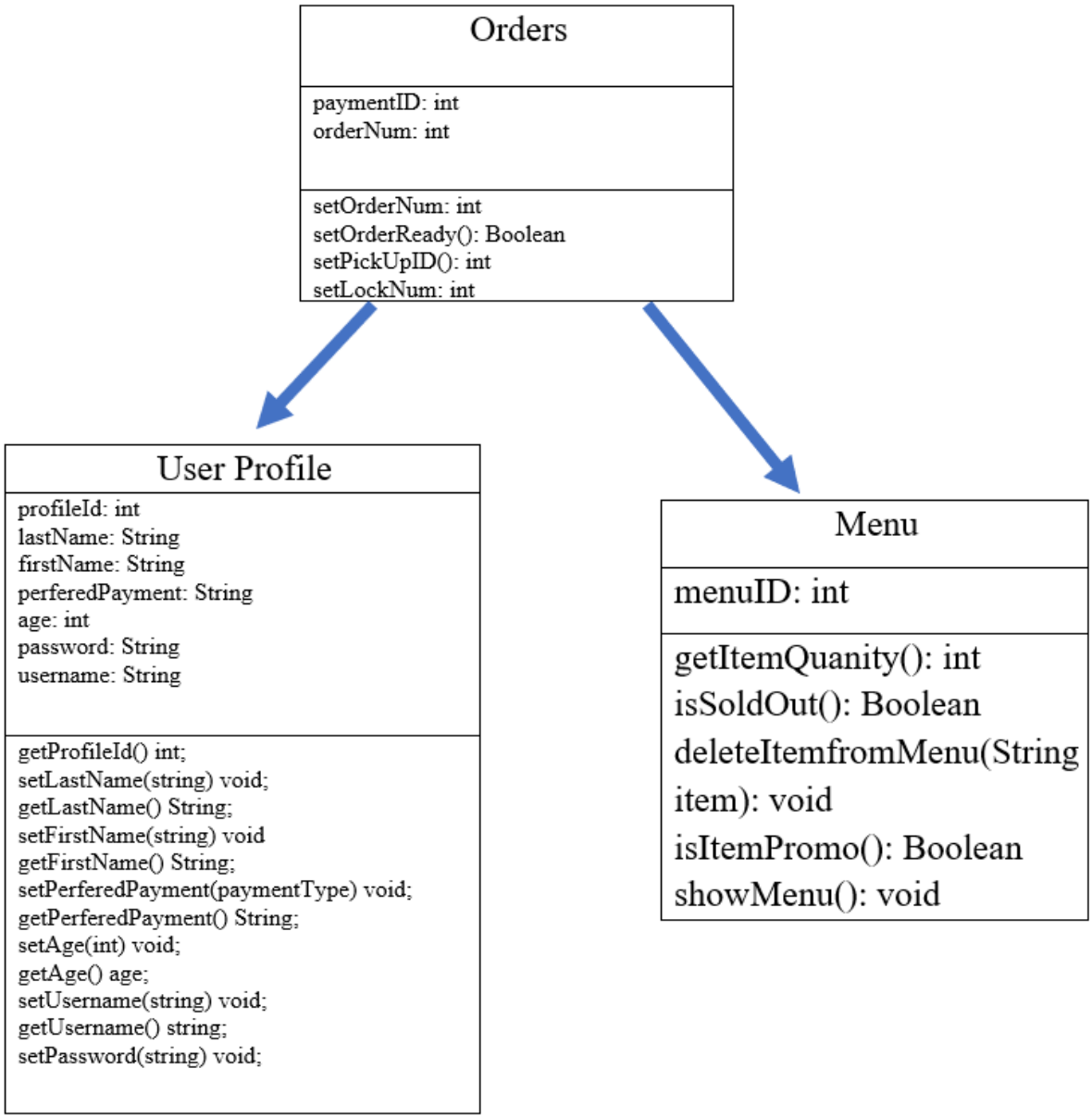


Sequence Diagrams




Menu
menuID: int
getItemQuanity(): int
isSoldOut(): Boolean
deleteItemfromMenu(String item): void
isItemPromo(): Boolean
showMenu(): void
Promotions
promotionID: int
setPercentageOffItem(String, item): void
Inventory
getItemID: int
Item: quantity: int
ItemDescription: string
setItemQuantity(): void
getItemQuantity(): Int
getItemName(): String


User Profile
firstName: String
perferedPayment: String
age: int
password: String
username: String
getProfileId() int;
setLastName(string) void;
getLastName() String;
setFirstName(string) void
getFirstName() String;
setPerferedPayment(paymentType) void;
getPerferedPayment() String;
setAge(int) void;
getAge() age;
setUsername(string) void;
getUsername() string;
setPassword(string) void;
Payment Method
paymentId: int
paymentType: string
getPaymentId(): int
getPaymentType(): String
setPaymentType(string) void
getOrderId() void
setCreditCard(creditNumber, expirationDate, code) void
setPayPal(paypalUsername) void
Order History
orderHistoryId: Int
getOrderID: int


Orders
paymentID: int
orderNum: int
setOrderNum: int
setOrderReady(): Boolean
setPickUpID(): int
setLockNum: int
User Profile
profileId: int
lastName: String
firstName: String
perferedPayment: String
age: int
password: String
username: String
getProfileId() int;
setLastName(string) void;
getLastName() String;
setFirstName(string) void
getFirstName() String;
setPerferedPayment(paymentType) void;
getPerferedPayment() String;
setAge(int) void;
getAge() age;
setUsername(string) void;
getUsername() string;
setPassword(string) void;
Menu
menuID: int
getItemQuanity(): int
isSoldOut(): Boolean
deleteItemfromMenu(String item): void
isItemPromo(): Boolean
showMenu(): void

Index

Glossary

All of the following definitions are not our words and was copied from the "Glossary of Requirements Engineering Terms from *Software Requirements, Third Edition* by Karl Wiegers and Joy Beatty (Microsoft Press, 2013)".

activity diagram: An analysis model that depicts a process flow proceeding from one activity to another. Similar to a flowchart.

actor: A person performing a specific role, a software system, or a hardware device that interacts with a system to achieve a useful goal. Also called a *user role*.
application: See *product*.

business analytics system: A software system used to convert large and complex data sets into meaningful information from which to make decisions.

class: A description of a set of objects having common properties and behaviors, which typically correspond to real-world items (persons, places, or things) in the business or problem domain.

class diagram: An analysis model that shows a set of system or problem domain classes, their interfaces, and their relationships.

customer: An individual or organization that derives either direct or indirect benefit from a product. Software customers might request, pay for, select, specify, use, or receive the output generated by a software product.

data flow diagram: An analysis model that depicts the processes, data stores, external entities, and flows among them that characterize the behavior of data flowing through business processes or software systems.

decision rule: An agreed-upon way by which a body of people arrives at a decision.

dependency: As used in requirements specification, a reliance that a project has on a factor, event, or group outside its control.

entity: An item in the business domain about which data is collected and stored.

entity-relationship diagram: An analysis model that identifies the logical relationships between pairs of entities. Used for modeling data.

exception: A condition that can prevent a use case from concluding successfully. Unless some recovery mechanism is possible, the use case's postconditions are not reached and the actor's goal is not achieved.

flowchart: An analysis model that shows the processing steps and decision points in the logic of a

process. Similar to an activity diagram.
gold plating: Unnecessary or excessively complex functionality that is specified or built into a product, sometimes without customer approval.

include relationship: A construct in which several steps that recur in multiple use cases are factored out into a separate sub-use case, which the other use cases then invoke when needed.

issue, requirement: A defect, open question, or decision regarding a requirement. Examples include items flagged as TBD, pending decisions, information that is needed, and conflicts awaiting resolution.

iteration: An uninterrupted development period, typically one to four weeks in duration, during which a development team implements a defined set of functionality selected from the product backlog or baselined requirements for the product.

navigation map: See *dialog map.*

peer review: An activity in which one or more persons other than the author of a work product examine that product with the intent of finding defects and improvement opportunities.

postcondition: A condition that describes the state of a system after a use case is successfully completed.

precondition: A condition that must be satisfied or a state the system must be in before a use case can begin.

procedure: A step-by-step description of a course of action to be taken to perform a specified activity, describing how the activity is to be accomplished.

process: A sequence of activities performed for a particular purpose. A *process description* is a documented definition of those activities.

process flow: The sequential steps of a business process or the operations of a proposed software system. Often represented by using an activity diagram, flowchart, swimlane diagram, or other modeling notation.

product: Whatever ultimate deliverable a project is developing. In this book, product, application, system, and solution are used interchangeably.

prototype: A partial, preliminary, or possible implementation of a software system. Used to explore and validate requirements and design approaches. Types of prototypes are evolutionary and throwaway; paper and electronic; and mock-up and proof-of-concept.

requirement: A statement of a customer need or objective, or of a condition or capability that a product must possess to satisfy such a need or objective. A property that a product must have to provide value to a stakeholder.

requirement attribute: Descriptive information about a requirement that enriches its definition beyond

the statement of intended functionality. Example attribute types are origin, rationale, priority, owner, release number, and version number.

reuse, requirements: The act of using existing requirements knowledge in multiple systems that share some similar functionality.

scenario: A description of a specific interaction between a user and a system to accomplish some goal. Alternatively, an instance of usage of the system, or a specific path through a use case.

scope: The portion of the ultimate product vision that the current project will address. The scope draws the boundary between what's in and what's out for a project that creates a specific release or for a single development iteration.

solution: All of the components delivered by a project to achieve a set of business objectives specified by an organization, including software, hardware, business processes, user manuals, and training.

system: A product that contains multiple software and/or hardware subsystems. Colloquially, *system* also is used interchangeably in this book with *application*, *product*, and *solution* to refer to whatever software-containing deliverable a team is building.

system requirement: A high-level requirement for a product that contains multiple subsystems, which could be all software or software and hardware.

TBD: Abbreviation for to be determined. TBD serves as a placeholder when you know you are missing some requirements information. See *issue, requirement*.

template: A pattern to be used as a guide for producing a complete document or other item.

UML: An abbreviation for the Unified Modeling Language, which describes a set of standard notations for creating various visual models of systems, particularly for object-oriented software development.

usage scenario: See *scenario*.

use case: A description of a set of logically related possible interactions between an actor and a system that results in an outcome that provides value to the actor. Can encompass multiple scenarios.

use case diagram: An analysis model that identifies the actors who can interact with a system to accomplish valuable goals and the various use cases that each actor might be involved with.

user: A customer who will interact with a system either directly or indirectly (for example, by using outputs from the system but not generating those outputs personally). Also called *end user*.

user requirement: A goal or task that specific classes of users must be able to perform with a system, or a desired product attribute. Use cases, user stories, and scenarios are common ways to represent user requirements.

verification: The process of evaluating a project deliverable to determine whether it satisfies the specifications on which it was based. Often stated as "Are we building the product right?"

vision: A statement that describes the strategic concept or the ultimate purpose and form of a new system.

vision and scope document: A collection of the business requirements for a new system, including business objectives, success criteria, a product vision statement, and a project scope description.

work product: Any interim or final deliverable created for a software project.

Work Cited

Glossary of Requirements Engineering Terms

Software Requirements, Third Edition by Karl Wiegers and Joy Beatty (Microsoft Press, 2013)