Software Design Specifications

Prepared by Ethan Fox, Jason Fabian, and Alan Yin

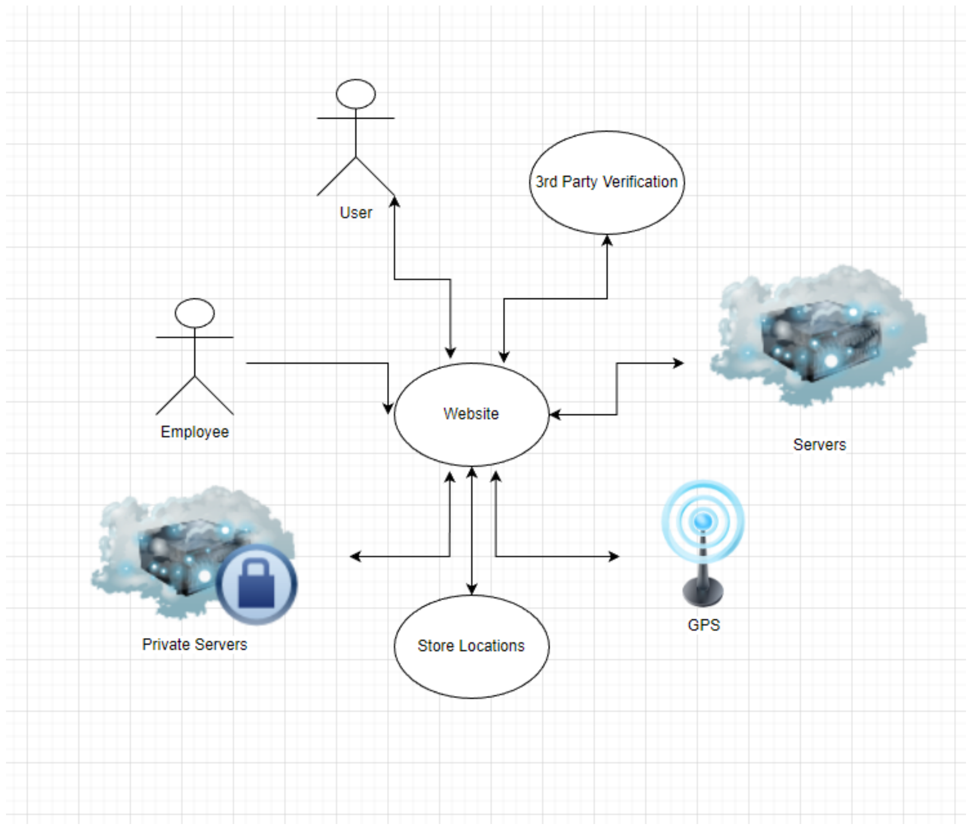
https://docs.google.com/document/d/1_LcDUKS_PxQ1RZAfRb8-GWF3s9dhu68_4FyvoHaFaG8/edit?usp=sharing

https://drive.google.com/file/d/1nOUxviw34cOgIJ3-jzb6wMNXtBK9e-8X/view?usp=sharing

System Description

The system requested is a online clothing store page. The client wants to utilize the page to allow online ordering and shopping at their shopping brand for various locations. It will include all shopping locations and be available to all customers for ordering as employees will utilize it to keep it updated with what’s in stock.

Software Architecture Overview

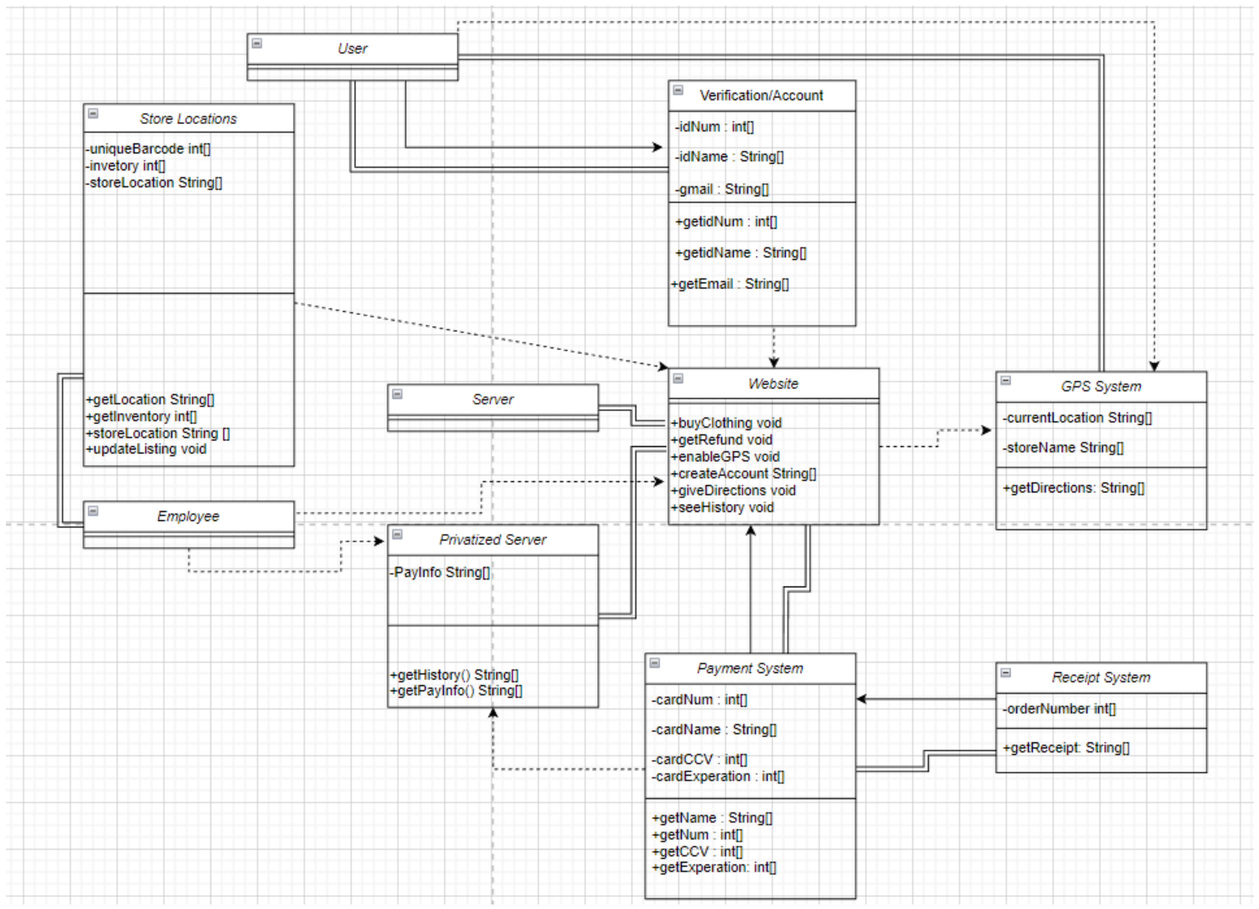


At the center of the software architecture, there is the website. That is where everything is connected and communicates with. For example, the user and website communicate back and forth because they are both providing each other with information. The website also does this with the 3rd party verification by sending and receiving each other verifications. Similarly, the website sends and inquires for information stored in the server. The GPS is constantly communicating back and forth with the website because the location is constantly updating. The store location also communicates a lot with the website because they're telling each other when someone buys something, or if there is something in stock still. The private server is the same as the server, but it just contains more sensitive information that needs to be stored privately. The employee is different because they're only sending information into the website, they do not receive information back from the website. That is because they use the store location in order to access information from the website.

https://drive.google.com/file/d/1nOUxviw34cOgIJ3-jzb6wMNXtBK9e-8X/view?usp=sharing

https://drive.google.com/file/d/1nugGaZDkeE5yriZS7KYrh8TE32CyDjAc/view?usp=sharing

UML Diagram



Similarly to the Software Architecture Overview, at the center of the UML Diagram is the website. The website uses the GPS System in order to get the users location and communicate with the user direction or whatever they need back to them. The website also has its own server and privatized server to store information in. It also has a payment system in the website in order that users can purchase things from the website. The payment system also uses the privatized server in order to store the sensitive payment information. It contains a receipt system as well in order to get the user a receipt. Another large factor in the UML diagram is the user, which has a GPS system and an account. The GPS system communicates the user's location constantly, and the account helps the user confirm payments and save information. The difference between the two is that the user uses a GPS System, but has an actual account for the website. The store location also is important, and it uses the website a lot in order to send information. It also has an employee class, which is also constantly communicating with the website.

Description of Classes

To begin with, our UML Diagram has 10 elements: user, account, website, server, privatized server, payment system, receipt system, GPS system, stores, and employee (admin) account.

Our UML Diagram begins with the **user**, which is representative of the user and in it we define what the user interacts with. The user uses their account and information in order to access the website. Furthermore the website will later use the user's GPS system in order for the user to receive directions to specific store locations.

Moving on we have the **account** class. This is representative of the user's identity / accumulation of information related to the user. While the user is logged into their account they will be able to access and use the website classes' functions.

The **website** class is the core of the diagram, it's the central hub in which all of the classes are related in one way or another. It serves as the interface between the user (public) classes and the "backdoor" (private) functions. The website is directly connected to the **server** which is used to store / host the website on the internet. The server is used to store information regarding the website / clothing stores such as inventory, number of accounts, etc. (less sensitive data). Also connected to the website is the **privatized server**, which will be used to store *sensitive information* for instance purchase history, transaction details, etc.

The website will have a class embedded into it called the **payment system** which will be a class that will allow the user to pay for clothing that they have "put in their cart" through the website's functions. The payment system will be directly connected to the privatized server as a way to store sensitive information. The only way it can be accessed is through the **employee** class which represents the employee / admin accounts. These classes are directly connected to stores since the employees will be the only one to have access to these accounts.

Another class connected to the payment system is the **receipt system** which is directly connected to the payment system only as it’s embedded / a part of the payment system. This class is expected to be a method for users to receive a receipt for their transactions via email.

The **Store Location** class is a class that is representative of specific physical store locations. It has employees with admin accounts and additionally is directly connected to the website as a means to update inventory / provide directions to their store.

Following the previous idea of providing directions we have the **GPS** class which is the user’s own personal GPS devices. It’s intended to be unassociated with the clothing store / website itself and “piggyback” off of the user's own devices to provide them with directions to specific store locations.

Description of Attributes

Our software system has multiple different attributes where it will depend on the type of data that the system needs. While all elements are important, not all of them contain attributes, we will start with the **Payment system** which contains 4 different attributes. The first being cardNum: int[] which is used to store an integer which is the customer's card numbers, then we have cardCCV: int[] which stores an integer which is the CCV for the card, both of these are stored as an integer because the card number and the CCV are both numbers. The next is the cardName: Sting [] which will store the customer's name as a string because it is a name, and finally we have cardExperation: int[] that will store the date as an integer since the date is a number.

Next we have **Verification/Account**, where it contains idNum: int[] which will be stored as an integer because the ID number from the User has to be a number. Then we have idName: String[] that is stored as a string because it will be the ID name of the user so it can contain letter, number or even including symbols. Finally we have gmail: String[] which stores the gmail of the user and it is a string because there can be multiple letters, numbers and even symbols in gmail.

Then we have a **GPS System** which has a currentLocation String[] that will be the current location of the user and it would be stored as a String because locations can have numbers and letters in them. Then we got storeName String[] which will be different store names and it is stored as string because it may be composed of letters, numbers and even symbols.

The **Store Locations** where components such as the uniqueBarcode int[] which will be the barcode of each item and each item will have a different barcode from each other to identify each item and it can only be numbers so it is stored as an integer. Then we have inventory int[] which stores the amount of cloth we have in store so it has to be a number. Finally we got storeLocation String[] which is the location of the store so it has to be a String because it contains numbers and letters and maybe symbols.

Then we have the **Privatized Server** which only has PayInfo String[] which will store the payment information and since it may include multiple things such as the customer information and the card’s information it is stored as a String.

Finally we have **Receipt System** which only has orderNumber int[] which is the number of orders from the customer so it will be returning a number and when needed it will provide an order number for the system.

Operations Descriptions

The class with the most operations is the **Website**, containing six. The first one, buyClothing void, it is used to access the store location and communicate that a piece of clothing has been purchased. It is a void return type because it does not get any information, but just changes it in store location. The next operation is getRefund Void, which communicates with the payment system in cases where a customer needs to get a refund. It is void because it just pulls up the refund from the payment system where it can be found. enableGPS void is the next operation and that is primarily just making sure the user wants the GPS mode enabled. It doesn't return anything because it just toggles it on and off through the GPS System. The operation after that is createAccount String[], which is mainly for new customers and users. For those, it helps them create an account with the clothing store, and returns a string because it has a username and password associated with it. The next operation is giveDirections void, which prompts the GPS System to give the user directions to the store. The last operation is seeHistory void, which goes to the privatized server and pulls up the history that can be accessed there.

Payment System constraints four operations, the first of which being +getName : String[]. This operation gets the name that is associated with the credit card being used and gets it as a string. Along with the name, getNum : int[] gets the credit card's number and getCCV : int[] gets the CCV number of the credit card. Both of these two are returning integer values because credit card numbers and CCV's are only integers. The last one is +getExpiration : int[], which gets the expiration date in only integers, so MonthYear, like 082026.

The **Verification class** contains three operations, one int and two strings. The integer operation, getidNum : int[], gets the id number as an int forms the user and uses that to confirm with the website. The first string operation is the getidName : String[], and its main function is to get the name of the account from the website. The other string operation is the getEmail : String[], which gets the accounts email, from there it can verify that the user's account is valid.

The **GPS System** class only has one operation, that being getDirections : String[]. This operation accesses the location of the user and then gets directions for how the user can get to the store. It is returned as a string because it tells them what streets to take.

Store Locations contains four operations, getLocation, getInventory, storeLocation, and updateListing. The first one, getLocation String[], gets the users location from the GPS System that the website contains. It gets it as a string because it returns the location in terms of latitude and longitude, which contains North, East, South, and West which are all strings. The next operation is getinventory int[], which returns the inventory of the store. That helps to make sure that the items on the website are actually in stock at the store. The next operation is the +storeLocation String [], which gets where the specific store is located and is a string because it has the name of the area it is located in. The last operation is updateList, which just updates the inventory list when a purchase is made in order to keep track of what items the store currently has in stock.

Privatized Server has only two operations, both of which get string values. The first operation is getHistory String[] and its main function is to get the history of what has been purchased at the store previously. It returns string values because the history has names of clothes. The other operation is getPayInfo() String[], which has a main function to get the payment information previously used at the store. It returns a string because there is a name associated with the payment information so it has to be a string.

Receipt System has only one operation, getReceipt : String[]. This operation gets the receipt that the user gets after they purchase something from the store. It returns a string because there are words on the receipt so it can’t just be an int.

Development plan and timeline

As a group we are going to all work together on the UML Class Diagram and Architectural Diagram of all major components. We will finish those two diagrams by the latest March 6th. After that we will partition the remaining parts to each person individually, so one person will do the class descriptions, another will do the description of attributes, and the last will do the description of the operations. These descriptions will be finished by March 9th to allow us to review the entire document one last time before the due date. That makes all of us responsible for the UML Class Diagram and Architectural Diagram. Individually, Ethan is responsible for the class description, Alan is responsible for the attributes description, and Jason is responsible for the operations description.