# Software Design Description

Amazon Go

*Authors*

*Student 1:*
Alperen ÇAYKUŞ
2237170

*Student 2:*
Aytaç Anıl Durmaz
2237295

# Change History

| Version | Date |
|---|---|

# Table of Contents

# List of Figures

# List of Tables

# 1 Introduction

## 1.1 Purpose of the System

The purpose of this system, namely 'Amazon Go', is to enable customers to shop without making them wait in line with a few requirements only, which are a smart phone and an Amazon account with a registered credit card. To achieve this purpose, Amazon has opened lots of stores, which are highly equipped with cutting edge sensors and cameras. These cameras and sensors track the customer and products, and after the customer is done with shopping, she/he just walks out of store. The amount of his/her shopping will be automatically deduced from his/her specified payment method.

## 1.2 Scope

- System will have a mobile application, which will enable users (i.e, store customers) to interact with the system. Customers will log in to store by using the QR code provided to them by their mobile application. Mobile application is also the place where customers sign up to the system, see their current cart status, and past shoppings.
- System will use remote servers to keep data of the customers, such as current shopping session, payment information, etc. Remote server will also communicate with the mobile application to enable a customer to see his/her current shopping session, past shoppings, etc.
- System will have physical stores, which are equipped with very sensitive sensors and cameras. Using Artificial Intelligence and Computer Vision algorithms, these cameras and sensors will gather information from the store and communicate with the remote server.
- System will use several APIs to communicate with the specified payment method of the customers, such as bank accounts. By doing so, system will be able to withdraw money from the customer's account when she/he is done with shopping.
- System will use a database to store customer related information (such as user ID), temporary customer shopping cart, store workers' information, products' information and a database table for the system admins. Those mentioned tables such as customer related information actually consist of multiple tables. Only the system admins are able to make changes and read the database.
- System will keep log of all the shoppings of the users on the remote server for legal purposes. Only system admins can see these logs.
- System will also have a store worker interface to enable the store worker to troubleshoot in case something goes wrong in the store, such as sensor malfunctioning, incorrect amount of money withdrawn from the customer, wrong product was added to a customer's cart, etc.

## 1.3 Stakeholders and their concerns

- **Amazon:** Being the owner of the whole system and the store, Amazon's main concern is reliability of the system so that Amazon will not lose money due to sensor failures or due to customers tricking the sensors and being charged less than they are supposed to be charged. Privacy of the customers is also important as this may have legal consequences.
- **Users:** Users are actually the customers that visit Amazon GO stores. They have four main concerns, which are, firstly, not waiting in line to pay for what they had bought;

secondly, precision of the sensors so that they do not pay extra money due to sensors' fault; thirdly, security of their personal information such as their credit card information, place of residence etc., which are stored in the Amazon servers; and lastly, having a mobile application which is easy to use.

- **System Developers:** Developers are very critical for this system, because it completely relies on the software. Their main concern is maintainability, due to complexity of the system. For this purpose, the documentation and the system itself should be organized very well. Also, the system must be set up in a way that if one component fails, restoration of this component and integration of the new component, which replaces the faulty one, with the system must be easy.

- **IT Staff:** IT Staff are basically the system administrators, who are responsible to maintain databases and communicate with the store workers when needed. They are essential in the continuity of the system. Their major concern is that sustainability of the database, which is created by the system developers when the system is being developed.

- **Store Workers:** Store workers are the people that are responsible for the product placement and shelf maintenance. They are mostly concerned about the illegal actions. Therefore, they should be trained to what to do for the most of the possible scenarios. Their main concerns are being able to easily reach a system admin in case of a failure, and being able to contact with the security in case of an illegal action by the customers.

# 2 References

**This document is written with respect to the specifications of the document below:**

1016-2009 - IEEE Standard for Information Technology–Systems Design–Software Design Descriptions

**Other sources:**

Cheng, A. (2019, January 13). Why Amazon Go May Soon Change The Way We Shop. Retrieved from *https://www.forbes.com/sites/andriacheng/2019/01/13/why-amazon-go-may-soon-change-the-way-we-want-to-shop*

Wingfield, N. (2018, January 21). Inside Amazon Go, a Store of the Future. Retrieved from *https://www.nytimes.com/2018/01/21/technology/inside-amazon-go-a-store-of-the-future.html*

# 3 Glossary

| Term | Definition |
|---|---|
| User | A customer that is signed up to Amazon Go application and has a unique user ID. |
| Database | A MySQL database. |
| User ID | A unique number which can be used to identify a user. |
| Store Worker | A real person working in an Amazon Go store. |
| API | Application Programming Interface |
| Mobile Application | An application that must be installed to an Android or iOS device in order to access Amazon GO features. |
| QR Code | An image which contains information about the user. It is generated by the mobile application. |
| System admins | The people who are responsible for the maintenance and the order of the Amazon GO system. |
| Remote server | A real physical computer away from the stores, at the Amazon HQ, which can be accessed via SSH by system admins. |
| SSH | Secure Shell |
| Sensor | A physical electronic device that collects information about the environment it is currently in. |
| Product | The goods that are sold at the Amazon GO stores. |

Table 1: Glossary

# 4 Architectural Views

## 4.1 Context View

In this viewpoint, the systems that Amazon GO interacts with are shown on the Context Diagram below. The following Use Case Diagram shows the actors and their possible interactions with different scenarios. The detailed information about these use case functions can be found on the tables after the Use Case Diagram. These tables give detailed information for each use case function including alternative scenarios. During the implementation, these tables shall be considered and implementation must follow the mentioned scenarios.

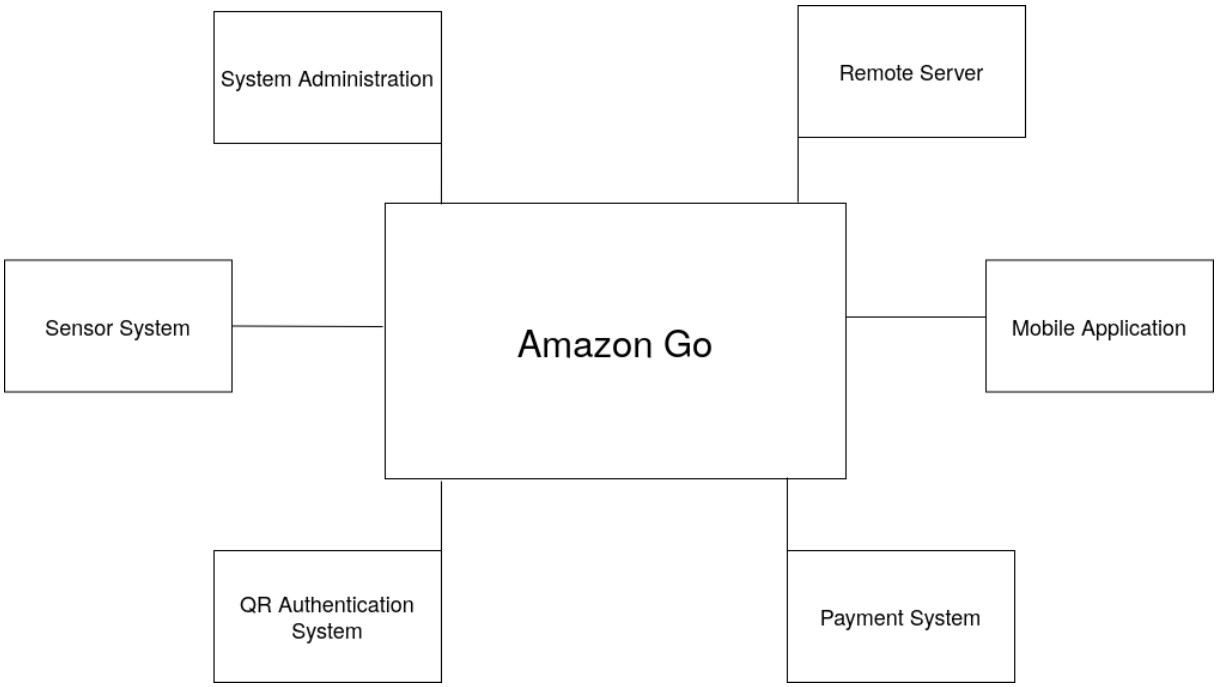


Figure 1: Context Diagram

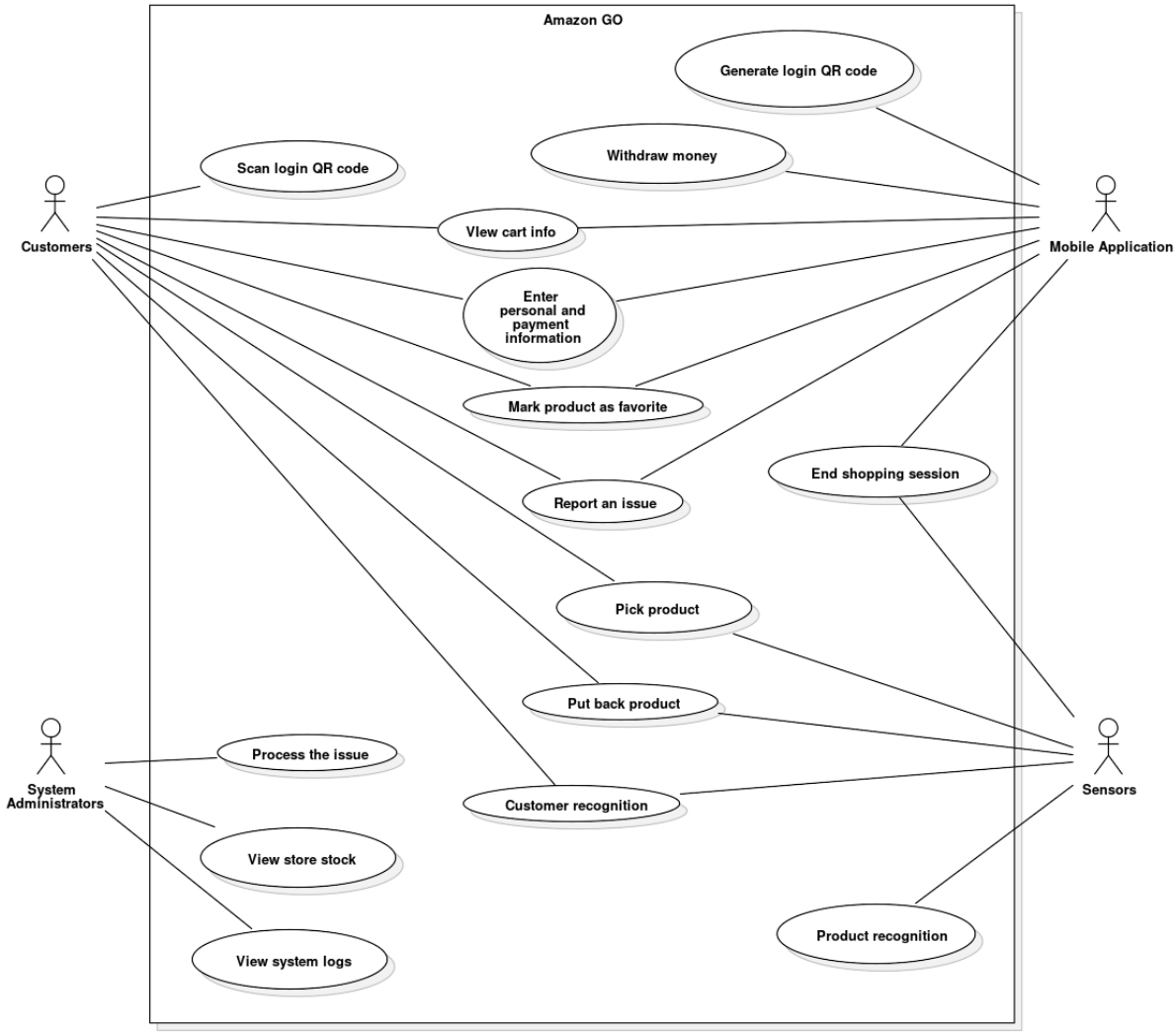


Figure 2: Use Case Diagram

| Use case name | Customer recognition |
|---|---|
| Actors | Customers, Sensors |
| Description | Immediately after the customer gets his/her QR code scanned, environmental cameras must track his/her movements and positions continuously. Cameras must regularly recognize the customers with their face, clothes etc. and must communicate with the remote server until the customer leaves the store. In order to not to lose visual contact with the customer, every angle in the store must be watched by the cameras and the sensors. |
| Data | Customers' physical details such as clothes, face, and body movements. |
| Stimulus | The customer getting his/her QR code scanned. |
| Response | Match the user ID with the acquired visual data of the customer. |
| Basic Flow | 1- Upon QR scanning, visual sensors receive a user ID and identify the unknown customer with the ID.<br>2- Visual sensors track customer's movements and position continuously.<br>3- Customer picks up a product.<br>4- Customer information is sent to the server.<br>5- Flow goes back to step 2. |
| Alternative Flow #1 | 3- Customer puts a product back to a shelf.<br>4- Customer information is sent to the server.<br>5- Flow goes back to step 2. |
| Alternative Flow #2 | 3- Customer leaves the store.<br>4- Customer information is sent to the server.<br>5- System stops tracking the customer. |
| Comments | Immediately after a customer logs in to store, sensors must assign a unique customer ID to that user. |

Table 2: Customer Recognition

| Use case name | Product recognition |
|---|---|
| Actors | Sensors |
| Description | Right after a product is placed to a shelf by the store staff and the product is marked as 'exist' on the store database, the corresponding ID which belongs to the same product set must be assigned to the product. After that point, until someone buys the product and leaves the store, product must be tracked by using its weight information, visual information, and shelf location. A customer is allowed to take a product, put it into his/her bag, and put the product back to any shelf. In that case, for some time, the visual contact with the product would be lost, but after the product is put back to the shelf, system must recover and continue tracking this product. A customer can put a product back to a different shelf, in that case, the product must be recognized by its visual data and weight again. |
| Data | Weight, physical characteristics, and the location of products. |
| Stimulus | A product is placed to a shelf by the store worker and marked as exist in the store. |
| Response | Match the product with the products in the database. |
| Basic Flow | 1- A product is placed to a shelf by the staff.<br>2- Query the database with the shelf location, visual data and weight.<br>3- Assign the received product ID to corresponding product set.<br>4- Sensors work continuously to detect product movements.<br>5- A product is taken from a shelf.<br>6- Product information is sent to the server.<br>7- Flow goes back to step 4. |
| Alternative Flow | 5- A product is placed to a shelf by a customer.<br>6- Product information is sent to the server.<br>7- Flow goes back to step 4. |
| Comments | Initially, products must be registered to the database and must be assigned to a shelf. |

Table 3: Product Recognition

| Use case name | Scan login QR code |
|---|---|
| Actors | Customers |
| Description | A procedure to log users in to store by scanning their QR code via turnstiles. Turnstiles must allow access after a successful scan. Turnstiles must not allow the customers to enter the store if the scan fails and must display the reason. For security reasons, if the same QR is scanned while there is a shopping session with that QR code, login must not be allowed and QR code must be disabled for security purposes. Additionally, customer must be informed about every login with an SMS. |
| Data | User information. |
| Stimulus | Users showing their QR code to turnstiles. |
| Response | Validation or invalidation of the QR code. |
| Basic Flow | 1- Customer shows the QR code to turnstile QR scanner.<br>2- Turnstile scans the QR and sends the data to the server.<br>3- Server processes the store information and the customer information.<br>4- Server responds with success.<br>5- Turnstile gives access to the customer.<br>6- Server sends SMS notification to the customer. |
| Alternative Flow | 4- Server responses with failure.<br>5- Failure reason is displayed.<br>6- Server sends SMS notification to the customer. |
| Comments | Customer must be a registered member. GPS can be used to improve the security. |

Table 4: Scan Login QR Code

| Use case name | Pick Product |
|---|---|
| Actors | Customers, Sensors |
| Description | A procedure to detect customers when they pick up a product from the shelves. When a customer picks a product, weight sensors must detect the change and must send the related data, also visual sensors must recognize the customer who picks up the product and provide additional data about the product. If the customer picks up multiple products at the same time, sensors must be precise enough to handle the situation. |
| Data | Customer information, product's price and other information related to that product. |
| Stimulus | Customers picking products. |
| Response | Add the picked product to customer's cart, update the cart information. |
| Basic Flow | 1- Customer picks a product up from a shelf.<br>2- Weight sensors detect the change.<br>3- Visual sensors detect the customer action and assists product recognition.<br>4- Sensors send data to server.<br>5- Server processes data, adds the related product to customer's cart. |
| Alternative Flow | - |
| Comments | Customers must get their QR code scanned successfully before shopping. |

Table 5: Pick Product

| Use case name | Put back product |
|---|---|
| Actors | Customers, Sensors |
| Description | A procedure to detect customers when they put a product back to shelves. When a customer puts a product back to a shelf, weight sensors must detect the change and must send shelf information, weight information and other related data, also visual sensors must recognize the customer who puts the product back and must provide additional data about the product. System must analyze the cart to be precise about the product. If the customer puts back multiple products at the same time, sensors must be precise enough to handle the situation. |
| Data | Customer information, product's price and other information. |
| Stimulus | Customers putting a product back to shelf. |
| Response | Delete the related product from the customer's cart. |
| Basic Flow | 1- Customer puts a product back to a shelf.<br>2- Weight sensors detect the change.<br>3- Visual sensors detect the customer action and assists product recognition.<br>4- Sensors send data to server.<br>5- Server processes data, removes the related product from customer's cart. |
| Alternative Flow | - |
| Comments | The product must have been picked up beforehand and this must be in the same session. |

Table 6: Put Back Product

| Use case name | View cart info |
|---|---|
| Actors | Customers, Mobile Application |
| Description | A customer must be able to see it's current cart status by using the mobile app. Customer shall see the products she/he picked up, with their quantity and price. Total price must be also displayed. |
| Data | The customer's cart status. |
| Stimulus | The customer pressing the 'My Cart' button on the app. |
| Response | Show the customer's current cart status. |
| Basic Flow | 1- Customer opens the app.<br>2- Customer logins if not signed in already.<br>3- Application opens the cart screen automatically.<br>4- Application sends view cart request to the server.<br>5- Server responds with the related data.<br>6- Application shows the cart status. |
| Alternative Flow | - |
| Comments | Data must be fetched from the remote server in case of an attempt to cheat. User must be shopping to be able to see the cart screen. |

Table 7: View Cart Info.

| Use case name | Enter personal and payment information |
|---|---|
| Actors | Customers, Mobile Application |
| Description | When a customer signs up, s/he must provide personal information and payment method. If the payment method is invalid, application must request a valid method and must not allow another action to be taken by the customer until a valid payment method is provided. |
| Data | Personal data and payment data. |
| Stimulus | When a customer signs up or updates his/her information. |
| Response | Validation or invalidation of the payment method. If validated, sign up or update the user. |
| Basic Flow | 1- Application shows a personal information form or a payment form.<br>2- User enters information.<br>3- Application checks the validity of the information at the same time.<br>4- If everything is valid, a button is enabled to finish the process.<br>5- User presses the button.<br>6- Server inserts the related information to the database. |
| Alternative Flow | - |
| Comments | If the customer is a member, then instead of signing him/her up, update the information. |

Table 8: Enter Personal and Payment Information

| Use case name | End shopping session |
|---|---|
| Actors | Mobile Application, Sensors |
| Description | Visual sensors must detect when the customer leaves the store and report to remote server. Details about the shopping session must be visible from the application after session ends. |
| Data | Customer ID, store information. |
| Stimulus | Customer leaving the store. |
| Response | Session information. |
| Basic Flow | 1- Customer walks out from the store.<br>2- Visual sensors detects the action.<br>3- End shopping session request is sent to servers with the customer ID.<br>4- Server processes the request and saves the sessions information to the database.<br>5- Visual sensors stop tracking that customer. |
| Alternative Flow | - |
| Comments | Log the current session to database for possible further use. |

Table 9: End Shopping Session

| Use case name | Generate login QR code |
|---|---|
| Actors | Mobile Application |
| Description | Application must generate a unique QR code to enable customers log in. A generated QR code can be used multiple times until it is disabled due to security or user prompt. QR codes must be user specific. |
| Data | Customer information. |
| Stimulus | User pressing the 'QR Code' button on the mobile app. |
| Response | Produce and show the QR code |
| Basic Flow | 1- A customer signs up.<br>2- Application requests a QR code for the customer.<br>3- Server generates a QR code that is unique to the customer. |
| Alternative Flow | 1- Customer presses the Refresh QR button.<br>2- Application requests a QR code for the customer.<br>3- Server generates a QR code that is unique to the customer. |
| Comments | QR code must be generated and sent by the remote server in case of a cheating attempt. |

Table 10: Generate Login QR Code

| Use case name | Withdraw money |
|---|---|
| Actors | Mobile Application |
| Description | When user leaves the store, withdraw money from his/her registered payment method. System must use the payment method provided by the customer. If the payment attempt fails due to lack of money or other reasons, notifications must be sent regularly until customer pays. Customer must not be allowed to shop until she/he pays. |
| Data | User ID. |
| Stimulus | Upon the 'End shopping session' procedure. |
| Response | Confirmation that payment is successful. |
| Basic Flow | 1- System is informed by the End Shopping Session function.<br>2- System requests a payment from the user's payment method via the Payment API.<br>3- Payment is successful.<br>4- Receipt is created. |
| Alternative Flow | 3- Payment is failed.<br>4- An SMS notification is sent to the customer.<br>5- The customer is blacklisted until the she/he pays. |
| Comments | Further actions can be taken by the authorities if a payment is not received for a long time. |

Table 11: Withdraw Money

| Use case name | View store stock |
|---|---|
| Actors | System Administrator |
| Description | System admins must be able to see current stock status of each store and main warehouse. User must be warned about low stocks. |
| Data | Store ID, admin ID. |
| Stimulus | User pressing the 'View Stock' button on the mobile app. |
| Response | Stock information . |
| Basic Flow | 1- System admin opens the administration page.<br>2- User presses the stock status tab.<br>3- Overall stock status of each store is listed to the user. Stocks are colored according to the amount.<br>4- User selects a store.<br>5- Detailed stock status of the selected store is shown to the user. |
| Alternative Flow | - |
| Comments | User must be authorized to view stock information. |

Table 12: View Store Stock

| Use case name | View system logs |
|---|---|
| Actors | System Administrators |
| Description | Administrators must be able to see previous shopping session details of customers. Any inconsistency in sessions must be reported. |
| Data | Time and date, Store ID, Customer ID |
| Stimulus | Administrators reaching the log files. |
| Response | Previous cart information, payment method |
| Basic Flow | 1- System admin opens the administration page.<br>2- User presses the view logs tab.<br>3- Several filter options are shown to the user.<br>4- User fills some of the filtering fields.<br>5- User presses the search button.<br>6- Results are listed. |
| Alternative Flow | - |
| Comments | Only administrators can see the log files. |

Table 13: View System Logs

| Use case name | Mark Product as Favorite |
|---|---|
| Actors | Customers, Mobile Application |
| Description | Customers should be able to add a product that they buy frequently or a product that they wish to buy to their favorite products list. |
| Data | Product ID, Customer ID |
| Stimulus | A user pressing the 'Star' on the page of the product using the mobile application. |
| Response | A message indicating whether the operation was successful or not. |
| Basic Flow | 1- Customer logs in to the mobile application.<br>2- User navigates to the product's page on the mobile application via the search button or via the categories.<br>3- System loads the product page.<br>4- User presses the dimmed 'Star' icon on the loaded page.<br>5- 'Star' icon is lighted up and the product is added to the user's favorites. |
| Alternative Flow | 4- User presses the already lighted up 'Star' icon<br>5- 'Star' icon is dimmed and the product is removed from the user's favorites. |
| Comments | On each press to 'Star' icon, the icon is basically toggled. |

Table 14: Mark Product as Favorite

| Use case name | Report an Issue |
|---|---|
| Actors | Customers, Mobile Application |
| Description | A user should be able to communicate with the system admins in case of any problems or questions. |
| Data | User ID, User Message |
| Stimulus | User presses the 'Report an Issue' button at the main page of the mobile application. |
| Response | Validation that issue is reported or not. |
| Basic Flow | 1- Customer logs in to the mobile application.<br>2- User navigates to the issue reporting page by pressing the 'Report an Issue' button.<br>3- A textbox appears on the screen which waits for user to enter his message.<br>4- User enters his/her message and presses the send button.<br>5- A notification appears if the message is transmitted successfully or not. |
| Alternative Flow | - |
| Comments | If a user sends two seperate messages consecutively, these messages shall be merged to avoid spam. Also a user should be able to send only one message in ten minutes. |

Table 15: Report an Issue

| Use case name | Process the Issue |
|---|---|
| Actors | System Administrators |
| Description | System admins inspect the issue provided by the user and take action accordingly. |
| Data | Response Message, User ID |
| Stimulus | An admin pressing the 'Inspect Issues' button on the admin page and selecting an issue. |
| Response | The status of the selected issue is updated (such as ongoing, solved etc.) |
| Basic Flow | 1- An admin logs into the admin page using his Admin ID and password.<br>2- The admin presses the 'Inspect Issues' button on the admin page.<br>3- A list of unsolved issues are shown to the admin.<br>4- The admin selects an issue by the date (early messages first).<br>5- The admin takes action (such as inspecting the logs) and sends a reply to the issuer. |
| Alternative Flow | 6- The admin marks the issue as solved. |
| Comments | |

Table 16: Process the Issue

## 4.2 Composition View

In this viewpoint, the components of the system are shown from a top-level point of view. Also, the design rationale for each decision is provided right after the Component Diagram.

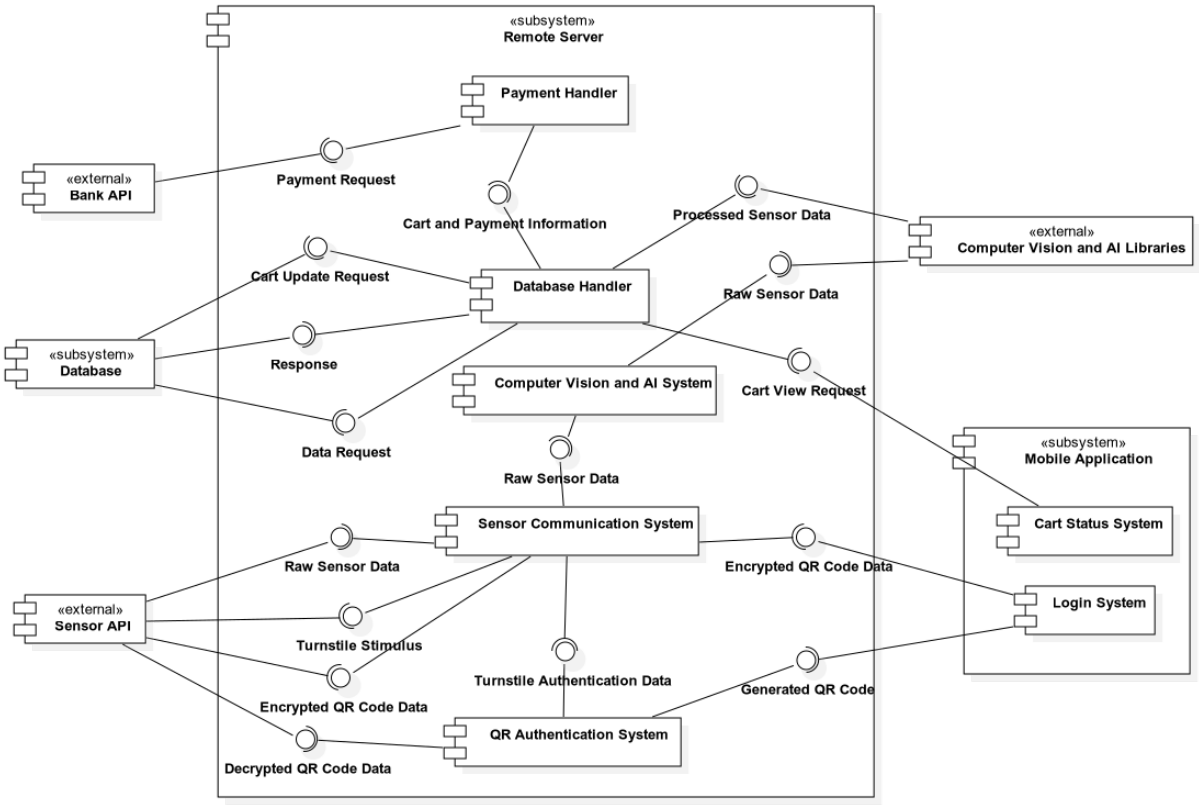


Figure 3: Component Diagram

**Design Rationale:**

- Remote Server is the component where most of the things are done. Any communication between components, interaction with database and any calculation that needs to be done for the AI and Computer Vision algorithms are done on the remote server.
- AI and Computer Vision calculations are done via the libraries, which reside on the server.
- Amazon Go is a system that needs to handle several different tasks at the same time. Some of these tasks follow similar routines. Therefore some generic components and interfaces like Sensor Communication system are used, which allows to keep the diagram simple and more understandable.
- QR Authentication System is basically the entrance point of the system. It produces the QR code, then forwards it to the Login System of the mobile application. Then, the user gets his/her QR code scanned by the turnstiles, which posts the encrypted QR code to Sensor Communucation System. Sensor Communication System forwards this data to the corresponding Sensor API, QR code gets decrypted, and sent back to the QR Authentication System. QR Authentication system either validates or invalidates the QR Code, and depending on the output, it sends information to Sensor Communucation System, which then may send a stimulus for the turnstiles depending on the validation output.
- Sensor Communication System is used as an abstraction to define communication of all kinds of sensors including cameras, QR scanners on turnstiles and weight sensors on shelves. Raw sensor data is the sensor data produced by the corresponding sensor. This might be image data or weight data. QR Code data is shown separately as it has a separate target component.
- Computer Vision and AI System is basically a bridge between the Sensor Communication System and Computer Vision and AI Libraries. Any raw data that is produced by the sensors first passes through Sensor Communication System, which forwards this raw data to Computer Vision and AI System, which then forwards this data to the required libraries to be processed.
- DBMS is the place where the interaction with the database and the manipulation of the database take place.
- Payment Handler is a component which is responsible of the payments. When a user exits the store, required payment information is taken from the database via DBMS, and this data is forwarded to Payment Handler. Depending on the bank, Payment Handler makes calls to the corresponding Bank API.
- Mobile application provides some functionalities for the end users. It must communicate with the remote server in order to provide QR authentication and cart view features.
- Database mostly holds user and product related information. These data might be customers' and products' physical properties, customers' name, products' price and etc. Computer vision and AI mostly depend on that information while operating.
- Bank API is an abstraction that is used to define different Bank APIs. Depending on the user's choice of bank, it uses a different bank's API.

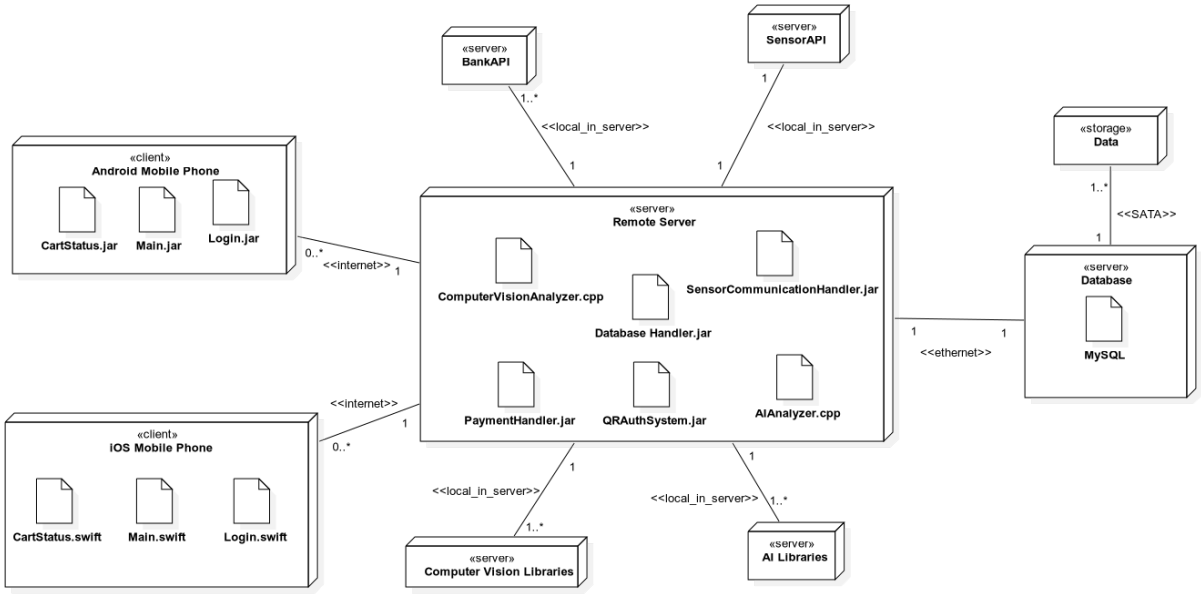


Figure 4: Deployment Diagram

**Design Rationale:**

- The database is separate from the remote server so that in case of a system failure, the data does not get corrupted.
- Storage devices that are used in the database are connected via SATA interface as HDDs will be used as storage devices since SSDs are too expensive to use as storage.
- There are two mobile applications, one for Android and one for iOS. These will be downloadable from the corresponding stores. Main.jar and Main.swift are shown to show that these mobile applications will be developed in Java and Swift, respectively.
- The connections that are represented by $<< local\_in\_server >>$ means that these parts reside on the same computer that acts as a server. In other words, they are basically stored in the storage devices of the server, which can be accessed by a basic disk seek operation.
- MySQL is used for DBMS since it is open source and free.
- For simplicity, only one storage device which is connected to database is shown. However, system will use more than one storage devices, and some of them will be used as backup disks.

## 4.3 Information View

In this view, the organization and the relations of the data in the database and their class counterparts, methods and attributes of these classes are shown. Detailed information about the operations are given after the Interface Class Diagram. Furthermore, classes that are needed by the database operations can be seen in detail on the Database Class Diagram.

### 4.3.1 Interfaces

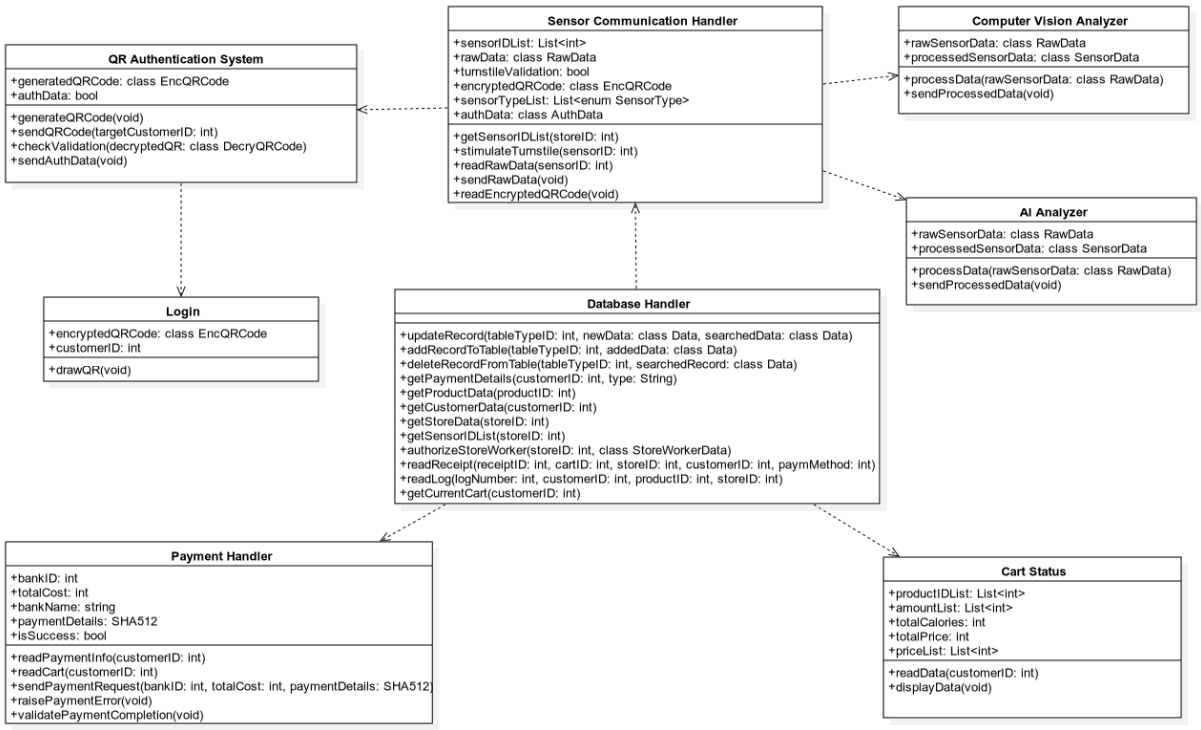


Figure 5: Interface Class Diagram

| **Operation** | **Description** |
|---|---|
| generateQRCode | Generates a QR code for authentication. |
| sendQRCode | Sends the generated QR code to the mobile app. |
| checkValidation | Takes the decrypted QR code as input and checks whether it is valid or not. |
| sendAuthData | Sends the validation that is obtained from the checkValidation to the Sensor Communication System. |
| drawQR | Draws the generated QR Code onto the screen of the user's mobile application. |
| readPaymentInfo | Gets the payment information of the given customer by making calls to related Database Handler getter functions. |
| readCart | Gets the current cart information of the given customer by making calls to related Database Handler getter functions. |
| sendPaymentRequest | Chooses the correct Bank API and sends a request to this API to make the payment. |
| raisePaymentError | If payment is not successful, displays an error message. |
| validatePaymentCompletion | If payment is successful, displays a success message. |
| getSensorIdList | Given storeID, retrieves all sensors in this store and writes it to its sensorIDList variable. |
| stimulateTurnstile | If the validation of QR code is successful, sends a stimulant to turnstiles to open them. |
| readRawData | Reads the raw data (not processed data) from the sensors onto rawData variable. |

| | |
|---|---|
| sendRawData | Forwards rawData variable to AI Analyzer and Computer Vision Analyzer. |
| readEncryptedQRCode | After the QR code is generated and displayed on the user end, this function is called to obtain the generated QR code. |
| processData | Makes required computations for Computer Vision and AI. |
| sendProcessedData | Posts the output of computations that are done by Computer Vision and AI to database handler to be registered. |
| readData | Retrieves the given customer's cart data by communicating with database handler.. |
| displayData | Displays the retrieved cart data on the user's mobile app. |
| updateRecord | Updates the given record on the corresponding table. Class Data is a base class which instantiates other data classes such as class CustomerData, class StoreData etc. |
| addRecordToTable | Inserts the given record to the corresponding table. |
| deleteRecordFromTable | Deletes the given record from the corresponding table by ID. |
| getPaymentDetails | Gets the payment details of the given customer from the database. |
| getProductData | Gets the product information of the given product from the database. |
| getCustomerData | Gets the customer information of the given customer from the database. |
| getStoreData | Gets the store information of the given store from the database. |
| getSensorIdList | Gets all the sensors' ID that are in the given store from the database. |
| authorizeStoreWorker | Adds a store worker to the given storeID. Used when employing a store worker. |
| readReceipt | Gets the receipt data according to given parameters from the database. |
| readLog | Gets the logs that are related to given parameters from the database. |
| getCurrentCart | Gets the given customer's current cart information from the database. |

Table 17: Operation descriptions

**Design Rationale:**

### 4.3.2 Database Operations

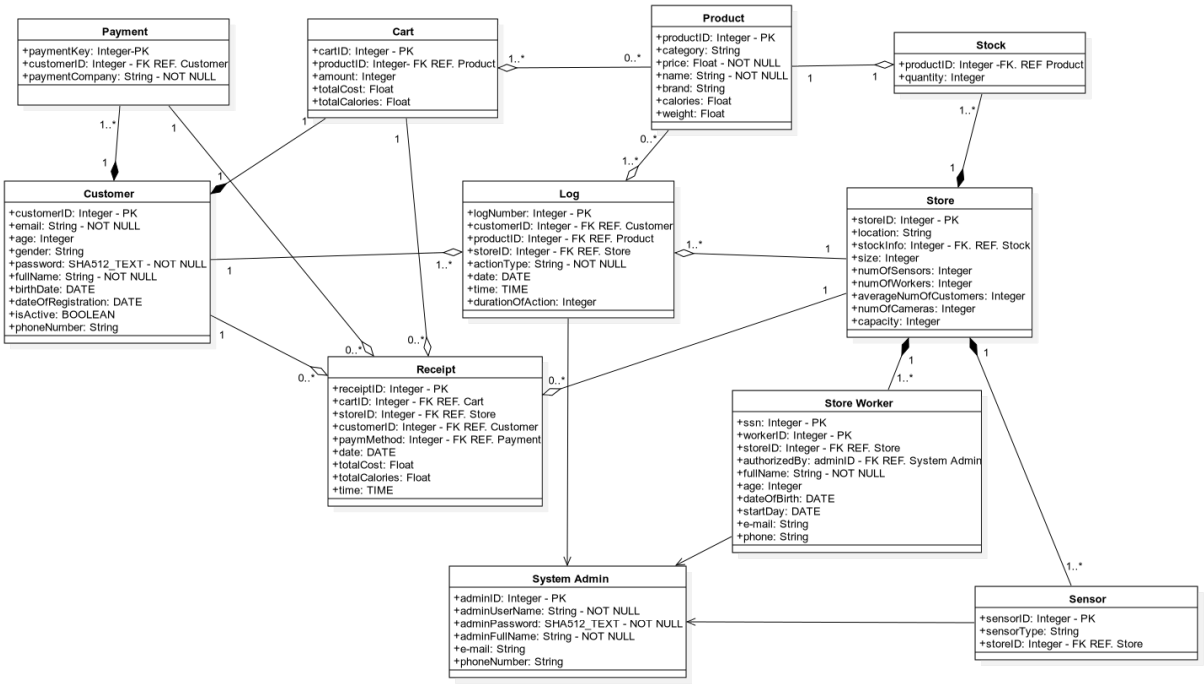


Figure 6: Database Class Diagram

| **Operation** | **CRUD** |
|---|---|
| generateQRCode | Create: Log<br>Read : Customer<br>Update:<br>Delete: |
| readPaymentInfo | Create:<br>Read : Payment<br>Update:<br>Delete: |
| readCart | Create:<br>Read : Customer, Cart<br>Update:<br>Delete: |
| validatePaymentCompletion | Create: Receipt, Log<br>Read :<br>Update:<br>Delete: Cart |
| sendProcessedData | Create: Log<br>Read :<br>Update: Cart<br>Delete: |

| | |
|---|---|
| updateRecord | Create: Log<br>Read :<br>Update: Payment, Customer, Cart, Product, Stock, Store, Store Worker, Sensor, System Admin<br>Delete: |
| addRecordToTable | Create: Payment, Customer, Cart, Product, Stock, Store, Store Worker, Sensor, System Admin, Receipt, Log<br>Read :<br>Update:<br>Delete: |
| deleteRecordFromTable | Create: Log<br>Read :<br>Update:<br>Delete: Payment, Customer, Cart, Product, Stock, Store, Store Worker, Sensor, System Admin |
| getPaymentDetails | Create:<br>Read : Payment<br>Update:<br>Delete: |
| getCustomerData | Create:<br>Read : Customer<br>Update:<br>Delete: |
| getStoreData | Create:<br>Read : Store<br>Update:<br>Delete: |
| getSensorIDList | Create:<br>Read : Sensor<br>Update:<br>Delete: |
| authorizeStoreWorker | Create: Store Worker, Log<br>Read :<br>Update:<br>Delete: |
| readReceipt | Create:<br>Read : Receipt<br>Update:<br>Delete: |
| readLog | Create:<br>Read : Log<br>Update:<br>Delete: |
| getCurrentCart | Create:<br>Read : Cart<br>Update:<br>Delete: |

Table 18: CRUD Operations

**Design Rationale:**

- MySQL will be used as the database management system.
- Important operations will create logs as it can be seen on the Table 18.
- updateRecord, addRecordToTable and deleteRecordFromTable are generic functions that are able to manipulate certain tables. These will also create a log every time they are called.

## 4.4 Interface View

In this view, the internal interfaces between the components of the system and the external interfaces between the Amazon Go and the other systems will be specified in detail.

### 4.4.1 Internal Interfaces

**The Interface between the Database Handler and the Sensor Communication Handler:**
Database Handler uses Sensor Communication Handler in order to send data to sensors and receive data from the sensors. Information that need to be stored are first passes through Sensor Communication Handler, and stop at Database Handler in order to be stored. When the data needs to be processed, Database Handler fetches this data and sends it to Computer Vision and AI analyzer via Sensor Communication Handler.
**Design Rationale:**

- Database Handler is the main part of the program as Amazon Go runs on lots of environmental data, and hence, interaction of the Database Handler and Sensor Communication handler is a must.
- Having the middleman Sensor Communication Handler between the Database and Computer Vision - AI Analyzer enables buffering of the data, meaning that Database Handler can serve to other components when there is no data to be stored or fetched. Hence, Sensor Communication Handler can be used as cache to Computer Vision and AI Systems.

**The Interface between the Database Handler and the Cart Status:**
Cart status is responsible for issuing a current cart read request to the Database Handler and storing it. In other words, it will ask the user's cart information to the Database Handler and Database Handler will fetch it from the database and will send it as a response. After that, it is responsible for displaying the retrieved cart data to the user.
**Design Rationale:**

- Cart Status component must be very responsive as it is an I/O operation and noone wants to wait when he/she clickes to an icon on the mobile device. Hence, there is no middleman between Database Handler and Cart Status
- Current cart status is stored in the database, so Database Handler is responsible from retrieving the current cart data.
- Since Cart Status is an I/O operation, Database Handler may give priority to it.

**The Interface between the Database Handler and Payment Handler:**
Paymend Handler is responsible for handling the payment related operations. It will ask for

the user's payment information and cart information to the Database Handler, then Database Handler will fetch them from the database and will send them as a response. After that, Payment Handler tries to complete the payment by using the retrieved data and depending on if the payment was a success or not, it either raises an error or returns a message about the successful completion of the payment.

**Design Rationale:**

- Since there are more than 1 banks, Payment Handler might be more than one. For simplicity, it is shown as one component but the operations for different banks shall follow the same procedures.
- Payment Handler is directly connected to Database Handler for safety reasons as well. Since critical information flows during this operation, by not introducing an additional middle-man, we decrease the chance of an intrusion.

**The Interface between the Sensor Communication Handler and Computer Vision Analyzer:**
Computer Vision Analyzer will receive visual data from the Sensor Communication Handler to process it. Then, it will send back the processed data. How Sensor Communication Handler works and design rationale for this is given in the first item of this subsection.

**The Interface between the Sensor Communication Handler and AI Analyzer:**
AI Analyzer will receive AI related data from the Sensor Communication Handler to process it. Then, it will send back the processed data. How Sensor Communication Handler works and design rationale for this is given in the first item of this subsection.

**The Interface between the QR Authentication System and Sensor Communication Handler:**
QR Authentication System receives decrypted QR code from Sensor Communication Handler to validate it. Then, sends the result as a response to Sensor Comminucation Handler. Sensor Comminucation Handler interacts with turnstiles for this particular case. It is responsible for validating the QR code and stimulating the turnstiles.

**Design Rationale:**

- QR Authentication System generates encrypted QR codes and validates decrypted QR codes. Decrypted ones comes from the ensor Communication Handler.
- Sensor Communication Handler acts as a buffer and a cache between QR Authentication System and Database Handler.

**The Interface between the QR Authentication System and Login System:**
Login system requires encrypted QR code to display it to user's phone. QR Authentication System sends that QR to login system.

**Design Rationale:**

- Users need QR code to enter the stores. Login System displays it and QR Authentication System handles the rest.
- QR code can be generated by the QR Authentication System only and Login System will request QR codes from it very frequently.

- Since Login is an I/O operation, Database Handler may give priority to it.
- As explained, authentication is QR code is made by System Communication Handler.

### 4.4.2 External Interfaces

#### 4.4.2.1 User Interfaces

**User Sign-up Interface:**
This interface provides a user friendly way for non-registered users to register. Users enter his/her personal and payment information here, then these information are sent to the server and stored in the database. After a valid registration, users will be informed that the registration is successfully completed. By creating an account, users agree to Amazon Go's Conditions of Use and Privacy Notice automatically, which is stated during the registration. There is no time restriction in this part, but closing the application during the registration will require to start the registration from the beginning. Most of the information given in this part can be replaced later, only the username cannot be changed.
**Design Rationale:**

- This interface is a user's first entry point to the whole Amazon Go system.
- Fields denoted with * must be filled to have a valid account.

**User Cart Interface:**
This interface provides a simple and informative way for the customers to see the details of their current session while shopping. Users need to open the app during the shopping and this screen will be directly opened when they open the app. Each item they took will be listed with the most important details: price, quantity, size-weight-volume and etc. Users must login to see this interface, but since they can not enter the shop before the login, this will be not the case most of the time. Yet, if they logout after entering, they can login back and see their cart. If they close the app, they can simply open it back and the cart will be shown. This screen will end after the customer leaves the store. This screen will be designed for mobile phones especially.
**Design Rationale:**

- This interface is designed to to be displayed on the users' smart phone. Therefore, it is simple and user-friendly.
- Depending on screen size, this interface may change.

**User Payment Interface:**
This interface is designed to be fully informative for the customers to see the every detail of their past shopping sessions. Users can see a list of their previous sessions and can select a session to see the every detail including the receipt. Also, there is a button available to change the payment method here, so that customers can reach this method easily. Input form of the payment method is identical to the input form at the registration. User will be informed about the validation of the payment method change after the change attempt. There is no time restriction in this part, receipts can bee seen immediately after a few minutes maximum and will be stored until it is deleted manually.
**Design Rationale:**

- Although a receipt can be deleted by the user, the receipt will be kept in a seperate table in the database for legal purposes. Duration of this may change depending on the local regulations.

**System Admin Interface:**
This interface is designed to be informative about the stores and the user actions. System admin must login to the system with a authorized username and password to use this interface. This interface can be only reached from the Amazon facilities, it is unreachable from external. A list of the stores and warehouses will be seen after entering the system. User will be able to select a store or a warehouse to see the stock details of that place. Moreover, there will be a tab to switch to the logs. Here, user can make a search about any user by their username, name and e-mail. Results can be altered with these action types: Registration time and date, entering to a specific store, Successful and failed payment attempts and blacklisted. User must login again after being idle for 30 minutes due to privacy of customer data.
**Design Rationale:**

- This interface is necessary for the system admins to take actions on the system.
- This screen shall be implemented for a computer screen as system interventions are done here.

**User Login Interface:**
This interface is designed to allow a user to enter an Amazon Go store. When clicked, this interface fetches the QR code that needs to be scanned and displays it on the customer's mobile device. The customer gets his/her QR Code scanned by the turnstiles and then, this interface displays whether the validation of the QR code was successful or not. If not, it prompts an error message alongside with an error code, which can be used when contacting to the system admins to solve login problem.

**Design Rationale:**

- This interface is necessary for customers to enter the Amazon Go stores.
- A Re-Generate button shall be present to overcome QR code related bugs.
- Must be designed for a mobile device and hence, design may change according to the screen size.

#### 4.4.2.2 System Interfaces

**Interface Between Sensor API and Database:**
This interface collects data from the corresponding sensor and sends them to database. This data is raw data (i.e, not processed by the Computer Vision and AI Libraries).

**Design Rationale:**

- A sensor must use its own API. For instance, if the sensor is a weight sensor, it must use the corresponding weight sensor API.
- Since this data comes from the sensors directly, it is raw data.

**Interface Between Sensor API and Computer Vision - AI:**
Sensor API collects the raw data and sends them to database. The data coming from the cameras and sensors need to be processed with the help of Computer Vision and AI libraries. This raw data is fetched from the database since it is stored in the database by the Sensor API.

**Design Rationale:**

- Sensors and cameras are the essential parts which differentiates the Amazon Go from the other stores.
- This interface is important for the collaboration between the cameras,sensors and the Computer Vision - AI system.

**Interface Between Database and Computer Vision - AI:**
Since Sensor API sends raw data to the database, the libraries that makes Computer Vision and AI calculations must fetch them from the database.

**Design Rationale:**

- Database management is done via MySQL.

**Interface Between Sensor API and Bank API:**
When users leave the Amazon Go stores, communication with the Bank to complete the payment is required.

**Design Rationale:**

- Users must have a verified payment method before entering the stores so that illegal activities can be prevented beforehand.

**Interface Between Database and Bank API:**

Since the users' payment information are stored in the database, to select which Bank API will be used and what parameters that this selected Bank API will use must be determined by fetching these informations from the database. This is done by this interface.

**Design Rationale:**

- Database shall store verified banks so that users can be informed about the supported payment methods.
- Receipts shall be sent to the database by the bank, directly with the defined format after successfull payments of the users.