SOFTWARE DESIGN DESCRIPTION DOCUMENT

Amazon Go

Estel Haliloğlu 2247955
Doruk Gerçel 2310027

Version	Date
1.0	24.06.2020

Contents

List of Figures

List of Tables

1 Introduction

1.1 Purpose of the System

Introduced by the Amazon itself, Amazon Go is an automated store model aiming to provide enhanced shopping experience. By automating the checkout procedure, the system eliminates lines in the grocery store, cutting waiting time from customers' visit. Thus, the system proposes a speedy shopping and higher customer satisfaction.

Amazon Go is a complex system, comprised of multiple subsystems. However, it could be simplified to two main components: The on-site systems with the hardware making the "Just Walk Out" technology functional and the mobile Amazon Go app enabling customers to utilize this system.

1.2 Scope

The scope of this project is to enable a speedy shopping experience for the users, by the use of automated systems in the store and a mobile app on the user side.

Potential user group is not specified clear-cut as anyone owning a smartphone and needing to visit a grocery store could use this service.

The system's user interface is a mobile app. Upon installing the app on their phones, the users will utilize the store's services directly from there.

On the app:

- The user creates a personal user profile.
- The user shopping profile is saved.
- The money transition between Amazon Go and the user is done directly via the application.

Besides the app, we have an automated system in the store. This system employs:

- Sensor fusion
- Computer vision
- Deep learning technologies

With these technologies, the Amazon Go system detects user activities, interacts with the user app to make changes on their cart, and completes payment as the user leaves the store. Thanks to these combined systems, the customers no longer have to wait in lines when shopping.

1.3 Stakeholders and Their Concerns

Users: Users are grocery shop customers who, in the case of Amazon Go - METU, are comprised of students, faculty members, and campus workers. Their main concerns are appropriate prices of the products (Prices should not go above a certain level, or else users will be disinclined to utilize the store), their quality considering their price and the quality of similar products in nearby stores, the range available for selection in comparison to other stores on campus, reduced shopping time, and ease of use with the Amazon Go app.

System Developers: They are divided into two groups in the Amazon Go case: Hardware developers and software developers. Their concerns are working in sync with the other team, and meeting functional and non-functional requirements put forward in the SRS and SDD.

Maintenance Crew: This is the group of people who are responsible for the upkeep of the system once the system is up and running, and the store is open to visit. Their concerns are having system components that are least liable to deterioration, prone to minimum malfunctions, and is easy to integrate changes into.

Store employees and manager: On-site grocery store workers. They are concerned with customer satisfaction, as negative feedback reflects on their position in the store.

Shareholders: Amazon stock owners. As they directly benefit from Amazon Go stores' success, they are concerned with the store's revenue and that the right people are on the directing board to manage the store's success.

2 References

This document is written with respect to the below document:

IEEE standard for information technology–systems design–software design descriptions. (2009). New York, NY: Institute of Electrical and Electronics Engineers.

Other sources:

amazon (December 5, 2016). "Introducing Amazon Go and the world's most advanced shopping technology" – via YouTube.

"Amazon's 1st high-tech grocery store opens to the public". CBC News. January 22, 2018. Retrieved March 25, 2020.

"Amazon opens store with no cashiers, lines or registers". Associated Press. January 22, 2018. Retrieved March 25, 2020.

Grewal, Dhruv; Roggeveen, Anne L.; Nordfält, Jens (March 2017). "The Future of Retailing". Journal of Retailing. The Future of Retailing. 93 (1): 1–6. doi:10.1016/j.jretai.2016.12.008

Johnston, Chris (January 22, 2018). "The supermarket with no checkouts". BBC News. Retrieved March 25, 2020.

3 Glossary

Term	**Definition**
DBMS	Database Management System
RAID	Redundant Array of Independent Disks
ID	Unique number to identify a person or an item
SATA	Serial ATA, a computer bus interface that is used for the storage units
App	Application
MySQL	A relational DBMS
Python File	.py file
Java Script File	.js file
Java File	.java file
User	A person who has an Amazon Go Account
Customer	A person who shops in the store
Employee	A person who works in the store
Admin	A person who manages the system, and has special access permissions
QR Code	Quick Response Code
CRUD	Create Read Update Delete

Table 3.1: Glossary

4 Architectural Views

4.1 Context View

In this viewpoint, the actors and systems engaging with the Amazon Go system and a brief glance at their interactions are given as a context diagram. A detailed peek to which actors/systems interact with which actors/systems and the exact manner of their exchanges ("use cases") are given in the use case diagram following it. In the tables below the use case diagram, the workings of each use case is explained in detail.

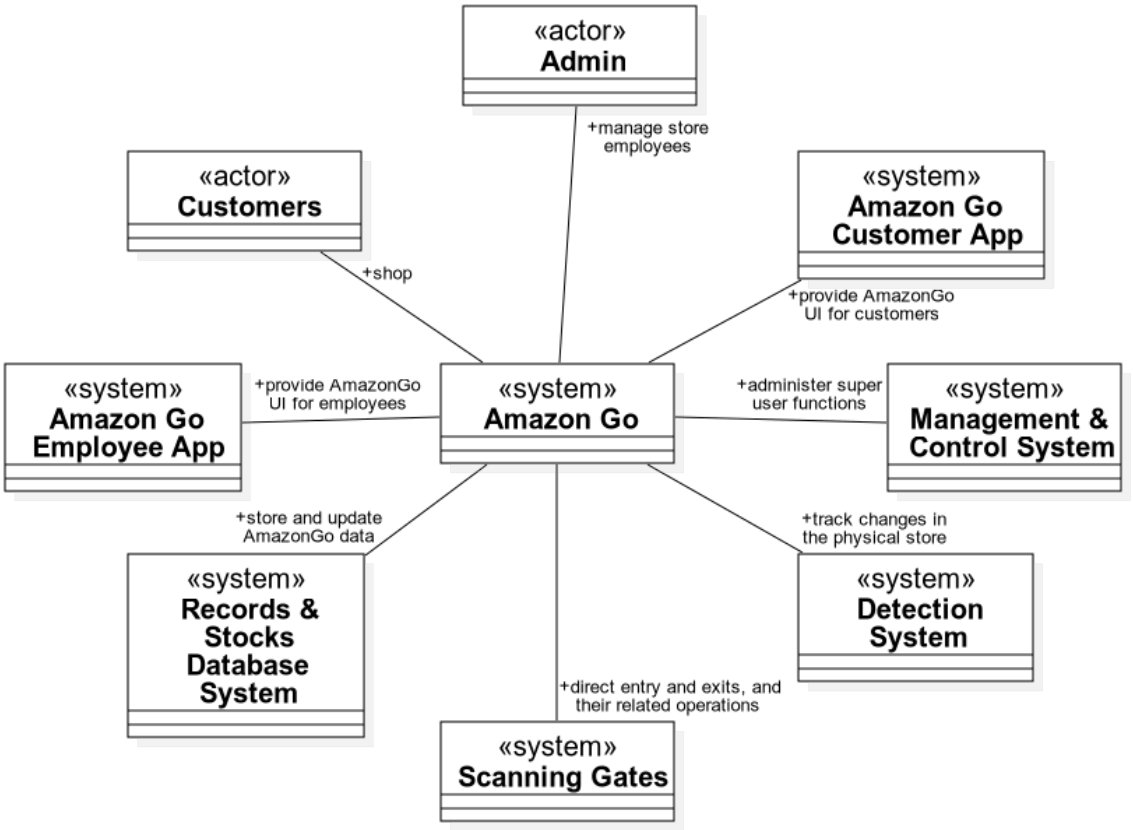

Figure 4.1: Context Diagram

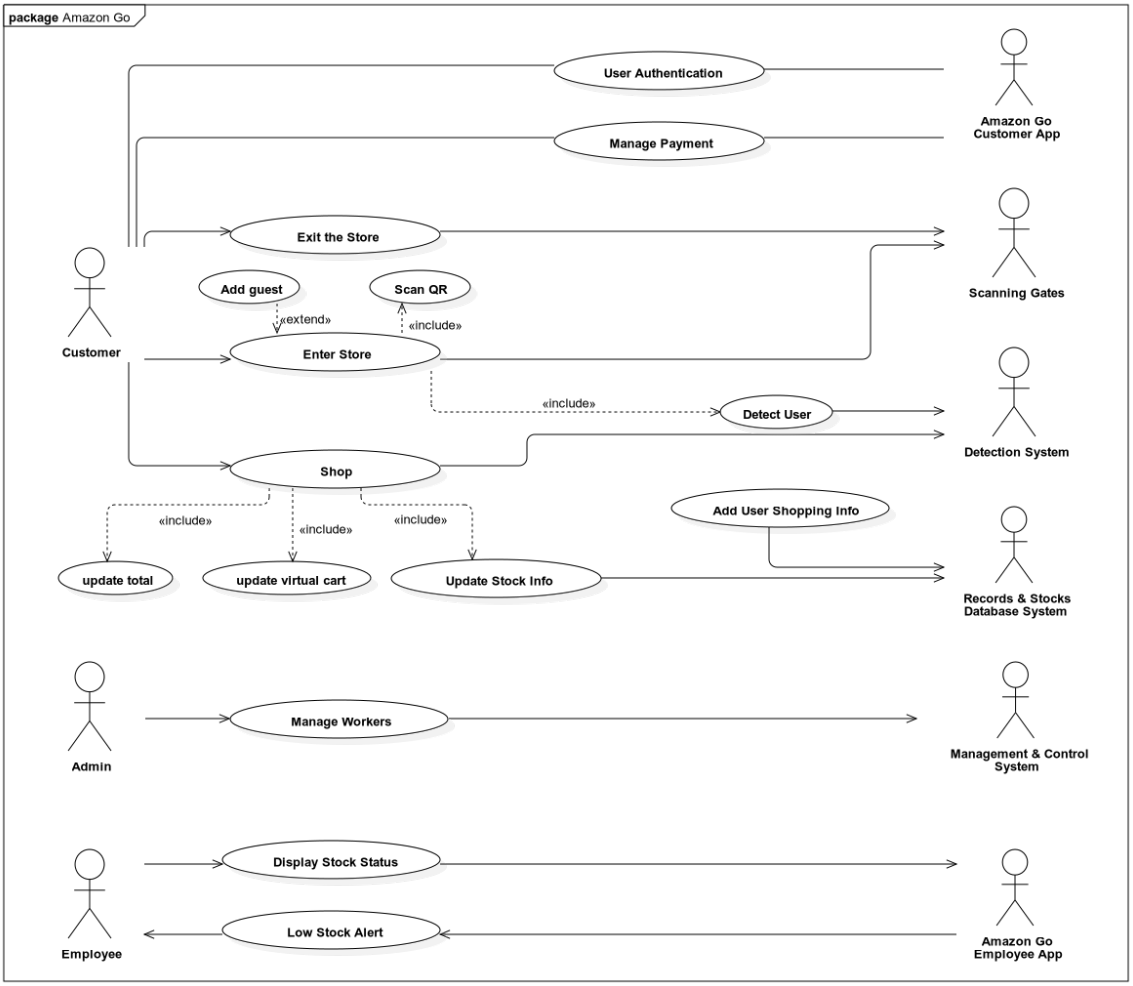

Figure 4.2: Usecase Diagram

Use case name	User Authentication
Actors	Customer, Amazon Go customer app
Description	Customer uses this method to create an account OR to log into an already existing account
Data	Name, e-mail address, password OR E-mail and associated Amazon password
Preconditions	Customer has the Amazon app installed on their mobile device
Stimulus	Clicking "Create your Amazon account" after entering name, e-mail, and password OR Clicking "Sign-In" after entering e-mail and password
Basic flow	I. Customer's name, e-mail, and password are received by the app II. Password is checked against preconditions and approved III. E-mail is authorized to be functional IV. An account is created in the Amazon Go database with name, e-mail and hashed password V. Customer is logged into the newly created account and redirected to the user start page on the app OR I. Customer's e-mail and password are received by the app II. E-mail and hashed password match is checked against user's information in the Amazon Go database III. User information is approved and customer is logged into the system
Alternative flow	User can substitute the e-mail address with their phone number.
Exception flow	If password fails to fulfill any of the following conditions: I. Password must be longer than 6 characters II. Password must contain at least 1 special character III. Password must contain at least 1 number IV. Password must contain at least 1 capital letter, display the Error Message "Please enter a password matching necessary conditions." If e-mail is unauthorized, display the Error Message "Please enter a valid e-mail." OR If the e-mail and password entered do not match, display the error message "E-mail or password incorrect."
Postconditions	Customer has successfully started an account. OR Customer has successfully logged in.

Table 4.1: User Authentication

Use case name	Manage Payment
Actors	Customer, Amazon Go customer app
Description	This function's main purpose is to choose a default card to be used for checkout. It also supports methods to: remove an existing card in payment methods, edit an existing card's information, add additional payment methods
Data	Card to be set to default OR Card to be removed OR Card to be edited and the new name, expiry date, or address to update it with OR Name, card number, and expiry date on a new card to be added
Preconditions	Customer has an existing Amazon account
Stimulus	Clicking "Set as default card" after picking a card from Payment cards on the Amazon app OR Clicking "Remove" under the desired card for removal on the Manage payment cards page OR Clicking "Edit" under the desired card for update on the Manage payment cards page OR Clicking "Add a payment method" on the Manage payment cards page
Basic flow	The preferred card for payment is chosen and set as default on More >Settings >Payment cards after clicking the "Set as default card button" OR The card chosen for removal is removed from the customer's payment methods on More >Settings >Payment cards >Manage payment cards after clicking the "Remove" button underneath OR On More >Settings >Payment cards >Manage payment cards > Edit payment method, an existing card is updated with new name, expiration date or billing address once desired changes are supplied and "Save" button is clicked OR On More >Settings >Payment cards >Manage payment cards > Add a payment method, a new card is added to the customer's payment methods once correct and complete information is entered and "Add you card" button is clicked
Alternative flow	-

Exception flow	[No exception flow for default card setting] OR [No exception flow for card removal] OR If the name entered does not match the card number present, display Error Message "Name entered is incorrect." If the expiry date entered does not match the card number present, display Error Message "Expiration date entered is incorrect." [No exception flow present for billing address change] OR If any of the info fields have inconsistent data, display Error Message "Please check all information entered is correct."
Postconditions	Default card has been chosen OR A card has been removed from the app and disassociated with the user in the Amazon Go database OR An existing card has been updated with new info and the changes were reflected to the database OR A new card has been added and associated with the user in the database

Table 4.2: Manage Payment

Use case name	Enter store
Actors	Customer, scanning gates
Description	The customer enters the store through the scanning gates. This function includes two other use cases, Scan QR and Detect User. It also has an "extend" use case to it, Add Guest
Data	QR code, bodily habitus and facial data
Preconditions	Customer has the app on their phone and their account has previously been set up
Stimulus	Scanning your phone at the scanning gates
Basic flow	Customer goes up to the gates, scans the QR code on their app and the gates open for them allowing entry
Alternative flow	-
Exception flow	-
Postconditions	Customer has entered the store

Table 4.3: Enter Store

Use case name	Add guest
Actors	Customer, scanning gates
Description	The customer can bring in guests alongside by associating them with their Amazon account at the gates. This function is an extension of the Enter Store use case
Data	QR code
Preconditions	Customer has the app on their phone and their account has previously been set up
Stimulus	Additional scans of the QR code after clicking "Add guest" on the app
Basic flow	Customer chooses the "Add guest" functionality on the app and scans their QR code an additional time for each guest they are bringing in. Items picked by the guests are added to the Amazon customer's virtual cart
Alternative flow	-
Exception flow	-
Postconditions	Customer's QR code is recognized by the system and entry of guests is allowed

Table 4.4: Add Guest

Use case name	Scan QR
Actors	Customer, scanning gates
Description	The customer shows their QR code from their app to the scanners on the gates
Data	QR code
Preconditions	Customer has the app on their phone and their account has previously been set up
Stimulus	Showing your phone to the scanning gates
Basic flow	Customer lets the scanners on the gates read the QR code from their app and the gates open for them allowing entry
Alternative flow	-
Exception flow	-
Postconditions	Customer's QR code is recognized by the system and entry is allowed

Table 4.5: Scan QR

Use case name	Detect user
Actors	Customer, detection system
Description	Customer's ID from the scanners is matched with physical characteristics of them from the cameras in the database. This function is included in the Enter Store use case
Data	Bodily habitus and facial data, customer ID from their QR code
Preconditions	Customer has just scanned their QR code to enter the store
Stimulus	Customer ID from the QR scan being received
Basic flow	I. Customer ID from the QR code scan at the gates is received II. Bodily habitus and facial data is recognized III. Customer ID from (I) is combined with the data from (II) and a complete profile is constructed for the customer IV. Customer is tracked during their shopping with this profile
Alternative flow	-
Exception flow	-
Postconditions	Customer is now being tracked by the store's system

Table 4.6: Detect User

Use case name	Shop
Actors	Customer, detection system
Description	Customer's shopping and total is tracked using their profile in the database. This function includes several use cases, Update Total, Update Virtual Cart, and Update Stock Info
Data	Bodily habitus and facial data, customer ID from their QR code, item data
Preconditions	Customer has entered the store
Stimulus	Successfully entering the store
Basic flow	I. Customer ID from the QR code scan at the gates is received II. Bodily habitus and facial data is recognized III. Customer ID from (I) is combined with the data from (II) and a complete profile is constructed for the customer IV. Customer is tracked during their shopping with this profile
Alternative flow	-
Exception flow	-
Postconditions	Customer has completed their shopping

Table 4.7: Shop

Use case name	Update total
Actors	Customer, detection system
Description	Customer's cart total is continuously updated in the database throughout their shopping. This function is included in the Shop use case
Data	Bodily habitus and facial data, price of the latest item that was added to/removed from the virtual cart (VC)
Preconditions	Customer has successfully entered the store and started picking items
Stimulus	The cameras recognizing the customer and the item they are picking up/putting down
Basic flow	I. Adding items to VC: Customer and item is recognized. Price of the item is added to the customer's total in the database II. Removing items from VC: Customer and item is recognized. Price of the item is removed from the customer's total in the database
Alternative flow	-
Exception flow	-
Postconditions	Customer's cart total in the database now reflects the latest changes

Table 4.8: Update Total

Use case name	Update virtual cart
Actors	Customer, detection system
Description	Customer's virtual cart (VC) is continuously updated in the database throughout their shopping. This function is included in the Shop use case
Data	Bodily habitus and facial data, item specifics such as dot codes recognized by the cameras and weight recognized by the load cells on the shelves
Preconditions	Customer has successfully entered the store and started picking items
Stimulus	The cameras recognizing the customer and the item they are picking up/putting down
Basic flow	I. Adding items to VC: Customer and item is recognized. Item ID is added to the VC in the database II. Removing items from VC: Customer and item is recognized. Item ID is removed from the VC in the database
Alternative flow	-
Exception flow	-
Postconditions	Customer's virtual cart in the database now reflects the latest changes

Table 4.9: Update Virtual Cart

Use case name	Update stock info
Actors	Records and Stocks Database System
Description	The stocks of the store changes constantly. (When a customer takes something from the shelf or puts something back on the shelf, the stock changes.) Therefore the store's stock database must be updated continuously in accordance with the changes on the shelf. This function is included in the Shop use case
Data	The dot code of the item, the type of the item, the boolean indicating whether the items are put on the shelf or taken from the shelf, the number of items
Preconditions	-
Stimulus	Change on the shelf
Basic flow	I. The pressure change on the shelf notifies the detection system II. The detection system catches the item, type of the item and the number of this item. III. Amazon Go sends that data to the Records and Stocks Database System.
Alternative flow	-
Exception flow	If an undefined type of data is detected, it is written to the Error log file.
Postconditions	Records and Stocks Database System makes the changes according to the data received.

Table 4.10: Update Stock Info

Use case name	Exit store
Actors	Customer, scanning gates
Description	The customer leaves the store passing through the scanning gates once more
Data	Bodily habitus and facial data that was recorded by the Detect User function
Preconditions	Customer (and guests) has successfully entered the store and completed their shopping
Stimulus	The cameras recognizing the customer at the scanning gates using the aforementioned data
Basic flow	I. Customer's location at the scanning gates and their direction as exit is detected by the cameras II. Their cart total is deducted from their account upon exit III. A bill is generated IV. The e-receipt is sent to their app to be shown in the receipts history
Alternative flow	-
Exception flow	-
Postconditions	Customer (and guests) has left the store. Virtual cart total has been deducted from their account. Customer can now see their most recent receipt on their app

Table 4.11: Exit Store

Use case name	Add user shopping info
Actors	Records and Stocks Database System
Description	To create a user preferences profile (which customer buys which items), customers' shopping data is collected in the Records and Stocks Database System.
Data	The user ID, the dot codes of the items
Preconditions	-
Stimulus	Customer buys the items by exiting the store
Basic flow	I. Amazon Go confirms payment from the user. II. Amazon Go sends both the user ID and the dot codes of the items to the Records and Stocks Database System.
Alternative flow	-
Exception flow	If an undefined type of data is detected, it is written to the Error log file.
Postconditions	Records and Stocks Database System adds the items that are bought to the user profile.

Table 4.12: Add User Shopping Info

Use case name	Manage workers
Actors	Admin, management & control system
Description	With the use of this usecase admins can arrange shifts among store workers and also check who has which shifts.
Data	Shift times, Employee Id
Preconditions	-
Stimulus	Clicking the "Assign Shift" button on the "Manage Workers" page
Basic flow	I. Admin goes to the "Manage Workers" page from the launch page of the employee app. II. Enters employee Id and shift times they desire. III. Clicks "Assign Shift"
Alternative flow	-
Exception flow	I. "Employee does not have free hours" error is displayed if employee has taken on the maximum work hours. II. "Shift is already covered." error is displayed if the shift does not require any more employees.
Postconditions	Employee has been successfully assigned to the desired shift.

Table 4.13: Manage Workers

Use case name	Display stock status
Actors	Employee, Amazon Go employee app
Description	Employee finds out the remaining stock of a certain product by entering product id code or scanning its dot code
Data	Product id number or product dot code
Preconditions	-
Stimulus	Employee clicks the "Check Stock" button after supplying product id or dot code
Basic flow	Employee enter the product id number or scan the dot code of the product on the Stock Info>Check Stocks page and clicks "Display Stock Status" button to see how much of an item is still available in the store
Alternative flow	-
Exception flow	If product is looked up using its id number and id number does not match anything in the database, display error message "Product not offered" and redirect to the Stocks page to supply another id number or dot code
Postconditions	-

Table 4.14: Display Stock Status

Use case name	Low stock alert
Actors	Employee, Amazon Go employee app
Description	When the stock count for a given product in the database drops below a preset STOCK_THRESHOLD, Amazon Go employee app presents a notification to the employee user responsible for that product, notifying them that the stock level is low.
Data	Stock count of a particular item, STOCK_THRESHOLD of said item, related employee id
Preconditions	-
Stimulus	Stock count falls below STOCK_THRESHOLD
Basic flow	I. Stock count for a given item drops below its STOCK_THRESHOLD II. A notification is displayed to the related employee on Amazon Go employee app III. An option to place an order for the item is presented to the employee on the app
Alternative flow	-
Exception flow	-
Postconditions	"Attention: Low stock levels" alert shown with the associated product on the Stock Info page. An option to place an order is presented to the employee.

Table 4.15: Low Stock Alert

4.2 Composition View

In this viewpoint, components of the system and their relations are shown with a top-level point of view. More detailed information will be provided in the following subsections. Design entities and their interactions are given in Composition and Deployment Diagrams.

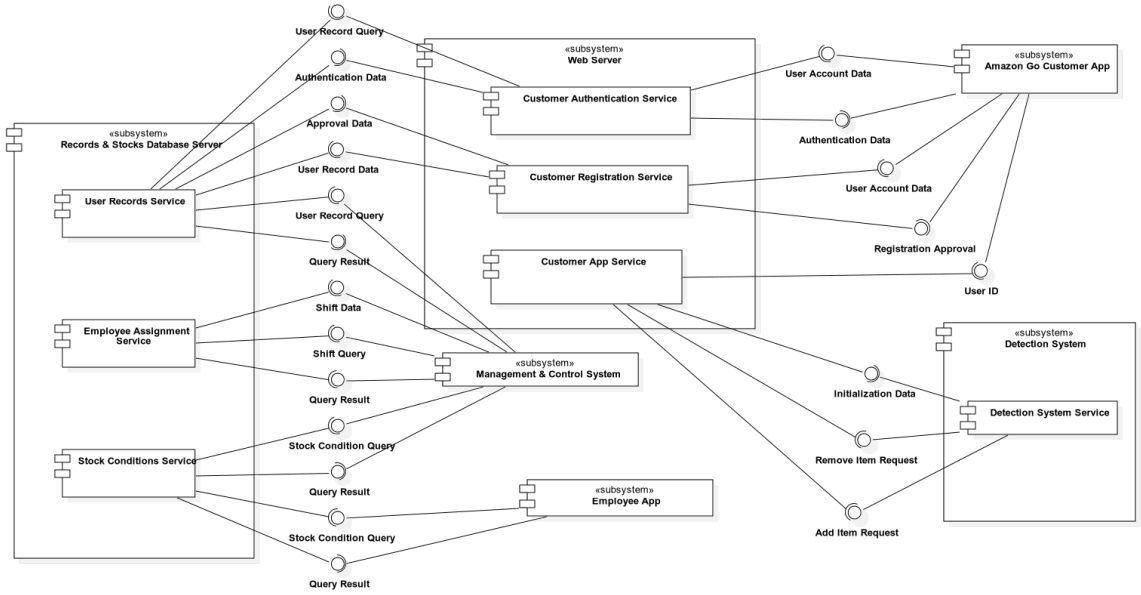

Figure 4.3: Component Diagram

Design Rationale:

- Web Server subsystem contains different components in-order to communicate with the other subsystems. The components that it includes are Customer Authentication Service, Customer Registration Service and Customer App Service.
- Customer Authentication Service component is required for user authentication in login operations. It receives the required user account data from Amazon Go Customer App for user authentication, then it sends this data to User Records Service component as a query to check if any user record corresponding to these information exist. Then it receives an authentication data from User Records Service according to the query result, and sends this authentication data back to the Amazon Go Customer App.
- Customer Registration Service component is required for user registration to the system. It receives the required user account data from Amazon Go Customer App for user registration, then it sends this data to User Records Service component to add this user record to the database system. Then it receives an approval data

from User Records Service according to the success of the operation, and sends a registration approval data back to the Amazon Go Customer App.

- Customer App Service component is required for the communication between the Amazon Go Customer App and the Detection System Service. Therefore this component is active throughout the user's shopping process in the store. This component receives user ID from the Amazon Go Customer App (it is actually only used in initializing the Customer App Service when the user enters the store). Then is sends an initialization data to Detection System Service, to register the user as an active user in the store (required operation for the Detection System, it holds a list of active users in the store). According to the actions of the customer in the store, Detection System Service sends either "add item request" or "remove item request" to the Customer App Service.

- Management & Control System is a subsystem that is required for admin operations. It sends user record query to the User Records Service and receives the query result from it. Also it sends stock condition query to the Stock Conditions Service and receives the corresponding query result.

- Employee App is a subsystem that is used by the employees in-order to check the stock conditions. Therefore it sends stock condition query to the Stock Conditions Service and receives the corresponding query result.

- Records & Stocks Database Server subsystem contains two separate components. One of them is the User Records Service and the other one is the Stock Conditions Service. In physical layer, there are actually two different database systems (one of them only stores the user records and the other one stores the stock condition info) in-order to manage the parallel access to two different types of information (but the combined name of these database systems is Records & Stocks Database Server for simplicity).

- User Records Service component is used to manage database operations about the user records.

- Stock Conditions Service component is used to manage database operations about the stock condition information.

- Detection System is a subsystem which contains the Detection System Service component. Detection System Service is actually kind of an interface that manages the interactions of the Detection System with the other systems.

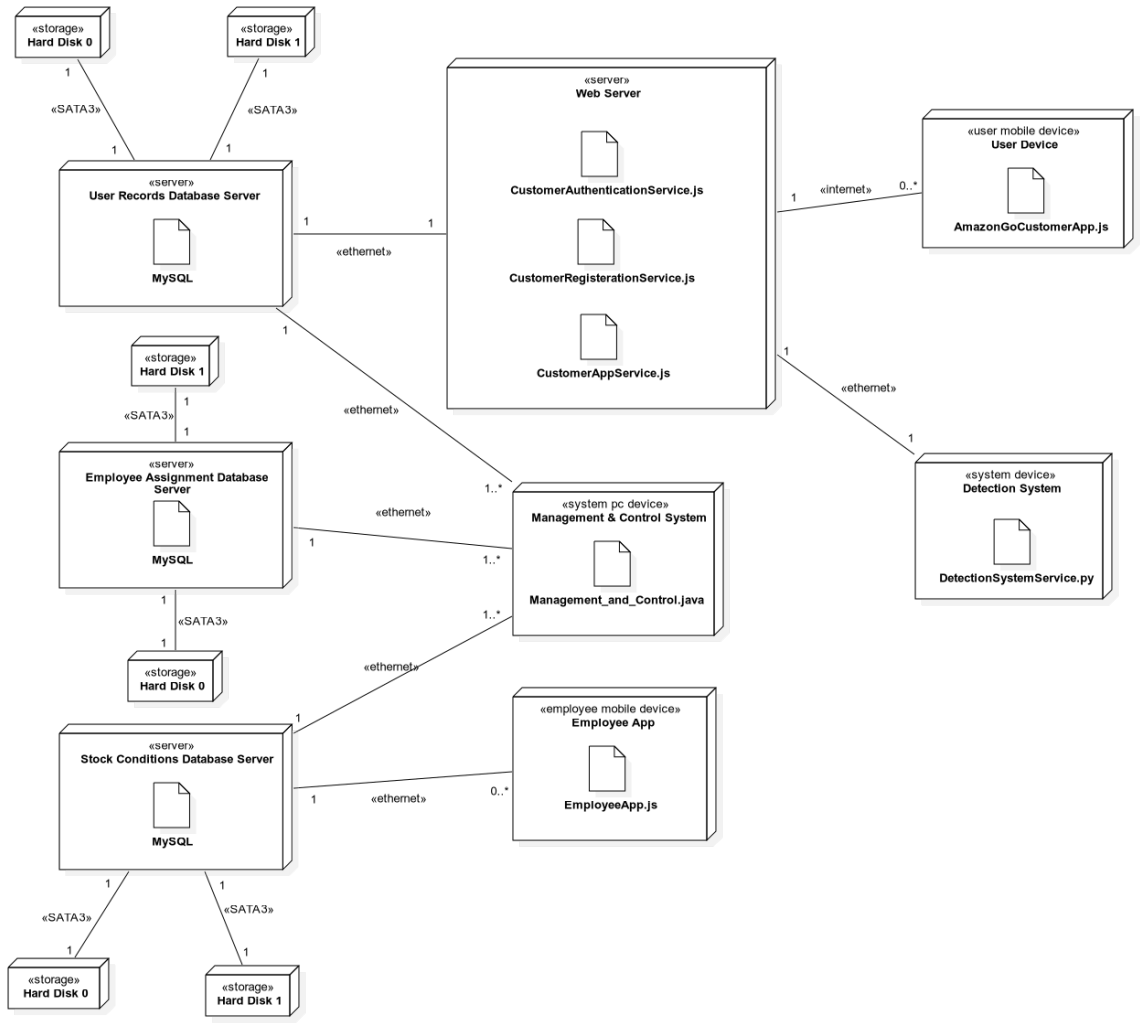

Figure 4.4: Deployment Diagram

Design Rationale:

- The Database Servers and the Web Server are separated from each other in-order to achieve scalability. Database Servers are one of the most significant parts of the system, therefore there might be changes in them in the future use (considering requirements of increasing the database size). By separating these servers, this kind of changes are much more manageable.
- There are two different database servers. One of them is User Records Database Server and the other one is Stock Conditions Database Server. Both of these servers are accessed frequently for both read and write operations. Therefore to access both type of data more efficiently (and without blocking each other), these

servers are separated from each other. Also it is useful as there is a grouping between two different types of data.

- There is an ethernet connection between the Web Server, Database Servers, Management & Control System, Employee App and Detection System to localize the connection and make it impenetrable to outer attacks.
- Python is used in Detection System as a programming language as it contains lots of libraries and modules in the field of Deep Learning (Deep Learning algorithms are used in the Detection System).
- Applications are implemented with Java Script as its React Native library is compatible with most of the mobile platform operating systems.
- Web Server components are implemented with Java Script as it is widely used in web development.
- MySQL is used as the relational DBMS as it is an open source product and very easy to use.
- There are 2 different hard disks with the same content for both of the databases. RAID 1 is used and mirroring is achieved. Therefore the system is much more reliable as all the data in the system has a backup.

4.3 Information View

In this viewpoint, organization and structure of the data that will be stored in the databases will be shown. Also the effects of the operations on the data will be assessed in terms of CRUD operations.

4.3.1 Service Interfaces

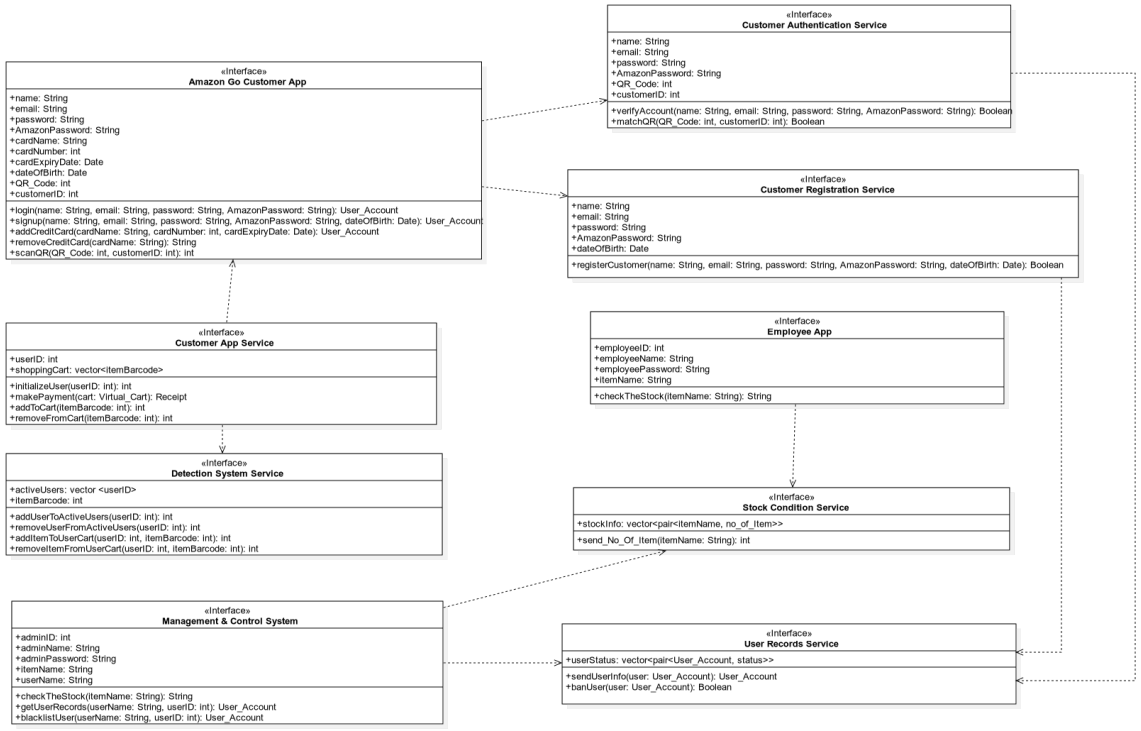

Figure 4.5: Interfaces Class Diagram

Operation	**Description**
login	The user enters the required info (via the Amazon Go Customer App), to login to his/her user account and these info are sent to Customer Authentication Service for verification.
signup	The user enters the required info (via the Amazon Go Customer App), to get registered to the system and these info are sent to Customer Registration Service.
addCreditCard	The user can add a new credit card to his/her user profile inorder to manage payment.
removeCreditCard	The user can remove a credit card that was already added to his/ her user profile.
scanQR	The Customer App forms a QR code for the enterance to the store. A special integer number is sent to Customer Authentication Service to match the QR code with the customer ID.
verifyAccount	The user account information are verified with the records in the database system. (via Customer Authentication Service)

matchQR	The user ID is matched with the QR code that was scaned to the scanning gates in entry. (via Customer Authentication Service)
registerCustomer	The user account registration info which are sent by the Customer App are stored to the database records. (via Customer Registration Service)
initializeUser	When a user enters a store, this function sends the user ID to the Detection System Service, therefore the user is initialized.
makePayment	When a user leaves the store, it recieves a signal from the Detection System Service, and makes the payment of the items that are in the user's virtual cart and forms a receipt.
addToCart	This function recieves signal from the Detection System Service, and adds the item, that the user took from the shelf, (detected by the system) to the virtual cart of the user.
removeFromCart	This function recieves signal from the Detection System Service, and removes the item, that the user placed back to the shelf, (detected by the system) from the virtual cart of the user.
addUserToActiveUsers	This function recieves a signal from the Customer App Service, and initializes the user by adding his/her user ID to the active users list.
removeUserFromActiveUsers	As Detection System notices that the user left the store, it calls this function, this function removes the user from the active users list and sends a special integer to the Customer App Service in-order to manage payment.
addItemToUserCart	As Detection System notices that the user took an item from the shelf, it calls this function. This function sends the barcode of the selected item, followed by a special integer to the Customer App Service.
removeItemFromUserCart	As Detection System notices that the user took an item from the shelf, it calls this function. This function sends the barcode of the selected item, followed by a special integer to the Customer App Service.
checkTheStock	This function sends a special query string to the Stock Condition Service, to check the stock condition of the item which is given as parameter.
send_No_Of_Item	This function returns the number of item in the stock which was given as parameter. (Database operation)
getUserRecords	This function sends a special query string to the User Records Service, to check the user account info of the user who is given as parameter.
blacklistUser	This function sends a special query string to the User Records Service, to blacklist the user account of the user who is given as parameter.
sendUserInfo	This function returns the user account info of the user who was given as parameter. (Database operation)
banUser	This function changes the status of the user account to 'FORBIDDEN' who was given as parameter. (Database operation)

Table 4.16: Service interfaces operation descriptions

Operation	Inputs	Outputs	Exceptions
login	name email password AmazonPassword	User_Account (class) object (Only necessary members, related to the login operation, are assigned to a value in the object, they are also given as parameters to the function)	-Database connection error occurs
signup	name email password AmazonPassword dateOfBirth	User_Account (class) object (Only necessary members, related to the signup operation, are assigned to a value in the object, they are also given as parameters to the function)	-Database connection error occurs
addCreditCard	cardName cardNumber cardExpiryDate	User_Account (class) object (Only members which are related to credit card info are assigned to a value)	-Database connection error occurs
removeCreditCard	cardName	Name of the credit card as string	-Database connection error occurs
scanQR	QR_Code customerID	An integer value which is a combination of customer ID, QR code and special integer value which indicates initialization	-Database connection error occurs -An erroneous QR code is generated
verifyAccount	name email password AmazonPassword	True if user account info match with the records in the database system, else false	-The user, with the given information can't be found in the database, therefore the user can't be verified
matchQR	QR_Code customerID	True if QR Code and customer ID are matched successfuly, else false	-The corresponding userID is erroneous
registerCustomer	name email password AmazonPassword dateOfBirth	True if customer is successfuly registered to the database system, else false	-A user with these information already exists

initializeUser	userID	An integer value which is a combination of user ID and a special integer value which indicates initialization for detection system	-Connection error with the Detection System Service occurs
makePayment	cart	Receipt (class) object which is formed as the customer finishes shopping	-Connection error with the Detection System Service occurs
addToCart	itemBarcode	Item barcode value as an integer	-Connection error with the Detection System Service occurs
removeFromCart	itemBarcode	Item barcode value as an integer	-Connection error with the Detection System Service occurs
addUserToActiveUsers	userID	An integer value which is a combination of user ID and an special integer value which indicates the 'active' status for an user	-The corresponding userID is erroneous
removeUserFromActiveUsers	userID	An integer value which is a combination of user ID and an special integer value which indicates the 'deactive' status for an user	-The corresponding userID is erroneous
addItemToUserCart	userID itemBarcode	Item barcode value as an integer	-The corresponding userID is erroneous -The corresponding itemBarcode is erroneous
removeItemFromUserCart	userID itemBarcode	Item barcode value as an integer	-The corresponding userID is erroneous -The corresponding itemBarcode is erroneous
checkTheStock	itemName	A query as a string which includes the name of the item	-Database connection error occurs
send_No_Of_Item	itemName	The number of items as an integer	-The itemName can't be found in the database system

getUserRecords	userName userID	User_Account (class) object (Only userName and userID members are set, for the database query operations)	-Database connection error occurs
blacklistUser	userName userID	User_Account (class) object (Only userName and userID members are set, for the database query operations)	-Database connection error occurs
sendUserInfo	user	User_Account (class) object (All available members, in the record, are set)	-The user, with the given information as an input, can't be found in the database system
banUser	user	True if operation is successful, else false	-The user, with the given informations as an input, can't be found in the database system -The user was already banned

Table 4.17: Service interfaces operation design

Design Rationale:

- Detection System Service and Customer App Service are have a high interaction rate. Customer App Service calls initializeUser() function, and sends the output to the Detection System Service, therefore the user becomes one of the active users in the store. Detection System Service calls addItemToUserCart() or removeItemFromUserCart() functions and sends the output to Customer App Service to modify the users virtual cart.
- Stock Condition Service and User Records Service, both perform database related operations.
- User can perform both login() and signup() operations with the Amazon Go Customer App. The output of login() function is sent to Customer Authentication Service, and output of signup() function is sent to Customer Registration Service. Necessary operations are performed by them.
- Both the Management & Control System and Employee App, mostly form queries and sent the outputs of their operations to the related Database APIs.

- There are two different database APIs (for stock info and user records), in-order to access these two different type of data in paralel.

4.3.2 Database Operations

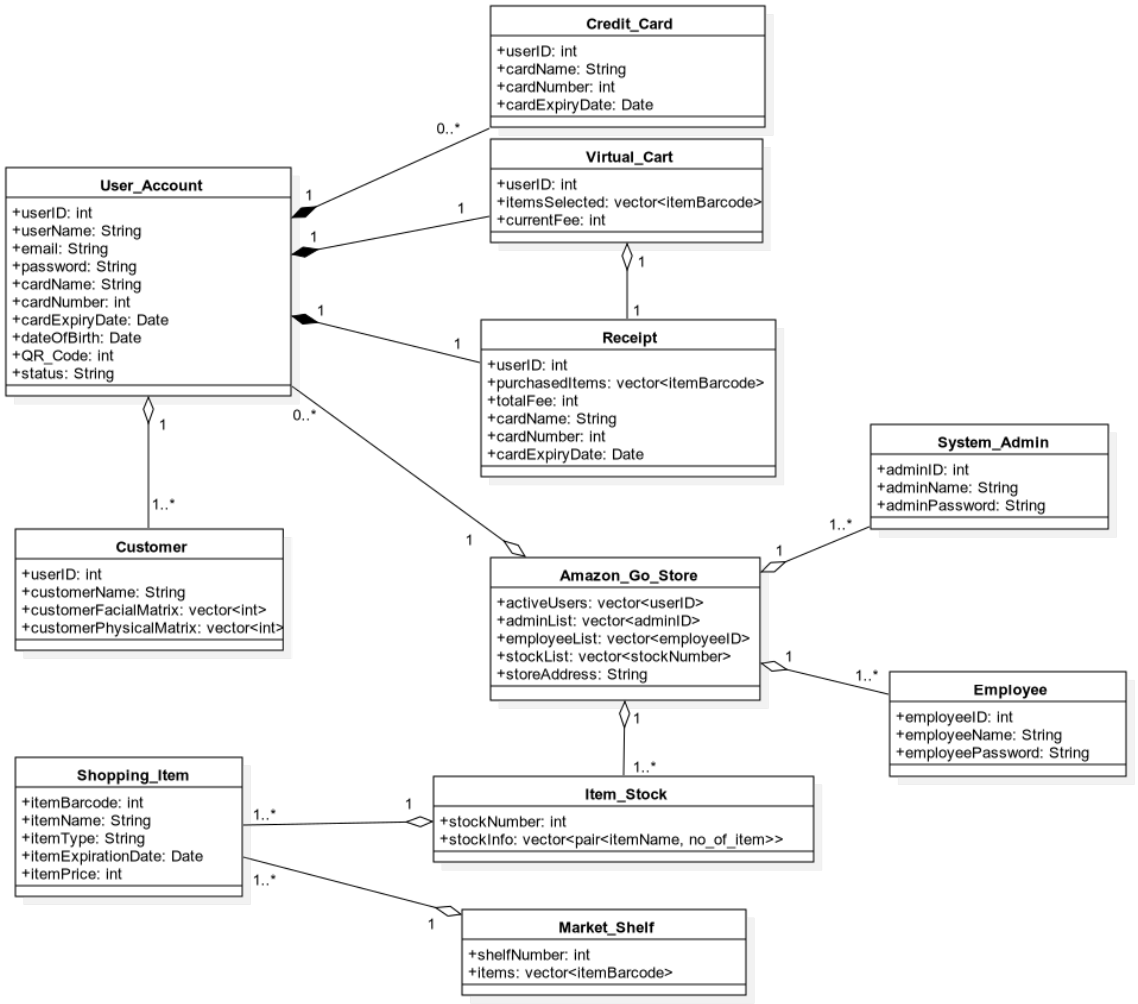

Figure 4.6: Logical Database Diagram

Operation	**CRUD Operations**
login	Create: Read: User_Account Update: Delete:
signup	Create: User_Account, Credit_Card, Customer Read: Update: Delete:
addCreditCard	Create: Credit_Card Read: Update: Delete:
removeCreditCard	Create: Read: Credit_Card Update: Delete: Credit_Card
scanQR	Create: Read: Update: User_Account Delete:
verifyAccount	Create: Read: User_Account Update: Delete:
matchQR	Create: Read: User_Account Update: Delete:
registerCustomer	Create: User_Account Read: User_Account Update: Delete:
initializeUser	Create: Read: User_Account, Customer Update: Delete:
makePayment	Create: Receipt Read: Virtual_Cart, Credit_Card Update: Delete: Virtual_Cart
addToCart	Create: Read: Shopping_Item Update: Delete:

removeFromCart	Create: Read: Shopping_Item Update: Delete:
addUserToActiveUsers	Create: Read: User_Account, Customer Update: Amazon_Go_Store Delete:
removeUserFromActiveUsers	Create: Read: User_Account, Customer Update: Amazon_Go_Store Delete:
addItemToUserCart	Create: Read: Shopping_Item, User_Account, Customer Update: Virtual_Cart, Item_Stock Delete:
removeItemFromUserCart	Create: Read: Shopping_Item, User_Account, Customer Update: Virtual_Cart, Item_Stock Delete:
checkTheStock	Create: Read: Item_Stock Update: Delete:
send_No_Of_Item	Create: Read: Item_Stock Update: Delete:
getUserRecords	Create: Read: User_Account Update: Delete:
blacklistUser	Create: Read: Update: User_Account Delete:
sendUserInfo	Create: Read: User_Account Update: Delete:
banUser	Create: Read: Update: User_Account Delete:

Table 4.18: CRUD Operations

Design Rationale:

- MySQL is the relational DBMS which is used in the system.
- One or more than one Customer can use the same User_Account. Therefore the physical info of the customers (required for Detection System) must be matched with the User_Account.
- Credit_Card, Virtual_Cart and Receipt are weak entities of User_Account as they wouldn't exist on their own, without being connected to an User_Account.
- Single Receipt can be formed from a single Virtual_Cart.
- There is only one store. Therefore it has an one-to-many relationship with System_-Admin, Employee and Item_Stock (there must exist at least one entity of each type). Also it has an one-to-many relationship with User_Account, but in a selected time, there can be zero active users in the store.
- Both Item_Stock and Market_Shelf have an one-to-many relationship with the Shopping_Item as there can be one or more than one items in a single stock or a single shelf.

4.4 Interface View

In this viewpoint, both the internal interfaces and the external interfaces will be explained and specified in details.

4.4.1 Internal Interfaces

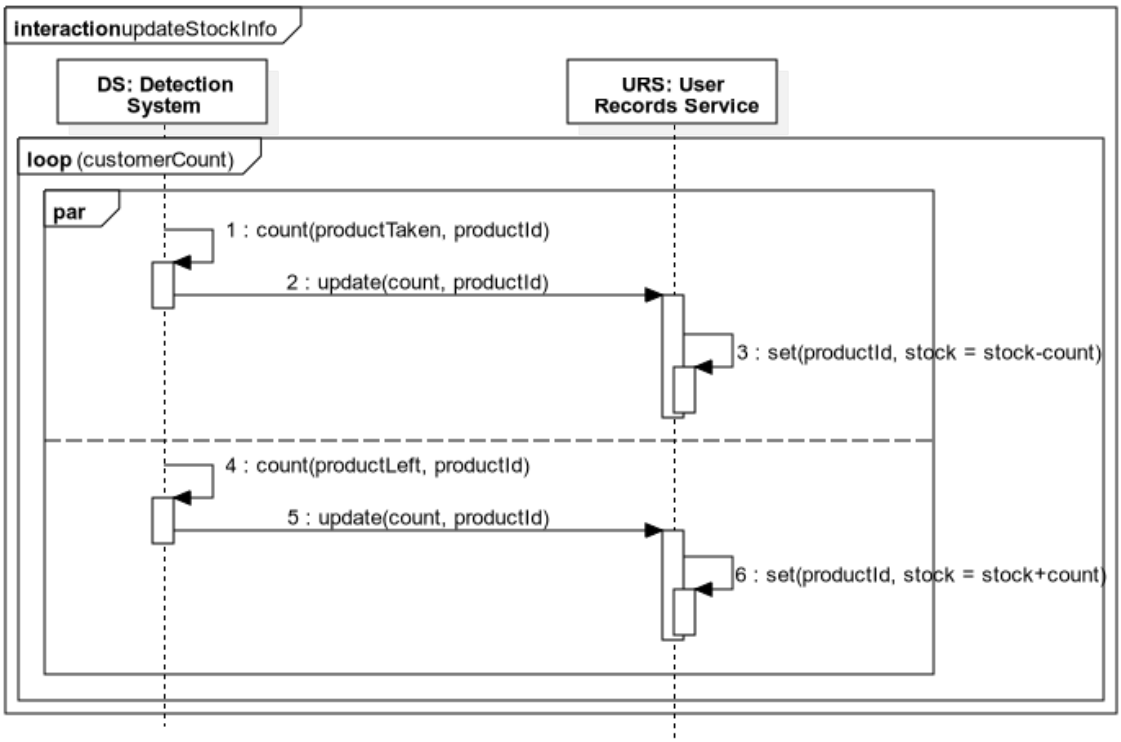

Figure 4.7: Update Stock Info Usecase Featuring The Interface Between Detection System and User Records Service

The Interface Between Detection System and User Records Service: The username, password pairs created at registration are matched with data (bodily habitus, facial recognition etc.) from the detection system to create a full profile with the help of this interface.

In the shopping following above matching, the detection system accesses this full profile from the user records and tracks the customer in Amazon Go with that data.

Additionally, collection of information about the customer's shopping preferences happens here.

Design Rationale:

- The shopping preferences of the customer are not to be shared with/sold to no other entity than Amazon's own partners.

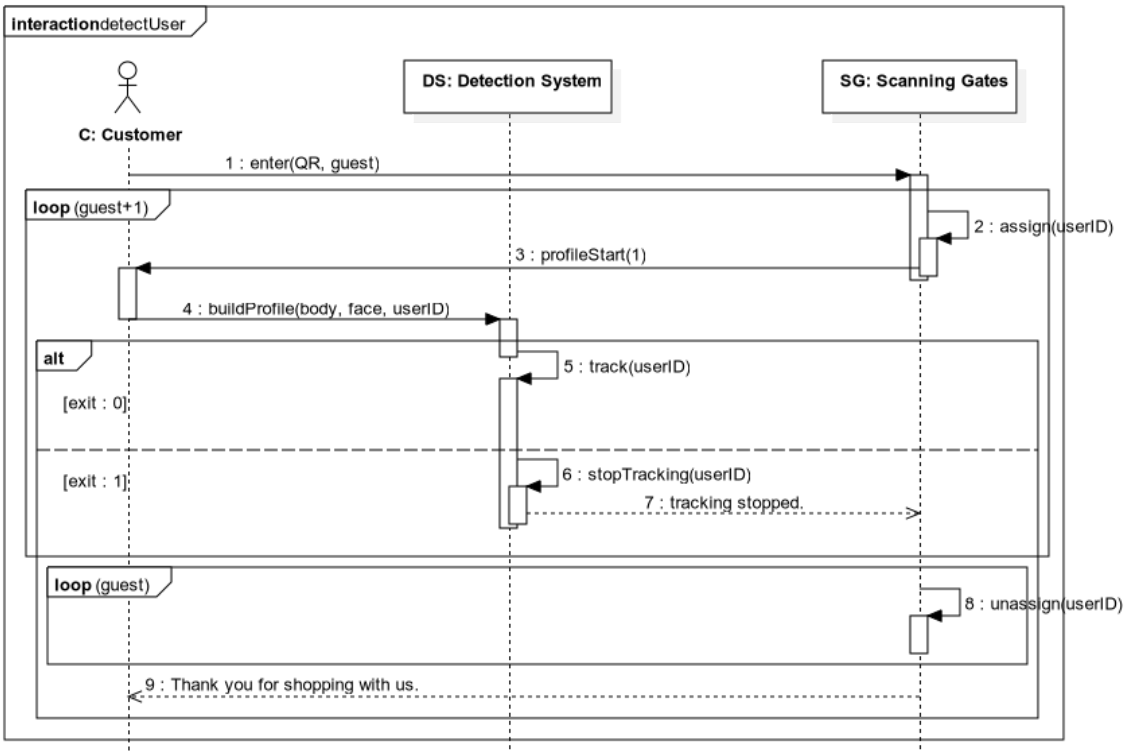

Figure 4.8: Detect User Usecase Featuring The Interface Between Detection System and Scanning Gates

The Interface Between Detection System and Scanning Gates: This interface matches the user ID created from the QR code and the bodily habitus, facial recognition etc. data and uses that joint data to track that user's shopping around the store.

Design Rationale:

- The shopping habits, bodily habitus, facial recognition etc. associated with the user is not discarded upon customer exit (but rather sent to different interfaces to update user records).

The Interface Between Customer Authentication Service and User Records Service: The customer authentication service takes previously entered customer login data and compares the password associated with the username to the username,password pair supplied by the User Records Service. If it's a match, then the login is authorized.

Design Rationale:

- Password is stored by the User Records Service in hashed form due to security concerns.

- In case of a mismatch appropriate error message should be formed.

The Interface Between Customer Registration Service and User Records Service: The customer registration service takes previously entered customer sign-up data and compares the username to all the other usernames in user records service. If there is no match registration commences. Otherwise, it is terminated.

Design Rationale:

- Newly entered password should be hashed and sent to User Records Service as such.
- Appropriate error message should be prepared if username is already in the User Records.

4.4.2 External Interfaces

4.4.2.1 User Interfaces

The Interface Between Amazon Go Customer App and Customer Authentication Service: This user interface takes username and password information from the user. This information is necessary for the login process and will later be passed on to the Interface Between Customer Authentication Service and User Records Service to start the sign-in procedure.

Design Rationale:

- If the user is not already logged in, the app will display a sign-in form first upon app launch.
- An option to display password is presented to the user, so they can avoid mistyping.
- The agreements the user becomes subject to upon sign-in are names right below the sign-in button with links provided for each for ease of access.

The Interface Between Amazon Go Customer App and Customer Registration Service: Via this interface new username,password pairs are accepted from the customer, to be passed to the Interface Between Customer Registration Service and User Records Service to start the registration procedure.

Design Rationale:

- The app will display a create account button upon app launch right below the sign-in form for existing users, enabling quick and uncomplicated registration access.
- An option to display password is presented to the user once they proceed to the registration page, so they can avoid mistyping.
- Password requirements are displayed on the sign-up page to eliminate ambiguity.

The Interface Between Amazon Go Customer App and Customer App Service: This is the GUI giving the user a look into what is happening during their shopping, such as updates to their virtual cart and their owed total.

Design Rationale:

- The QR code on the launch page of the app is discarded and replaced with the virtual cart and total of the user upon entry for easy tracking.

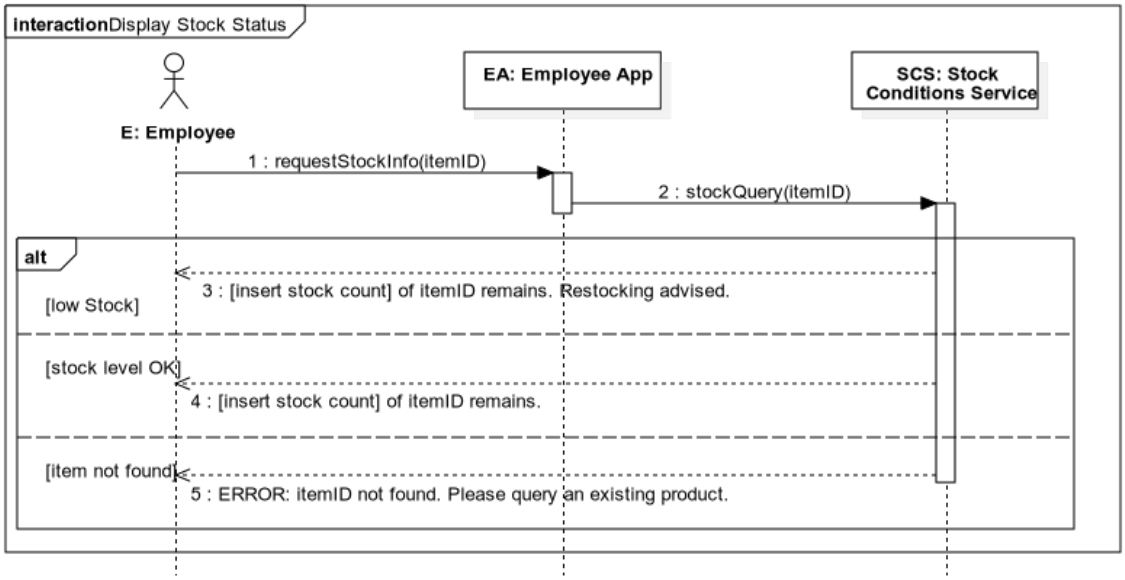

Figure 4.9: Display Stock Status Usecase Featuring The Interface Between Employee App and Stock Conditions Service

The Interface Between Employee App and Stock Conditions Service: The channel to control and query after stocks of a specific product.

Design Rationale:

- Easy access from the employee app launch page using the "Display Stock Status" button
- A search bar to look for certain items.
- Item lookup can also be done using dotcodes which can be scanned using smartphone cameras.

4.4.2.2 System Interfaces

The Interface Between Management & Control System and User Records Service: This interface enables admins to reach non-sensitive information about the customers that they may need, such as billing address, refund history etc.

Design Rationale:

- User Record Query functionality is added to the employee app methods on admins' launch pages for fast and easy access.

The Interface Between Management & Control System and Stock Conditions Service: Admin channel for controlling and querying after stocks of a specific product.

Design Rationale:

- Apart from the design they share with the Employee App, they have an extra button to place an order for low stock products.

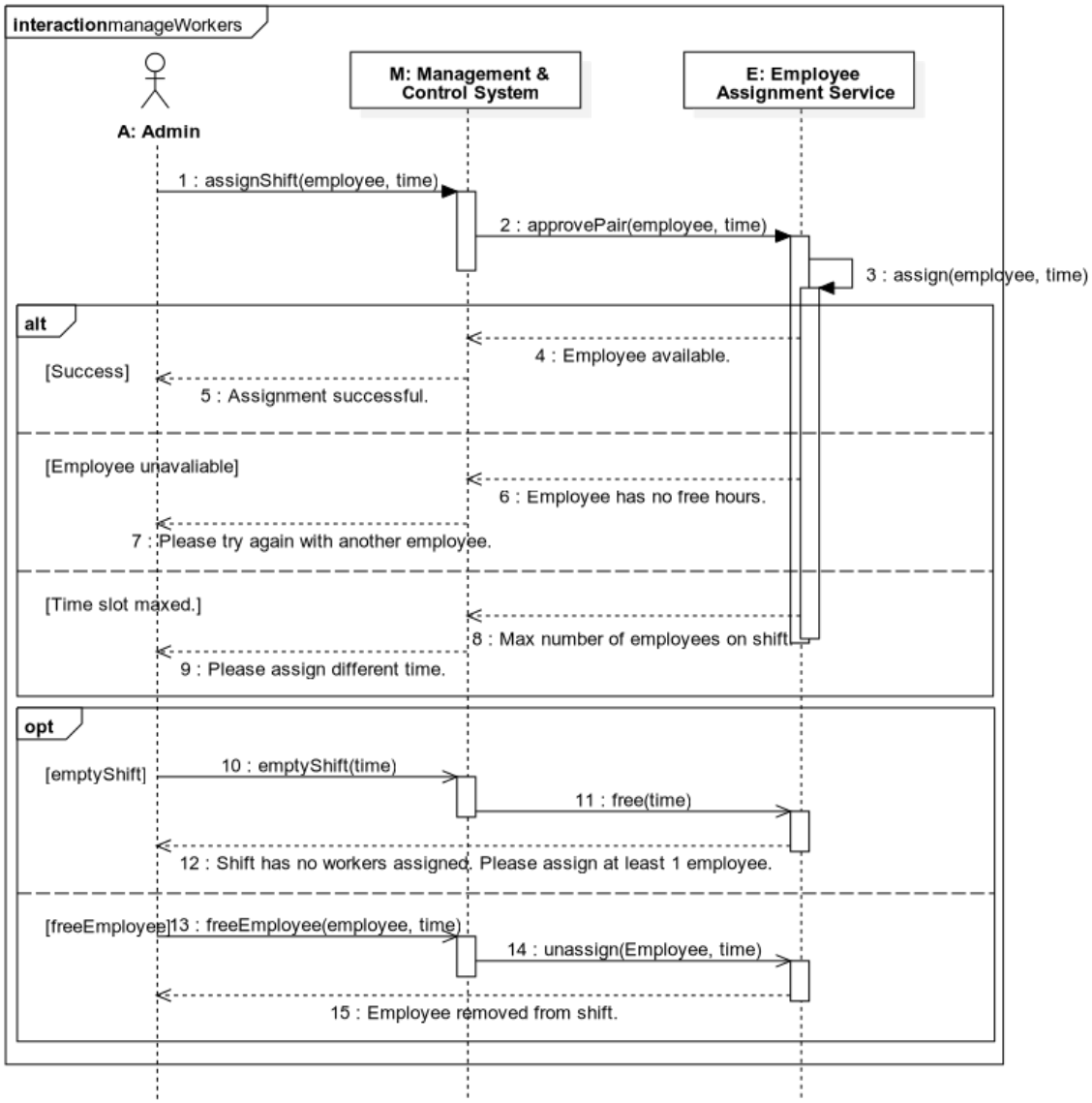

Figure 4.10: Manage Workers Usecase Featuring the Interface Between Management & Control System and Employee Assignment Service

The Interface Between Management & Control System and Employee Assignment Service: This interface aids admins in arranging shifts, enables the "Manage Workers" usecase. Via this interface, admins can change employees working certain hours to different hours or change the hours to be worked.

Design Rationale:

- Access to interface is through the employee app. Upon launch admins simply click

"Manage Workers".

- A GUI in the form of a calendar is presented to the admins, so they can easily plan and see the results of the shift changes.

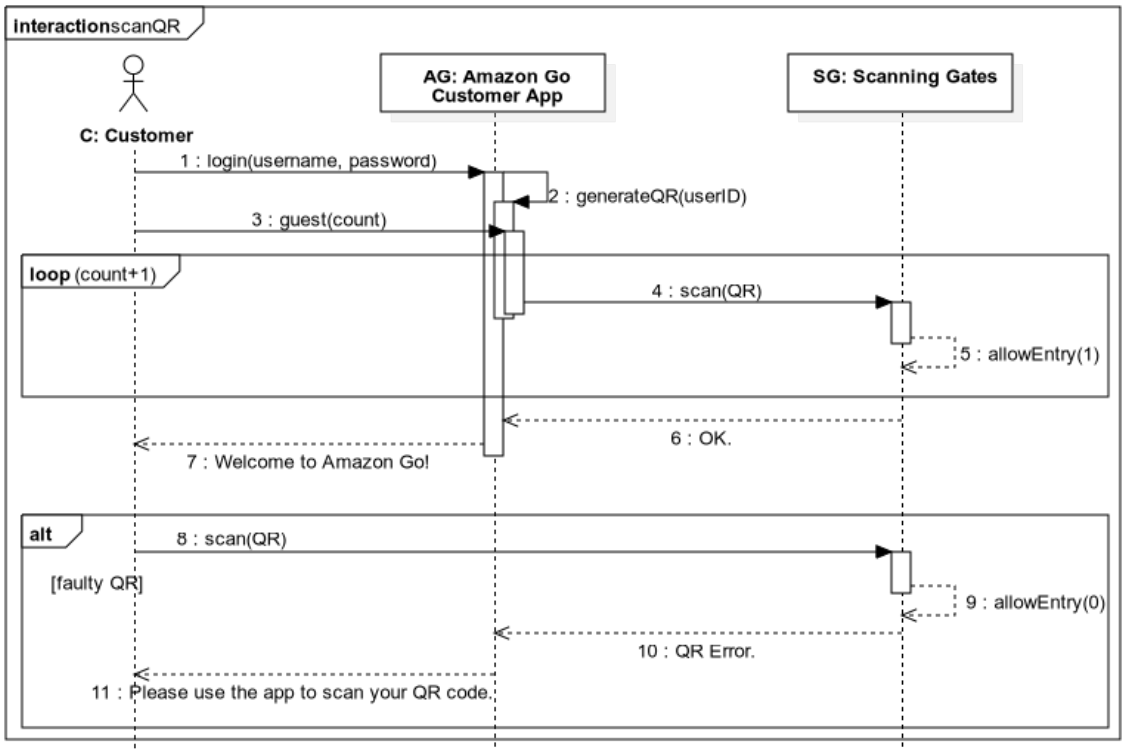

Figure 4.11: Scan QR Usecase Featuring The Interface Between Scanning Gates and Amazon Go Customer App

The Interface Between Scanning Gates and Amazon Go Customer App: The entry into Amazon Go stores is enabled through this interface. The doors slide open for those with functional QR code available on their mobile app only, i.e. screenshots of the QR code etc. are not accepted by the scanning gates on this interface.

Design Rationale:

- The QR code is generated and displayed to the user upon launch, so users do not have to spend time generating it following a long complex path through the menus.

The Payment Interface: Here customers can employ the "Manage Payment" usecase and add or remove cards, update existing payment methods, apply for refunds etc. This interface interacts with the scanning gates upon exit to finalize shopping payment.

Design Rationale:

- This interface stores sensitive information like credit card numbers and thus is protected with an EV SSL Certificate.
- As to not clutter the customer app's launch page, the methods on this interface is reachable from the bottom menu, following "More" > "Settings" > "Payment Cards".

The Messaging Interface: This interface is for messaging within the Amazon Go app. It has two sides: Employee side and customer side. Customers can use the email service integrated into the Amazon Go customer app from "More" > "Contact Us" > "Email Customer Service". Employee side of the app enables in-store workers to communicate with one another. Employees can message the admin to request certain shift arrangements and to notify them about restocking needs.

Design Rationale:

- On the Employee App messages are separated from the functions displayed upon launch. The inbox is located on the far right corner of the bottom menu and displays a red bubble with the number of messages unread written on it when unread messages are present.
- As customers would not frequent the app unless they are in the store, the replies to their customer service requests are sent directly to the email associated with their account, where they are more likley to be seen quickly.