SOFTWARE DESIGN DESCRIPTION

Amazon Go

Deniz Polat - 2237790

Hüsna Yılmaz - 2237964

Change History

Version	**Date**
1.0	06.09.2020
2.0	06.20.2020

Table of Contents

List of Figures

List of Tables

1 Introduction

This is a Software Design Description document for the system Amazon Go. The software and equipment mentioned in this document are the system itself and associated tools used for this system. Purpose of this document is to provide a design description for the system.

1.1 Purpose of the System

The purpose of Amazon Go system is to reduce the time and effort spent by customers during their shopping by allowing customers to purchase products without being checked out by a cashier or using a self-checkout station. Instead, customers show the QR code and use the Bluetooth technology when they both *enter* and *leave* the store. Between these two actions, they do shopping just as in regular stores.

1.2 Scope

This documentation describes the Amazon Go system at an architectural level with its context diagram, use cases and their tabulated descriptions, components of the system, hardware equipment that the system include, internal and external interfaces with class diagrams, operation designs and detailed description of the interaction between these interfaces, and the system database description with a class diagram and a table describing all create, read, update, delete operations done in each operation in every class.

1.3 Stakeholders and Their Concerns

Customers Customers are people who has at least once visited an Amazon Go Store, or even has an Amazon Go Account. Their main concern is to do their shopping from Amazon Go Stores and let the system have some marketing-purposed information about them.

Retailers Retailers are the people or companies who supply products for Amazon Go Store to sell. Their concern is to act like a meta-store to the system's stores, i.e. supplying them products which are being sold in the system's stores.

Staff Members Staff members include all people showing up in stores and IT Staff members. Their concern is to handle errors, testing the system frequently for security, updating the system when needed and solve basic problems before they become bugs and make necessary changes in order to improve the system.

Marketing Specialists Marketing specialists are the ones who prepare advertisement materials. Their concern is to prepare materials in order to use in stores and sent to customers by using the information coming from data scientists and the system's database.

Data Scientists Data scientists are people in Amazon Go system who analyze data about customers and products and send the results of these data to marketing specialists. Their concern is to easily access the data via the system's database interface, read and analyze data collected for the system.

2 References

This document is written with respect to IEEE 1016-2009 standard:

IEEE Standard for Information Technology–Systems Design–Software Design Descriptions. Institute of Electrical and Electronics Engineers, 2009.

Other Sources:

Amazon.com: : Amazon Go. (n.d.). Retrieved June 3, 2020, from https://www.amazon.com/b?node=16008589011

Martin, T., Wang, H., Artis, M. W. J. J. D., & Uncleback, A. Amazon Go! Cashierless Retail Analysis.

3 Glossary

Term	**Definition**
DBMS	Database Management System
Companion	Candidate person to enter an Amazon Go Store.
formDATA	Information about the user himself/herself filled through any kind of form.
Store	Physical Amazon Go Store.
WSC	Amazon Go Web Server Controller
SQL	Structured Query Language
Query	Requesting information.
ADScreen	Digital billboards in Amazon Go Stores.
Server	A computer or computer program which manages access to a centralized resource or service in a network.
URL	Uniform Resource Locator, web address
User	Candidate person to enter an Amazon Go Store.
Customer	Any person doing shopping in an Amazon Go store.
meta-store	All places, people and associations supplying products for Amazon Go stores, i.e. retailers.
AUMS	Amazon User Management System
SAS	Serial Attached Small Computer System, a computer bus interface for transferring data between a computer and any kind of storage device.
SATA	Serial Advanced Technology Attachment, a computer bus interface for transferring data between a computer and any kind of storage device.
Cookie	Small data that user has supplied via computer and stored to remember back with the same computer.

Table 1: Glossary

4 Architectural Views

4.1 Context View

In this viewpoint, the interactions between systems, services and users are considered. While Context Diagram is basically showing the connections, with the Use Case Diagram and Use Case Descriptions interactions are expressed in detail. These details will be instructive for the design and implementation of Amazon Go.

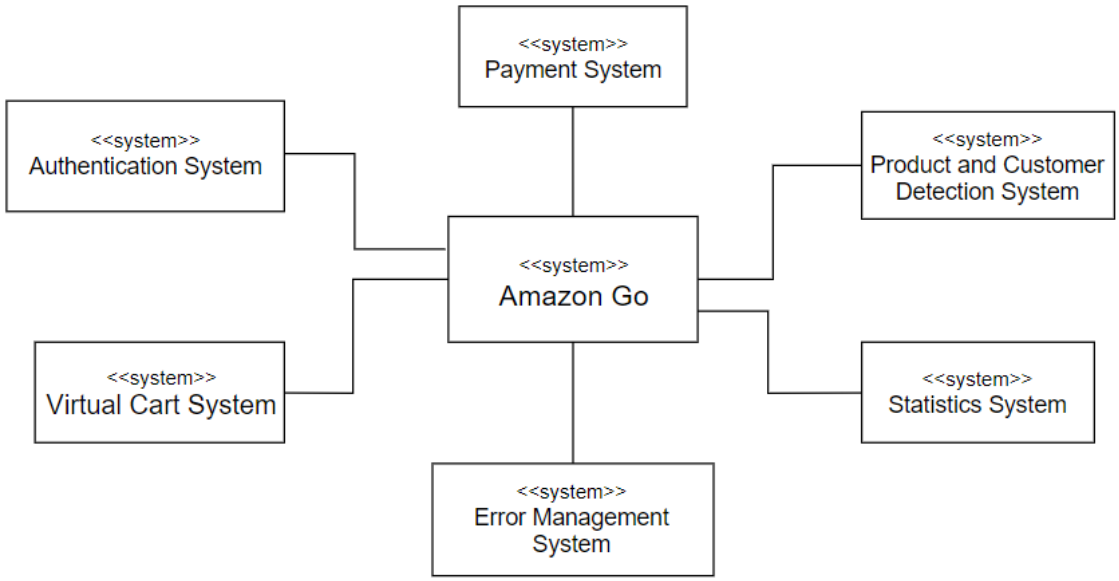

Figure 1: Context Diagram

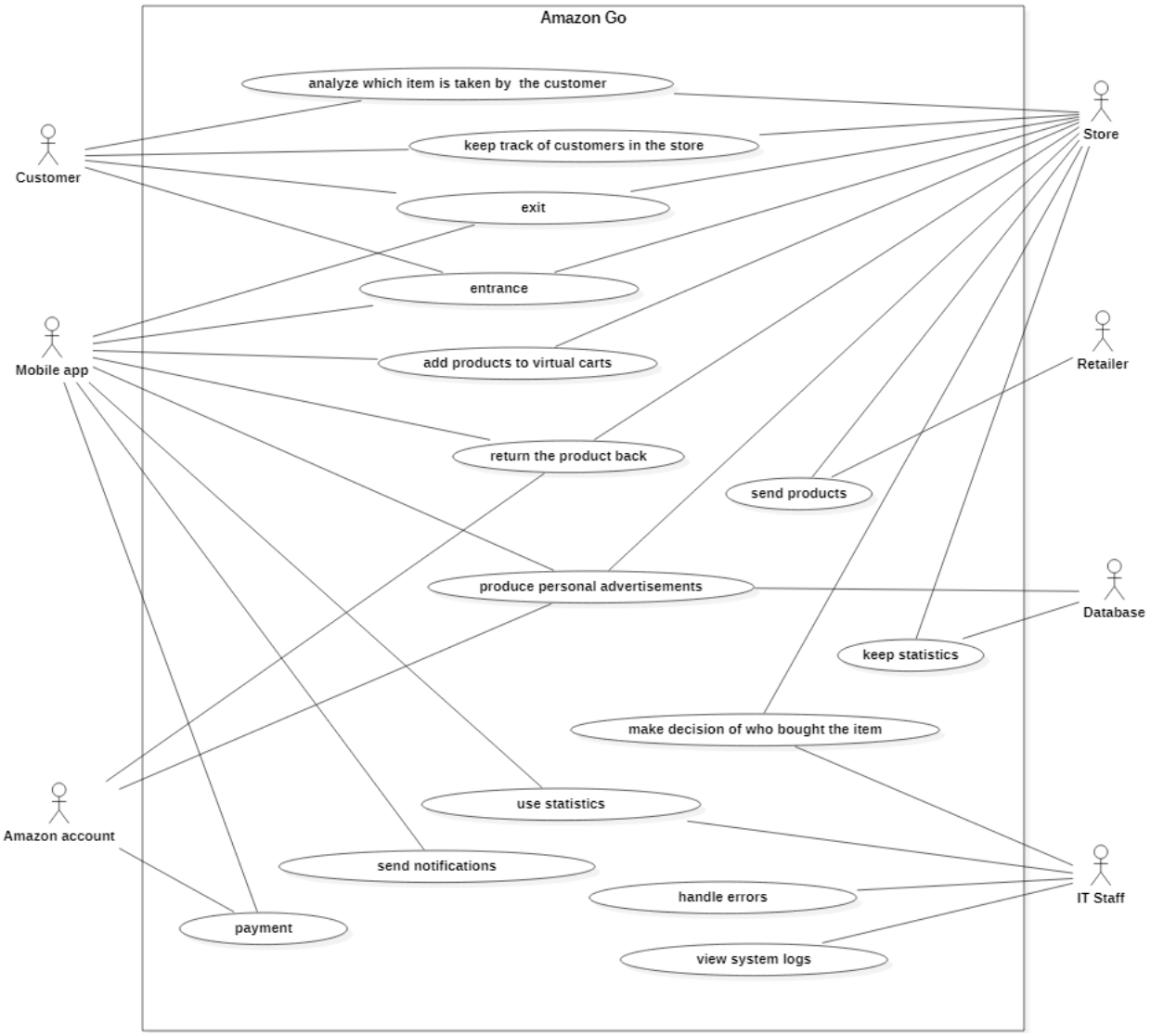

Figure 2: Use Case Diagram

Use Case Name	Make decision of who bought the item
Actors	IT Staff, Store
Description	When a user grabs an item from the shelf, this item is instantaneously added to the customer's virtual cart.
Data	Camera recordings, camera-based tracking system, IT Staff's observations
Preconditions	Customers and their companions (if any) must scan the QR code from Amazon Go mobile app when they enter the store. Moreover, there should be at least one staff member as back-up.
Stimulus	When an item is picked up, the black-box technology awakes the load and weight sensors on shelves and cameras recording the right angle..
Basic Flow	Step 1 - Cameras start keeping track of new customers. Step 2 - When an item is taken from a shelf, black-box technology of the store determines who bought it with the help of relevant shelf's sensors and cameras on ceiling. Step 3 - The item is being added to the detected user's virtual cart.
Alternative Flow	Step 3 - If the customer taking the item is detected correctly and added to that customer's virtual cart but the product is not at that customer when (s)he leaves the shop, the user can remove it from virtual cart. Step 4 - When a removal happens, IT Staff is informed and staff members decide whether the removal is appropriate or not according to camera records.
Exception Flow	If the system is not able to detect who has taken the item due to system failure or ethical issues, then the system sends an error to staff and staff members take over.
Postconditions	Item is added to detected customer's virtual cart.

Table 2: Make decision of who bought the item

Use Case Name	Produce personal advertisements
Actors	Database, Store, Amazon Account, Mobile App
Description	The data retrieved from customers' Amazon account and examined by store is collected in the database, Amazon Go blackbox technology produces personal advertisement technology using this data. Mobile app and store is used to broadcast the advertisement.
Data	Personal data as customer interests, ages, genders, income level, the time they take in each aisle and their purchase history
Preconditions	None (Stepping into Amazon Go, or having an account is more than enough.)
Stimulus	Start using Amazon Go app
Basic Flow	Step 1 - Customer downloads the Amazon Go app, and logs in with Amazon account. Step 2 - Customer enters an Amazon Go store with the aid of mobile app. Step 3 - Store technology keeps track of the customer, saves customer's path and movements,the time taken to decide and in the each aisle. Step 4 - Amazon Go technology retrieves personal data and interests from the customer's Amazon account, store examines the customer's behaviours in detail, and this collection of data simultaneously sent to the database. Step 5 - The blackbox technology makes the necessary calculations and sends the results to store billboards in the aisles which the customer currently in and mobile app. Step 6 - Customer encounters personalised ads on the billboards and in the mobile app.
Alternative Flow	Step 2 - If with a mobile app, multiple customers enter the store Amazon Go, marks every companion with the same account name. Step 3 - Store technology keeps track of account holder and the companions, saves their path and movements,the time taken to decide and in the each aisle. Step 4 - Basic flow works for the account holder. Step 5 - The companions don't buy anything or add anything to virtual cart. Step 6 - Store analyzes the behaviours of the companions and creates advertisements based on their locations in the store and time they take in the aisles. Step 7 - The data collected by store about the companions is not saved in the database and deleted immediately after they left the store.
Exception Flow	If the companions cause a product to enter to the virtual cart, even though they don't buy it Amazon Go do not use any data to the related shopping and deletes the retrieved data from the database as the companions' purchase may lead to miscalculations for the personalized ads.
Postconditions	Personalized ads increase the profit.

Table 3: Produce personal advertisements

Use Case Name	Entrance
Actors	Customer, Store, Mobile App
Description	User who has the Amazon Go mobile app walks into the store.
Data	-
Preconditions	Having a functioning mobile app and Amazon Go-QR Code
Stimulus	Scanning the QR Code to the gate machine
Basic Flow	Step 1 - Scan the Amazon Go-QR Code to the gate machine. Step 2 - QR Code is checked, whether it is valid or not. Step 3 - Gate opens and lets the customer in.
Alternative Flow	-
Exception Flow	If the QR code does not function gate does not let the customer in.
Postconditions	The customer enters the store.

Table 4: Entrance

Use Case Name	Exit
Actors	Customer, Store, Mobile app
Description	Leaving the store with the bought goods.
Data	The data of the virtual cart is transferred to the receipt.
Preconditions	Making a transaction of 1$.
Stimulus	Awake the gate sensors.
Basic Flow	Step 1 - When a customer is leaving the store, gate sensors determine the customer. Step 2 - Transaction payment area tries to make a transaction of 1$ from the leaving customer's virtual cart. Step 3 - After making the transaction, the customer leaves the store.
Alternative Flow	-
Exception Flow	Step 3 - If the transaction does not occur, the gate does not open and does not allow the customer to leave the store.
Postconditions	The customer is outside of the store.

Table 5: Exit

Use Case Name	Payment
Actors	Mobile App, Amazon account
Description	When a customer leaves the store, the customer's Amazon account is charged and (s)he is sent a receipt.
Data	The goods bought and their prices
Preconditions	Having an Amazon account and doing shopping.
Stimulus	User's leaving the store.
Basic Flow	Step 1 - User passes from the transaction payment area. Step 2 - The customer's Amazon account is charged by 1$. Step 3 - User leaves the store. Step 4 - The rest of the bill is charged from the customer's Amazon account. Step 5 - The customer is sent a receipt. Step 6 - Stocks are updated.
Alternative Flow	Step 2 - The customer's payment settings are not integrated with Amazon, therefore payment operation is done via other methods such as PayPal. Step 3 - The customer is sent a receipt. Step 4 - Stocks are updated.
Exception Flow	Payment is not rendered, and the customer has to deal with bank's policies.
Postconditions	The products on the customer's virtual cart is paid.

Table 6: Payment

Use Case Name	Send Notifications
Actors	Mobile app
Description	Sending notifications about promotions, nearest stores and so on.
Data	-
Preconditions	Having the mobile app
Stimulus	Data retrieved from database
Basic Flow	Step 1 - Data was recorded to the database based on the customer's previous shoppings, location permission and account information. Step 2 - Products that the customer is interested in are detected with the help of database and promotions about these products are sent to the user as notification.
Alternative Flow	Step 2 - If user is close to a store, then the app informs the user with a notification.
Exception Flow	-
Postconditions	The user is informed.

Table 7: Send Notifications

Use Case Name	Return the product back
Actors	Amazon account, Store, Mobile app
Description	After a customer does shopping, (s)he can return product(s) back.
Data	-
Preconditions	Having purchased any product.
Stimulus	Pressing *remove* button from the app.
Basic Flow	Step 1 - User presses *remove* button from the app. Step 2 - The item is removed from bill. Step 3 - User brings the product back to the store. Step 4 - The money is paid back to the customer.
Alternative Flow	-
Exception Flow	If the user does not bring the item back to the store, then returning does not happen.
Postconditions	The item is returned back and its price is paid back to the customer.

Table 8: Return the product back

Use Case Name	Add products to virtual carts
Actors	Store, Mobile app
Description	When a user picks an item from a shelf, the product is added to the customer's virtual cart.
Data	The details of the products in the virtual cart
Preconditions	Customer's being close to the related shelf.
Stimulus	Customer's taking a product from a shelf.
Basic Flow	Step 1 - Customer takes an item from the shelf. Step 2 - Item is added to the virtual cart. Step 3 - As long as the item has not put back to the shelf, it remains in the virtual cart.
Alternative Flow	Item is put back to the shelf and removed from the virtual cart.
Exception Flow	If a customer picks up an item and then gives it to another customer, the item is not removed from the virtual cart. The customer who takes the product from the shelf is charged.
Postconditions	Virtual cart is updated.

Table 9: Add Products to Virtual Carts

Use Case Name	Keep statistics
Actors	Database, Store
Description	Information about customers' shopping history and behaviours in store are kept in the database.
Data	-
Preconditions	There should be customer(s) in the store. Their doing shopping is a plus.
Stimulus	Any user in the store
Basic Flow	Step 1 - Customer enters the store. Step 2 - Cameras keep track of the customer while (s)he is in store. Step 3 - Sensors detect the items that customer has taken. Step 4 - System writes all these information to the database.
Alternative Flow	If customer has companions and companions picks up some products, the details of this shopping is marked, since the companions' behaviours can be misleading for the statistics.
Exception Flow	-
Postconditions	System enlarges its database.

Table 10: Keep Statistics

Use Case Name	Keep track of customers in the store
Actors	Store, Customer
Description	Using the hardware in store, the system keeps track of customers' moves.
Data	The coordinates of the customers and the time taken in the each aisle and in front of the each section.
Preconditions	There should be at least one user in the store.
Stimulus	A user's entering from the transaction area.
Basic Flow	Step 1 - A user enters the store passing by transaction area. Step 2 - Cameras in the store detects the user. Step 3 - Cameras having different angles record the customer's movements until (s)he leaves the store. Step 4 - If multiple users entered the store with the same account, system keeps track of the others, too.
Alternative Flow	-
Exception Flow	-
Postconditions	This data is recorded to database for statistical and marketing purposes and used while detecting whom to add the product.

Table 11: Keep Track of Customers in the Store

Use Case Name	Send products
Actors	Retailer, store
Description	When a product is needed in the store, it is supplied by retailers.
Data	The statistics of sales
Preconditions	Need of one or more specific product
Stimulus	System informs the retailer
Basic Flow	Step 1 - The statistics of sales are hold in a stock management system. Step 2 - Stocks are updated with each change of virtual carts and each payment. Step 3 - For each product, information of till when the stocks will be enough is kept in the stock management system. Step 4 - If store is about to get out of a specific product in a relatively close time, the regular retailer is notified and the system makes an automated order.
Alternative Flow	Step 4 - If there is no regular retailer or a new candidate retailer exists, the system decides retailer to give order. Step 5 - New order is created and the retailer sends chosen product(s) to given store.
Exception Flow	If the product is not available in the stocks unexpectedly, it staff is informed and it staff takes the necessary actions.
Postconditions	Stocks are updated.

Table 12: Send Products

Use Case Name	Use Statistics
Actors	Mobile App, IT Staff
Description	The statistics in the database can be seen and used by IT staff.
Data	All the information that is processed in the Amazon Go system, customer accounts, personal data, shopping behaviours etc.
Preconditions	The statistics are saved in the database.
Stimulus	IT Staff make use of statistics and controls the store stocks.
Basic Flow	Step 1 - The data in the database is reviewed by IT staff. Step 2 - According to the statistics the number of necessary products is determined.
Alternative Flow	-
Exception Flow	If there is an error in the database, the statics cannot be used and the IT Staff is informed.
Postconditions	The data is also can be used for other development and marketing purposes.

Table 13: Use Statistics

Use Case Name	Analyze which item is taken by the customer
Actors	Customer, Store
Description	An item is taken from the shelf by a customer
Data	The information of the section of the product and the data from the cameras
Preconditions	There is a customer near the section.
Stimulus	The item is taken by a customer.
Basic Flow	Step 1 - A customer picks up an item from the shelf Step 2 - The weight detectors in the shelf weigh the product, the cameras send the data to the Amazon Go system Step 3 - System decides which item is being taken
Alternative Flow	-
Exception Flow	System fails to decide which item is taken and an error thrown to the IT Management System.
Postconditions	Item is eventually added to the virtual cart.

Table 14: Analyze which item is taken by the customer

Use Case Name	Handle errors
Actors	IT Staff
Description	IT staff handles the errors to make Amazon Go function properly.
Data	The time and description of the error
Preconditions	Every Amazon Go store must have IT staff.
Stimulus	An error occurred in the system.
Basic Flow	Step 1 - An erroneous input corrupted the system or affected its decision mechanism. Step 2 - IT staff is informed about the error by a notification. Step 3 - IT staff handles the error by either finding and editing the source of the error or deciding instead of Amazon Go.
Alternative Flow	Step 2 - If the notification system doesnot function, IT staff is informed by other staff members or customers. Step 3 - IT staff handles the error.
Exception Flow	If Amazon Go IT members are not able to solve the problem, the system corrupts for some time.
Postconditions	Amazon Go continues to function properly.

Table 15: Handle errors

Use Case Name	View system logs
Actors	IT staff
Description	IT staff can see the logs in detail. Information like how many customers are in the store at that moment or who entered when is readable by IT staff.
Data	System logs
Preconditions	The IT staff member must be authorized.
Stimulus	An IT staff member requests to read the logs.
Basic Flow	Step 1 - The request gets to the database with an annotation of the time interval asked. Step 2 - Logs are listed chronologically and written to an output file.
Alternative Flow	Step 1 - If logs are asked to have some spesifications, IT staff makes the request accordingly. Step 2 - Logs are listed as wished and written to the desired file.
Exception Flow	If there is an authorization or connection problem, IT personnel is notified.
Postconditions	System logs are visible to the IT staff member.

Table 16: View system logs

4.2 Composition View

In this viewpoint, how the system's individual parts are combined together is the main issue. A top-level approach is used without going into details of the components. Component Diagram and Deployment Diagram are given with their design rationale.

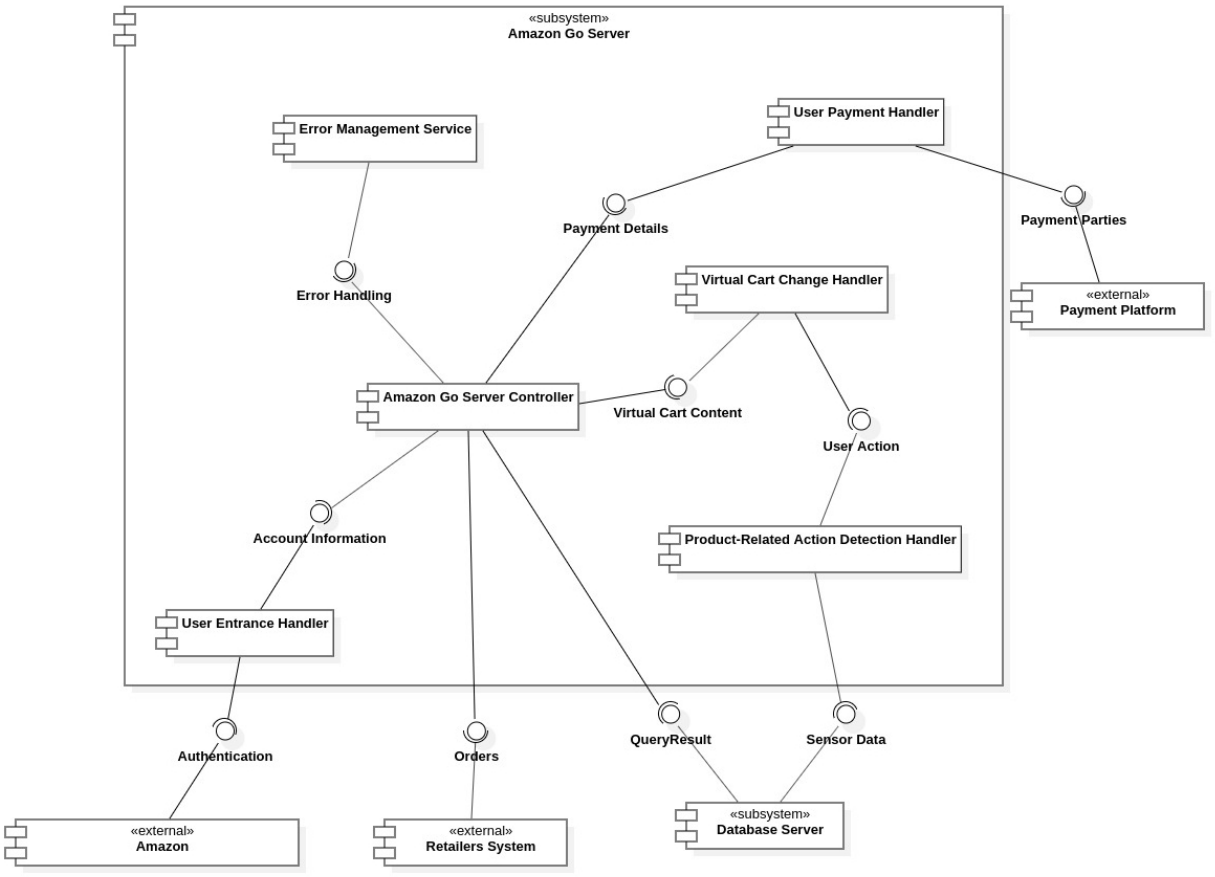

Figure 3: Amazon Go Component Diagram

Design Rationale:

- Amazon Go Server is a subsystem with the main component Amazon Go Server Controller, which organizes the traffic.
- Payment Platform is an external system which can be Visa, Skrill or many more . User Payment Handler component provides the Payment Parties information and directs the payment request to the related money acquiring platform. User Payment Handler shares the Payment details with the Controller.
- Amazon Go Server Controller gets QueryResults from Database Server.
- Amazon User Management System is an external system which the User Entrance Handler takes the user information from.
- User Entrance Handler provides the account details of the new customer to the Amazon Go Server Controller.
- Error Management Service aims to keep system going by Error Handling. The bugs are handled here.

- Retailers System is an external system. Amazon Go Server Controller processes the stock information retrieved from Database Server and creates orders. Orders are directed to Retailers System.
- Product-Related Action Detection Handler gets the Sensor Data directly from the Database Server in order to be fast. This way Virtual Cart Change Handler can function real-time.

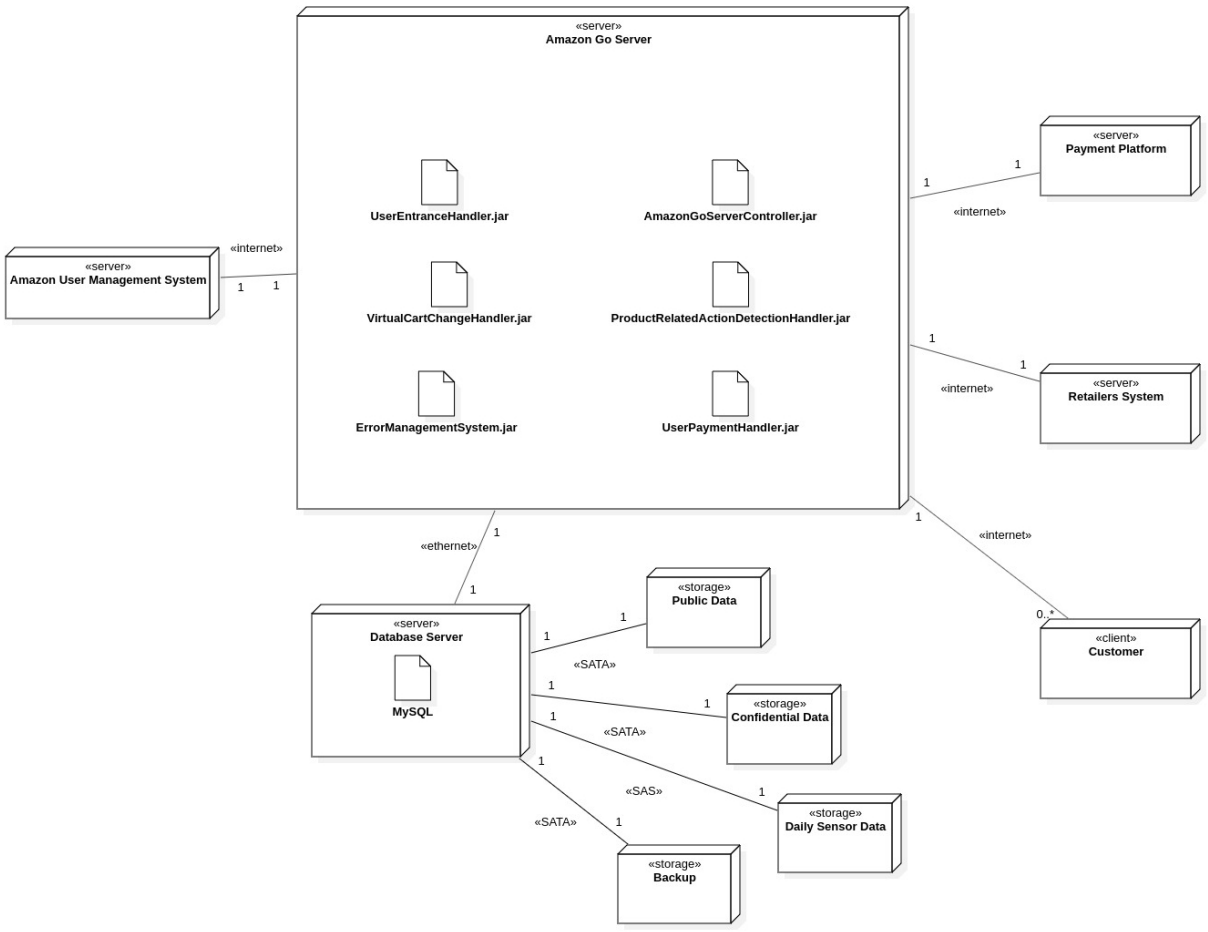

Figure 4: Amazon Go Deployment Diagram

Design Rationale:

- In the Database Server,the data can be used for marketing purposes or for the stock management is called as public data and the data about users or staff that should not be shared with any other parties is called as confidential data. These datum are hold in server in different drives. SATA drives are preferred, though they are relatively slow compared to SDDs they are less prone to errors. Sensor Data should be accessible fast and as it is deleted daily, errors are not a significant issue therefore Daily Sensor Data provided by Store Sensors is held in a SAS drive.
- There is backup drive in Database Server, which backups the Public and Confidential data.
- As DBMS, MySQL is chosen. The motive for this selection is MySQL being open source and easy to use for IT Staff.

4.3 Information View

In this view, the organization of the database and relations between data kept in database is given via class diagrams. Their descriptions, design of the operations kept in database and create, read, update and delete functionalities assigned to each class are also given.

4.3.1 Interfaces

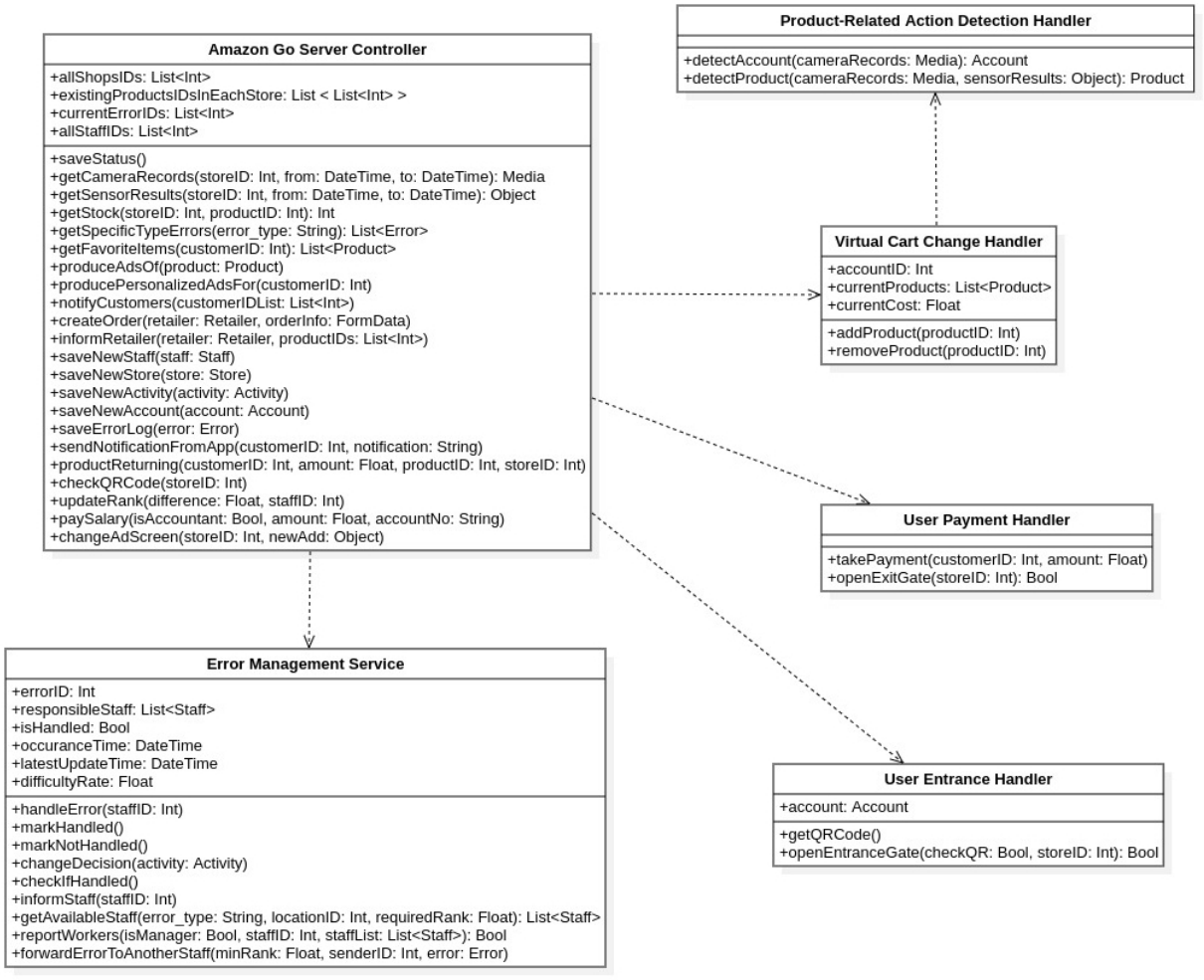

Figure 5: Amazon Go Interfaces Class Diagram

Operation	Description
saveStatus	Migrates all the changes to database.
getCameraRecords	Returns camera recordings of given store between given time interval.
getStock	Returns the stock of a product in a store.
getSpecificTypeErrors	Returns all existing errors with the same type.
getFavoriteItems	Returns favorite items of given customer.
produceAdsOf	Informs advertisement department of Amazon Go to make them produce advertisement of given product.
producePersonalizedAdsFor	analyzes data of given customer to detect potential shopping products in order to prepare advertisements of them.
notifyCustomers	Sends notification to the customers given in the list. Notification is prepared in this operation.
createOrder	Creates a new order to get products from retailers.
informRetailer	Informs the given retailer that the system has made a new order or changes information about a currently-retailing order.
saveNewStaff	Saves new staff member, analyzes his/her previous experiences etc. in order to detect his/her ranking.
saveNewStore	Called when a new store is opened. Saves the store information to database with necessary additional operations, when needed.
saveNewActivity	Saves a new activity to database.
saveNewAccount	Checks whether the account is suitable to create. If so, adds the just-created account to database.
saveErrorLog	Saves the error log. Informs IT staff members, when needed.
sendNotificationFromApp	Sends notification to all customers having Amazon Go Mobile App.
productReturning	Checks whether the product is damaged etc. and if the product is sturdy and acceptable, then accepts the product as returned, pays its money back to the customer and updates stocks.
checkQRCode	Scans QR Code from Mobile App with scanners on gates and returns whether the code exists and acceptable or not.
updateRank	Changes the rank of given staff with given new rank. Called when an error handled or abandoned, without success. New rank is calculated by the system and not changeable by hand.
paySalary	Pays salary of given customer.
changeAdScreen	Manages advertisement screens in the given store and selects a new one from given product's advertisements.
reportWorkers	If an IT Manager suspects that a decision on an error of a staff member may be wrong then a group of staff approved by manager is informed about the error and the staff member to inspect his/her decisons.

forwardErrorToAnotherStaff	When the staff member who is responsible from given error cannot solve the problem, (s)he does this operation in order to transfer the responsibility to his/her colleague.
handleError	Refers to solving the given error.
markHandled	Marks the given error as "handled".
markNotHandled	Marks the given error as "not handled". When this operation is done by a staff member, some other operations, such as updateRank is automatically operated.
changeDecision	Used when the hardware decision system of the store makes an unfair decision, or could not decide at all. The staff member is able to intervene the decision by changing the activity record.
checkIfHandled	Checks whether an error is handled or not.
informStaff	Informs given staff member about the error and makes that member responsible from the error.
getAvailableStaff	Finds all the staff members whose occupation fits and meets the needed minimum rank.
getQRCode	Produces QR Code from the app.
openEntranceGate	Opens the very first gate of given store in order to let the user come in to the store.
addProduct	Adds given product to related account's cart.
removeProduct	Removes given product from related account's cart.
takePayment	Withdraws given amount of money from given user's account.
openExitGate	Opens the latest gate of given store in order to let the user go out of the store.
detectAccount	Detects who it adding a product to his/her cart vie cameras.
detectProduct	Detects what is the product taken or left via cameras and sensors..

Table 17: Interfaces Operation Descriptions

Design Rationale

- Amazon Go Server Controller is the main controller that synchronises database changes and responding to other internal interactions on Mobile App, IT Staff and Store interfaces. Errors, staff members, stores and enrolled accounts are kept in this class for convenience.
- Here, notifyCustomers operation is personalized, such as sending advertisal notifications to targeted accounts. On the other hand, sendNotificationFromApp operation is used to inform *all* the customers about an update, any changes about concept, a new feature and so on.
- IT Staff Members could simply do some operations, such as forwarding an error which (s)he could not handle to another staff member, via Error Management Service class.
- Virtual Cart Change Handler is the class which handles the physical operations happening in the store related with shopping, such as picking an item from a shelf.
- User Payment Handler class handles the physical operations during a customer's entrance. Therefore, it works as a bridge from physical world to the system.

- All the payment operations just before PayPal, Visa etc. are handled in the User Payment Handler class. In other words, this class provides the connection between Amazon Go system and external payment systems.
- All the errors which needs to be considered are controlled and most of the time handled in Error Management Service class. Here, only the "changeDecision" operation could be intervened by humans. Other operations are automatically done and cannot change any attribute.

Operation	**Inputs**	**Outputs**	**Exceptions**
saveStatus	-	-	Database synchronization failed.
getCameraRecords	-storeID -from -to	Media (Video)	There is no saved recording or cameras are broken.
getStock	-storeID -productID	Int	Given store or product does not exist.
getSpecificTypeErrors	-error_type	List<Error>	There exists no such type for an error.
getFavoriteItems	-customerID	List<Product>	There is no recorded customer with the given ID.
produceAdsOf	-product	-	System is not able to produce advertisement of given product.
producePersonalizedAdsFor	-customerID	-	There is not enough data about the given customer or there is no such recorded customer in database.
notifyCustomers	-customerIDs	-	There exists a problem with one (or more) customer(s).
createOrder	-retailer -orderInfo	-	Database connection failed.
informRetailer	-retailer -productIDs	True if operation is successful, false otherwise.	Database connection failed.
saveNewStaff	-staff	-	Database synchronisation failed.
saveNewStore	-store	-	Database synchronisation failed.
saveNewActivity	-activity	-	Database synchronisation failed.
saveNewAccount	-account	-	Database synchronisation failed.
saveErrorLog	-error	-	Database synchronisation failed.
sendNotificationFromApp	-customerID -notification	-	Connection failed.
productReturning	-customerID -amount -productID -storeID	True if product is acceptable, false otherwise.	There is no such object(s) exist: customer, amount, product, store.

checkQRCode	-storeID	True if QR Code is suitable, false otherwise.	Connection failed.
updateRank	-difference -staffID	New rank of given staff.	Rank exceeded boundary.
paySalary	-isAccountant -amount -accountNo	Returns true if operation is successful, false otherwise.	Connection failed. Payment did not occur.
changeAdScreen	-storeID -newAdd	Media (Image of new ad billboard)	There is no advertisement generated for given product.
reportWorkers	-isManager -staffID -staffList	True if error is fixed, false otherwise.	Problem occurred: Worker(s) are not reported..
forwardErrorToAnotherStaff	-minRank -senderID -error	-	At least one of the object(s) does not exist: sender, error.
handleError	-staffID	-	Error is not handled!
markHandled	-	-	Database connection failed.
markNotHandled	-	-	Database connection failed.
changeDecision	-activity	New decision string	Existing decision is same with the new one.
checkIfHandled	-	True if given error is marked as handled, false otherwise.	Database connection failed: Cannot gather information.
informStaff	-staffID	-	There is no such staff member with given ID.
getAvailableStaff	-error_type -locationID -requiredRank	List<Staff>	No such staff member exist.
getQRCode	-	QR Code	Account is not created yet!
openEntranceGate	-checkQR -storeID	True if gate is opened, false otherwise.	Cannot read QR Code. Try again.
addProduct	-productID	Product	Could not determine product.
removeProduct	-productID	Product	Could not determine product.
takePayment	-customerID -amount	-	Payment is not taken. See your bank.
openExitGate	-storeID	True if gate is opened, false otherwise.	A hardware problem happened.
detectAccount	-cameraRecords	Account	Account cannot be recognized.
detectProduct	-cameraRecords -sensorResults	Product	Undetected product.

Table 18: Interfaces Operation Design

4.3.2 Database Operations

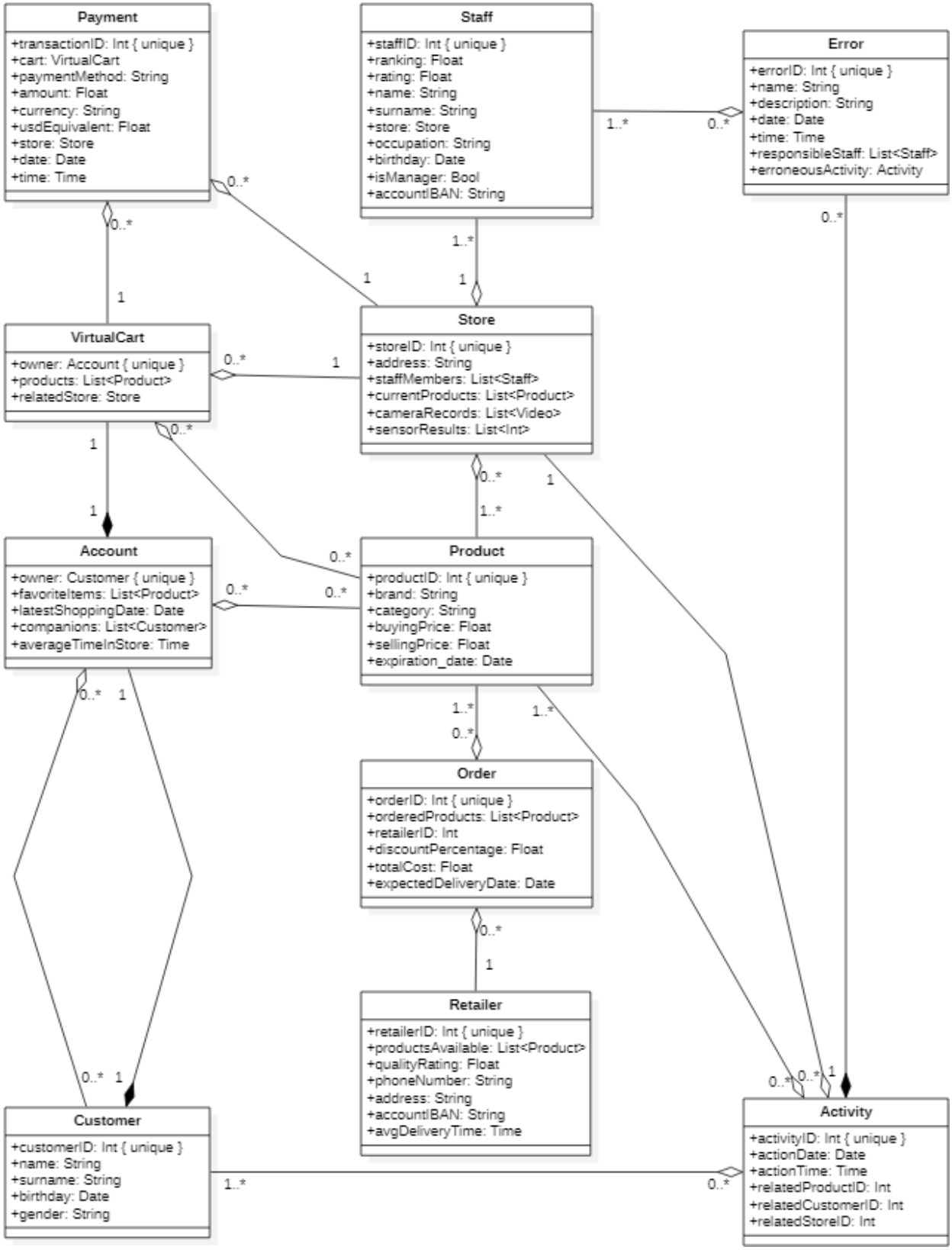

Figure 6: Amazon Go Database Class Diagram

Operation	Description
saveStatus	Create: Payment, VirtualCart, Account, Staff, Store, Product, Order, Customer, Retailer, Error, Activity Read: - Update: Payment, VirtualCart, Account, Staff, Store, Product, Order, Customer, Retailer, Error, Activity Delete: Payment, VirtualCart, Account, Staff, Store, Product, Order, Customer, Retailer, Error, Activity
getCameraRecords	Create: - Read: Store Update: - Delete: -
getStock	Create: - Read: Store, Product, VirtualCart Update: - Delete: -
getSpecificTypeErrors	Create: - Read: Error Update: - Delete: -
getFavoriteItems	Create: - Read: Account, Customer Update: - Delete: -
produceAdsOf	Create: - Read: Product Update: - Delete: -
producePersonalizedAdsFor	Create: - Read: Account, Customer Update: - Delete: -
notifyCustomers	Create: - Read: Customer, Account, Product Update: - Delete: -
createOrder	Create: Order Read: Retailer, Product, Store Update: - Delete: -
informRetailer	Create: - Read: Retailer, Order Update: - Delete: -
saveNewStaff	Create: Staff Read: - Update: - Delete: -
saveNewStore	Create: Store Read: - Update: - Delete: -

saveNewActivity	Create: Activity Read: - Update: - Delete: -
saveNewAccount	Create: Account Read: - Update: - Delete: -
saveErrorLog	Create: Error Read: - Update: - Delete: -
sendNotificationFromApp	Create: - Read: Account Update: - Delete: -
productReturning	Create: Activity Read: Account, Payment Update: Store, Activity, Payment Delete: -
checkQRCode	Create: - Read: Account Update: Store Delete: -
updateRank	Create: - Read: - Update: Staff Delete: -
paySalary	Create: - Read: Staff Update: STaff Delete: -
changeAdScreen	Create: - Read: Store, Product, Account, Customer Update: Store Delete: -
reportWorkers	Create: - Read: Error, Staff Update: Error Delete: -
forwardErrorToAnotherStaff	Create: - Read: Error Update: Staff, Error, Activity Delete: -
handleError	Create: - Read: Error, Staff, Activity Update: Error Delete: -
markHandled	Create: - Read: - Update: Error, Activity, IT Staff Delete: -
markNotHandled	Create: -

	Read: - Update: Error, Staff Delete: -
changeDecision	Create: - Read: - Update: Activity, VirtualCart Delete: VirtualCart
checkIfHandled	Create: - Read: Error Update: - Delete: -
informStaff	Create: - Read: - Update: Error, Staff Delete: -
getAvailableStaff	Create: - Read: Staff, Activity, Error Update: - Delete: -
getQRCode	Create: - Read: - Update: Account Delete: -
openEntranceGate	Create: - Read: - Update: Store Delete: -
addProduct	Create: Activity Read: Store Update: VirtualCart, Store Delete: -
removeProduct	Create: Activity Read: Store Update: VirtualCart, Store Delete: VirtualCart
takePayment	Create: Payment Read: Account, VirtualCart Update: Account Delete: VirtualCart
openExitGate	Create: - Read: - Update: Store Delete: -
detectAccount	Create: - Read: Store Update: - Delete: -
detectProduct	Create: - Read: Store Update: - Delete: -

Table 19: CRUD Operations

Design Rationale

- MySQL is chosen as the Database Management System due to ensure the satisfaction of integrity and object oriented design requests.
- Customer and Account are designed as different classes in order to differenciate the owner of the account and companions. Therefore, a better precise marketing strategy could be applied both digitally via Mobile App and physically inside the store, using digital billboards.
- Most of the time, if one object is aggregated to another object, the former object holds the latter one's primary key but not itself, due to save time and space.
- Error is a weak entity, shown as composition relation in the database class diagram, because an error could not occur individually if no action has happened.
- Since a VirtualCart is created when a user is doing shopping in store, VirtualCart's existance is dependent to Account. Therefore, it is considered as weak entity against Account. On the other hand, it has aggregation between its other relations with Product, Store and Payment, because a VirtualCart can individually exist even if a Store is closed, a Product has run out or a Payment hasn't yet occurred.
- Since the system needs all different Retailer objects in order to decide which one is the best option for an order, a Retailer object is kept in database separately. Moreover, because of the same reason, i.e. the system keeps a Retailer's information even if there has not been any previous Order with that Retailer object, Retailer has an aggregation with Order.

4.4 Interface View

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

4.4.1 Internal Interfaces

The Interface between Database Server and Amazon Go Server Controller

Whenever an operation related with the database occurs in the system, the Amazon Go Server Controller will query the database with a string in SQL. Then, the database server will execute it in MySQL DBMS. If the string is executable in MySQL, then the database will send the result query to Amazon Go Server Controller. Otherwise, the error message is sent to Amazon Go Server Controller.

Design Rationale

- Since marketing has a big role in the system and data is essential for marketing, database storage is an individual system. Therefore, an interface is needed between the controller of the system and the database.
- Database is not reachable from interfaces different than the system controller except Product-Related Action Detection Handler, therefore only Amazon Go Server Controller will compose queries.

The Interface Between Database Server and Product-Related Action Detection Handler

For Product-Related Action Detection Handler to decide if a product is taken or put back, Sensor Data is required. The data kept in the Database Server is sent to Action Detection Handler with this interface. Product id and its location relative to customers are taken account and the type of action (put back to shelf or taken from a shelf) is determined via this interface.

Design Rationale

- The reason Product-Related Action Detection Handler does not access the Database Server over Amazon Go Server Controller is that Action Detection Handler must function fast (it should make about 50 decisions in a second).

The Interface Between Virtual Cart Change Handler and Product-Related Action Detection Handler

Every time an item is taken from or put back to a shelf, Product-Related Action Detection Handler detects the move and the user. Matching product-customer couple information is sent to the Virtual Cart Change Handler. The customer's virtual cart status and store's stocks change.

Design Rationale

- This interface must operate fast in order to get real-time results.
- To increase its speed, the products which are only close to the customer and their companions(if any) should be considered in Product-Related Action Detection Handler.

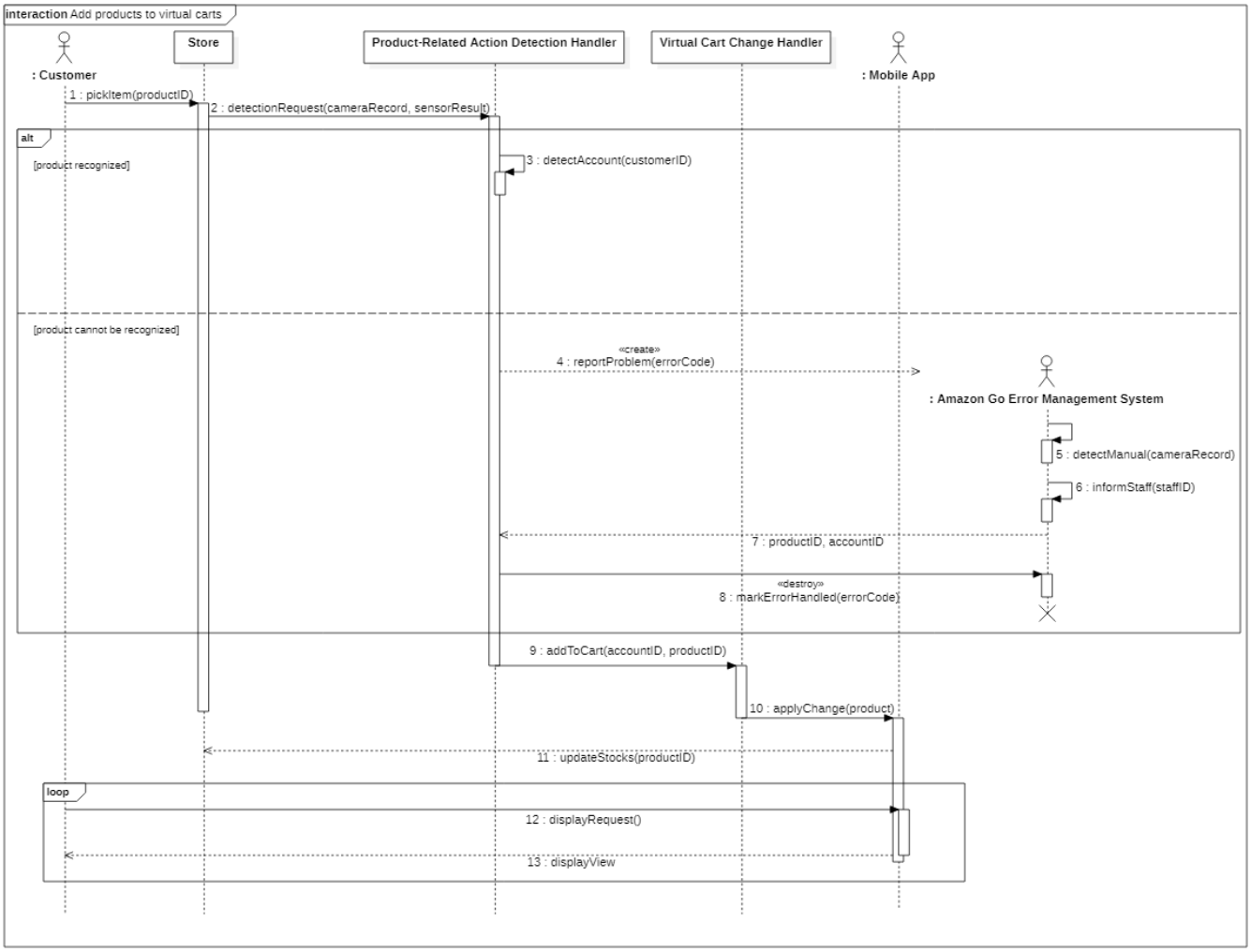

Figure 7: *Add products to virtual carts* sequence diagram showing the interface between Virtual Cart Change Handler and Product-Related Action Detection Handler components of the system

The Interface Between Virtual Cart Change Handler and Amazon Go Server Controller

All the actions about *customers' shopping* happening in the store are handled by Virtual Cart Change Handler and related data manipulation and extraction is handled by Amazon Go Server Controller. Amazon Go Server Controller polls cameras and takes interrupt from sensors above shelves, and when the controller detects an action, it triggers Virtual Cart Change Handler with a cookie. At last, this action is reflected to user's virtual cart via Virtual Cart Change Handler.

Design Rationale

- In order to increase security, Virtual Cart Change Handler is not able to reach directly to database.

The Interface Between User Payment Handler and Amazon Go Server Controller

The customer's account details as well as the total amount of the products in their virtual carts at the time of exit are collected from the related components with Amazon Go Server Controller. This interface is the bridge between User Payment Handler and Amazon Go Server Controller. Through this interface the payment details are sent to Payment Handler.

Design Rationale

- This interface's initiator is the customer's request to exit the store.
- If an error occurs in the Payment Handler, Amazon Go Server Controller will be immediately notified.

The Interface Between User Entrance Handler and Amazon Go Server Controller

As a customer tries to enter an Amazon Go store, a request will be made in the User Entrance Handler. User Entrance Handler will deal with this request and result will be transmitted to Amazon Go Server Controller with this interface. In case of the customer is accompanied by some others, the necessary information about them is also dispatched through this interface.

Design Rationale

- Since whether the customer is alone or with others is critical for some system functions, this interface always sends the notice of how many people entered store with the same account.

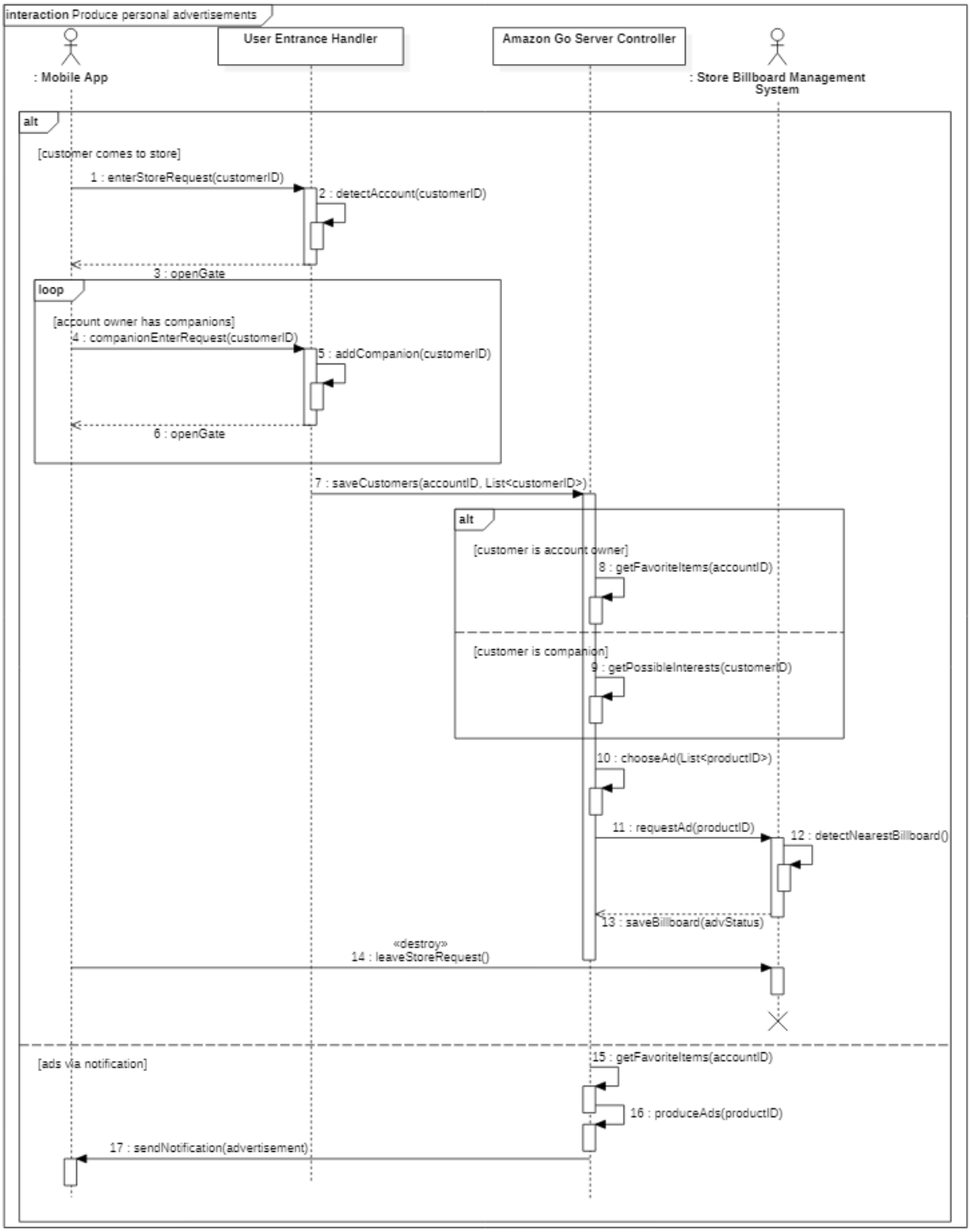

Figure 8: *Produce personal advertisements* sequence diagram showing the interfaces between Amazon Go Mobile App, Amazon Go Server Controller, User Entrance Handler and Store Billboard Management System

The Interface Between Error Management Service and Amazon Go Server Controller

Amazon Go Server Controller collects all the errors and misjudgments from other components. These flaws are sent to Error Management System through this interface.

Design Rationale

- In which component the error occurred is also provided to Error Management System.

4.4.2 External Interfaces

4.4.2.1 User Interfaces

User interfaces of the system are mainly all the interfaces that the users relevant with the system use to communicate with Amazon Go. These include all but not just Mobile App interfaces that let customers do their shopping in stores. Interfaces that allow retailers to get informed and response and that make IT Staff interfere with the system and other staff members are also included and explained with more detail in this section.

IT Staff Interface

IT Staff will be able to check system functions through this interface. The errors on the Error Management System, the decisions that are have to be made in case Amazon Go algorithms cannot decide who took an item and the malfunctions are checked and fixed. This interface includes a communication panel where the staff can communicate as well as a surveillance panel.

Design Rationale

- To access this interface a staffID and password is required.
- The sensor system and the decisions of the algorithms can be observed real-time by Staff via this interface. A customer or a product can be on focus where a general surveillance is also possible.

QR Code Interface

QR Code is the main element of authentication. As soon as a user scans the QR Code in their mobile app, the authentication procedure will start. QR Code interface produces QR Codes specific to user. QR Code information will be valuable whole shopping time, as it will identify the customers.

Design Rationale

- If a screenshot is taken of the QR Code, the interface notifies the user with "For your security screenshots will not work for entry.".
- In case the customer is accompanied by others, the QR Code will be same and it will be scanned separately for each user.

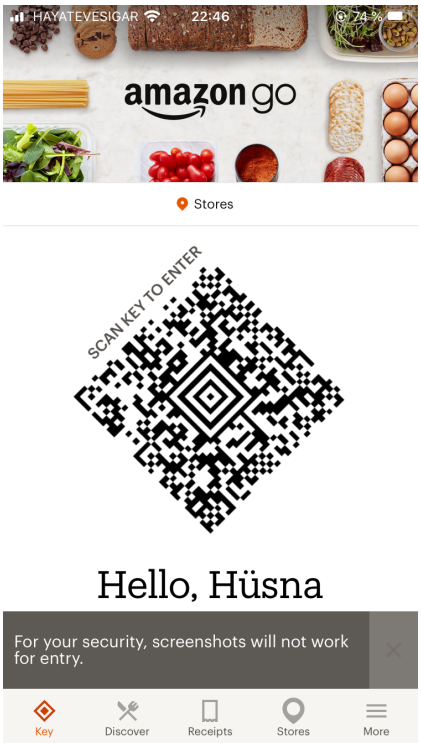

Figure 9: QR Code Interface

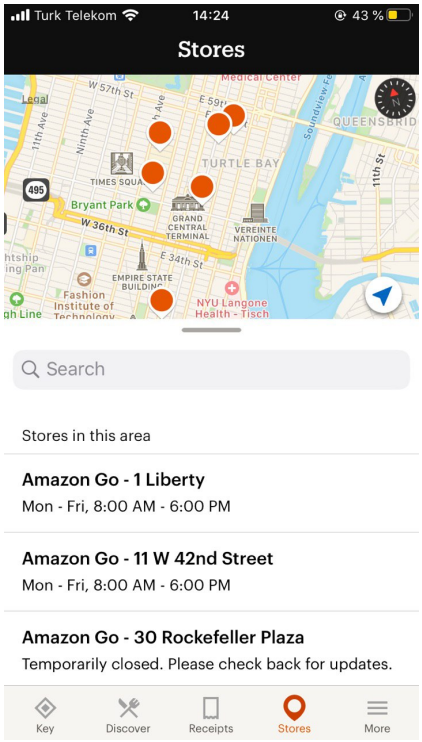

Figure 10: Stores-Map Interface

Stores-Map Interface

Currently Amazon Go has 26 stores in USA and 1 Amazon Go Grocery store in Seattle. A Google Maps interface is used to locate all this stores. Accessible through Mobile App, users can see which store is the closest to them, see the stores' working hours and may get directions.

Design Rationale

- Amazon Go Mobile app uses different map services based on the platform. In iOS Apple's Map service is used, and for Android Google Maps provide the map.

Payment Method Selection Interface

Users can choose any eligible account to pay for their future shopping. This interface lets users see the list of their saved-account details, add new accounts, delete accounts and choose the active account.

Design Rationale

- All the users of Amazon Go must have at least one eligible account.

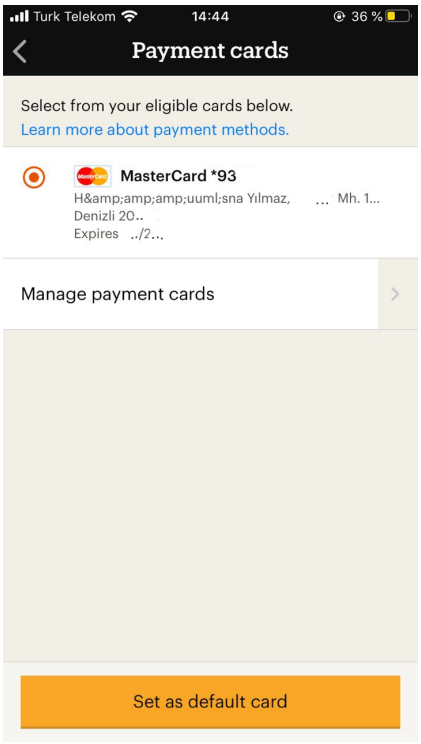

Figure 11: Payment Method Selection Interface

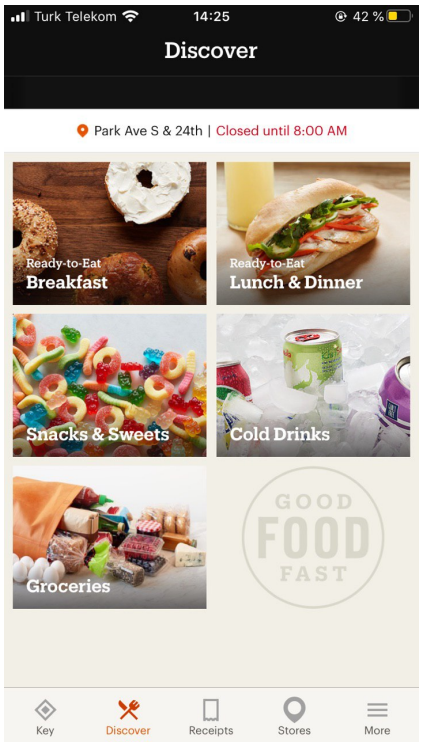

Figure 12: Discover Interface

Discover Interface

Amazon Go offer different products at each store. To ease the users' choice of store, products are listed in this interface. Users may check the stores nearby if they have the desired product.

Design Rationale

- To browse products, one must select a store first.
- Products are categorized according to their types. Breakfast, Lunch, Drinks etc.

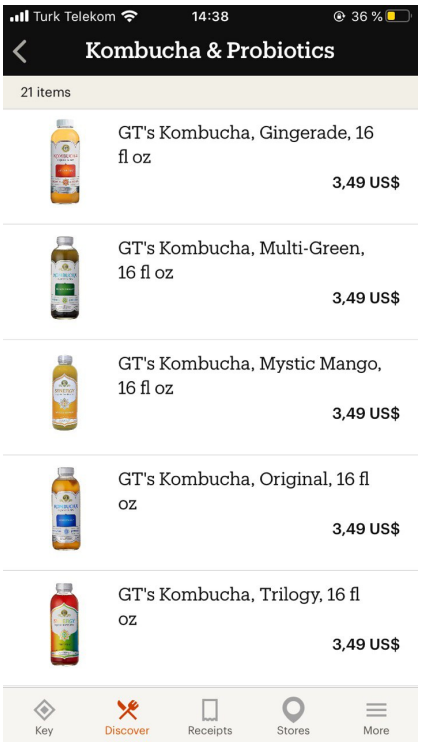

Figure 13: Discover Interface - Product List

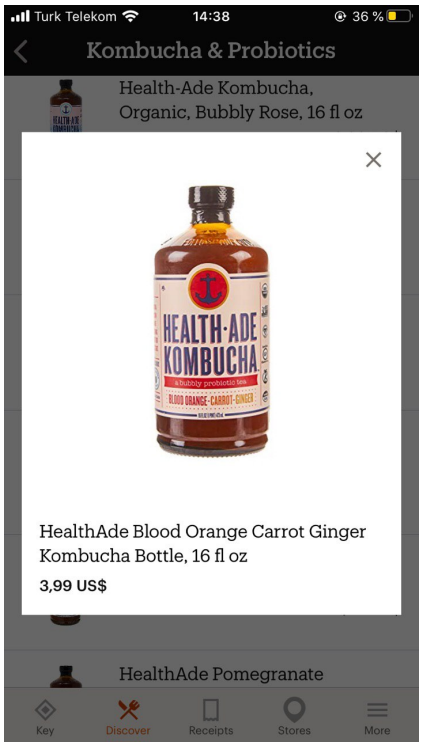

Figure 14: Discover Interface - Product Details

Receipts Interface

Past shopping details can be seen through this interface. The prices of products, tax payed and the total amount is shown. The receipts are sorted by date-time.

Design Rationale

- The customers are also informed about their trip time.
- Information about refunds is accessible through this interface.

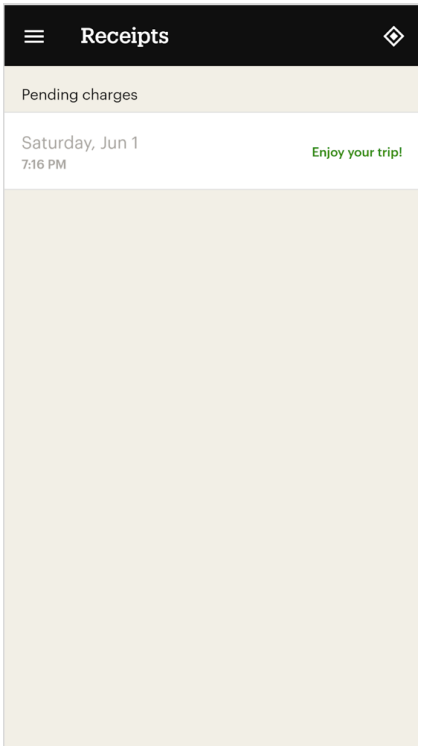

Figure 15: Receipts Interface - History

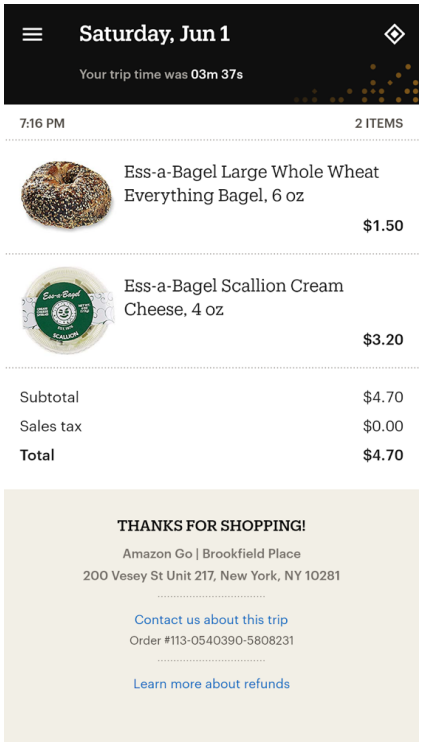

Figure 16: Receipts Interface - Detail

Orders Interface

When necessary the stocks must be filled in by placing orders. In Amazon Go ordering is an automated system. Responsible staff can see the current order details in the Orders Interface. Amount, order date, productID, retailerID are the information listed.

Design Rationale

- If order can not be given due to a retailer issue, an error is thrown.
- Staff can see past orders, filter them, sort them according to the desired property and search among them. This makes it easier for staff to inspect the orders and make predictions when required.

4.4.2.2 System and Service Interfaces

The Interface Between Amazon Go Server Controller and Retailers System

Retailers supply products when needed. For inventory management, retailers must be informed of stocks. This interface foresee the date for the stocks to run out and give the orders beforehand. Prediction is made using the statistics. Amazon Go Server Controller retrieves the statistics from Database Server, whereas calculations are made by this interface.

Design Rationale

- It is this interface that matches the products with its retailer.
- To avoid stock-outs, orders are given two days in advance.

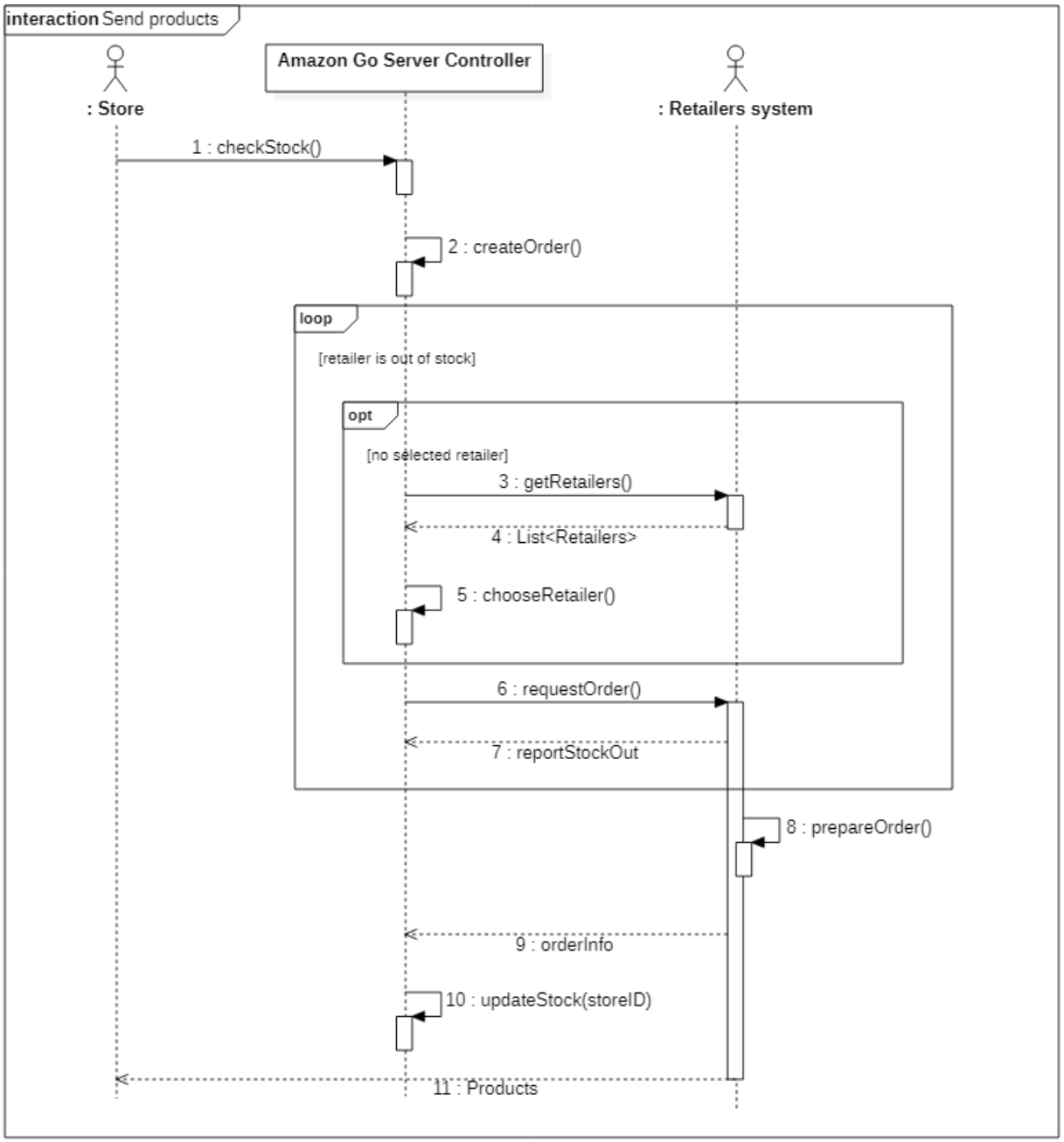

Figure 17: *Send products* sequence diagram showing the interfaces between Amazon Go Stores, Amazon Go Server Controller and the external Retailers System

The Interface Between User Payment Handler and Payment Platform

Amazon Go users give their bank or another payment service information, while signing in. This interface takes the payment information including the amount and the payment service (bank, PayPal, Skrill etc.) the user makes use of. With this information it gets in touch with the right Payment Platform, by delivering the account information,the amount and discounts, coupons if exist.

Design Rationale

- Payment currency is the choice of the customer, exchange will be handled by Payment Platform.
- If payment is not successful, an error message will be thrown, Amazon Go Server Controller will transmit the error to Error Management System will handle the error.

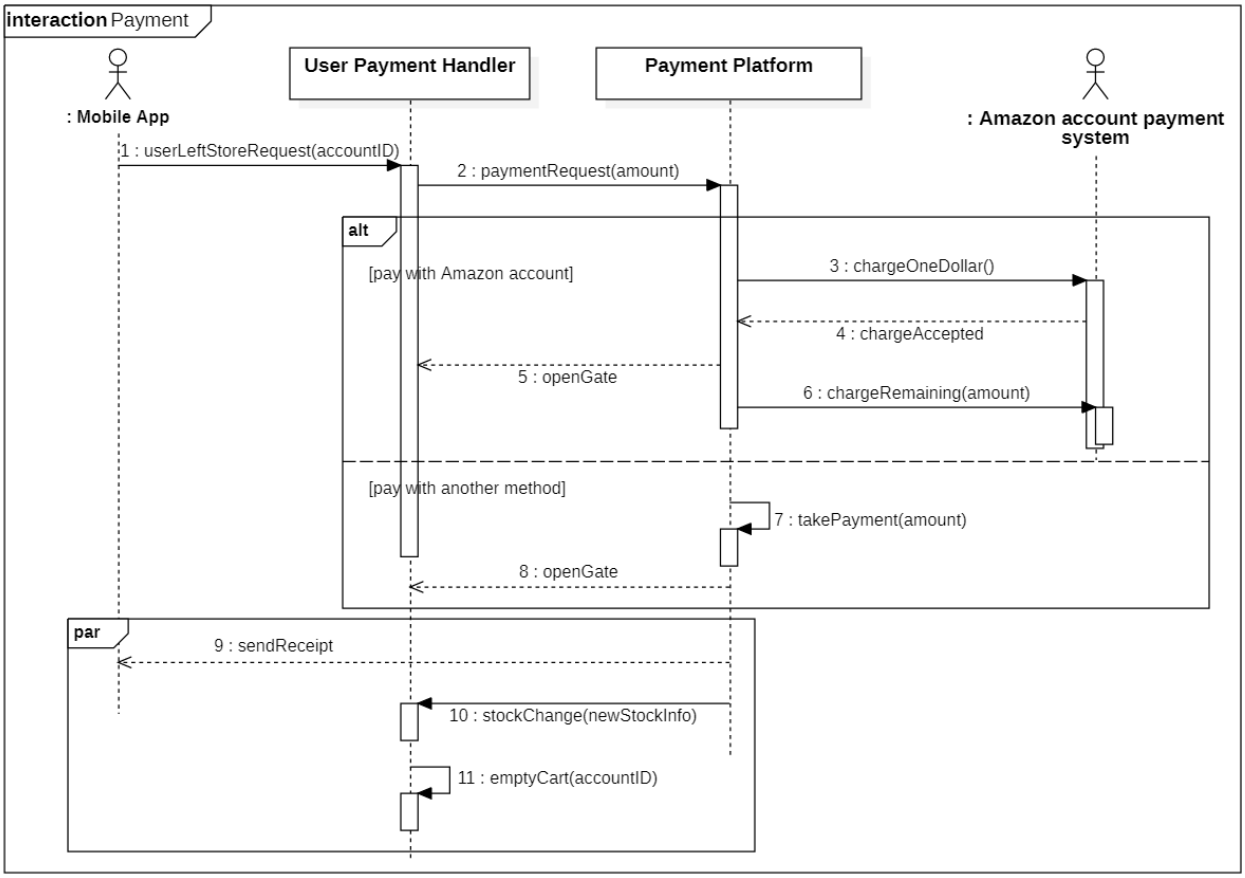

Figure 18: *Payment* sequence diagram showing the interfaces between Amazon Go Mobile App, User Payment Handler and external interfaces Payment Platform and Amazon Account Payment System

The Interface Between User Entrance Handler and Amazon User Management Service

A customer-to-be trying to enter an Amazon Go store by scanning the QR Code makes a request to User Entrance Handler. User Entrance Handler decodes the QR-Code, resulting accountID is sent through this interface to AUMS. If it is OK for user to come in the User Entrance Handler is informed and gate opens. "Entrance" use case requires this interface.

Design Rationale

- If AUMS, sends a negative response, or the interface encounters an error, Amazon Go Server Controller is informed through its connection with User Entrance Handler.

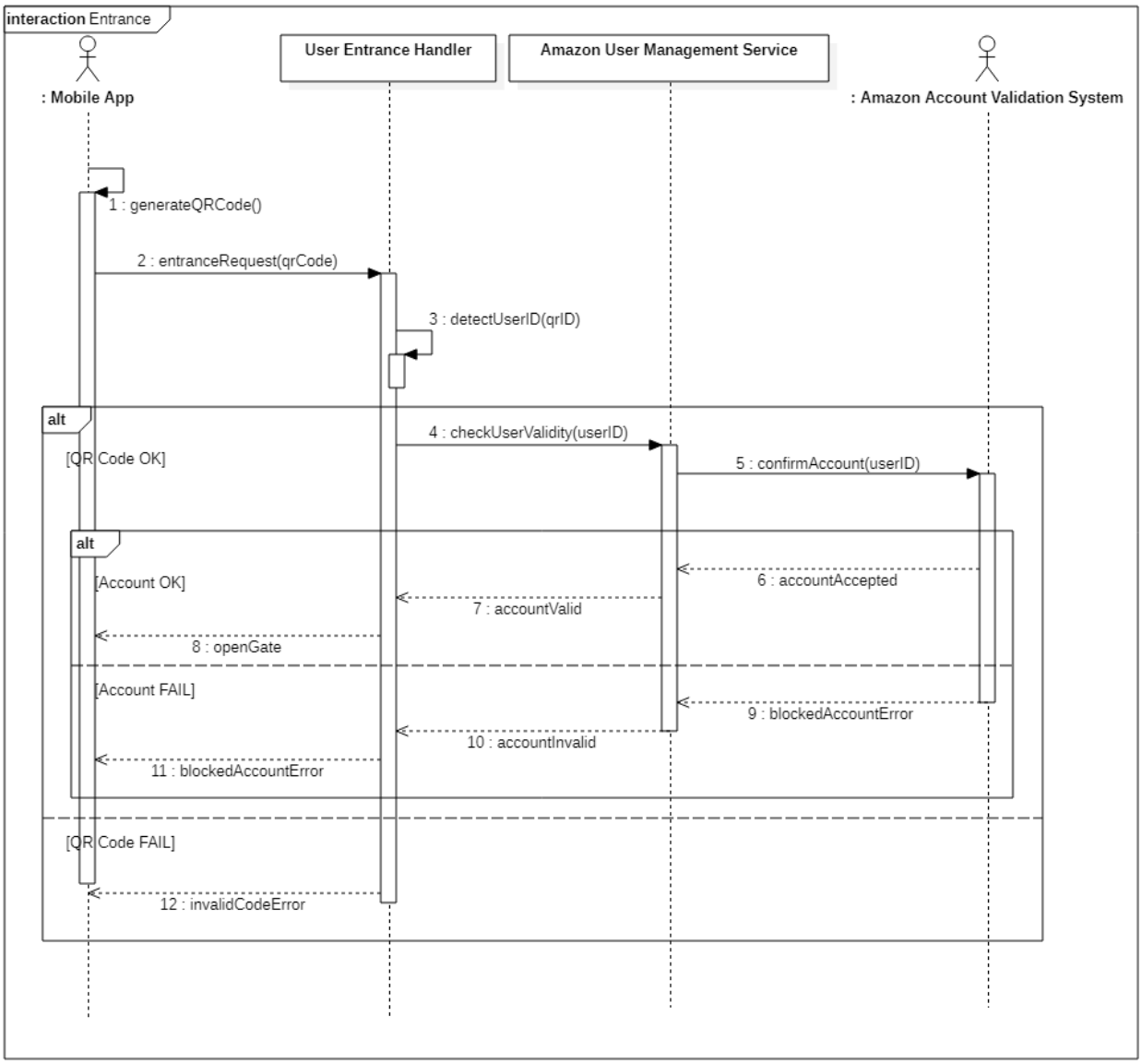

Figure 19: *Entrance* sequence diagram showing the interfaces between Amazon Go Mobile App, User Entrance Handler and external interfaces Amazon User Management Service and Amazon Account Validation System