

Amazon Go

Software Requirements Specification

Mustafa Ozan ALPAY - 2309615

Birkan ARSLAN - 2110252

Spring 2020

Revision History

- 1 March 3rd, 2020 Initial draft
- 2 April 8th, 2020 Final version

Table of Contents

1	Introduction	1
1.1	Purpose	1
1.2	Scope	1
1.3	System Overview	2
1.3.1	System Perspective	2
1.3.1.1	System Interfaces	3
1.3.1.2	User Interfaces	3
1.3.1.3	Hardware Interfaces	9
1.3.1.4	Software Interfaces	10
1.3.1.5	Communication Interfaces	10
1.3.1.6	Memory Constraints	10
1.3.1.7	Operations	10
1.3.2	System Functions	12
1.3.3	User Characteristics	13
1.3.4	Limitations	14
1.4	Definitions	16
2	References	18
3	Specific Requirements	19
3.1	External Interfaces	19
3.1.1	User Interface	20
3.1.2	Store Interface	20
3.1.3	Support Ticket Interface	20
3.1.4	Site Settings Interface	21
3.2	Functions	23
3.3	Usability Requirements	39
3.4	Performance Requirements	39
3.5	Logical Database Requirements	40
3.6	Design Constraints	41
3.7	Software System Attributes	43
3.7.1	Reliability	43
3.7.2	Availability	43
3.7.3	Security	43
3.7.4	Maintainability	44
3.7.5	Portability	44

List of Figures

1	Context Diagram	2
2	Sign-in screen of the application	4
3	Payment Information Screen	4
4	Steps of the Educational Animation, Part 1	5
5	Steps of the Educational Animation, Part 2	6
6	Sample QR Code to enter the store	7
7	In-store Employee Inventory Interface	7
8	System Admin Interface	8
9	Support Staff Interface	9
10	External Interfaces Class Diagram	19
11	Class Diagram of the System	22
12	Use Case Diagram	24
13	Entering the Store Sequence Diagram	28
14	Buying a Product Sequence Diagram	30
15	The Entity-Relationship Model for the Database of the System	42

List of Tables

1	System Functions	13
2	Definitions	17
3	Use case for Sign-in	23
4	Use case for Sign-up	25
5	Use case for Entering Payment and Billing Information	26
6	Use case for Entering the Store	27
7	Use case for Buying a Product	29
8	Use case for Displaying Previous Purchases	31
9	Use case for Displaying Invoices	32
10	Use case for Returning a Product	34
11	Use case for Opening a Support Ticket	35
12	Use case for Authenticating Employees	36
13	Use case for Viewing Inventory Details	37
14	Use case for Viewing Support Tickets	38

1 Introduction

This document provides the Software Requirement Specification (SRS) for the smart store which is called as “Amazon Go”.

1.1 Purpose

The purpose of this project is to reduce the time spent during the shopping process and reduce labour-based costs for a regular convenience store.

1.2 Scope

- The system will have a mobile app that allows users to login, generate QR codes to enter the store, update payment and billing information, and show previous purchases.
- The system will have a cloud based storage solution that uses Amazon Web Services, which contains the back-end development.
- The system will have a database server that is being used to store user profiles, payment and billing information, and purchases.
- The system will contain an image processing unit that processes data gathered from the in-store cameras to track users' movements inside the store, and properly compute the virtual shopping basket contents.
- The system will contain an embedded systems manager unit that gathers and process data from weight and proximity sensors at each shelf, cameras that scan barcodes and some special recognition tags, QR code scanners to allow customers enter the store, and RFID antennas to recognize items.
- The system will contain a customer behaviour tracker subsystem that scans for the customer's previous purchases to successfully determine who picked which item at moments where face and motion recognition systems fail.
- The system will have a subsystem for payments that directly deducts the amount from the credit card of the user.
- The system will contain an admin panel that allows the system administrators to change system settings and see reports and logs of the current systems to ensure a seamless store.
- The system will have a customer support panel that allows customers to contact to a support service that helps with refunds, returns, and questions.

- The system will have a store employee panel to allow the in-store employees track inventory and make necessary changes.

1.3 System Overview

1.3.1 System Perspective

Amazon Go is not a part of a larger system but it interacts with the Bank that Amazon has an agreement with, in order to facilitate the payment methodologies.

The system can be divided into subsystems, however it should be kept in mind that each of these systems are in constant communication with each other, in order to make the system work. For instance the embedded systems module and the image processing modules can be built as separate entities, however they need external data from the main system to actually work as intended.

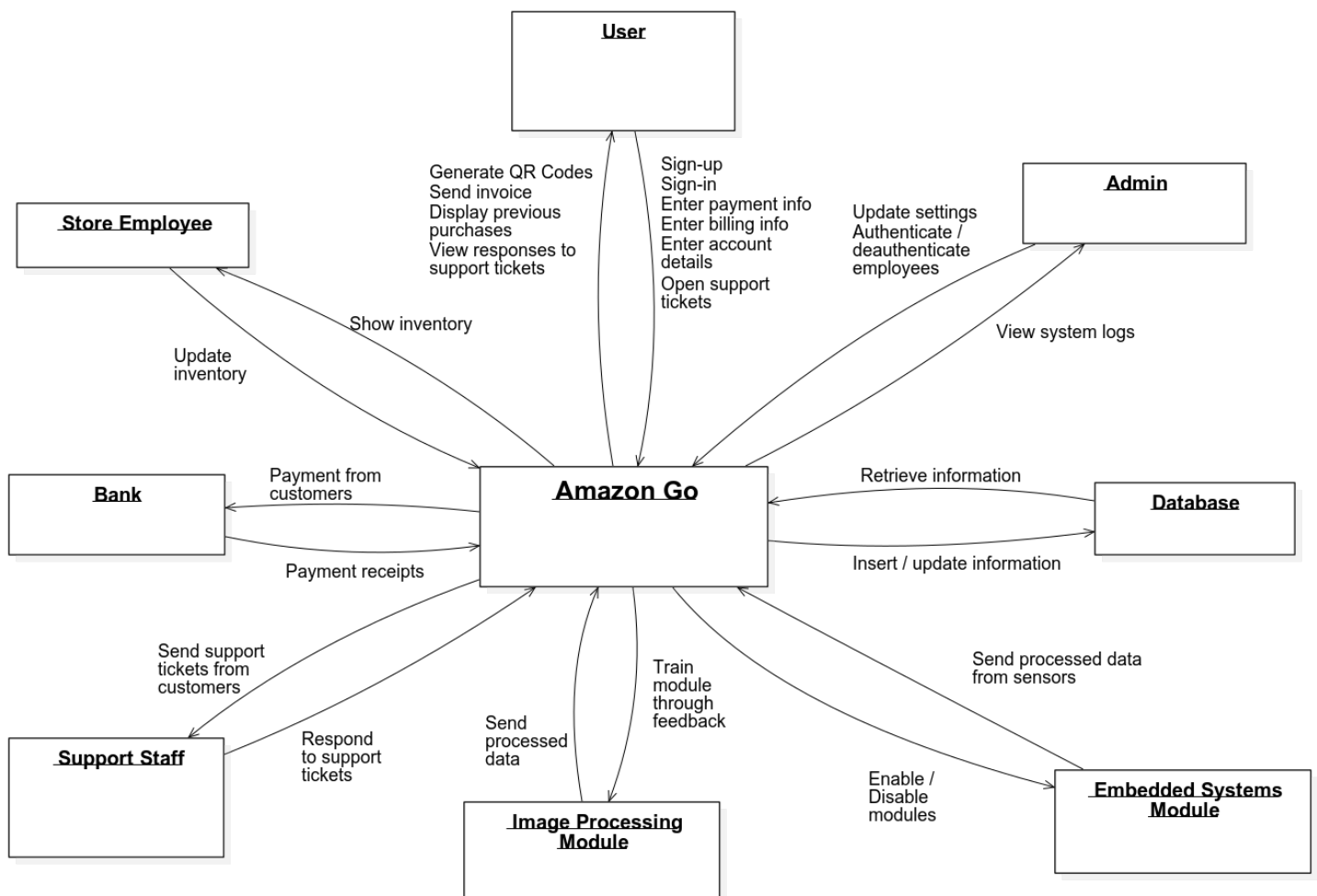


Figure 1: Context Diagram

1.3.1.1 System Interfaces

Within the Amazon Go system, the interfaces can be divided into two sub-parts, namely internal and external interfaces. The internal interfaces consist of the Image Processing, Embedded Systems, and Database Management interfaces, and the external interfaces consist of the User, Store, Support Ticket and Site Settings interfaces. More information regarding the external interfaces can be found at Section 3.1.

- **Image Processing Interface:** The image processing interface uses in-store cameras to track users and decide if any of the users take an item, or put an item back. It also uses facial recognition systems to avoid mistakes. The system uses user ID's to generate virtual data containers per user when they enter a store. Each of these containers have a unique ID, and the rest of the system uses this unique ID to communicate with this interface.
- **Embedded Systems Interface:** The embedded systems interface uses in-store weight and proximity sensors to detect if any item taken from the shelf and amount of the items left on the shelf. This information is stored inside the database that allows the in-store employees to track inventory as well. Any action taken at any shelf is transmitted to the image processing interface to determine the taker of that specific action, which allows the correct tracking of in-store movements.
- **Database Management Interface:** The database management interface is one of the main components of the system, since the system needs to store actions taken by users, and act accordingly. This includes but not limited to user information, purchases, support messages, bills, payments, user behaviour, et cetera. The components of the Database Management Interface allows any other interface that needs to retrieve or send data to the database use the specific endpoints to ensure stability.

1.3.1.2 User Interfaces

Within the Amazon Go system, there are 4 types of user interfaces, namely: Mobile Application Interface, In-Store Employee Interface, Support Staff Interface, and System Admin Interface. This section is intended to provide an overview, and more information will be given at Section 3.1.

Mobile Application Interface

The mobile application serves as the first endpoint that a customer connects to the system. It allows the users to register, login, update their billing and payment information, enter the store, and display their previous purchases. It is available on the Apple App Store and Google Play Store for free.

After installation, the application takes the user to the Sign-in screen, which allows previously registered users to sign-in, and new users to create a new account. This screen can be seen at Figure 2.

After the authentication process, the user is required to fill in their billing and payment information before entering a store. This is achieved through the Payment Information Panel, which can be seen at Figure 3.

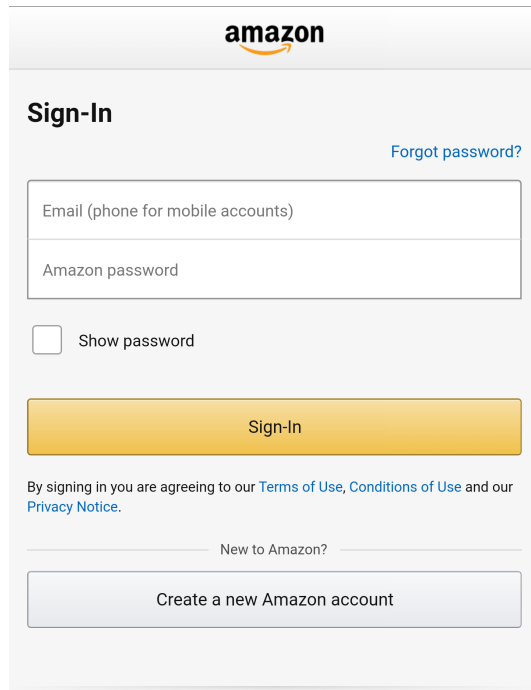
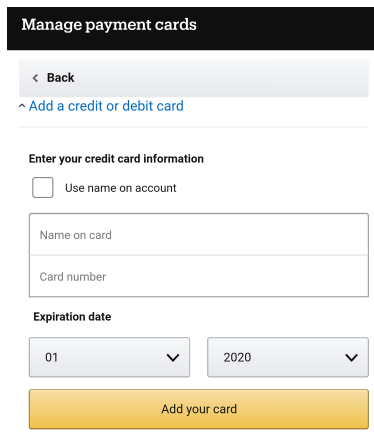
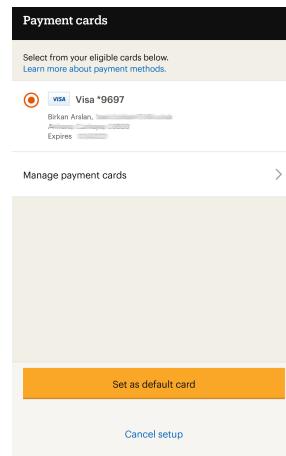


Figure 2: Sign-in screen of the application



(a) Entering payment information



(b) Viewing payment information

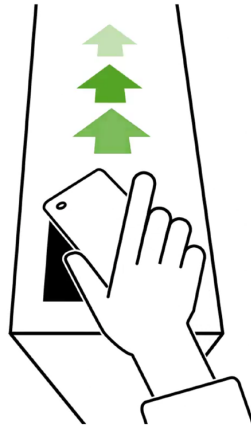
Figure 3: Payment Information Screen

After completing the payment and billing information, the main screen with a QR Code shows up. This is the default screen for the app, if the prerequisites are fulfilled. A sample QR Code can be seen at Figure 6.

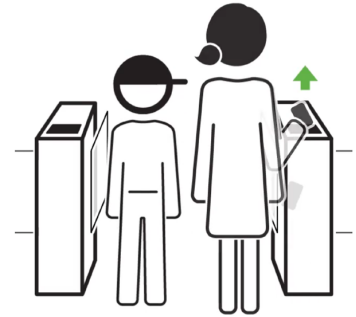
If the user is running the application for the first time, an educational animation will be displayed to guide the user through the process. Sample excerpts from the animation can be seen at Figures 4 and 5.



Use your app to enter the store by holding your phone's screen over the gate scanner to unlock.



Use your app to enter the store by holding your phone's screen over the gate scanner to unlock.



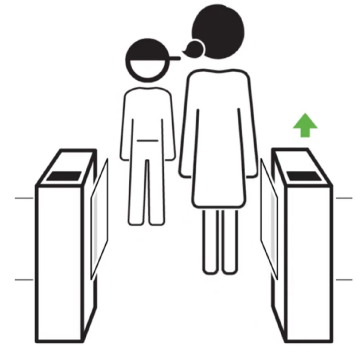
You can also use your app to let your friends and family in. Simply scan to let them in first.



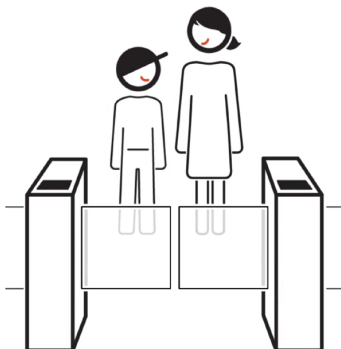
You can also use your app to let your friends and family in. Simply scan to let them in first.



You can also use your app to let your friends and family in. Simply scan to let them in first.



You can also use your app to let your friends and family in. Simply scan to let them in first.



You can also use your app to let your friends and family in. Simply scan to let them in first.



Anything you put back on the shelf comes out of your virtual cart.

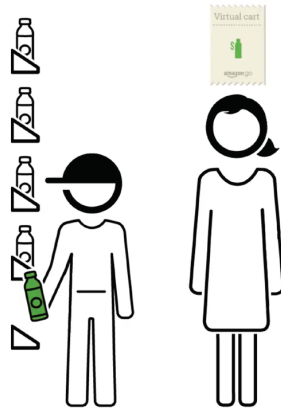


Anything you take off the shelf is automatically added to your virtual cart.

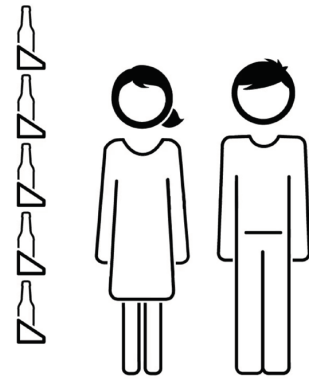
Figure 4: Steps of the Educational Animation, Part 1



Anything you put back on the shelf comes out of your virtual cart.



It works the same for items taken (or put back) by friends and family you scan in.



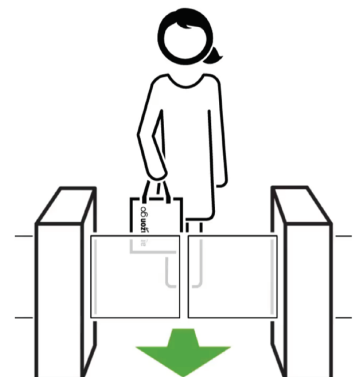
Since products you take go in *your* virtual cart, please don't take things for other shoppers.



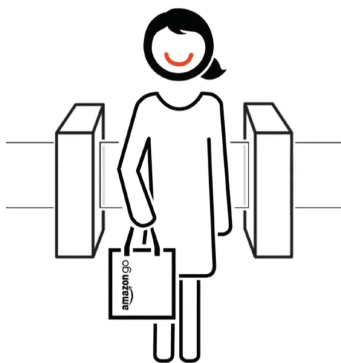
Since products you take go in *your* virtual cart, please don't take things for other shoppers.



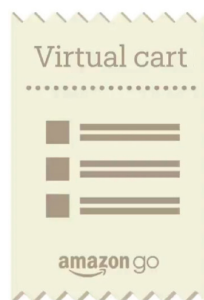
Since products you take go in *your* virtual cart, please don't take things for other shoppers.



When you're done, you're good to go. No lines, no checkout.



When you're done, you're good to go. No lines, no checkout.



Soon after, we'll notify you that your receipt is ready and charge your card.



Seriously, you can go!

Figure 5: Steps of the Educational Animation, Part 2



Hello, Birkan

Figure 6: Sample QR Code to enter the store

In-Store Employee Interface

This interface allows the in-store employees to track and update inventory details of the store that they are assigned to. It is designed as a simple panel that can be used either through a tablet or a computer and requires no technical training. A sample screen of listing current inventory can be seen at Figure 7.

Inventory Details			amazon go
			Madison Center, Seattle, WA
Display Current Inventory Add Items Remove Items			Logout
Barcode ▼	Product Name	Price	Quantity
481518156	Coca-Cola Regular 8 fl. oz	\$0.99	188
481518157	Coca-Cola Zero 8 fl. oz	\$0.99	164
481518158	Coca-Cola Cherry Coke 8 fl. oz	\$1.12	18
481518159	Coca-Cola Vanilla 8 fl. oz	\$1.12	45
481518160	Fanta 8 fl. oz	\$0.99	121
481516161	Sprite 8 fl. oz	\$0.99	142
518621892	Pepsi Regular 8 fl. oz	\$0.99	197
518621893	Pepsi Max 8 fl. oz	\$0.99	164
518621894	Mountain Dew 8 fl. oz	\$0.99	139
968150156	La Croix Pure 8 fl. oz	\$1.24	45
968150157	La Croix Berry 8 fl. oz	\$1.24	15
968150158	La Croix Cran-Raspberry 8 fl. oz	\$1.24	0
968150159	La Croix Lemon 8 fl. oz	\$1.24	42
968150160	La Croix Lime 8 fl. oz	\$1.24	34
968150161	La Croix Orange 8 fl. oz	\$1.24	12
968150162	La Croix Passion Fruit 8 fl. oz	\$1.24	8
968150163	La Croix Apricot 8 fl. oz	\$1.24	33
968150164	La Croix Mango 8 fl. oz	\$1.24	0

Figure 7: In-store Employee Inventory Interface

System Admin Interface

This interface allows the System Admins to maintain and make changes on the Amazon Go system. It has been designed to automate mundane tasks with a few simple mouse clicks. It allows the admins to authenticate and deauthenticate users, as well as displaying logs and system status, and act accordingly if needed. A simple authentication screen can be seen at Figure 8.

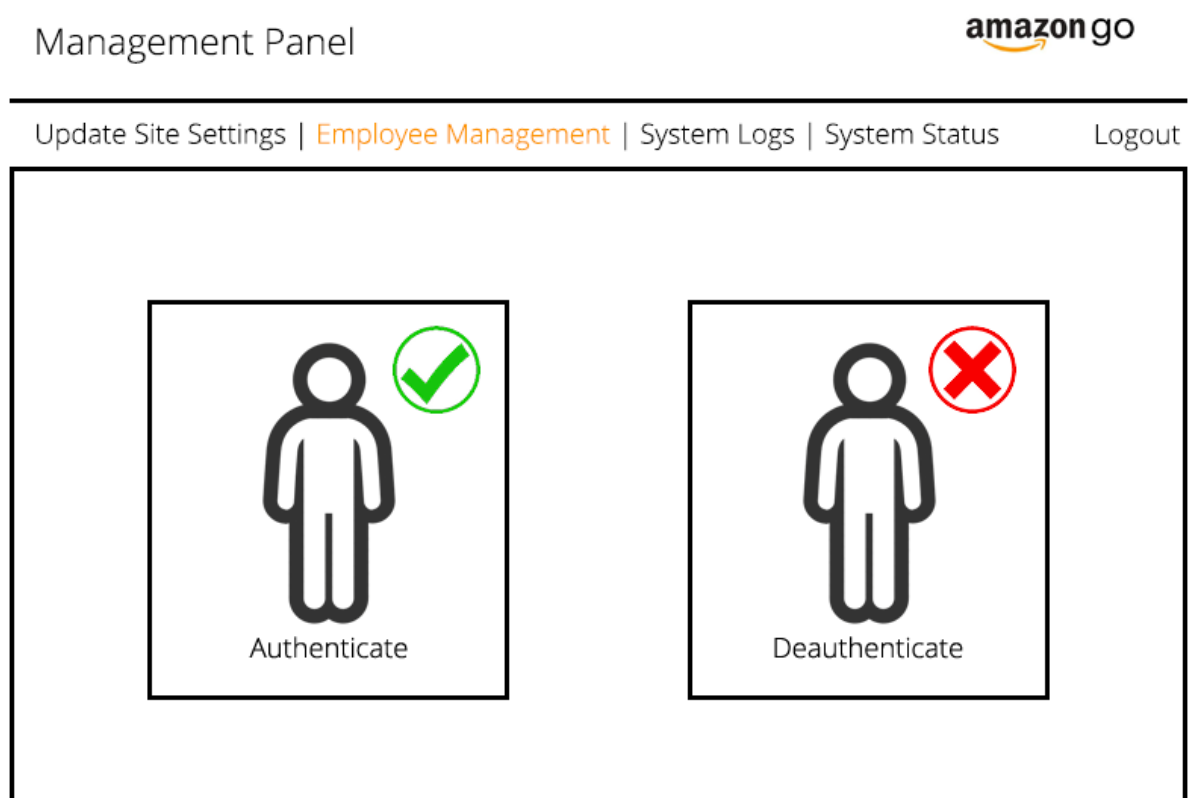
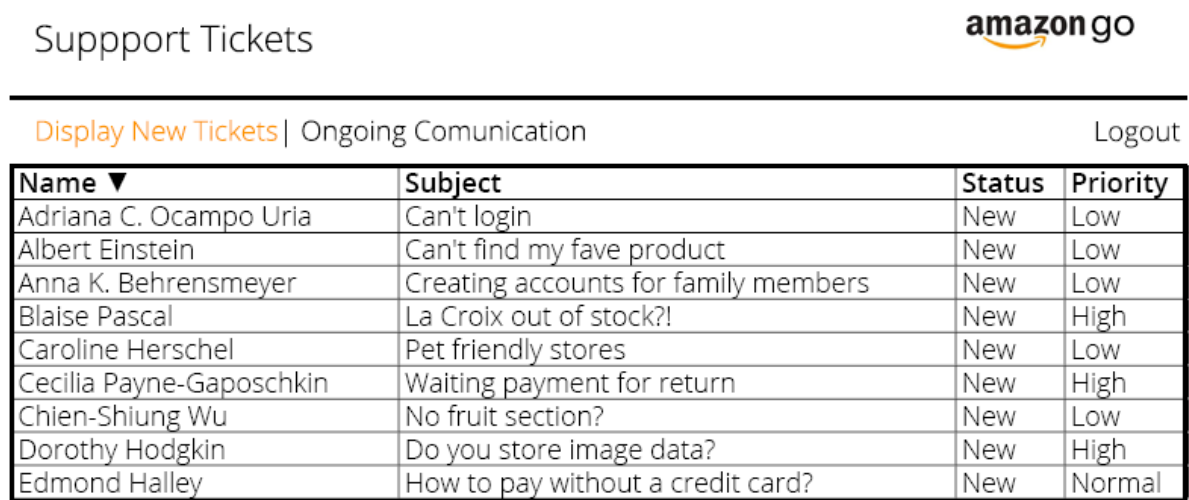


Figure 8: System Admin Interface

Support Staff Interface

This interface allows the support staff to track and respond to open support tickets. It is designed as a simple panel that can be used through a computer and requires no technical training. A sample of this interface to display support tickets can be seen at Figure 9.



Support Tickets		amazon go	
Display New Tickets Ongoing Communication			Logout
Name ▼	Subject	Status	Priority
Adriana C. Ocampo Uria	Can't login	New	Low
Albert Einstein	Can't find my fave product	New	Low
Anna K. Behrensmeyer	Creating accounts for family members	New	Low
Blaise Pascal	La Croix out of stock?!	New	High
Caroline Herschel	Pet friendly stores	New	Low
Cecilia Payne-Gaposchkin	Waiting payment for return	New	High
Chien-Shiung Wu	No fruit section?	New	Low
Dorothy Hodgkin	Do you store image data?	New	High
Edmond Halley	How to pay without a credit card?	New	Normal

Figure 9: Support Staff Interface

1.3.1.3 Hardware Interfaces

The hardware interface of the system should have the following parts:

- Storage server: Stores all necessary user information and acts as the mainframe. Uses Amazon Web Services (AWS) and automatically scales up to meet the demand.
- In-store data handling server: Uses all data gathered by the in-store cameras and sensors to determine the actions of the user. Constantly improves itself by means of machine learning. Uses Amazon Web Services (AWS) and automatically scales up to meet the demand.
- Mobile devices: Used as a medium to enter and interact with the store by the users. Also used by the store employees for the inventory actions.

1.3.1.4 Software Interfaces

- Java SE 13 is used to handle the back-end services of the system.
- PostgreSQL 12.2 is used as database solution.
- Python 3.8.1 is used to track user activities in the shop using TensorFlow 2.1.0.
- Swift 5.1.4 is used for the iOS mobile app development.
- Kotlin 1.3.61 is used for the Android mobile app development.

1.3.1.5 Communication Interfaces

Within the store, each sensor and camera has access to the in-store network. The continuous data flow from these sources gets processed and communicated to the mainframe to track the users virtual basket. This system uses the IEEE 802.3 standard.

The in-store data handling server and the mainframe communicates through the specific API's, using the internet connection. The system uses wired connection, by using IEEE 802.3 standard.

The users of the store use their mobile devices to enter the store. This requires an active mobile data connection. In this interface IEEE 802.1X protocol is used.

1.3.1.6 Memory Constraints

Since the system uses AWS as the main server solution, and since AWS is autoscaling itself as needed, the system does not have any memory constraints.

1.3.1.7 Operations

The operations of the Amazon Go system can be represented as:

- New Customer

1. Sign-up
2. Open Support Ticket

- Registered Customer

1. Sign-in
2. Open Support Ticket
3. Display Invoices
4. Display Previous Purchases
5. Enter Billing Information

6. Update Billing Information
7. Enter Payment Information
8. Update Payment Information
9. Generate QR Code
10. Enter Store
11. Buy a Product
12. Leave Store
13. Make Payment
14. Add Item to the Basket
15. Remove Item from the Basket
16. Return a Product

- **System Admin**

1. Update Site-wide Settings
2. Authenticate Employees
3. Deauthenticate Employees
4. View System Logs

- **Store Employee**

1. See Inventory Details
2. Update Inventory

- **Support Staff**

1. View Support Tickets
2. Respond to Support Tickets

1.3.2 System Functions

No	Functionality	Description
1	Sign-up	The user signs up to the system in order to use the system.
2	Sign-in	The user signs in to the system in order to use the system.
3	Open Support Ticket	The user opens a support ticket in order to communicate with the Support Staff.
4	Display Invoices	The user views their invoices.
5	Display Previous Purchases	The user displays their previous purchases through the Amazon Go mobile app.
6	Enter Billing Information	The user enters their billing information to the system.
7	Update Billing Information	The user updates their billing information on the system.
8	Enter Payment Information	The user enters their payment information to the system.
9	Update Payment Information	The user updates their payment information on the system.
10	Generate QR Code	The system generates a QR Code to let the user enter the store.
11	Enter Store	The user enters an Amazon Go store.
12	Buy a Product	The user purchases one or more products.
13	Leave Store	The user leaves an Amazon Go store.
14	Make Payment	The user makes a payment after their purchase at the Amazon Go store.
15	Add Item to the Basket	The user adds an item to their virtual basket.

16	Remove Item from the Basket	The user removes an item from their virtual basket.
17	Return a Product	The user returns a product to the store that they have purchased earlier.
18	Update Site-wide Settings	The System Admin updates site-wide settings.
19	Authenticate Employees	The System Admin authenticates a new employee to use the system.
20	Deauthenticate Employees	The System Admin deauthenticates a former employee from the system.
21	View System Logs	The System Admin views system logs.
22	See Inventory Details	The store employee views inventory details.
23	Update Inventory	The store employee updates inventory details.
24	View Support Tickets	The support staff views open support tickets.
25	Respond to Support Tickets	The support staff responds to open support tickets.

Table 1: System Functions

1.3.3 User Characteristics

Within the system, there are different groups of users.

1. Users

The users are the customers who do or do not have an account on the system. They should have a mobile device, a credit card, an e-mail address, and a data package on their mobile device.

2. System Admins

The system admins should have enough technical skills to ensure that the system is running smoothly. They should keep the backend of the system and its services up-to-date. They also have the authority to update site-wide settings,

as well as authenticating or deauthenticating employees. They must have a Computer Science or an equivalent degree.

3. Store Employees

The store employees do not need to have any hard technical skills. They are in charge of keeping the shelves filled, and placing the misplaced items to the related shelves. They only interact with the system using the inventory module on their mobile device that was given by the company.

4. Support Staff

The support staff is in charge of handling open support tickets that was generated by the users. They must have good written English skills, with enough technical knowledge to guide the user to the various solutions.

5. Researchers

The researchers are responsible of following the buying trends, user behaviour, peak hours, making user experience trials and using these data to maximize the profitability. They should have a Data Science related degree (Computer Science, Math, Statistics, etc.) or a Human Behaviour related degree (Psychology, Sociology, Human Resources, etc.).

1.3.4 Limitations

- a) **Regulatory policies:** Since the system stores what the users bought, any data of the users should be preserved according to the General Data Protection Regulation.
- b) **Hardware limitations:** The cameras and the sensors in the store should be able to follow each user without any problem. The servers that process the data retrieved from the sensors should be powerful enough to handle that.
- c) **Interfaces to other applications:** The system should be compatible with the payment system of the bank in order to allow payments.
- d) **Parallel operation:** The system should allow multiple users use the system at the same time, thus it should allow multiple threads running at the same time, one for each customer.
- e) **Audit functions:** The system should generate accurate invoices in a timely manner, in order to allow the auditing work smoothly.

- f) **Control functions:** Only the system admins can change the site-wide settings, thus any user that falls outside of this category cannot access to that part of the system.
- g) **Higher-order language requirements:** The mobile app that the users use was written in Swift and Kotlin, thus they are required.
- h) **Signal handshake protocols:** In-store systems use the in-store network to communicate with each other using the HTTPS protocol. All the systems outside of the store communicate with each other using the HTTPS protocol as well.
- i) **Quality requirements:** Since the system deals with finances and user behaviour, it is crucial to have multiple backups taken regularly.
- j) **Criticality of the application:** In case of errors, the system admins can view the system logs and find out the problem and solve it. Therefore as long as the logging system works, error recovery is possible within the system.
- k) **Safety and security considerations:** The system admins are directly responsible for keeping the system safe and secure by taking necessary measures. Any unauthorized access should be blocked and logged immediately.
- l) **Physical/mental considerations:** Even though the Amazon system is highly accessible, the store has limitations in terms of physical disabilities. Persons with visual impairment might have trouble entering or navigating through the store.

1.4 Definitions

Term	Definition
New Customer	A user who has not signed up to the system yet.
Registered Customer	A user who has completed the sign-up process for the system.
System Admin	The administrators of the system who has the highest permission level on the system.
Store Employee	Employees who are in charge of managing the inventory and order of the store.
Support Staff	Employees who are in charge of responding to the support tickets that was generated by the users.
AWS	Amazon Web Services (AWS) is a subsidiary of Amazon that provides on-demand cloud computing platforms and APIs to individuals, companies, and governments, on a metered pay-as-you-go basis. The system allows auto-scaling to meet the dynamic demand of the customer.
HTTPS	Hypertext Transfer Protocol Secure (HTTPS) is an extension of the Hypertext Transfer Protocol (HTTP). It is used for secure communication over a computer network. In HTTPS, the communication protocol is encrypted using Transport Layer Security (TLS) or, formerly, its predecessor, Secure Sockets Layer (SSL).
QR Code	QR code (abbreviated from ‘Quick Response code’) is the trademark for a type of matrix barcode (or two-dimensional barcode) first designed in 1994 for the automotive industry in Japan. Currently, QR codes often contain data for a locator, identifier, or tracker that points to a website or application.
API	Application Programming Interface.
SHA512	SHA-2 (Secure Hash Algorithm 2) is a set of cryptographic hash functions designed by the United States National Security Agency (NSA) and first published in 2001.

Java	Java is a general-purpose programming language that is class-based, object-oriented, and designed to have as few implementation dependencies as possible.
Python	Python is an interpreted, high-level, general-purpose programming language.
Swift	Swift is a general-purpose, multi-paradigm, compiled programming language developed by Apple Inc.
Kotlin	Kotlin is a cross-platform, statically typed, general-purpose programming language with type inference.
PostgreSQL	PostgreSQL, also known as Postgres, is a free and open-source relational database management system (RDBMS) emphasizing extensibility and technical standards compliance. It is designed to handle a range of workloads, from single machines to data warehouses or Web services with many concurrent users.
TensorFlow	TensorFlow is a free and open-source software library for dataflow and differentiable programming across a range of tasks. It is a symbolic math library, and is also used for machine learning applications such as neural networks.

Table 2: Definitions

2 References

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

29148-2011 - ISO/IEC/IEEE International Standard - Systems and software engineering – Life cycle processes –Requirements engineering. (2013, January 24). Retrieved March 15, 2016, from <https://standards.ieee.org/findstds/standard/29148-2011.html>

Other Sources:

Amazon Go Store. (n.d.). Retrieved March 1, 2020, from <https://www.amazon.com/b/?node=20931384011>

Sommerville, I. (2018). *Software Engineering*. Hallbergmoos/Germany: Pearson.

3 Specific Requirements

3.1 External Interfaces

The External Interfaces Class Diagram can be seen at Figure 10. To provide a better understanding of the whole system, the full class diagram of the system is also available at Figure 11. To ease readability, most of the generic getter-setter methods and all internal classes have been removed from the class diagram while creating the External Interfaces Class Diagram.

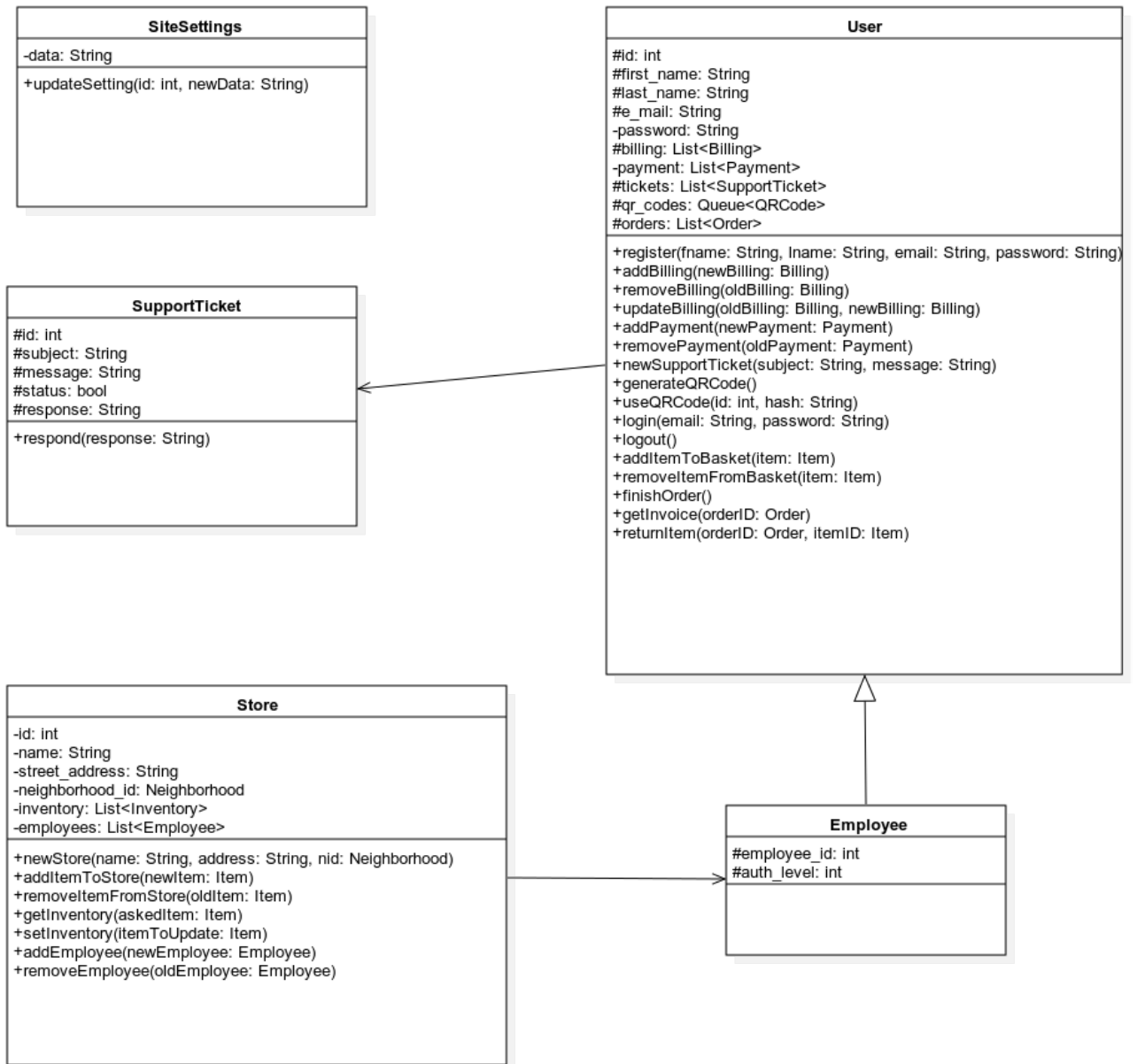


Figure 10: External Interfaces Class Diagram

3.1.1 User Interface

1. If the user tries to register using invalid information, the system shall warn the user and give examples of valid information.
2. In the case of receiving multiple requests from the same IP address, the system shall determine if this is an attack or not. If it is categorized as an attack, the system first force CAPTCHA's to the IP address. If requests persist, the system shall ban the IP address for 24 hours from creating new accounts.
3. If the user tries to enter invalid billing or payment information, the system shall warn the user and give examples of valid information.
4. The system shall allow the user to generate a QR Code to enter the store when the mobile application is opened.
5. If the user tries to login using invalid information, the system shall warn the user and after 5 unsuccessful logins, the system will force CAPTCHA's to the IP address. If requests persist, the system shall ban the IP address for 24 hours from logging-in for 5 minutes.
6. When the user leaves the store, a payment transaction shall be triggered, and an automated bill shall be generated. The user shall get an e-mail regarding this transaction.

3.1.2 Store Interface

1. When a new store gets opened, the system admins shall enter its information to the system to allow storing related information to the aforementioned store.
2. If the in-store employee changes the inventory of the store, the system shall update the database accordingly.
3. If a new employee gets assigned to the store, the system shall update the database and allow the employee to access the inventory panel.
4. If an existing employee gets dismissed from the store, the system shall update the database accordingly, and revoke any existing access permission to the inventory panel.

3.1.3 Support Ticket Interface

1. If the user enters the Support Tickets section, the system shall ask the user to type in their problem, and by using the existing problems and solutions

database, recommend possible solutions. If the user clicks on the “My question is not listed above” button, then the system shall allow for generating a support ticket.

2. When a new support ticket gets generated, the system shall assign it to a support staff with the lowest number of assigned open tickets.
3. If the user gets a response from the support staff and does not respond, the system shall warn the user to either respond or mark the ticket as closed. If the user does not close a ticket after 3 days, the system shall automatically mark it as closed.
4. When a support ticket gets marked as “closed”, the system shall ask the user to rate the support process with a scale of 1 to 5 (where 1 being the lowest score).
5. If a support staff gets a rating lower than 3, the system shall do an analysis to determine the cause of the problem and notify researchers.
6. If a support staff uses a word from the restricted vocabulary, this shall be logged and an automated system shall determine the context of the usage. In the case of verbal abuse, the system shall warn the support staff, and notify the superior officer if this behavior persists.
7. In the case of receiving multiple support requests from the same IP address, the system first force CAPTCHA’s to the IP address. If requests persist, the system shall ban the IP address for 24 hours from creating new support tickets.

3.1.4 Site Settings Interface

1. The system shall allow only system administrators and employees with proper authentication to reach this interface. In the case of any other person tries to reach it, the system shall give a “404 Not Found” error to distract naïve users.
2. When a site-wide setting gets changed, the system shall update the database accordingly, and if the update does not affect the work of the system, updates shall go live immediately. If it might affect the work of the system, the store shall wait for the maintenance hours to go live.
3. In the case of a setting change causing a problem within the system, any support staff can revert the system to the most recent trusted version of the system with a single click.

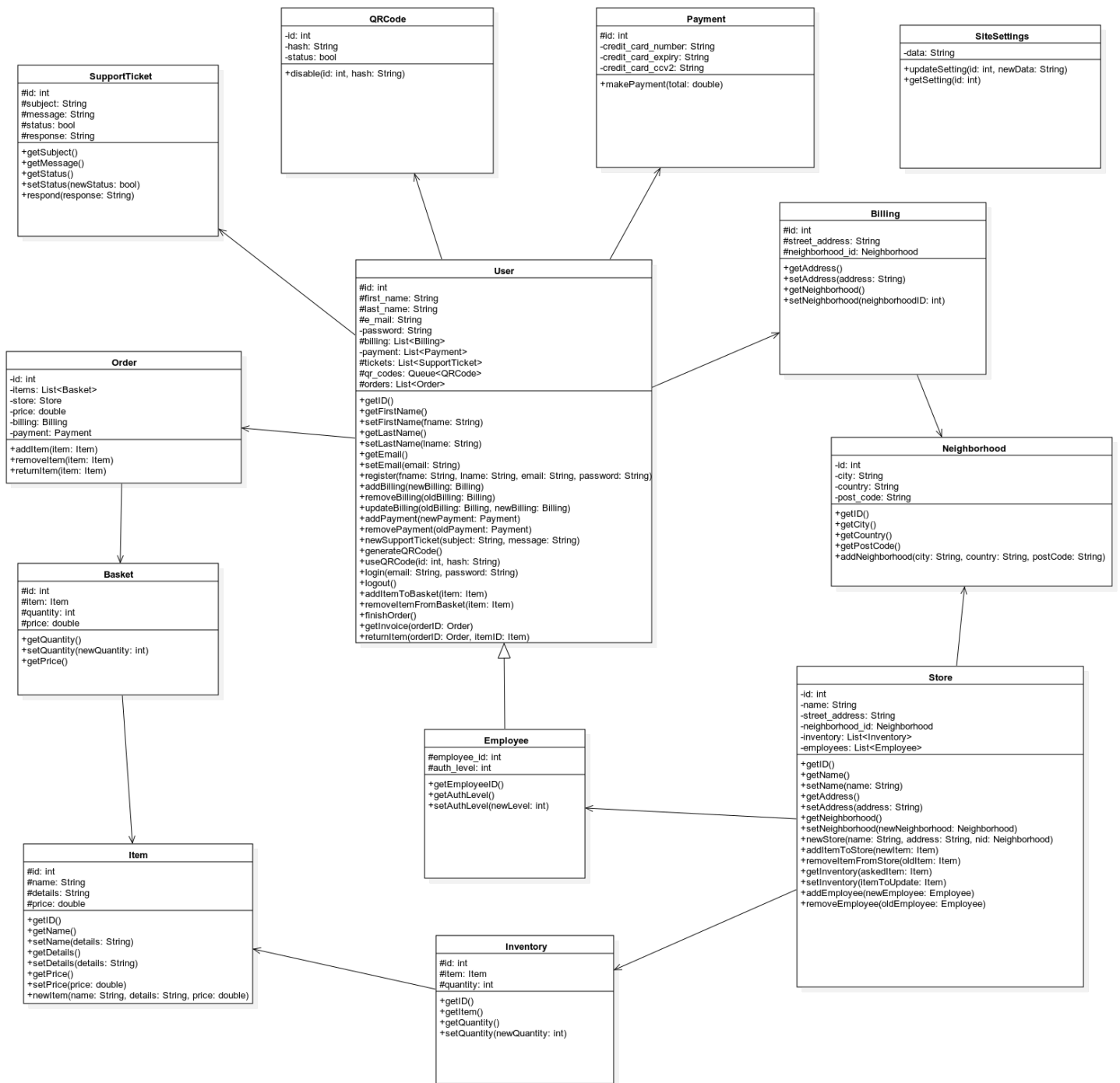


Figure 11: Class Diagram of the System

3.2 Functions

The use case diagram can be seen at Figure 12.

Use Case ID	1
Use Case Name	Sign-in
Actors	User
Descriptions	The user signs in to the system in order to use the system.
Preconditions	User exists in the system with valid username, password, and e-mail address.
Postconditions	The user has successfully logged in.
Normal Flow	<ol style="list-style-type: none">1. User clicks on the “sign-in” button.2. The system prompts the user to enter their username (or e-mail) and password.3. The user types in their username (or e-mail) and password to the system.4. System verifies the typed in information.5. System signs in the verified user.
Exceptions	<ol style="list-style-type: none">4. If the entered information is not valid, the system displays “The entered user information is not correct” message. Use case returns on step 2 of normal flow.

Table 3: Use case for Sign-in

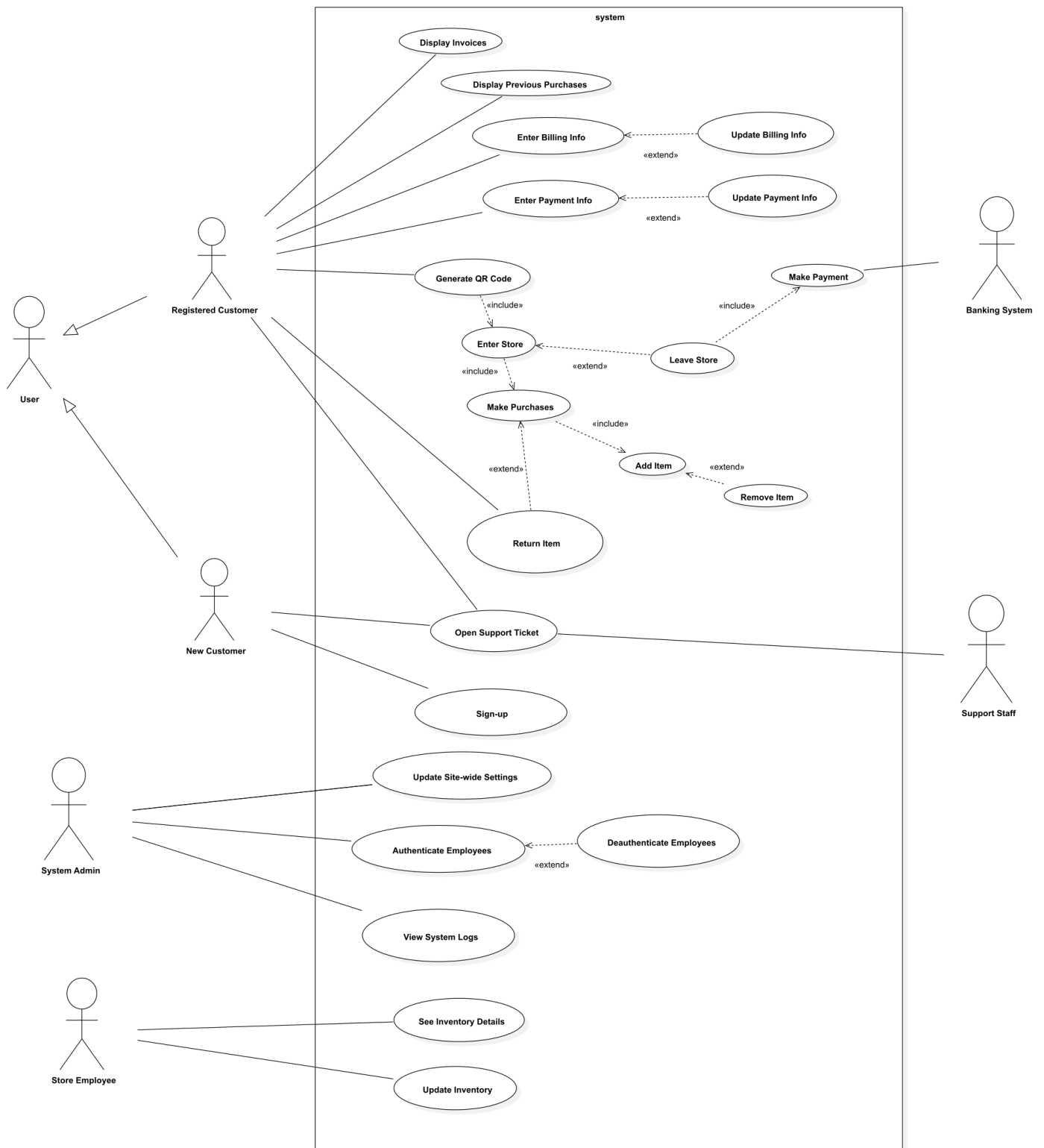


Figure 12: Use Case Diagram

Use Case ID	2
Use Case Name	Sign-up
Actors	User
Descriptions	The user signs-up to the system in order to use its functionalities.
Preconditions	The user that is not on the system enters their e-mail address, username, password, first name, and last name.
Postconditions	The user has successfully signed up to the system.
Normal Flow	<ol style="list-style-type: none"> 1. User installs the Amazon Go mobile app. 2. User clicks the “Sign-up” Button. 3. User fills in the registration form with their e-mail address, username, password, first name, and last name, and checks the “I agree to the Terms and Conditions” checkbox. 4. System verifies the entered information. 5. User gets validated. 6. User receives an e-mail to activate their account. 7. User clicks the link in their e-mail to activate their account. 8. The account gets activated.
Exceptions	<ol style="list-style-type: none"> 4. If the entered e-mail and/or the username is already in the system, the user will be shown a “The entered e-mail and/or username is already in use” message. Use case returns on step 3 of normal flow. 8. If the user does not click on the activation link, the account never gets activated, therefore the user cannot use their account. Use case returns on step 6 of normal flow.

Table 4: Use case for Sign-up

Use Case ID	3
Use Case Name	Enter payment and billing information
Actors	User, Banking System
Descriptions	The user enters their billing and payment information to the system.
Preconditions	The user must be signed up to Amazon Go. The user must be signed in to Amazon Go mobile app. The Amazon Go mobile app is active and running.
Postconditions	The user successfully entered their billing and payment information to the system.
Normal Flow	<ol style="list-style-type: none"> 1. The user clicks on “Payment & Billing Information” button. 2. The user fills in the number, expiry date and CVC2 number of the credit card, as well as the card holders full name. 3. The system verifies the validity of the credit card information using an API call through the Banking System. 4. The credit card gets validated. 5. The user fills in their street address, and tax number. 6. The system validated the address. 7. The billing and payment details gets saved.
Exceptions	<ol style="list-style-type: none"> 4. If the entered credit card information is not valid, the API call from the Banking System returns a false value. Use case returns on step 3 of normal flow. 6. If the entered address information is invalid, the user cannot save that information. Use case returns on step 5 of normal flow.

Table 5: Use case for Entering Payment and Billing Information

Use Case ID	4
Use Case Name	Enter the store
Actors	User
Descriptions	The user enters an Amazon Go store.
Preconditions	<p>The user must be signed up to Amazon Go.</p> <p>The user must be signed in to Amazon Go mobile app.</p> <p>The user must have their payment and billing info entered to the system.</p> <p>The user must have their mobile phone with Amazon Go mobile app installed on themselves.</p>
Postconditions	The user successfully entered the store.
Normal Flow	<ol style="list-style-type: none"> 1. The user opens the Amazon Go mobile app. 2. The application shows a QR Code. 3. The user shows the QR Code to the turnstile QR Code reader sensor. 4. The turnstile allows the user to enter.
Exceptions	<ol style="list-style-type: none"> 1. The user fails to run the Amazon Go mobile app (due to software or hardware error). Use case returns on step 1 of normal flow. 2. The mobile app fails to generate a valid QR Code. Use case returns on step 2 of normal flow. 3. The turnstile QR Code reader fails to read the QR Code. Use case returns on step 3 of normal flow.

Table 6: Use case for Entering the Store

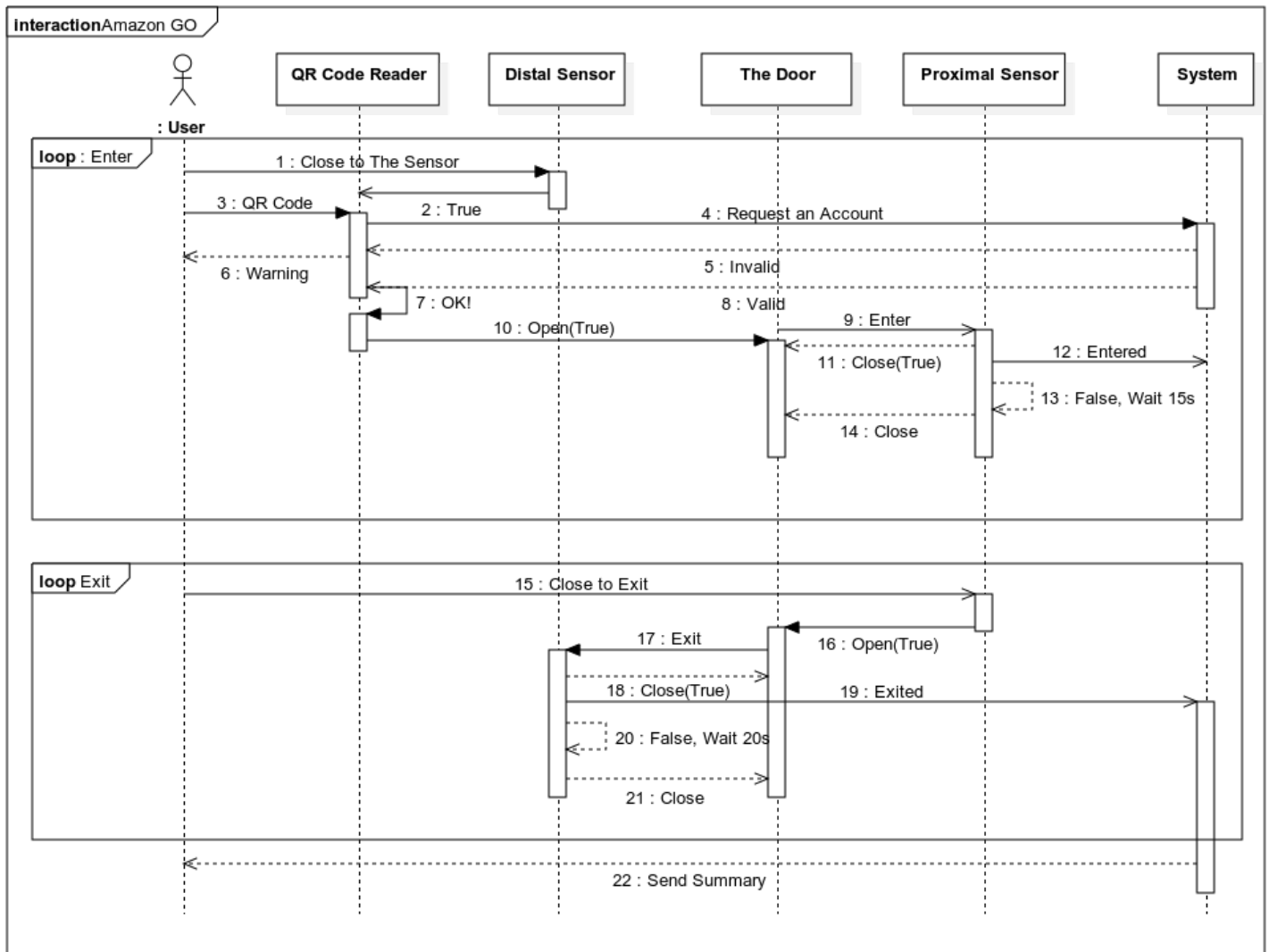


Figure 13: Entering the Store Sequence Diagram

Use Case ID	5
Use Case Name	Buy a product
Actors	User
Descriptions	The user purchases one or more products.
Preconditions	<p>The user must be signed up to Amazon Go.</p> <p>The user must be signed in to Amazon Go mobile app.</p> <p>The user must have their payment and billing info entered to the system.</p> <p>The user must have their mobile phone with Amazon Go mobile app installed on themselves.</p> <p>The user must be entered to the store.</p>
Postconditions	The user has successfully purchased one or more products.
Normal Flow	<ol style="list-style-type: none"> 1. The user picks an item from the shelf. 2. The system adds the item to the users virtual basket. 3. The user walks through the exit turnstile. 4. The system computes the total number of items and automatically deducts the amount from the credit card of the user. 5. An automated invoice gets generated and sent to the user.
Exceptions	<ol style="list-style-type: none"> 1. If the user puts back an item that they have picked earlier (into the related or unrelated shelf), the system removes the item from the users virtual basket. The use case returns on step 0 of normal flow. 4. If the system is unable to deduct the amount from the credit card of the user (in the case of the credit card being maxed out), the user becomes in debt to Amazon and the user will get a warning stating that they should update their payment information as soon as possible. The use case returns on step 3 of normal flow.

Table 7: Use case for Buying a Product

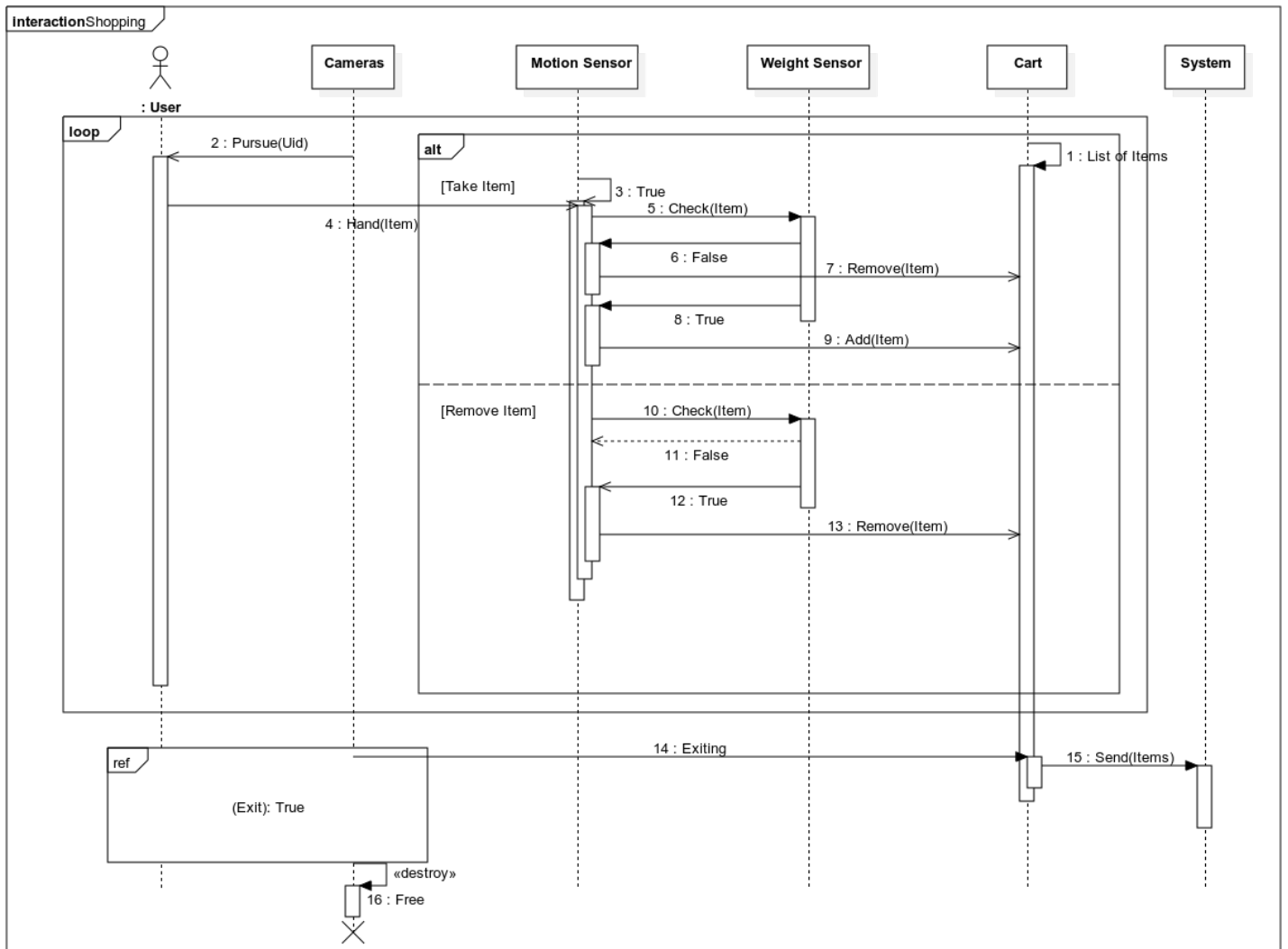


Figure 14: Buying a Product Sequence Diagram

Use Case ID	6
Use Case Name	Display previous purchases
Actors	User
Descriptions	The user displays their previous purchases through the Amazon Go mobile app.
Preconditions	The user must be signed up to Amazon Go. The user must be signed in to Amazon Go mobile app. The Amazon Go mobile app is active and running.
Postconditions	The user views their previous purchases.
Normal Flow	<ol style="list-style-type: none"> 1. The user clicks on “Previous Purchases” button. 2. The mobile app displays a list of previous purchases as clickable objects that allows the details of the previous purchases to be seen.
Exceptions	<ol style="list-style-type: none"> 2. If the user has no previous purchases, the system displays a message with “You have no previous purchases” text. The system returns on step 1 of normal flow.

Table 8: Use case for Displaying Previous Purchases

Use Case ID	7
Use Case Name	Display invoices
Actors	User
Descriptions	The user views their invoices.
Preconditions	<p>The user must be signed up to Amazon Go.</p> <p>The user must be signed in to Amazon Go mobile app.</p> <p>The Amazon Go mobile app is active and running.</p> <p>The user is on the “Previous Purchases” page.</p> <p>The user has made at least one purchase.</p>
Postconditions	The user views their invoices related to their purchases.
Normal Flow	<ol style="list-style-type: none"> 1. The user clicks on the purchase that they want to view the invoice of. 2. The mobile app displays the invoice and allows a PDF version to be downloaded via the button “Download Invoice”.
Exceptions	None

Table 9: Use case for Displaying Invoices

Use Case ID	8
Use Case Name	Return a product
Actors	User
Descriptions	The user returns a product to the store that they have purchased earlier.
Preconditions	<p>The user must be signed up to Amazon Go.</p> <p>The user must be signed in to Amazon Go mobile app.</p> <p>The Amazon Go mobile app is active and running.</p> <p>The user must have made a purchase.</p>
Postconditions	The user successfully returned an item that they have purchased earlier to Amazon.
Normal Flow	<ol style="list-style-type: none"> 1. The user views their related purchase through the “Previous Purchases” button. 2. The user clicks the “Return item(s)” button. 3. The user selects the item(s) that they want to return. 4. The user clicks on the “Approve” button. 5. The system generates a QR Code. 6. The user scans the QR Code to the return lockers of the store. 7. The system will unlock one of the available lockers door. 8. The user puts the item(s) that they want to return to the locker and shuts the door. 9. The system validates the action. 10. The user gets a refund.

Exceptions	<ol style="list-style-type: none"> 5. The mobile app fails to generate a valid QR Code. Use case returns on step 4 of normal flow. 6. The QR Code reader fails to read the QR Code. Use case returns on step 5 of normal flow. 7. The system fails to unlock a locker door due to either mechanical error or not having an empty locker. The mobile app displays “The return system is currently unavailable, please try again later.” message. Use case returns on step 6 of normal flow. 8. The locker door fails to lock. The mobile app displays “The return system is currently unavailable, please try again later.” message. Use case returns on step 7 of normal flow. 9. The system detects that the user did not put the items that they wanted to return, or put irrelevant items to the locker. The user gets a warning. Use case returns on step 8 of normal flow. 10. The banking system fails to make a refund. The user gets notified and selects either getting Amazon Credits instead or using the Banking System later again. Use case returns on step 9 of normal flow.
------------	---

Table 10: Use case for Returning a Product

Use Case ID	9
Use Case Name	Open a support ticket
Actors	User, Support Staff
Descriptions	The user opens a support ticket in order to communicate with the Support Staff.
Preconditions	The Amazon Go mobile app is active and running.
Postconditions	The user successfully opened a support ticket.
Normal Flow	<ol style="list-style-type: none"> 1. The user clicks on the button “Support”. 2. If the user has already signed-in, the user enters the subject and message fields. If the user is a non-signed-up user, the user must fill their full name, and e-mail. 3. The system verifies entered information. 4. The support ticket gets opened with a unique ticket ID. 5. The ticket ID gets shown to the user.
Exceptions	<ol style="list-style-type: none"> 3. If the entered information is not valid, the system displays an error message. Use case returns on step 2 of normal flow. 4. If the system fails to generate a unique ID, the support ticket cannot get generated. The system displays an error message. Use case returns on step 3 of normal flow.

Table 11: Use case for Opening a Support Ticket

Use Case ID	10
Use Case Name	Authenticate employees
Actors	System Admin, Store Employee
Descriptions	The System Admin authenticates a new employee to use the system.
Preconditions	The employee must be signed up to the system. The system admin must be signed in to the management panel.
Postconditions	The employee has successfully authenticated.
Normal Flow	<ol style="list-style-type: none"> 1. The system admin clicks on “Authenticate Employees” button. 2. The system admin selects the store that the employee should be authenticated to. 3. The system admin approves the selection. 4. The system checks the entered information. 5. The employee has successfully authenticated for the aforementioned store.
Exceptions	<ol style="list-style-type: none"> 4. If the system admin does not fills all the required information, the system will display an error message. Use case returns on step 3 of normal flow.

Table 12: Use case for Authenticating Employees

Use Case ID	11
Use Case Name	Viewing inventory details
Actors	Store Employee
Descriptions	The store employee views inventory details.
Preconditions	The store employee must be authenticated. The store employee must be signed-in to the system.
Postconditions	The inventory details have been viewed successfully by the store employee.
Normal Flow	<ol style="list-style-type: none"> 1. The store employee clicks the “Inventory Details” button. 2. The store employee scans the items barcode, or QR code. 3. The inventory details of the product gets displayed.
Exceptions	<ol style="list-style-type: none"> 2. If the barcode or the QR code is not visible or readable, the system will give a warning and request the store employee to type in the name of the product. Use case returns on step 1 of normal flow. 3. If the entered information is invalid, no inventory detail can be displayed. The store employee will receive an error message. Use case returns on step 2 of normal flow.

Table 13: Use case for Viewing Inventory Details

Use Case ID	12
Use Case Name	View support tickets
Actors	Support Staff
Descriptions	The support staff views open support tickets.
Preconditions	The support staff must be signed-in to the system.
Postconditions	The support tickets have been successfully viewed.
Normal Flow	<ol style="list-style-type: none"> 1. The support staff clicks on the button “Support Tickets”. 2. The system displays support tickets that are currently open.
Exceptions	<ol style="list-style-type: none"> 2. If there are no support tickets that are open, the system will display a message stating that. Use case returns on step 1 of normal flow.

Table 14: Use case for Viewing Support Tickets

3.3 Usability Requirements

1. Generating a QR Code to enter the store shall be done automatically when the user opens the mobile app.
2. When the user first installs the application, a tutorial shall be displayed to teach the user how to use the system properly.
3. The mobile application should have a simple design to draw the users attention to the main concept of the store.
4. The mobile app should have a dark mode feature for users who prefer it.
5. The mobile app should ask for the user for feedback if their behaviour with the app changes (i.e. if they start to use the app less or more often than they did before)

3.4 Performance Requirements

1. The QR Code readers of the store entrance shall be high-resolution so the QR Code scanning process should not take longer than 2 seconds.
2. The enter/exit turnstiles must act fast to user activities. The user should not wait for the turnstile to recognize them for more than 5 seconds.
3. Billing and payment systems should act fast. The user should receive their receipt less than 15 minutes of leaving the store.
4. The store sensors and the server that processes the data from the sensors must be fast and reliable enough to allow having multiple users at the same section buying similar items at the same time.
5. The store must have enough computing power to handle an average of 2 users per square-meter inside the store.
6. The network system within the store shall be built with high-bandwidth cables and routers (for example CAT6E cables, gigabit routers), reducing delays as much as possible for a seamless shopping experience.
7. The internet connection of the stores must be high-bandwidth dedicated connections such that the delay between two nodes never surpasses 100 milliseconds.
8. The main server should have enough storage and computing power to handle requests coming in from different stores all over the country. An auto-scaling AWS solution shall be used.

3.5 Logical Database Requirements

The Entity-Relationship Model of the system can be seen at Figure 15.

1. Only a System Admin can access to the database tables.
2. Only a System Admin can authenticate an employee to a new store.
3. Only a System Admin can add new stores.
4. Only a System Admin can update the `site_settings` table.
5. The tables `support_tickets`, `qr_codes`, `billing`, `payment`, `employee`, `orders`, `order_basket` are weak entities since they cannot exist without having a *user*.
6. The table `inventory` is a weak entity since it cannot exist without having a *store* or an *item*.
7. The table `authentication` is a weak entity since it cannot exist without having a *store* or an *employee*.
8. The table `order_basket` is a weak entity since it cannot exist without having an *order* or an *item*.
9. The tables `stores`, `users`, `neighborhoods`, `items` are strong entities since they do not need any other entity to exist.
10. New entries to the `user` table shall be added automatically when a user signs-up using the mobile app. Similarly, new billing and payment information shall be added automatically by the system when a user adds such information using the mobile app.
11. A new QR Code entry on the `qr_codes` table shall be created when the mobile app generates a new QR Code for the user, and the generated QR Code shall become invalid as soon as it's used.
12. A new support ticket entry on the `support_tickets` table shall be created when a user generates a new support ticket.
13. A new employee entry on the `employee` table shall be created when a System Admin marks a user as an employee. Similarly, when a System Admin authenticates a user as an employee, an entry on `authentication` table shall be generated with the respective store and employee information.
14. All neighbourhoods within the country should have their information prefilled inside the `neighborhoods` table.

15. All stores should exist in the **stores** table.
16. All items in the stores should have their data stored in the **items** table. For each store, the **inventory** table should keep inventory information regarding each item.
17. When a user places an order, an entry on the **orders** table shall be created, and new entries for each item on the weak entity table **order_basket** shall be created, with references to items.
18. Users can view their own order, payment, billing details, and their own support tickets.
19. Store Employees can view inventory details and update them if necessary.
20. Support Staff can view support tickets with their statuses marked as *open*.
21. Passwords of users shall be stored as hashes that gets computed by using one of the cryptographic hash functions.
22. Integrity of database environment shall be checked by Database Management Interface and System Admins. Database Management Interface shall check system every 15 minutes and System Admins should run checks when the store is not open.
23. Database shall be backed up every hour, and when the store gets closed in the evening, the daily backup shall be sent to two different locations.

3.6 Design Constraints

1. All personal data of users shall be kept in accordance with the General Data Protection Regulation.
2. The **payment** table must be marked as confidential. Access to this table shall not be allowed and every action that requires a payment shall go through proper API protocols that hide the credit card information to ensure that the customers financial information is always kept secure.
3. The IP addresses and actions of the users shall be kept as long as the local regulations require. Afterwards, this information shall be anonymized and kept for research purposes only.

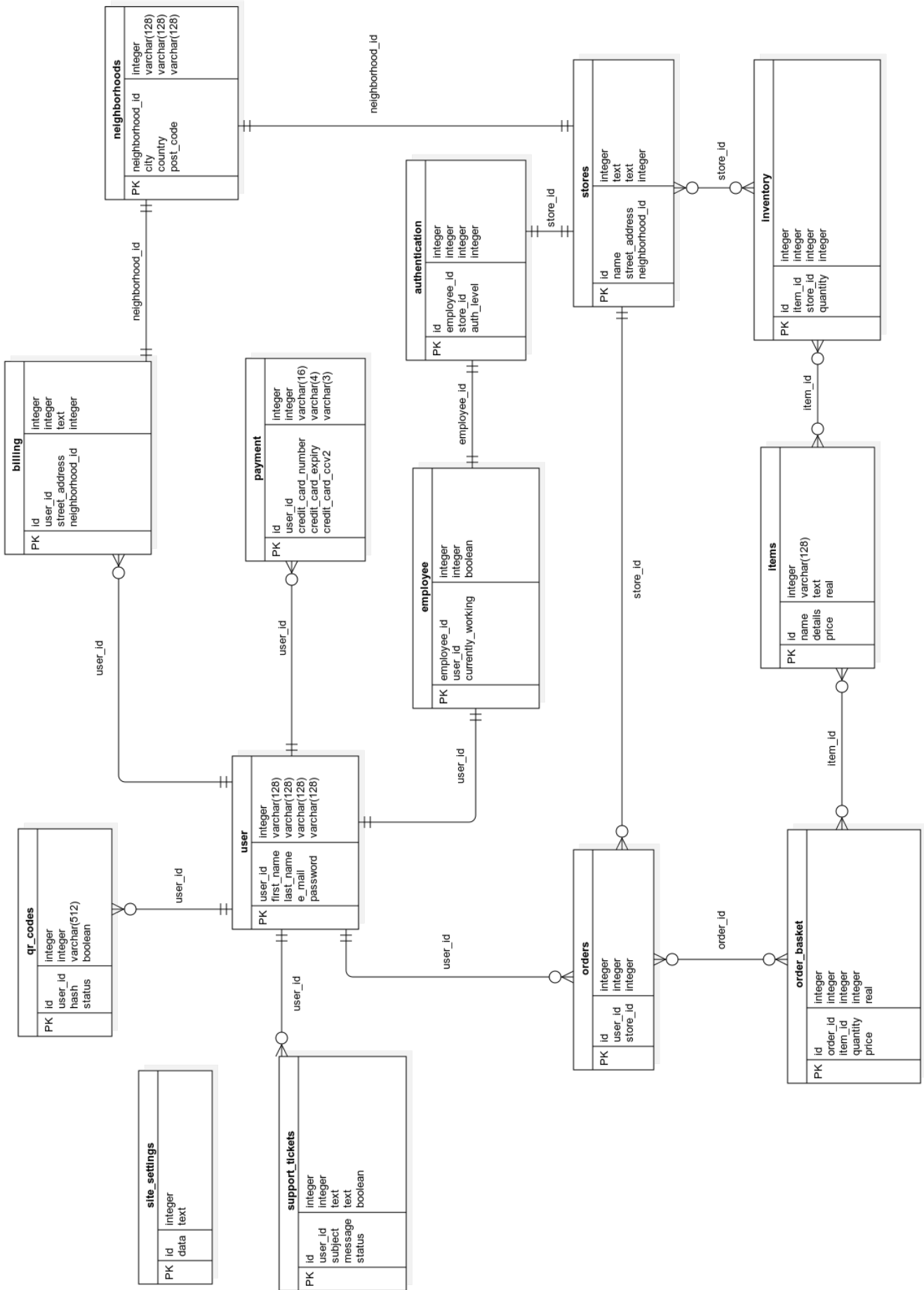


Figure 15: The Entity-Relationship Model for the Database of the System

3.7 Software System Attributes

3.7.1 Reliability

1. The system must be checked thoroughly before going live with automated tests. The probability of having an error with these tests shall not exceed 0.0001.
2. Any data corruption must be avoided. The system shall achieve this by having at least 5 concurrent database servers, and error handling mechanisms.
3. The system shall make daily backups automatically after the closing hour.
4. The system shall be tested if any component or module gets changed, updated, upgraded or if a new component gets installed.

3.7.2 Availability

1. The system shall face no downtime during operating hours. In order to achieve this, concurrent servers shall be used.
2. In case of failure, the system shall restart in less than a minute.
3. The system shall be able to auto-scale when in need.
4. The system shall have exception handling mechanisms to be able to recover from any exceptions by itself.
5. The System Admins shall work in a manner that allows at least 2 System Admins can be reachable if needed.
6. Any maintenance shall be done when the store is not operating.

3.7.3 Security

1. The Database shall be secure enough to withstand any SQL injection, or similar attacks.
2. The user passwords shall be hashed using SHA512 or a similar, state-of-the-art hashing algorithm. Plaintext storing of passwords must be avoided.
3. Remote connection to the Administration panel shall be authenticated using a username, a password and a physical key that gets connected to the terminal via a USB port.

3.7.4 Maintainability

1. The development process shall be completed using a Git version control environment and each commit must be well documented.
2. Any bug fix or feature development must be done by the development guidelines.
3. Developers shall abide by the coding style that is decided prior development. The aforementioned style document shall be available inside the Git repository.

3.7.5 Portability

1. The system shall be available for all mobile operating systems that have been in use for the last 4 years.
2. A mobile website version of the mobile app shall be available at most 6 months after the launch of the mobile apps.