# **Software Design Description**
for
Amazon Go

Prepared by
Alper Kocaman (2169589)
Ahmet Fatih Eser (2166387)

Middle East Technical University
Software Engineering

19 June 2020

Contents

Revision History

Revision	Date	Author(s)	Description
1.0	25.06.2020	Alper Kocaman Fatih Eser	Framework created

List of Figures

List of Tables

Chapter 1

Introduction

This document presents the smart store project which aims to make the shopping process easier and efficient for everyone. This document aims to give the related information needed to build the system with a planned and powerful way. The reader should be aware of the sections she/he needs refer to and investigate the document regarding the IEEE standards [1], as well as having general knowledge on UML diagrams [2] [3] [4] and software design background. The standards in this book are mostly taken from the book of Sommerville [5], any discrepancy should be cleared with the related section of the book.

1.1 Purpose of the System

The purpose of this project is to build the smart store project, namely "Amazon Go" [6]. "Amazon Go" project is a smart store project launched by Amazon whose objective is to present better, faster, and smarter shopping for everyone. In order to increase the automation along with the efficient use of resources, this project uses IoT and AI technologies to process data that come from sensors and cameras. This project diminishes the human error and inefficiency in the shopping process.

1.2 Scope

This document is to give detailed information on design and implementation process of the Amazon Go system. Many viewpoints and design specifications exist in this document as a both reference and knowledge source. Most of the document consists of kinds of views for the system. These architectural views describe the system in different ways with different details.

This document can be referred whenever implementation, test, maintenance or management processes need. Both low and high level descriptions are utilized for different views of the system. The standards in this document are based on the Software Requirements Specification document, and they modifications to the system should stick to these standards.

1.3 Stakeholders and their concerns

Different kinds of stakeholders are given in this section. It is best to present the identities, design concerns, addressing of the design concerns of the stakeholders which is given in table 1.1.

Stakeholder Identity	**Extra Explanation**	**Addressing Issue**
User	Users are loosely defined as every person that actively or passively uses the system. A person without any interference with the system, but exploits it with any way is defined as the user. Example users are customers, friends or relatives of customers, store staff, and logistic staff.	Context View
CV Developers	CV developers are defined as technical staff who designs, implements and tests the computer vision and product take/put system. More specifically developers for the Amazon Go system is ML engineers, pattern recognition engineers, optimization engineers and test engineers.	Composition View and Information View
Mobile App Developers	Mobile app developers are the ones who develop mobile application and its extensions. The group represents both customer interface and internal staff management interface. They are identified as software engineers, graphic designers, and back-end developers.	Composition View and Information View
System Integrator	Customer services, store functionalities, bank processes, database systems are all connected by the system integrator. They are composed of back-end engineers, test engineers, transaction managers and security specialists.	Composition View
Software Maintainers	Software maintainers are group of designers and engineers who are to improve appearance, functionality, speed etc. in order to create better customer experience.	Context View and Composition View
Server Maintainers	Server maintainers are the ones who are responsible for dependability of different kinds of servers such as database servers and web servers. They are web service engineers, database specialists and security engineers.	Context View and Composition View
Project Managers	Project managers are people who determine the general rules and steps. They are responsible for the cost, efficiency, overall performance and market perspective. They are mostly high-level managers, marketing staff, investors and quality engineers.	Context View and Interface View

Table 1.1: Stakeholder Composition and Explanation

Chapter 2

References

[1] IEEE, "Iso/iec/ieee international standard - systems and software engineering – life cycle processes – requirements engineering," *ISO/IEC/IEEE 29148:2011(E)*, pp. 1–94, 2011.

[2] J. Rumbaugh, I. Jacobson, and G. Booch, *Unified Modeling Language Reference Manual, The (2nd Edition)*. Pearson Higher Education, 2004.

[3] G. Booch, J. Rumbaugh, and I. Jacobson, *Unified Modeling Language User Guide, The (2nd Edition) (Addison-Wesley Object Technology Series)*. Addison-Wesley Professional, 2005.

[4] J. Erickson and K. Siau, "Theoretical and practical complexity of modeling methods," *Commun. ACM*, vol. 50, pp. 46–51, 08 2007.

[5] I. Sommerville, *Software Engineering*, 10th ed. Pearson, 2015.

[6] "Official amazon go website," 2020. [Online]. Available: https://www.amazon.com/b?ie=UTF8&node=16008589011

Chapter 3

Glossary

Term	**Definition**
CV	Computer Vision. The part of the system that engages in data processing of customer card.
ML	Machine Learning. Also refers to learning based algorithms or procedures.
SMS	Store Management System. More info available in Composition View.
CMA	Customer Mobile App. More info available in Composition View.
SED	Security Enhanced Database. More info available in Composition View.
SSD	Solid State Disk
HDD	Hard Disk Drive
RAID	Redundant Array of Independent Disks
PCI	Payment Card Industry Data Security Standard

Table 3.1: Glossary

Chapter 4

Architectural Views

4.1 Context View

In this view, the context diagram of the system is represented at first. After that, use case models of the system is given and they are explained with the description tables. The table descriptions of use cases include alternative flows and error flows if exist. These descriptions show how the system should behave in various scenarios and they are meant to be a baseline for how the functionalities of the system shall be implemented. Context diagram represents the actors in the Amazon GO project and how they interact with each other. On the other hand, Use Case model shows different kinds of internal or external users and how systems interact with Amazon Go system through a variety of cases.

4.1.1 Context Diagram

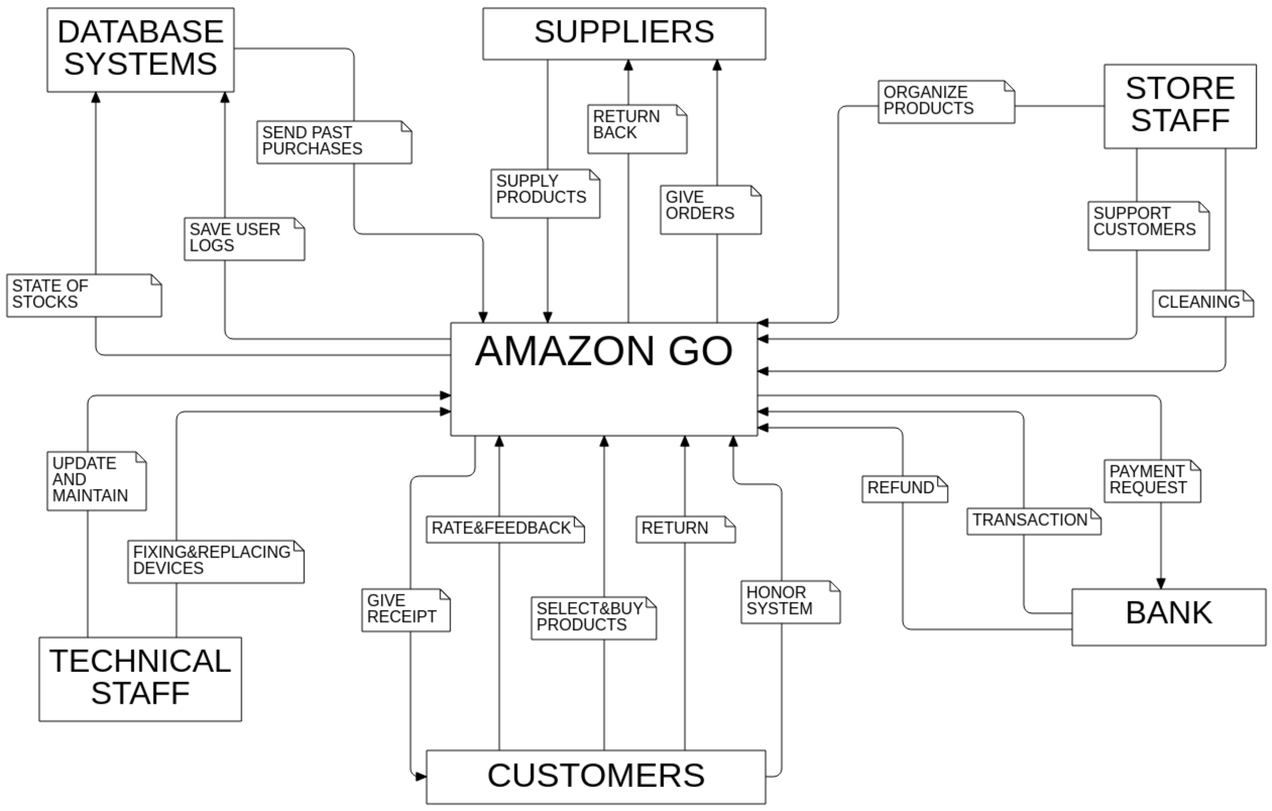


Figure 4.1: Context Diagram

4.1.2 Use Case Diagram

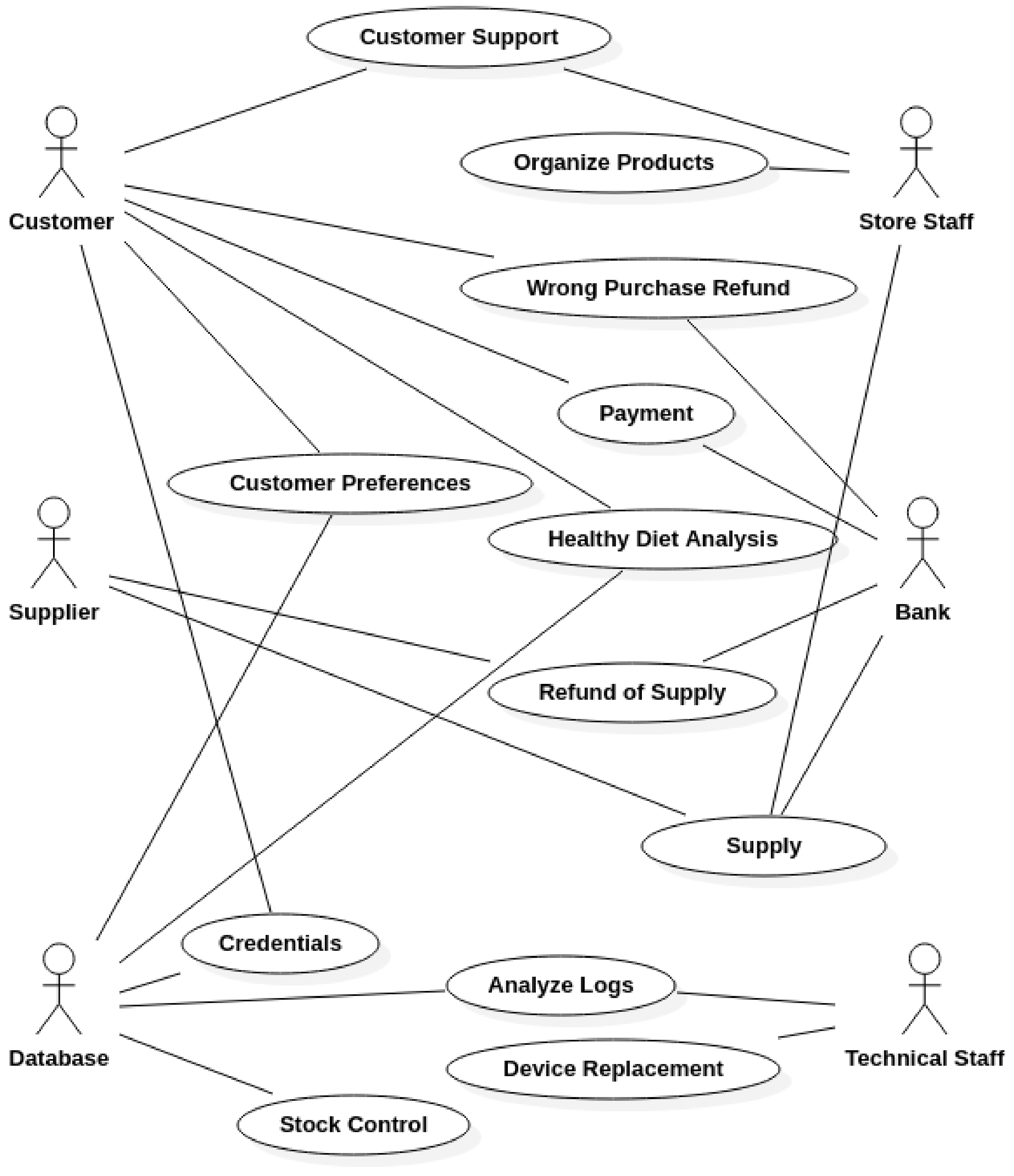


Figure 4.2: Use Case Diagram

4.1.3 Use Case Descriptions

Use case name	Healthy Diet Analysis
Actors	Customer, Database
Description	Users learn their dietary habit and see what they ate, when they ate, how frequently they ate. Also, users learn which products contain which ingredient and sustenance values. User can set notifications.
Data	All past purchases, purchase times, frequencies, and amounts.
Preconditions	At least one purchase from the user, food information is in database.
Stimulus	Customer creates the request, and the request stimulates the system to respond.
Basic flow	1) User requests the analysis by using their interface. 2) Database system takes this request from the interface and returns the purchase list. 3) Healthy diet analysis method prepares the report and sends it to the user.
Alternative flow	4) User decides that a product is harmful, the system notifies the user at the future purchases.
Exception flow	If no past purchase data exist, or food database is missing, terminate the process.
Post-conditions	User is notified in case of harmful products specified by the authorities.

Table 4.1: Healthy Diet Analysis Function

Use case name	Customer Support
Actors	Store Staff, Customer
Description	When customer needs any support, information request, system feature related information, or make any wrong transaction, store staff responds to the customer needs.
Data	Questions asked by the customer.
Preconditions	When user is in trouble or in a situation where information is lacking.
Stimulus	Any difficulty user experiences during the purchase which stimulates the store staff.
Basic flow	1) Customer request or asks about some stuff 2) Store staff responds to the needs of the customer
Alternative flow	-
Exception flow	-
Post-conditions	-

Table 4.2: Customer Support Function

Use case name	Organize Products
Actors	Store Staff
Description	Store staff organizes the products in case of displacement of products through different shelves, or the products are broken because of some external force. Any physical damage to the objects that are not in the category of technical devices in the market are fixed by the store staff.
Data	Broken object names and ids, displaced product names and ids.
Preconditions	When some product is displaced because of any reason or some objects that are not counted as technical devices get damaged because of any external force.
Stimulus	Any broken, displaced, damaged object excites the staff.
Basic flow	1) An object gets some external impact. 2) Store staff fixes the issue.
Alternative flow	3) Store may choose to wait until the crowd in the market to decrease.
Exception flow	4) If the fix is not practical, then send the broken objects to external repair center.
Post-conditions	Customers are notified in case of a possible danger because of the physical damage.

Table 4.3: Organize Products Function

Use case name	Wrong Purchase Refund
Actors	Customer, Bank
Description	When a user decides to return the good, or there is a mistake in the decision making process resulting in wrong billing; the user can request a payback and get her/his refund from bank.
Data	Purchase history, customer feedback and declaration, store product stock, transaction
Preconditions	Wrongly done payment, incorrect item in checkout list of a user, damaged products cause the refund process.
Stimulus	User applies for refund when he or she notices the extra payment or damaged product.
Basic flow	1) User requests for refund. 2) The system is designed with the honor system(with an understanding that those looking to trick the system and steal things are in the minority). Thus, system accepts the refund request. 3) Bank returns the extra money to the user back.
Alternative flow	2) If the user makes frequent refund requests, system checks the user account and previous purchases in order to decide whether the user tries to trick the system or not.
Exception flow	3) If user ID cannot be resolved, the refund request is ignored.
Post-conditions	User is notified about refund process by the bank.

Table 4.4: Refund Function

Use case name	Payment
Actors	Customer, Bank
Description	When a customer finishes shopping and exits from the store, total price of the products is drawn by the bank from the bank account of the user. This money will be deposited to the bank account of the Amazon Go store.
Data	Customer's latest shopping cart, product stock of the store
Preconditions	User enters into the store by using mobile app QR code and purchases at least 1 product.
Stimulus	When user leaves from store with products(a nonzero bill).
Basic flow	1) User exits from store with bought product/s. 2) The system associates user with purchases. Also, it detects that user exited from the store. 3) System automatically requests the total price of the products from the bank by giving customer credentials. 4) Bank draws the total cost from customer's account and put that money into store's account.
Alternative flow	-
Exception flow	4) If user's account does not have sufficient money, bank uses credit for the user and put the money into the store's account.
Post-conditions	Both user and store are notified by the bank about the latest transaction.

Table 4.5: Payment Function

Use case name	Customer Preferences
Actors	Database, Customer
Description	For each customer, system can predict next purchases of the customer since it keeps past purchases and their contents in the database system.
Data	Past purchases of customers, purchase dates of customers, store
Preconditions	At least 5 purchase has been completed by the customer.
Stimulus	The system automatically starts to analyze past purchases when customer completed 5 purchases.
Basic flow	1) When a customer enters a store, get past purchase records and past purchase date data for all stores. Since past purchase information from the current store is more valuable, get the store id and separate past purchases information from the current store. 2) The system uses its own methods backed by machine learning to guess.
Alternative flow	-
Exception flow	1) If the database connection is lost due to a technical problem, system cannot take the necessary data.
Post-conditions	System compares the actual bought products with its predictions for self learning/training mechanism.

Table 4.6: Customer Preference Function

Use case name	Supply
Actors	Supplier, Bank, Store Staff
Description	Suppliers get their payment from the bank when they supply mass amount of products to the store.
Data	Supplier credentials, total value of the supplied products, store
Preconditions	Some of the products' stock values are decreased below a certain threshold(this threshold is defined for each product separately).
Stimulus	When a supplier brings requested products to one of the Amazon GO stores.
Basic flow	1) A supplier supplies products to a store at the same time store staff controls new products for damage/expire date. 2) Supplier finishes his/her job in the store. 3) Total bill is calculated by the system. 4) System automatically start transaction in order to pay supplier.
Alternative flow	-
Exception flow	1) If new products do not have proper conditions, supply process is canceled.
Post-conditions	Database system is stimulated for stock renewing. Store is notified for the money transaction by the bank.

Table 4.7: Mass Payment Function

Use case name	Credentials
Actors	Database, Customer
Description	For each customer, system keeps his/her credentials like name, surname, contact information and bank account where further purchases can be drawn.
Data	Customer
Preconditions	A potential customer would like to use Amazon Go store since he/she wants to utilize store's innovation.
Stimulus	A new customer downloads store's mobile application and starts typing his/her credentials.
Basic flow	1) Customer downloads the mobile app and launches it. 2) In the welcome page, completes the tutorials/reads the guidelines. 3) In the registration page, he/she gives his/her name, surname, e-mail, address, phone number and credit card information. 4) Customer is successfully registered to the system.
Alternative flow	2) Some customers may skip this tutorials. If a customer doesn't want to give credit card info at that moment, system lets the customer to register but it doesn't let the customer to shop until he or she give the credit card information.
Exception flow	3) If given credit card or phone number is not valid, the potential customer is not registered into the customer database. 4) If any technical issue occurs in the process above, mobile app returns to the welcome page back and opens an error dialog.
Post-conditions	Customer is informed for successfully registration. A welcome mail is sent to customer's mail address by the system automatically.

Table 4.8: Credentials Function

Use case name	Analyze Logs
Actors	Database, Technical Staff
Description	Technical Staff views the system logs in order to maintain system.Logs will be saved in database system.
Data	System Logs
Preconditions	System is used by customers and logs are generated from different components of the system. Logs are sent and stored at database system.
Stimulus	A specified system log review time for a technical personal comes.
Basic flow	1) Customer actions generate logs and these logs are saved on database. 2) One of the technical staff personal logins and gets logs from database by using technical staff interface. 3) Technical staff personal reviews logs to learn about system usage.
Alternative flow	-
Exception flow	-
Post-conditions	System is maintained and improved by technical staff based on logs.

Table 4.9: Analyze Logs Function

Use case name	Device Replacement
Actors	Technical Staff
Description	Technical Staff repairs/replaces a non-operating/broken sensors/-electronic devices. These devices are detected by analyzing system logs.
Data	System Logs
Preconditions	System's electronic components logs their current situation into system.
Stimulus	A device's logs are not in an expected fashion or no log can be taken from a dedicated device.
Basic flow	1) Technical staff detects that a device might not be working properly. 2) One of the technical staff goes to the near place of it. 3) If the device is examined and is not working, technical staff removes it from the place. 4) In order not to effect the effectiveness of the system, a new device is mounted at that place. 5) Technical staff tries to repair non-operating component.
Alternative flow	3) Device might be operating correctly but it doesn'l log due to a technical error. 5) If non-operating device is broken, technical staff cannot repair it.
Exception flow	-
Post-conditions	System is maintained and improved by technical staff. The replacing operation is saved into logs as well.

Table 4.10: Device Replacement Function

Use case name	Stock Control
Actors	Database
Description	Database system checks the stock condition of the products regularly. If the remaining product number is less than a defined threshold (this threshold is unique for each product), it alerts the system.
Data	Stock in database
Preconditions	A properly working database system is sufficient.
Stimulus	A predefined time interval exceeds and database system automatically does the regular check.
Basic flow	1) Timer is set after each stock control. 2) If timer interrupts, then database system does its regular check. 3) If everything is fine, timer is set up again.
Alternative flow	3) If one of the product stock level is less than predefined threshold, database system gives alert.
Exception flow	-
Post-conditions	The regular check and its results are saved into logs.

Table 4.11: Stock Control Function

Use case name	Refund of Supply
Actors	Supplier, Bank
Description	When supplied products are not in a good condition(broken/expire date is not sufficient), store may request a refund from the supplier .
Data	Supply
Preconditions	A group of product/s is supplied by the supplier.
Stimulus	Store staff requests a refund bu using interface.
Basic flow	1) Supplier supplies requested products. 2) Store staff checks the condition of products. 3) If condition of some products are not good, refund process is initiated with the help of store staff interface. 4) Total price of the supply is drawn from the supplier's bank account and put back to the store's account.
Alternative flow	3) If products are in good condition, payment function is applied instead of refund function.
Exception flow	-
Post-conditions	Bank notifies both the supplier and store.

Table 4.12: Refund Function

4.2 Composition View

Composition view is explained in this section to its full extent. Component diagram and deployment diagram are given in figures 4.3 and 4.4 respectively and their design explanations are given first in respective sections. Then, reasoning is given for these components.

4.2.1 Component Diagram

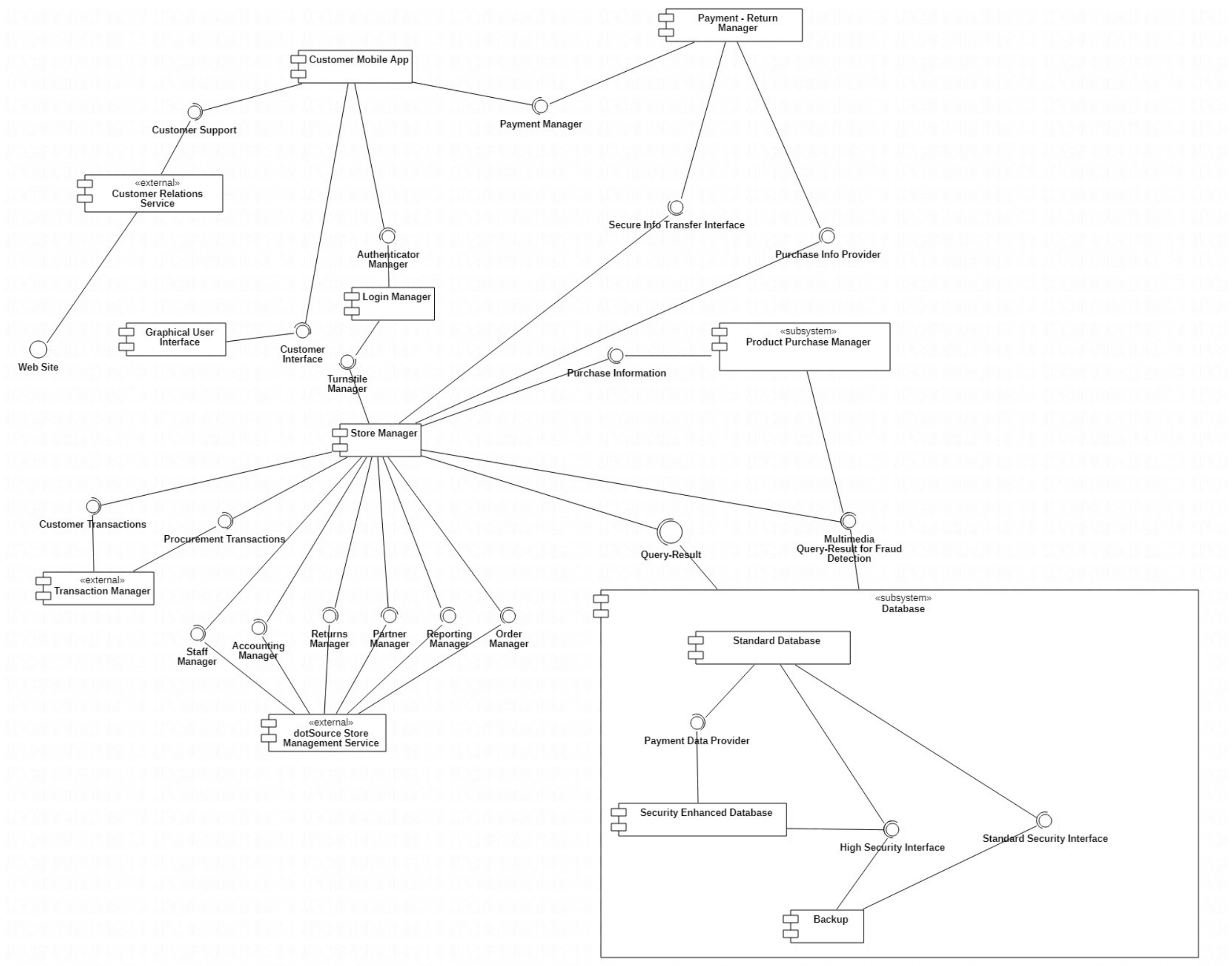


Figure 4.3: Component Diagram

The system composes of two main module, which are "Customer Mobile App" and "Store Management System". Other components are either their sub-systems, i.e. parts, or small systems their design is not complicated as such.

Firstly, CMA ("Customer Mobile App") is the component to provide all needs of customer through her/his smartphone. This component is not directly connected to the "Store Management System", but through some interfaces. This component also provides an interface that contains required functionalities for the best customer experience and reliable feedback system defined in the requirements document.

Secondly, SMS ("Store Management System") is a virtual system that manages the operations done by the store. Store is an abstract object that arranges most of the components of the system. Every physical store has an SMS. A customer interacts the SMS of the store she/he enters via the mobile application, sensors, turnstile. Also, SMS is responsible for procurement of lacking goods, physical cleanness, transactions, determination of staff acts. Moreover, purchase management, user data etc. all passes through the store. This configuration, where most of the services are accessed through store manager, allows developers to trace the error, fix security issues and make more reliable system. Moreover, problems can be fixed locally in the stores.

4.2.2 Deployment Diagram

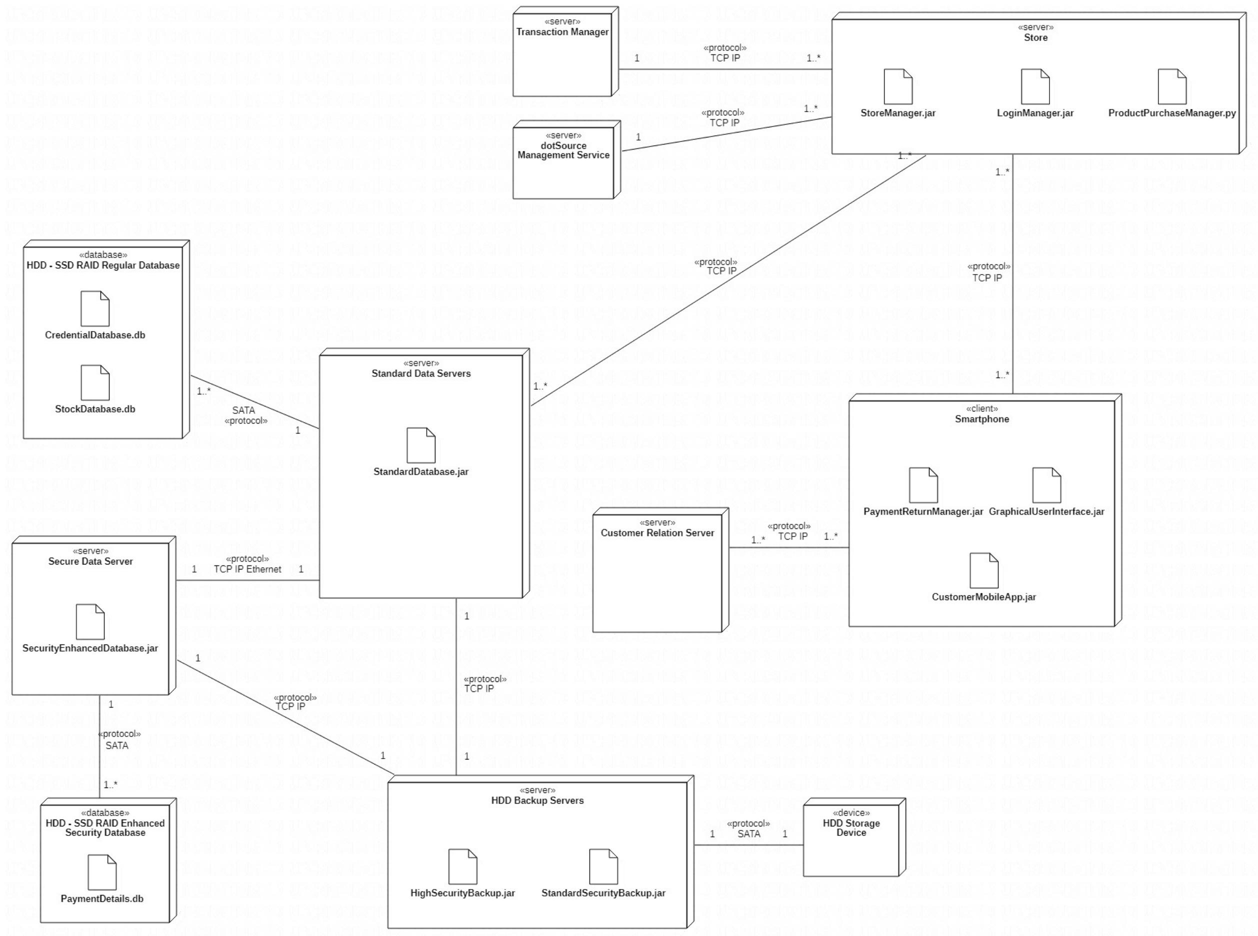


Figure 4.4: Deployment Diagram

Component	Purpose	Reasoning
Login Manager	Provides interfaces for customer login to the mobile app and login to the store by using QR code authentication. These interfaces are between App - Physical Store and App - Virtual Store, which is the server that represents the store.	LoginManager is designed as a separate component so that security specialists can work efficiently on the special subject as well as mechanical engineers can work on turnstile with proper information transfer from those.
Payment - Return Manager	The manager is the one of the core parts of the Amazon Go system where payments, receipts, total price information is transferred from SMS to CMA and vice versa. Registering the new credit cards and payment information is done also through this channel.	There are special requirements put by the government for the security issues. Therefore, this part is implemented by industry experts and security specialists.
Product Purchase Manager	This module is created in order to handle decision making of items purchased. Sensors, cameras, customer information are all supplied to this module directly from the physical devices and decisions are made in this module. Moreover, it stores data which stores videos and sensor data in memory for a while when a suspicious user detected.	Purchase decision is a complicated process, and interfacing such a procedure as a component makes the system operated easier. Therefore, all of the processes required are implemented under this component and not details are given to the other components to avoid confusions.

Transaction Manager	All transactions are processed and managed by his component. It is used by the accounting staff and it makes sure all of the transactions are valid and useful. It is provided by the banks that are in operation which is why it is external.	Accounting managers, data scientists, and company managers work together since this component requires interdisciplinary design and use process.
Store Management Service	This subsystem is a platform where store staff, health status of equipment and sensors, clean logs etc. are managed and arranged. It is used by the managers of the store as well as the staff personnel.	This is decided to be a separate component since it does not resemble to the remaining system and has unique attributes.
Standard Database	Standard database is created for many operations that the system implement. It connects with both SMS and "Product Purchase Manager" which is because there are many logs from SMS in addition to video and sensors logs that are for fraud detection. Note the RAID system utilized in the configuration.	This database is the first layer of the database system where standard security measures are applied. Since it is the first layer, it is fast and reliable and able to work in real time.
Security Enhanced Database	This component is separate from the standard database and provides extra security by providing a firewall, which watches the traffic for important decisions. It also occupies a different location than the standard database. Moreover, access to it requires first access to the standard database.	It is a separated component for security reasons and different requirements for standard and SED. Standard database requires speed and reliability whereas SED requires security in the first place.

Table 4.13: Component Descriptions

Deployment Location	**Description**
dotSource, Transaction Manager, Customer Relation Server	These are external services managed by other companies.
Standard Data Server	This is the server that hosts "Standard Database" component.
Secure Data Server	This is the server that hosts "Security Enhanced Database" and has a firewall and monitoring at the entrance.
HDD - SSD RAID Regular Database	This is a database environment for regular database.
HDD - SSD RAID Enhanced Security Database	This is a database environment for security enhanced database.
HDD Backup Servers	These are the servers and database management platform for the backup process.
HDD Storage Device	This is the storage device for backup.

Table 4.14: Deployment Details

4.2.3 Design Rationale

Note that some design rationale is given under the component itself for description. However, in this section more detailed rationale is given for both component selection, structure and deployment.

- Standard database provides reliability and availability which is why there are numerous of them. Redundancy is important for reliable and faster operation.
- Secure database is placed in a different environment than the standard database which is why these information is protected by the most sophisticated standards and PCI rules. Note that, a separate storage provides a very powerful way of monitoring the network traffic and block suspicious attempts.
- PPM is stored in the store in order to reduce latency and distribute the computation. Moreover, moving data would be a very costly solution which requires all data of video / sensors to be transferred to the processing server. Therefore, localized data is used whenever possible.

- Note that data is moved from PPM to database which may occupy a great bandwidth. However, another database design and implementation would yield higher costs than moving data. Also, PPM system is going to only store suspicious data which implies a little data to be transferred, quantitatively approximated as 0.001 %.
- Availability times are also improved to 0.99 times with the multiple store servers across the country. The main reason for many stores are both reliability and availability in addition to locality of data.
- RAID system is utilized for database with 1-to-many relations in order to boost both speed and life time of the components.
- Hybrid SSD - HDD system is used for the best optimization. SSD access times are much better than HDD however their lifetime for write operation is very limited. Therefore, data that is not expected to change frequently but access frequently such as credit card information etc. are stored on SDD where dynamic data is stored on HDD such as multimedia, logs etc. This way, both performance and lifetime requirements are satisfied.

4.3 Information View

Information view specifies persistent data of the system by giving interface class diagram and database class diagram. For the interface diagram, description of the operations are explained in the relevant section. Also, input/output and exception information of these operations are supplied to the user. For the database class diagram, related subsection explains how data can be stored on the database subsystem by giving necessary database operations. Furthermore, for each database operation, which CRUD(create-read-update-delete) operation is applied and what is the data type questions are answered. At the end of this view, in the design rationale part, the reason behind the selection of this interface/database design and strengths of this design are clarified.

4.3.1 Class Diagram

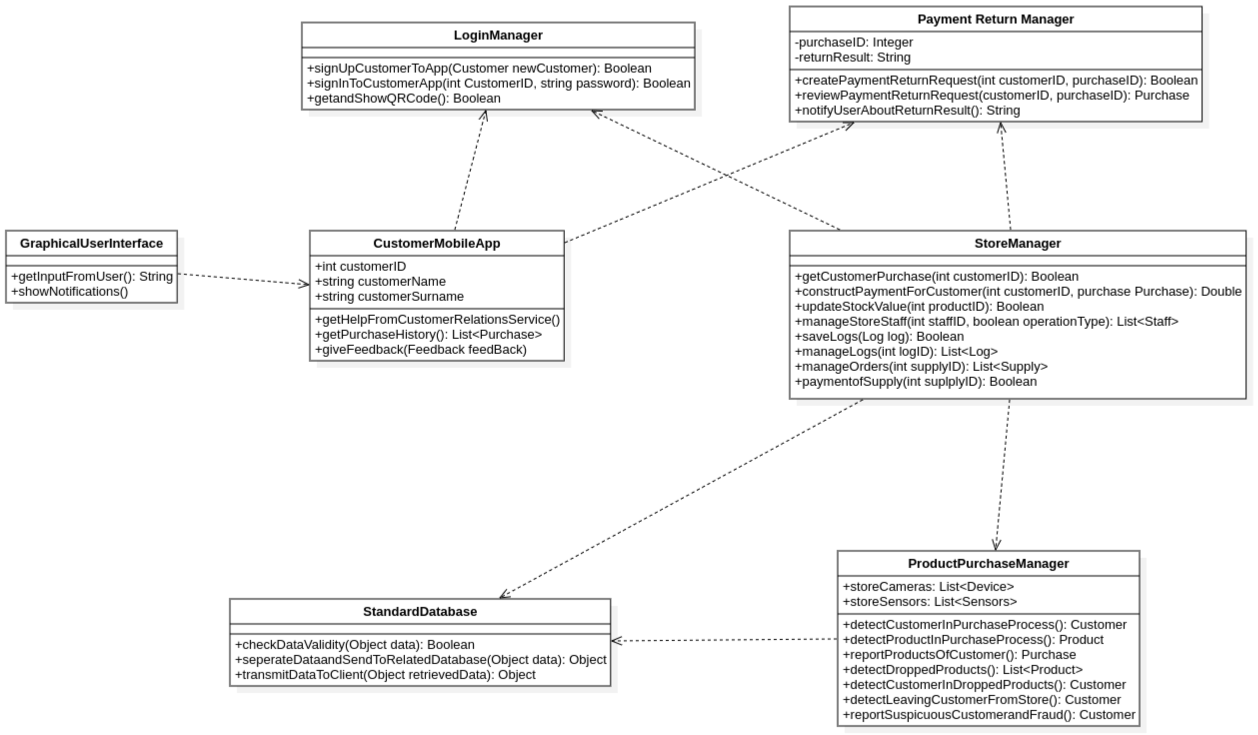

LoginManager
+signUpCustomerToApp(Customer newCustomer): Boolean
+signInToCustomerApp(int CustomerID, string password): Boolean
+getandShowQRCode(): Boolean
Payment Return Manager
-purchaseID: Integer
-returnResult: String
+createPaymentReturnRequest(int customerID, purchaseID): Boolean
+reviewPaymentReturnRequest(customerID, purchaseID): Purchase
+notifyUserAboutReturnResult(): String
GraphicalUserInterface
+getInputFromUser(): String
+showNotifications()
CustomerMobileApp
+int customerID
+string customerName
+string customerSurname
+getHelpFromCustomerRelationsService()
+getPurchaseHistory(): List<Purchase>
+giveFeedback(Feedback feedBack)
StoreManager
+getCustomerPurchase(int customerID): Boolean
+constructPaymentForCustomer(int customerID, purchase Purchase): Double
+updateStockValue(int productID): Boolean
+manageStoreStaff(int staffID, boolean operationType): List<Staff>
+saveLogs(Log log): Boolean
+manageLogs(int logID): List<Log>
+manageOrders(int supplyID): List<Supply>
+paymentofSupply(int suplplyID): Boolean
ProductPurchaseManager
+storeCameras: List<Device>
+storeSensors: List<Sensors>
+detectCustomerInPurchaseProcess(): Customer
+detectProductInPurchaseProcess(): Product
+reportProductsOfCustomer(): Purchase
+detectDroppedProducts(): List<Product>
+detectCustomerInDroppedProducts(): Customer
+detectLeavingCustomerFromStore(): Customer
+reportSuspicuousCustomerandFraud(): Customer
StandardDatabase
+checkDataValidity(Object data): Boolean
+seperateDataandSendToRelatedDatabase(Object data): Object
+transmitDataToClient(Object retrievedData): Object

Figure 4.5: Service Interfaces Diagram

Operation	**Description**
getCustomerPurchase	Customer purchased products are obtained the store management system from product purchase manager.
constructPaymentForCustomer	Given purchase and customerID, the bill is created.
updateStockValue	After each shopping processes, products' stocks with given IDs are decreased.
manageStoreStaff	Store staff and store managers are viewed, controlled and updated.
saveLogs	During the working process, logs need to be created and saved to the database.
manageLogs	Created logs are managed via this interface.
manageOrders	Supplies of the store are created and controlled.
paymentofSupply	Suppliers are paid when they supply the store via this interface.
createPaymentReturnRequest	When customers return a product, their money is returned.
reviewPaymentReturnRequest	Customers can see the progress in their return request.
notifyUserAboutReturnResult	When return operation is finished, the result is declared to the customer.
detectCustomerInPurchaseProcess	Cameras determine the customer by using cutting edge computer vision algorithms.
detectProductInPurchaseProcess	Cameras and sensors determine the product taken by the customer by using computer vision and sensor fusion algorithms.

detectDroppedProducts	This operation is applied when a product is dropped by a customer.
detectCustomerInDroppedProducts	Cameras and sensors detect the customer in the dropping process and drop that ıtem from customer's current shopping cart.
detectLeavingCustomerFromStore	This operation is applied when one person leaves the store.
reportProductsOfCustomer	Send customer purchase to the store management system for further processes.
reportSuspiciousCustomerandFraud	This operation is always running in the background for detecting malicious users of the system. If a person detected, s/he is reported to the store staff and store security.
signUpCustomerToApp	Customer creates a new account to use the mobile application.
signInToCustomerApp	Customer logins his or her account by filling necessary credentials.
getAndShowQRCode	In order to physically enter the store, customer should show QR code to the turnstile. This method creates the necessary QR code and show it to the customer.
getHelpFromCustomerRelationsService	With the help of this operation, customer can connect to the customer relations service.
getPurchaseHistory	Customer can see all of his/her purchases.
giveFeedback	This operation provides a feedback mechanism to the system.
checkDataValidity	Incoming data integrity and validity should be check before saving to the actual database.
seperateDataAndSendToRelatedDatabase	This operation separates incoming datum and send it to the related database.
transmitDataToClient	When client needs to the data, this operation act as a layer. It connects to the related database, obtain the data and send it to the client.
getInputFromUser	This operation takes the data from the customer and send it to the further processes.
showNotifications	When a notification comes to the app, this operation is activated. It shows the notification to the customer.

Table 4.15: Service interfaces Operations and Descriptions

Operation	**Inputs**	**Outputs**	**Exceptions**
getCustomerPurchase	customerID	Boolean(operation result)	Technical problem when detecting customer, then customer info is not valid.
constructPaymentForCustomer	customerID purchase	Double	CustomerID is not valid Purchase is not valid.
updateStockValue	productID	Boolean(operation result)	Database connection is not available ProductID is not valid.
manageStoreStaff	staffID operationType	List<Staff>	Database connection is not available staffID is not valid.
saveLogs	log	Boolean(operation result)	Created log is not valid Database connection is not available.

manageLogs	logID	List<Log>	Database connection is not available StaffID is not valid.
manageOrders	supplyID	List<Supply>	SupplyID is not present in the database.
paymentofSupply	supplyID	Boolean(operation result)	SupplyID is not present in the database. Error in the payment manager.
createPaymentReturnRequest	customerID purchaseID	Boolean(operation result)	CustomerID or purchaseID is not valid. Error in the database connection.
reviewPaymentReturnRequest	customerID purchaseID	Purchase with purchaseID	CustomerID or purchaseID is not valid. Error in the database connection.
notifyUserAboutReturnResult	-	String	-
detectCustomerInPurchaseProcess	-	Customer	Customer cannot be detected(by the cameras) for some reason(i.e. changed image).
detectProductInPurchaseProcess	-	Product	Product cannot be determined by ProductPurchaseManager for some reason.
detectDroppedProducts	-	List<Product>	No dropped product is found.
detectCustomerInDroppedProducts	-	Customer	Customer cannot be detected by computer vision algorithms.
detectLeavingCustomerFromStore	-	Customer	Customer cannot be detected by computer vision algorithms.
reportProductsOfCustomer	-	Purchase	Customer hasn't taken anything and go out.
reportSuspiciousCustomerandFraud	-	Customer	Customer cannot be detected by computer vision algorithms.
signInToCustomerApp	customerID password	Boolean(operation result)	Login credentials are not correct. Database connection is not available.
signUpCustomerToApp	newCustomer	Boolean(operation result)	Given customer information is not valid. Error in database connection.
getAndShowQRCode	-	Boolean(operation result)	-
getPurchaseHistory	-	List<Purchase>	Customer hasn't done shopping yet.
giveFeedback	feedback	-	Database connection is lost.
checkDataValidity	data(object class instance)	Boolean(operation result)	Incoming data is not valid.
seperateDataAndSend-	data(object class instance)	Object	Database connection is lost.

ToRelatedDatabase			
transmitDataToClient	retrieveData(object class instance)	Object	Error in the database connection.
getInputFromUser	-	String	Supplied input is not valid.
showNotifications	-	-	Customer closes the notification permission.

Table 4.16: Service interfaces Operations and Design

4.3.2 Design Rationale

- StoreManager handles the communication with external systems which are transaction manager and dotSourceManagementService.
- StoreManager class is the biggest class and all the event happened in store are known by it.
- Every login operation is handled in LoginManager class. These login operations are login to customer mobile app and login to store physically.
- Cameras and sensors are controlled by the ProductPurchaseManager. Interface operations of this class are used to detect products and customers in the shopping process.
- CustomerMobileApp handles the communication with external system customer relations service.
- Payment return manager handles the return processes of products and pay customers' money back.

4.3.3 Database Operations

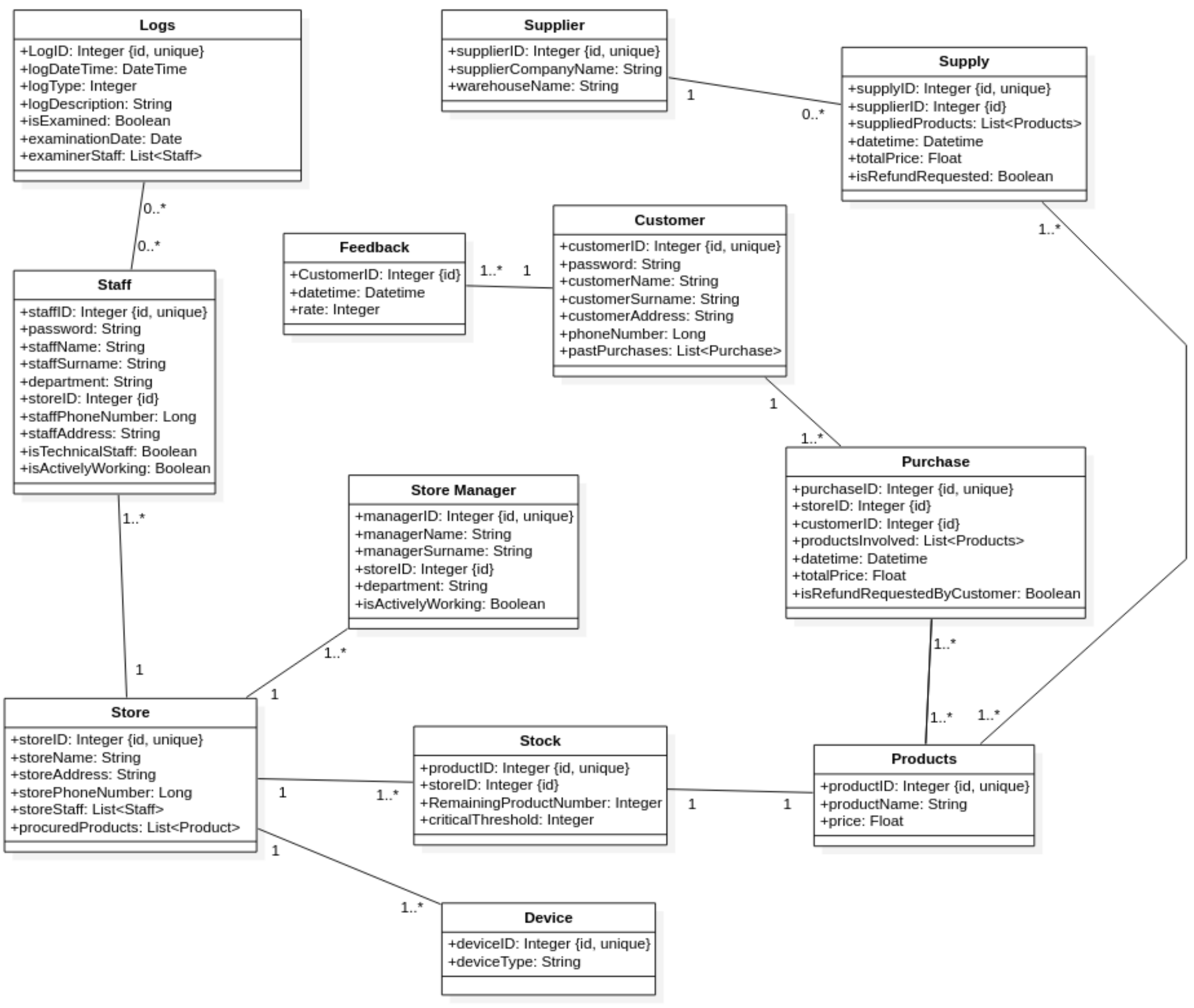


Figure 4.6: Database Class Diagram

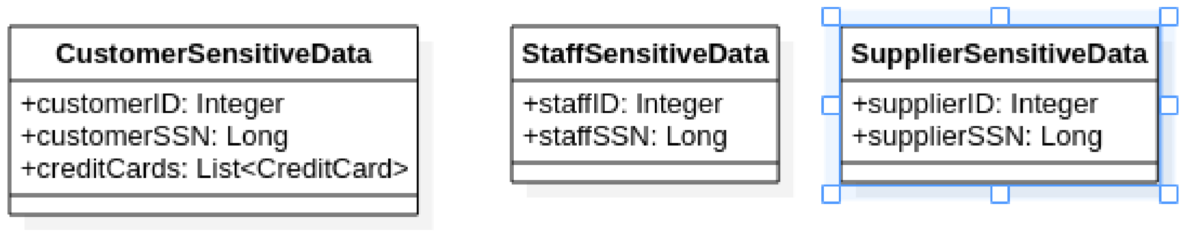


Figure 4.7: Database Class Diagram- Continued

Operation	**Description**
getCustomerPurchase	create: read: Purchase

	update: delete:
constructPaymentForCustomer	create: read: Purchase update: delete:
updateStockValue	create: read: Stock update: Stock delete: Stock
manageStoreStaff	create: Staff read: Staff update: Staff delete:
saveLogs	create: Log read: update: delete:
manageLogs	create: read: Log update: Log delete:
manageOrders	create: Supply read: Supply update: Supply delete:
paymentofSupply	create: read: Supply, Supplier update: delete:
createPaymentReturnRequest	create: read: update: Purchase, Customer delete:
reviewPaymentReturnRequest	create: read: Purchase , Customer update: delete:
notifyUserAboutReturnResult	create: read: Purchase update: delete:
detectCustomerInPurchaseProcess	create: read: update: delete:
detectProductInPurchaseProcess	create: read: update: delete:
detectDroppedProducts	create: read: update: delete:
detectCustomerInDroppedProducts	create: read:

	update: delete:
detectLeavingCustomerFromStore	create: read: update: Customer, Purchase, Product delete:
reportProductsOfCustomer	create: Purchase read: update: delete:
reportSuspiciousCustomerandFraud	create: read: Customer update: delete:
signUpCustomerToApp	create: Customer read: update: delete:
signInToCustomerApp	create: read: Customer update: delete:
getAndShowQRCode	create: read: update: delete:
getHelpFromCustomerRelationsService	create: read: update: Customer delete:
getPurchaseHistory	create: read: Purchase update: delete:
giveFeedback	create: Feedback read: update: Supply delete:
checkDataValidity	create: read: Object update: delete:
seperateDataAndSendToRelatedDatabase	create: Object read: Object update: delete:
transmitDataToClient	create: read: Object update: delete:
getInputFromUser	create: read: update: delete:
showNotifications	create: read:

	update: delete:

Table 4.17: Database CRUD Operations

4.3.4 Design Rationale

- MySQL is chosen as relational database management system and all the operations on the database are done via using MySQL.
- In the first database system, information needed by the system will be kept such as products, customers and staffs.
- There are associations across entities and these associations were shown in the database class diagram.
- Since this shopping application needs sensitive information such as customers' credit cards and store staffs' SSN number, these sensitive information will be kept into the second database system, these private information sections are not kept in the database with other entities.
- Second database system will be much more secure than first database system since it checks authenticity by using more credentials. Also, interface with this database highly secure.

4.4 Interface View

Interfaces are depicted in component diagram given as 4.3. This section is dedicated to elaboration and specialization of these interfaces. There are many interfaces which are naturally binding different types of components or some of them interest the customer and not an internal component which is why they are kept unconnected.

4.4.1 Internal Interfaces

CME - Payment Manager

Payment manager is the back-end for the all jobs related to payment. Credit card registration, return requests of the customer, information change about payment preferences and refunds etc. All of these are provided by the interface of the Return - Payment Manager.

When talking about **Design Rationale**; since payment and personal information is an important issue, it is not included in the CME; instead it is designed as an interface. This way, security and privacy concerns can be minimized as well as reducing the complexity of the CME. Another reason for the interface is to encapsulate the payment process. Note the standards are continuously changing and their evolution requires persistent update of the source code. Encapsulating the system with Payment Manager interface makes it possible to use the internal code as it is and only update the payment code for the changing standards.

Authenticator Manager

Authenticator manager provides the required interface for customer login for both store and app login. Note that store login is done through QR code read by the turnstile whereas app login is through the password and username. Login manager provides a single interface that is used by the CME for both store login and app login.

Design Rationale for this component is that information should flow between components which require much sophisticated techniques and standards. This interface provides simple operations like, sign in, generate QR code etc.; however, the operations are monitored and secured by specialists and it made the system very powerful in observability and security. Moreover, security technical staff and store-customer can work independently through the interface.

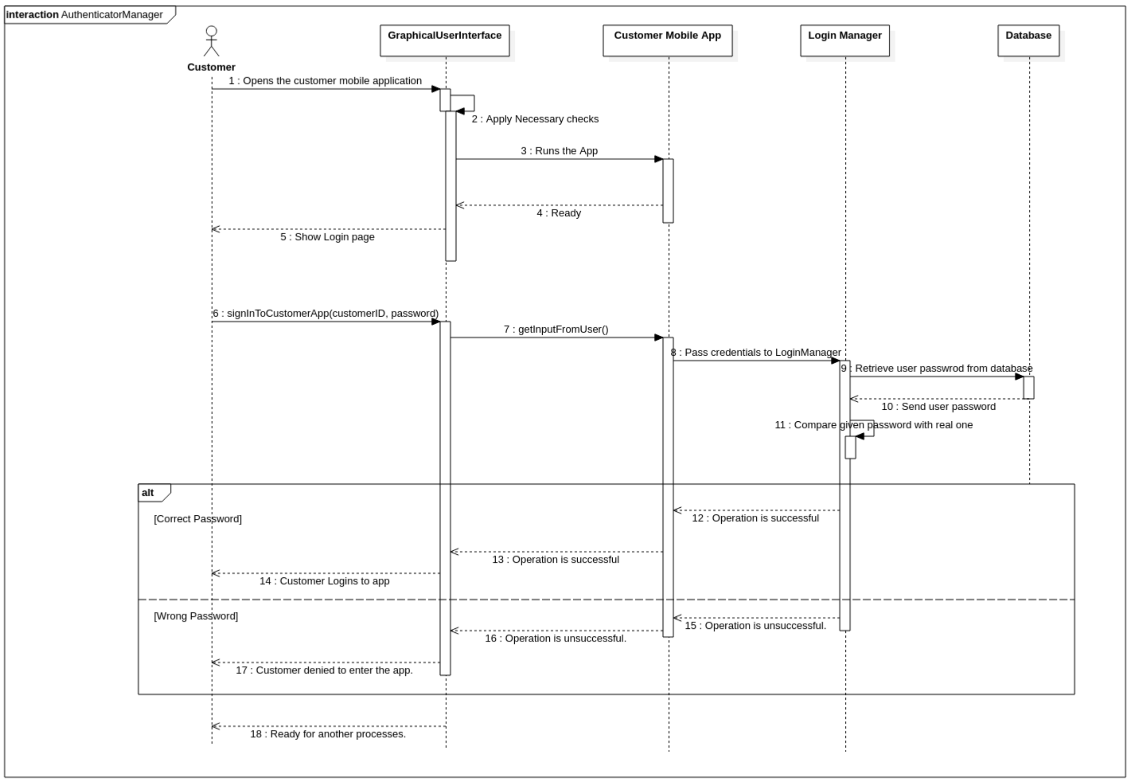

interaction AuthenticatorManager
Customer
GraphicalUserInterface
Customer Mobile App
Login Manager
Database
1 : Opens the customer mobile application
2 : Apply Necessary checks
3 : Runs the App
4 : Ready
5 : Show Login page
6 : signInToCustomerApp(customerID, password)
7 : getInputFromUser()
8 : Pass credentials to LoginManager
9 : Retrieve user passwrod from database
10 : Send user password
11 : Compare given password with real one
alt
[Correct Password]
12 : Operation is successful
13 : Operation is successful
14 : Customer Logins to app
[Wrong Password]
15 : Operation is unsuccessful.
16 : Operation is unsuccessful.
17 : Customer denied to enter the app.
18 : Ready for another processes.

Figure 4.8: Authenticator Manager Sequence Diagram

Turnstile Manager

Turnstile manager is the other interface that "Login Manager" provides. It is an important one because it is a bridge between customers and physical store. Store itself does not record who entered or exited the store, but it controls and uses methods of "Turnstile Manager" for managing information and controlling the turnstile.

Design Rationale for this interface is related to different disciplines used in the system. Turnstile is build and managed by hardware staff that is going to provide simple operation interface for it since technical details are far from easy instructions. Therefore, this interface allows the turnstile, i.e. control of entrance to the system, to be very easily controlled through the interface as well as customer data is transferred to the store system without any technical details.

Purchase Information

Purchase information from the "Product Purchase Manager" flows through this interface. Which customer bought which product, which products are placed to the shelves, setting customers inside the store etc. are all managed by this interface. Moreover, it provides information about the store status such as sensor status, crowd status etc. in case of need.

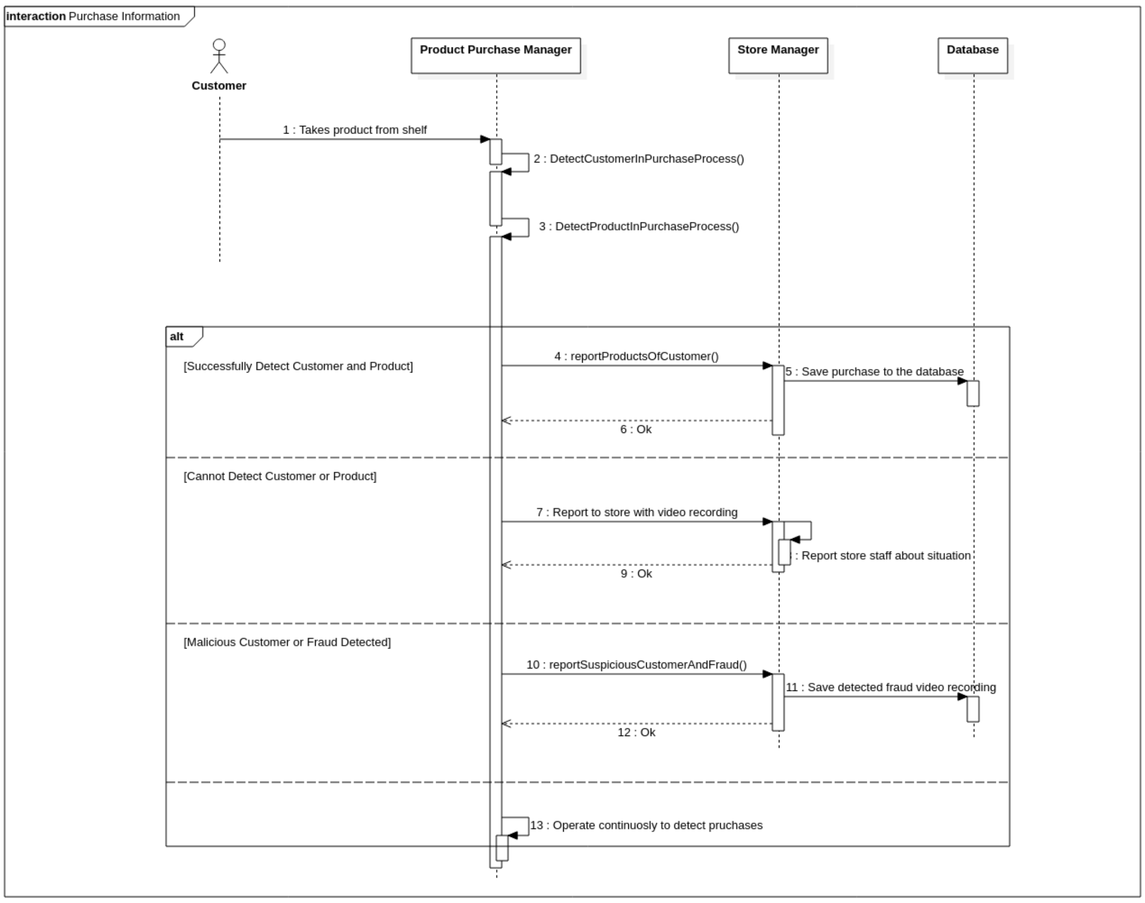

interaction Purchase Information
Customer
Product Purchase Manager
Store Manager
Database
1 : Takes product from shelf
2 : DetectCustomerInPurchaseProcess()
3 : DetectProductInPurchaseProcess()
alt
[Successfully Detect Customer and Product]
4 : reportProductsOfCustomer()
5 : Save purchase to the database
6 : Ok
[Cannot Detect Customer or Product]
7 : Report to store with video recording
: Report store staff about situation
9 : Ok
[Malicious Customer or Fraud Detected]
10 : reportSuspiciousCustomerAndFraud()
11 : Save detected fraud video recording
12 : Ok
13 : Operate continuosly to detect pruchases

Figure 4.9: Purchase Information Sequence Diagram

Design Rationale for this interface is the complexity of the operations done in "Product Purchase Manager". Operations are completely done in the component and only required ones are supplied to the store, and only important information is returned to the component. This way, both technical staff and store management become very fluent in operation. Moreover, newly registered store managers can easily manage store without any need for the knowledge for very technical part.

4.4.2 External Interfaces

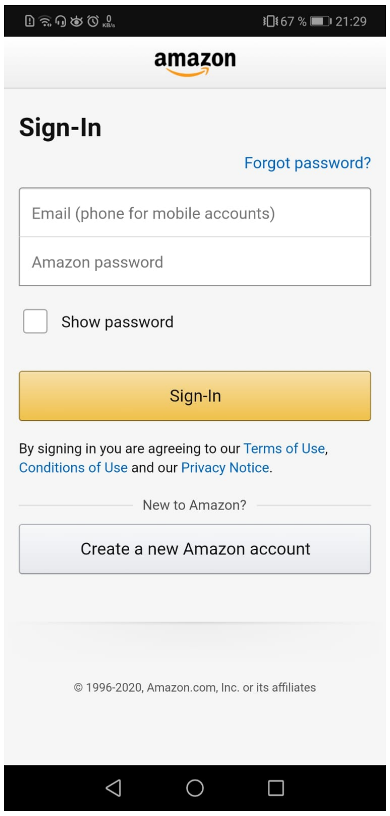
User Interface - Mobile App

The mobile app is the platform where customer interacts with the system. It is not shown in component diagram for a clear representation. Customer can log in, log out, manage account, configure the app and do regular app functionalities through that interface.

Design Rationale for this interface is the need for interaction with the user. Also, communication, feedback, customer satisfaction is created through this interface. User can handle all jobs without knowing any technical details.

User Interface - Accounting Manager

Accounting is an important aspect of the store management. dotSource company provides the management software with already installed tools and third-parties available. Accounting management system is for financial staff and should be free of technical details.


amazon

Sign-In

Forgot password?

Email (phone for mobile accounts)

Amazon password

Show password

Sign-In

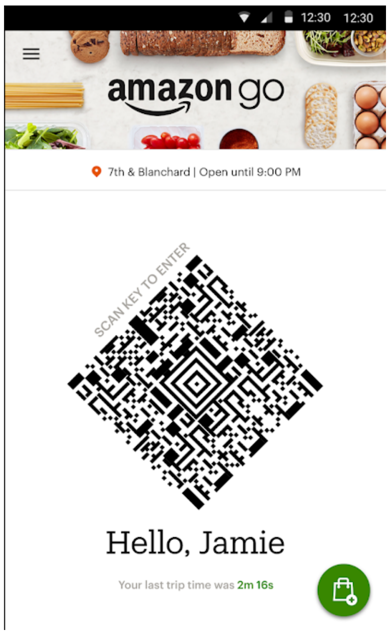
By signing in you are agreeing to our Terms of Use, Conditions of Use and our Privacy Notice.

New to Amazon?

Create a new Amazon account

© 1996-2020, Amazon.com, Inc. or its affiliates

(a) Log In Screen


amazon go
7th & Blanchard | Open until 9:00 PM
SCAN KEY TO ENTER
Hello, Jamie
Your last trip time was 2m 16s

(b) Sample Catalog

Figure 4.10: Sample User Interface

Design Rationale for this interface is that accounting requires a lot of operations. All of these operations are difficult to build from scratch and there are already developed tools which are cost efficient. Both cost and time management, it is best to use this external interface.

User Interface - Web Site

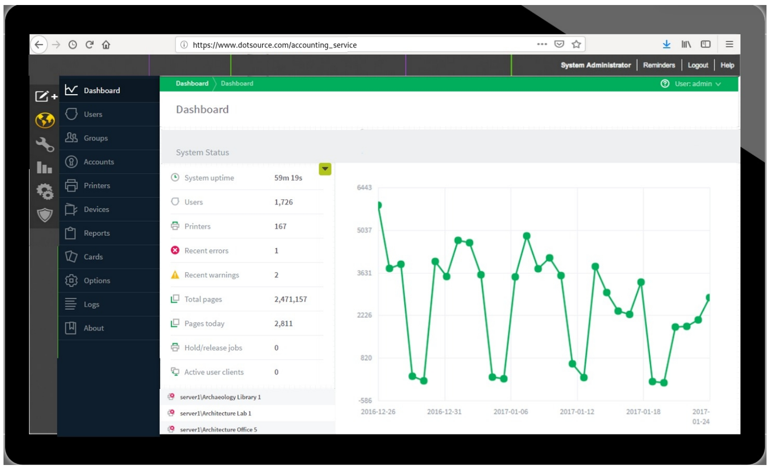
Web site is the official representation of the Amazon Go system. Many works and customer relations are done over the web site. Note that web site is hosted by component namely "Customer Relations Service" which is any host and customer relation provider. Feedbacks, purchases, advertisements, vision, mission and more are all represented in the web site.

Design Rationale for this part is actually fundamental. Almost all organizations have a website today and such a big company need a web site too. The reason for external interface is because its huge time consumption to have such an internal interface. Many job areas, such as designers, engineers, IT staff, call centers etc., serve for both build and maintenance of the web site which otherwise would by costly.

User Interface - Staff Manager

Staff manager is created for management of the staff, getting their feedback and creating the balance inside the store. Moreover, it helps other management tools by means of extra information and ordering.

Design Rationale for this interface is that already created systems are easily deployed without extra cost and maintenance requirements. The external interface provides common and stable tools for the most effective solution.



System uptime	59m 19s
Users	1,726
Printers	167
Recent errors	1
Recent warnings	2
Total pages	2,471,157
Pages today	2,811
Hold/release jobs	0
Active user clients	0

server1\Archaeology Library 1
server1\Architecture Lab 1
server1\Architecture Office 5

Figure 4.11: Sample Interface for Accounting Manager

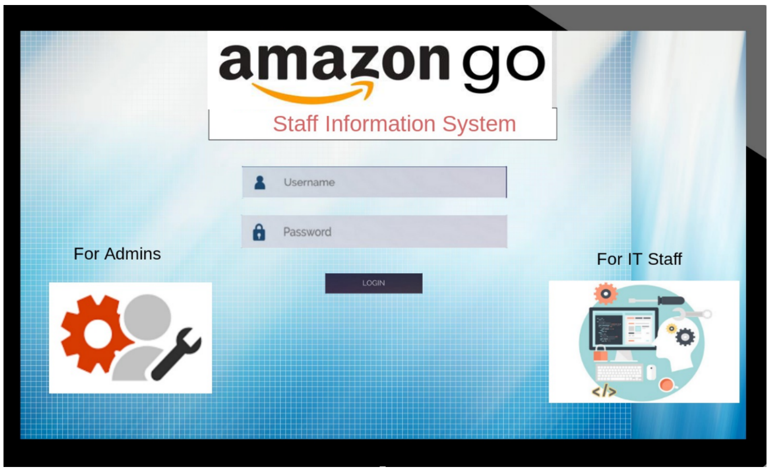


Figure 4.12: Sample Interface for Staff Manager

User Interface - Reporting Manager

This is also a user interface manager. It gathers reports automatically upon user request and creates a shape for the reports. Also, it is responsible for all connected subsystem data readings for the most accurate result.

Design Rationale for this interface is related to the hard part of the data gathering and management. Reports require a lot of data, graphs, condition records. This interface encapsulates all of the required work with its sophisticated algorithm with state of the art techniques and provides a simple interface which is less costly and faster.

System/Service Interface - Returns Manager

All returns are done through the mobile app and no human interaction is present. Moreover, all returns are logged and processed in the SMS. This interface provide the system to easily manage all of the data automatically by SMS.

Design Rationale for this part is the need for a management system without many details. Note that dotSource company provides this product for free which is one of the main reasons to select it. Moreover, organizing with already developed tools is obviously user friendly.

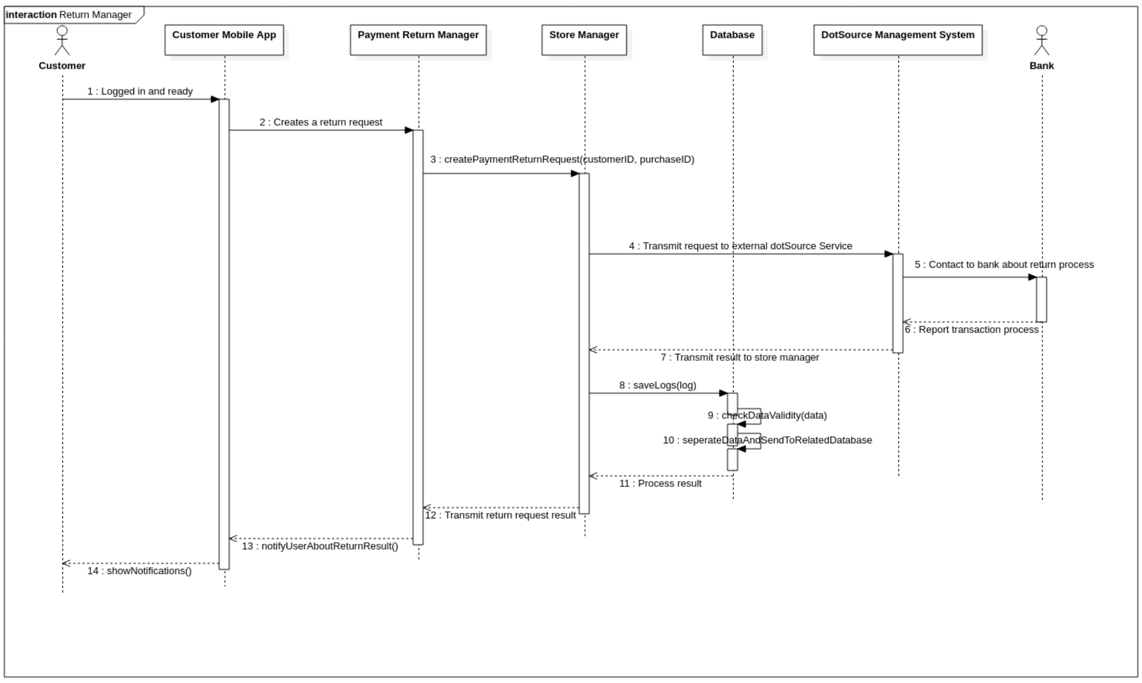

interaction Return Manager
Customer
Customer Mobile App
Payment Return Manager
Store Manager
Database
DotSource Management System
Bank
1 : Logged in and ready
2 : Creates a return request
3 : createPaymentReturnRequest(customerID, purchaseID)
4 : Transmit request to external dotSource Service
5 : Contact to bank about return process
6 : Report transaction process
7 : Transmit result to store manager
8 : saveLogs(log)
9 : checkDataValidity(data)
10 : seperateDataAndSendToRelatedDatabase
11 : Process result
12 : Transmit return request result
13 : notifyUserAboutReturnResult()
14 : showNotifications()

Figure 4.13: Returns Manager Sequence Diagram

System/Service Interface - Partner Manager

Partners are core stones of the business and a successful business firm has to manage their partners for the best prices, costs and delivery qualities. Partner manager makes all arrangements over IP and mail and automatically manages the most suitable choice for the partners. Moreover, this interface provides statistics to SMS.

Design Rationale is actually related to the need of partner management and easy way of using external service which has partners in its database and give better decisions with advanced algorithms tendering best results.

System/Service Interface - Order Manager

Order manager is an interface used by SMS in order to make relevant orders and make the store products always available. In addition to ordering lacking products, it can optimize the purchase by using data obtained in the partner manager.

Design Rationale for this part is firstly its free of charge property when used with the other components. In addition, it serves for best purchases that are efficient and faster. Since these algorithms and data are both supplied by the company dotSource, it is chosen to be used by this external source.

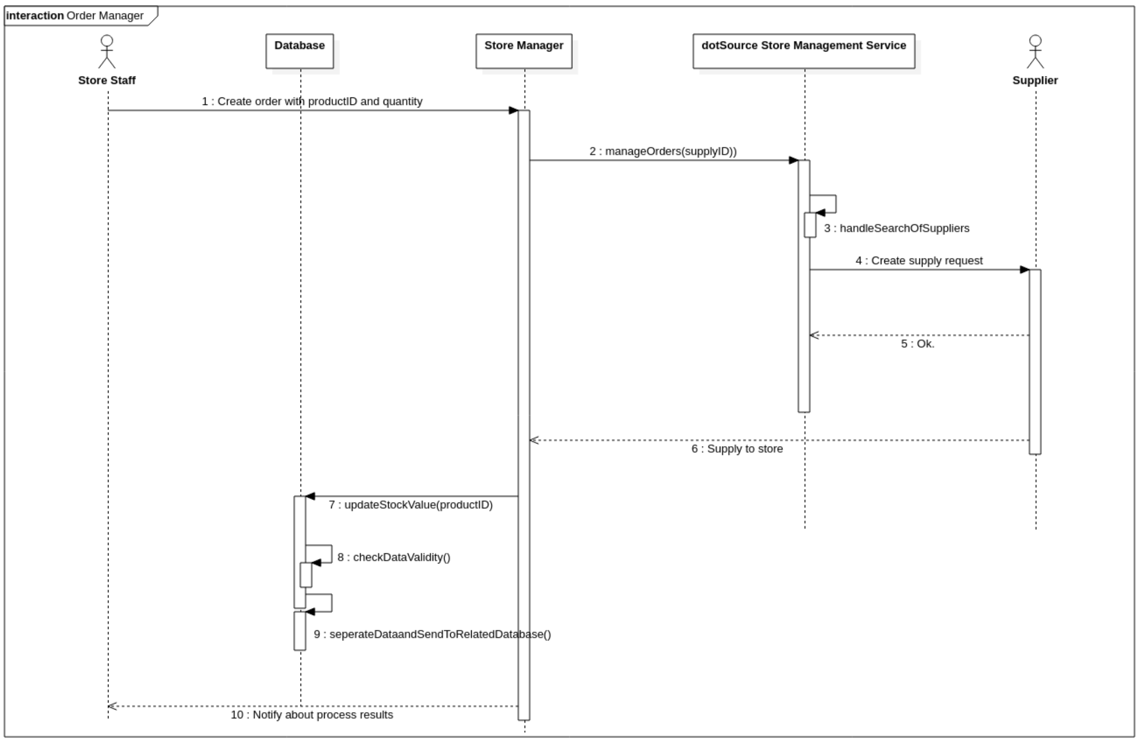

interaction Order Manager
Store Staff
Database
Store Manager
dotSource Store Management Service
Supplier
1 : Create order with productID and quantity
2 : manageOrders(supplyID))
3 : handleSearchOfSuppliers
4 : Create supply request
5 : Ok.
6 : Supply to store
7 : updateStockValue(productID)
8 : checkDataValidity()
9 : seperateDataandSendToRelatedDatabase()
10 : Notify about process results

Figure 4.14: Order Manager Sequence Diagram

System/Service Interface - Customer Transactions

Customer transaction interface is to manage all transactions done between customers and the Amazon Go system. It is supplied by the bank and it is designed to be user interface by the bank. Figure 4.16 shows the proposed interface for regular customers. However, banks are going to supply an API so that SMS can automatically handle it.

Design Rationale for this part is to obtain transactions and check with the account management for any discrepancies. This is actually not a selection but a must since it is the only interface for B2C transactions available in the current technology.

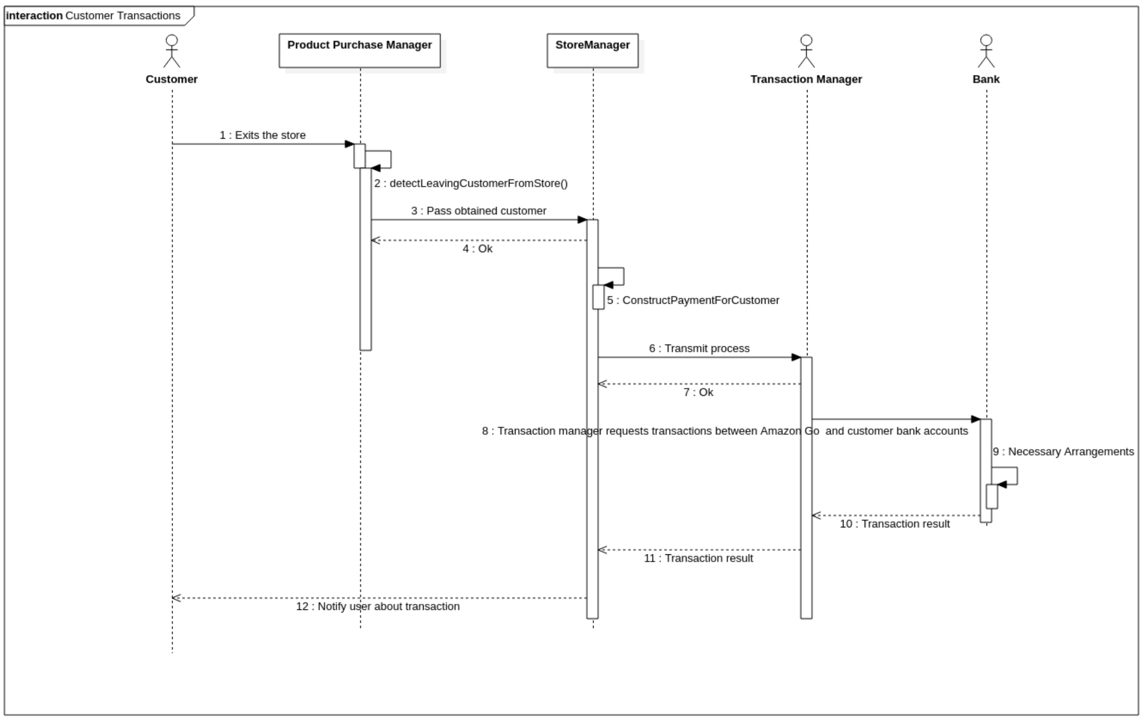
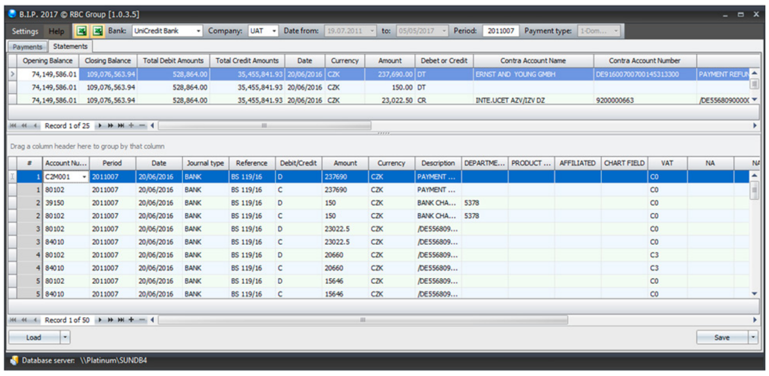

interaction Customer Transactions
Customer
Product Purchase Manager
StoreManager
Transaction Manager
Bank
1 : Exits the store
2 : detectLeavingCustomerFromStore()
3 : Pass obtained customer
4 : Ok
5 : ConstructPaymentForCustomer
6 : Transmit process
7 : Ok
8 : Transaction manager requests transactions between Amazon Go and customer bank accounts
9 : Necessary Arrangements
10 : Transaction result
11 : Transaction result
12 : Notify user about transaction

Figure 4.15: Customer Transactions Sequence Diagram


B.I.P. 2017 © RBC Group [1.0.3.5]
Settings
Help
Payments
Statements
Record 1 of 25
Drag a column header here to group by that column
Record 1 of 50
Load
Save

Bank: UniCredit Bank | Company: UAT | Date from: 19.07.2011 | to: 05/05/2017 | Period: 2011007 | Payment type: 1-Dom...

Opening Balance	Closing Balance	Total Debit Amounts	Total Credit Amounts	Date	Currency	Amount	Debet or Credit	Contra Account Name	Contra Account Number	
74,149,586.01	109,076,563.94	528,864.00	35,455,841.93	20/06/2016	CZK	237,690.00	DT	ERNST AND YOUNG GMBH	DE91600700700145313300	PAYMENT REFUN
74,149,586.01	109,076,563.94	528,864.00	35,455,841.93	20/06/2016	CZK	150.00	DT			
74,149,586.01	109,076,563.94	528,864.00	35,455,841.93	20/06/2016	CZK	23,022.50	CR	INTE.UCET AZV/IZV DZ	9200000663	/DE556809000C

#	Account Nu...	Period	Date	Journal type	Reference	Debit/Credit	Amount	Currency	Description	DEPARTME...	PRODUCT ...	AFFILIATED	CHART FIELD	VAT	NA	NA
1	C2M001	2011007	20/06/2016	BANK	BS 119/16	D	237690	CZK	PAYMENT ...					C0		
1	80102	2011007	20/06/2016	BANK	BS 119/16	C	237690	CZK	PAYMENT ...					C0		
2	39150	2011007	20/06/2016	BANK	BS 119/16	D	150	CZK	BANK CHA...	5378				C0		
2	80102	2011007	20/06/2016	BANK	BS 119/16	C	150	CZK	BANK CHA...	5378				C0		
3	80102	2011007	20/06/2016	BANK	BS 119/16	D	23022.5	CZK	/DE556809...					C0		
3	84010	2011007	20/06/2016	BANK	BS 119/16	C	23022.5	CZK	/DE556809...					C0		
4	80102	2011007	20/06/2016	BANK	BS 119/16	D	20660	CZK	/DE556809...					C3		
4	84010	2011007	20/06/2016	BANK	BS 119/16	C	20660	CZK	/DE556809...					C3		
5	80102	2011007	20/06/2016	BANK	BS 119/16	D	15646	CZK	/DE556809...					C0		
5	84010	2011007	20/06/2016	BANK	BS 119/16	C	15646	CZK	/DE556809...					C0		

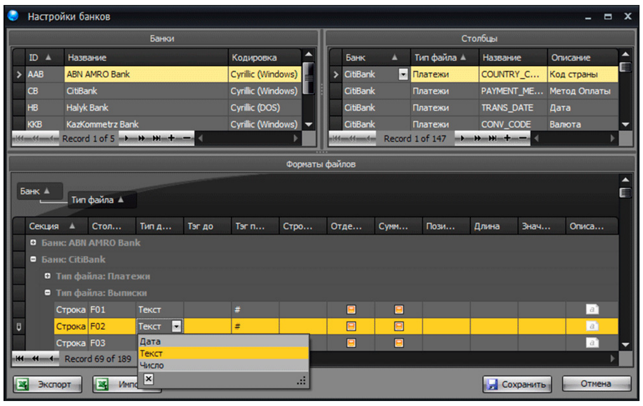
Database server: \\Platinum\SUNDB4

Figure 4.16: Sample Interface for Transactions

System/Service Interface - Procurement Transactions

Procurement from the suppliers are logged into bank databases for their payments. All payments are through banks and banks provide the information to the system via Procurement Transactions interface.

Design Rationale for this interface is that there is no other option for obtaining bank data. Banks, as case in the Customer Transactions, provide an interface that looks like in figure 4.17; however, Amazon Go system uses the API provided. Noe, this is both free and well developed by the banks.



Настройки банков

Банки

ID	Название	Кодировка
AAB	ABN AMRO Bank	Cyrillic (Windows)
CB	CitiBank	Cyrillic (Windows)
HB	Halyk Bank	Cyrillic (DOS)
KKB	KazKommetrz Bank	Cyrillic (Windows)

Record 1 of 5

Столбцы

Банк	Тип файла	Название	Описание
CitiBank	Платежи	COUNTRY_C...	Код страны
CitiBank	Платежи	PAYMENT_ME...	Метод Оплаты
CitiBank	Платежи	TRANS_DATE	Дата
CitiBank	Платежи	CONV_CODE	Валюта

Record 1 of 147

Форматы файлов

Банк

Тип файла

Секция	Стол...	Тип д...	Тэг до	Тэг п...	Стро...	Отде...	Сумм...	Пози...	Длина	Знач...	Описа...
Банк: ABN AMRO Bank											
Банк: CitiBank											
Тип файла: Платежи											
Тип файла: Выписки											
Строка	F01	Текст		#		▣	▣				a
Строка	F02	Текст		#		▣	▣				a
Строка	F03					▣	▣				a

Record 69 of 189

Figure 4.17: Sample Interface for Transactions