# AMAZON GO

## Software Design Description

| Name | Student Id |
|---|---|
| Hilmi Cihan Yıldırım | 2237949 |
| Zeynep Özalp | 2237691 |

# Table of Contents

## List of Figures

# List of Tables

# 1. Introduction

## 1.1 **Purpose of the System**

The purpose of this project is to make the customers' experience of shopping efficient. With the world's most advanced shopping technology, Just Walk Out technology, customers will be able to shop with no lines and no checkout. In order to accomplish this, customers simply use the Amazon Go app, scan the QR Code unique to their account, shop and leave. The Amazon Go system is developed so that the experience of the customers are smooth and enjoyable.

## 1.2 **Scope**

- System will have a user interface where the users can login to their existing accounts or create a new account. After login, users will specify their personal information and credit cards. Using the user interface, users will be able to display their QR Code, discover meals, see their receipts, search for the Amazon Go stores nearby and get help/contact/sign out.

- System will have an interface for the IT staff. IT staff will be able to view system logs. The IT staff will display the system reports, errors and will be notified in terms of emergency.

- System will keep the user information, unique QR Code, card information, receipt address and past receipts for each user via database interface. After scanning the QR Code, system let the user and its dependents pass the gate of the store.

- System will keep track of the user in the store and the products s/he takes via Sensor Fusion. When the user leaves the store, system will charge the account and send the receipt via email.

- The system will not provide online shopping.

## 1.3 **Stakeholders and their concerns**

The amazon go, by its nature is an app that its main concern and purpose is to convert the shopping stuff automated and gather information about the people shopping behaviors. It is an industrial project; therefore, it is analyzing this information to improve shopping strategies better.

**Users:** Users are the people that utilize the offerings of the Amazon Go stores and application. Their concerns are to get informed about how to use Amazon Go application and buy products from stores. The main concern is the reliability and accuracy of the Amazon Go system. Users want to be sure that there is no mistake with products that is added to their cart, their prizes and the receipt info that is sent to their accounts are correct. These metrics shall be in real time, fast and still be precise.

**System Developers:** Systems developers are the ones who are responsible of the development of Amazon Go system. Their concern is having no ambiguity with the software that they are going to develop. Also, they are dependent on SRS and SDD documents.

**Researchers:** Researchers are the people whose concerns are to gather information about Amazon Go. They use the information to analyze the future of buying habits and smart stores. Thus, they are interested in the outcomes of the main use of Amazon Go among people. The research done may lead to development of Amazon Go and smart stores.

**Investors**: Investors are the people whose concerns are made profit from their investments. Therefore, they are concerning the profit of each new amazon go shop. It is also interested in the information which are collected from the users to compose better strategies for advertising.

## 2. References

Dhruv Grewal, Anne L. Roggeveen, Jens Nordfält, The Future of Retailing, Journal of Retailing, Volume 93, Issue 1, 2017, Pages 1-6, ISSN 0022-4359, https://doi.org/10.1016/j.jretai.2016.12.008.

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

Wikipedia, Amazon Go, 5 June 2020, https://en.wikipedia.org/wiki/Amazon_Go.

## 3. Glossary

| Term | Definition |
|---|---|
| AI | Artificial Intelligence |
| ML | Machine Learning |
| User | People who have signed in to the system |
| IT Staff | Staff that are working in the Information Technology department |
| Sensor Fusion | System for combining different data streams from cameras and weight sensors to get more accurate data |
| DBMS | Database Manament System |

| | |
|---|---|
| SKU | Stock Keeping Unit |
| API | Application Programming Interface |
| OS | Operating System |
| Cookie | Small data which are stored in the user's phone. |
| Admin | Person who are responsible of maintanability. |
| IOS | Iphone/Ipad Operating System |
| R&O | Recommendation & Optimization |
| TOA | Team of Associates |
| SKU | Stock-Keeping Unit |

*Table 1: Glossary*

# 4. Architectural Views

## 4.1 Context View

In this viewpoint, context of the system with all actors are defined in general and detailed viewpoints. In the context diagram below, actors and their interactions with the Amazon Go system are showed. All use cases are specified in the use case diagram. The descriptions of all use cases are specified below the use case diagram as tables.

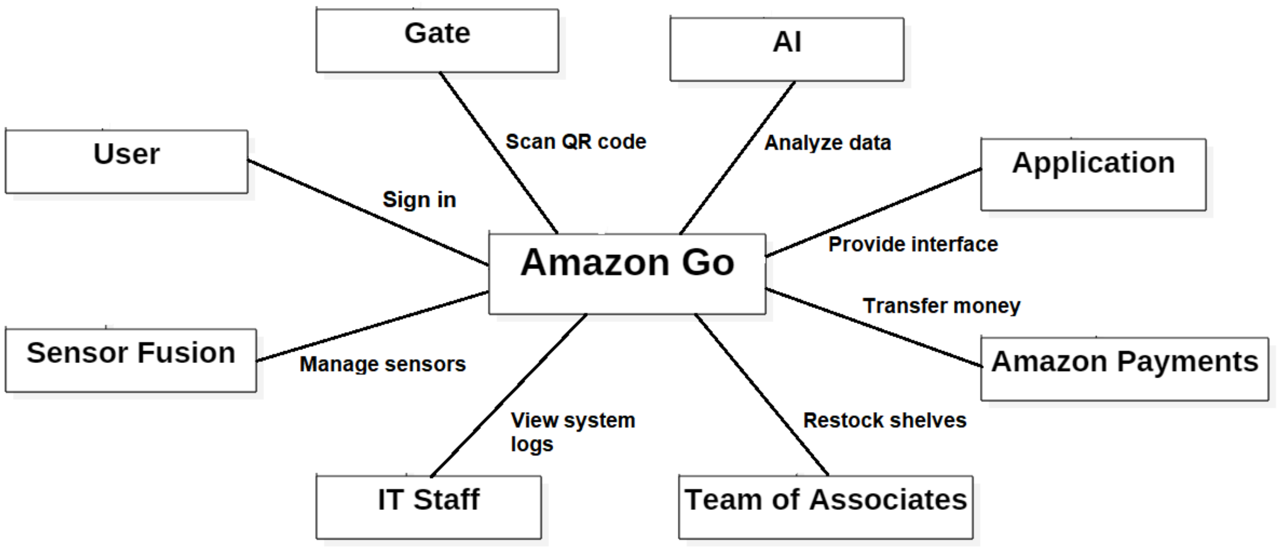


*Figure 1: Context Diagram*

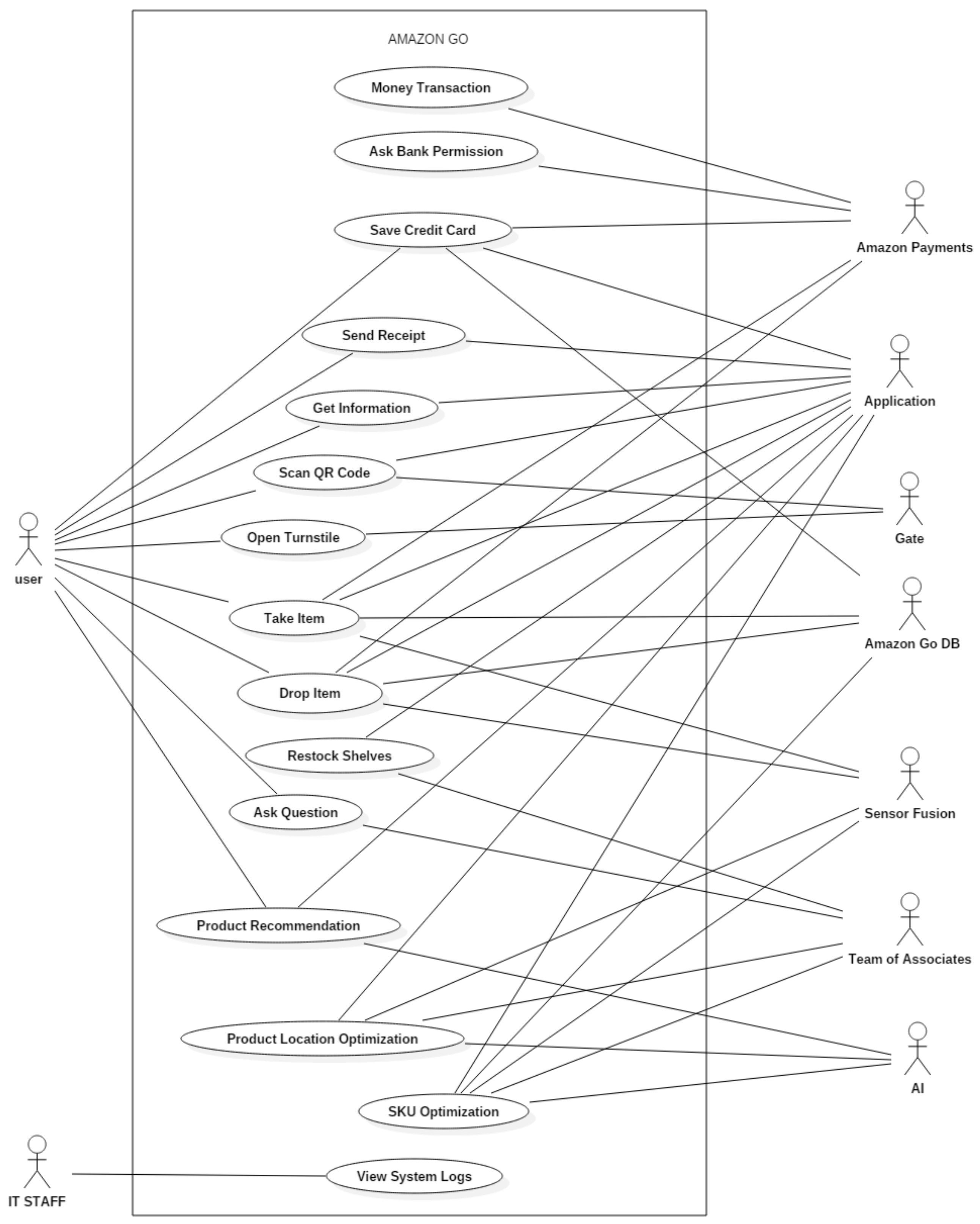


*Figure 2: Use Case Diagram*

| | |
|---|---|
| **Use case name** | Money transaction |
| **Actors** | Amazon Payments |
| **Description** | Amazon Payments system transfers the amount of money that is written on the receipt from the user’s selected credit card |
| **Data** | User’s credit card information, money amount |
| **Preconditions** | User should have an associated account, selected valid credit card, shop from an Amazon Go store, press “pay” button in the receipt interface |
| **Stimulus** | Bank gives permission |
| **Basic Flow** | Step 1 – Bank gives permission<br>Step 2 – Withdraw the money from credit card’s bank account |
| **Alternative Flow** | - |
| **Exception Flow** | Bank do not give the permission. Not enough money on the user’s credit card’s bank account. |
| **Post Conditions** | Notify the user that the receipt is paid and mark the receipt as paid |

*Table 2: Money Transaction*

| | |
|---|---|
| **Use case name** | Ask bank permission |
| **Actors** | Amazon Payments |
| **Description** | Amazon Payments system asks permission from the related bank’s system |
| **Data** | User’s credit card information |
| **Preconditions** | Credit card should be valid i.e. correct card information |
| **Stimulus** | Pressing the “pay” button in the receipt interface |
| **Basic Flow** | Step 1 – User leaves Amazon Go store with at least one product in the user’s virtual cart<br>Step 2 – Default card is used<br>Step 3 – Ask permission from the corresponding bank to withdraw money |
| **Alternative Flow** | Step 2 – User selects another card |
| **Exception Flow** | Bank do not give the permission |
| **Post Conditions** | Notify the user that the receipt is paid and mark the receipt as paid |

*Table 3: Ask Bank Permission*

| | |
|---|---|
| **Use case name** | Save credit card |
| **Actors** | Amazon Payments, Application, Amazon Payments, Amazon Go DB |
| **Description** | Once a credit card is used to pay a receipt, it is saved automatically and can be seen in the credit cards section in the application interface. User does not involve in this use case. |
| **Data** | User's credit card information |
| **Preconditions** | Credit card should be valid i.e. correct card information |
| **Stimulus** | Using a valid credit card to pay a receipt |
| **Basic Flow** | Step 1 – Save the credit card data to DBMS<br>Step 2 – Notify the user that the corresponding credit card is saved |
| **Alternative Flow** | Step 3 - User can delete the card or select it as default card. |
| **Exception Flow** | DBMS fails to save the credit card data |
| **Post Conditions** | Add the credit card in "Payments card" section in the application. |

*Table 4: Save Credit Card*

| | |
|---|---|
| **Use case name** | Take item |
| **Actors** | User, Application, Sensor fusion, Amazon Payments, Amazon Go DB |
| **Description** | User takes a product from the shelves; sensor fusion detects it and the product is added to/removed from the virtual cart of the user |
| **Data** | Weight sensor data, camera data, product id, user data |
| **Preconditions** | User should have an associated account |
| **Stimulus** | Taking the product from the shelves |
| **Basic Flow** | Step 1 - After taking the product, process the weight sensor and camera data in sensor fusion system<br>Step 2 - Send the data from Sensor Fusion to Application<br>Step 3 - Product is added to the user's virtual cart and displayed in the user interface |
| **Alternative Flow** | - |
| **Exception Flow** | The product is taken by some user but hands it to another user |
| **Post Conditions** | Taken product is recorded in the DBMS and total cost is updated |

*Table 5: Take Item*

| | |
|---|---|
| **Use case name** | Drop item |
| **Actors** | User, Application, Sensor fusion, Amazon Payments, Amazon Go DB |
| **Description** | User drops a product from the shelves, sensor fusion detects it and the product is added to/removed from the virtual cart of the user |
| **Data** | Weight sensor data, camera data, product id, user data |
| **Preconditions** | User should have an associated account |
| **Stimulus** | Dropping the product from the shelves |
| **Basic Flow** | Step 1 – After dropping the product, process the weight sensor and camera data in sensor fusion system<br>Step 2 - Send the data from Sensor Fusion to Application<br>Step 3 - Product is removed from the user’s virtual cart and displayed in the user interface |
| **Alternative Flow** | - |
| **Exception Flow** | The user hands over the product to someone but the product will stay on the user’s cart or user puts the product to a different shelf |
| **Post Conditions** | Dropped product is recorded in the DBMS and total cost is updated |

*Table 6: Drop Item*

| | |
|---|---|
| **Use case name** | Send receipt |
| **Actors** | User, Application |
| **Description** | User receives the receipt via email after leaving the store |
| **Data** | User’s email address, user’s receipt address, products in the virtual cart, total cost |
| **Preconditions** | User should leave the store with product(s) |
| **Stimulus** | User walks out of the store |
| **Basic Flow** | Step 1 – Information that user left the shop comes to the application<br>Step 2 – The data is collected from DBMS<br>Step 3 – Receipt is displayed in the application and added to the database |
| **Alternative Flow** | - |
| **Exception Flow** | If the DBMS connection is lost, the IT staff gets a notification |
| **Post Conditions** | Application notifies the user and charges the card |

*Table 7: Send Receipt*

| | |
|---|---|
| **Use case name** | Scan QR code |
| **Actors** | User, Application, Gate |
| **Description** | User gets into the store by scanning the QR code which is displayed in the application at the gate |
| **Data** | QR Code, user data |
| **Preconditions** | User should have an appropriate QR code |
| **Stimulus** | User scans the QR Code with the scanner |
| **Basic Flow** | Step 1 – Scan QR Code<br>Step 2 – Check if the QR code is in the database |
| **Alternative Flow** | - |
| **Exception Flow** | The QR Code may not be readable because of the obstacles in front of the screen (such as fingers) or the angles |
| **Post Conditions** | Send or do not send a permission to the system |

*Table 8: Scan QR Code*

| | |
|---|---|
| **Use case name** | Open turnstile |
| **Actors** | User, Gate |
| **Description** | Open turnstile in front of the store if there is a permission |
| **Data** | Permission |
| **Preconditions** | Permission need to be given |
| **Stimulus** | Confirmation of the permission by the system |
| **Basic Flow** | Step 1 – Open the gate |
| **Alternative Flow** | - |
| **Exception Flow** | Permission is not given, or the gate is broken and do not open |
| **Post Conditions** | User enters the store |

*Table 9: Open Turnstile*

| | |
|---|---|
| **Use case name** | Get information |
| **Actors** | User, Application |
| **Description** | User gets information about the system and application in user interface |
| **Data** | Scripts of questions and answers |
| **Preconditions** | User should have a signed-up account and the application |

| | |
|---|---|
| **Stimulus** | User clicks the More > Help button |
| **Basic Flow** | Step 1 – Display the subtitles in the user interface<br>Step 2 – User clicks the one of subtitles except Contact Us<br>Step 3 – Display the scripts of questions and answers related to that subtitle |
| **Alternative Flow** | - |
| **Exception Flow** | Scripts data cannot be provided |
| **Post Conditions** | User reads the scripts |

*Table 10: Get Information*

| | |
|---|---|
| **Use case name** | Restock Shelves |
| **Actors** | Team of associates, Sensor fusion, AI |
| **Description** | If any shelf is empty , it is detected by cameras and sensors then this shelf is refilled with new products |
| **Data** | Weight sensors data, cameras data, product id, shelf location |
| **Preconditions** | At least one empty shelf, available worker to refill shelves |
| **Stimulus** | At least a shelf became empty |
| **Basic Flow** | Step 1- Empty shelf is notified by fusion system<br>Step 2- Fusion system send an a warning to system<br>Step 3- System detect necessary product and location of shelve<br>Step 4- System controls stocks for this product<br>Step 5- System send an order to team of associates for refill shelve with necessary product<br>Step 6- Team of associated refill shelve with necessary product<br>Step 7- Sensor fusion detect shelve is refilled<br>Step 8- System marks the warning as handled |
| **Alternative Flow** | Step 4-If product is not in stock, system gives an order for this product and wait until it is available in stock again |
| **Exception Flow** | If product no longer available anywhere, system notifies AI |
| **Post Conditions** | Shelf is refilled by products |

*Table 11: Restock Shelves*

| | |
|---|---|
| **Use case name** | SKU Optimization |
| **Actors** | Application, Sensor Fusion, AI, Team of Associates |

| | |
|---|---|
| **Description** | AI keeps data that is sent from sensors and analyze it to improve product variety according to product demand |
| **Data** | Camera data, product id |
| **Preconditions** | System sends a receipt to improve efficiency of product variety |
| **Stimulus** | Product variety is not appropriate for customers |
| **Basic Flow** | Step 1- System sends a request to improve product variety<br>Step 2-AI analyzes data which is sent from sensor fusuion system<br>Step 3-AI detects necessary and unnecessary products<br>Step 4-AI sends this data to system<br>Step 5-System rearrangse product variety list |
| **Alternative Flow** | Step 5-If there is no change  in product variety, system do not reaarange product variety |
| **Exception Flow** | - |
| **Post Conditions** | Products are arranged according  to new list |

*Table 12: SKU Optimization*

| | |
|---|---|
| **Use case name** | Product Location Optimization |
| **Actors** | Application, Sensor Fusion, AI, Team of Associates |
| **Description** | AI keeps data sent from sensors and analyze it to improve products' locations |
| **Data** | Camera data, product id |
| **Preconditions** | System send an receipt improve product location |
| **Stimulus** | Product location is not appropriate for customers |
| **Basic Flow** | Step 1- Application sends a request to improve product locations<br>Step 2-AI analyzes data which sent from fusuion systems<br>Step 3-AI detects appropriate locations for each item<br>Step 4-AI sends this data to system<br>Step 5-System rearranges products locations according to this data |
| **Alternative Flow** | Step 5-If there is no change in product locations, system do not reaarange them |
| **Exception Flow** | - |
| **Post Conditions** | Products are arranged according to new data |

*Table 13: Product Location Optimization*

| | |
|---|---|
| **Use case name** | View System Logs |
| **Actors** | IT staff |
| **Description** | IT staff can list the system logs of Amazon Go and logs can be arranged by their date of formation and where the events take place. These logs include technical details of Amazon Go application. |
| **Data** | System logs |
| **Preconditions** | IT staff must be assigned and authorized to read system logs. |
| **Stimulus** | IT staff send a request to system for reaching system logs. |
| **Basic Flow** | Step 1-System logs request is sent to system.<br>Step 2-System gives permission to IT staff.<br>Step 3-Logs became visible in IT staff interface |
| **Alternative Flow** | - |
| **Exception Flow** | If connection or logs are lost, system notifies IT staff. |
| **Post Conditions** | System logs became visible on IT Staff's interface. |

*Table 14: View System Logs*

| | |
|---|---|
| **Use case name** | Ask Question |
| **Actors** | User, Team of Associates |
| **Description** | Users can ask question to Team of Associates about the store, application, etc. |
| **Data** | - |
| **Preconditions** | At least one non-busy worker |
| **Stimulus** | - |
| **Basic Flow** | Step 1-User has a question<br>Step 2-User finds an appropriate worker<br>Step 3-User asks question to worker<br>Step 4-Worker communicates with the user and answers the question |
| **Alternative Flow** | Step 3-If all workers are busy user waits someone to be avaliable |
| **Exception Flow** | If no worker is available in the store |
| **Post Conditions** | User's question is answered |

*Table 15: Ask Question*

| | |
|---|---|
| **Use case name** | Product Recommendation |
| **Actors** | User, Application, AI |
| **Description** | Application recommends users specific products. Recommended products are user specific and selected by analyzing the user's past data by AI |
| **Data** | User's receipt data, i.e.  the products that are bought in the last month by the user |
| **Preconditions** | User should have a signed-up account and the application |
| **Stimulus** | User opens the application |
| **Basic Flow** | Step 1 – Analyze user's receipt data using ML<br>Step 2 – Find the best recommendations for the user<br>Step 3 – Display recommended products in the "Discover" section in the application |
| **Alternative Flow** | Step 3 – Send a notification to user one time per week |
| **Exception Flow** | If user is new i.e. does not have any receipt data, randomize the recommendations |
| **Post Conditions** | Recommended products are displayed |

*Table 16: Product Recommendation*

## 4.2 Composition View

In this viewpoint, components of the system such as subsystems and the functionalities of these components will be shown from a top-level point of view. Further information about the components is explained in their corresponding sections.

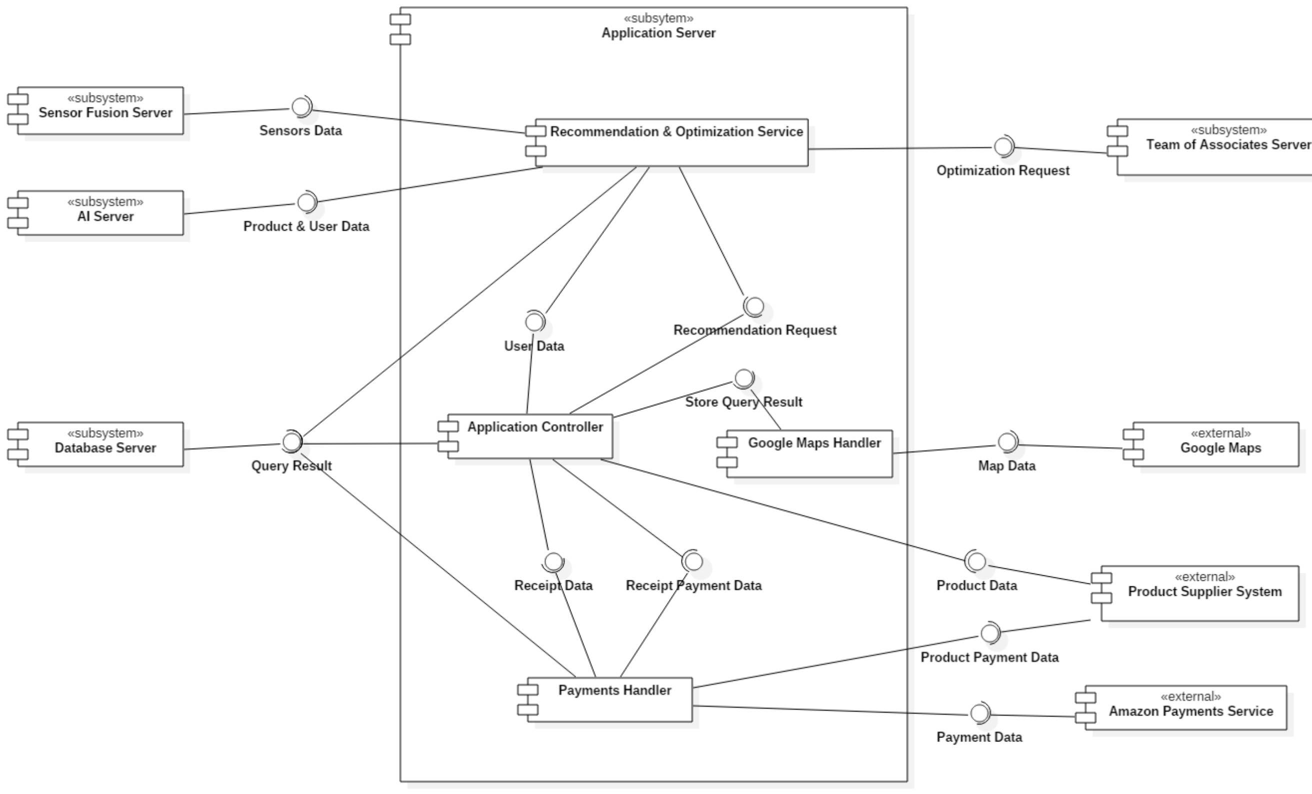


*Figure 3: Component Diagram*

**Design Rationale:**

- Application Server subsytem has different components to communicate with external systems or subsystems.
- All of Amazon Go's application functionalities, namely the different user interfaces employed in different pages, modifying the database upon user input and querying the database in order to display data, are done by the component named Application Controller. The purpose for having just one component to handle all of the user interfaces was because they have similar functionalities such as displaying data, taking input and reaching the database.
- After querying the database to acquire needed data, Application Controller sends data to Google Maps Handler component to be transformed into a suitable format and sent to Google Maps, allowing the data to be shown on map.
- Google Maps is an external system and used to visualize stores' locations. Google Maps requires the map data coming from Google Maps Handler.
- Payments Handler component is required to compose the data from Database Server and Application Controller and deliver the processed data to Amazon Payments Service subsystem.
- Amazon Payments Service is a subsystem of Amazon e-commerce company that is used for money withdraws etc.
- Recommendation & Optimization Service component is used to recommend products to users and optimize the product locations inside stores. For recommendation purpose, Application Controller request user data from Database Server and this data sent to Recommendation & Optimization Service component. Every time when Application Controller display a recommendation, it sends a request to Recommendation & Optimization Service component.

For optimzation purpose, sensors data coming from SensorFusion Server and product & user data coming from AI Server are received. Team of Associates Server requests optimizations specific to store.

- Sensor Fusion Server is used to store sensor data and fusion them properly.
- AI Server is used to store product & user data. It provides product-user relationship using ML.
- Team of Associates worker can see the optimization needed to be performed using Team of Associates Server.

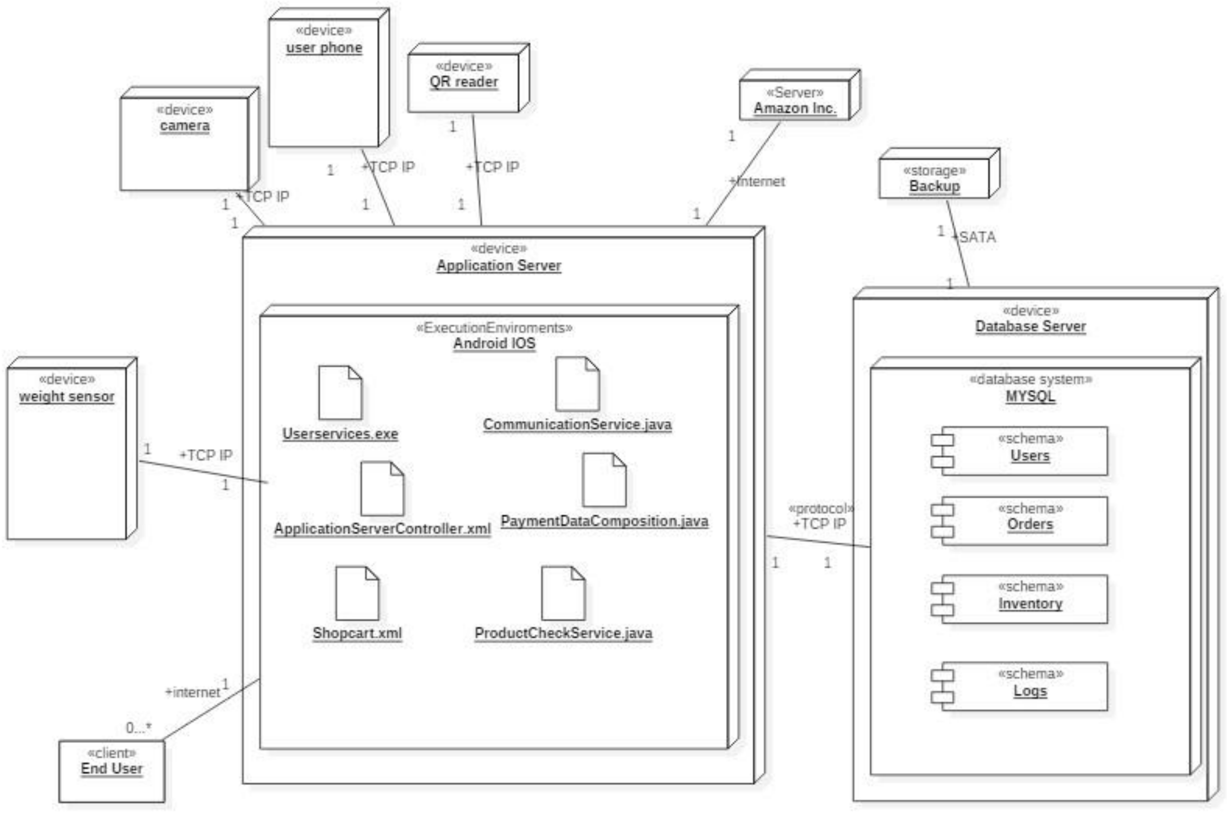


*Figure 4: Deployment Diagram*

**Design Rationale:**

- Application server devoloped with java and working with oracle for android.
- Application server devoloped with swift and working with oracle for android.
- Database server and application server communicate with each other over TCP IP protocol.
- Database system keep informations which are necessary for application like users name, id, passwords etc. It is also keeps informations like orders and inventories.
- The user communicate with application with internet.
- All system components and database communicate with application with a protocol which is TCP/IP because of security and protection issues.
- User and application communication with internet has end to end encryption to maintain security.
- Users data like camera records etc. are encrypted according to personal information related laws.

## 4.3 **Information View**

In this view, we are showing the organization and the services and their relationships among each other. In addition, we also showing the operations. The effects of operations on the data explained in terms of their effect type like create, read, update and delete.

### 4.3.1 Interfaces

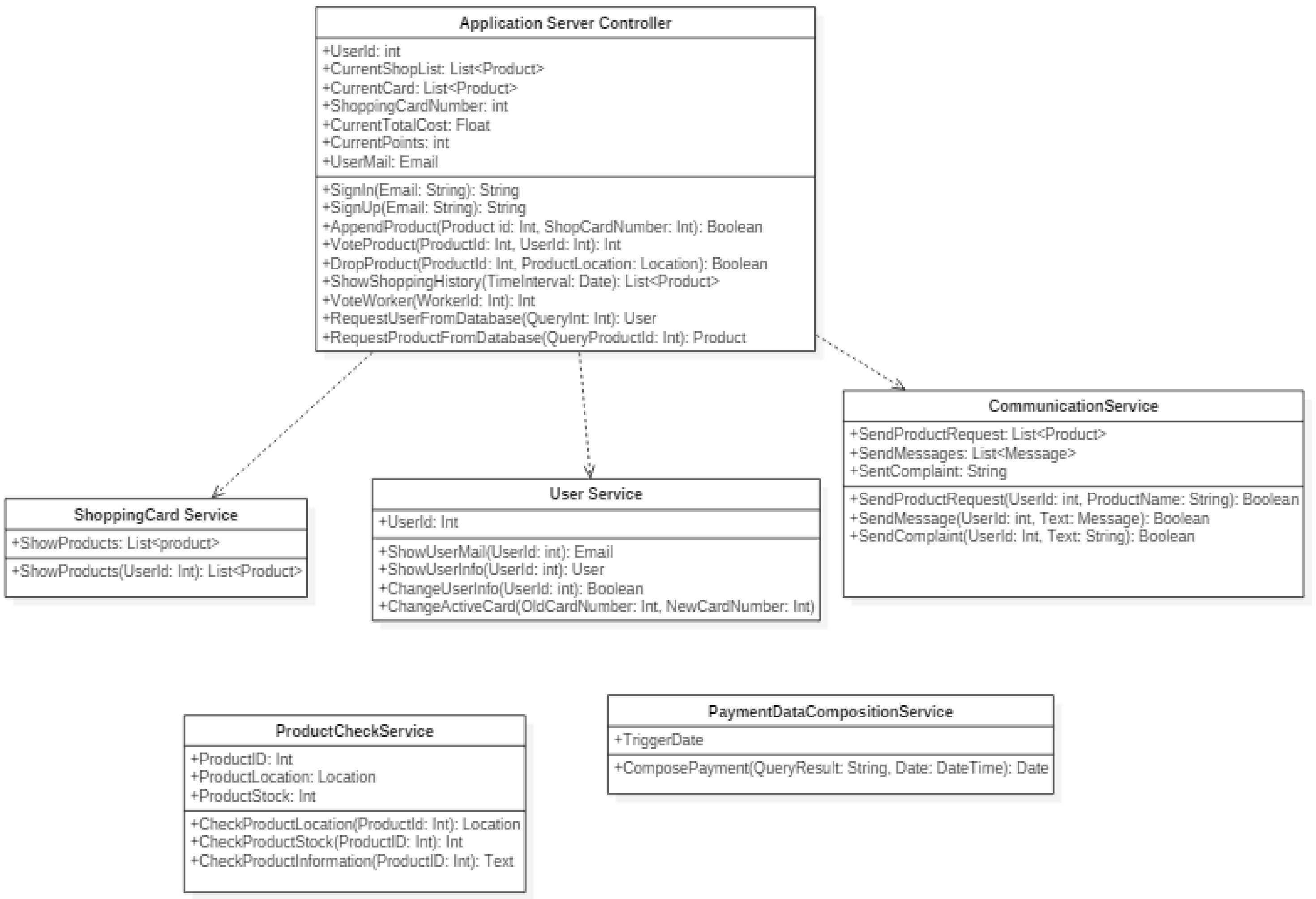

Application Server Controller
+UserId: int
+CurrentShopList: List<Product>
+CurrentCard: List<Product>
+ShoppingCardNumber: int
+CurrentTotalCost: Float
+CurrentPoints: int
+UserMail: Email
+SignIn(Email: String): String
+SignUp(Email: String): String
+AppendProduct(Product id: Int, ShopCardNumber: Int): Boolean
+VoteProduct(ProductId: Int, UserId: Int): Int
+DropProduct(ProductId: Int, ProductLocation: Location): Boolean
+ShowShoppingHistory(TimeInterval: Date): List<Product>
+VoteWorker(WorkerId: Int): Int
+RequestUserFromDatabase(QueryInt: Int): User
+RequestProductFromDatabase(QueryProductId: Int): Product
CommunicationService
+SendProductRequest: List<Product>
+SendMessages: List<Message>
+SentComplaint: String
+SendProductRequest(UserId: int, ProductName: String): Boolean
+SendMessage(UserId: int, Text: Message): Boolean
+SendComplaint(UserId: Int, Text: String): Boolean
ShoppingCard Service
+ShowProducts: List<product>
+ShowProducts(UserId: Int): List<Product>
User Service
+UserId: Int
+ShowUserMail(UserId: int): Email
+ShowUserInfo(UserId: int): User
+ChangeUserInfo(UserId: int): Boolean
+ChangeActiveCard(OldCardNumber: Int, NewCardNumber: Int)
ProductCheckService
+ProductID: Int
+ProductLocation: Location
+ProductStock: Int
+CheckProductLocation(ProductId: Int): Location
+CheckProductStock(ProductID: Int): Int
+CheckProductInformation(ProductID: Int): Text
PaymentDataCompositionService
+TriggerDate
+ComposePayment(QueryResult: String, Date: DateTime): Date

*Figure 5: Interface class Diagram*

| **Operation** | **Description** |
|---|---|
| SignIn | This operation give permission the system to user can connect. |
| SignUp | This operation allows the users to register themselves to the system. |

| | |
|---|---|
| AppendProduct | This operation appends a product to user shop card with product id. |
| VoteProduct | This operation gives a right to user to show their opinion about product. |
| DropProduct | After this operation product cost is decreased from user shop card. |
| ShowShoppingHistory | This operation take history of shopping a user with user id and show it on the user screen. |
| VoteWorker | This operation gives a chance to user to vote workers like team of associates over the application. |
| RequesUserFromDatabase | This operation send request to database for a specific user with user id and take their information from the database. |
| RequestProductFromDatabase | This operation sends a query to database with unique product id and take information about this product from the database. |
| ShowProducts | This operation sends a user id to database and take shopping card current product in that user after that show this on the screen of user. |
| ShowUserMail | Users see their active mail in the application |
| ShowUserInfo | This operation called when the user triggers my account screen in app it sends a request to database after that database gives information and this operation send this information to the screen. |
| ChangeUserInfo | This operation takes old information which is associated user id and new information and change them in the database. |
| ChangeActiveCard | This operation takes old card number and new card number and delete old one from database append new one to database. |
| SendProductRequest | This function takes product requests from user, worker or team of associates then send request to AI. |

| | |
|---|---|
| SendMessage | This operation triggered after user click chat screen and takes user id with their text message transmit this message to workers. |
| SendComplaint | This operation only for complaints it takes text message complaints from user transmit this info to AI and workers. |
| CheckProductLocation | This operation check product location while communicating with camera, AI and shelve weight sensors to validate location of product. |
| CheckProductStock | This operation check product location while communicating with camera, AI and shelve weight sensors to validate number of products it is also take info from database then return number of items in total. |
| CheckProductInformation | This operation takes information from database and check with camera information. |
| ComposePayment | This operation collects the payment data from the database and application then compose the payment in amazon services. |

*Table 15: Operation descriptions*

| **Operation** | **Inputs** | **Outputs** | **Exceptions** |
|---|---|---|---|

|  |  |  |  |
|---|---|---|---|
| SignIn | - Email | Session id given by the system as a cookie | - Given Email is not recognized<br>- Database connection error occurs |
| SıgnUp | - Email | Successful text returned if operation succeed | - Email is not valid<br>- There is a user with this email |
| AppendProduct | - Product id<br>- ShopCardNumber | True if operation was successful, otherwise false | - Product is not valid<br>-Card number not recognized<br>-There is not valid card |
| VoteProduct | - ProductId<br>- UserId | Return vote | - Vote is empty<br>- There is not valid vote |
| DropProduct | - ProductId<br>- ProductLocation | True if successfully dropped the item false otherwise | - Product id not recognized<br>- It is not valid location for this product |

| | | | |
|---|---|---|---|
| ShowShoppingHistory | - TimeInterval | List of products | - There is no product to listed<br>- There in not valid interval<br>- Database crash |
| VoteWorker | - WorkerId | Vote which is given the worker | - The worker does not exist<br>- Worker cannot vote |
| RequestUserFromDatabase | - OueryInt | Worker information if exist error otherwise | - There is not a valid Query<br>- There is no worker with this id |
| RequestProducFromDatabase | - QueryProductid | If valid product id returns information of this product<br>Else<br>Return error there is no product with this id | - Product information does not exist<br>- Product id changed |
| ShowProducts | - Userid | List of products belong the user | - User does not exist<br>- There is no product for this user |

| | | | |
|---|---|---|---|
| ShowUserMail | - UserId | Mail of this user if valid null otherwise | - The mail is not valid<br>- User has changed mail |
| ShowUserInfo | - UserId | Information about user | - The user cannot find<br>- Information is not written the database |
| ChangeUserInfo | - UserId | True if change is successful false otherwise | - The changes are not valid<br>- Mail is not valid |
| ChangeActiveCard | - OldCardNumber<br>- NewCardNumber | True is change occurred false otherwise | - New number is not valid |
| SendProductRequest | - UserID<br>- ProductName | True is request valid and recognized, false otherwise | - There is no product with this name<br>- User id is not valid<br>- Requested before |

| | | | |
|---|---|---|---|
| SendMessage | - UserId<br>- TextMessage | True if message forwarded false otherwise | - Text include invalid characters<br>- User is not valid |
| SendComplaint | - UserId<br>- Text | True is complaint forwarded false otherwise | - Text include invalid characters<br>- User is not valid |
| CheckProductLocation | - ProductId | Location of product if founded Null otherwise | - More than one location for this product<br>- Product is not recognized |
| CheckProductStock | - ProductId | Number of stocks for this id | - There is no product info founded in database<br>- Product not recognized |
| CheckProductInformation | - ProductId | Information in text format about this product | - No information exists in database |

|  |  |  |  |
|---|---|---|---|
| ComposePayment | - QueryResult<br>- Date | Date of payment if successful payment composed, invalid date if unsuccessful | - The date is not valid<br>- Query does not exist in server |

Table 16: Operation design

**Design Rationale:**

- Application Server controller is the main controller which is responsible for user interactions with database and application.
- Embedded systems like camera, weight sensors communicate application server and it forward them to AI. However, this communication made over TCP IP protocol to protect information security.
- Data operations support the data types of ".xls, .jason, .csv".
- Drop product and Append product operations are asynchronous operations.
- ComposePayment operation need to operate after consumer leaved the market asynchronously.
- Application server need to protect data before sent to database so it has to encrypt information.

### 4.3.2 Database Operations

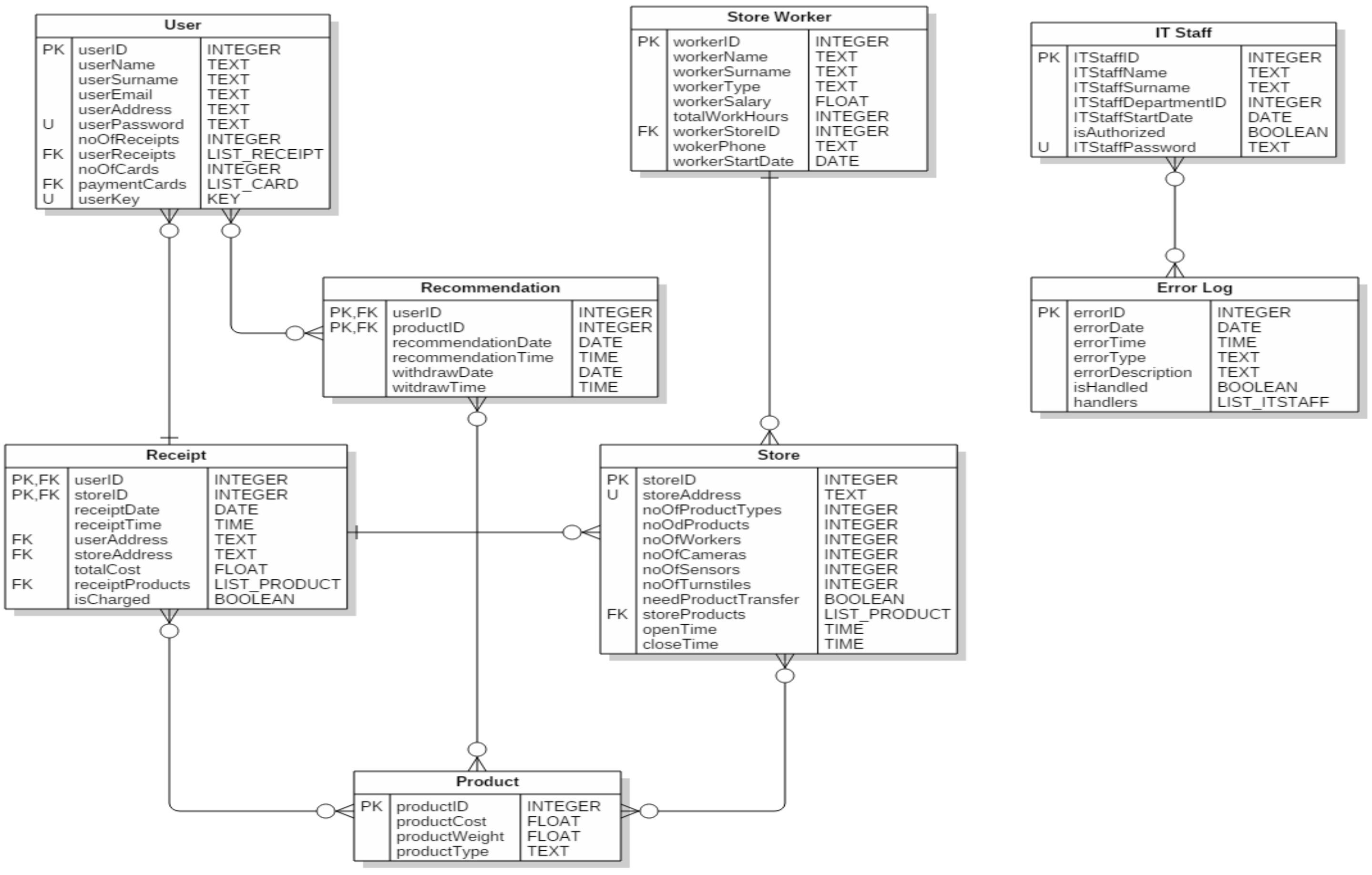


*Figure 6: Database Class Diagram*

| **Operation** | **CRUD Operations** |
|---|---|
| SignIn | Create :<br>Read : User<br>Update :<br>Delete : |
| SignUp | Create : User<br>Read :<br>Update :<br>Delete : |
| AppendProduct | Create : Product<br>Read : Store<br>Update :  Product<br>Delete : |
| VoteProduct | Create : IT Staff<br>Read : User<br>Update :  IT Staff<br>Delete : |

| | |
|---|---|
| DropProduct | Create :<br>Read : Store<br>Update :  Store<br>Delete : Product |
| ShowShoppingHistory | Create : Receipt<br>Read : User<br>Update :<br>Delete : |
| VoteWorker | Create :<br>Read : Store Worker<br>Update :  Store Worker<br>Delete : |
| RequesUserFromDatabase | Create :  User<br>Read : User<br>Update :<br>Delete : |
| RequestProductFromDatabase | Create : Product<br>Read : Product<br>Update :<br>Delete : |
| ShowProducts | Create :<br>Read : Product<br>Update :<br>Delete : |
| ShowUserMail | Create :<br>Read : User<br>Update :<br>Delete : |
| ShowUserInfo | Create :<br>Read : User<br>Update :<br>Delete : |
| ChangeUserInfo | Create :<br>Read : User<br>Update : User<br>Delete : |
| ChangeActiveCard | Create :<br>Read : User<br>Update :  User<br>Delete : |

| | |
|---|---|
| SendProductRequest | Create :<br>Read : Store<br>Update : Store<br>Delete : |
| CheckProductLocation | Create :<br>Read : Product, Store<br>Update :<br>Delete : |
| CheckProductStock | Create :<br>Read : Store, Product<br>Update :<br>Delete : |
| CheckProductInformation | Create :<br>Read : Product, Store<br>Update :<br>Delete : |
| ComposePayment | Create : Receipt<br>Read : User<br>Update :  Receipt<br>Delete : |

**Design Rationale:**

- Every single data which is created recorded to database and analyzed by ML.
- MySQL and Oracle using in database together.
- Python using to analyze and visualize data.
- After every Compose mail operation an email is sent to user.
- Every Show info operation create a request to database.

## 4.4 Interface View

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

### 4.4.1 Internal Interfaces

***The interface between Database Server and Application Controller:***

Application Controller will be responsible for storing the needed data in the Database Server for preserving the integrity of the system. Application Controller will query the database by sending a query string in SQL. When database server receives the query it will execute the query string in MySQL. In case of a MySQL error, database server component will report the error to the Application Controller component.

**Design Rationale:**

- Web Server Controller will be responsible for composing the corresponding SQL queries for each operation.
- Database Server component is responsible for storing user, receipt and product data and ensure the integrity of them.

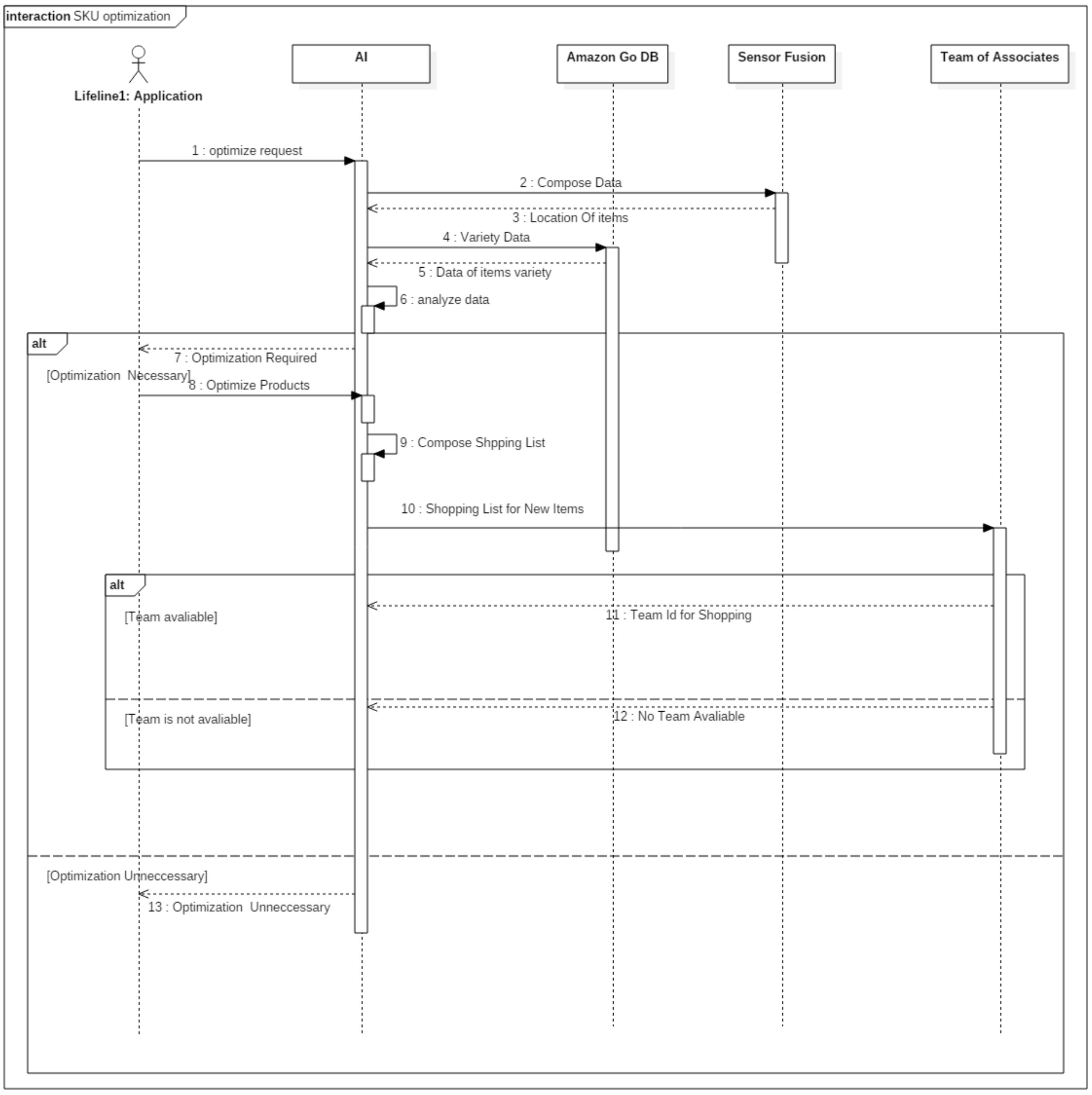


*Figure 7: SKU Optimization Sequence Diagram*

***The interface between Payments Handler and Application Controller:***

Payments Handler component will query the Application Controller component for receipt data of a user. Payments Handler component is necessary for receipt payments. It is a bridge to Amazon Payments Service component.

**Design Rationale:**

- When Application Controller receives the query it will execute the query string in MySQL.
- In case of a MySQL error, Application Controller will report the error to the Payments Handler component.

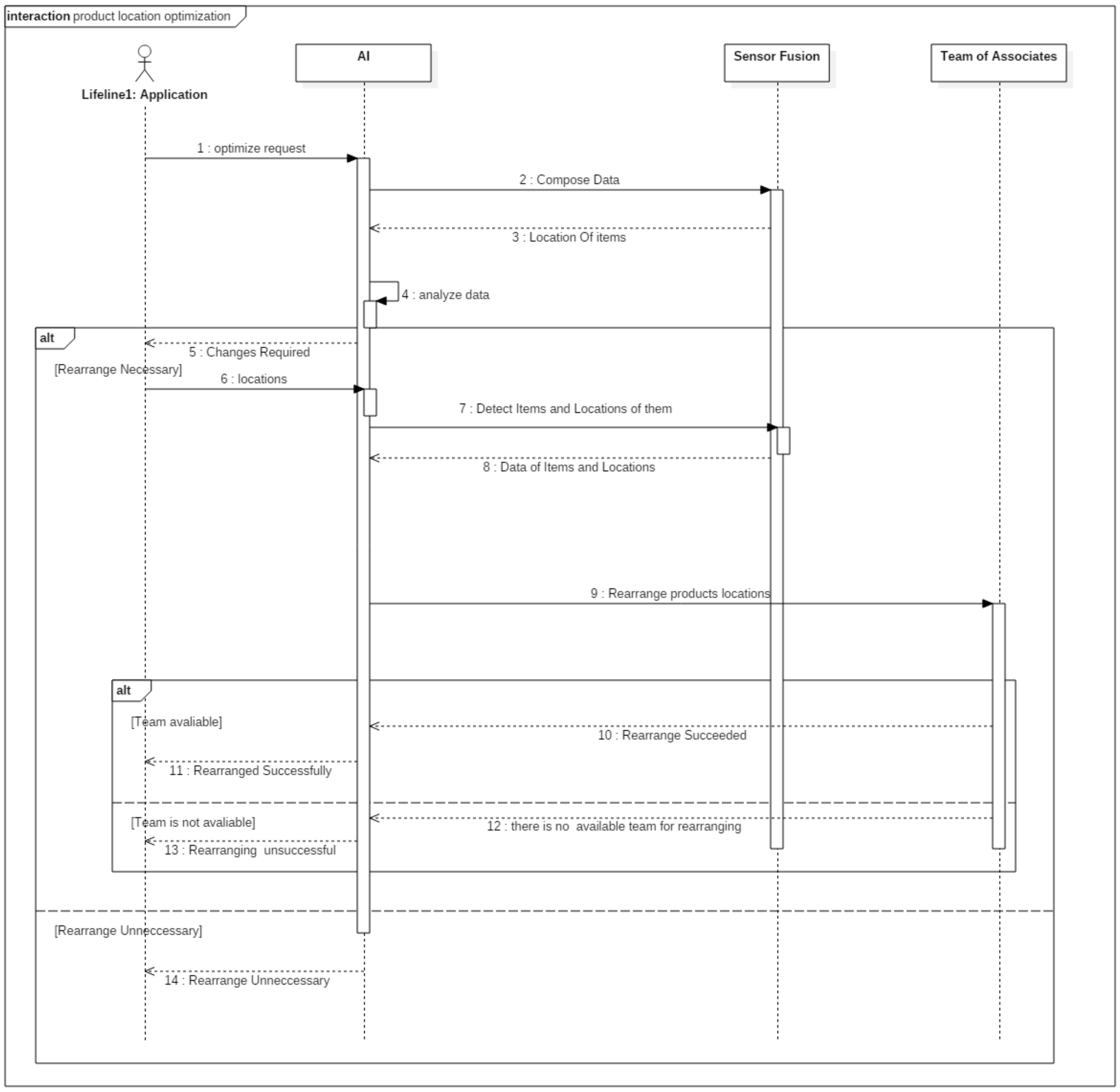


*Figure 8: Product Location Optimization Sequence Diagram*

***The interface between Google Maps Handler and Application Controller:***

Google Maps Handler component is used to communicate with the external Google Maps component. This component converts the location data of stores to the appropriate format for Google Maps API.

**Design Rationale:**

- The direct query result cannot be understood by Google Maps API; therefore, the Application Controller does not handle with its conversion and uses Google Maps Handler's interface to communicate with Google Maps. Application Controller will send the store's location data to Google Maps Handler component.
- Application Controller will display the location of the nearby stores on Google Maps.

***The interface between R&O Service and TOA Server:***

Team of Associates Server will query the Recommendation & Optimization Service and request a product optimization.

**Design Rationale:**

- In this interface, all the product information among the store will be displayed.
- TOA worker can see the optimization that can be made in the store by notifications.

### 4.4.2 External Interfaces

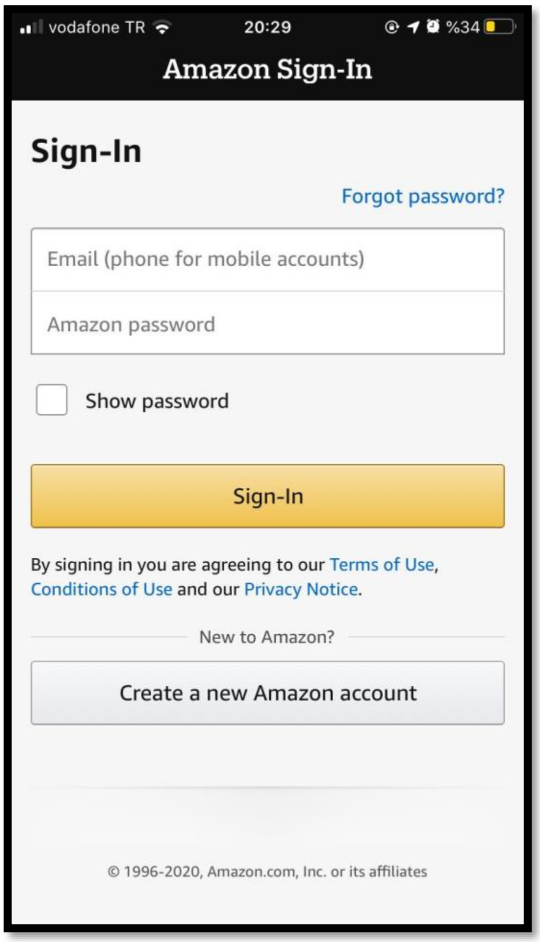
#### 4.4.2.2 User Interfaces

***Registration Interface:***

This interface serves as a guide for nonmember users to become members. Non-member users enter name, surname and accept some contracts. Very organized and simple interface. It consists of very few boxes to enter text inputs. All the input data are text inputs and click inputs. After the progress finished application direct the user to payment card interface.

**Design Rationale:**

- Any invalid input gives error message like special characters or missing characters like "@" for email box.
- All fields must be filled. Any empty area is not allowed otherwise sign in will fail.
- There is no time limit for this interface, so timing is not crucial. However, when app close or refreshing attempt cause data to be lost.


vodafone TR 20:29 %34

Amazon Sign-In

Sign-In

Forgot password?

Email (phone for mobile accounts)

Amazon password

Show password

Sign-In

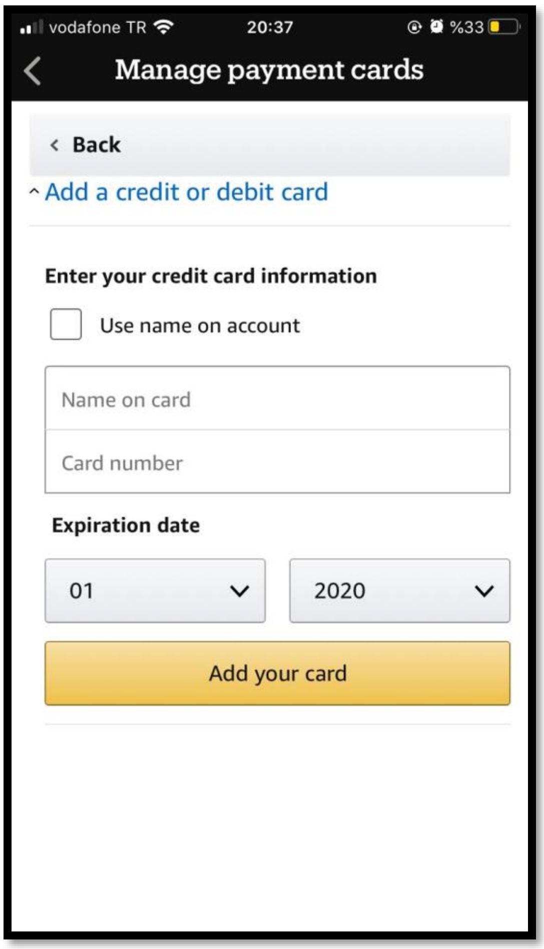
By signing in you are agreeing to our Terms of Use, Conditions of Use and our Privacy Notice.

New to Amazon?

Create a new Amazon account

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

*Figure 9: Sign-in Interface*


vodafone TR 20:37 %33

Manage payment cards

Back

Add a credit or debit card

Enter your credit card information

Use name on account

Name on card

Card number

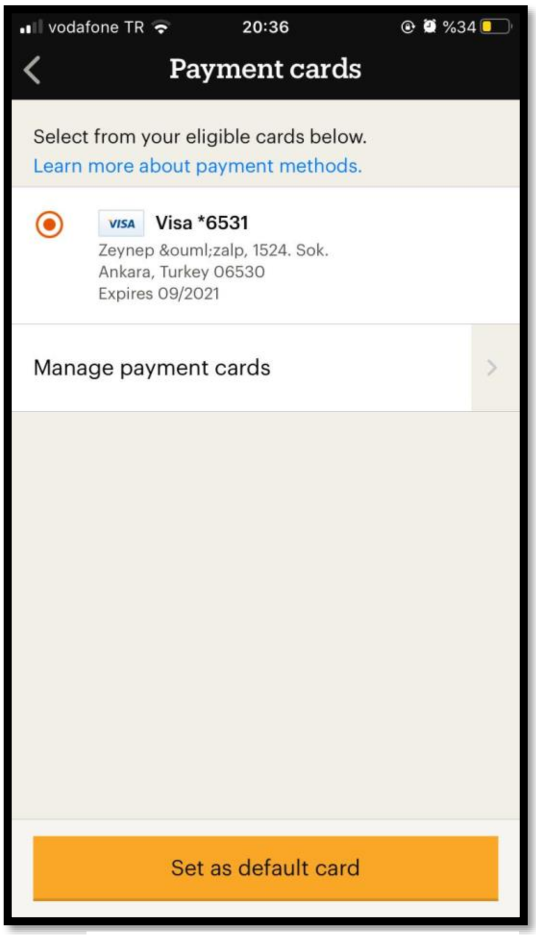
Expiration date

01

2020

Add your card

*Figure 10: Manage Payments Cards*


vodafone TR 20:36 %34

Payment cards

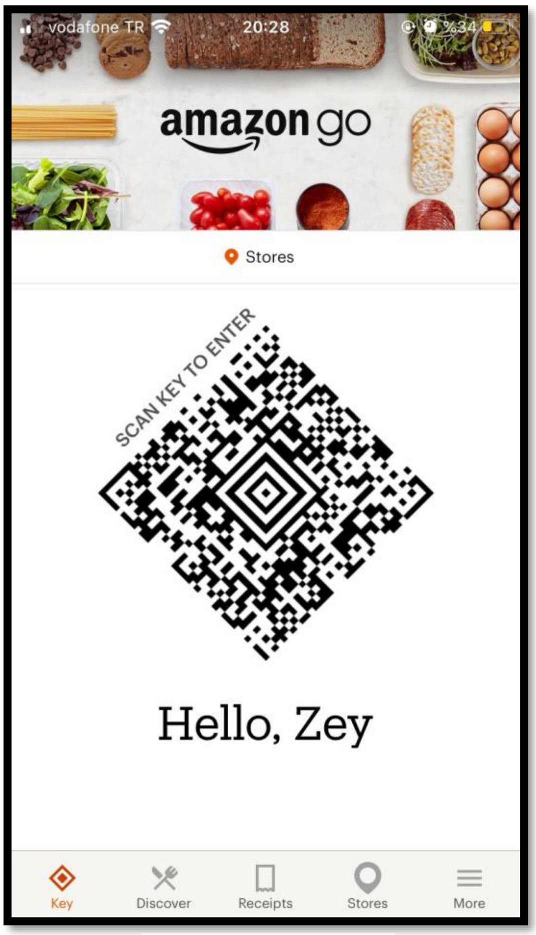
Select from your eligible cards below.
Learn more about payment methods.

VISA Visa *6531
Zeynep &ouml;zalp, 1524. Sok.
Ankara, Turkey 06530
Expires 09/2021

Manage payment cards

Set as default card

*Figure 11: Payment Cards Interface*


vodafone TR 20:28 %34

amazon go

Stores

SCAN KEY TO ENTER

Hello, Zey

Key Discover Receipts Stores More

*Figure 12: User Interface*

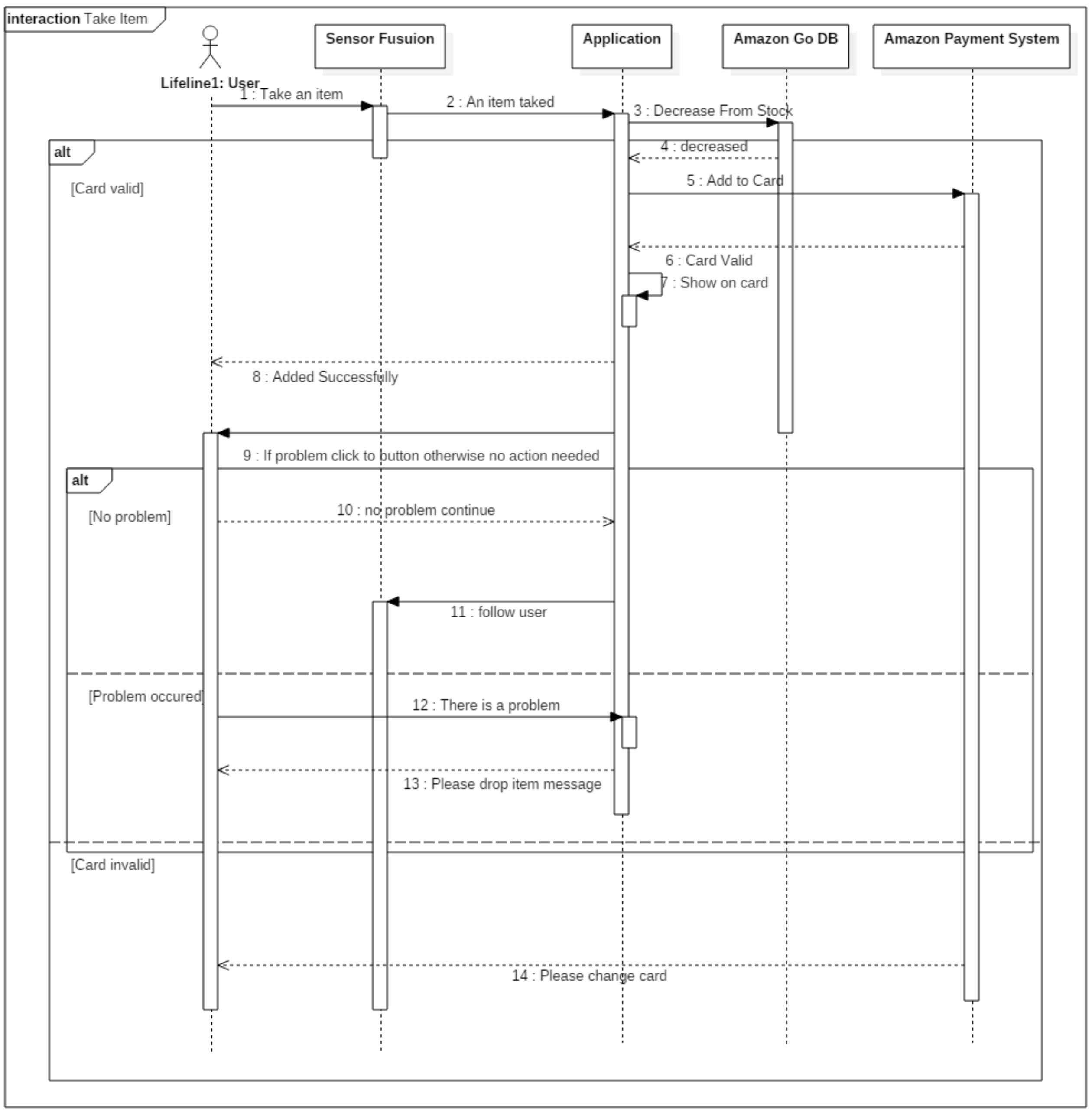


*Figure 13: Take Item Sequence Diagram*

***Payment Card Interface:***

This interface is to assign a card to user to charge them for the products they buy. Users enter card number, card name etc. Very organized and simple interface. It consists of very few boxes to enter text inputs. After the progress finished application direct the user to user interface.

**Design Rationale:**

- Credit card data is saved and sent to database and checked for validity. If card number is invalid or if there is any inconsistency with card number and name, application gives error messages like inconsistent card number or inconsistent number name coupling.
- All data must consist of alphanumerical inputs.
- There is no time limit for this interface, so timing is not crucial. However, when app close or

refreshing attempt cause data to be lost.

***User Interface:***

The purpose of this interface is to enable users to see some data like their key at the gate, receipts etc. and communicate with workers to get information.

**Design Rationale:**

- User inputs consist of clicks and some texts and this input goes application and application triggers related interface.
- Users can receive a warning message if there is no data about clicked area. For example, if user did not do any shopping there is no data in receipts.
- There is no time limit for this interface, so timing is not crucial.

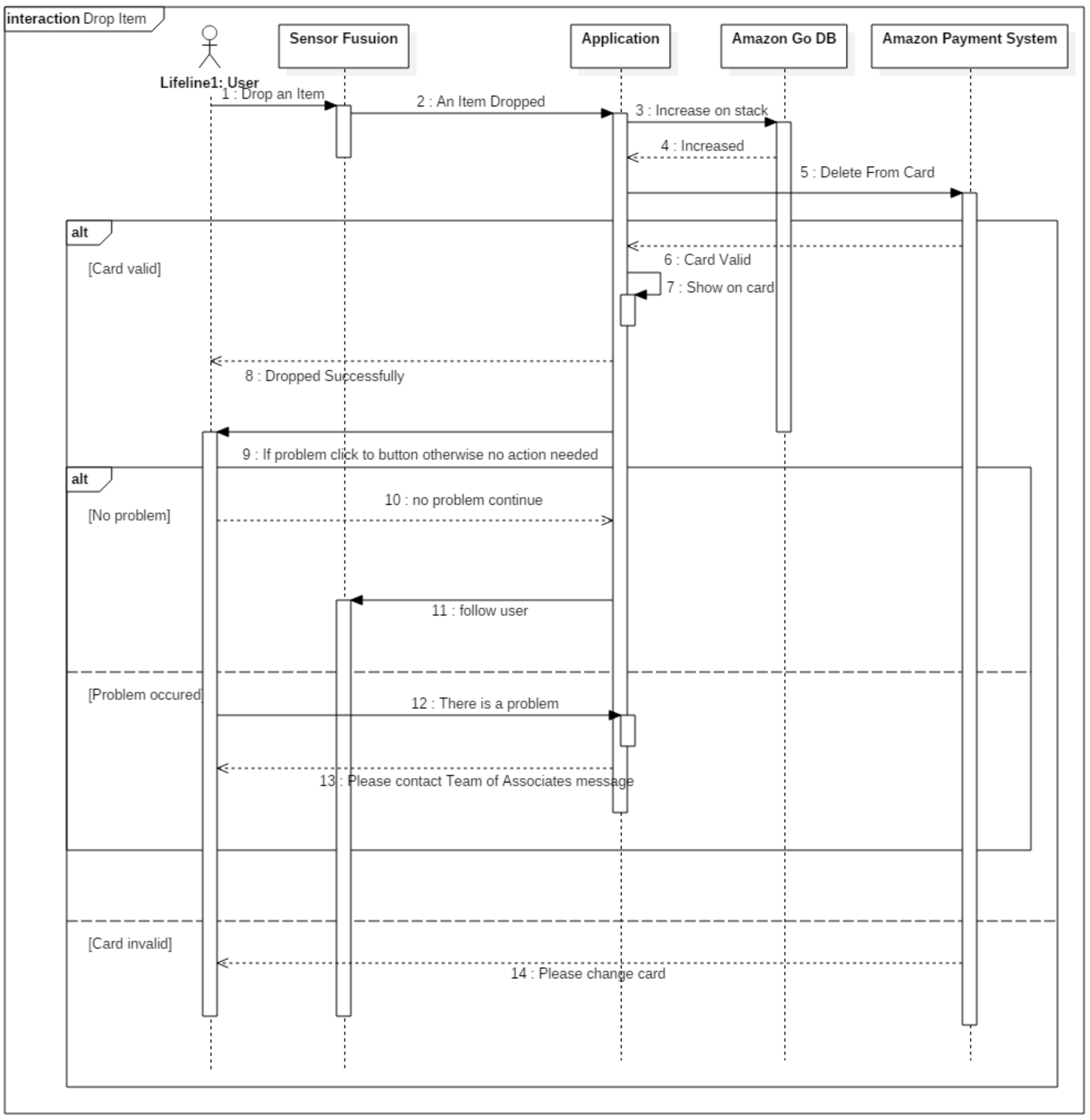


*Figure 14: Drop Item Sequence Diagram*

***Researcher Interface:***

Researcher interface is used by researchers to reach and analyze the Amazon Go data. However, it is forbidden to display user's private data such as name, surname and credit cards. Researchers are only allowed to generic data stored in database server such as product listings, product prizes among stores, activity of users, most selling product types etc.

**Design Rationale:**

- Via this interface researchers will query the database without need of the SQL knowledge.
- Error messages will be displayed when a privacy inversion violation.
- Researchers will be able to create charts from the outputs of queries. Available chart types are line chart, pie chart, column chart and customable tables.

***Team of Associates Interface:***

The purpose of this interface is to maintain products availability and variety. This interface provides to manage the store effectively. Organized simple and easy to use interface.

**Design Rationale:**

- The inputs of this interface come from application and sensor fusion. Inputs consist of some orders with location of data and name, id of product.
- Inputs consists of orders and output consist of click boxes and text messages.
- The output is mission completed or mission is not completed because not enough product or not enough worker to restocking and this output goes application.
- End messages like mission is completed or not completed are displayed.
- There is time limit after order comes, it is 30 minutes.

***Admin Interface:***

Admin interface will provide access to all functionalities and data stored in the system for the admin. This interface is only used by the admins who have appropriate staff ID and password. This is where IT staff can see error logs, which is implemented by the "View error logs" use case.

**Design Rationale:**

- Admins can query the database server to display necessary data.
- Private data such as credit card information cannot be accessible by any admin.
- Unlike Researcher Interface, the Admin Interface is capable and allowed to perform manipulative operations on the database. In other word, the Admin Interface provides a "Read-Write" database access.
- Functionality is more important than the design of this interface.

#### 4.4.2.2 System Interfaces

***The interface between Payments Handler and Amazon Payments Service:***

Amazon Payments Service will query the Amazon Payments Handler for payment data. Payments Handler is responsible for composing and transmitting the payment data to the Amazon Payments Service. Payments Handler will require an authorization from Amazon Payments Service. If it is successful, composed data will be sent to Amazon Payments Service.

**Design Rationale:**

- If an error occurs in any part of the processes, it will be reported back to the Payments Handler component.
- Amazon Payments Service is responsible for successful money transaction. If the payment is successful, Payments Handler component will be notified.

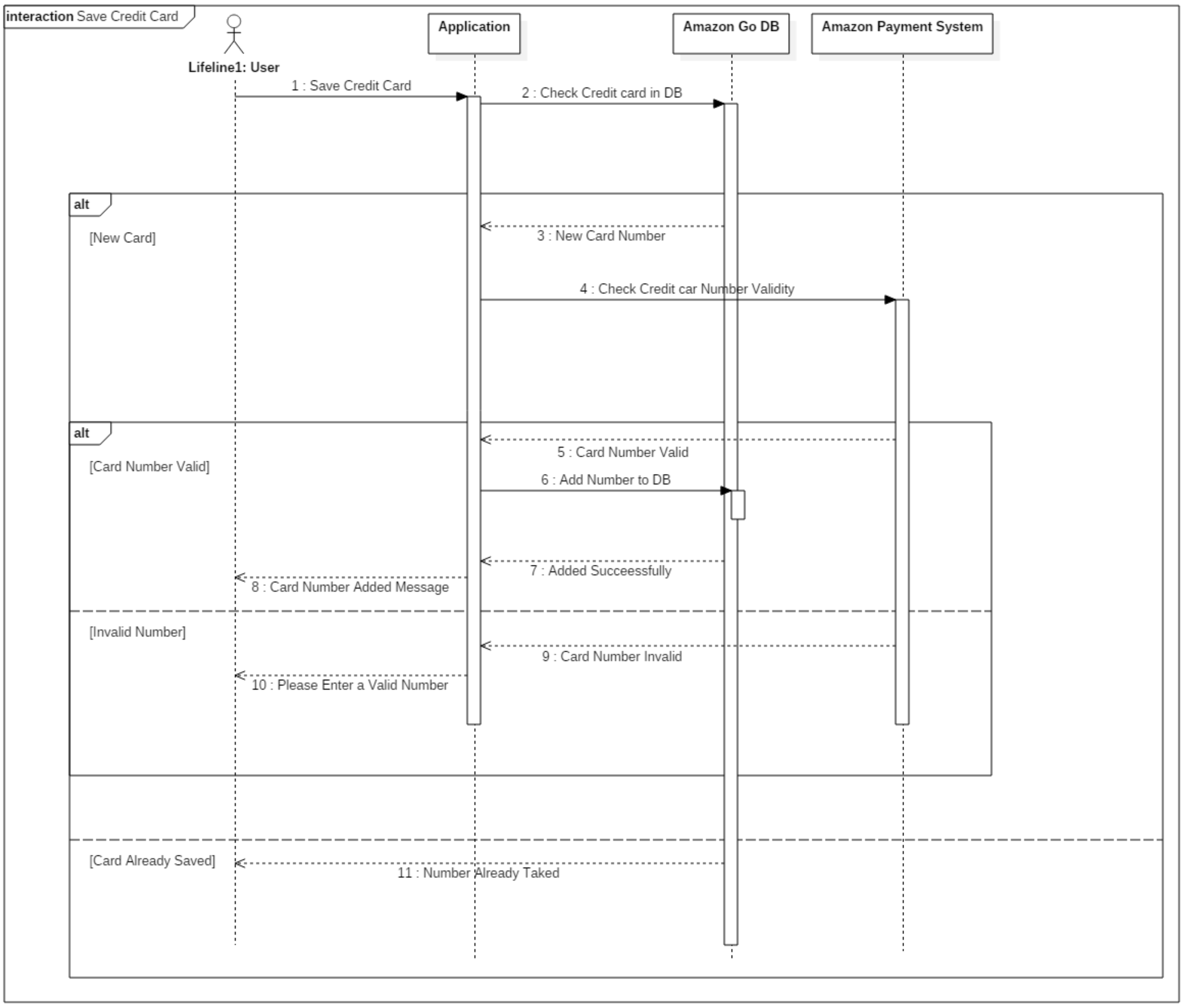


*Figure 15: Save Credit Card Sequence Diagram*

***The interface between Application Controller and E-mail Service:***

While user is signing-in, email authentication is necessary to move on. An automated e-mail sent to the user's registered e-mail. E-mail service is used to sent e-mails. Application Controller sends user data to E-mail Service component.

**Design Rationale:**

- If an error occurs while sending an e-mail to the corresponding e-mail address, Application Controller will be notified and the user will be notified via sign-in interface.
- Application Controller will be notified after a successful operation by E-mail Service so that user can continue to the application.

***The interface between Google Maps Handler and Google Maps:***

Google Maps will query the Google Maps Handler for appropriate map data of the Amazon Go stores. Google Maps Handler component is used to communicate with the external Google Maps component. This component converts the location data of stores to the appropriate format for Google Maps API.

**Design Rationale:**

- Google Maps handler sent the Amazon Go store's location data in appropriate form to Google Maps.
- If an error occurs, Google Maps handler will receive an error message.

***The interface between Payments Handler and Product Supplier System:***

Payments Handler will be responsible for composing and reporting the payment data to the corresponding Product Supplier System. After the composition of the data, Payments Handler component will request authorization from Product Supplier System with its credentials. If the authorization is successful, it will send the composed data to the Product Supplier System as Payment Data.

**Design Rationale:**

- If an error occurs in any part of the processes, it will be reported back to the Payments Handler component.