Bilkent University

Department of Computer Engineering

Senior Design Project

High Level Design Report

Project short-name: GoodBuy

Pelin Çeliksöz - 21600850

Radman Lotfiazar - 21600450

Asım Güneş Üstünalp - 21602271

Turan Mert Duran - 21601418

Ömer Faruk Kayar - 21602452

Supervisor: İbrahim Körpeoğlu

Innovation Expert: Serkan Köse

Jury Members: Shervin Arashloo, Hamdi Dibeklioğlu

1. Introduction

With the decrease in agricultural production and the increase in industrial production, people started to meet their nutritional needs by consuming more packaged products. Users find it difficult to read because the contents of the products are written very small. In addition, they may not understand because it contains foreign words. Reading the ingredients section can be a waste of time. This difficulty causes users to be hesitant when consuming products. One of the reasons for this is that the allergenic substances in the contents of packaged foods can negatively affect people. The prevalence rates of food allergies with people vary between 6–10.5%, however current data suggest that the prevalence of this rate is increasing worldwide [1]. Considering this condition all over the world, the number of people allergic to certain substances is quite high. People who have adopted vegan and vegetarian lifestyles are also hesitant when purchasing packaged products. In order to understand whether the product complies with that life requirement, it is necessary to examine the ingredients in detail, and at this stage, users have difficulties. In addition, some users do not want to waste time by examining the product and trying to read small texts to learn the calorie of the product, the national stamps, the place of production and the price. In addition to that, the increase in the consumption of packaged products encouraged people to use the market. It is seen that the same product is sold at different prices in different markets [2]. Users have to visit all the markets one by one and keep in mind the prices of the products in order to be able to shop more economically, and this causes both physical and mental fatigue. The GoodBuy application which we will create allows people to protect their health, save time and make shopping more economical.

This report will contain the information about the purpose of the system, design goals, definitions, acronyms and abbreviations and overview of the project. After that, the current software architecture which is similar to the application which we will develop will be mentioned. Then, the software architecture overview of the application we will develop, subsystem decomposition, hardware/software mapping, persistent data management, access control and security, global software architecture and boundary conditions information will be given. Then, the templates given in the proposed software architecture under the subsystem services title will be detailed. Later, consideration of various factors in engineering design and teamwork details will be mentioned.

1.1 Purpose of the System

The purpose of the Good Buy mobile application is for users to perform their grocery shopping in a healthier and more affordable way. Thanks to the interface provided by the application, users can access the information about the packaged products by simply holding the interface camera to the product correctly. Users who are members can also be warned according to the features of the product. For example, the user may choose whether or not to buy the product by being aware of undesirable ingredients such as allergy-inducing substances. In this way, GoodBuy aims to protect the health of users. The application also presents the total price of the market basket created by the users in the markets around the selected location to the user as an interface. At the same time, the location information of these markets can be accessed. By sorting the markets according to location or price, the user can choose the market that suits him or her and shop from that market. In this way, GoodBuy aims to save users money and time.

1.2 Design Goals

1.2.1 Scalability

Scalability is an essential component of enterprise software. Prioritizing it from the start leads to lower maintenance costs, better user experience, and higher agility[3]. Since our software is used in markets (somewhere people are in there in their daily life) it is crucial for our project to be scalable as much as possible. In other words, our application has to have the ability to respond to a large number of consumers' requests in the minimum amount of time in order to encourage people to use our app in their daily life. To achieve this, we will be using cloud services that can handle possible large scales of data in the future.

1.2.2 Reliability

Application reliability is the probability of a piece of software operating without failure while in a specified environment over a set duration of time. In a perfect world, a reliable piece of software is completely defect free, does not create downtime, and performs correctly in every scenario[4]. Since our project's goal is to be used in one of the most important activities in people's daily life, it is urgent for our project that all functionality and features run smoothly and accurately. Unity and cloud servers give us enough opportunities for achieving a high reliability in our project.

1.2.3 Availability

Application availability is a measure used to evaluate whether an application is functioning properly and usable to meet the requirements of an individual or business[5]. We are not designing or developing this project for a company or business. However, Bilkent University is asking about a requirement report in which we should specify all functionalities.. Therefore, we are doing our best to design and develop all functionality with high accuracy and lowest fault.

1.2.4 Maintainability

Understanding software maintainability allows organizations to identify improvement areas as well as determine the value supplied by current applications or during development changes[6]. Since we want to be nominated for at least one of the prizes in the CS Fair, we have to design and implement our project as maintainable as possible. Therefore, high maintainability of the project helps us to make changes and add features easily in order to make our app more and more powerful. We can achieve high maintainability by having a perfect software architecture. Hence, we are planning to have meetings with software architecture experts for deciding our software architecture in different parts of the project.

1.2.5 Security

Since we are storing images, users' information and their medical information, security is one of the most important non-functional requirements for our project. Therefore, we decided to use Firebase which is reliable and highly secure for storing this information. Furthermore, for storing images we are using Vuforia's cloud servers hence, our data will be highly secured there by Vuforia.

1.2.6 Usability

The interface of the application will provide necessary and sufficient features to give the best user-friendly experience to the users. It will consist of accurate images of the products, different language options and simplistic design.

1.3 Definitions, Acronyms, and Abbreviations

AR: Augmented Reality

App: Application

GDPR: General Data Protection Regulation

GPS: Global Positioning System

Vuforia: Comprehensive and scalable AR platform

Unity: Cross-platform game engine

C#: multi-paradigm programming language

Firebase: helps to build and run applications

IOS: Mobile operating system created and developed by Apple

Android: A software package developed by Google. It is a linux based operating system for mobile devices such as tablet computers and smartphones.

App Store: Provides the process of downloading applications for IOS.

Google Play Store: Provides the process of downloading applications for Android.

Google Maps: A navigation application created by Google.

1.4 Overview

GoodBuy is a mobile application. Users who want to buy packaged products take care of their health and save time and money thanks to GoodBuy. A feature of this application is that the application can recognize the packaged products from the camera of the application, thanks to the augmented reality technology. In this way, information about the product can be accessed by the customers in the market, without touching the product, by simply holding the application camera towards the product. This information is intended to facilitate the life of the user.

Users who are members of the system register the substances that their bodies have allergic reactions to the system. Then, if this allergen is found in the product that the user shows into the system, the system informs the user by giving an alert. In this way, users buy the product after they are sure that the product will not cause an allergic reaction, and the possibility of experiencing product-related health problems is reduced. Users also specify their lifestyles when registering to the system (such as vegan or vegetarian). When the user shows the product in the market to the system, he can easily understand whether the content of the product fits his lifestyle or not. For example, when a user registered to the system as a vegan shows a non-vegan product to the application's camera, the application alerts the user and notifies the user. Thus, users can shop without hesitation, making sure that they buy products suitable for their lifestyle. At the same time, when the user points the

camera of this application to a packaged product, he can easily access the calorie, the national stamps, the place of production and the price information about the product. Thus, he does not waste time by taking the product in his hands and examining it.

The same products can be sold at different prices in the markets. GoodBuy also helps users spend less money on their market shopping. This convenience is provided in two ways. First, when the user reads the product to the camera of the application, the system shows the price of that product in other markets to the user, indicating that the user can reach the product at a more affordable price. The second is to create a shopping list before the user goes to the market through the system. The system shows the total price of the products in this shopping list for each market. At the same time, these markets are ranked according to the proximity of the person's location. The person can also access the locations of the markets through the system. In this way, the user decides where to do his shopping in the most economical way and goes to that market.

There will also be an admin website for managing the application. In addition, the GoodBuy application will have a section where users report changing or not found product packaging to the admin.

2. Current Software Architecture

Nowadays, applications which are working by augmented reality are increasing dramatically. Each of them have different functionalities and provide interesting services to users. If we want to mention one of the most outstanding ones we can mention Google Lens which not only can recognize the objects but also it can provide some services such as connecting to wifi by camera or find out where you are exactly. However, in the marketing industry we are not encountered with different applications. If we want to mention the applications in the market industry which are working by augmented reality we can mention all mobile cameras which can recognize the QRcode and connect the user to a website. Therefore, GoodBuy has a different software architecture than the current application but in the same base. Here we will discuss Google Lens software architecture because it is one of the most successful applications in augmented reality and has some features which are similar to GoodBuy.

- **Google Lens**

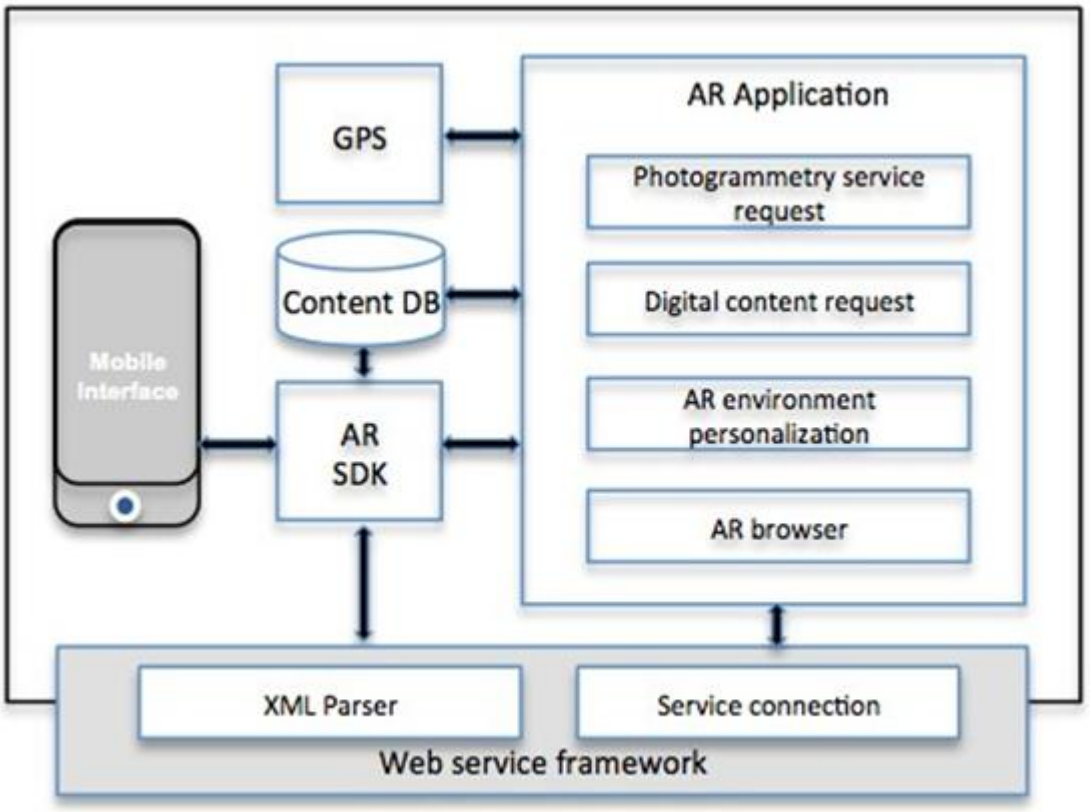

Figure 2.1. Google Lens Web Service Framework

This is the basis of augmented reality applications. As you can see there are three subsystems in there which include, mobile interface, AR application and web service framework. Google Lens also includes all these subsystems and you are able to see them in the design of Google Lens software architecture. Our application also has a similar software architecture as the same as Google Lens and the software architecture which we provide here. However, the difference is in GoodBuy we are using Vuforia Engine which is one of the functionalities and libraries of Unity Game Engine. In Vuforia we are not implementing any image recognition algorithm and all properties will be done by Vuforia Engine which we will discuss our software architecture in the next section.

3. Proposed Software Architecture

3.1 Overview

For our software architecture we decided to use a 4+1 architectural view model. Therefore, we have 4 views including logical, development, physical and process views and a scenario. Furthermore, for each of these diagrams we used a different style such as Client-Server style from Component & Connector style for process view. We will discuss these styles more in this section. Moreover, here we will provide all diagrams which we designed for our application and we will discuss them in subsystem decomposition, hardware/software mapping and in section 4 subsystem services. Furthermore, in this section we will discuss the data management of the application and how they will be secured and controlled. In the boundary conditions subsection we will discuss the initialization, termination and failure boundary conditions of the system.

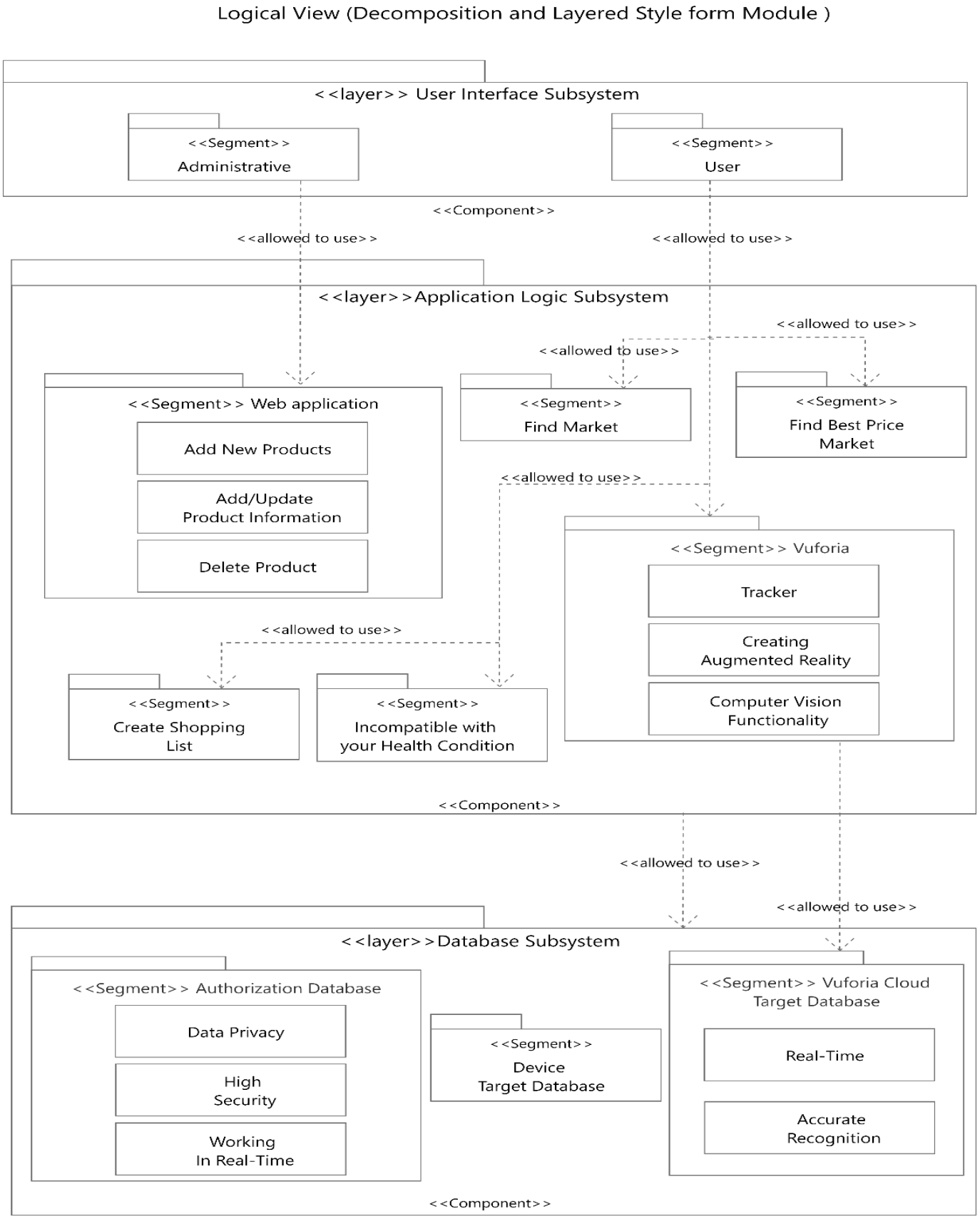

Figure 3.1. GoodBuy Logical View

Development View (UML Object and Class Diagram)

Figure 3.2. GoodBuy Development View

Process View (Client and Server Sttyle from Component & Connector)

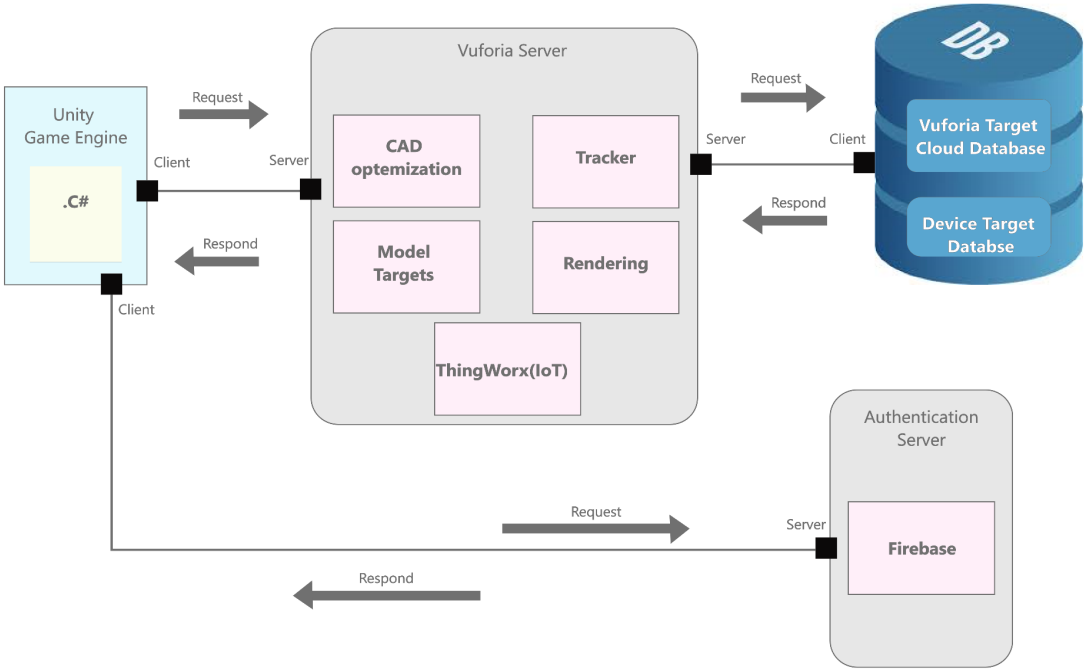

Figure 3.3. GoodBuy Process View

Physical View (Deployment Style form Allocation)

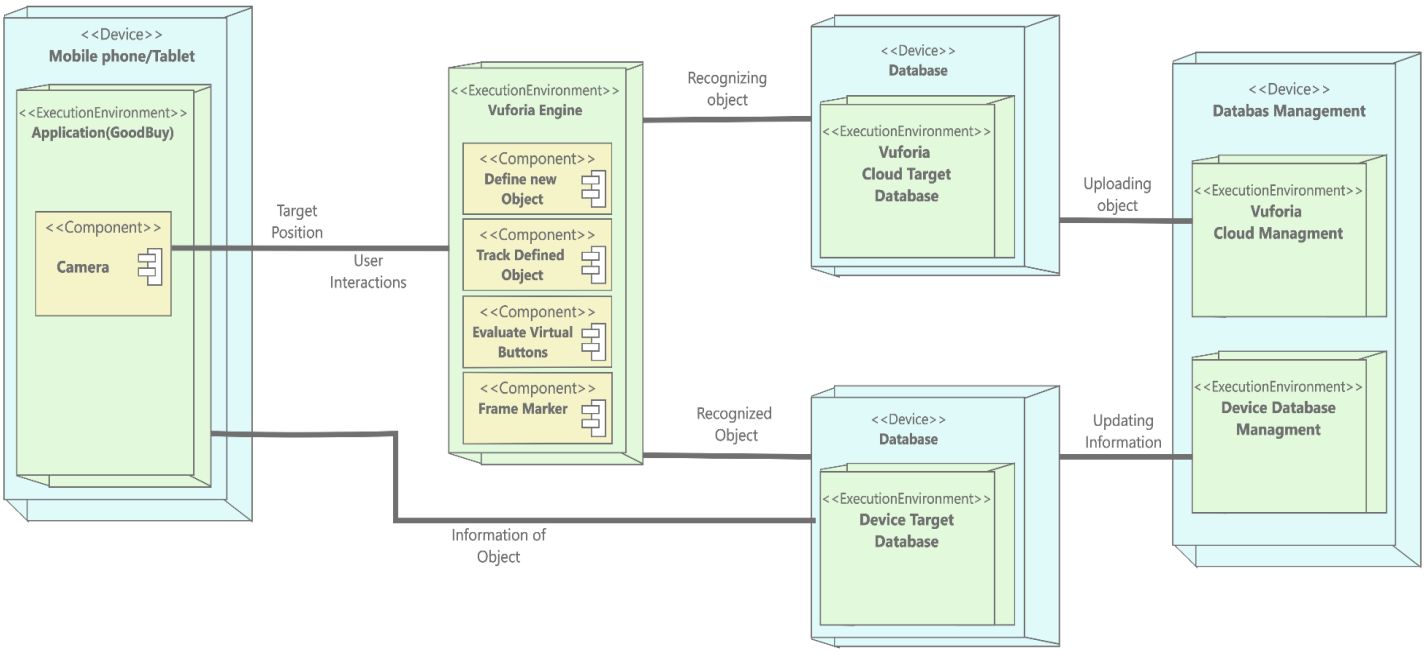

Figure 3.4. GoodBuy Physical View

Scenario (UML Use-Case Diagram)

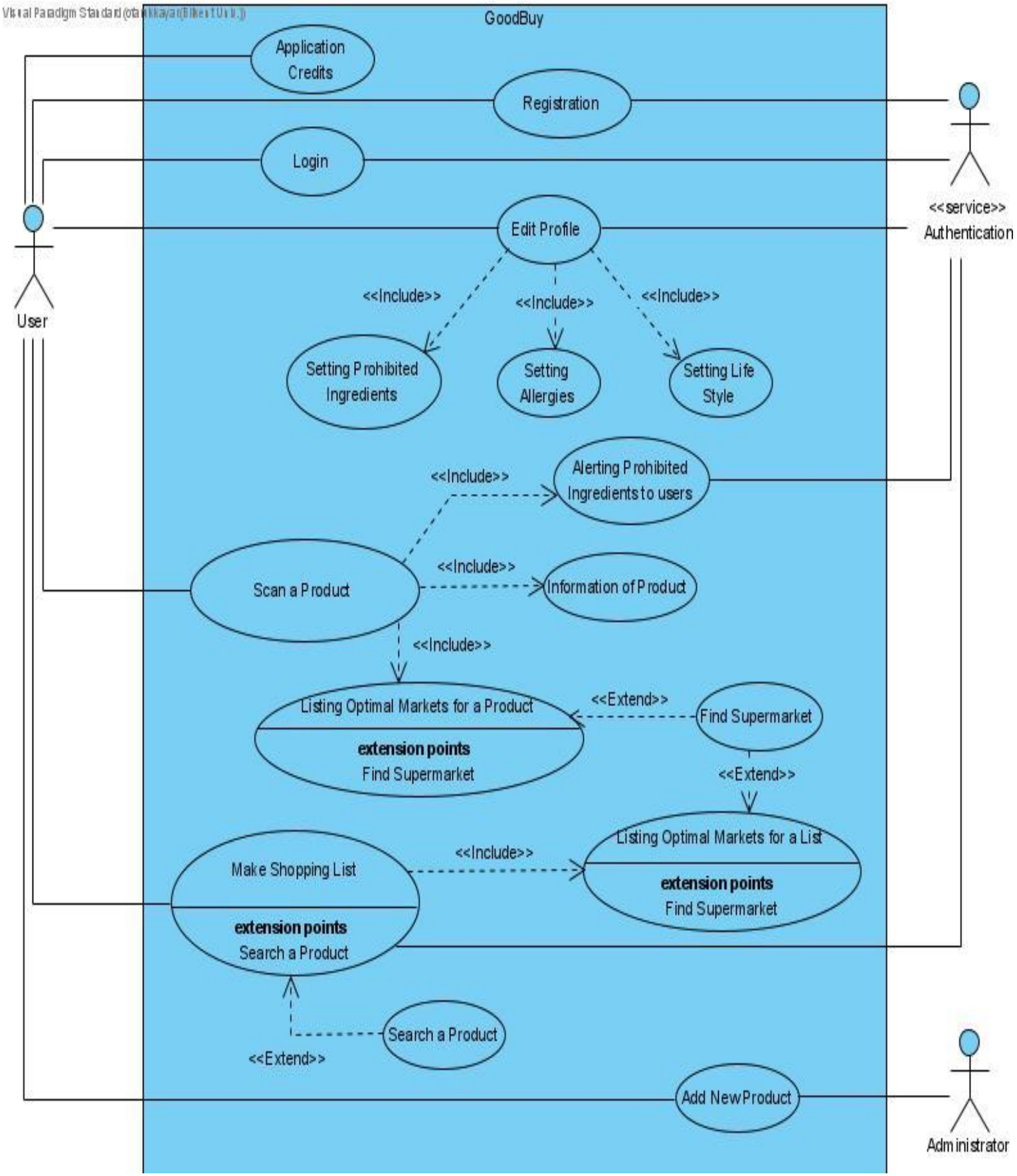

Figure 3.5. GoodBuy Scenario

3.2 Subsystem Decomposition

If you look at figure 3.1 there is a logical view from 4+1 architectural view model which is designed by Decomposition and Layered style from Module style. By this diagram we try to indicate the subsystem decompositions. Each layer indicates our subsystem. For instance, the first layer is the user interface subsystem and we are able to show that it includes segmentations which are Administrative and User. The next subsystem is Application Logic which includes some segmentations which indicate the services of this application logic subsystem. In addition, the last but not least subsystem which we can mention from figure 3.1 is the database system or in other words logical subsystem. By getting help from Decomposition and Layered styles we could easily indicate our subsystems decomposition and the way they are allowed to use each other.

Moreover, the process view from 4+1 architectural style (figure 3.3) view "deals with the dynamic aspects of the system, explains the system processes and how they communicate, and focuses on the run time behavior of the system"[7] . Therefore, we are able to indicate the process which is happening to each subsystem and how they communicate with each other. As we state in the figure 3.3 for designing the process view we used Client-Server styles from Component & Connector style which leads to dividing our application into servers and client parts. Moreover, this diagram is designed only for user segmentation and administrative segmentation is declined because our main features and focus is on user segmentation. Therefore, by this diagram you are able to find out each of our subsystem decompositions are related to each other and they include which processes. Client-Server Style helps us to divide our subsystems into two groups: clients which are user interface and storages and servers including Vuforial Server and Authentication Server.

Vuforia Engine is defined as a server here because most of the recognition and processes of the image is done there. In other words, it is a server because the client which is our camera will send an image or object picture to Vuforia Engine in order to find out its information which is stored in the database. Moreover, we also have an authentication server which allows users to login by their Google account, facebook or instagram account.

In addition to these clients and servers, we indicate that our storage is also client because it needs a server for asking the data which is stored in there and after that it can respond to that request. Therefore, our storage or database subsystem decomposition is also client in this representation.

3.3 Hardware/Software Mapping

For this section we can mention the Physical view from 4+1 architectural view model (figure 3.4). “The physical view (aka the deployment view) depicts the system from a system engineer's point of view. It is concerned with the topology of software components on the physical layer as well as the physical connections between these components”. Therefore, by this diagram we are able to indicate the software and hardware parts of our project and their relation and how they communicate with each other. As it is stated in figure 3.4 for designing this diagram we are using Deployment Style from allocation style. Development Style helps us to indicate what are the hardware and software parts of the project by the component and what is their relation.

As you can see in figure 3.4, GoodBuy includes databases and mobile phones and tablets which are hardware. Those hardware includes some software or they have a connection with some software. For instance, the mobile phone which is hardware in this project includes an execution environment which is GoodBuy and it uses the mobile phone camera for detecting objects and connected to the Vuforia Engine for processing the object image. After that Vuforia engine asks for pictures and information from the database in order to find out the correct image and information in the database. Figure 3.5 helps us to understand better the hardware/software mapping of our project.

3.4 Persistent Data Management

We plan to keep users' personal information, including health details, inside the devices. The information of the users registered in our application will be kept in the cloud system. We are using FireBase for the cloud database. We plan to take authorization and authentication measures for the security of this registration system. The software language of our servers will be C# and control and development will be done through Unity. In addition, the feedback loop we plan to establish for the sustainability of our system will also run through this cloud database.

Speaking of costs, we plan to use the free option of FireBase as our project is in the prototype stage. Paid tariffs may be revised in the later stages of the project.

3.5 Access Control and Security

Due to the fact that our application is built around a key feature, it can be used without registration. However, registration is recommended to enjoy all aspects of the key feature. The following types of information are required to register for the application: name, surname, email, password, username. After entering the information, the system will send a confirmation email for security reasons. We plan to add features such as time limit and new device alerts to ensure security.

There will be two types of users in the system: customers and administrators. Customers will be bound to their email address and opening a second account with the same email address will be prevented. Administrator accounts, on the other hand, will only be able to manually register to the server. Customers' private information will not be shared with any third parties. Except for the registration information, the health information in the local environment will be under the sole responsibility of the user.

3.6 Global Software Control

ApplicationManager mainly manages the GoodBuy application. The ApplicationManager contains all the pages of the application and which page the application is on. The User Manager manages who is using the application and operations such as login, register, delete user, logout, update recommended, update not recommended, update password. Database Manager has the url of the application and the password of the user. Database Manager manages get, delete, add, update for users and products. The Product Manager has the information of the current product scanned by the application. It manages the get and update process for the current product's information and the prices of the same product in other markets. Price Analysis Manager manages the search markets operation. It also manages the get operation for the cheapest price and the cheapest market name. Sound Manager manages sound operations in the application. GPS Manager has location information of the markets. Based on this information, it manages to get the current location. According to the result, it manages the redirection of the user to Google Maps. GPS Manager also manages finding the closest market. The Shopping List Manager, on the other hand, has a product list and source location information. Manages get, remove and add product jobs. The Shopping List Manager takes care of calculating prices.

3.7 Boundary Conditions

3.7.1 Initialization

To use the GoodBuy application, it is needed that an IOS or Android phone and an internet connection to download the application. To initialize the application, it must first be downloaded from the App Store for IOS and Google Play Store for Android. Afterward, users can open the application by clicking the application on their phones and start using it as a member or not.

3.7.2 Termination

Users who are members can perform the logout operation by pressing the logout button on the interface of the application. To exit the application, they must click the phone's exit button for applications. Even if the application exit button is pressed, if the phone does not close the application completely, it will continue to run in the background. To delete the GoodBuy application from the phone, the application deletion of the phone is used.

3.7.3 Failure

If there is internet loss while downloading the application, the GoodBuy application cannot be downloaded and initiated. If the application is downloaded and used, if there is an internet loss and the user is not a member of the system, the transactions while using the system are lost and cannot use the application until the internet comes back. If the user is a member of the system, he can access all of the transactions he made before the internet loss and continue to operate after the internet comes back.

4. Subsystem services

As we stated in the Subsystem Decomposition we have three subsystems including user interface, application logic and storage. In this section we will discuss services and classes which include each of these subsystems.

4.1 User Interface

In the figure 3.1 we show that the User Interface is including tw segment s including Administrative and users. In this part we will not discuss the administrative side because it only requests the database to add a new product to the database. Therefore we will discuss the user segmentation which is our main focus in this project.

4.1.1 User

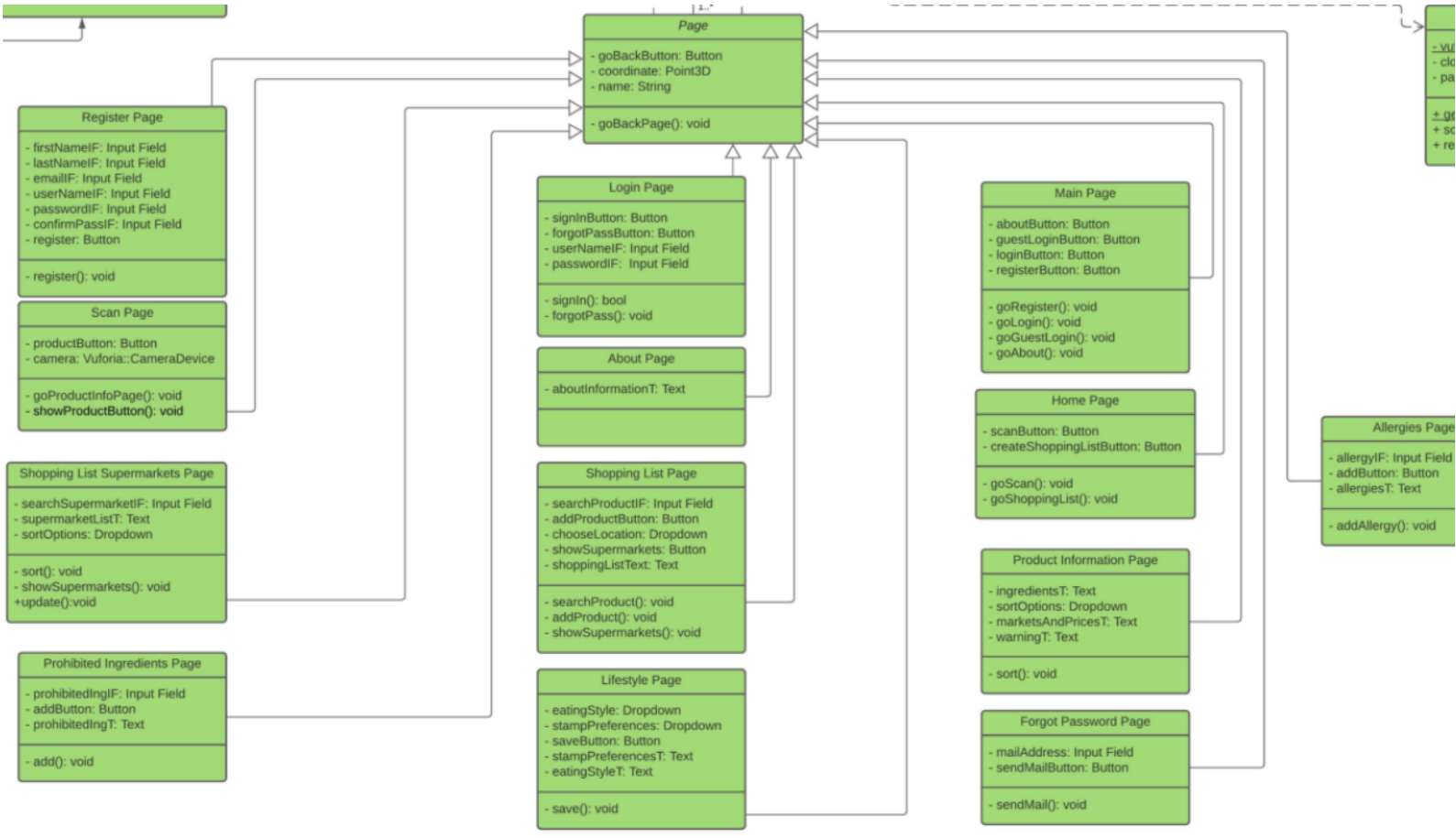

Figure 4.1.GoodBuy Class Diagram for User Interface

As you can see figure 4.1 is a part of the Development view from the 4+1 architectural view model of our project which is designed by UML object and class diagram. Figure 4.1 indicates that each page of the user segmentation of the user interface subsystem consists of which services and they communicate with each other.

- **Page:** It is an abstract class that includes buttons for go back, coordinate of pages in 3D plane and name.
- **Register Page:** This class keeps necessary UI elements and methods for registration page.
- **Scan Page:** This class keeps necessary UI elements and methods for scanning product pages.
- **Shopping Cart Supermarkets Page:** This class keeps necessary UI elements and methods for shopping list supermarkets page.
- **Prohibited Ingredients Page:** This class keeps necessary UI elements and methods for prohibited ingredients page.
- **Login Page:** This class keeps necessary UI elements and methods for the login page.
- **About Page:** This class keeps necessary UI elements for the page.
- **Shopping Cart Page:** This class keeps necessary UI elements and methods for creating a shopping cart page.
- **Lifestyle Page:** This class keeps necessary UI elements and methods for lifestyle pages.
- **Main Page:** This class keeps necessary UI elements and methods for the main page.
- **Home Page:** This class keeps necessary UI elements and methods for the home page.
- **Product Information Page:** This class keeps necessary UI elements and methods for the product information page.
- **Forgot Password Page:** This class keeps necessary UI elements and methods for forgetting password pages.
- **Allergies Page:** This class keeps necessary UI elements and methods for allergies page.

4.2 Application Logic

In the Logical View (figure 3.1) we explained that another subsystem decomposition which we are faced with in GoodBuy application is Application Logic which actually manages the connection between the user interface with Vuforia Engine, database and Users information.

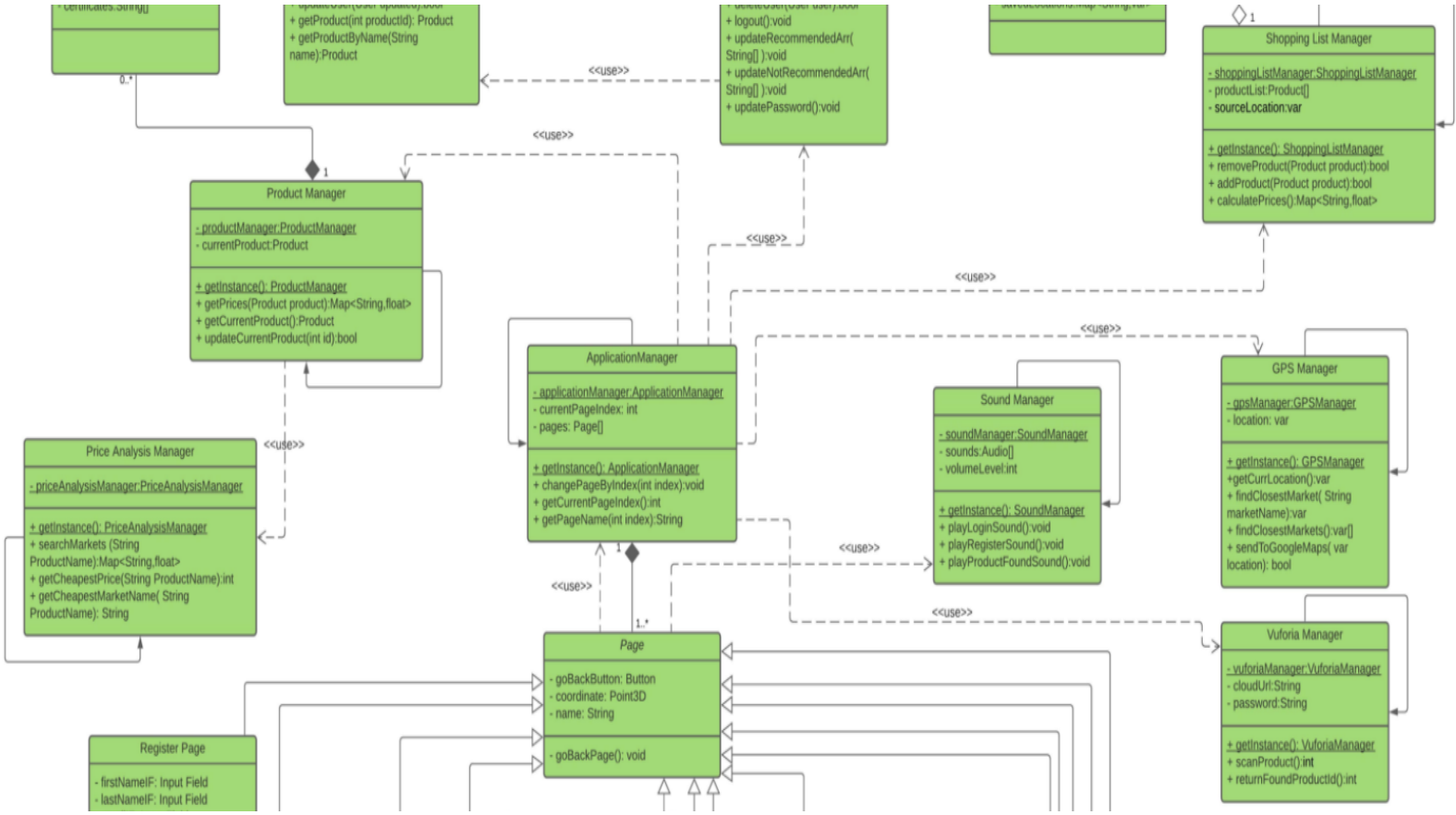

Figure 4.2. GoodBuy Class Diagram for Application Logic

Figure 4.2 is part of our Development view (figure 3.2) which includes all management of our application. These management is the heart of application because they are controlling the application. Application manager class which you can find in figure 4.2 is a singleton class which leads to all managers connecting to each other and specially with the page which is our user interface.

- **Application Manager:** Application manager is working as a wrapper class that arranges and controls initializations and other manager classes. On the other hand, it manages the Unity camera and controls page transitions.
- **Sound Manager:** Sound manager controls sounds and sound effects on actions.

- Vuforia Manager: Vuforia manager controls the processes of communication with Vuforia cloud and login procedures.
- **GPS Manager:** GPS manager handles finding closest markets by getting location of user and connects our application with Google Map.
- **User Manager:** User manager handles user information, login, register, recommended list arrays.

4.2 Database and Storage

The third subsystem decomposition which we defined in Logical View (figure 3.1) is the database of our application. In GoodBuy we will have 3 databases, one of them is for users information which is indicated in Development View (figure 3.2 and 4.3) and Vuforia Cloud target database and device target database which are used for detecting the object and information of of object respectively.

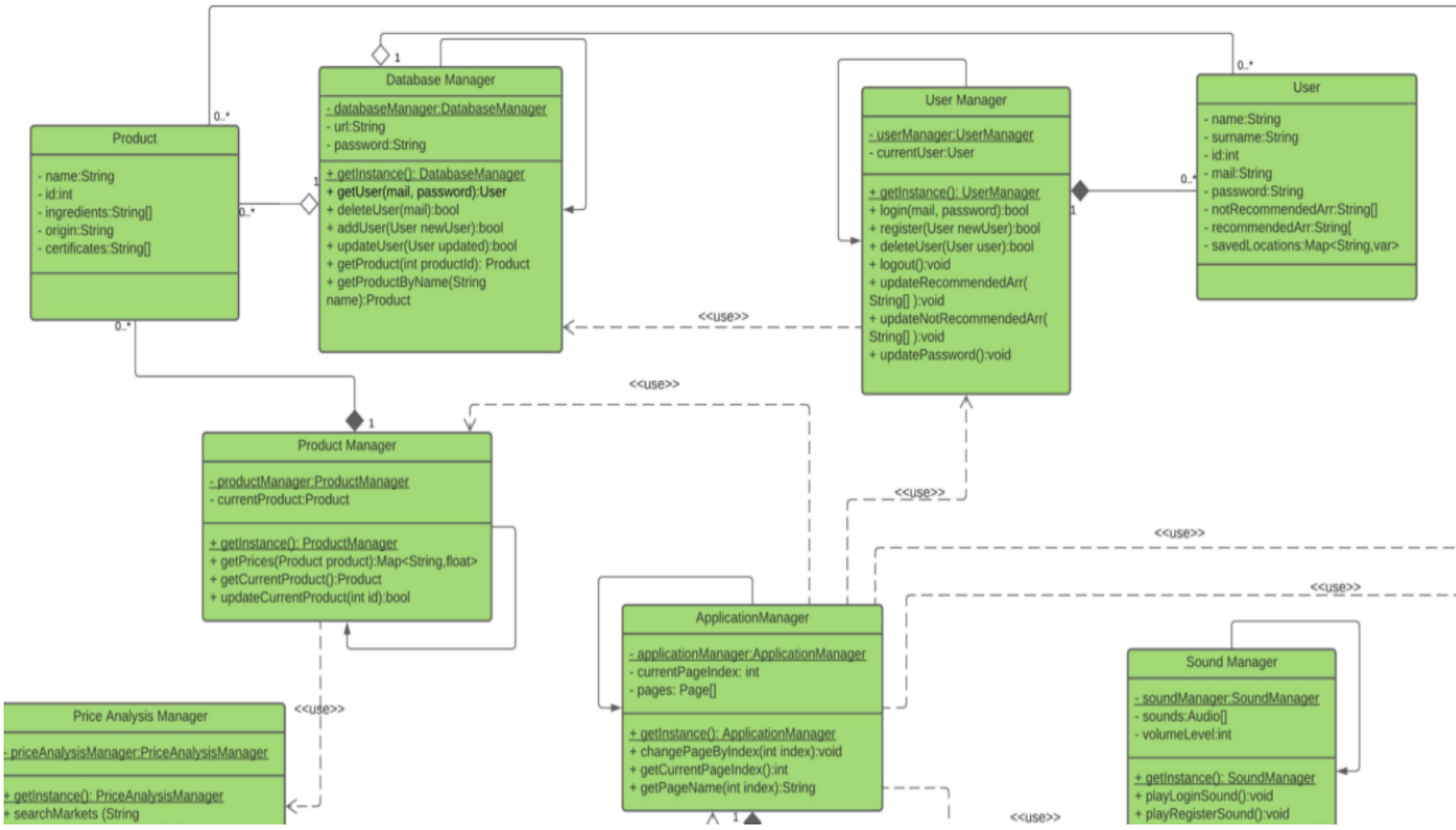

Figure 4.3. GoodBuy Class Diagram for Database and Storage

This is a part of the Development View (Figure 3.2) which indicates the database we use for storing users information and product information. As you can see we have a database singleton manager which has access to the user and product information.

- **User:** This class keeps track of user information.
- **Database Manager:** Database manager controls connection with our database and handles database needed processes.
- **Product:** This class keeps track of product information.
- **Product Manager:** This manager keeps track of prices of the products and controls product information.

5. Consideration of Various Factors in Engineering Design

Following topics are the factors that can possibly be affected by our project.

5.1 Security Concerns

Since we collect personal information from our users, the security of these data will be crucial. We will be following most accepted regulations such as GDPR so that users can trust the product. In addition, we use the GPS data of the user and the supermarkets in the application. Thus, there will not be any sharing of location data of the users to the servers for the sake of best security. We will be setting the boundary of security tight so that there cannot be any leakage.

5.2 Public Health and Sustainability

We are basically developing an application where you can see the ingredients of a packaged product so this means we make it easier for users to detect the best option for them. This best option could be in terms of price and health. Presenting a better option is an example of sustainability since there will be less consumption of unwanted products. According to our estimates, in the long term, extensive use of this application will cause a positive effect on public health.

5.3 Communication and Sustainability

In our application, we will be implementing a feedback loop for the renewal and addition of products that come from the users. This will provide a sustainable environment for our continuing system. In addition, we will be adding an extra language option for the menu and context since all the tourists are our possible customers. This feature will bring a variety to the communication inside the application.

5.4 Social Factors

GoodBuy has no apparent factor on gender, age, weight, height or race. Our application can be used by any individual of society.

6. Teamwork Details

6.1 Contributing and Functioning Effectively on the Team

Since we evaluated the ideas and interests of all group members while the project was still at the idea stage, we maximized the motivation of the group for the project. This motivation will play an important role in contribution and functionality. We want to keep communication on the line so that the group works as a whole. Being aware of the importance of communication, we will ensure that any member does not fall short of their work pace. In case of possible decision changes on the project, each group member will adapt quickly to the changes. In addition, we determine the distribution of tasks according to the interests and areas of expertise of each member, so that their motivation can stay fresh.

6.2 Helping Creating a Collaborative and Inclusive Environment

Our main goal is to know the strengths of the team members and to create the project by using these strengths at the most efficient level. In order to fulfill this purpose, we must work as open-mindedly as possible and to look out for each other's weaknesses. At this point, we can again emphasize the importance of communication. creating a collaborative and inclusive environment will only be possible with a strong communication network.

6.3 Taking Lead Role and Sharing Leadership on the Team

We plan to assign group members a leadership role according to the strengths mentioned in the previous section. More than one person can take the leadership role at the same time. At the points in between, we indicate the important points of the options and present them to group voting. The last word, which is usually the result of voting, is always the leader of that topic. This concept not only allows everyone to take a leadership position, but also ensures that leadership is shared and the project develops in the best possible environment.

7. Glossary

AR: Augmented Reality
App: Application
GDPR: General Data Protection Regulation
GPS: Global Positioning System

8. References

1. J. Jiang, C. M. Warren, and R. S. Gupta, “Epidemiology and racial/ethnic differences in food allergy,” *Pediatric Food Allergy*, pp. 3–16, 2020.
2. H. Görg, L. Halpern, and B. Muraközy, “Why do within-firm-product export prices differ across markets? evidence from Hungary,” *The World Economy*, vol. 40, no. 6, pp. 1233–1246, 2016.
3. “The importance of scalability in software design,” *Orlando Mobile App And Web Development Company*. [Online]. Available: https://www.conceptatech.com/blog/importance-of-scalability-in-software-design. [Accessed: 10-Oct-2021].
4. “Application reliability defined: |free demo: | video explanation,” *Default*. [Online]. Available: https://www.castsoftware.com/glossary/application-reliability. [Accessed: 10-Oct-2021].
5. “The importance of scalability in software design,” *Orlando Mobile App And Web Development Company*. [Online]. Available: https://www.conceptatech.com/blog/importance-of-scalability-in-software-design. [Accessed: 10-Oct-2021].
6. “Software maintainability: | free demo: | video explanation,” *Default*. [Online]. Available: https://www.castsoftware.com/glossary/software-maintainability. [Accessed: 10-Oct-2021].
7. “4+1 architectural view model,” *Wikipedia*, 13-Dec-2021. [Online]. Available: https://en.wikipedia.org/wiki/4%2B1_architectural_view_model. [Accessed: 24-Dec-2021].