Bilkent University

Department of Computer Engineering

Senior Design Project

Low 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

Instructors: Erhan Dolak, Tağmaç Topal

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 includes comparison for object design trade-offs of GoodBuy application. It also describes the interface guidelines documentation and engineering standards of the GoodBuy application. The packages we use while implementing the GoodBuy application and the classes in the class diagram of the application are also included in this report. In the title of Class interface, the classes used while implementing the GoodBuy application and the variables and methods contained in that class are explained and examined.

1.1 Object design trade-offs

1.1.1 Usability - Functionality

There are some requirements such as sufficient phone software, internet access and user login in order for our users to use all features of the GoodBuy easily. Therefore, there will be some functionality-based conditions that must be met for the application to provide the best usability.

1.1.2 Security - Cost

Some information, especially about health, of customers using GoodBuy will be stored in the application and in databases. Since the best protection of users' information is one of our top priorities, we have taken into account the extra expenses that these measures will cause. In this context, we used online services that have been proven to be most safe, regardless of their fees.

1.1.3 Content - Reliability

We plan to obtain the image and price information that we will use in the database, in the first place, from the supermarkets' own sites and from some sites that collect this information. In the next stages, we want to keep our database up to date and enrich it with the feedback from the users. however, feedback in this case may cause reliability issues. As a solution to this, we will work on the feedback control system.

1.2 Interface documentation guidelines

In this report, we are providing classes, variables and methods names and their information. Hence, for begging more understandable and easy to follow we are pursuing some conventions which we are mentioning here. All class names are in PascalCase format and singular such as DatabaseManager, SoundManager and UserManager. However, we are providing method and variable names as ‘methodName()’ and ‘variableName’ respectively. Furthermore, in the table which we are providing an example below the order is the same as ClassName, variableName and methodName.

ClassName
A brief description of class
Variables
variableName: variableType
Methods
methodReturn:methodNames(parameters) A brief description of class

1.3 Engineering standards (e.g., UML and IEEE)

Since, UML Diagrams is one of the most important part of object oriented method of programming due to their varieties in diagrams, methods and components for indication such as hardware-software components depiction, user case, Class, scenario, activity and subsystem decomposition we decided to use it for indicating and describing different part of GoodBuy. In addition, for the architecture of the GoodBuy project we are using a 4+1 software architectural view model which again is based on UML diagrams. For documentation of the project we are using IEEE format for citation and formatting the all documents.

1.4 Definitions, acronyms, and abbreviations

AR: Augmented Reality
App: Application
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.

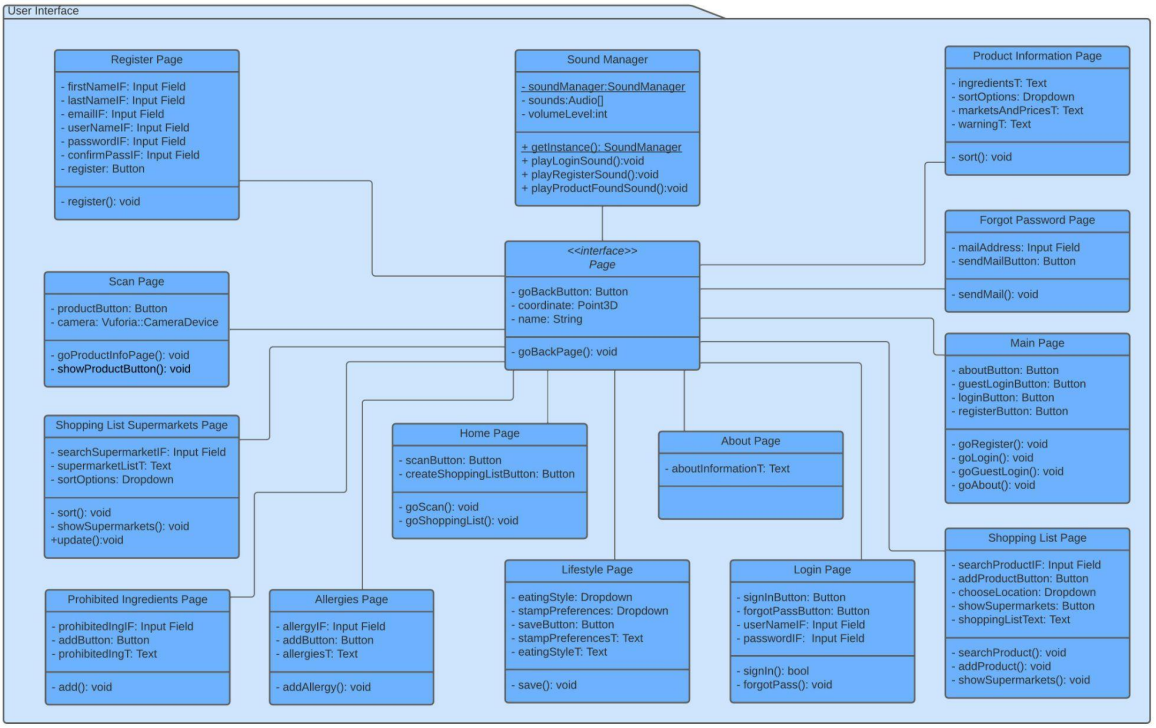
App Store: Provides the process of downloading applications for IOS.
Google Play Store: Provides the process of downloading applications for Android.
Google Maps: A navigation application created by Google.
UML: Unified Modeling Language

2. Packages

GoodBuy consists of 3 subsystems: User Interface, Application Logic and Storage. User Interface subsystem deals with the User Interface(UI) aspect of the application as its name suggests. Application Logic subsystem deals with the necessary computations that need to be done in order to perform the features of the application. Storage subsystem deals with the communication with the databases and data processing. Each subsystem will have its own package.

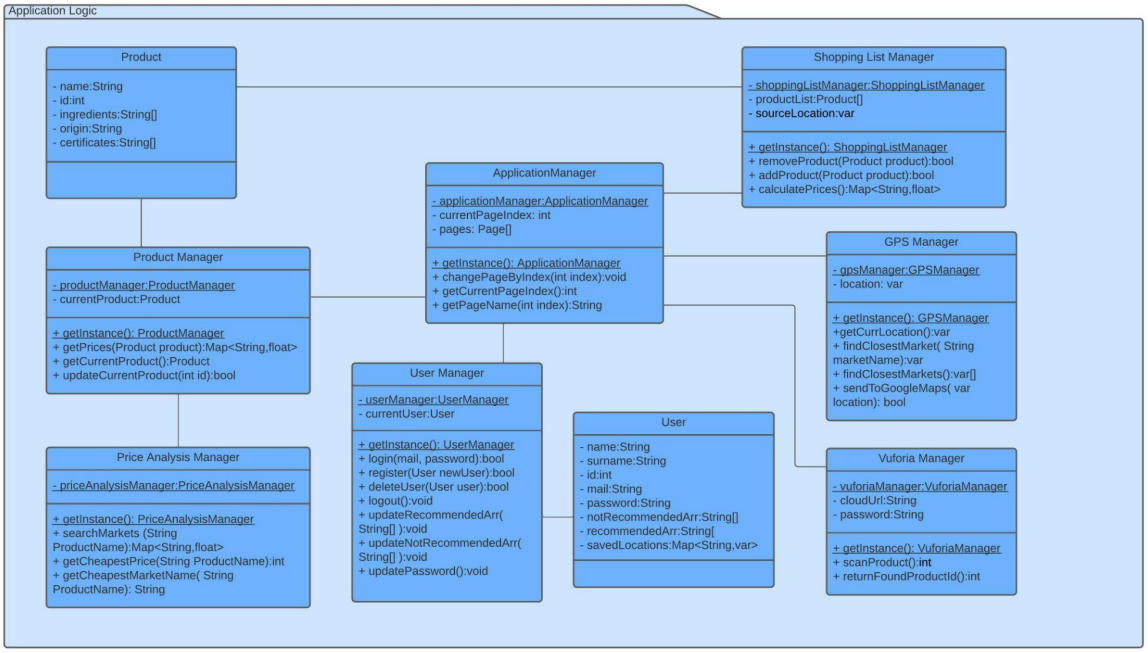
2.1 User Interface

User Interface subsystem deals with the UI part of the application. It contains the pages and the sound manager.



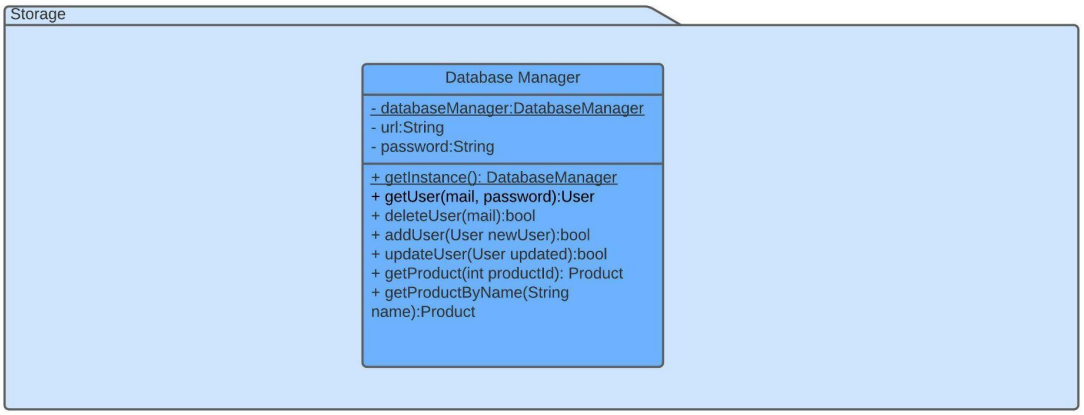
2.2 Application Logic

Application Logic subsystem deals with the back-end side of our application. It contains the necessary managers and plain objects.



2.3 Storage

Storage subsystem deals with the communication and data transfer to/from the database. It contains the Database Manager.



3. Class Interfaces

Database Manager
This class handles Database connection and requests.
Variables
- databaseManager:DatabaseManager - url:String - password:String
Methods
DatabaseManager:getInstance() It is a singleton class and this method returns it's instance. User:getUser(mail, password) This method returns the user from the database by providing mail and password. bool:deleteUser(mail) This method deletes user by specified mail address. True if user found, false otherwise. bool:addUser(User newUser) This method adds user to the database.True if user found, false otherwise. bool:updateUser() This method updates User information. True if user found, false otherwise. Product:getProduct(productId) This method returns product if product found with given id, null otherwise. Product:getProductByName(name) This method returns product if product found with given name, null otherwise.

Product
This class specifies the product features.
Variables
- name:String - id:int

- ingredients:String[]
- origin:String
- certificates:String[]

Product Manager
This class manages products.
Variables
- productManager:ProductManager - currentProduct:Product
Methods
ProductManager:getInstance() It is a singleton class and this method returns it's instance. Map<String,float>:getPrices(product) This method returns the prices of the product in different markets. Product:getCurrentProduct() This method returns the product that the user scanned last. bool:updateCurrentProduct(id) This method updates the last scanned product. True if id is valid, false otherwise.

Price Analysis Manager
This class manages prices of the products.
Variables
- priceAnalysisManager:PriceAnalysisManager
Methods
PriceAnalysisManager:getInstance() It is a singleton class and this method returns it's instance. Map<String,float>:searchMarkets(product name) This method returns the prices of the product in different markets.

float:getCheapestPrice(product name)

This method returns the cheapest price of the product.

String:getCheapestMarketName(product name)

This method returns the name of the market where the product is sold cheapest.

User
This class specifies the user features.
Variables
- name:String - surname:String - id:int - mail:String - password:String - notRecommendedArr:String[] - recommendedArr:String[] - savedLocations:Map<String,var>

User Manager
This class manages users.
Variables
- userManager:UserManager - currentUser:User
Methods

UserManager:getInstance()

It is a singleton class and this method returns it's instance.

bool:login(mail, password)

This method handles the login process.

bool:register(User newUser)

This method registers the user to the database.

bool:deleteUser(User user)

This method deletes the user given as parameter. True if the user found, false otherwise.

void:logout()

This method logout the user.

void:updateRecommendedArr(RecommendedList)

This method updates the recommended list of the user.

void:updateNotRecommendedArr(NotRecommendedList)

This method updates the not recommended list of the user.

void:updatePassword(password)

This method updates the current user's password.

Shopping List Manager
This class manages shopping list.
Variables
- shoppingListManager:ShoppingListManager - Product[]:productList - sourceLocation:var
Methods

ShoppingListManager:getInstance()

It is a singleton class and this method returns it's instance.

bool:removeProduct(product):bool

This method removes the product from the database. Returns true if the product is found, false otherwise.

bool:addProduct(product)

This method adds product to the database. True if such product is not

found in database, false otherwise.

bool:updateCurrentProduct(id)

This method updates the last scanned product. True if id is valid, false otherwise.

GPS Manager
This class manages GPS.
Variables
- gpsManager:GPSManager - location:var
Methods

GPSManager:getInstance()

It is a singleton class and this method returns it's instance.

var:getCurrLocation()

This method returns the current location of the user.

var:findClosestMarket(marketName)

This method finds the closest market by given name from the user that is provided by get current location.

var:sendToGoogleMaps(location)

This method send user to the given location.

var:findClosestMarkets()

This method finds the closest all markets.

Sound Manager
This class manages the sound of the system.
Variables
- soundManager:SoundManager - sounds:Audio[] - volumeLevel:int
Methods

SoundManager:getInstance()

It is a singleton class and this method returns it’s instance.

void:playLoginSound()

This method plays login sound.

void:playRegisterSound()

This method plays register sound.

void:playProductSound()

This method plays product found sound.

Vuforia Manager
This class manages the Vuforia methods.
Variables
- vuforiaManager:VuforiaManager - cloudUrl:String - password:String
Methods

VuforiaManager:getInstance()

It is a singleton class and this method returns it’s instance.

void:scanProduct()

This method scans product from the camera.

int:returnFoundProductId()

This method returns found product id.

Application Manager
This class manages the application.
Variables
- applicationManager:ApplicationManager - currentPageIndex:İnt - pages:Page[]

Methods
applicationManager:getInstance() It is a singleton class and this method returns it’s instance. void:changePageByIndex(index) This method changes current page to given index page. int:getCurrentPageIndex() This method returns current page id. String:getPageName(index) This method returns indexed page name.

Application Manager
This class manages the application.
Variables
- applicationManager:ApplicationManager - currentPageIndex:İnt - pages:Page[]
Methods
applicationManager:getInstance() It is a singleton class and this method returns it’s instance. void:changePageByIndex(index) This method changes current page to given index page. int:getCurrentPageIndex() This method returns current page id. String:getPageName(index) This method returns indexed page name.

Page
This class is an abstract class for pages.
Variables

- goBackButton: Button
- coordinate: Point3D
- name: String

Methods

void:goBackPage()

This method closes current page and opens back page.

Login Page extends Page

This class is responsible for the login page.

Variables

- signInButton: Button
- forgotPassButton: Button
- userNameIF: Input Field
- passwordIF: Input Field

Methods

bool:signIn()

This method handles the sign in process.

void:forgotPass()

This method closes current page and opens forgot password page.

About Page extends Page

This class is responsible for the about page.

Variables

- aboutInformationT: Text

Shopping List Page extends Page

This class is responsible for the shopping list page.

Variables

- searchProductIF: Input Field
- addProductButton: Button
- chooseLocation: Dropdown
- showSupermarkets: Button
- shoppingListText: Text

Methods

void:searchProduct()

This method handles the searching product process.

void:addProduct()

This method handles the add product process.

void:showSupermarkets()

This method closes current page and opens shopping list supermarkets page.

Lifestyle Page extends Page

This class is responsible for the lifestyle page.

Variables

- eatingStyle: Dropdown
- stampPreferences: Dropdown
- saveButton: Button
- stampPreferencesT: Text
- eatingStyleT: Text

Methods

void:save()

This method saves users' current choices.

Lifestyle Page extends Page

This class is responsible for the lifestyle page.

Variables

- eatingStyle: Dropdown
- stampPreferences: Dropdown

- saveButton: Button
- stampPreferencesT: Text
- eatingStyleT: Text

Methods

void:save()

This method saves users' current choices.

Register Page extends Page

This class is responsible for the register page.

Variables

- firstNameIF: Input Field
- lastNameIF: Input Field
- emailIF: Input Field
- userNameIF: Input Field
- passwordIF: Input Field
- confirmPassIF: Input Field
- register: Button

Methods

void:register()

This method registers user to the system.

Scan Page extends Page

This class is responsible for the scan product page.

Variables

- productButton: Button
- camera: Vuforia::CameraDevice

Methods

void:goProductInfoPage()

This method sends user to product info page.

void:showProductButton()

This method shows the product.

Shopping List Supermarkets extends Page
This class is responsible for the shopping list supermarkets page.
Variables
- searchSupermarketIF: Input Field - supermarketListT: Text - sortOptions: Dropdown
Methods

void:sort()

This method sorts supermarkets according to sort options.

void:showSupermarkets()

This method shows the supermarkets.

void:update()

This method updates the page after sort options change.

Main Page extends Page
This class is responsible for the main page.
Variables
- aboutButton: Button - guestLoginButton: Button - loginButton: Button - registerButton: Button
Methods

void:goRegister()

This method sends user to Register Page

void:goLogin()

This method sends user to Login Page

void:goGuestLogin()

This method sends user to Guest Login Page

void:goAbout()

This method sends user to About Page

Main Page extends Page
This class is responsible for the main page.
Variables
- aboutButton: Button - guestLoginButton: Button - loginButton: Button - registerButton: Button
Methods

void:goRegister()

This method sends user to Register Page

void:goLogin()

This method sends user to Login Page

void:goGuestLogin()

This method sends user to Guest Login Page

void:goAbout()

This method sends user to About Page

Home Page extends Page
This class is responsible for the main page.
Variables
- scanButton: Button - createShoppingListButton: Button
Methods
void:goScan()

This method sends the user to scan page.

void:goShoppingList()

This method sends the user to shopping list page.

Product Information Page extends Page
This class is responsible for the product information page.
Variables
- ingredientsT: Text - sortOptions: Dropdown - marketsAndPricesT: Text - warningT: Text
Methods

void:sort()

This method sorts markets according to sort option.

Forgot Password Page extends Page
This class is responsible for the forgot password page.
Variables
- mailAddress: Input Field - sendMailButton: Button
Methods

void:sendMail()

This method is responsible for sending rearrange password mail to user.

Allergies Page extends Page
This class is responsible for the allergies page.

Variables
- allergyIF: Input Field - addButton: Button - allergiesT: Text
Methods
void:addAllergy() This method is responsible for adding allergies to the user database.

4. Glossary

AR: Augmented Reality

App: Application

GDPR: General Data Protection Regulation

GPS: Global Positioning System

5. 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.