Bilkent University

Department of Computer Engineering

Senior Design Project

Project short-name: HandsGiving

Low-Level Design Report

Fırat Yönak, Ege Hakan Karaağaç, Oğuzhan Dere, Mehmet Tolga Tomris, Atakan Arslan

Supervisor: Özgür Ulusoy

Low-Level Design Report

February 8, 2021

This report is submitted to the Department of Computer Engineering of Bilkent University in partial fulfillment of the requirements of the Senior Design Project course CS491/2.

Contents

1 Introduction **2**
1.1 Object Design Trade-Offs 2
1.1.1 Functionality vs Usability 2
1.1.2 Extensibility & Modularity vs Performance 3
1.1.3 Space vs Time 3
1.1.4 Robustness vs Cost 3
1.1.5 Compatibility vs Programmability 4
1.2 Interface Documentation Guidelines 4
1.3 Engineering Standards 5
1.4 Definitions, Acronyms, and Abbreviations 5

2 Packages **7**
2.1 Client 7
2.1.1 UI 7
2.1.1.1 Needy 7
2.1.1.2 Volunteer 8
2.1.2 Controller 9
2.1.2.1 Needy 9
2.1.2.2 Benevolent 10
2.2 Server 11
2.2.1 I/O Layer 12
2.2.2 Processor Layer 12
2.2.3 Storage Layer 12

3 Class Interfaces **13**
3.1 Client 13
3.1.1 UI 13
3.1.2 Controller 19
3.2 Server 25

4 References **30**

1 Introduction

With the increase in the number of people in need in the world and the extension of human life, solidarity has become more important. People started to need each other more both materially and spiritually. However, the distance between people and their inability to communicate easily with each other made it more difficult to cooperate. HandsGiving is an application intended to be a solution to all these problems faced by people. It will provide a platform for people in need of help to have their voices heard. Those people who are in need will have the opportunity to express their needs to volunteers through the application. The application will also be designed to reduce the negative effects of the COVID-19 virus on people's social life [1]. As part of COVID-19 measures, elderly people had to be restricted from going out to the streets [2]. Both the elderly who were left alone in their homes due to these restrictions and those who already had fewer people around became more in need for social activities and their needs became more difficult to meet. Thanks to the video chat feature that HandsGiving will offer, people will gain the chance to meet with different volunteers and meet their social needs. HandsGiving is going to be a free application, which has the sole purpose of connecting people of any age with each other for various reasons. The application feeds on the mutual benefits of the communal life, which leads to a win-win situation rather than what most applications were created for. It is also worth mentioning that the scope of this application is even more than social concerns.

1.1 Object Design Trade-Offs

1.1.1 Functionality vs Usability

Two of the most important aspects to consider about design trade-offs are HandsGiving's functionality and usability. Functionality refers to whether Handsgiving's functionalities are working as intended whereas the usability refers to the ease of use and intuitiveness of these presented functions. Even though HandsGiving provides beneficial and practical functionalities, without making them understandable and usable they become worthless. The main group of people who will benefit from the application are needy and mostly old people that's why creating a simple and understandable interface is the most

significant aspect for us. Thus, it is possible to say that our design favors usability over functionality.

1.1.2 Extensibility & Modularity vs Performance

The first versions of applications mostly are a kind of demo and they are updated many times after the first release of the apps. Because of that extensibility and modularity is significant. Without them, the cost of adding new features may be considerably high.

Each of the packages in this project is easily separated from one another to be developed independently, making the overall system quite modular. With modularity, the new features can be added more easily without changing the system, just a few alterations should be enough. Unfortunately, depending on how performance-hungry certain packages are, the overall performance might be hampered.

1.1.3 Space vs Time

One of the main features of the HandsGiving is the video call. It requires constant view transferring between two clients. The transfer process should be less than a second and it should not be noticed by users, so there will be an inevitable trade-off between space and time. To provide a fine video call experience, our primary focus will be response time since it is the key for a good user experience.

1.1.4 Robustness vs Cost

The people who will benefit from HandsGiving are needy and mostly old people that's why they may not solve problems which may occur due to an error in the app easily. Therefore, one of the main purposes of us is to provide a service that is reliable in terms of its outputs and offers full functionality to users without any errors. This aim requires us to benefit from better services rather than its minimal requirements, for example using a more reliable database to store our clients information is better than using a less secure and cheaper one. As a result of preferring better services, the cost will be higher. Despite the higher cost of keeping HandsGiving robust, it is our priority to realize this goal as much as our budget allows.

1.1.5 Compatibility vs Programmability

The low-level design proposed in this report is prepared considering an Android application that is programmed mainly by using Java. As mentioned in the previous stages of the project (with the reports that are previously submitted) and this report, one of the main goals of the application is to help needy people who suffer from economic problems. Due to these issues, we decided to keep the minimum Android version as little as possible in order to meet the needings of many more people. Because of this decision, some of the core functionalities provided by the operating system may not be used which are provided for the higher versions of the Android platform. This fact has a negative effect on the programmability of the application but has a positive effect on the compatibility of the application.

1.2 Interface Documentation Guidelines

Class	ClassName
Class description	
Attributes	
attribute1	Type - Attribute Description
attribute2	Type - Attribute Description
attribute3	Type - Attribute Description
Methods	
method1(param1,param2)	Method description
method2(param1,param2)	Method description

Table 1: Interface Documentation Guidelines

1.3 Engineering Standards

In our reports, we based on the Unified Modeling Language (UML) standard [3] for modelling class interfaces and interactions, the former for object, entity, package and deployment diagrams, and the latter for operation and sequence diagrams. References are made using the IEEE standard [4].

1.4 Definitions, Acronyms, and Abbreviations

App: Stands for 'application'. In this report's context, an application is a smartphone program that is designed in various programming languages for various purposes. Such programs are reachable through 'application stores' of the corresponding smartphone.

HandsGiving: The name of the application. There were various other ideas such as HelpOthers or Help.me, yet the team chose this one.

Request: A request is a demand of someone for asking help for the things which cannot afford or do.

Social Platform: The social platform is a part of the application which provides opportunity to the users to socialize with other users.

Video Chat: It is a communication way in the social platform of the application. By using this feature the users can talk and see each other at the same time.

Feedback: It is the evaluation process of the help process by the involved users. With this feature, the users can rate each other and comment about other users. The application uses these evaluations to enable the benevolents to participate in physical help and also the significant amount of negative feedback may result in the deletion of the user account.

Client: A user of the application. Then, the client side of HandsGiving is the interface in which a benevolent person can respond to help requests, take video calls and a needy person and an old person can create help requests and use the social platform .

Server: The back-end of the application. This is where information about the requests, feedback and accounts are kept.

Account: The personal data of a user such as completed help processes, their username, password and location information is the contents of their account. An account can be created and be logged into inside the HangsGiving app.

Needy: It is the word to describe the users who use the application to get help for the things which they cannot afford or do.

Benevolent: It is the word to describe the users who use the application to help needy people.

Correction Process: This is the process which is used to be sure the email address provided by the users actually belongs to them.

API: Is the abbreviation for the term Application Programming Interface. The term can be defined as an interface to an application component that is designed to ease the usage of such components by other developers than the owner.

UML: Unified Modelling Language. A standardized modeling language consisting of an integrated set of diagrams, developed to help system and software developers for specifying, visualizing, constructing, and documenting the artifacts of software systems[5].

2 Packages

2.1 Client

2.1.1 UI

The UI package manages the app's menus and screens. UIMainControl class controls all the UI operations in the application.

2.1.1.1 Needy

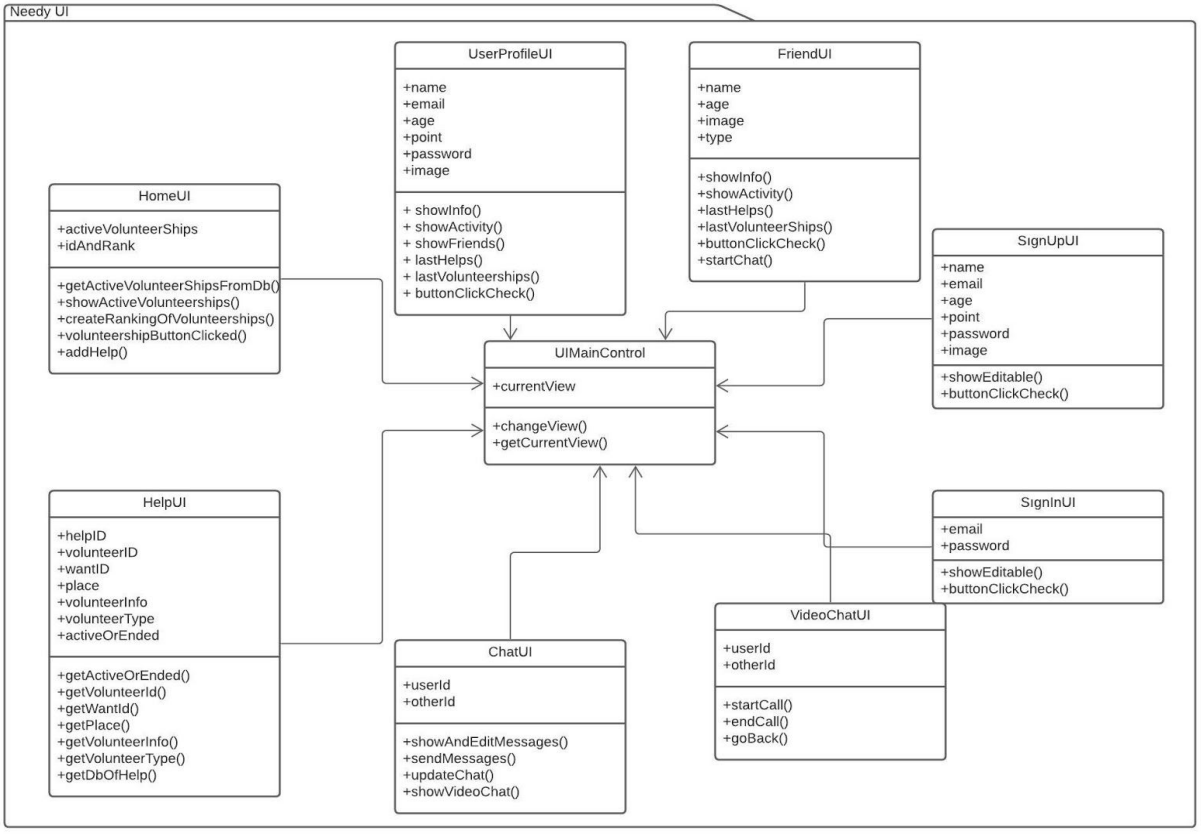

Figure 1: Needy UI package

UIMainControl: The main control part of the UI which controls all the views. Sends requests to UIRequestHandler that come from the views parts and changes view according to the response.

SignInUI: Authentication view for sign in.

SignUpUI: Authentication view for sign up.

HomeUI: View of the home page of the needy person, can add a help request in this view.

HelpUI: View of Help Requests of the needy people.

ChatUI: View of chat and video call operations of the client.

FriendUI: View of all of the friends the user has.

VideoChatUI: View of video chat among the users.

UserProfileUI: View of the profile of users. Shows the properties of a user account.

2.1.1.2 Volunteer

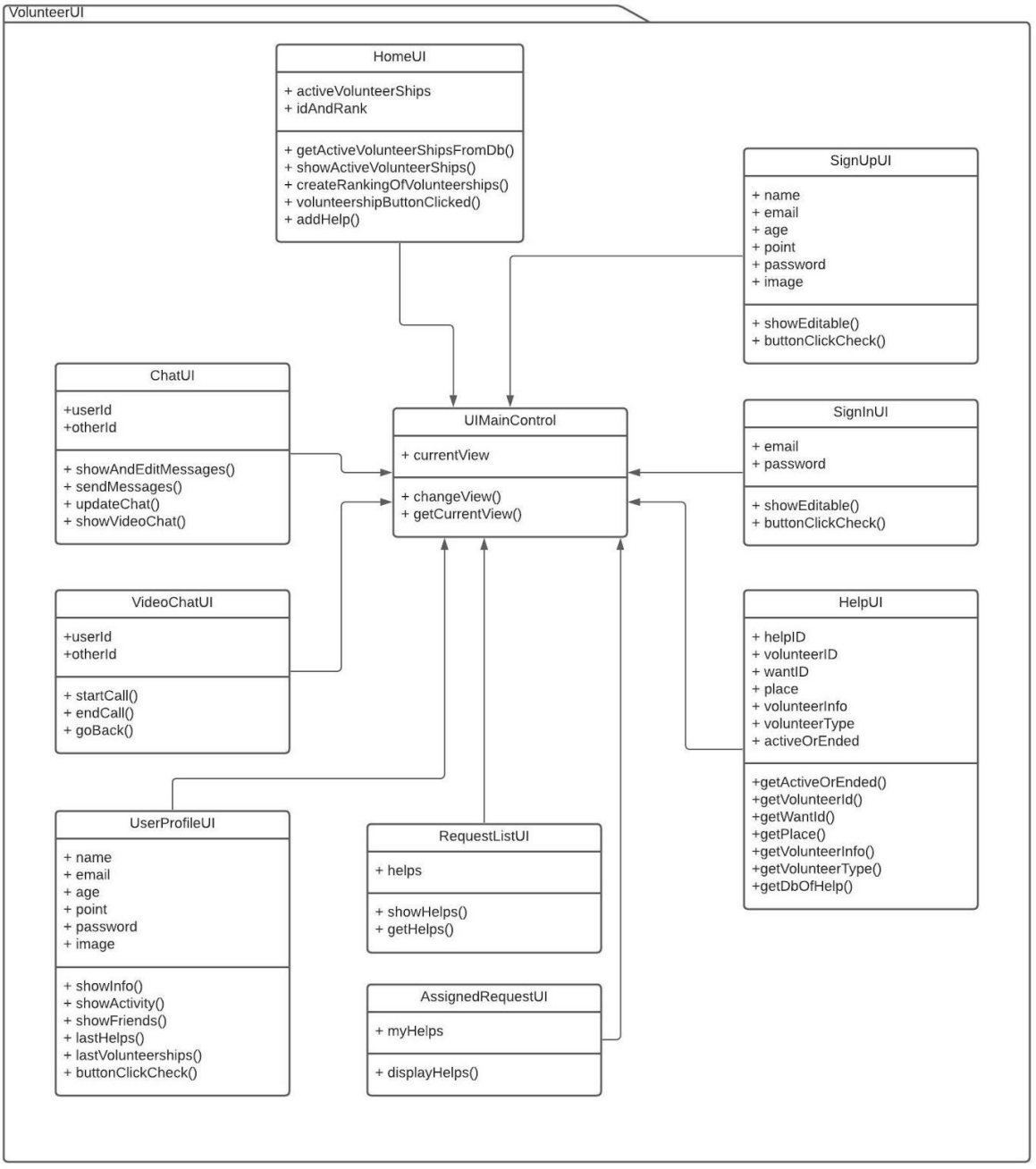

Figure 2: Volunteer UI package

UIMainControl: The main control part of the UI which controls all the views. Sends requests to UIRequestHandler that come from the views parts and changes view according to the response.

SignInUI: Authentication view for sign in.

SignUpUI: Authentication view for sign up.

HomeUI: View of the home page of the volunteers.

HelpUI: View of the completed help process of the user.

ChatUI: View of chat and video call operations of the client.

VideoChatUI: View of video chat among the users.

UserProfileUI: View of the profile of users. Shows the properties of a user account.

RequestListUI: View of the unaccepted help requests of needy people.

AssignedRequestUI: View of the assigned requests to the user.

2.1.2 Controller

2.1.2.1 Needy

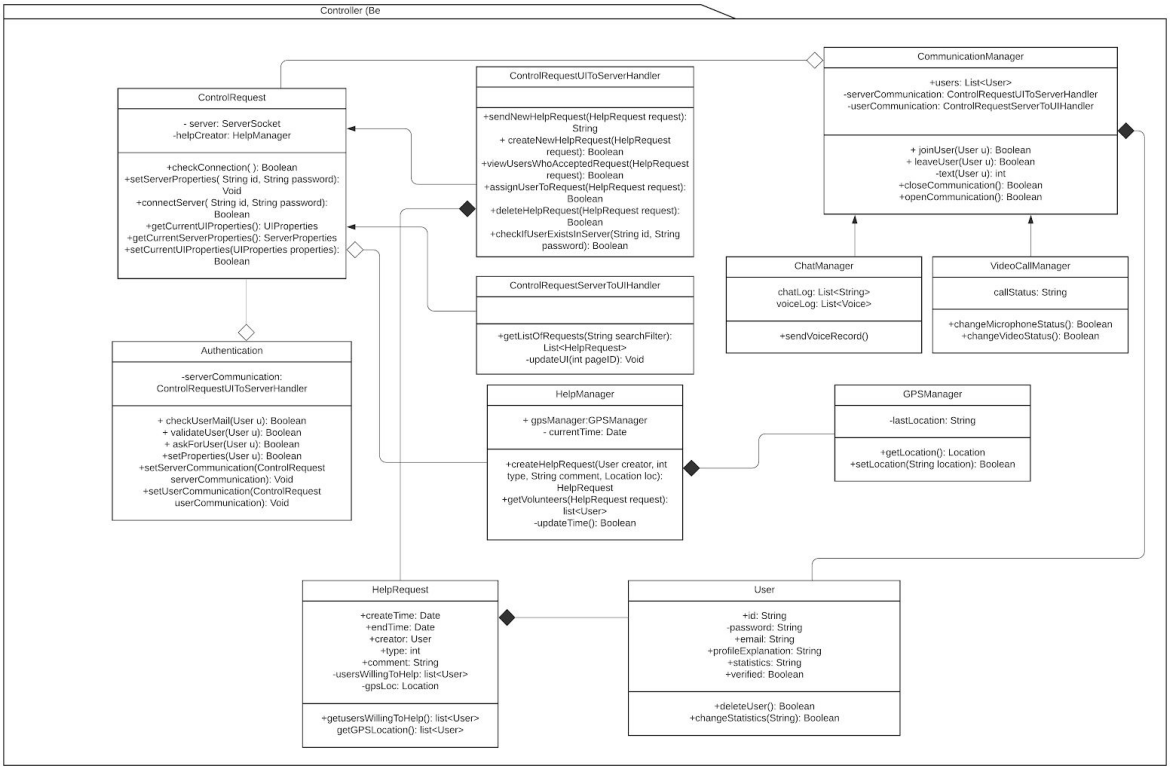

Figure 3: Needy controller package

ControlRequest: Abstract parent class for UI-Server and Server-UI handler classes.

ControlRequestUIToServerHandler: Child class of ControlRequest which handles UI to Server communication.

ControlRequestServerToUIHandler: Child class of ControlRequest which handles Server to UI communication.

Authentication: Class which has a function of authenticating the logged in user.

HelpRequest: Class modelization of the help requests created by users.

User: Class modelization of a single user.

HelpManager: Helper class for creating and modifying help requests created by users.

GPSManager: Class for handling GPS related issues, such as getting the location of a user.

CommunicationManager: Abstract class which is inherited by ChatManager and VideoCallManager.

ChatManager: Child class of CommunicationManager which handles chatting functionalities between users.

VideoCallManager: Child class of CommunicationManager which handles video calling functionalities between users.

2.1.2.2 Benevolent

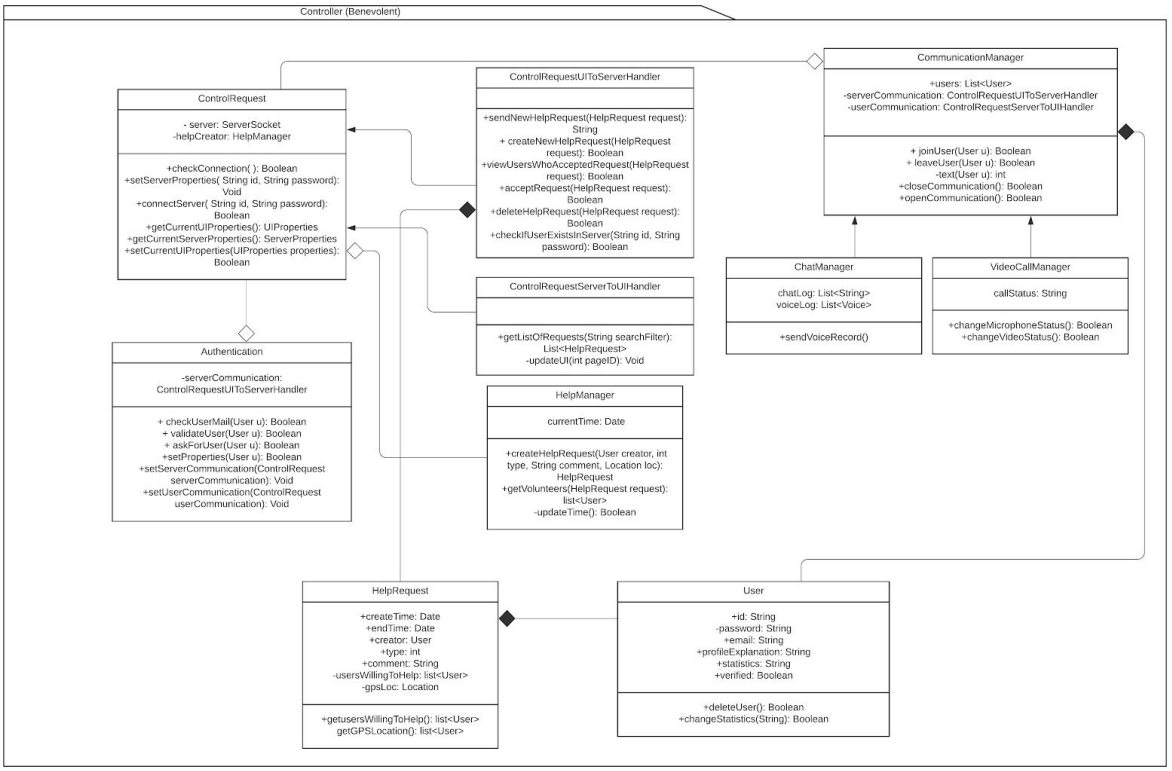

Figure 4: Benevolent controller package

ControlRequest: Abstract parent class for UI-Server and Server-UI handler classes.
ControlRequestUIToServerHandler: Child class of ControlRequest which handles UI to Server communication.
ControlRequestServerToUIHandler: Child class of ControlRequest which handles Server to UI communication.

Authentication: Class which has a function of authenticating the logged in user.
HelpRequest: Class modelization of the help requests created by users.
User: Class modelization of a single user.
HelpManager: Helper class for modifying help requests created by users.
CommunicationManager: Abstract class which is inherited by ChatManager and VideoCallManager.
ChatManager: Child class of CommunicationManager which handles chatting functionalities between users.
VideoCallManager: Child class of CommunicationManager which handles video calling functionalities between users.

2.2 Server

The server part of the project stores all the data and manages the requests that come from the client side of the project. The server creates help requests, authenticates the user, gets information of the users, and manages chat feature by connecting to the MySQL database and creating appropriate queries. The connection between client side and server side is handled by a REST API which gets the the according information from client side and sends a response back. The server side has 3 main packages: I/O, Processor and Data_Storage. I/O package is responsible for the connection between client side and server side with the REST API. Processor layer handles the authentication, chat, help requests, and user information by creating propper requests to the Data_Storage part of the project. Data_Storage part handles the connection of the database and creates queries to update or receive information from the wanted table.

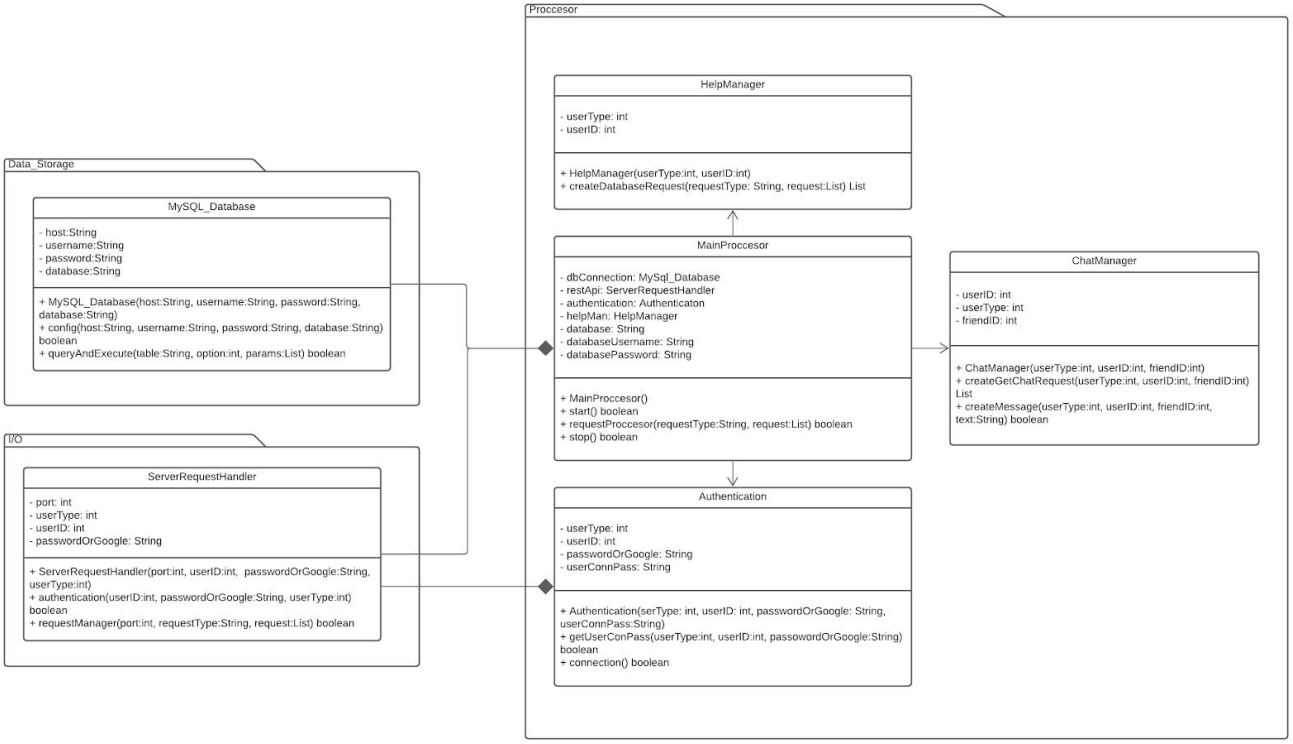

Figure 5:Server side of the HandsGiving

2.2.1 I/O Layer

This is the subsystem that is the server's gateway to the outside world.

ServerRequestHandler: Provides a REST API to receive client requests and deliver them to the processor layer to get responses to requests.

2.2.2 Processor Layer

Authentication: This module authenticates the information of the users while they are accessing the server.
HelpManager: This module processes help requests of the users.
Main Processor: This module gets the request from the I / O layer and organizes the processor layer to provide the required data to the clients.
ChatManager: This module provides a chat feature to the users.

2.2.3 Storage Layer

The storage layer deals with the file database storage of the server.

MySQL_Database: MySQL database is used for storing the information of all users in the related tables.

3 Class Interfaces

3.1 Client

3.1.1 UI

Class	UIMainControl
The main control part of the UI which controls all the views.	
Attributes	
currentView	View - The current view of UI
Methods	
changeView(View view)	Changes the current view of UI.
getCurrentView()	Gets the current view of UI.

Table 2: UIMainControl Class Interface

Class	SignUpUI
Authentication view for sign up.	
Attributes	
email	String- User's Email
name	String - User's Name
password	String - User's Password
age	int - User's Age
point	int - User's Point

image	Image - User’s Image
Methods	
showEditable()	Shows edit button.
buttonClickCheck(Button button)	Checks whether the button is clicked.

Table 3: SignUpUI Class Interface

Class	SignInUI
Authentication view for sign in.	
Attributes	
email	String- User’s Email
password	String - User’s Password
Methods	
showEditable()	Shows edit button.
buttonClickCheck(Button button)	Checks whether the button is clicked.

Table 4: SignInUI Class Interface

Class	HomeUI
View of the home page of the users.	
Attributes	
activeVolunteerships	ArrayList - User’s Active Volunteership
idAndRank	String - User’s ID and Rank
Methods	
getActiveVolunteershipsFromDb()	Gets the active volunteerships from the database.
showActiveVolunteerships()	Shows active volunteerships.
createRankingOfVolunteerShips(User user)	Creates ranking of volunteerships.
volunteershipButtonClicked(Button button)	Checks whether the volunteership button is clicked.

addHelp(Help help)	Adds help.

Table 5: HomeUI Class Interface

Class	RequestListUI
View of the unaccepted help requests of needy people.	
Attributes	
helps	ArrayList - Unaccepted help requests
Methods	
showHelps()	Shows the help requests.
getCurrentView()	Gets the help requests.

Table 6: RequestListUI Class Interface

Class	AssignedRequestUI
View of the assigned requests to the user.	
Attributes	
myHelps	ArrayList - Accepted help requests
Methods	
displayHelps()	Displays the help requests.

Table 7: AssignedRequestUI Class Interface

Class	HelpUI
View of the completed help process of the user.	
Attributes	
helpID	int - id of the help
volunteerID	int - id of the volunteer
wantID	int - id of the needy
place	Location - location of the help
volunteerInfo	String - information about the volunteer
volunteerType	String - type of the volunteer
activeOrEnded	boolean - check whether the help is active or ended
Methods	
getActiveOrEnded()	Get information about the help whether it is activated or ended.
getVolunteerId()	Get the id of a volunteer.
getWantId()	Get id of the needy.
getVolunteerInfo()	Get information about volunteer.
getVolunteerType()	Get volunteer type.
getDbOfHelp()	Get help from database.

Table 8: HelpUI Class Interface

Class	ChatUI
View of chat and video call operations of the client.	
Attributes	
userId	int - id of the first user in chat
otherId	int - id of the second user in chat
Methods	
showAndEditMessages()	Show the messages and edit them.
sendMessages(String message)	Send message to second user in chat.
updateChat()	Update view of the chat
showVideoChat()	Show video chat properties..

Table 9: ChatUI Class Interface

Class	VideoChatUI
View of video chat among the users.	
Attributes	
userId	int - id of the first user in chat
otherId	int - id of the second user in chat
Methods	
startCall(User user1, User user2)	Start video call.
endCall()	End video call.
goBack()	Go back to chat screen.

Table 10: VideoChatUI Class Interface

Class	FriendUI
View of all of the friends the user has.	
Attributes	
name	String- User's name
age	int - User's age
image	Image - User's image
type	String - User's type
Methods	
showInfo()	Show information about friends
showActivity()	Show activities of friends
lastHelps()	Show last helps of friends
lastVolunteerships()	Show last volunteerships of friends
buttonClickCheck(Button button)	Check whether the button is clicked.
startChat(User, user1, User user2)	Start chat with friends

Table 11: FriendUI Class Interface

Class	UserProfileUI
View of the profile of users. Shows the properties of a user account.	
Attributes	
email	String- User's Email
name	String - User's Name
password	String - User's Password
age	int - User's Age
point	int - User's Point
image	Image - User's Image
Methods	

showInfo()	Show information about the user.
showActivity()	Show activities of the user.
showFriends()	Show friends of the user.
lastHelps()	Show last helps done by the user.
lastVolunteerships()	Show last volunteerships of the user.
buttonClickCheck(Button button)	Check whether the button is clicked.

Table 12: UserProfileUI Class Interface

3.1.2 Controller

Class	ControlRequest
Abstract parent class for UI-Server and Server-UI handler classes.	
Attributes	
server	ServerSocket - Server socket instance
helpCreator	HelpManager - For creating the HelpRequest instances.
Methods	
checkConnection()	Checks server-app connection.
setServerProperties()	Sets server properties (mostly authentication related properties).
connectServer()	Tries to connect the server.
getCurrentUIProperties()	Acquires UI information from the related UI class.
getCurrentServerProperties()	Acquires current server information from the server if it is possible.
setCurrentUIProperties()	Gives updated UI information to the related UI class.

Table 13: ControlRequest Class Interface

Class	ControlRequestUIToServerHandler (Benevolent)
Child class of ControlRequest which handles UI to Server communication.	
Methods	
sendNewHelpRequest(HelpRequest request)	Sends a newly created help request to the server.

createNewHelpRequest(HelpRequest request)	Creates a help request, by using HelpManager.
viewUsersWhoAcceptedRequest(HelpRequest request)	Views users who accepted the help request.
acceptRequest(HelpRequest request)	Let a user accept a request.
deleteHelpRequest(HelpRequest request)	Deletes a help request from the server. This function will be written such that the request will not be totally gone from the records, for safety reasons.
checkIfUserExistsInServer(String id, String password)	Checks if a user exists on the server.

Table 14: ControlRequestUIToServerHandler Class Interface(Benevolent)

Class	ControlRequestUIToServerHandler (Needy)
Child class of ControlRequest which handles UI to Server communication.	
Methods	
sendNewHelpRequest(HelpRequest request)	Sends a newly created help request to the server.
createNewHelpRequest(HelpRequest request)	Creates a help request, by using HelpManager.
viewUsersWhoAcceptedRequest(HelpRequest request)	Views users who accepted the help request.
assignUserToRequest()	Chooses and assigns a user to a request.
deleteHelpRequest(HelpRequest request)	Deletes a help request from the server. This function will be written such that the request will not be totally gone from the records, for safety reasons.
checkIfUserExistsInServer(String id, String password)	Checks if a user exists on the server.

Table 15: ControlRequestUIToServerHandler Class Interface(Needy)

Class	ControlRequestServerToUIHandler
Child class of ControlRequest which handles Server to UI communication.	
Methods	
getListOfRequests(String filter)	Returns a list of requests to the user with the given filter.
updateUI(int pageID)	updates UI by using the related UI class.

Table 16: ControlRequestServerToUIHandler Class Interface

Class	Authentication
Class which has a function of authenticating the logged in user.	
Attributes	
serverCommunication	ControlRequestUIToServerHandler - handles GUI-to-Server communication during authentication
Methods	
checkUserMail(User u)	checks if user mail is verified
validateUser(User u)	checks if the account is verified.
askForUser(User u)	asks the user to verify components that require verifying.
setProperties(User u)	sets the properties of a user
setServerCommunication(ControlRequest serverCommunication)	sets server communication objects.
setUserCommunication(ControlRequest userCommunication)	sets user communication.

Table 17: Authentication Class Interface

Class	HelpRequest
Class modelization of the help requests created by users.	
Attributes	
createTime	Date - The request creation time.
endTime	Date - The request
creator	User - Creator of request
type	int - Type of request.
comment	String - i.e. explanation created by the creator user.
usersWillingToHelp	list<User> - List of users willing to help the case.
gpsLoc	Location - Location information from the creator user.
Methods	
getusersWillingToHelp()	Returns users willing to help.
getGPSLocation()	Returns GPS location.

Table 18: HelpRequest Class Interface

Class	User
Class modelization of a single user.	
Attributes	
id	String - id of user
password	String - password of user
email	String - email of user
profileExplanation	String - profile explanation for the user
statistics	String - user statistics
verified	Boolean - if user is verified or not
Methods	
deleteUser()	deletes the logged in user
changeStatistics(String newStatistics)	changes the statistics of the user

Table 19: User Class Interface

Class	HelpManager
Helper class for modifying help requests created by users.	
Attributes	
currentTime	Date - holds current time
Methods	
createHelpRequest(User creator, int type, String comment, Location loc)	Creates a new help request, return a HelpRequest object
getVolunteers(HelpRequest request)	gets volunteers of the request that is given as a parameter.
updateTime()	updates time, returns true if it is successful

Table 20: HelpManager Class Interface

Class	GPSManager
Class for handling GPS related issues, such as getting the location of a user.	
Attributes	
lastLocation	String/Location - holds the current location of the user that is acquired from the last use of getLocation()
Methods	
getLocation()	gets the location of the user using GPS.
setLocation(String location)	sets the location of the user as the given parameter.

Table 21: GPSManager Class Interface

Class	CommunicationManager
Abstract class which is inherited by ChatManager and VideoCallManager.	
Attributes	
users	
serverCommunication	ControlRequestUIToServerHandler - handles GUI-to-Server communication during authentication
userCommunication	ControlRequestServerToUIHandler - handles Server-to-GUI communication during authentication
Methods	
joinUser(User u)	adds a user to the chat/video call.
leaveUser(User u)	removes a user from the chat/video call.
text(User u, String text)	lets a user text and sends the text to the related UI class.
closeCommunication()	closes the communication between users.
openCommunication()	opens the communication between users.

Table 22: CommunicationManager Class Interface

Class	ChatManager
Child class of CommunicationManager which handles chatting functionalities between users.	
Attributes	
chatLog	List<String> - holds the chat log
voiceLog	List<Voice> - holds the voice log
Methods	
sendVoiceRecord(Voice v)	lets the user send a voice record.

Table 23: ChatManager Class Interface

Class	VideoCallManager
Child class of CommunicationManager which handles video calling functionalities between users.	
Attributes	
callStatus	String - holds the call state
Methods	
changeMicrophoneStatus()	Boolean - changes microphone status, on or off.
changeVideoStatus()	Boolean - changes video status, on or off.

Table 24: VideoCallManager Class Interface

3.2 Server

Class	MySQL_Database
This class communicates with the database to save, change, access to data.	
Attributes	
host	String – holds the name of the connected device
username	String – holds the name of the user
password	String – holds the password of the user
database	String – holds the name of database
Methods	
MySQL_Database(host:String, username:String,password:String,database:String)	Constructor
config(host:String, username:String, password:String,database:String)	connects the server to the database
queryAndExecute(table:String, option:int, params:List)	executes the queries

Table 25: MySQL_Database Class Interface

Class	ServerRequestHandler
This class handles the input coming from the user.	
Attributes	
port	int – holds the port number

userType	int – holds number according to the type of the user
userID	int – holds the id of the user
passwordOrGoogle	String – holds the password or the Google account of the user
Methods	
ServerRequestHandler(port:int, userID:int, passwordOrGoogle:String, userType:int)	Constructor
authentication(userID:int, passwordOrGoogle:String,userType:int)	authenticates the user if there is a user with given information.
requestManager(port:int, requestType:String, request:List)	creates help request.

Table 26: ServerRequestHandler Class Interface

Class	HelpManager
Class for creating help requests into the database.	
Attributes	
userType	int – holds the userType
userID	int – holds the id of the user
Methods	
HelpManager(userType:int, userID:int)	Constructor
createDatabaseRequest(requestType:String, request:List)	creates help request into the database.

Table 27: HelpManager Class Interface

Class	MainProcessor
Class for managing all operations of the application	
Attributes	
dbConnection	MySQL_Database - Instance of MySQL_Database
restApi	ServerRequestHandler – Instance of ServerRequestHandler
authentication	Authentication – Instance of Authentication
helpMan	HelpManager – Instance of HelpManager
database	String – holds name of the database
databaseUsername	String – holds name of the user
databasePassword	String – holds password of the user
Methods	
MainProcessor()	Constructor
start()	starts the server services
requestProcessor(requestType: String, request:List)	creates requests
stop()	stops the server services

Table 28: MainProcessor Class Interface

Class	Authentication
Class for authenticating users	
Attributes	
userType	int - holds the userType
userID	int - holds the id of the user
passwordOrGoogle	String - holds the password or the Google account of the user
userConnPass	String – holds the user connection pass.
Methods	
Authentication(userType:int, userID: int, passwordOrGoogle: String, userConnPass: String)	Constructor
getUserConnpass(userType:int, userID:int, passwordOrGoogle:String)	Returns the userConnPass
connection()	connects the client and the server

Table 29: Authentication Class Interface

Class	ChatManager
Class for managing the chat process	
Attributes	
userID	int - holds the id of the user
userType	int - holds the userType
friendID	int – holds the ID of the friend
Methods	
ChatManager(userType:int, userID:int, friendID:int)	Constructor
createGetChatRequest(userType:int, userID:int, friendID:int)	creates chat
createMessage(userType:int, userID:int, friendID:int, text:String)	Creates message from the user to the friend

Table 30: ChatManager Class Interface

4 References

[1] Li, Sijia, et al. "*The Impact of COVID-19 Epidemic Declaration on Psychological Consequences: A Study on Active Weibo Users.*" *MDPI*, Multidisciplinary Digital Publishing Institute, 19 Mar. 2020.[Online]. Available: www.mdpi.com/1660-4601/17/6/2032/htm. [Accessed: 4- Feb-2021].

[2] Hürriyet." *65 Yaş Üstü Ne Zaman Sokağa Çıkacak?".*[Online]. Available: www.hurriyet.com.tr/galeri-65-yas-ustu-ne-zaman-sokaga-cikacak-41503410/2. [Accessed: 4- Feb-2021].

[3] "Unified Modeling Language". [Online]. http://www.uml.org/. [Accessed: 4- Feb-2021].

[4] IEEE REFERENCE GUIDE. 2020. [Online]. https://ieeeauthorcenter.ieee.org/wp-content/uploads/IEEE-Reference-Guide.pdf. [Accessed: 4- Feb-2021].

[5] "Unified modelling language." https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-uml/. [Accessed: 5- Feb- 2021]