Garcon

Software Design Description

Ramazan Selim Şahin 2171999

Gürkan Kısaoğlu 2171726

Table of Contents

List of Figures

List of Tables

1. Introduction

1.1 Purpose of the System

The purpose of the project is making METU campus smarter than ever. Making the campus an interactive environment for both students and workers. When there is a campus wide security issue, or environment issue students can immediately inform workers of campus with this system. In addition, it enables students to get information about campus transportation and food possibilities.

1.2 Scope

The scope of this project is providing users to talk interactively with campus workers or
gathering information about campus. To accomplish this task an embedded system will be developed. Two potential groups of users exists:
● Students that wants to gather information about campus or open a ticket about a security or environment issue.
● Workers which is a group of campus employees that can see and close issues opened
by students.
Garcon makes campus an interactive and informative environment with functionalities
like gathering transportation data and event data around campus, opening security or
cleaning issues and ordering food.
Therefore, the software has four main
products:
● Mail Service
● Server
● Speech to Text Service
● Third party software called Yemeksepeti API
By this software, users will be able to talk with this system to gather information instead of searching on the web or calling some people.

1.3 Stakeholders and Their Concerns

User: Users are only University students, who can access the system with their ID Cards. Their primary concern is getting campus information no matter where they are and giving feedbacks to relevant persons via opening issues to make campus a better place.

Worker: Workers are University employees who can access the system with their ID Cards just like Users. Their concern is solving issues opened by the Users.

Admin: Admins are the people who can access the system via a web interface. Their primary concern is manage the system users(students, workers).

2. References

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

3. Glossary

Term	Definition
User	The end user(student) who is interacting with Garcon to query for information.
Worker	Campus personnel that uses Garcon for closing issues.
Admin	The most privileged person who registers new users/workers to the system and deletes them. Also responsible from maintenance.
Yemeksepeti	An online food ordering company working with Garcon Project.
Yemeksepeti API	An external API which provides ordering food functionality.
SATA	Serial AT Attachment
CRUD	Create, read, update, delete operations of persistent Storage.
DB	Database
API	Application programming interface.
Speech to text Service	A service that can analyse the speech and decide whether it is about an issue or is an information request

Table 0: Glossary

4. Architectural Views

4.1 Context View

In this viewpoint, all system use cases and descriptions explained with details. These descriptions show how system should behave in specific situations and how the functionalities implemented. Context diagram shows how users and other subsystems interacts with each other and use case diagram and functionality descriptions specified below the context diagram.

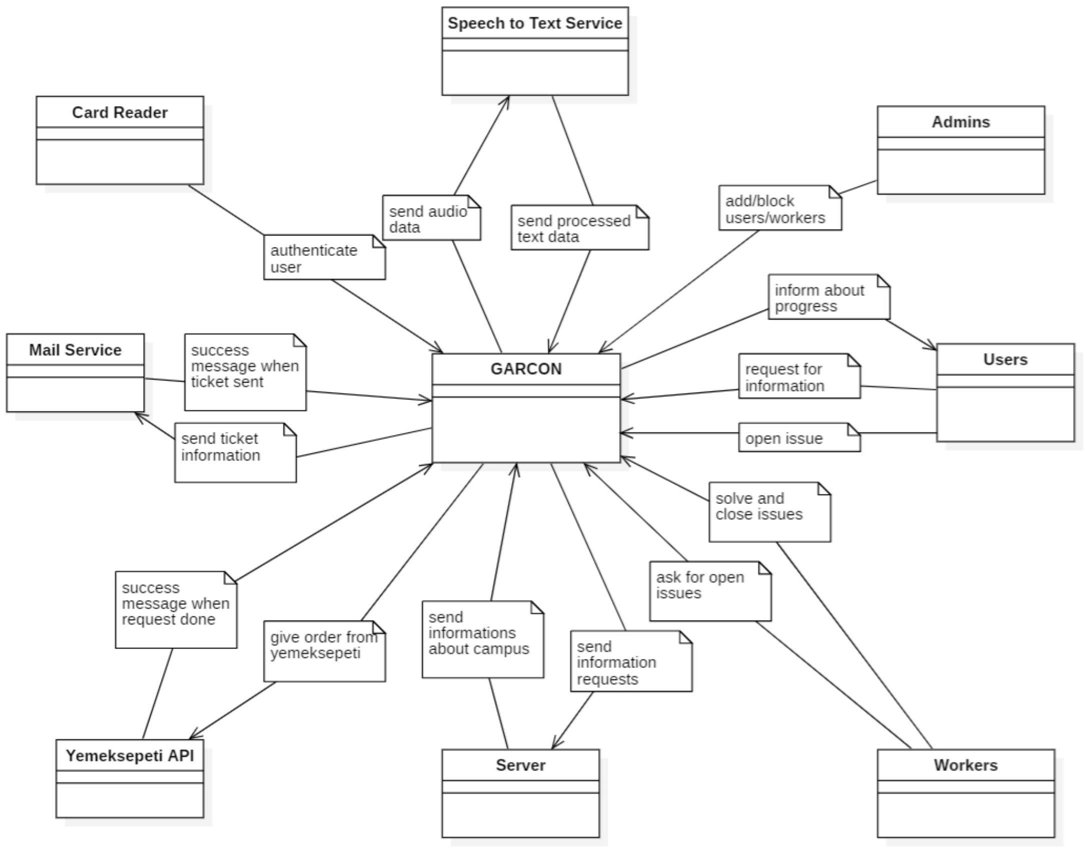


Figure 1: Context Diagram

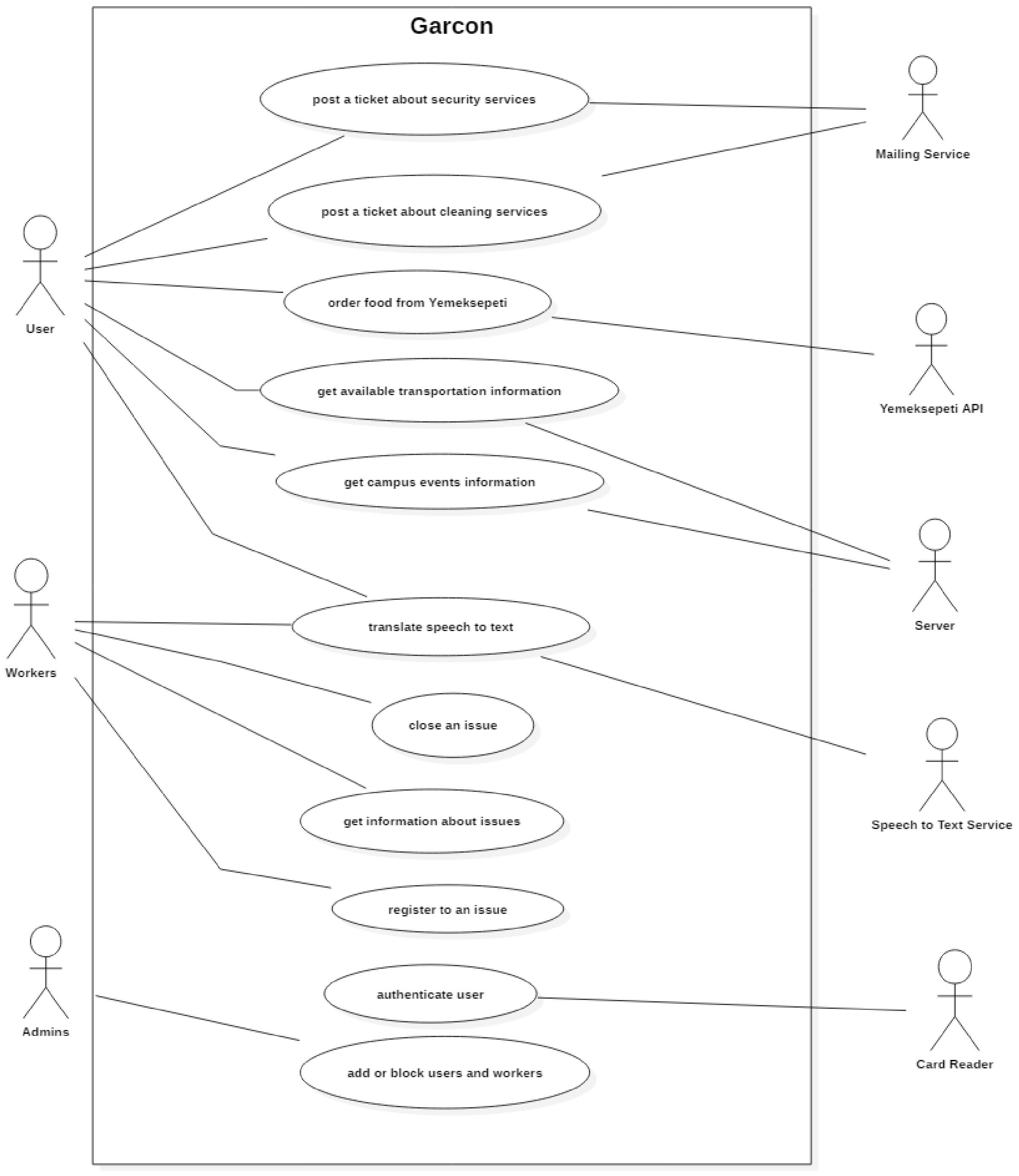

Garcon
post a ticket about security services
post a ticket about cleaning services
order food from Yemeksepeti
get available transportation information
get campus events information
translate speech to text
close an issue
get information about issues
register to an issue
authenticate user
add or block users and workers
User
Workers
Admins
Mailing Service
Yemeksepeti API
Server
Speech to Text Service
Card Reader

Figure 2: Use Case Diagram

Use case name	Issue about security service
Actors	Users, Speech to Text Service, Mailing Service, Server
Description	If a user notices a security issue he/she can notify related workers via Garcon.
Data	Audial input from user
Preconditions	User should be authenticated
Stimulus	User giving audial input about issuing a security request
Basic Flow	Step 1 – User gives audial input Step 2 – Audial input gets processed by Speech to Text Service Step 3 – Issue is created and server side is informed Step 4 – Processed text of audial input is sent as an email to the related workers Step 5 – User is informed that issue is registered
Alternative Flow	Step 3 – System detects same request already issued Step 4 – Importance level of request is updated Step 5 – User is informed that issue is already registered
Exception Flow	-
Post conditions	An issue instance is created on system and related workers are informed.

Table 1: Open security issue function

Use case name	Issue about cleaning service
Actors	Users, Speech to Text Service, Mailing Service, Server
Description	If a user notices a cleaning issue he/she can notify related workers via Garcon.
Data	Audial input from user
Preconditions	User should be authenticated
Stimulus	User giving audial input about issuing a cleaning request
Basic Flow	Step 1 – User gives audial input Step 2 – Audial input gets processed by Speech to Text Service Step 3 – Issue is created and server side is informed Step 4 – Processed text of audial input is sent as an email to the related workers Step 5 – User is informed that issue is registered
Alternative Flow	Step 3 – System detects same request already issued Step 4 – Importance level of request is updated Step 5 – User is informed that issue is already registered
Exception Flow	-
Post conditions	An issue instance is created on system and related workers are informed.

Table 2: Open cleaning issue function

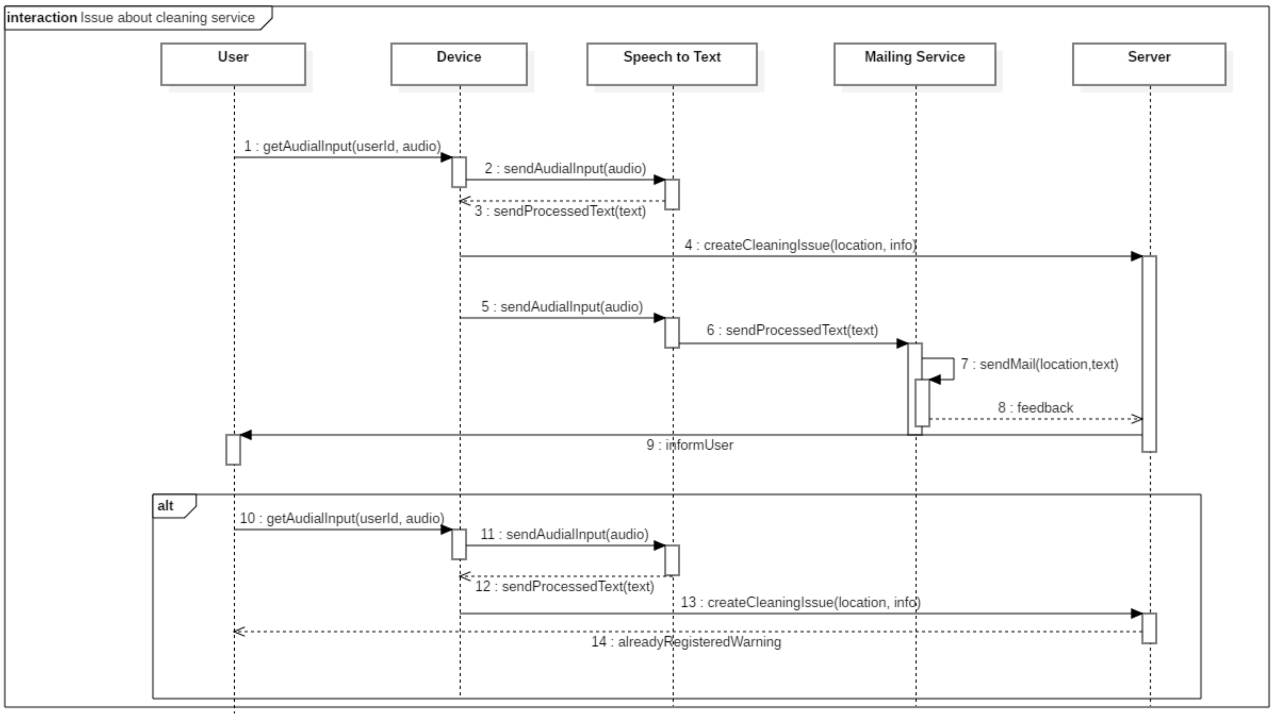

interaction Issue about cleaning service
User
Device
Speech to Text
Mailing Service
Server
1 : getAudialInput(userId, audio)
2 : sendAudialInput(audio)
3 : sendProcessedText(text)
4 : createCleaningIssue(location, info)
5 : sendAudialInput(audio)
6 : sendProcessedText(text)
7 : sendMail(location,text)
8 : feedback
9 : informUser
alt
10 : getAudialInput(userId, audio)
11 : sendAudialInput(audio)
12 : sendProcessedText(text)
13 : createCleaningIssue(location, info)
14 : alreadyRegisteredWarning

Figure 3: Issue about cleaning service Sequence Diagram

Use case name	Order food from Yemeksepeti
Actors	Users, Speech to Text Service, Yemeksepeti API
Description	When user asks for ordering food, the request translated into text first, gets analyzed and then system automatically give an order from Yemeksepeti
Data	Audial input from user
Preconditions	User should be authenticated
Stimulus	User giving audial input about ordering food
Basic Flow	Step 1 – User gives audial input Step 2 – Audial input gets processed by Speech to Text Service Step 3 – A request is posted to Yemeksepeti Api Step 4 – Success message shown to the user
Alternative Flow	-
Exception Flow	If any error occurs or restaurant is closed , system will show a log message.
Post conditions	User gives an order from Yemeksepeti.

Table 3: Order food function

Use case name	Get transportation information
Actors	Users, Speech to Text Service, Server
Description	Information service on transformation info. User can get a possible route to a direction from the device he/she is now interacting with and user can get information about transportation schedules.
Data	Audial input from user, current transportation services' data, map data
Preconditions	Worker should be authenticated
Stimulus	User giving audial input about getting informed on transportation
Basic Flow	Step 1 – User gives audial input asking directions Step 2 – Audial input gets processed by Speech to Text Service Step 3 – Best possible route and its transformation information is obtained from the server Step 4 – Information is converted to audio form by Speech to Text Service Step 5 – User is informed
Alternative Flow	Step 1 – User gives audial input asking transportation schedules Step 2 – Audial input gets processed by Speech to Text Service Step 3 –Transformation information on ring, bus, subway services is obtained from the server Step 4 – Information is converted to audio form by Speech to Text Service Step 5 – User is informed
Exception Flow	-
Post conditions	User is informed with best routes and transportation information and information about this query is saved to database to inform further queries faster.

Table 4: Get transportation info function

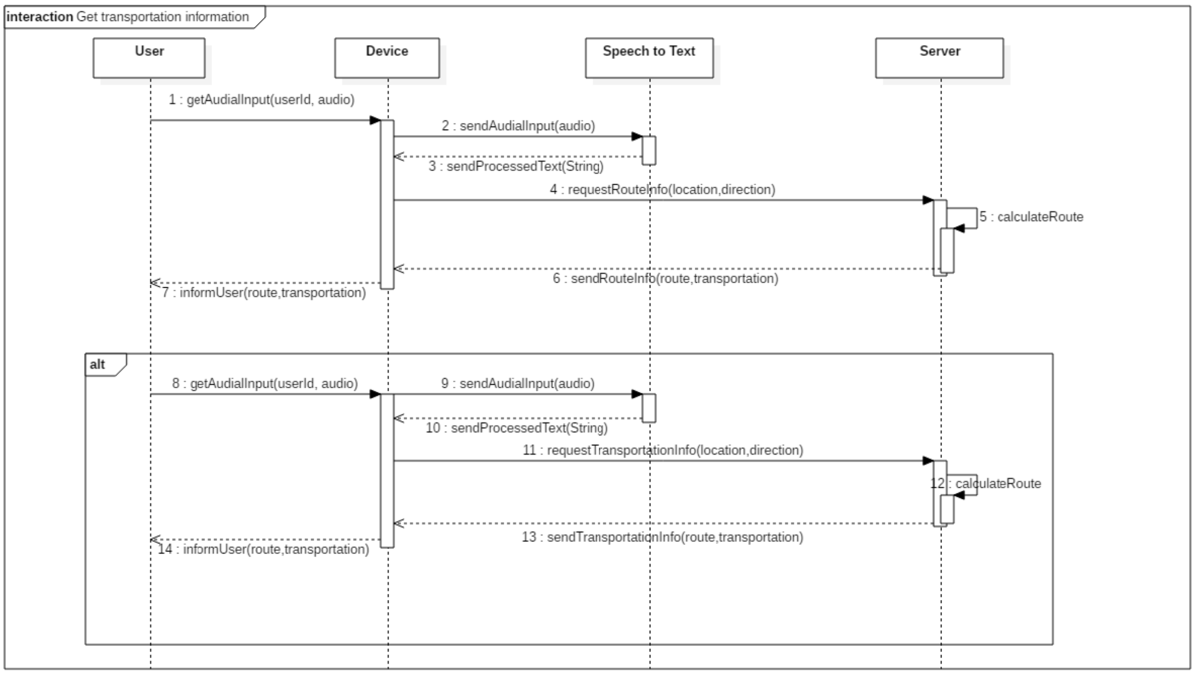

interaction Get transportation information
User
Device
Speech to Text
Server
1 : getAudialInput(userId, audio)
2 : sendAudialInput(audio)
3 : sendProcessedText(String)
4 : requestRouteInfo(location,direction)
5 : calculateRoute
6 : sendRouteInfo(route,transportation)
7 : informUser(route,transportation)
alt
8 : getAudialInput(userId, audio)
9 : sendAudialInput(audio)
10 : sendProcessedText(String)
11 : requestTransportationInfo(location,direction)
12 : calculateRoute
13 : sendTransportationInfo(route,transportation)
14 : informUser(route,transportation)

Figure 4: Get transportation information Sequence Diagram

Use case name	Get campus events informations
Actors	Users, Speech to Text Service, Server
Description	When user asks for available events in campus, server interacts the database and submits events for user’s information.
Data	Audial input from user
Preconditions	User should be authenticated
Stimulus	User giving audial input about campus event informations
Basic Flow	Step 1 – User gives audial input Step 2 – Audial input gets processed by Speech to Text Service Step 3 – Available events searched in server Step 4 – Available events are converted to audio format. Step 5– Events listed to the user.
Alternative Flow	Step 4 – If no event is available on campus, Garcon will not give any listings.
Exception Flow	If any error occurs, system will show an error message
Post conditions	System shows all events

Table 5: Get event information function

Use case name	Close an issue
Actors	Workers, Speech to Text Service, Mailing Service, Server
Description	Workers close issues they have handled.
Data	Audial input from worker
Preconditions	Worker should be authenticated
Stimulus	Worker giving audial input about closing an issue
Basic Flow	Step 1 – Worker gives audial input Step 2 – Audial input gets processed by Speech to Text Service Step 3 – Issue is closed and server side is informed Step 4 – Worker is informed that issue is closed
Alternative Flow	-
Exception Flow	-
Post conditions	The issue is marked as solved on system.

Table 6: Close issue function

Use case name	Register to an issue
Actors	Workers, Speech to Text Service, Server
Description	Workers register to an issue to prevent possible conflicts.
Data	Audial input from worker, current registered workers on the issue
Preconditions	Worker should be authenticated
Stimulus	Worker giving audial input about registering to an issue
Basic Flow	Step 1 – Worker gives audial input Step 2 – Audial input gets processed by Speech to Text Service Step 3 – Current workers on issue are displayed to worker Step 4 – Worker is asked a confirmation after seeing current workers on issue Step 5 – With workers confirmation he is registered to issue (Database update)
Alternative Flow	-
Exception Flow	If worker does not confirm after seeing current workers on the issue the process is aborted.
Post conditions	The worker is registered to issue.

Table 7: Register issue function

Use case name	Getting informations about issues
Actors	Workers, Speech to Text Service
Description	Worker scans his/her id card and Garcon gets activated. Then waits for worker to talk to decide what to do.
Data	Audial input from worker
Preconditions	Worker should be authenticated
Stimulus	Worker giving audial input about open issues
Basic Flow	Step 1 – Worker gives audial input Step 2 – Audial input gets processed by Speech to Text Service Step 3 – Server returns open issues from Database
Alternative Flow	-
Exception Flow	-
Post conditions	Worker can see the whole available/open issues from database.

Table 8: Get issue information function

Use case name	Authentication
Actors	Users, Workers, Admins, Card Reader
Description	
Data	Chip from Id Cards
Preconditions	-
Stimulus	Id Card must be scanned
Basic Flow	Step 1 – User/Worker/Admin holds Id Card to the device Step 2 – Device reads the Card and authenticate
Alternative Flow	-
Exception Flow	Step 1 – Card doesn’t recognized or cannot be scanned Step 2 – System displays a visual error output
Post conditions	Authenticate successful and device gets waiting for audial input to work.

Table 9: Authentication

Use case name	Adding users
Actors	Admin
Description	Admin adds new user to database
Data	-
Preconditions	Admin should be authenticated
Stimulus	Admin giving information about new user
Basic Flow	Step 1 – Admin gives information about user Step 2 – Input gets processed by Speech to Text Service Step 3 – User Id Card Scanned Step 4 – New user added
Alternative Flow	-
Exception Flow	Step 1 – Admin gives information about user Step 2 – Input cant be recognized Step 3 – System gives a warning
Post conditions	New user added to the database.

Table 10: Add user function

Use case name	Blocking users
Actors	Admin
Description	Admin blocks user from database
Data	-
Preconditions	Admin should be authenticated
Stimulus	Admin gives delete command and gives information
Basic Flow	Step 1 – Admin gives information about user to be deleted Step 2 – Input gets processed by Speech to Text Service Step 3 – User deleted
Alternative Flow	-
Exception Flow	Step 1 – Admin gives information about user Step 2 – Input cant be recognized Step 3 – System gives a warning
Post conditions	User deleted from the database.

Table 11: Block user function

4.2 Composition View

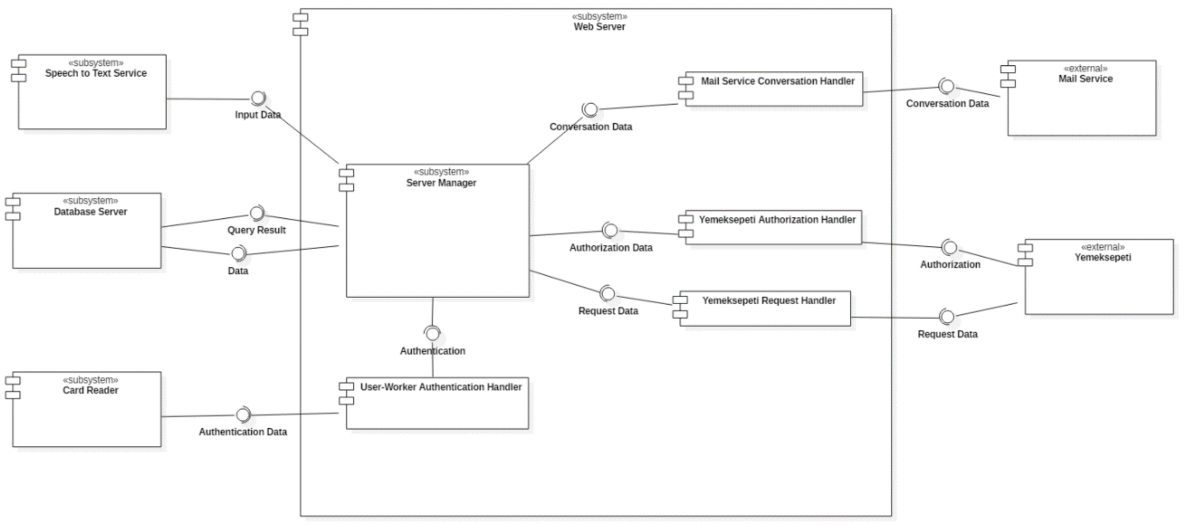


Figure 5: Component Diagram

Design Rationale:

- Server Manager is the fundamental component, which manages/handles information flow and data flow in the Web Server. Moreover, it authorizes/authenticates users and workers.
- Card reader is the component for authentication of users and workers. When Card Reader reads data from ID card, it will direct Authentication Data to User-Worker Authentication handler.
- User-Worker Authentication handler component is responsible from user and worker authentication to the system. It processes raw authentication data coming from card reader.
- Speech to Text Service is the main input provider for system. It converts given audial data to processable data that will be forwarded to Server Manager.
- Database Server is the component that securely stores the related data and provides it when needed.
- Since mailing is the main conversation between workers and system itself, Mail Service is an external component, which is dedicated for managing conversation.
- Mail Service Conversation Handler is the component responsible for providing processed conversations and related data (timestamps, target workers…).
- Yemeksepeti is the external component for ordering food. Two components are responsible from the interaction between Server Manager and Yemeksepeti. Firstly, Yemeksepeti Authorization Handler manages the authorization of user to Yemeksepeti. Secondly, Yemeksepeti Request Handler processes and manages users' orders from Yemeksepeti.

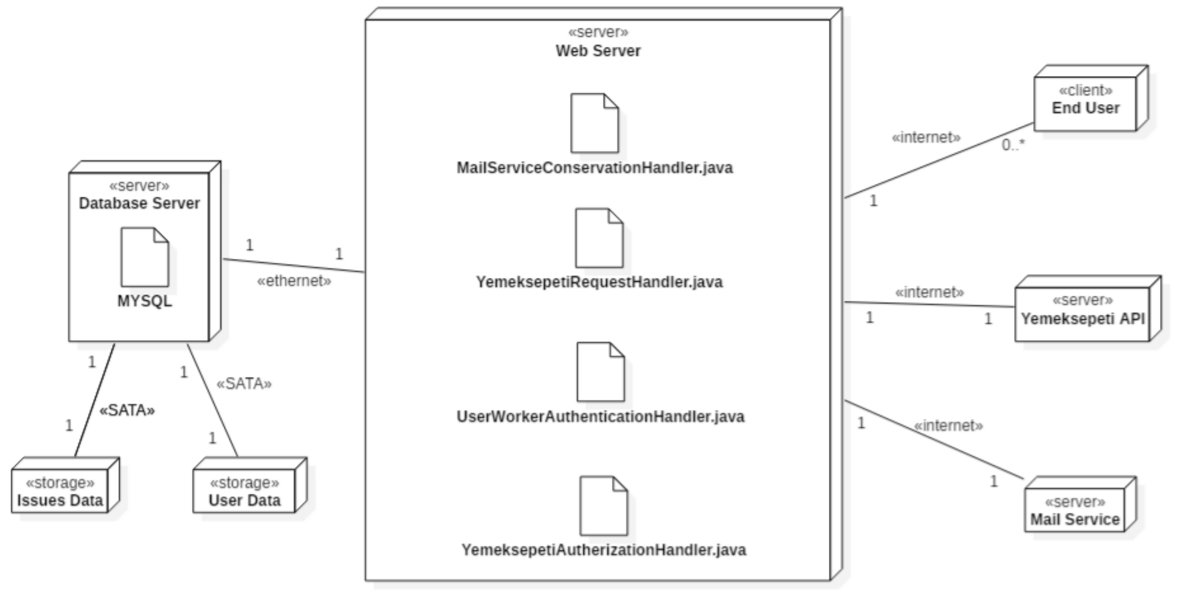


Figure 6: Deployment Diagram

Design Rationale:

- We use Java/Spring in web server side and DB is managed by MySQL.
- The end users who uses Garcon to get information or opening an issue will communicate with system with talking.
- Garcon will take audial input and translate into a text input and sends appropriate handler.
- There are two separate DB storage:
    - Issue storage is for storing all issues.
    - User storage is for storing all information about admins/workers/regular users.
- We SATA for all storage since it has much larger storage space to store whole issue/user.
- All internet-based connections will use encrypted communication protocols.

4.3 Information View

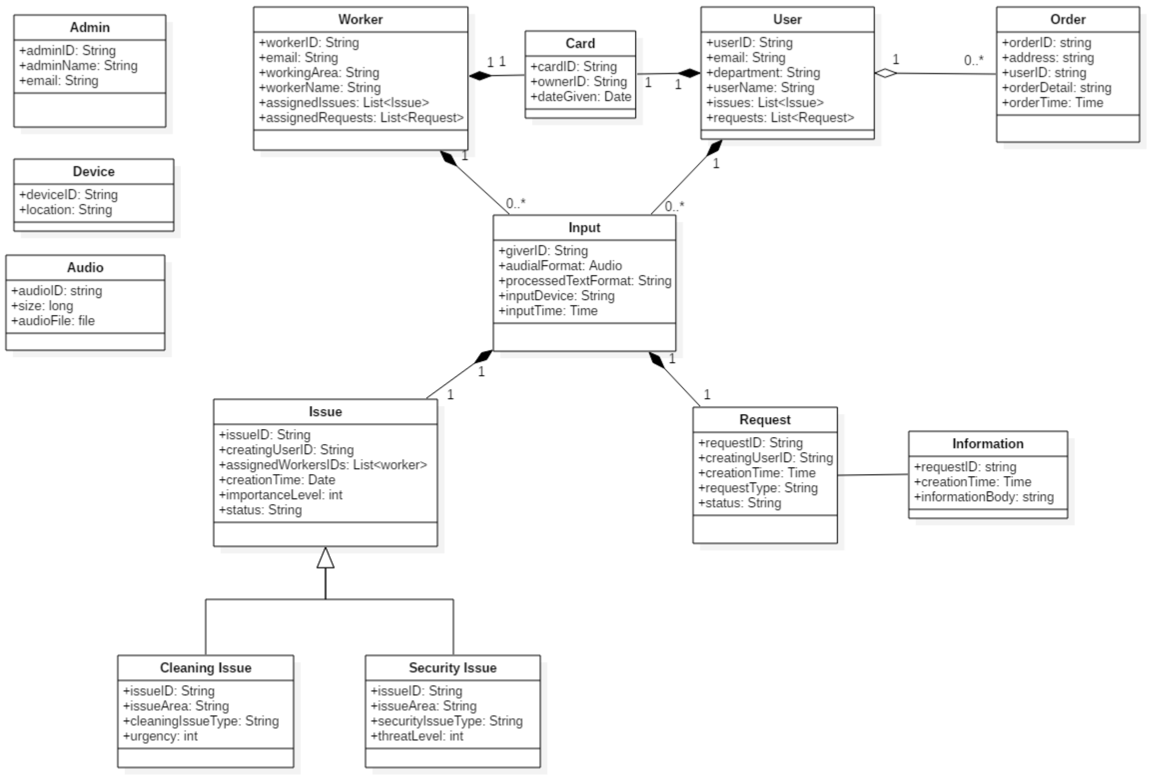


Figure 7: Database Class Diagram

Operation	**CRUD Operations**
addUser	Create – User Read – Update – Delete –
deleteUser	Create – Read – Update – Delete – User
addWorker	Create – Worker

	Read – Update – Delete –
deleteWorker	Create – Read – Update – Delete – Worker
convertAudioToText	Create – Input Read – Audio Update – Delete –
convertTextToAudio	Create – Audio Read – Input Update – Delete –
createIssue	Create – Issue Read – Update – Delete –
registerToIssue	Create – Read – Update – Issue, Worker Delete –
closeIssue	Create – Read – Update – Issue, Worker Delete –
createRequest	Create – Request Read – Update – Delete –
getAssignedIssues	Create – Read – Worker Update – Delete –
getAssignedRequests	Create – Read – Worker Update –

	Delete –
forwardOrder	Create – Read – Order Update – Delete –
forwardOrderReply	Create – Read – Update – Order Delete –
authorizeUser	Create – Read – User Update – Delete –
authenticateUser	Create – Read – User Update – Delete –
authenticateWorker	Create – Read – Worker Update – Delete –

Table 12: CRUD Operations

Design Rationale:

- MySQL is the DBMS used in the project.
- An Issue is either a cleaning issue or a security issue.
- Request and Information classes always coexist.
- Users has Orders but orders can outlive users.
- Users, Workers are stored in DBMS for usage of admins and Issues/Requests are stored in DBMS for usage of workers and users.
- Every entity is stored in database in order to provide persistency of data.

4.4 Interface View

4.4.1 Internal Interfaces

The Interface Between the Database Server and the Server Manager: Server manager queries the Database Server when an operation on a specific information. The query passed as string and dbms runs it as SQL. If query fails, server returns error message, if query is successful, result would be sent back.

Design Rationale:

- Since some students will graduate and some new students will come to the university, keeping user data is important. Thus an interface between DBMS and Server Manager is required.

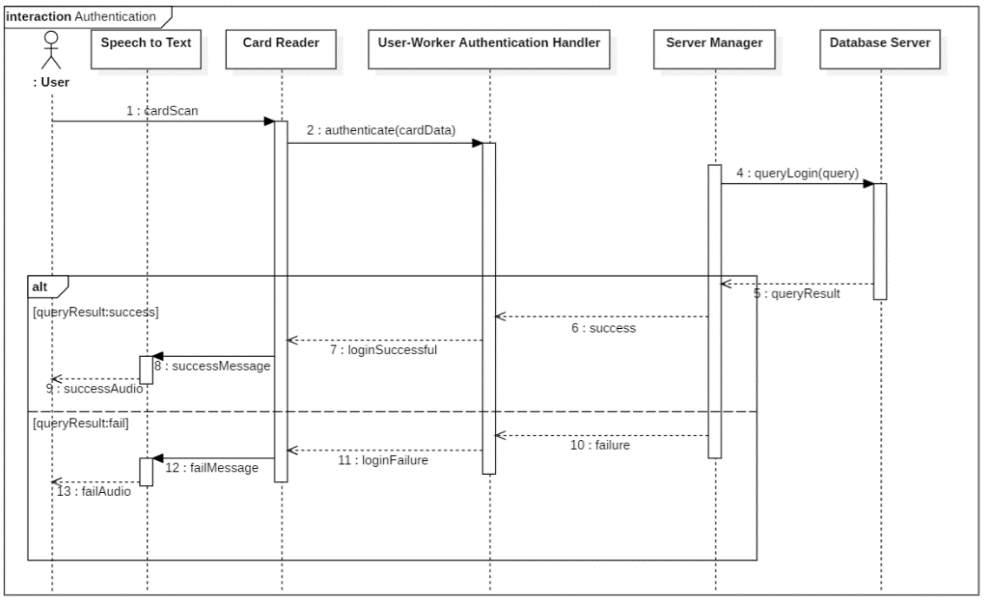


Figure 8: Sign in sequence diagram between Server Manager and Card Reader

The Interface Between Server Manager and Card Reader: When a users show the card reader to their ID cards, all information on card scanned by Card Reader. And then, the information passes to the DBMS to check whether there is a user with given informations. If there is, Garcon will wait for user request but if there is not a user with given information, Garcon will return error message.

Design Rationale:

- System is available only the campus student so there must be an authentication system which involves the Card Reader.

The Interface Between Server Manager and the Speech to Text Service: In order to communicate system with audial input, there must be a speech to text converter. System takes input as audio file and passes to the speech to text service, then this service returns a string of given input.

Design Rationale:

- There is a need for interfacing server manager with speech to text service since system cannot understand the user’s audial input directly and it needs to communicate with help of speech to text service. Thus, there have to be an interface between these components.

4.4.2 External Interfaces

4.4.2.1 User Interfaces

Student Interface: Student interface allows students to interact with Garcon system and use all functionalities of the system. All interactions done via audial input and outputs. Once students show their ID Cards to the system, the system will wait for the audial input from student.

Design Rationale:

- The audio based interface design provides user to access Garcon functionalities easily.

Worker Interface: Worker interface allows workers to query about open issues, and functionalities like closing/updating issues. This interface works with audial input like in the student interface.

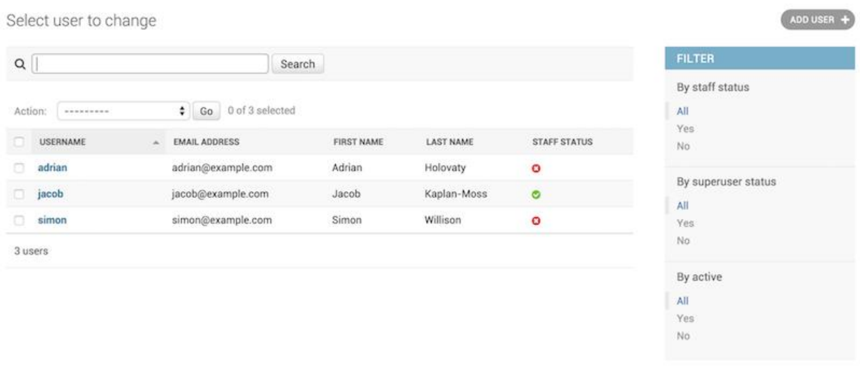
Design Rationale:

- The audio based interface design provides workers to access functionalities about issues easily.

Admin Interface: This interface allows admins to do operations on users and workers. They can add workers and users to the database or they can delete workers and users from database. This interface operates on a web browser , unlike user and worker interface. To login the admin system, admin should enter id and password. And then, they can do operations on users/workers.

Design Rationale:

- Since the design of the interface is simple, probability of misusing the service is reduced.
- Single paged interface design provides functionality and information to be accessible altogether at one place.



USERNAME	EMAIL ADDRESS	FIRST NAME	LAST NAME	STAFF STATUS
adrian	adrian@example.com	Adrian	Holovaty	✗
jacob	jacob@example.com	Jacob	Kaplan-Moss	✓
simon	simon@example.com	Simon	Willison	✗

3 users

Figure 9: Admin interface

4.4.2.2 System Interfaces

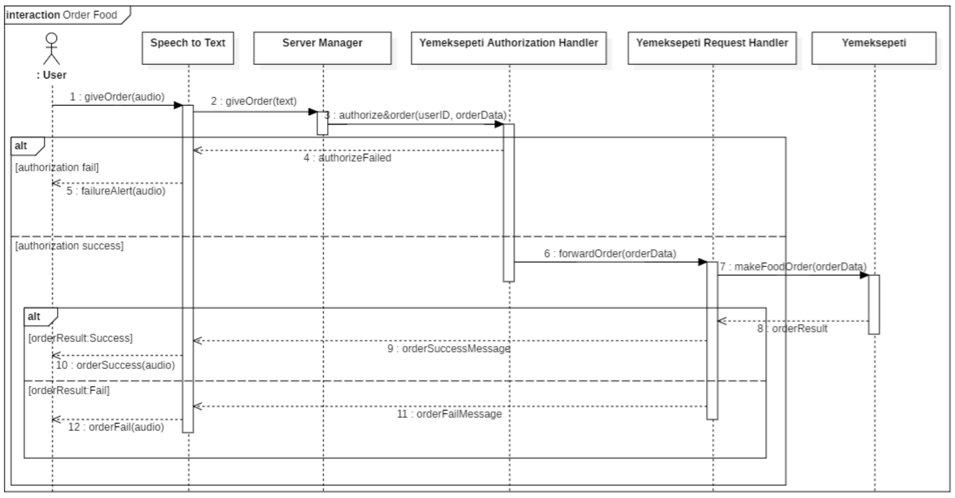


Figure 10: Sequence diagram showing ordering food from Yemeksepeti API

The Interface between the Server Manager and Yemeksepeti API: Server Manager will be responsible for composing food order data and getting report from the Yemeksepeti API. After getting this report, according to report type system takes action. If report is success report, system will return user a success message, if report is failure, then system will return user the reason of this failure.

Design Rationale:

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

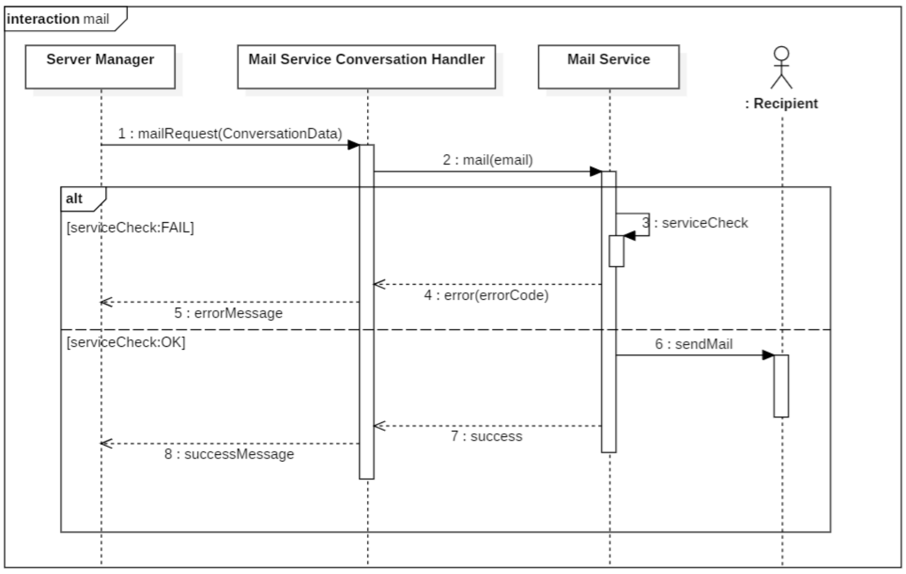


Figure 11: Sequence Diagram showing the interface between mail service and server manager

The Interface between the Server Manager and Mail Service:
Server manager will be responsible for composing mail and sent this mail data to the mail service. When mail service takes this mail data, it will generate a mail, and sends it to correspondant users. After sending the mails system returns success or failure message.

Design Rationale:

- Communication between Server Manager and Mail Service is alive only when a request opened by user is about a issue.

4.4.2.3 Service Interfaces

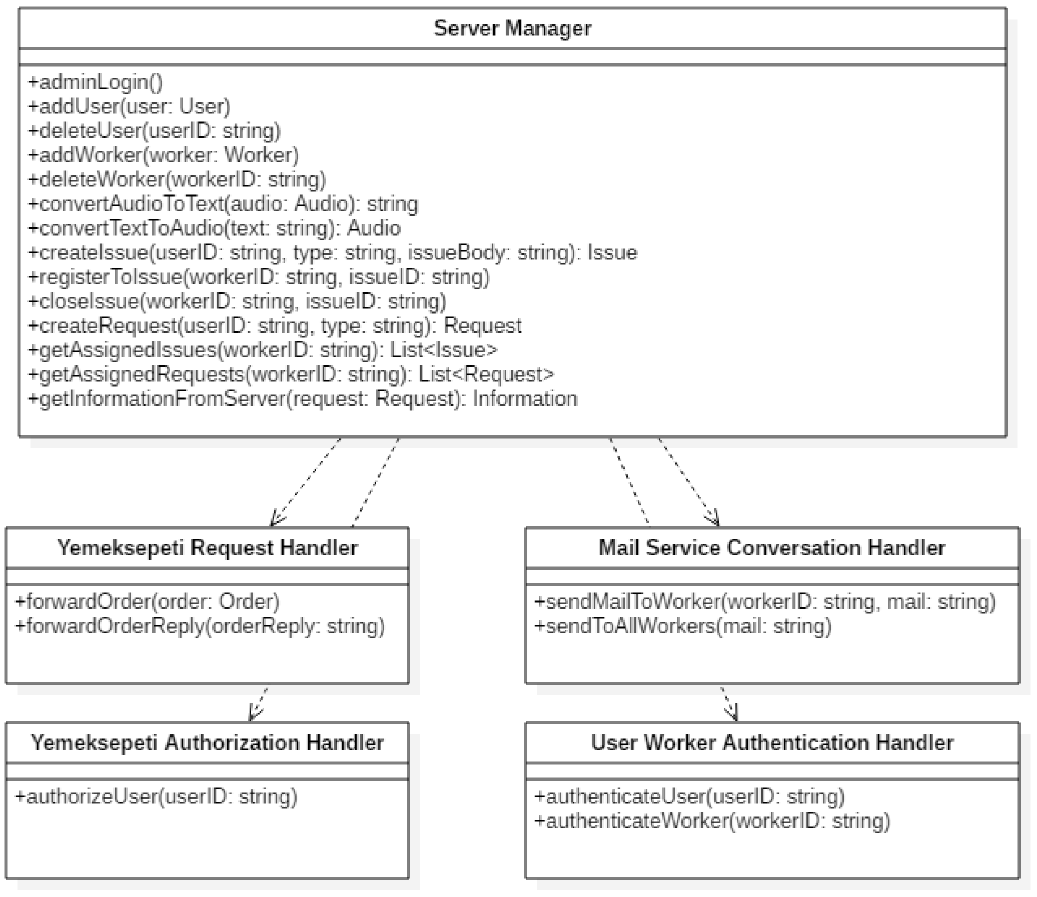

Server Manager
+adminLogin()
+addUser(user: User)
+deleteUser(userID: string)
+addWorker(worker: Worker)
+deleteWorker(workerID: string)
+convertAudioToText(audio: Audio): string
+convertTextToAudio(text: string): Audio
+createIssue(userID: string, type: string, issueBody: string): Issue
+registerToIssue(workerID: string, issueID: string)
+closeIssue(workerID: string, issueID: string)
+createRequest(userID: string, type: string): Request
+getAssignedIssues(workerID: string): List<Issue>
+getAssignedRequests(workerID: string): List<Request>
+getInformationFromServer(request: Request): Information
Yemeksepeti Request Handler
+forwardOrder(order: Order)
+forwardOrderReply(orderReply: string)
Mail Service Conversation Handler
+sendMailToWorker(workerID: string, mail: string)
+sendToAllWorkers(mail: string)
Yemeksepeti Authorization Handler
+authorizeUser(userID: string)
User Worker Authentication Handler
+authenticateUser(userID: string)
+authenticateWorker(workerID: string)

Figure 12: Service Interfaces Class Diagram

Operation	**Description**
adminLogin	Admin logs into the system
addUser	Admin registers new user to the system
deleteUser	Admin deletes existing user from the system
addWorker	Admin registers new worker to the system
deleteWorker	Admin deletes existing worker from the system
convertAudioToText	System interacts with the Speech To Text Service and get audio converted to text

convertTextToAudio	System interacts with the Speech To Text Service and get text converted to audio
createIssue	User creates a new issue which is a security or a cleaning issue
registerToIssue	Worker registers to an issue before handling the issue
closeIssue	After the worker handles the issue he/she closes the issue
createRequest	User creates a new request of information
getAssignedIssues	Shows the current assigned issues of the worker
getAssignedRequests	Shows the current assigned requests of the worker
getInformationFromServer	Gets the information requested by user from server
forwardOrder	Forwards user’s food order to yemeksepeti
forwardOrderReply	Forwards reply of yemeksepeti to user’s food order back to user
authorizeUser	Authorizes current user to yemeksepeti system
sendMailToWorker	Sends the mail to a specified worker
sendMailToAllWorkers	Sends the mail to all workers
authenticateUser	Authenticate user to the system via the card info read by card reader
authenticateWorker	Authenticate worker to the system via the card info read by card reader

Table 13: Operation Descriptions

Operation	**Inputs**	**Outputs**	**Exceptions**
adminLogin		If successful returns True else False	-Database Server Error
addUser	user	If successful returns True else False	-User already added -Database Server Error
deleteUser	userID	If successful returns True else False	-User doesn’t exist -Database Server Error
addWorker	worker	If successful returns True else False	-Worker already added -Database Server Error
deleteWorker	workerID	If successful returns True else False	-Worker doesn’t exist -Database Server Error
convertAudioToText	audio	Converted text	-Connection error
convertTextToAudio	text	Converted audio	-Connection error
createIssue	userID type issueBody	If successful returns True else False	-Database Server Error
registerToIssue	workerID issueID	If successful returns True else False	-Worker already registered to issue -Database Server Error
closeIssue	workerID issueID	If successful returns True else False	-Issue doesn’t exist -Database Server Error

createRequest	userID type issueBody	Returns created request	-Database Server Error
getAssignedIssues	workerID	List of issues	-Database Server Error
getAssignedRequests	workerID	List of requests	-Database Server Error
getInformationFromServer	request	Returns information gathered from server	-Database Server Error
forwardOrder	order	If successful returns True else False	-Connection error
forwardOrderReply	orderReply	If successful returns True else False	-Connection error
authorizeUser	userID	If successful returns True else False	-Connection error
sendMailToWorker	workerID mail	If successful returns True else False	-Connection error
sendMailToAllWorkers	mail	If successful returns True else False	-Connection error
authenticateUser	userID	If successful returns True else False	-ID doesn’t match -Database Server Error
authenticateWorker	workerID	If successful returns True else False	-ID doesn’t match -Database Server Error

Table 14: Operation Design

Design Rationale:

- Server manager is the main structure that responsible for database operations.
- Handlers are responsible for intercommunication between external/subsystems and server manager; therefore, they have operations providing functionality of that.
- All the provided methods are asynchronously working methods that are awoken by components.
- authenticateUser() and authenticateWorker() methods are using/validating the information that is read by card reader component.
- forwardOrder() method is always followed by forwardOrderReply() method.