The German International University
Of Applied Sciences

Department of Computer Science &
Software Engineering

Project Phase #1
Package Delivery Robot

An SRS submitted by:

Names:

-- Amr Ahmed Shawki

-- Mohamed Bolbol

-- Omaima Ahmed

-- Omar Abdullah

-- Youssef Henna

Supervised By:

-- Dr Amr Desouky

Nov 2021

Table of contents:

1. Introduction

1.1 Purpose

The purpose of this system is to assist students and staff in sending packages across the GIU premises more conveniently.

1.2 Scope

The navigation system is scoped and limited to the following functionalities. Broadcasting the robot's location, determining the fastest route, helping it avoid obstacles. The software will also send a notification when the robot reaches the drop off/pick up point and the user will need to input the correct pin code so that the user can retrieve/send their order

1.3 Definitions, acronyms, and abbreviations

Definitions used are:

Users: When referring to a user in this document, we're referring to GIU students & staff. Admins are addressed as a separate entity.

Backwards traceability: which is the ability to trace an improved requirement from its source (preliminary definition) which in essence is about understanding why a requirement had to be improved and thought process in which the requirement reached its current state. This in this current document can be correlated to the "Issues with preliminary definitions".

Forwards Traceability: Which in essence is about investigating the impact of improving the preliminary requirement on the components that are based on it, and analyzing what changes need to be made on those components to better reflect the new requirement. This can also lead to modification in test cases of those components as well.

Acronyms & Abbreviations used are:

'*' → symbol means combined requirement

'(x,y,z)' → symbol for showing combined points and their lines (instead of x y z placeholders). Can also be used with FR# and NFR# for points with similar improvement or issues.

SRS → to software requirements specification

GIU → German International University of Applied Sciences

M → Building M

S → Building S

App → Client side user application

(I)FR → Functional Requirement. The (I) stands for “improved”.

(I)FR#X → no. of Functional Requirement where X denotes the number. The (I) stands for “improved”.

(I)NFR → Non Functional Requirement. The (I) stands for “improved”.

(I)NFR#X → no. of Non Functional Requirement where X denotes the number. The (I) stands for “improved”.

2. Preliminary Definition

2.1 Problem

The problem at hand is the lack of a convenient way to send and receive packages across the increasingly large campus. This package can come in any form, such as: assignments, books, food, electronics etc. It is quite troublesome to manually deliver these packages on foot and can be a waste of time.

2.2 Goal

The goal of this document is to fully describe a system that manages automatic deliveries across the campus with the help of an autonomous robot. This functionality of the system will be elaborated extensively throughout this document.

2.3 Domain

The domain is the GIU, in particular outdoor areas on campus.

2.4 Stakeholders

The stakeholders will be any person associated with the GIU that wants to send and receive an item. In particular, we'll have 4 types of users. Students, Staff, Admins and Developers.

2.5 Functional Requirements

Listed below is the list of FRs that will define the scope of our robot and smartphone application functionalities. It will be abbreviated as FR#X:

FR#1 User shall provide pickup location
FR#2 User shall provide dropoff location
FR#3 User shall see robot location
FR#4 Robot shall emit location
FR#5 Robot shall move a certain distance
FR#6 Robot shall stop at the appropriate location
FR#7 * Robot shall turn when necessary
FR#8 * Robot shall detect obstacles
FR#9 * Robot shall avoid obstacles
FR#10 Robot shall send notification when destination is reached
FR#11 Robot shall unlock storage bin when correct user pin is provided
FR#12 Robot shall send notification when something is wrong
FR#13 User shall sign up on app with university email, ID, and password
FR#14 User shall be able to login to app
FR#15 User shall specify type of package being sent from a pre specified list of items
FR#16 User shall set a pin at sign up that will be used for unlocking robot's compartment for sending / receiving packages

FR#17 * User shall be notified of waiting time till robot's arrival at pick up / drop off point
FR#18 Robot shall emit a notification when it is busy on delivery and unable to receive packages
FR#19 User should be able to submit a complaint to the apps help center in case of unsuccessful deliveries
FR#20 Robot should have different compartments with different doors for simultaneous deliveries
FR#21 User shall have a contact list of sending / receiving users
FR#22 System shall (RE)calculate the ETA using parameters sent by the robot
FR#23 User shall be able to search for users by name or university ID and add / remove them to / from their contact list
FR#24 User shall be able to choose a user to send / receive a package from their contact list
FR#25 User shall be able to accept / reject incoming / outgoing delivery requests
FR#26 User can order food from pre-defined restaurants in the contact list
FR#27 User shall be able to type a message when specifying a delivery
FR#28 Robot should be able to tell which compartment is associated with which sending / receiving user when pin is entered
FR#29 Robot should be able to tell if an item is taken out of a

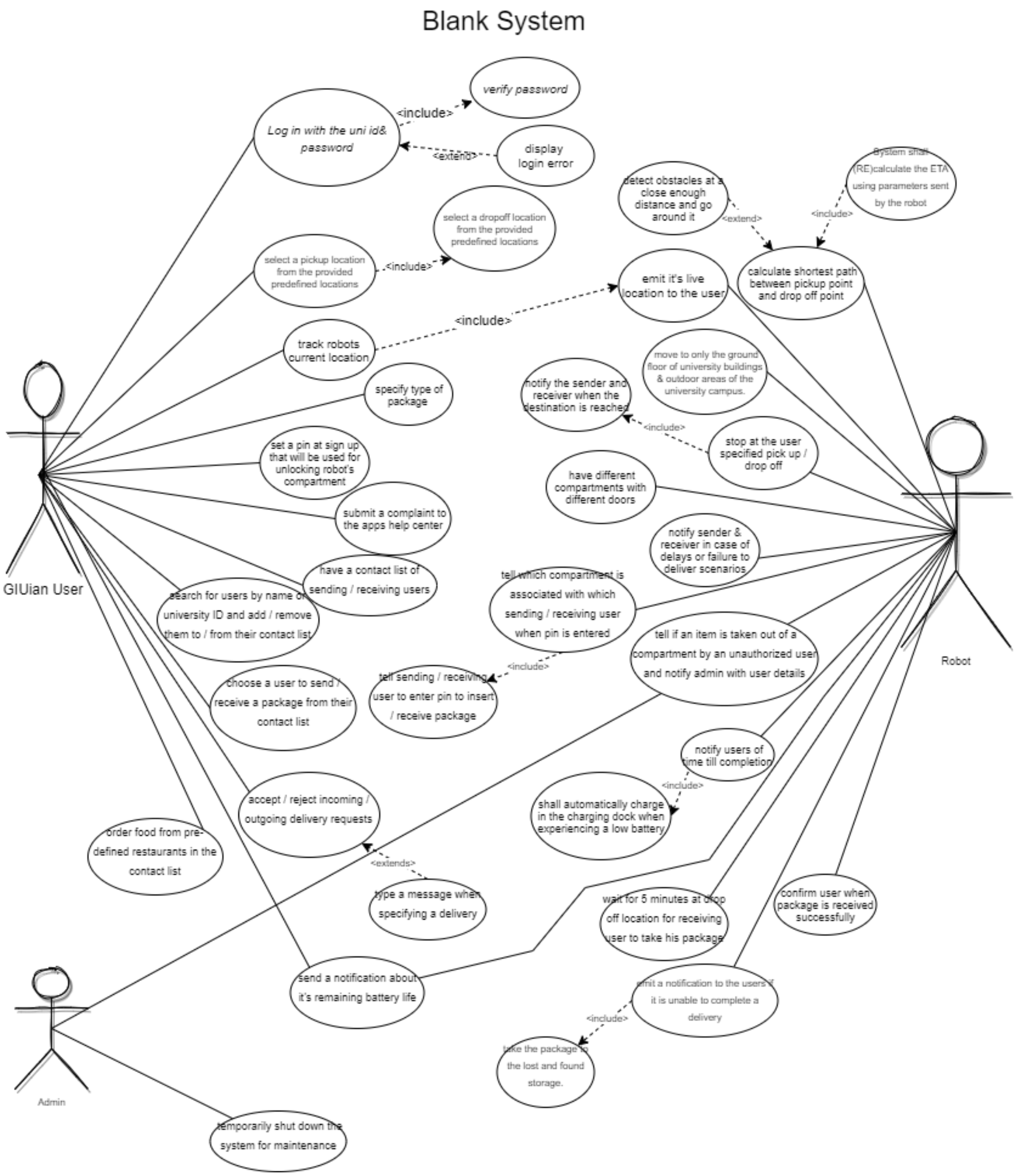
compartment by an unauthorized user and notify admin with user details
FR#30 Robot shall tell sending / receiving user to enter pin to insert / receive package
FR#31 Robot shall calculate shortest path between pickup point and drop off point & recalculate after obstacle resolution
FR#32 Robot shall wait for 5 minutes at drop off location for receiving user to take his package
FR#33 Robot shall confirm user when package is received successfully in case of sending user is different from receiving user
FR#34 Robot shall emit a notification to the users if it is unable to complete a delivery, and take the package to the lost and found storage.
FR#35 Robot shall send required data for calculating ETA
FR#36 Admins should be able to temporarily shut down the system for maintenance and notify users beforehand.
FR#37 * Robot shall send a notification about the current package in the waiting queue
FR#38 Robot shall send a notification about it's remaining battery life
FR#39 Robot shall automatically charge in the charging dock when experiencing a low battery and notify users of time till completion.
FR#40 Robot shall have a physical key slot for opening the compartments in case of a bug occuring or for maintenance

2.6 Non Functional Requirements

Below we'll list the NFRs that will define the scope of our robot and smartphone application functionalities. It will for the abbreviation NFR#X:

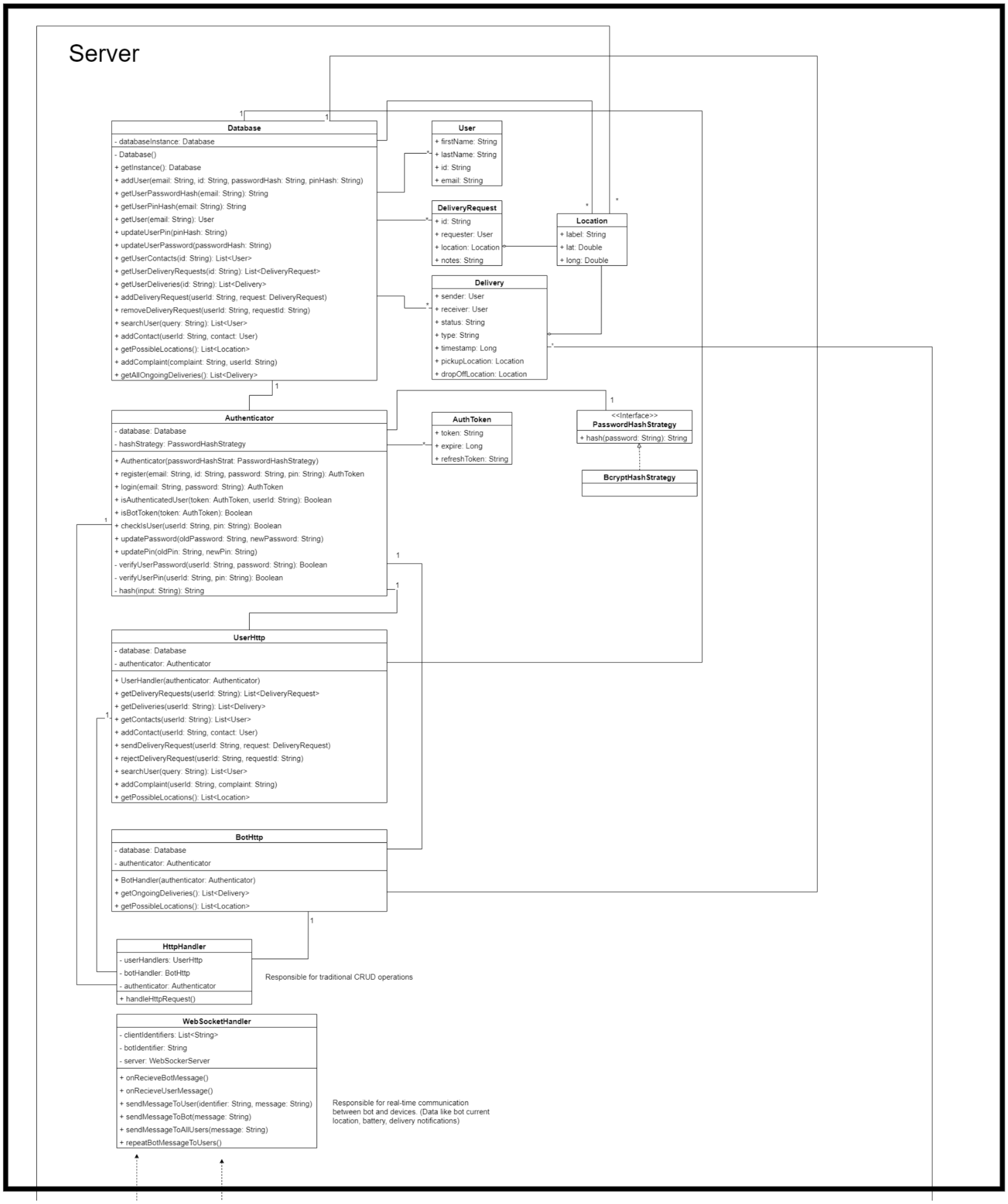
NFR#1 System shall assist robot to safely navigate on campus
NFR#2 System shall determine the fastest route
NFR#3 System shall navigate robot through selected route
NFR#4 Robot should be able to handle packages up to 10kg
NFR#5 Robot should be able to survive severe weather conditions
NFR#6 System should have security measures to prevent unintended parties from receiving the package
NFR#7 Robot should make deliveries quickly
NFR#8 System shall be easily accessible to all members of the university
NFR#9 System should have over 90% successful deliveries
NFR#10 System should be able to handle several simultaneous deliveries
NFR#11 System will only be accessible to users currently on campus

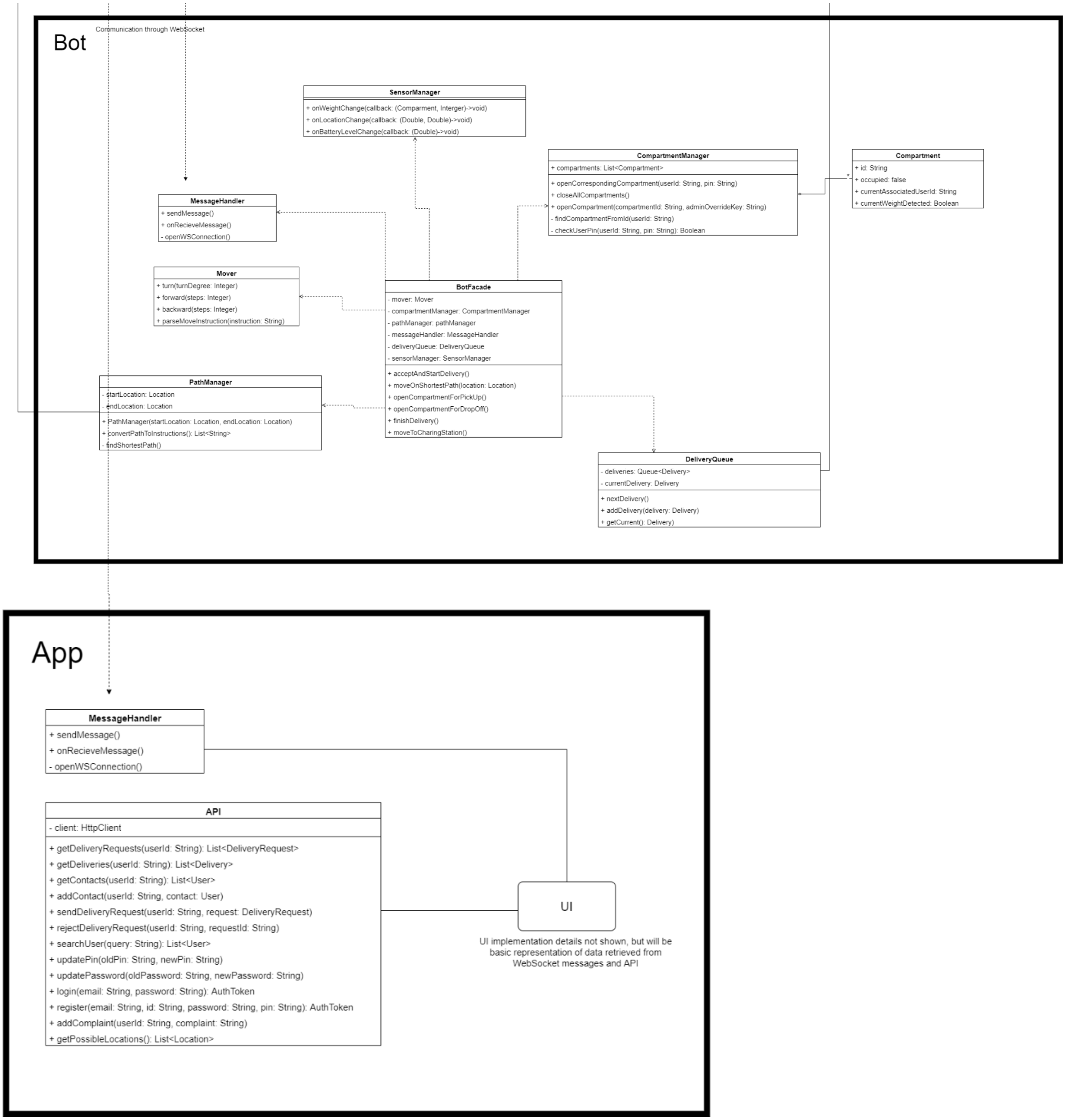
2.7 Use Case Diagram (Full resolution: https://ibb.co/304pLpq)


Blank System
GIUian User
Robot
Admin
Log in with the uni id& password
<include>
verify password
display login error
<extend>
select a pickup location from the provided predefined locations
<include>
select a dropoff location from the provided predefined locations
track robots current location
<include>
emit it's live location to the user
detect obstacles at a close enough distance and go around it
<extend>
System shall (RE)calculate the ETA using parameters sent by the robot
<include>
calculate shortest path between pickup point and drop off point
move to only the ground floor of university buildings & outdoor areas of the university campus.
specify type of package
notify the sender and receiver when the destination is reached
<include>
stop at the user specified pick up / drop off
set a pin at sign up that will be used for unlocking robot's compartment
have different compartments with different doors
submit a complaint to the apps help center
notify sender & receiver in case of delays or failure to deliver scenarios
have a contact list of sending / receiving users
search for users by name or university ID and add / remove them to / from their contact list
tell which compartment is associated with which sending / receiving user when pin is entered
<include>
tell sending / receiving user to enter pin to insert / receive package
tell if an item is taken out of a compartment by an unauthorized user and notify admin with user details
choose a user to send / receive a package from their contact list
notify users of time till completion
<include>
shall automatically charge in the charging dock when experiencing a low battery
accept / reject incoming / outgoing delivery requests
order food from pre-defined restaurants in the contact list
<extends>
type a message when specifying a delivery
wait for 5 minutes at drop off location for receiving user to take his package
confirm user when package is received successfully
send a notification about it's remaining battery life
emit a notification to the users if it is unable to complete a delivery
<include>
take the package to the lost and found storage.
temporarily shut down the system for maintenance

2.8 Class Diagram

Some implementation details have been disregarded. The following class diagram captures how the whole system will operate, with all 3 components. ibb.co/xJKgGvG for a full resolution diagram


Server
Database
- databaseInstance: Database
- Database()
+ getInstance(): Database
+ addUser(email: String, id: String, passwordHash: String, pinHash: String)
+ getUserPasswordHash(email: String): String
+ getUserPinHash(email: String): String
+ getUser(email: String): User
+ updateUserPin(pinHash: String)
+ updateUserPassword(passwordHash: String)
+ getUserContacts(id: String): List<User>
+ getUserDeliveryRequests(id: String): List<DeliveryRequest>
+ getUserDeliveries(id: String): List<Delivery>
+ addDeliveryRequest(userId: String, request: DeliveryRequest)
+ removeDeliveryRequest(userId: String, requestId: String)
+ searchUser(query: String): List<User>
+ addContact(userId: String, contact: User)
+ getPossibleLocations(): List<Location>
+ addComplaint(complaint: String, userId: String)
+ getAllOngoingDeliveries(): List<Delivery>
User
+ firstName: String
+ lastName: String
+ id: String
+ email: String
DeliveryRequest
+ id: String
+ requester: User
+ location: Location
+ notes: String
Location
+ label: String
+ lat: Double
+ long: Double
Delivery
+ sender: User
+ receiver: User
+ status: String
+ type: String
+ timestamp: Long
+ pickupLocation: Location
+ dropOffLocation: Location
Authenticator
- database: Database
- hashStrategy: PasswordHashStrategy
+ Authenticator(passwordHashStrat: PasswordHashStrategy)
+ register(email: String, id: String, password: String, pin: String): AuthToken
+ login(email: String, password: String): AuthToken
+ isAuthenticatedUser(token: AuthToken, userId: String): Boolean
+ isBotToken(token: AuthToken): Boolean
+ checkIsUser(userId: String, pin: String): Boolean
+ updatePassword(oldPassword: String, newPassword: String)
+ updatePin(oldPin: String, newPin: String)
- verifyUserPassword(userId: String, password: String): Boolean
- verifyUserPin(userId: String, pin: String): Boolean
- hash(input: String): String
AuthToken
+ token: String
+ expire: Long
+ refreshToken: String
<<Interface>>
PasswordHashStrategy
+ hash(password: String): String
BcryptHashStrategy
UserHttp
- database: Database
- authenticator: Authenticator
+ UserHandler(authenticator: Authenticator)
+ getDeliveryRequests(userId: String): List<DeliveryRequest>
+ getDeliveries(userId: String): List<Delivery>
+ getContacts(userId: String): List<User>
+ addContact(userId: String, contact: User)
+ sendDeliveryRequest(userId: String, request: DeliveryRequest)
+ rejectDeliveryRequest(userId: String, requestId: String)
+ searchUser(query: String): List<User>
+ addComplaint(userId: String, complaint: String)
+ getPossibleLocations(): List<Location>
BotHttp
- database: Database
- authenticator: Authenticator
+ BotHandler(authenticator: Authenticator)
+ getOngoingDeliveries(): List<Delivery>
+ getPossibleLocations(): List<Location>
HttpHandler
- userHandlers: UserHttp
- botHandler: BotHttp
- authenticator: Authenticator
+ handleHttpRequest()
Responsible for traditional CRUD operations
WebSocketHandler
- clientIdentifiers: List<String>
- botIdentifier: String
- server: WebSockerServer
+ onRecieveBotMessage()
+ onRecieveUserMessage()
+ sendMessageToUser(identifier: String, message: String)
+ sendMessageToBot(message: String)
+ sendMessageToAllUsers(message: String)
+ repeatBotMessageToUsers()
Responsible for real-time communication between bot and devices. (Data like bot current location, battery, delivery notifications)
1
1
*
*
*
*
*
1
1
*
1
1
1
1
1
*



SensorManager

+ onWeightChange(callback: (Comparment, Interger)->void)
+ onLocationChange(callback: (Double, Double)->void)
+ onBatteryLevelChange(callback: (Double)->void)

CompartmentManager

+ compartments: List<Compartment>

+ openCorrespondingCompartment(userId: String, pin: String)
+ closeAllCompartments()
+ openCompartment(compartmentId: String, adminOverrideKey: String)
- findCompartmentFromId(userId: String)
- checkUserPin(userId: String, pin: String): Boolean

Compartment

+ id: String
+ occupied: false
+ currentAssociatedUserId: String
+ currentWeightDetected: Boolean

MessageHandler

+ sendMessage()
+ onRecieveMessage()
- openWSConnection()

Mover

+ turn(turnDegree: Integer)
+ forward(steps: Integer)
+ backward(steps: Integer)
+ parseMoveInstruction(instruction: String)

BotFacade

- mover: Mover
- compartmentManager: CompartmentManager
- pathManager: pathManager
- messageHandler: MessageHandler
- deliveryQueue: DeliveryQueue
- sensorManager: SensorManager

+ acceptAndStartDelivery()
+ moveOnShortestPath(location: Location)
+ openCompartmentForPickUp()
+ openCompartmentForDropOff()
+ finishDelivery()
+ moveToCharingStation()

PathManager

- startLocation: Location
- endLocation: Location

+ PathManager(startLocation: Location, endLocation: Location)
+ convertPathToInstructions(): List<String>
- findShortestPath()

DeliveryQueue

- deliveries: Queue<Delivery>
- currentDelivery: Delivery

+ nextDelivery()
+ addDelivery(delivery: Delivery)
+ getCurrent(): Delivery

MessageHandler

+ sendMessage()
+ onRecieveMessage()
- openWSConnection()

API

- client: HttpClient

+ getDeliveryRequests(userId: String): List<DeliveryRequest>
+ getDeliveries(userId: String): List<Delivery>
+ getContacts(userId: String): List<User>
+ addContact(userId: String, contact: User)
+ sendDeliveryRequest(userId: String, request: DeliveryRequest)
+ rejectDeliveryRequest(userId: String, requestId: String)
+ searchUser(query: String): List<User>
+ updatePin(oldPin: String, newPin: String)
+ updatePassword(oldPassword: String, newPassword: String)
+ login(email: String, password: String): AuthToken
+ register(email: String, id: String, password: String, pin: String): AuthToken
+ addComplaint(userId: String, complaint: String)
+ getPossibleLocations(): List<Location>

UI implementation details not shown, but will be basic representation of data retrieved from WebSocket messages and API

3. Issues with Preliminary Definition

3.1 Domain

Original preliminary definition domain specified domain for outdoor areas of the GIU campus only. We added support for ground floors of the M and S buildings, thus expanding the domain to be: GIU outdoor areas and building ground floors.

3.2 Functional Requirement

FR#(1,2,3,5,6,12) → FR is too vague and ambiguous. Needed additional details to solidify its definition.

FR#4 → Needed more clarification on when / how often location will be emitted

FR#(7,8,9,17,37) → Too similar and could be combined into fewer and clearer requirements

FR#10 → Doesn't specify who the notification will be sent to

3.3 Non Functional Requirement

NFR#(1,3,6,7) → Needed more clarification

4. Improved Understanding

4.1 World

World assumptions made about the domain are:

Campus is well suited for robot to navigate with no unavoidable obstacles in the possible routes.

There is an existing University system that holds all student and staff data and can communicate with the new delivery system.

There is a dedicated entrance for the delivery robot in each of the buildings (M, S) so that the robot can make deliveries on the ground floor.

The delivery robot as well as users are both always connected to high speed WIFI through routers that are evenly distributed across the GIU campus.

When the delivery robot is about to run out of battery life, it automatically goes to recharge at the nearest charging dock.

There will be a "lost and found" room present in the campus with one security guard present at all times to drop off failed to receive deliveries.

We will assume that the shortest delivery path is also the fastest one.

The physical compartment opening key as well as other physical maintenance tools will be present with the Administrator of the system.

We will assume that restaurants have access to the university payment system. This will allow food orders made with the delivery bot to be paid on the student’s university semester fees or deducted from the salary of staff members. This can be mitigated if the order is paid before a certain date.

4.2 Requirements Specification

FR#1 → User will use the application related to the service, after accessing the app the user will select a specific location inside the university. It can not be outside the range of university because it will not be accessible for the robot. Selecting the location is done through selecting from predefined locations.

FR#2 → User will select the drop off location after selecting the pick up location, user will select a specific location inside the university. It can not be outside the range of university because it will not be accessible for the robot. Selecting the location is done through selecting from predefined locations.

FR#3 → User will be able to track the robot live while the delivery is in progress. This will allow the user to know where the robot is and to prevent location conflicts. This also prevents doubts from users about the robot’s exact location.

FR#4 → The robot will be able to emit its location and this will be also recorded and shown on the tracking screen with the whole progress that’s happening.

FR#5 → Robots will only be inside the university campus and there will be no access outside the campus at all. It will be forbidden because robots will have specific jobs to do and they are just inside the campus. While inside the campus the movement will be on buildings’ ground floors and in the outside areas of the campus.

FR#6 → Robots will stop at the specified locations the user selected from predefined locations for pick up and drop off. Robots shouldn’t have any issues related to the pickup and drop off location.

FR#7 → Robots will be using high quality cameras and high quality sensors, so they will be able to detect any obstacles they will face in the route and they must find solutions from the many solutions they are programmed to overcome. For example, like turning around this obstacle or finding another easy clear way to go through it.

FR#8 → There will be a notification system that will notify each of the sender or the receiver when the robot reaches its specific destination, this notification will be sent when the process is finished.

FR#9 → Robot will notify the sender or the receiver of any delays that could happen or any issues that are faced related to the route or the process in general, so that they are aware of any issues that could happen.

FR#10→ Robot will notify users about the pickup and drop off time and the estimated arrival time. 2 minutes before reaching the destination, another notification is sent. User will also be told of the current position in the queue.

FR#11 → The robot will unlock its storage bin when the user puts his or her own pin correctly by typing the pin on a keypad constructed on the robot’s body, the robot will check the database after receiving the pin. If that pin is correct it will respond to the user by unlocking the bin. If it is not correct it will tell the user to re-enter the pin and this is of course for security and privacy. Each pin is saved with each account that has been made for this service so the pin code is like the id the robot can check to ensure it belongs to the user.

FR#12 → The user will have to sign up in the app to use the robot services by entering his or her information like the university email, ID, Password and the pin code which is the most important thing related to the user.

FR#13 → The user has to specify the package’s type so the robot considers it accepted or not according to the package's rules schema which has all the specifications related to the packages , the user can choose the types from already recorded types in a list.

FR#14 → The user will set a pin code at the sign up process that will be used later for unlocking the robot’s bin for sending or receiving packages the user should be aware of that pin and not giving it to anyone not trusted and also to take care of making it easy pin for not forgetting it easily.

FR#15 → Robot will send a notification when it is busy with deliveries and unable to receive packages. When users are sending them requests, users are notified that they should wait.

FR#16 → User will be able to submit complaints to the apps help center in case of unsuccessful deliveries so that the admins can work on that immediately for more efficiency in work general improvement.

FR#17 → Robots will have their own different compartments with different doors for simultaneous deliveries and for saving time.

FR#18 → Each user will have a saved contact list of senders and receivers around the university campus for later uses.

FR#19 → The system has to (RE)calculate the estimated time of arrival using parameters sent by the robot so the system can notify the user about the estimated time of arrival.

FR#20 → Each user can search for other users by their names or university ID for adding them to their contact list or for removing them for later use.

FR#21 → User will be able to choose the other user from the contact list for sending or receiving packages.

FR#22 → User will be able to accept or reject the coming requests from other users from the notifications they received.

FR#23 → User will be able to order food from predefined restaurants in the contact lists; those restaurants will only be university restaurants and not outside restaurants.

FR#24 → User can type a message specifying the delivery request for the other user so the communication will be easier between both sides.

FR#25 → Robot will be able to specify which compartment is associated with which pin is entered so everything is safe and organized.

FR#26 → Robot should know if some item has been taken out of a compartment by an unauthorized user and notify the admin about the user information in detail by sending the notification to ensure safety.

FR#27 → Robot will tell the sending or receiving user to enter their pins so they can insert or receive the package from the robot’s bin.

FR#28 → Robot will be able to calculate the shortest path between pickup point and drop off point using the Dijkstra algorithm for finding the shortest path nodes which are the pick up and drop off points and the robot will be able to recalculate after any obstacle resolution.

FR#29 → The robot will wait for the user for 5 minutes for the user to get to the robot and receive the package, if the user did not appear for any reason so the package will be returned to the lost and found. Plus the sender will be notified about that.

FR#30 → Robot will notify the sender with a notification that says the package has been received successfully by the receiver and the process has been confirmed. This will be in case if the sending user is different from the receiving user.

FR#31 → Robot will notify the user if the delivery can not be completed for any reason and the package will be taken to the lost and found storage and will also notify in which section in the storage so the user can go and get it.

FR#32 → Robots send data about its location, speed, and sensor data so that the ETA can be calculated accurately.

FR#33 → Admins will have access and the ability to shutdown the whole system temporarily for any reasons or for maintenance, admins will notify all the users for this process before it happens so all will be ready, also robots will go back to their stations and all orders will be done before this process happens.

FR#34 → Robots will send notification about their battery life so if a robot's battery is almost depleted the robot will delay or cancel all the orders and will go to the station for battery recharge.

FR#35 → Robots will automatically charge themselves when they experience a low battery state and will notify all the users about that including the completion time for fully recharging, if there are some requests it will be delayed or canceled.

FR#36 → Robots will have a physical key slot for accessing the bin using a physical key. If users face any issues of opening the robot's bin such as bugs or other problems, the key will be with the admin so a notification will be sent to the admin for fixing this issue.

FR#37 → User will be able to change forgotten passwords or pin code for any reasons while changing the password or the pin code the app will require to write the current password or pin and the new one also will send a verification code to the user's email to write it back in the app to change the password or the pin code for more safety.

FR#38 → The user will be able to login to the app or to logout from the app, information will be saved for later login process.

4.3 Functional Requirements

Below is the final list of the IFRs. Note that many of the original FRs didn't need improvement as they were already clear enough.

IFR#1 User should be able to select a pickup location from the provided predefined locations
IFR#2 User should be able to select a dropoff location from the provided predefined locations
IFR#3 User should be able to track robots current location on the app
IFR#4 Robot shall emit it's live location to the user
IFR#5 Robot shall be able to move to only the ground floor of university buildings & outdoor areas of the university campus.
IFR#6 Robot shall stop at the user specified pick up / drop off or predefined location
IFR#7 Robot should be able to detect obstacles at a close enough distance and go around it (turn) in order to avoid collision
IFR#8 Robot shall notify the sender and receiver when the destination is reached
IFR#9 Robot shall notify sender & receiver in case of delays or failure to deliver scenarios
IFR#10 Robot shall notify Users about pick up / drop off general

arrival time, 2 min to arrival time & package waiting time in delivery queue
IFR#11 Robot shall unlock storage bin when correct user pin is provided
IFR#12 User shall sign up on app with university email, ID, and password
IFR#13 User shall specify type of package being sent from a pre specified list of items
IFR#14 User shall set a pin at sign up that will be used for unlocking robot's compartment for sending / receiving packages
IFR#15 Robot shall emit a notification when it is busy on delivery and unable to receive packages
IFR#16 User should be able to submit a complaint to the apps help center in case of unsuccessful deliveries
FR#17 Robot should have different compartments with different doors for simultaneous deliveries
IFR#18 User shall have a contact list of sending / receiving users
IFR#19 System shall (RE)calculate the ETA using parameters sent by the robot
IFR#20 User shall be able to search for users by name or university ID and add / remove them to / from their contact list
IFR#21 User shall be able to choose a user to send / receive a package from their contact list

IFR#22 User shall be able to accept / reject incoming / outgoing delivery requests
IFR#23 User can order food from pre-defined restaurants in the contact list
IFR#24 User shall be able to type a message when specifying a delivery
IFR#25 Robot should be able to tell which compartment is associated with which sending / receiving user when pin is entered
IFR#26 Robot should be able to tell if an item is taken out of a compartment by an unauthorized user and notify admin with user details
IFR#27 Robot shall tell sending / receiving user to enter pin to insert / receive package
IFR#28 Robot shall calculate shortest path between pickup point and drop off point & recalculate after obstacle resolution
IFR#29 Robot shall wait for 5 minutes at drop off location for receiving user to take his package
IFR#30 Robot shall confirm user when package is received successfully in case of sending user is different from receiving user
IFR#31 Robot shall emit a notification to the users if it is unable to complete a delivery, and take the package to the lost and found storage.
IFR#32 Robot shall send required data for calculating ETA

IFR#33 Admins should be able to temporarily shut down the system for maintenance
IFR#34 Robot shall send a notification about it's remaining battery life
IFR#35 Robot shall automatically charge in the charging dock when experiencing a low battery and notify users of time till completion.
IFR#36 Robot shall have a physical key slot for opening the compartments in case of a bug occuring or for maintenance
IFR#37 User shall be able to change forgotten password or private pin code
IFR#38 User shall be able to login to the app or to logout
IFR#29 Admins should be able to temporarily shut down the system for maintenance and notify users beforehand.

4.4 Non Functional Requirements

Below is the final list of the INFRs. Note that many of the original NFRs didn't need improvement as they were already clear enough.

INFR#1 System shall assist robot to navigate safely by providing routes suitable for robot
INFR#2 System should be able to convert route to a series of instruction that robot can follow

INFR#3 System should have app level & physical level security measures to prevent unintended parties from receiving the package
INFR#4 Robot should move at high speeds in order to make deliveries quickly
INFR#5 System shall determine the fastest route
INFR#6 Robot should be able to handle packages up to 10kg
INFR#7 Robot should be able to survive severe weather conditions
INFR#8 System shall be easily accessible to all members of the university
INFR#9 System should have over 90% successful deliveries
INFR#10 System should be able to handle several simultaneous deliveries
INFR#11 System will only be accessible to users currently on campus

5. Backwards/Forward Traceability

Requirements Traceability according to “Goten & Finkelstein” is:

“Requirements traceability refers to the ability to describe and follow the life of a requirement, in both a forwards and backwards direction (i.e., from its origins, through its development and specification, to its subsequent deployment and use, and through all periods of on-going refinement and iteration in any of these Phases.” (Leite, Doom, 2004, P 92 → P 94)

While this definition has become a common one when explaining requirements traceability, a simpler definition that captures the essence of requirements traceability is given below:

“Requirements traceability refers to the ability to define, capture, and follow the traces left by requirements on other elements of the software development environment and the traces left by those elements on requirements.” (Leite, Doom, 2004, P 92 → P 94)

Traceability has modes include:

Backwards traceability: Defined in 1.3
Forwards Traceability: Defined in 1.3

Below is table with the requirements (prelim & I(N)FR) with it's backwards / forwards traceability.

Table:

Prelim Reqs	Backwards / issues	Improved Reqs	Forward / effect
FR#1 User shall provide pickup location	FR is too vague and ambiguous. Needed additional details to solidify its definition.	IFR#1 User should be able to select a pickup location from the provided predefined locations	We can clearly define the pick up points that need to be placed on the map for the robot to navigate too
FR#2 User shall provide dropoff location	FR is too vague and ambiguous. Needed additional details to solidify its definition.	IFR#2 User should be able to select a dropoff location from the provided predefined locations.	We can clearly define the drop off points that need to be placed on the map for the robot to navigate too
FR#3 User shall see robot location	FR is too vague and ambiguous. Needed additional details to solidify its definition.	IFR#3 User should be able to track robots current location on the app	User can see current location on the map and ETA can be calculated based on current location

FR#4 Robot shall emit location	Needed more clarification on when / how often location will be emitted	IFR#4 Robot shall emit it's live location to the user	ETA and GPS location can be continuously updated in the app and not periodically giving better tracking details to the user.
FR#5 Robot shall move a certain distance	FR is too vague and ambiguous. Needed additional details to solidify its definition.	IFR#5 Robot shall be able to move to only the ground floor of university buildings & outdoor areas of the university campus.	Can define the perimeter of movement for the robot and better calculate obstacle free paths (due to the limited area and thus limited number of possible safe routes)
FR#6 Robot shall stop at the appropriate location	FR is too vague and ambiguous. Needed additional details to solidify its definition.	IFR#6 Robot shall stop at the user specified pick up / drop off predefined location	User tracking for receiving the his/her package will be more accurate
FR#7 Robot shall turn when necessary	Too similar and could be combined into fewer and clearer requirements	IFR#7 Robot should be able to detect obstacles at a close enough distance and go around it (turn) in	No change to functionality but combined in order to decrease requirement redundancy

		order to avoid collision	
FR#8 Robot shall detect obstacles	Too similar and could be combined into fewer and clearer requirements	IFR#7 Robot should be able to detect obstacles at a close enough distance and go around it (turn) in order to avoid collision	No change to functionality but combined in order to decrease requirement redundancy
FR#9 Robot shall avoid obstacles	Too similar and could be combined into fewer and clearer requirements	IFR#7 Robot should be able to detect obstacles at a close enough distance and go around it (turn) in order to avoid collision	No change to functionality but combined in order to decrease requirement redundancy

FR#10 Robot shall send notification when destination is reached	Doesn't specify who the notification will be sent to	IFR#8 Robot shall notify the sender and receiver when the destination is reached	We will now be sure that the notification is going to be receiving by both sender and receiver so they can receive their deliveries
FR#11 Robot shall unlock storage bin when correct user pin is provided	Same requirement no changes	IFR#11 Robot shall unlock storage bin when correct user pin is provided	Same requirement no changes
FR#12 Robot shall send notification when something is wrong	too vague and ambiguous. Needed additional details to solidify its definition.	IFR#12 Robot shall notify sender & receiver in case of delays or failure to deliver scenarios	Reduces scope of possible error messages for easier debugging on the app maintainers side
FR#13 User shall	Same requirement no changes	IFR#12 User shall	Same requirement no changes

sign up on app with university email, ID, and password		sign up on app with university email, ID, and password	
FR#14 User shall be able to login to app	Same requirement no changes	IFR#38 User shall be able to login to the app or to logout	Same requirement no changes
FR#15 User shall specify type of package being sent from a pre specified list of items	Same requirement no changes	IFR#13 User shall specify type of package being sent from a pre specified list of items	Same requirement no changes
FR#16 User shall set a pin at sign up that will be used for unlocking robot's compartment for sending / receiving packages	Same requirement no changes	IFR#14 User shall set a pin at sign up that will be used for unlocking robot's compartment for sending / receiving packages	Same requirement no changes
FR#17 User shall be notified of	Too similar and could be combined into fewer and	IFR#10 Robot shall	Combining requirements has lead to reduced

waiting time till robot's arrival at pick up / drop off point	clearer requirements	notify Users about pick up / drop off general arrival time, 2 min to arrival time & package waiting time in delivery queue	redundancy and thus savings in costs for the company and better functionality for the user.
FR#18 Robot shall emit a notification when it is busy on delivery and unable to receive packages	Same requirement no changes	IFR#15 Robot shall emit a notification when it is busy on delivery and unable to receive packages	Same requirement no changes
FR#19 User should be able to submit a complaint to the apps help center in case of unsuccessful deliveries	Same requirement no changes	IFR#16 User should be able to submit a complaint to the apps help center in case of unsuccessful deliveries	Same requirement no changes

FR#20 Robot should have different compartments with different doors for simultaneous deliveries	Same requirement no changes	FR#17 Robot should have different compartments with different doors for simultaneous deliveries	Same requirement no changes
FR#21 User shall have a contact list of sending / receiving users	Same requirement no changes	IFR#18 User shall have a contact list of sending / receiving users	Same requirement no changes
FR#22 System shall (RE)calculate the ETA using parameters sent by the robot	Same requirement no changes	IFR#19 System shall (RE)calculate the ETA using parameters sent by the robot	Same requirement no changes
FR#23 User shall be able to search for users by name or university ID and add / remove them to / from their	Same requirement no changes	IFR#20 User shall be able to search for users by name or university ID and add / remove them to / from their	Same requirement no changes

contact list		contact list	
FR#24 User shall be able to choose a user to send / receive a package from their contact list	Same requirement no changes	IFR#21 User shall be able to choose a user to send / receive a package from their contact list	Same requirement no changes
FR#25 User shall be able to accept / reject incoming / outgoing delivery requests	Same requirement no changes	IFR#22 User shall be able to accept / reject incoming / outgoing delivery requests	Same requirement no changes
FR#26 User can order food from pre-defined restaurants in the contact list	Same requirement no changes	IFR#23 User can order food from pre-defined restaurants in the contact list	Same requirement no changes
FR#27 User shall be able to type a message when specifying a delivery	Same requirement no changes	IFR#24 User shall be able to type a message when specifying a delivery	Same requirement no changes

FR#28 Robot should be able to tell which compartment is associated with which sending / receiving user when pin is entered	Same requirement no changes	IFR#25 Robot should be able to tell which compartment is associated with which sending / receiving user when pin is entered	Same requirement no changes
FR#29 Robot should be able to tell if an item is taken out of a compartment by an unauthorized user and notify admin with user details	Same requirement no changes	IFR#26 Robot should be able to tell if an item is taken out of a compartment by an unauthorized user and notify admin with user details	Same requirement no changes
FR#30 Robot shall tell sending / receiving user to enter pin to insert / receive package	Same requirement no changes	IFR#27 Robot shall tell sending / receiving user to enter pin to insert / receive package	Same requirement no changes
FR#31 Robot shall	Same requirement no changes	IFR#28 Robot shall	Same requirement no changes

calculate shortest path between pickup point and drop off point & recalculate after obstacle resolution		calculate shortest path between pickup point and drop off point & recalculate after obstacle resolution	
FR#32 Robot shall wait for 5 minutes at drop off location for receiving user to take his package	Same requirement no changes	IFR#29 Robot shall wait for 5 minutes at drop off location for receiving user to take his package	Same requirement no changes
FR#33 Robot shall confirm user when package is received successfully in case of sending user is different from receiving user	Same requirement no changes	IFR#30 Robot shall confirm user when package is received successfully in case of sending user is different from receiving user	Same requirement no changes
FR#34 Robot shall emit a notification to the users if it is unable to complete	Same requirement no changes	IFR#31 Robot shall emit a notification to the users if it is unable to complete	Same requirement no changes

a delivery, and take the package to the lost and found storage.		a delivery, and take the package to the lost and found storage.	
FR#35 Robot shall send required data for calculating ETA	Same requirement no changes	FR#32 Robot shall send required data for calculating ETA	Same requirement no changes
FR#36 Admins should be able to temporarily shut down the system for maintenance and notify users beforehand.	Same requirement no changes	IFR#29 Admins should be able to temporarily shut down the system for maintenance and notify users beforehand.	Same requirement no changes
FR#37 * Robot shall send a notification about the current package in the waiting queue	Too similar and could be combined into fewer and clearer requirements	IFR#10 Robot shall notify Users about pick up / drop off general arrival time, 2 min to arrival time & package waiting time in delivery queue	Combined to reduce redundancy which leads to lower costs for company and better experience for the user
FR#38 Robot shall send a notification about it's remaining battery life	Same requirement no changes	IFR#34 Robot shall send a notification about it's remaining battery life	Same requirement no changes

FR#39 Robot shall automatically charge in the charging dock when experiencing a low battery and notify users of time till completion.	Same requirement no changes	IFR#35 Robot shall automatically charge in the charging dock when experiencing a low battery and notify users of time till completion.	Same requirement no changes
FR#40 Robot shall have a physical key slot for opening the compartments in case of a bug occuring or for maintenance	Same requirement no changes	IFR#36 Robot shall have a physical key slot for opening the compartments in case of a bug occuring or for maintenance	Same requirement no changes
NFR#1 System shall assist robot to safely navigate on campus	Needed more clarification	INFR#1 System shall assist robot to navigate safely by providing routes suitable for robot	Same functionality, no changes, just more clarification. Test cases will be changed to only include routes for robot instead of general navigation tests
NFR#2 System shall determine the fastest route	Same requirement no changes	INFR#5 System shall determine the fastest route	Same requirement no changes

NFR#3 System shall navigate robot through selected route	Needed more clarification	INFR#2 System should be able to convert route to a series of instruction that robot can follow	Better clarifies for developer what he needs to do but same general functionality
NFR#4 Robot should be able to handle packages up to 10kg	Same requirement no changes	INFR#6 Robot should be able to handle packages up to 10kg	Same requirement no changes
NFR#5 Robot should be able to survive severe weather conditions	Same requirement no changes	INFR#7 Robot should be able to survive severe weather conditions	Same requirement no changes
NFR#6 System should have security measures to prevent unintended parties from receiving the package	Needed more clarification	INFR#3 System should have app level & physical level security measures to prevent unintended parties from receiving the package	Requirement is less vague and the system will be more robust due to multiple layers of security (physical and app level wise).
NFR#7 Robot should make deliveries quickly	Needed more clarification	INFR#4 Robot should move at high speeds in order to make deliveries quickly	Robot might need better alerting systems for passing people to warn them of it incoming towards them. More battery consumption due

			to higher speeds to battery needs to be upgraded. Route recalculation in case of obstacles needs to happen at a fraction of a second to make use of the high speeds of the robot.
NFR#8 System shall be easily accessible to all members of the university	Same requirement no changes	INFR#8 System shall be easily accessible to all members of the university	Same requirement no changes
NFR#9 System should have over 90% successful deliveries	Same requirement no changes	INFR#9 System should have over 90% successful deliveries	Same requirement no changes
NFR#10 System should be able to handle several simultaneous deliveries	Same requirement no changes	INFR#10 System should be able to handle several simultaneous deliveries	Same requirement no changes
NFR#11 System will only be accessible to users currently on campus	Same requirement no changes	INFR#11 System will only be accessible to users currently on campus	Same requirement no changes

References:

https://link.springer.com/book/10.1007/978-1-4615-0465-8