NOVEMBER 30, 2019

K-MAINTENANCE

Swe215 project - Submission 3

HAITHAM AL-SAEED

SALMAN AL-GHAMDI HASHIM AL-GHAMDI SALEM BAMUKHIER

Table of Content

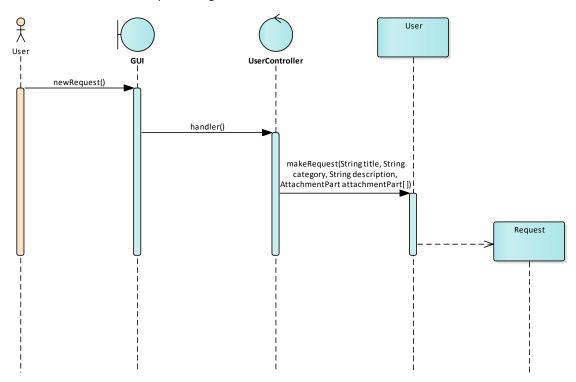
TABLE OF CO	TENT	0
Analysis Mo	DELS	1
Α.	Interaction Diagrams	1
В.	Class Diagrams	6
C.	Assumptions	8

Analysis Models

a. Interaction Diagrams

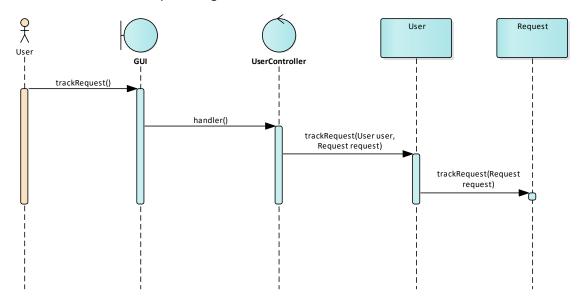
The team chose 4 use cases and made their sequence diagrams.

i. Make Request diagram:



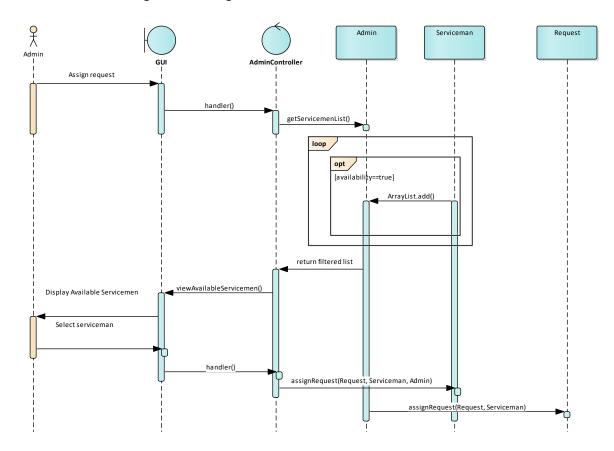
INTERACTION MESSAGES		
■ 1.0 'newRequest' from 'User' sent to 'GUI'.		
Synchronous Call.		
■ 1.1 'handler' from 'GUI' sent to 'UserController'.		
Synchronous Call.		
■ 1.2 'makeRequest' from 'UserController' sent to 'User'.		
Synchronous Call		
■ 1.3 " from 'User' sent to 'Request'. Synchronous Call.		
Synchronous Can.		

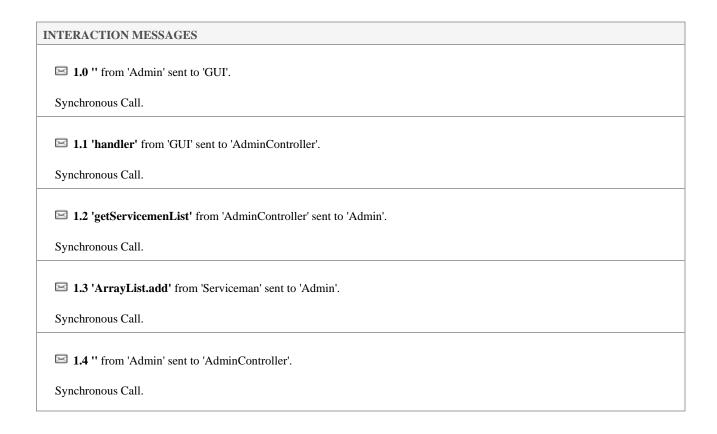
ii. Track Request diagram:



INTERACTION MESSAGES		
■ 1.0 'trackRequest()' from 'User' sent to 'GUI'.		
Synchronous Call.		
■ 1.1 'handler()' from 'GUI' sent to 'UserController'.		
Synchronous Call.		
■ 1.2 'trackRequest(User user, Request request)' from 'UserController' sent to 'User'.		
Synchronous Call.		
■ 1.3 'trackRequest(Request request)' from 'User' sent to 'Request'.		
Synchronous Call. Returns void.		

iii. Assign Service diagram:





■ 1.5 'viewAvailableServicemen' from 'AdminController' sent to 'GUI'.

Synchronous Call. Returns void.

■ 1.6 " from 'GUI' sent to 'Admin'.

Synchronous Call.

■ 1.7 " from 'Admin' sent to 'GUI'.

Synchronous Call.

■ 1.8 'handler' from 'GUI' sent to 'AdminController'.

Synchronous Call.

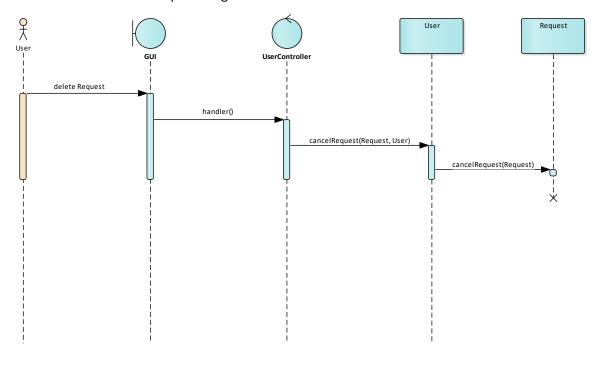
■ 1.9 'assignRequest' from 'AdminController' sent to 'Serviceman'.

Synchronous Call.

■ 1.10 'assignRequest' from 'Admin' sent to 'Request'.

Synchronous Call.

iv. Cancel Request diagram:

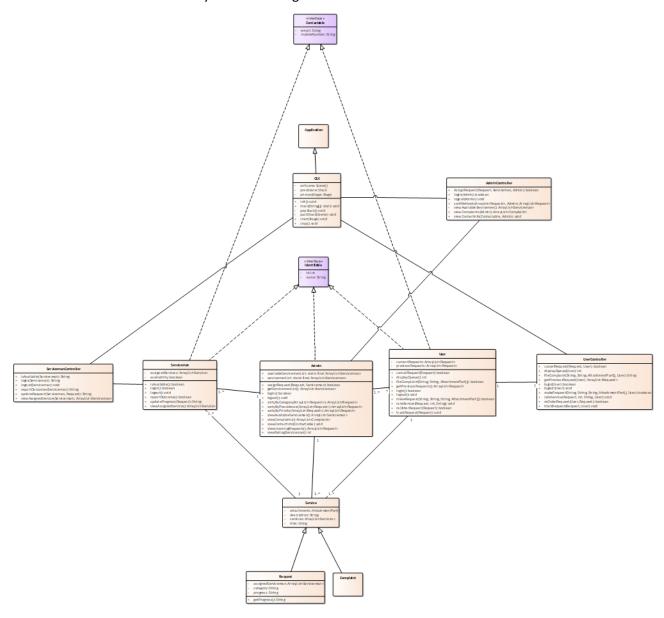


INTERACTION MESSAGES		
■ 1.0 " from 'User' sent to 'GUI'.		
Synchronous Call.		
■ 1.1 'handler' from 'GUI' sent to 'UserController'.		
Synchronous Call.		
■ 1.2 'cancelRequest' from 'UserController' sent to 'User'.		
Synchronous Call.		
■ 1.3 'cancelRequest' from 'User' sent to 'Request'.		
Synchronous Call.		

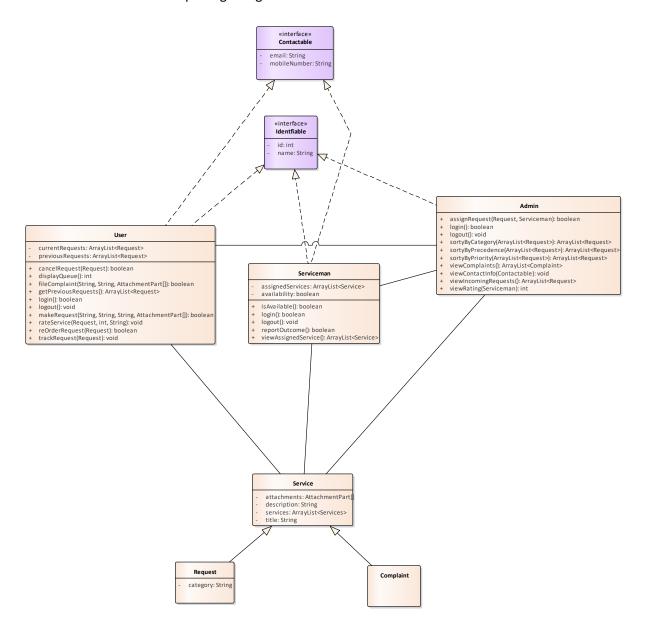
b. Class Diagrams

The team used Model–View–Controller (MVC) software design pattern to design the class diagrams.

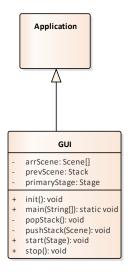
i. The whole system class diagram:



ii. Model package diagram:



iii. View package diagram:



iv. Controller package diagram:

ServicemanController + isAvailable(Serviceman): String + login(Serviceman): String + logout(Serviceman): void + reportOutcomes(Serviceman): String + viewAssignedService(Serviceman): ArrayList<Serviceman>

AdminController + AssignRequest(Request, Serviceman, Admin): boolean + login(Admin): boolean + logout(Admin): void sortMethods(Arraylist<Request>, Admin): ArrayList<Request> + viewComplaints(Admin): ArrayList<Complaint> + viewContactInfo(Contactable, Admin): void

cancelRequest(Request, User): boolean displayQueue(User): int fileComplaint(String, String, AttachmentPart[], User): Srting getProviousRequest(User): ArrayList<Request> login(User): boolean logout(User): void makeRequest(String, String, String, AttachmentPart[], User): boolean rateService(Request, int, String, User): void reOrderRequest(User, Request): boolean trackRequest(Request, User): void

c. Assumptions

- i. The team assumed that the ID used for users will not exceed 9 digits (i.e. the integer limit).
- ii. The team assumed that verifying credentials will be done by KFUPM system, so no need for including it in the component designed.
- iii. The team assumed that the user interface will be implemented using JavaFX, so the View package was designed based on JavaFX.