Faculty of Computers, Informatics and Microelectronics

Technical University of Moldova

AMSI

Laboratory work #3

Project description. Modeling your project with Sequence Diagrams. Functional and Non-Functional Requirements.

Authors:
Tanașciuc Macarie

Supervisor:
Mihail Gavriliță

Chișinău, 2017

Laboratory Work #3

Topic:

Modeling your project with Sequence Diagrams. Functional and Non-Functional Requirements.

Tasks:

- Model the application using 3 Sequence Diagrams;
- Analyze the Functional and Non-Functional Requirements for the project.

1 Model the application using 3 Sequence Diagrams

The following actions in all 3 Sequence Diagrams presented bellow are **common for all users**:

- **AplicationStartup()** represents the initialization of the application and it returns the Interface output to all types of users through **OutputInterface()**
- **Continue(selection)** represents the passing point from initial login form to the selection form.It request the form from the database through **RequestContinue(selection)** and returns the form through **ReturnContinue()**
- **Continue(output)** represents the passing point from selection from to the output form.It request the form from the database through **RequestContinue(output)** and returns the form through **ReturnContinue()** once done the output is show to the user by **OutputForm()**.
- **UpdateView()** from *output form* represents a view update that will give the readable onscreen filtered data to the user by **OutputFilteredView()** it will request the **UpdateView()** from the database apply the filter previously set **"ApplyFilter()"** then a request of data from the TrackMethods will be done "**GetData()** and **GetDataReturn()**" then the data will be returned to the interface through **ReturnUpdateView()**
- **SaveData()** represents a similar execution of **UpdateView()** just that the output will be in a specific data format;It will request the **SaveData()** from the database apply the filter previously set **"ApplyFilter()"** then a request of data from the TrackMethods will be done "**GetData()** and **GetDataReturn()**" then the data will be returned to the interface through **SaveDataReturn()** and the user will get the save the data.*format* through **SaveDataFormat()**
- **OpenData()** is a instance that will be issued by the user to open a data.*format* from his OS and use the **SetFilter()**,**UpdateView()** and **SaveData()** with the context of the data that was opened with **OpenData()** request to the Local OS Database and returned with **OpenDataReturn()** and **OutputDataForm()**

Sequence diagram №1

- **AllowOnline()** is a checkbox for **SetMode(pass,online)** but this option isn't needed nor can be checked for ***OfflineUser*** therefor it is unchecked.
- **MethodsSelection()** represents the selection of tracking messages that is then transmitted to the database to ask for their initialization through **InitializationMethods()** and individual initialization through **StartMethods()**
- **SetFilter()** represents the setting of a filter the the data that will be saved in the database and then used in **UpdateView()**,**OpenData()** and **SaveData()** through **ApplyFilter()**.

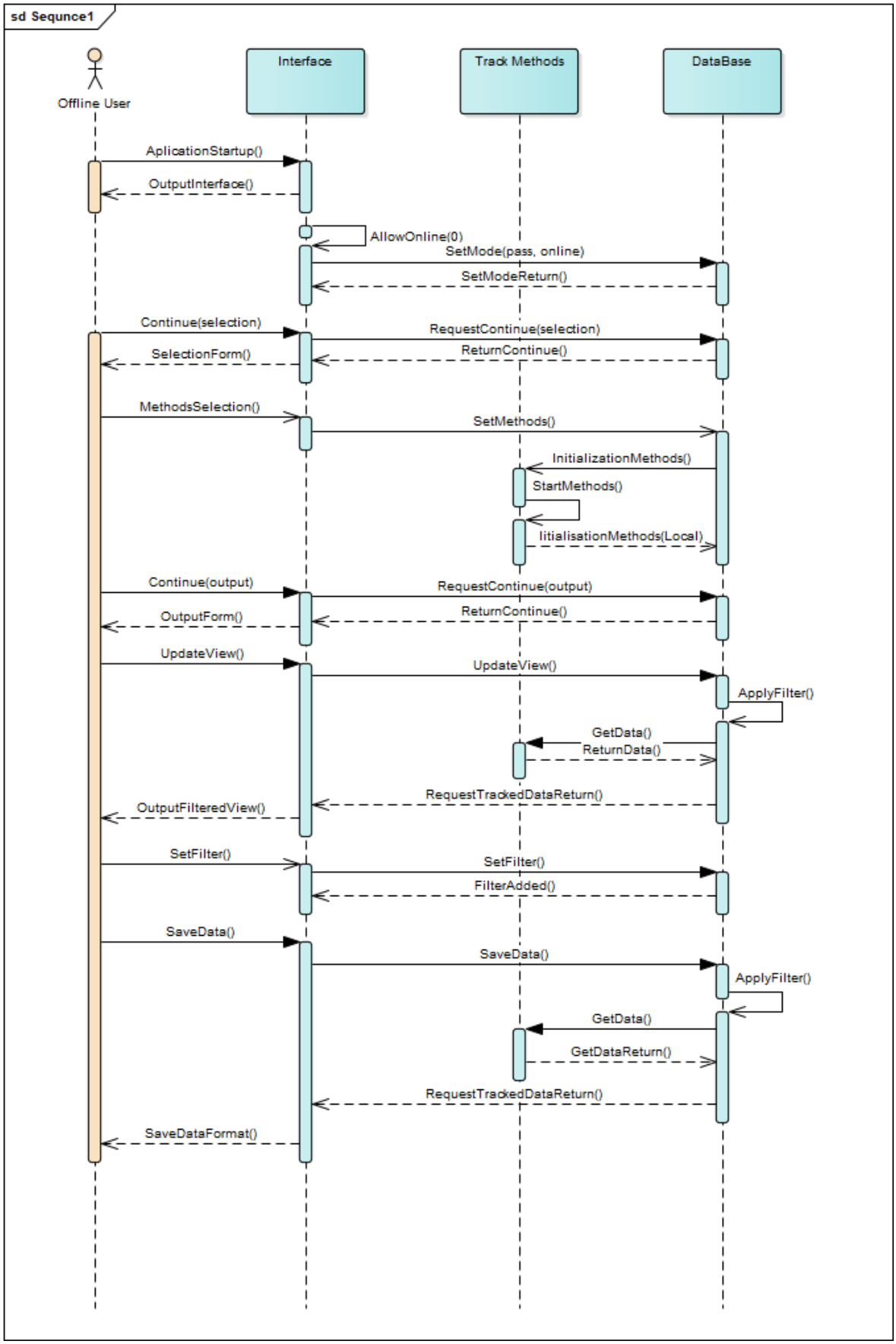

Figure 1.1 – OfflineUser

Sequence diagram №2

- **Addpass()** represents the input of a password in the initialization from where the user sets the mode he wants to work in; As the user is operating in **Online(Admin)**:
  - **SetPass()** sets a security password in the database, which allows checking the **AllowOnline()** to be checked and afterwards saving the mode through **SetMode(pass,online)** in database which will allow to access **AddTrackID(ID,Pass)**.
  - **AddTrackID(ID,Pass)** represents an option to add as many ID's with their respective password to track.
- **MethodsSelection()** represents the selection of tracking messages that is then transmitted to the database to ask for their initialization through **InitializationMethods()** and individual initialization through **StartMethods()** and will return the activation response through **InitializationMethods(Local)** afterwards it will send and request to the local users that the admin has choosed to track **AdminMethods(Send)** and will receive through **AdminMethods(return)** if the user is connected and will see the output on screen through **LocalUserConnected():Yes**.
- **SetFilter()** represents the setting of a filter the the data that will be saved in the database thorough **SetFilter()** function which will afterwards send the filter to the local users and request the filtered data **RequestData()**,**ReturnRequestData()** and then used in **UpdateView()**,**OpenData()** and **SaveData()** through **ApplyFilter()**.

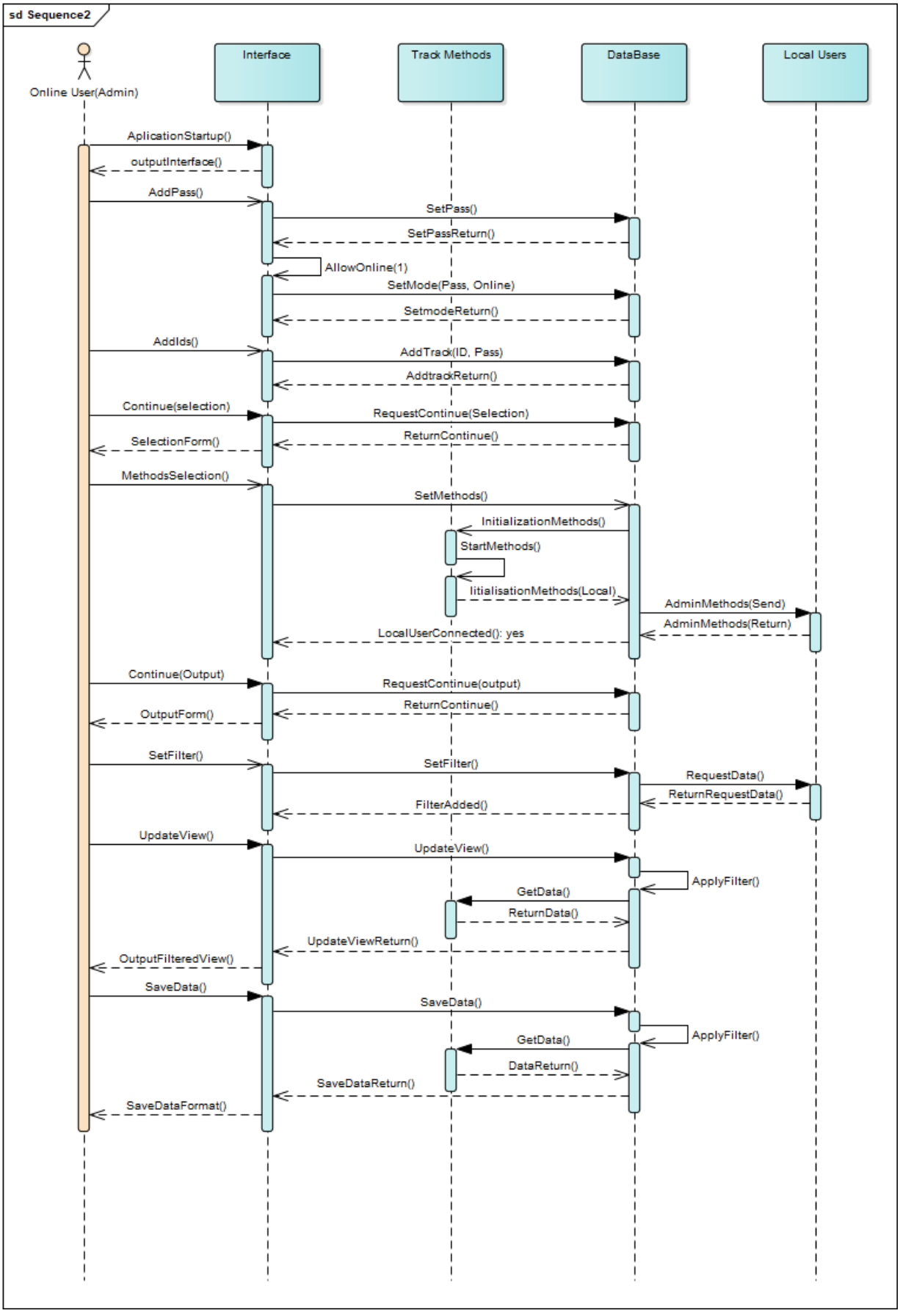

Figure 1.2– OnlineUser(Admin)

Sequence diagram №3

- **SetPass()** sets a security password in the database, which allows checking the **AllowOnline()** to be checked and afterwards saving the mode through **SetMode(pass,online)** in database.
- **MethodsSelection()** represents the selection of tracking messages that is then transmitted to the database to ask for their initialization through **InitializationMethods()** and individual initialization through **StartMethods()** and will return the activation response through **InitializationMethods(Local)** afterwards if it will receive the **AdminMethods(send)** from the admin of the group which will ask for initialization of the methods that weren't initialized through **InitializationMethods()** and individual initialization through **StartMethods()** and will return the activation response through **InitializationMethods(Admin)**
- **SetFilter()** represents the setting of a filter the the data that will be saved in the database thorough **SetFilter()** function which will be used in **UpdateView()**,**OpenData()** and **SaveData()** through **ApplyFilter()**.

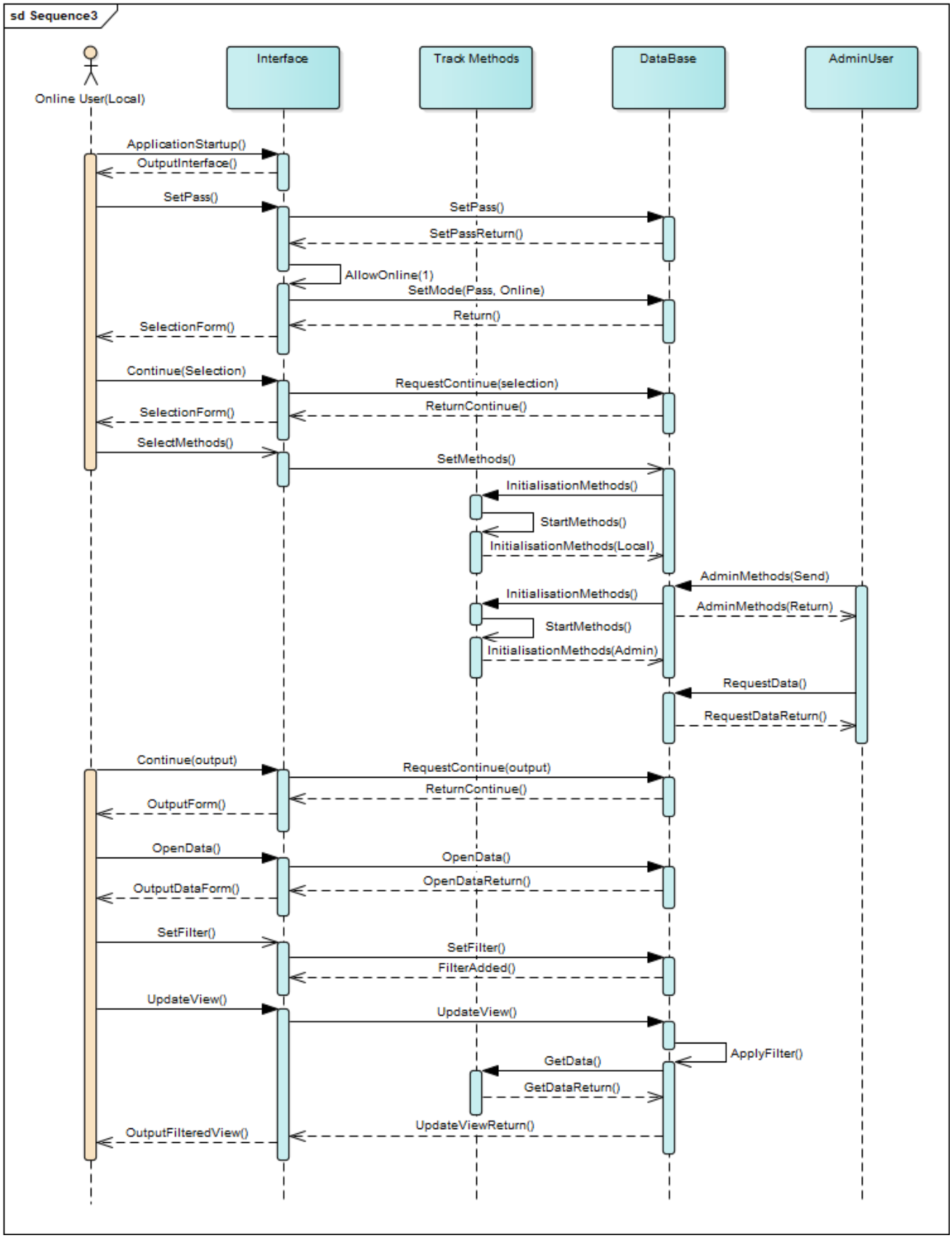

Figure 1.3– OnlineUser(Local)

2 Analyze the Functional and Non-Functional Requirements for the project.

Functional Requirements

- For Track Methods to save tracked data in statistic form.
- For Track Methods to ask for the permissions from the user of what data can be tracked.
- For Filter to show what data can be accessed and requested from the database.
- For Showing the user how to use the Output form.

Non-Functional Requirements

- Product Requirements:
  - Number of Processor: 2
  - Disk capacity min: 6Gb
  - Operating system: Windows ,Ubuntu,Mac Os
  - Database vendor: Microsoft(MySql)
- Efficiency requirements:

  The application should operate with local users tracked data connected with an admin in within a reasonable time.
- Portability requirements:

  The application should run all modern operating systems and be able to interface major relational database systems from various vendors.
- Privacy requirements:

  The application should not reveal private passwords nor any tracked data to outside of network local/admin users.

Conclusions

In this laboratory work i learned how to create Sequence Diagrams and understanding the Functional and non-functional Requirements of the project.

References

1 Learn Unified Model Language,https://www.tutorialspoint.com/uml/

2 Wikipedia,https://en.wikipedia.org/wiki/Unified_Modeling_Language

3 Notes on UML course given by professor and laboratory assistant.

4 Collaboration Diagrams - UML - Interaction Diagrams https://www.tutorialspoint.com/uml/uml_interaction_diagram.htm