COMP 6231

Distributed System Design

Assignment 3 Report

Web Service Implementation of the Distributed Class Management System (DCMS)

Submitted to: Professor Mohamed Taleb

By

Stallone Mecwan: 40161375

Jay Patel: 40163706

TABLE OF CONTENTS

TABLE OF FIGURES

Technique used

- We used **Web Services** to implement the communication between Manager/Clients and servers (MTL, LVL and DDO).
- We used **UDP** to implement the communication between servers for counting Records on each server and transferring records across the servers.
- We used **HashMap** to store the records of teachers and students.
- We used **multithreading** technique so that multiple clients can act simultaneously.
- We used **synchronization and semaphore** technique to keep the integrity of data while modifying it, so the server can maximize the concurrency.
- The record class has been made to extend the Serializable class to implement the transfer record functionality as it allows writing of state of an object (here the record) into a byte-stream.

Design architecture

WebInterface

File that defines all the operations that can be used by the managers (Clients of this system).

- createTRecord
- createSRecord
- getRecordCounts
- editRecord
- displayAllRecords
- displayRecord
- transferRecord

MTL/LVL/DDOClass

These files' implements the WebInterface by specifying the route of Web Interface as endpointInterface. All the functions are implemented here. Some other functions are also defined here, i.e. validRecordID(), dateFormatChecker() (Date format checker for attribute statusDate of Student Record), setUpHashMap() (initialising HashMap here) and findRecord().

MTL/LVL/DDOServer

These files are responsible for publishing the server and running it.

The publish address is assigned here.

The Server file has the server's main() method, which:
• Creates an Endpoint variable which published the server by passing an address and the object of the class which implements the WebInterface.

• Further it creates an object of ClassService (MTL/LVL/DDOClassService), which is further used to create an object with the ClassPort to use the methods defined in the Interface and implemented in the Class files.

As a starting point, some records are created here (by some default managerID like MTL/LVL/DDO0000) to fill the initial database. There are direct method invocations here, so we have also put validations in the MTL/LVL/DDOClass files.

MTL/LVL/DDOClient

These files contain client specific code. Here choices are provided to the user (manager) for which operation is to be performed. Validation of inputs is provided in these files. These files are the starting point of the application. They ask for the managerID whose prefix (first three characters) are verified and if they match with the specific client, then only access is provided.

Here, validation of managerID takes place, if it is not in the format MTL/LVL/DDO followed by four digits, then it won't work.

Record

Record class implements Serializable class.It is used to create TeacherRecord and StudentRecord (subclasses). It has two main attributes of Record, RecordID and Name. Also, a toString() method is defined to appropriately format and print the data for the Client.

TeacherRecord

This class is used for storing various details of Teachers. It stores the following details of a student: First Name, Last Name, Address, Phone number, Specialization, Location. Similar to the Record class, Student Record class has a toStringT() method which formats and prints the data for the client.

StudentRecord

This class is used for storing various details of Students. It stores the following details of a student: First Name, Last Name, Courses he is registered in, status, status date. Similar to the Record class, Student Record class has a toStringS() method which formats and prints the data for the client.

Log

This file is used to create log files and add information of all activities taking place in the application.

Server logs are named as MTL.txt, LVL.txt, DDO.txt are created in the project directory. Client logs are saved as per the managerID used and are saved in the folder of "Logs".

Class Diagram - Web Service Implementation of the DCMS

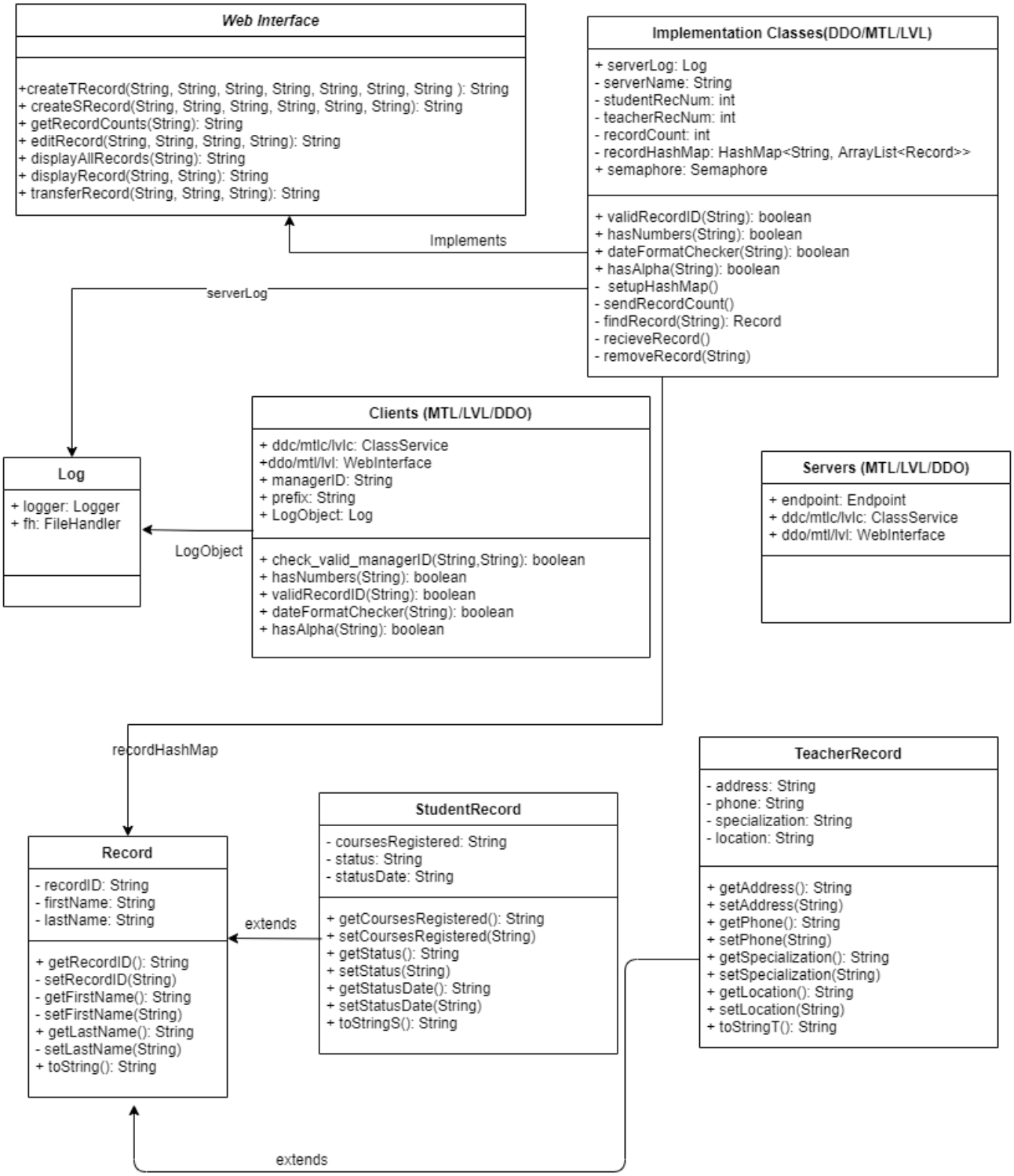

Figure 1: Class Diagram

Flowchart - Web Service Implementation of the DCMS

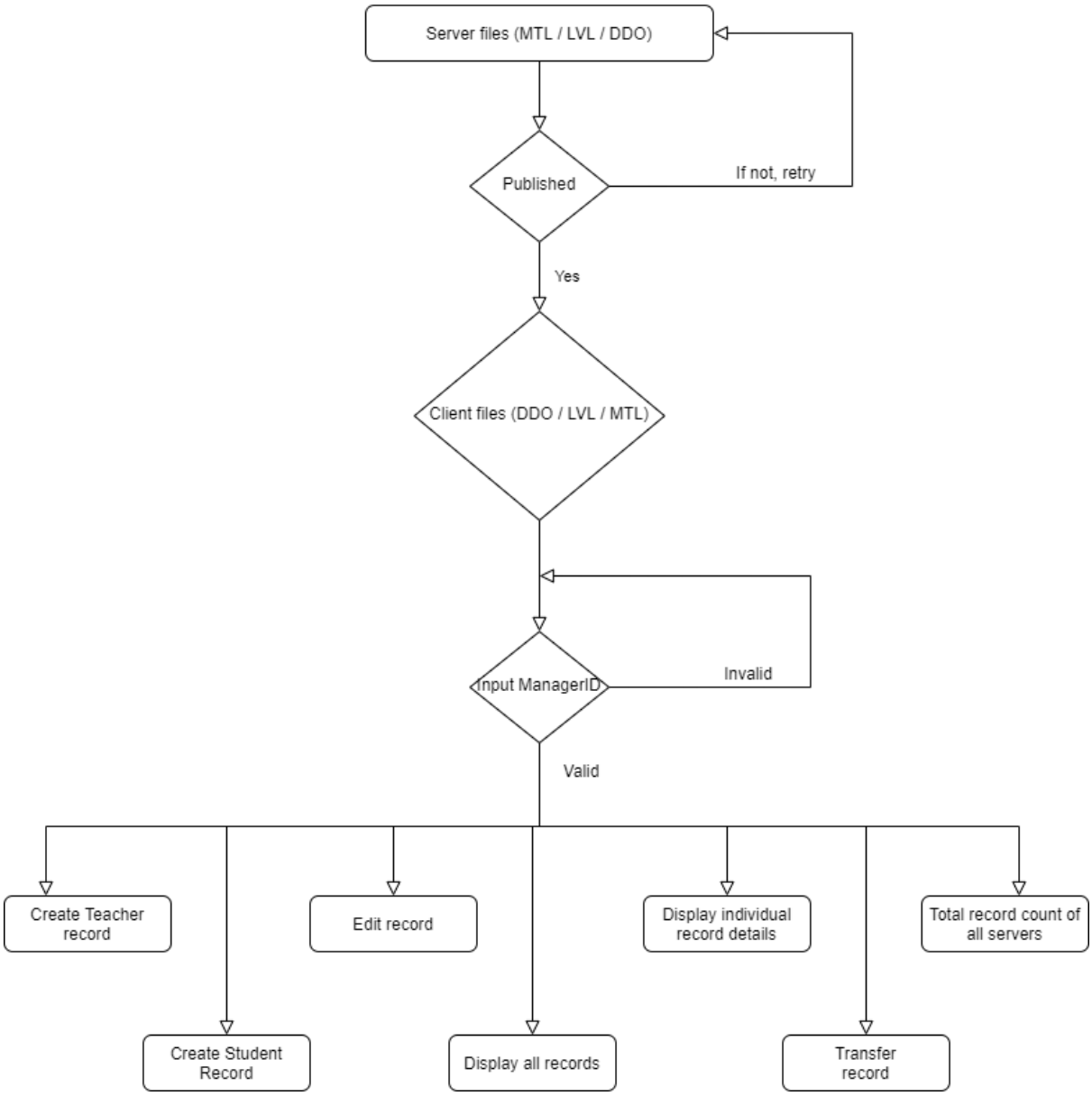

Figure 2: Flowchart

Development of the application

First, we developed the WebInterface, then coded the Class files (MTL/DDO/LVLClass) which implemented the interface (through endpointInterface) and defined all its methods.

Then we created the Server files which mainly worked for publishing at that time, we published all the servers and then used the wsgen command

wsgen -verbose -cp . webservices.Implementation.DDOClass -keep -wsdl

By running this command in the directory out->production->project name, we got 14 files generated in the jaxws folder, we copied all of them in the src folder.

Then we executed this command:

wsimport -keep -d . -p webservices.WsimportFiles http://localhost:8888/DDOClass?wsdl

We did the same for all servers and we got files in our out folder, again copied it back to the src folder.

Then at the end, we created the Client files which takes ManagerID as an input and displays the list of functions that the client can run

Before starting the application

We have used Intellij IDEA Community edition (version 2020.3.2) and JDK 1.8 for development of this application.

Starting the application

1) Start servers MTL, LVL and DDO (shown below) (all servers are required to be running, for record count and transfer record functionality)

2) Start Client files

Data structures

Array List

All the details of students and teachers are stored in an array list

HashMap

It consists of an id (string type) as the key and values are Array Lists that contain the records of students and teachers itself.

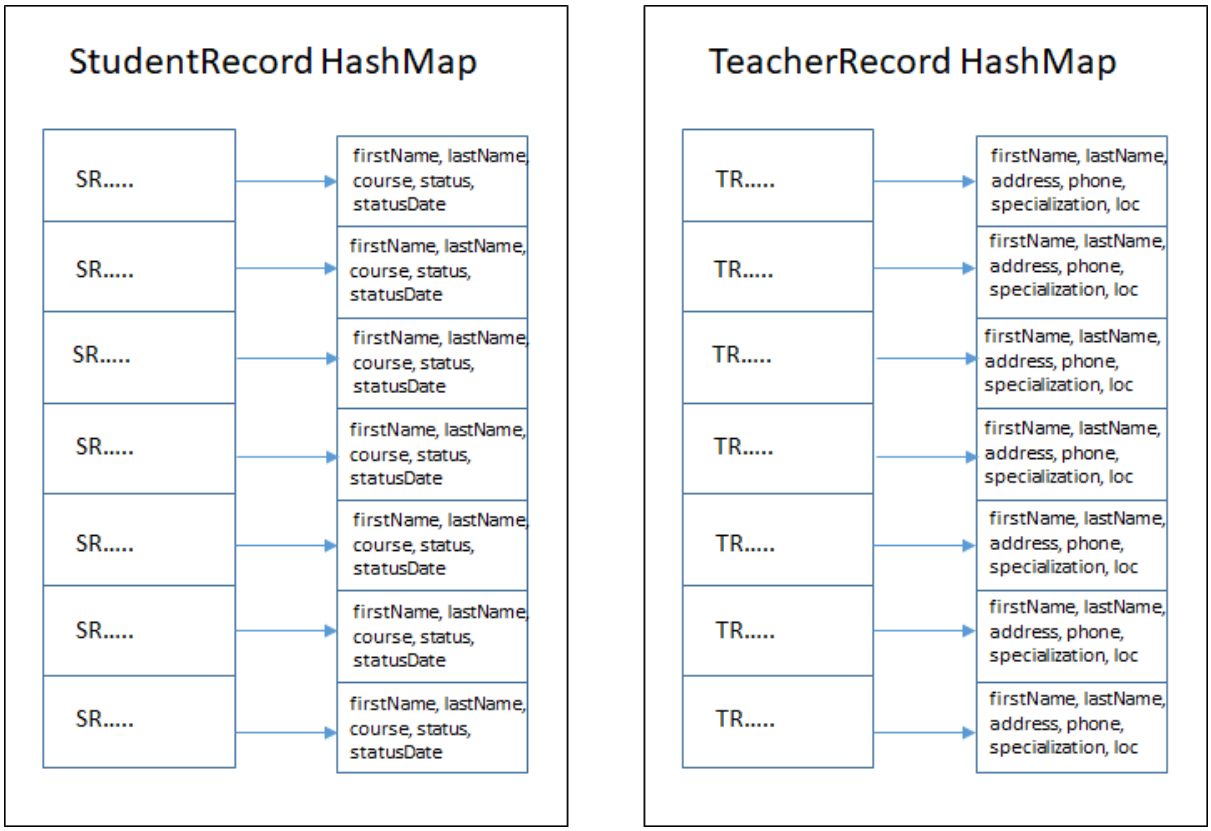

Figure 3: HashMap

Test Scenarios

No new functionality has been asked to be introduced in this assignment. Due to this fact the test scenarios from the previous builds remain the same.

However there is a change in our system where we are running Clients from separate files. Considering this, one new test case has been included.

- This case shows that when MTLClient is running, prefixes like DDO and LVL won't be allowed for managerID.
- When appropriate managerID is inserted as the input, then only the menu would be displayed.
- Similarly, we have put validations on all the Client files.

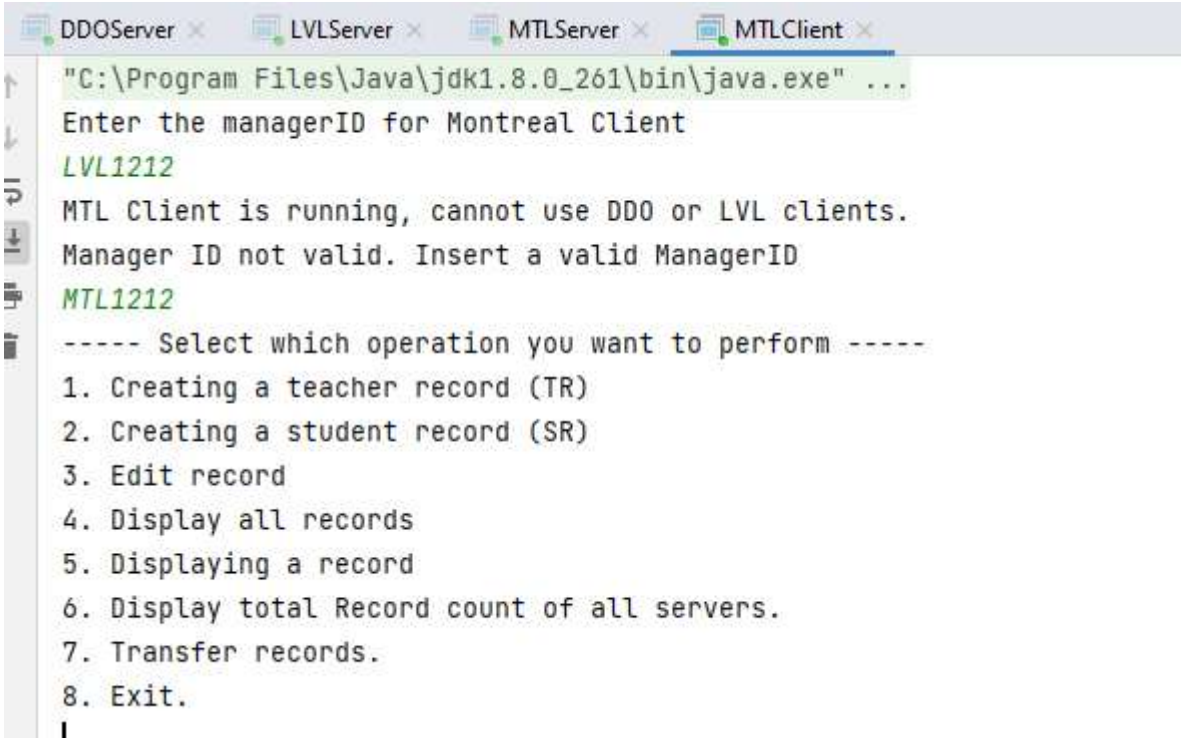

Figure 4: Invalid ManagerID

For the entire application we have created functions that test several inputs.

- validRecordID()
- hasNumbers()
- dateFormatChecker()
- hasAlpha()

These functions test the conditions on the input provided by the user to the system.

- Here, we will be demonstrating a server trying to transfer record back at it again (self-transfer) that would cause an error and that should not be accepted.

```
Enter manager ID
MTL1111
----- Select which operation you want to perform -----
1. Creating a teacher record (TR)
2. Creating a student record (SR)
3. Edit record
4. Display all records
5. Displaying a record
6. Display total Record count of all servers.
7. Transfer records.
8. Exit.

7
Enter the recordID of the record to transfer:
SR10000
Enter the center server name:
MTL
Transfer failed
Cannot transfer to the same server
```

Figure 5: Self transfer not allowed

Below is the screenshot of MTL1111 log file

```
Jul 04, 2021 11:53:55 AM Clients.MTLClient run
INFO: The Manager MTL1111 attempting to transfer a Record
Jul 04, 2021 11:54:08 AM Clients.MTLClient run
INFO: The Manager MTL1111 attempted to transfer the Record SR10000 to itself and it failed
```

Figure 6: Self transfer not allowed log entry

- If we enter an invalid Server name, then it would not work and ask for a valid server name

```
7
Enter the recordID of the record to transfer:
TR100
Invalid recordID, insert a valid ID
TR10000
Enter the center server name:
wed
Invalid Server Name mentioned
ed
Invalid Server Name mentioned
d
Invalid Server Name mentioned
LVL
Record transferred
```

Figure 7:Invalid server name

- If a valid server name is inserted, then the record gets successfully transferred.

```
Enter the recordID of the record to transfer:
TR10001
Enter the center server name:
LVL
Record transferred

Jul 04, 2021 12:03:44 PM Clients.MTLClient run
INFO: The Manager MTL1111 attempting to transfer a Record
Jul 04, 2021 12:03:47 PM Clients.MTLClient run
INFO: The Manager MTL1111 transferred the Record TR10001 to server LVL
```

Figure 8:Demonstration of transfer with log entry

This is the screenshot of LVLServer log file.

```
Jul 04, 2021 12:03:47 PM ServerImplementation.LVLClass recieveRecord
INFO: TR10001 has been added to the mentioned server
```

Figure 9: Serverlog file entry

- If we try to transfer a record which is not present in the server, then it would give a record not found message.

```
Enter the recordID of the record to transfer:
SR00000
Enter the center server name:
LVL
Record not found
```

Figure 10:Invalid record

- First Name and Last name are validated (cannot be empty and can't contain numbers)

```
Enter First name:
wed12
A name can not contain numbers, please insert valid input.
Stallone
Enter Last name:
Stallone145
A name can not contain numbers, please insert valid input.
Mecwan
```

Figure 11:First name last name validation

- Phone number must be in this format only, otherwise it would not be accepted

```
Enter Phone number in the format (514-888-9999):
5148889898
Invalid phone number
Try another
514-888-9999
```

Figure 12: Phone number validation

- Status date would be only accepted in this format dd/mm/yyyy:

```
Enter status date: (format - dd/mm/yyyy)
4/4/98
Invalid date format, please insert again in this format dd/mm/yyyy
04/16/1998
Invalid date format, please insert again in this format dd/mm/yyyy
04/04/1998
```

Figure 13: status date validation

All the test scenarios from the previous implementations continue to exist.

Important part/ Difficulty

The web service implantation was supposed to be created from the previous CORBA code.

In doing so, wsgen command posed a lot of difficulties in the beginning.

"*Inner classes annotated with @javax.jws.WebService must be static*". This error kept on repeating due to which we had to create the DDO/LVL/MTLClass files from the start again with the previous code to debug the problem.