*191071902*

*Srushti Shah*

*T. Y. B. Tech. C. S.*

*Software Engineering II*

***TA Assignment***

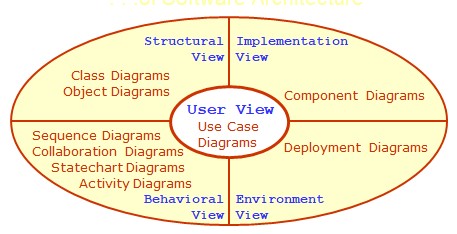
**QUESTION**  
OOAD USING UML

(i) Give structural view and Behavioural view.

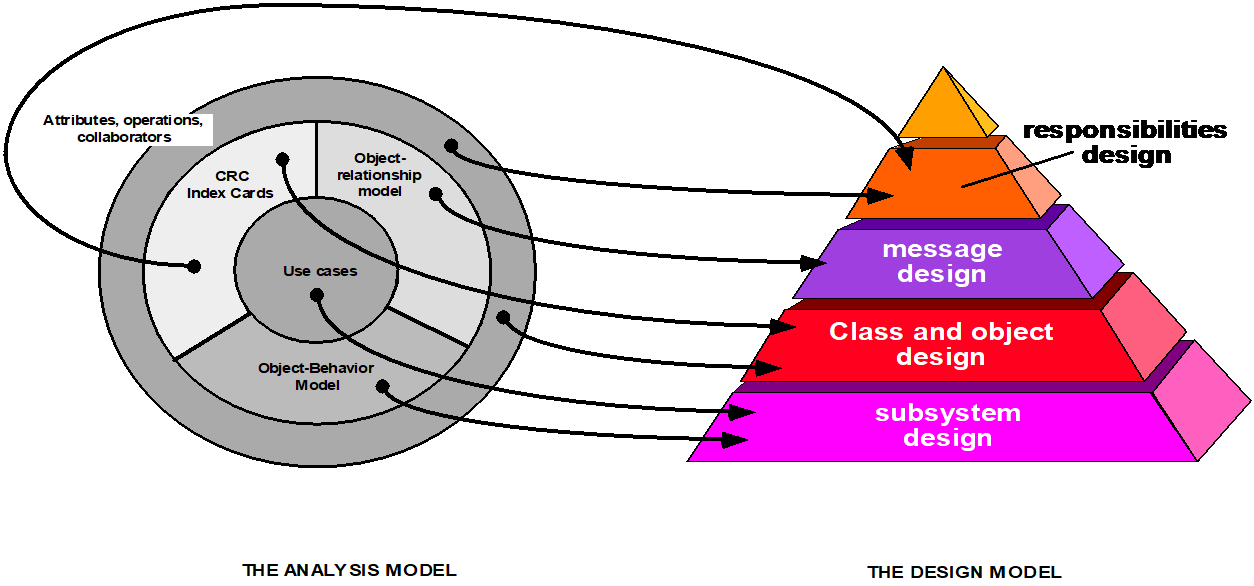
(iii) Implement the project and illustrate implementation view.

(iii) Illustrate the environmental view with specifications.

Use the following dimensions for answering your experiment.



**Map the analysis Model with design Model. Use the following dimensions for the design of object-oriented software.**

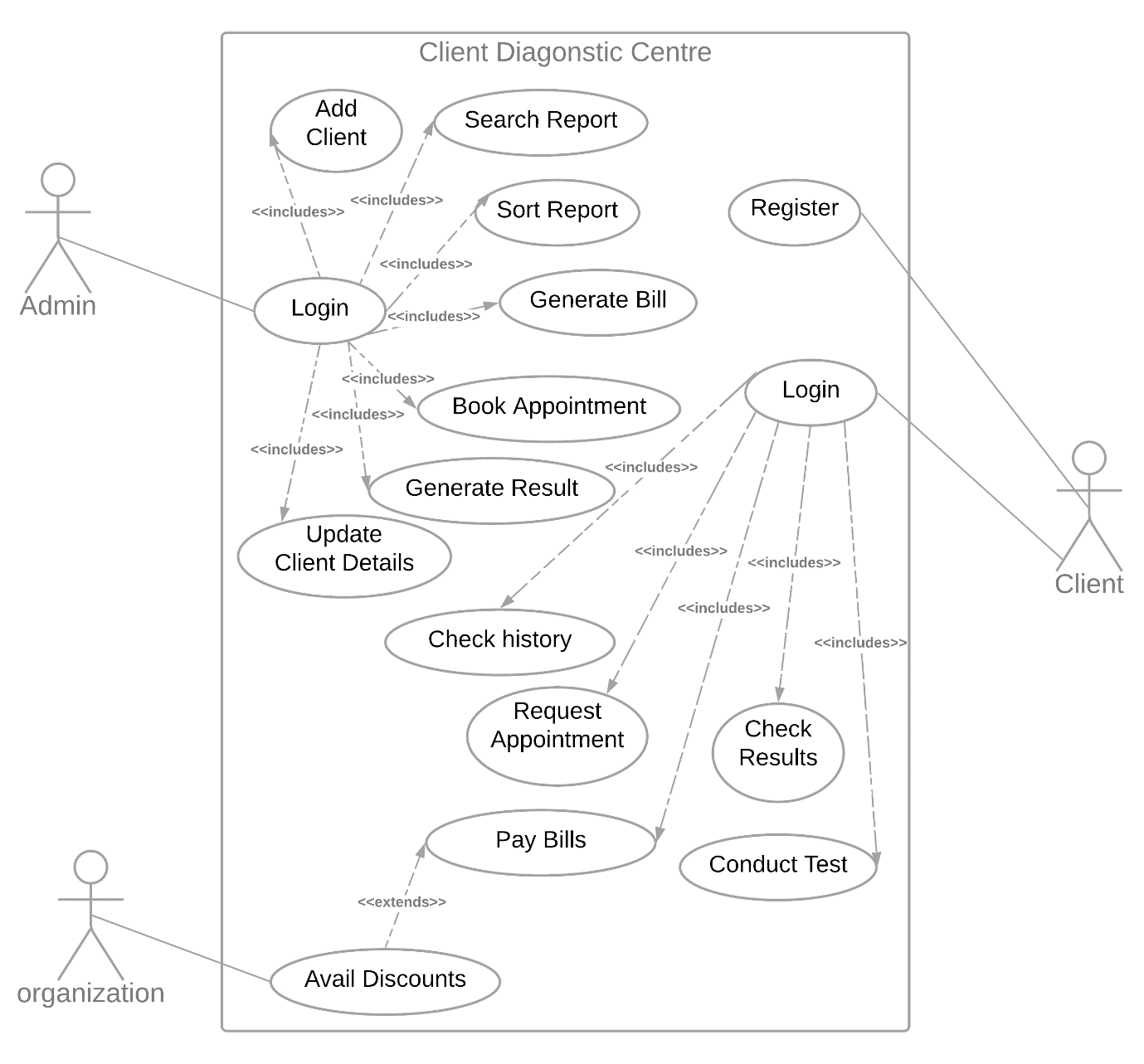


Design the software architecture, algorithm and data structure design and authoring system for the given project.

**CASE STUDY  
Diagnostic Client Co-ordination System**

Diagnosis centres need to maintain good relations with their clients. There is always a database of clients maintained in every diagnostics centre. The database includes client contact details as well as his tests done. It includes data like contact numbers, name, address, email etc. Clients may conduct various tests at the diagnostic centre including blood test, urine test as well as cholesterol, liver and kidney tests. The system must be capable of evaluating patient bills and providing them in printable format. The system also allows admin to store and update patient data for existing and new patients. Since every client must be working for some or the other organization and many organizations have contracts with diagnostics centre to provide discounts to their employees, the system tracks this information too. Now the system also keeps count of the number of tests undergone by a patient in a year. After 10 tests in a year the system provides 5% discount to the patient from the 11th test in a particular year. The system also allows admin to sort out tests frequency and search particular patient records. Company wise discount rates are fed in to the system.

***User View***

***USE CASE DIAGRAM: Client Diagnostic Centre***

***USE CASE TEMPLATE***

1. ***Register Client on System***

|  |  |
| --- | --- |
| Name | Register |
| Summary | A new user (a user which doesn’t have an account) should first register into the system to use it. This feature will allow the user to enroll into the system if the user is a new user and doesn’t already have an existing account. |
| Rationale | The user can access the system |
| Actor | Client |
| Pre-Condition | None |
| Basic course of Event | 1. User opens the desired module of the system. 2. The system displays GUI the registration. 3. User enters the valid details required to join the system 4. Submits the form |
| Post- Condition | User is registered |
| Alternate Flow | * + - 1. User goes back to the Home Page |

1. ***Login***

|  |  |
| --- | --- |
| Name | Login |
| Summary | The user can use the features of the system |
| Rationale | After registration of the user, the valid user can login into the system |
| Actor | Admin, Client |
| Pre-Condition | The login name and password should match with the login name and password provided while registering.  If the username and or password do not match, the user cannot login successfully into the system. |
| Basic course of Event | * 1. User opens the desired module of the system.   2. The system displays GUI the login form   3. The user writes its username, password and type   4. If valid, user can login into the system. If the username and or password do not match, the user cannot login successfully into the system. |
| Post- Condition | User goes to his/her dashboard |
| Alternate Flow | * + 1. The username and password are blank     2. The type of user is not specified     3. The username and password do not match |

1. ***Book Test Appointment***

|  |  |
| --- | --- |
| Name | Book Test Appointment |
| Summary | The patient requests to book test and admin assigns a slot |
| Rationale | The appointment is booked |
| Actor | Client and Admin |
| Pre-Condition | * + - 1. The client is a valid user       2. The slot he asks for is free |
| Basic course of Event | * 1. The client (patient) logs into the system   2. The patient requests for an appointment using test name, preferrable date and time   3. The admin checks if the slot is available   4. The admin assigns the slot if free to the patient (client) |
| Post- Condition | The client is given the free slot for test |
| Alternate Flow | * + 1. The admin denies the appointment as he finds the user malicious |
| Alternate Flow 2 | * + - 1. The slot is not available  1. The admin sends a message to request for another slot 2. The admin assigns a nearby time slot |
| Exceptions | * 1. The appointment is marked as urgent   The admin adjusts the time slot given to someone else   * 1. System fails to given appointment   Patient has to request again |

1. ***Generate Bill***

|  |  |
| --- | --- |
| Name | Generate Bill |
| Summary | The client is has to pay the bill for the corresponding test/s |
| Rationale | The amount to be payed is generated |
| Actor | Client and Admin |
| Pre-Condition | The appointment is booked |
| Basic course of Event | * + 1. The appointment for a particular test is booked     2. The system generated the Amount of the test to be paid by client as stored in its database |
| Post- Condition | The bill is printed and given to the client |
| Alternate Flow | * + - 1. If number\_of\_test of patient is 11  1. Provide 5% discount 2. Calculate the final amount |
| Alternate Flow 2 | 1. The client belongs to a organization who has a contract with the system 2. Provide corresponding discount to the client 3. Calculate Total |

1. ***Generate Test Report***

|  |  |
| --- | --- |
| Name | Generate Test Report |
| Summary | After giving the test, the client can check the report of the test |
| Actor | Client |
| Pre-Condition | The test is conducted |
| Basic course of Event | 1. Go to the Test section 2. Check the report of the test you want |
| Post- Condition | The list of report is given. |

1. ***Search Patient History***

|  |  |
| --- | --- |
| Name | Search Patient History |
| Summary | The administrator can check the search and check the record of patient |
| Actor | Admin |
| Pre-Condition | The list of record is displayed |
| Basic course of Event | 1. The admin logs into the system 2. The admin goes to the search tab 3. Enters the name of patient/patient ID 4. The details and history of the patient are displayed |
| Post- Condition | The patient history is displayed |
| Alternate Flow | 1. There is no patient with the given patient name/ID 2. No results displayed |

1. ***Store/Update Patient Details***

|  |  |
| --- | --- |
| Name | Store/Update Patient History |
| Summary | The administrator can store or update the record of patient |
| Actor | Admin |
| Pre-Condition | The patient/client is added to the system |
| Basic course of Event | 1. The admin logs into the system 2. The admin searches for the patient using patient ID/name 3. The patient is selected 4. The option to add/update the details is given 5. The details are entered/edited 6. The data is stored/updated into the database |
| Post- Condition | The database is updated |
| Alternate Flow | 1. There is no patient with the given patient name/ID 2. No results dispalyed 3. A new patient is added |

1. ***Sort the tests and manage frequency***

|  |  |
| --- | --- |
| Name | Manage frequency |
| Summary | The client when completed a test it is added to his list |
| Actor | Client |
| Pre-Condition | The client is registered to the system |
| Basic course of Event | 1. The test appointment is given to client 2. Client finishes the test 3. Update the no\_of\_tests in the database |
| Post- Condition | The database is updated |

***Structural View***

***CLASS DIAGRAM***

1. Identify the Classes

* Admin
* Client
* Organization
* Test
* Bill
* Report

1. Identify Attributes

* Admin

+ username:String

- password :String

+ email:String

+gender: Gender

-phone\_no: int

+designation: string

+created\_on: Date

* Client

+ name:String

+ patientID:int

- password :String

+ email:String

+gender: Gender

-phone\_no: int

+DOB: Date /age:int

-address:String

+email: String

+organizationID: int

+no\_of\_test: int

* Organization

+ name:String

+ oraganiztionID:int

+category:String

* Test

+testID:int

+testName: String

+createdOn:Date

+conductedOn:Date

+type: String

+patientID:int

+cost: float

+sample\_id:int

+test\_result:String

+comments:String

+ conductedBy:text

* Bill

+patientID: int

+testID:int

+billID: int

+paymentDate: Date

+dueAmount: float

+paidAmount: float

+TotalCost: float

-paymentMethod: String

+createdOn:Date

+discountpercent: int

* Report

+TestID: int

+PatientID: int

+MainReport: String

+Detailes: String

+Comments: String

1. Identify functions
   1. Admin

+login(username, password): Boolean

+addclient(client):Client -storedata()

-updatedata()

+generateBill(Client): Bill

+sort\_data()

-searchPatient(Client):Client

+givediscount()

+bookAppointment(Test, Client)

* 1. Client

+register(name, password)

+login(patientID, password): Boolean

+bookAppointment(Test, PatientID)

+givetest() -PayBill(Bill)

+checktestresult()

* 1. Organization

+availdiscount(Bill)

+addClient(Client)

* 1. Test

+Test(testID)

+addReport()

+addBill(): Bill

* 1. Bill

+Bill(patientID, testID)

+addPayment()

+updatePayment()

+discount()

+PrintBill()

* 1. Report

+ Report(TestID, PatientID)

-updateData()

1. Identify relationships

1:1 Relationships

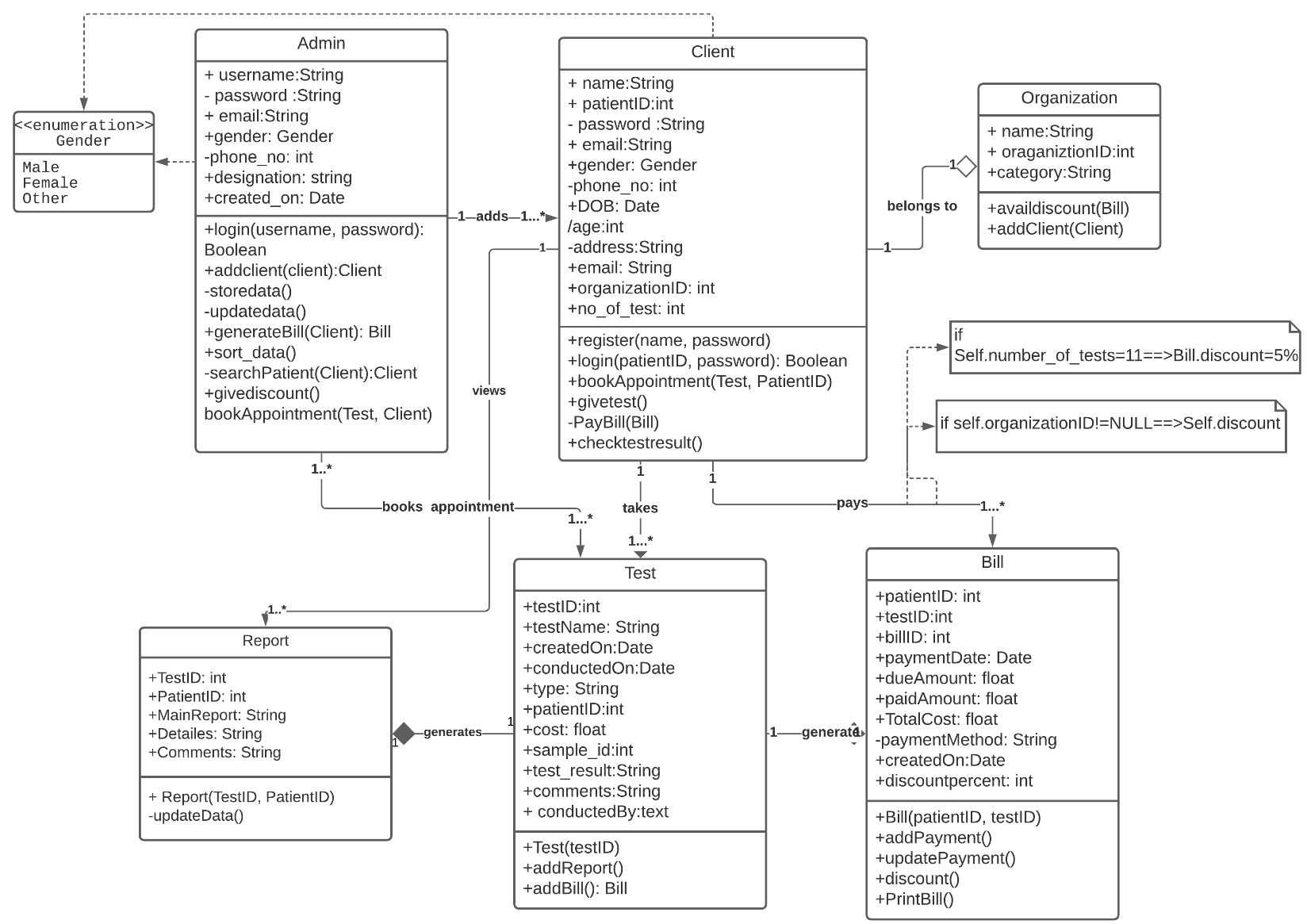
* 1. Admin adds Client
  2. Client belongs to a Organization
  3. Test gets a Bill
  4. Each test generate a report

1:n

1. Client takes one or more tests
2. Client has to pay one or more bills
3. Client can view his reports

n:n

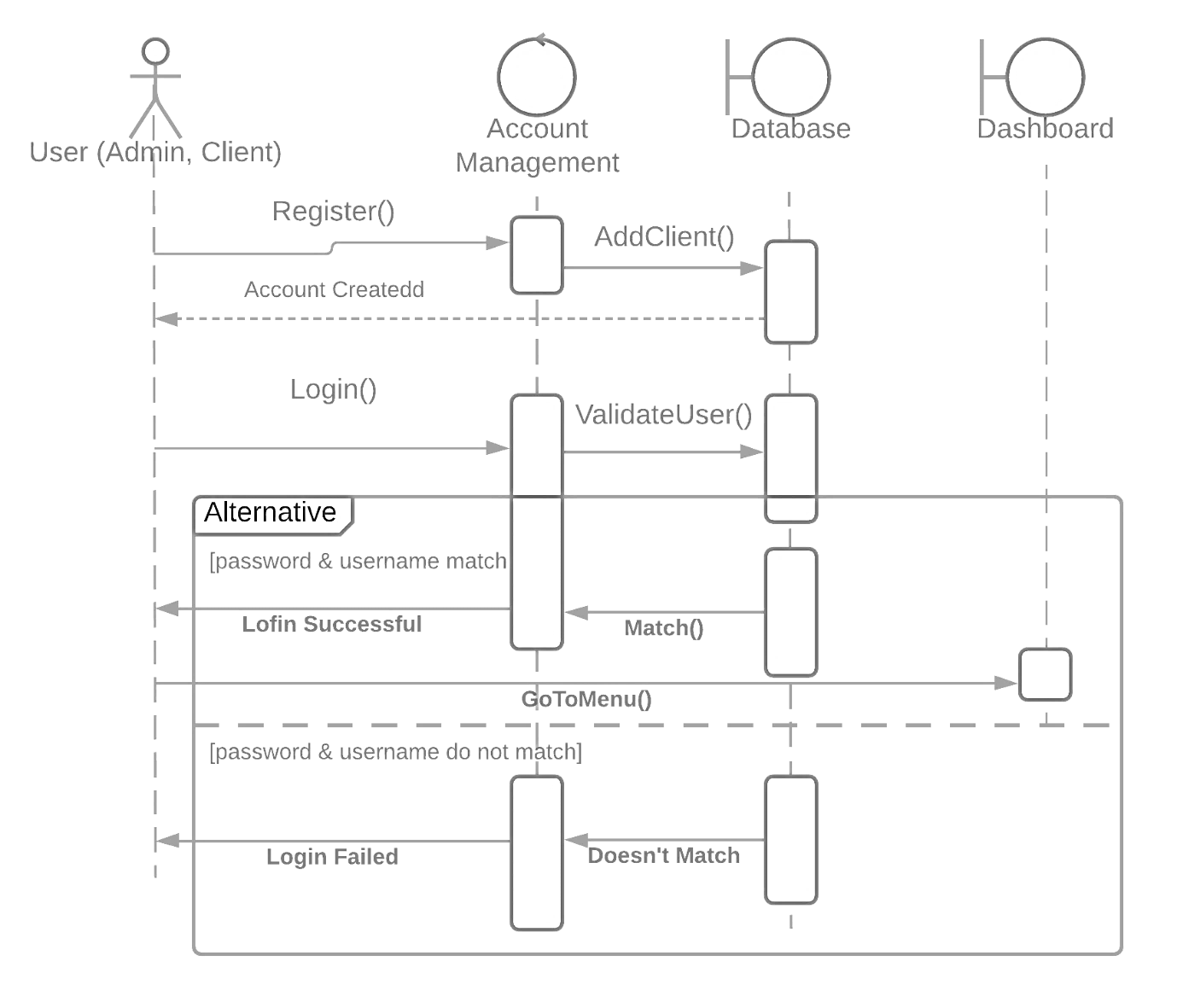
1. Admins book appointments
2. Constraints
   1. if no\_of\_tests ==11 client gets 5% discount
   2. Client belonging to an organizations gets discount accordingly

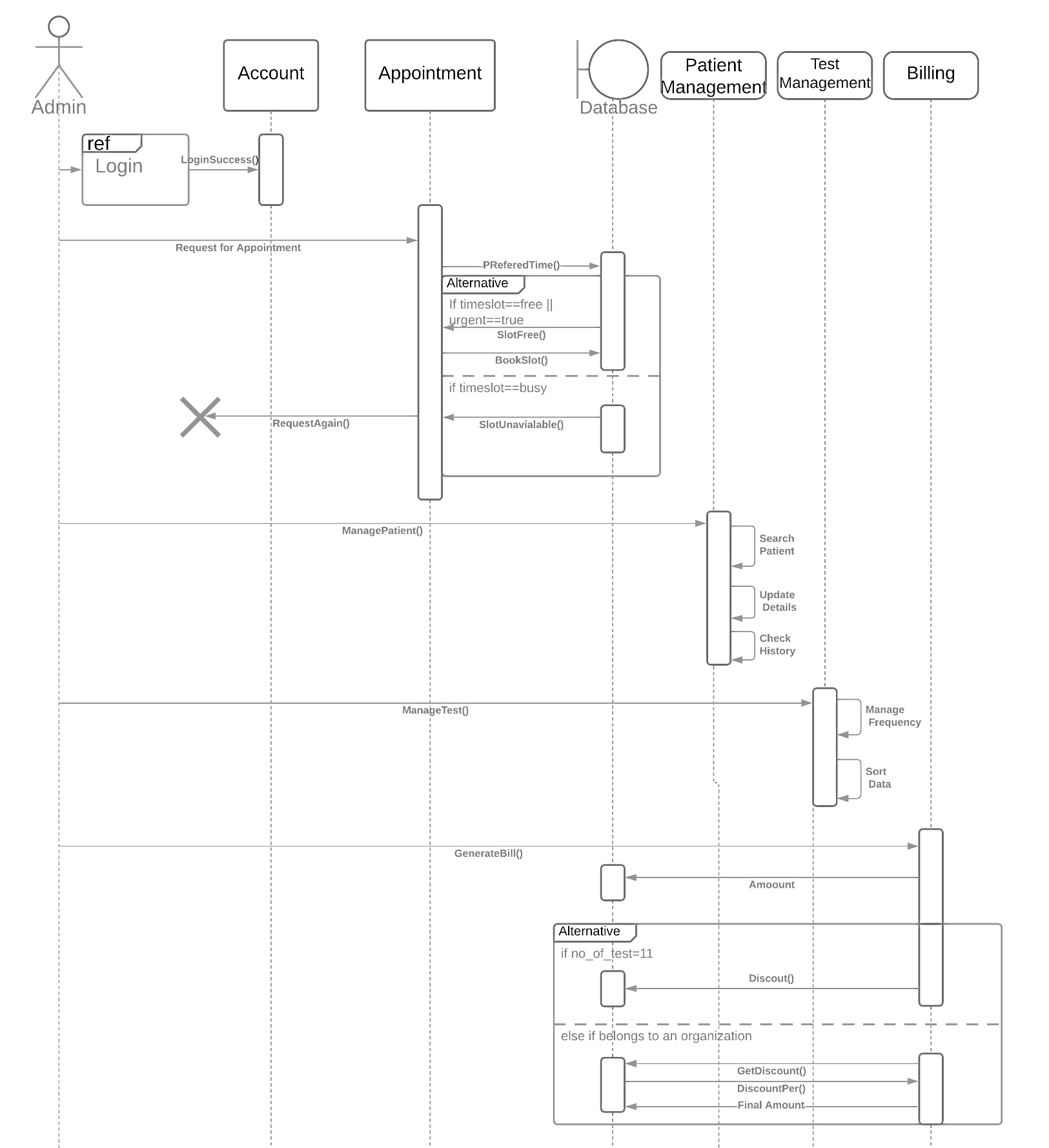
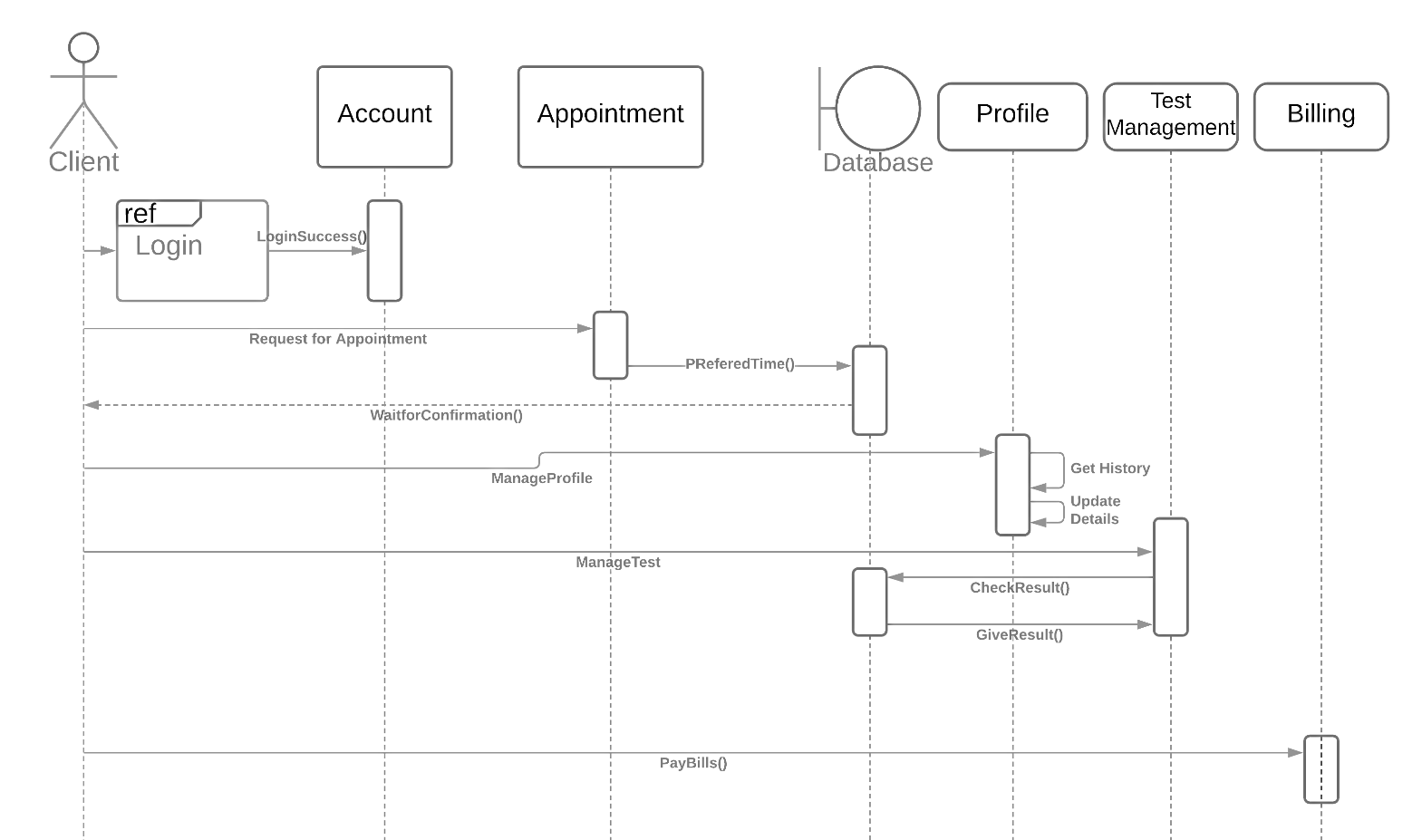
***Class Diagram: Client Diagnostic Centre ***

***Behavioural View***

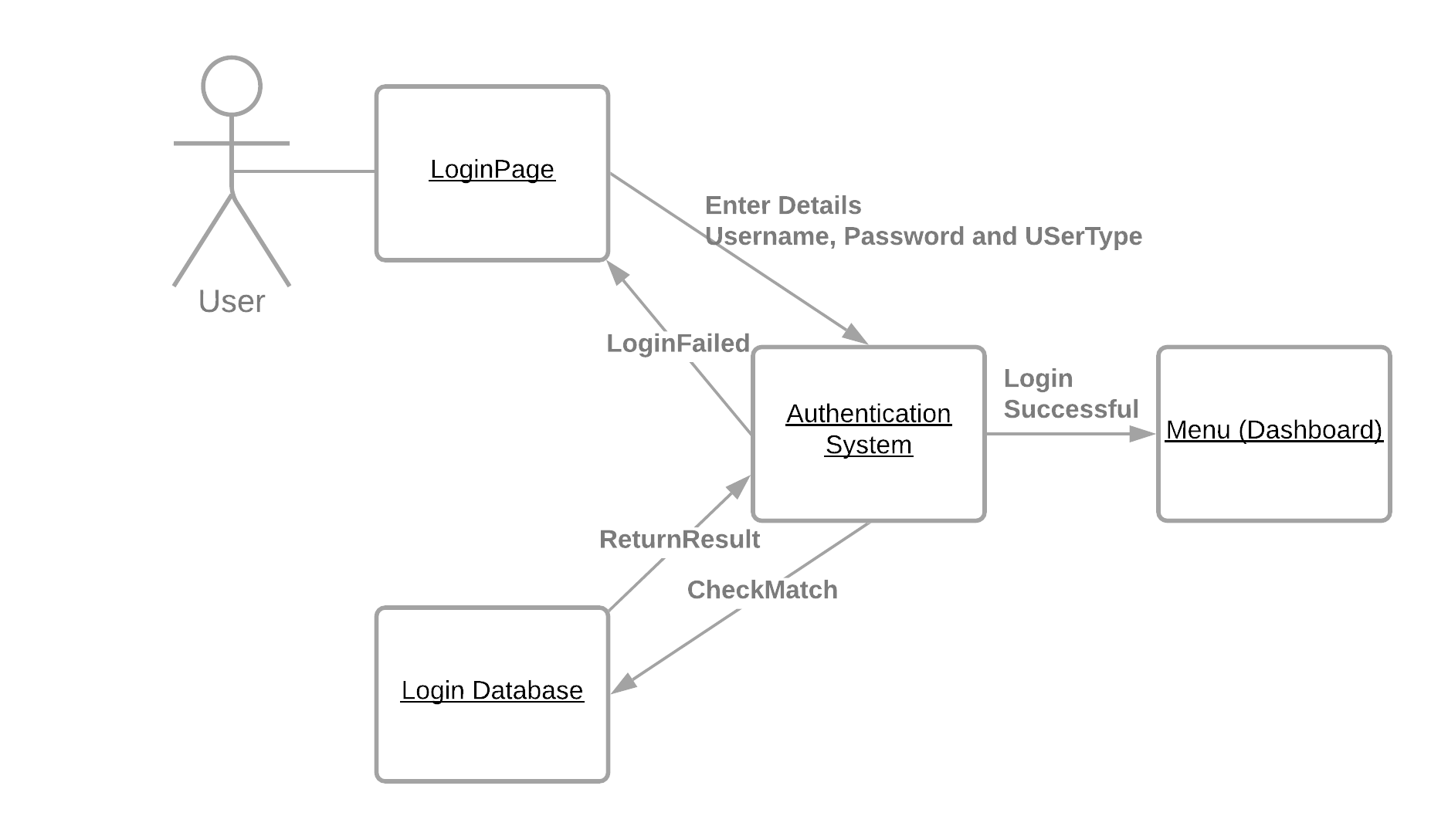
***Sequence Diagram***

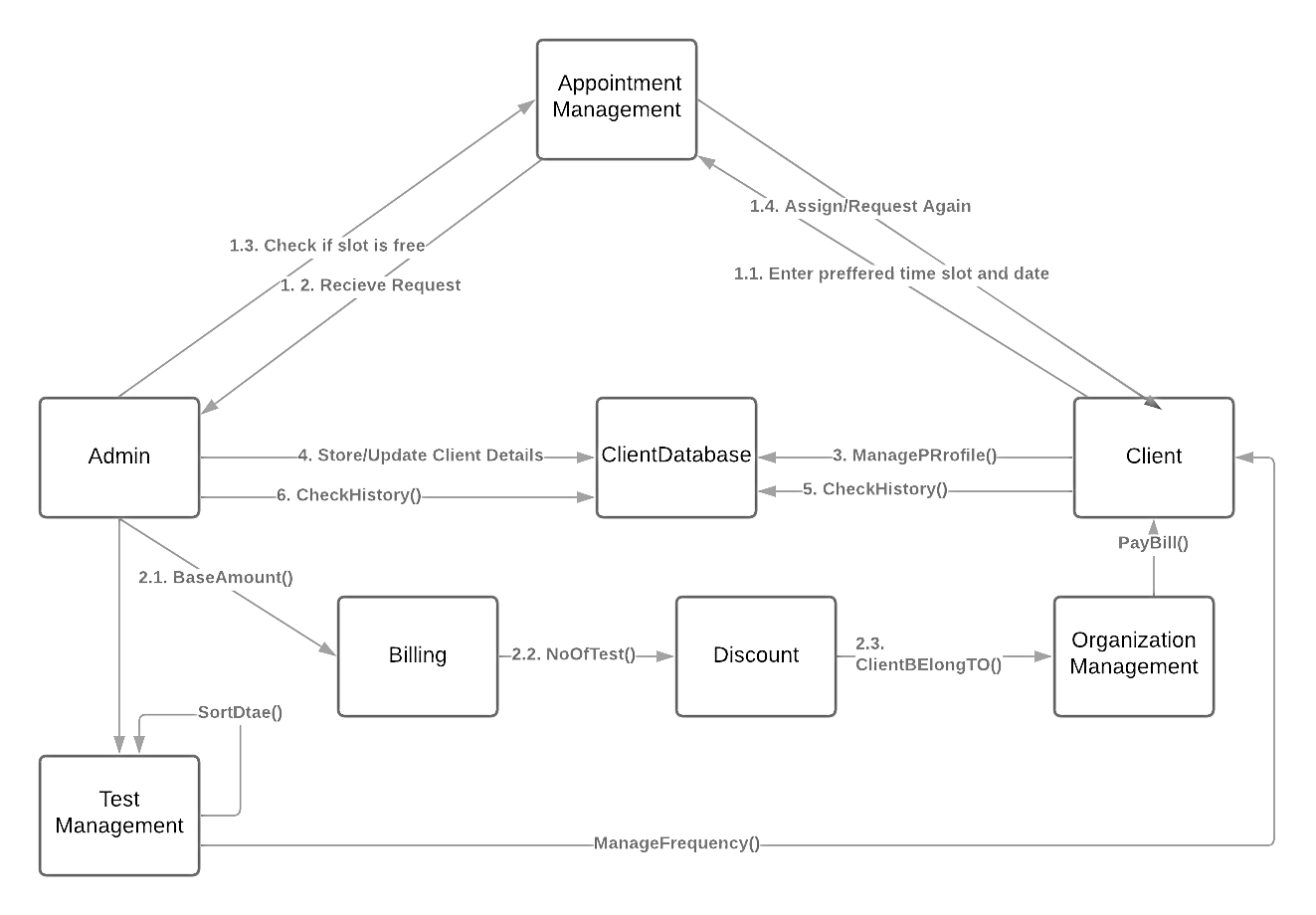
1. Login

******

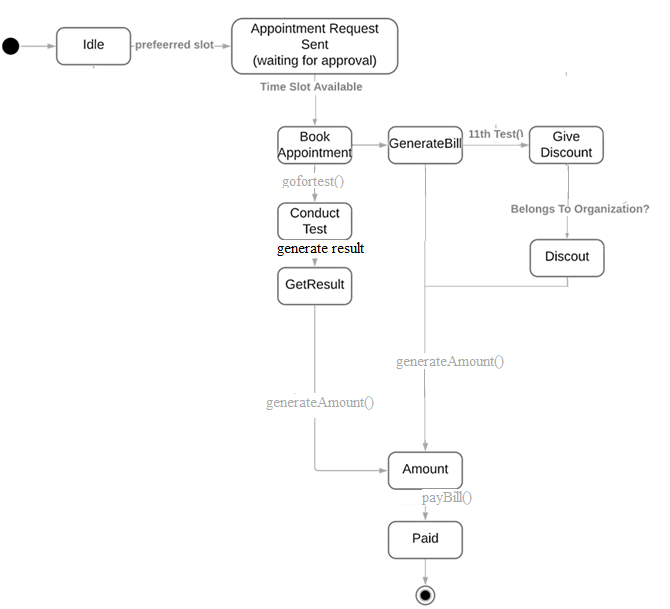
1. Admin***3.***Client

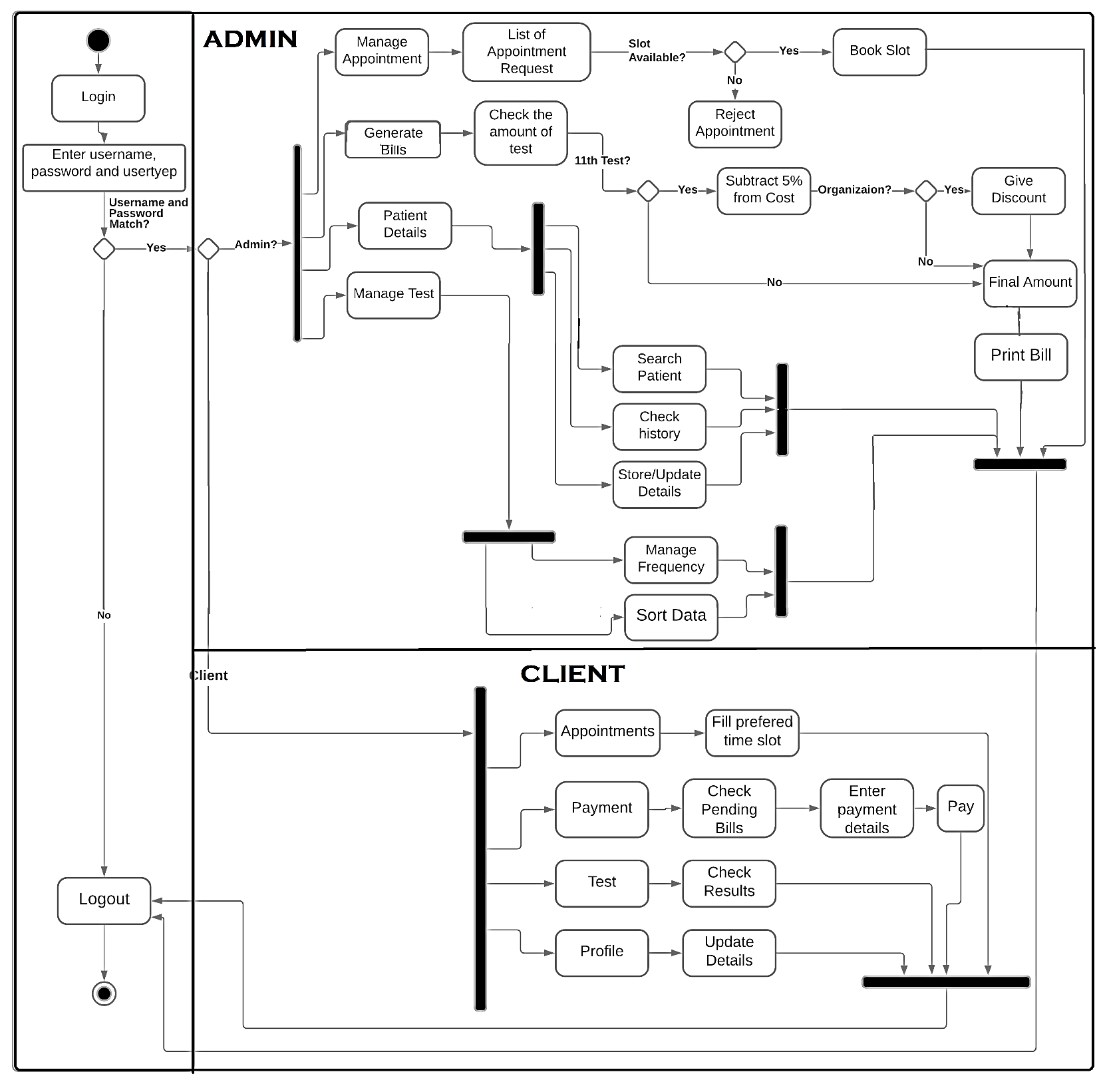
***Collaboration Diagram***

1. Login
2. Client Coordination

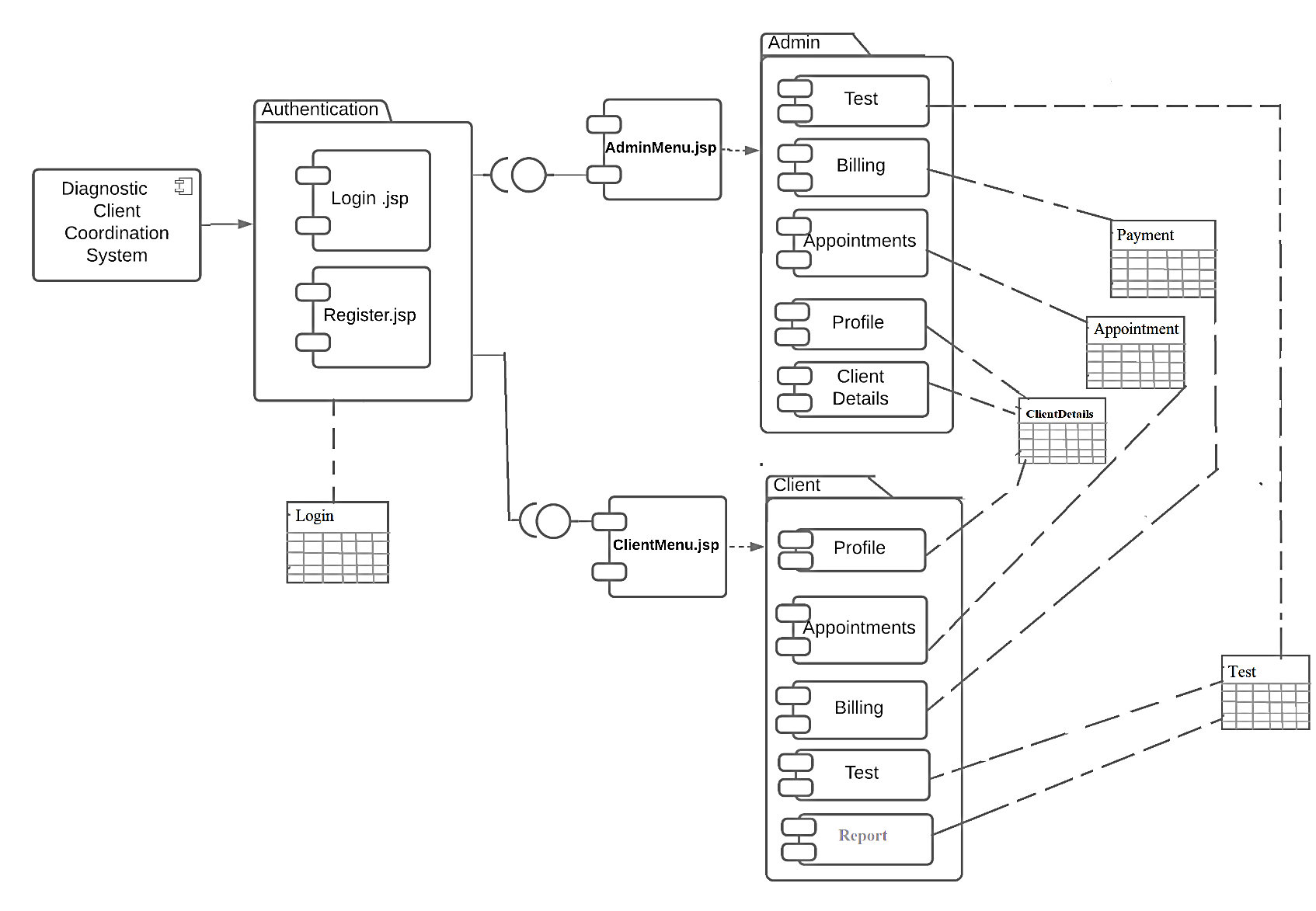
******

***State Chart Diagram:*** Diagnostic Client Coordination System (Appointment, Test, Bill)

******

***Activity Diagram: Diagnostic Client Co-ordiantion System***

***Component Diagram Diagnostic Client Co-ordiantion System***

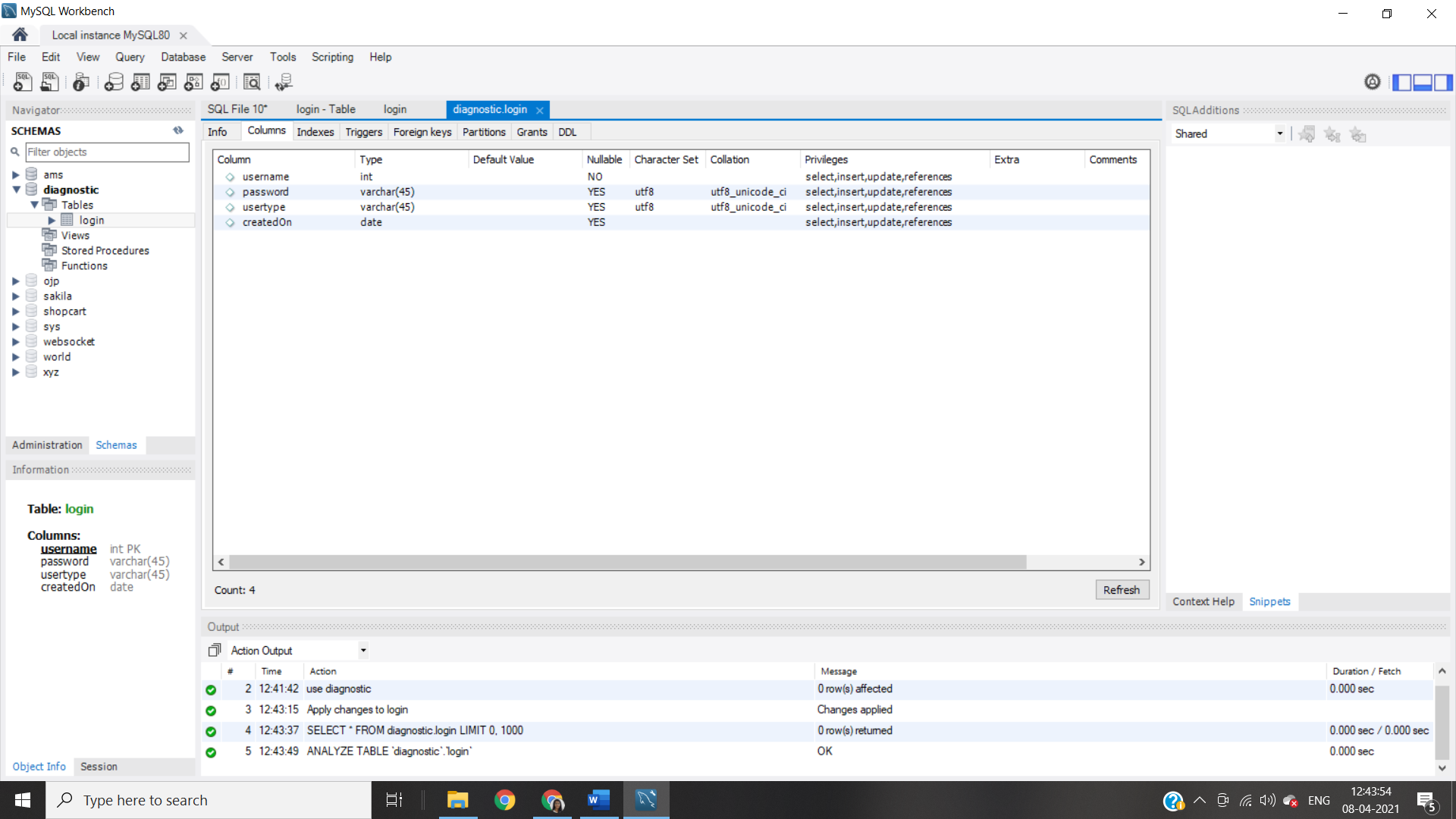
******

***Deployment Diagram: Diagnostic Client Co-ordiantion System*** 

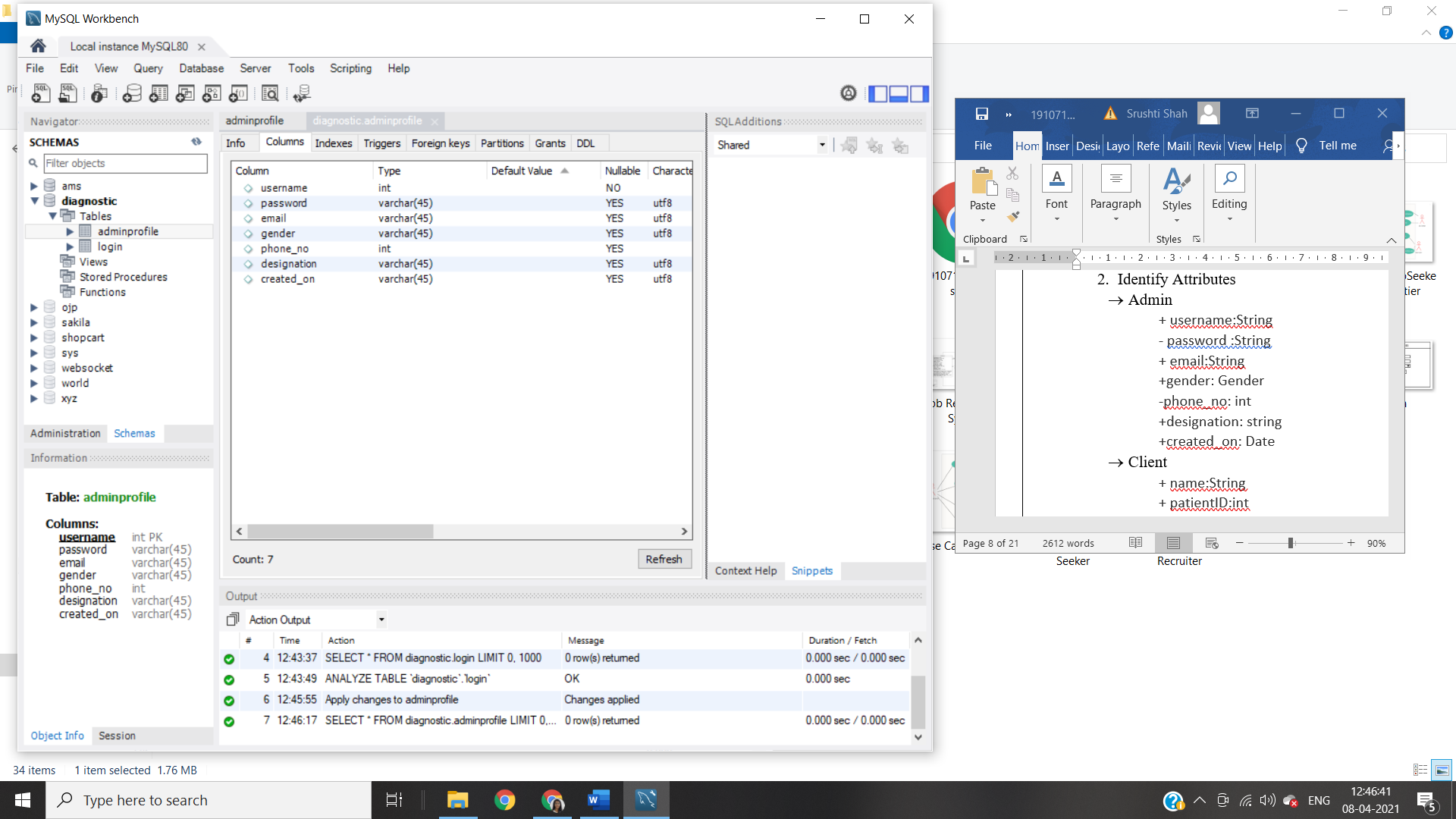
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Machine | **Software Configuration** | **Hardware Configuration** | **Operating System** | **Compiler** | **Any other software** | **Software Modules and path** |
| Client | Google Chrome  /Internet Explorer/  any browser with JSP Support |  | Windows 7 and after | Java | NetBeans, | GUI |
| Application Server | Sun GlassFish 4.1.1. Server | * Sun GlassFish Enterprise Server: 35 MB minimum * SDK: 250 MB minimum * 1GB Memory | Windows 7 and after, Red Hat, MacOS | HTML, Java, JSP,  SQL | Internet Explorer, Chrome, Mozilla, Safari |  |
| Database Server | MySQL Connector/J Driver 5.1 |  | Windwos 7 & after, Unbuntu |  |  | tables |

***Data Strcuture***

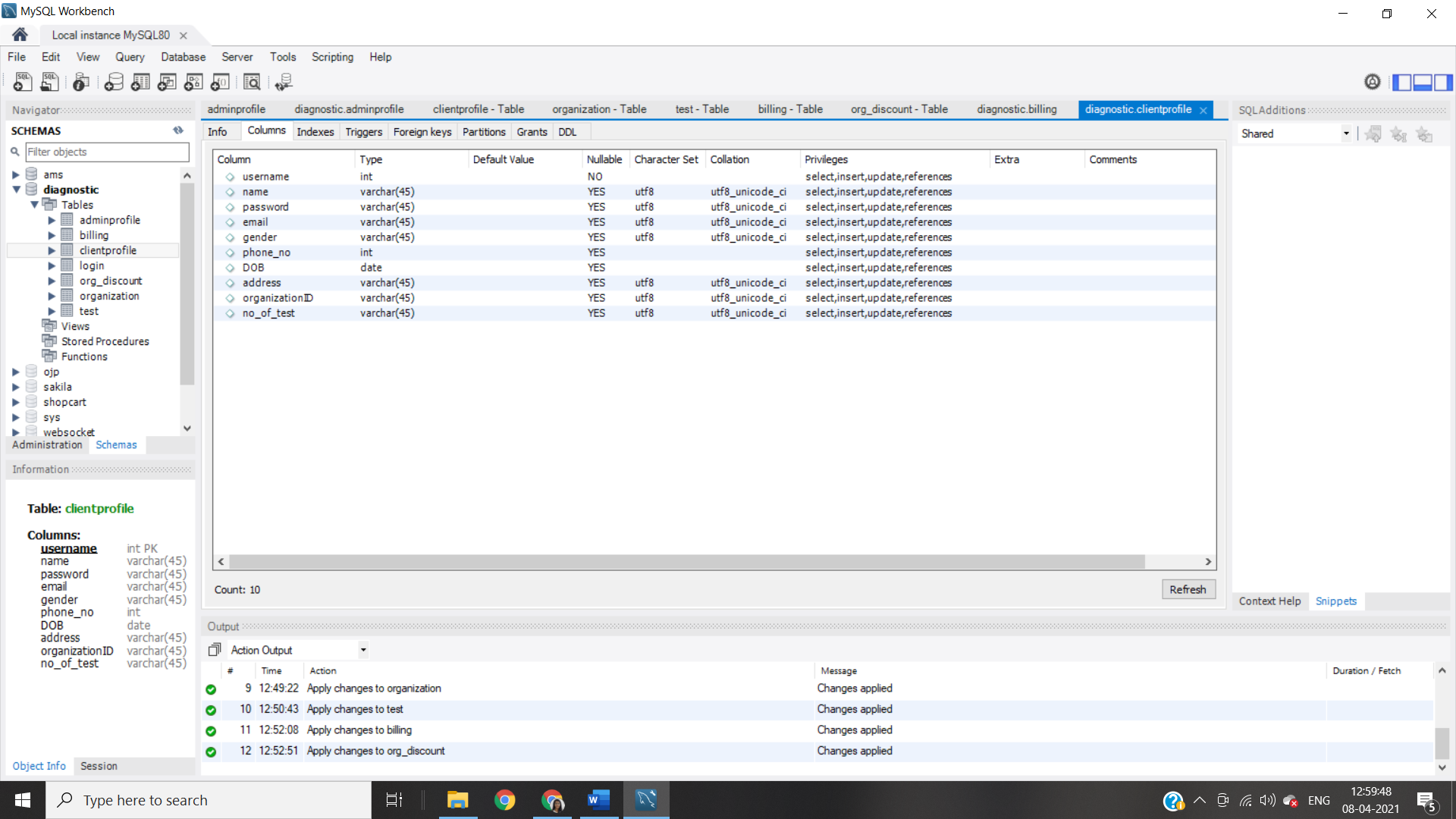
**Login**

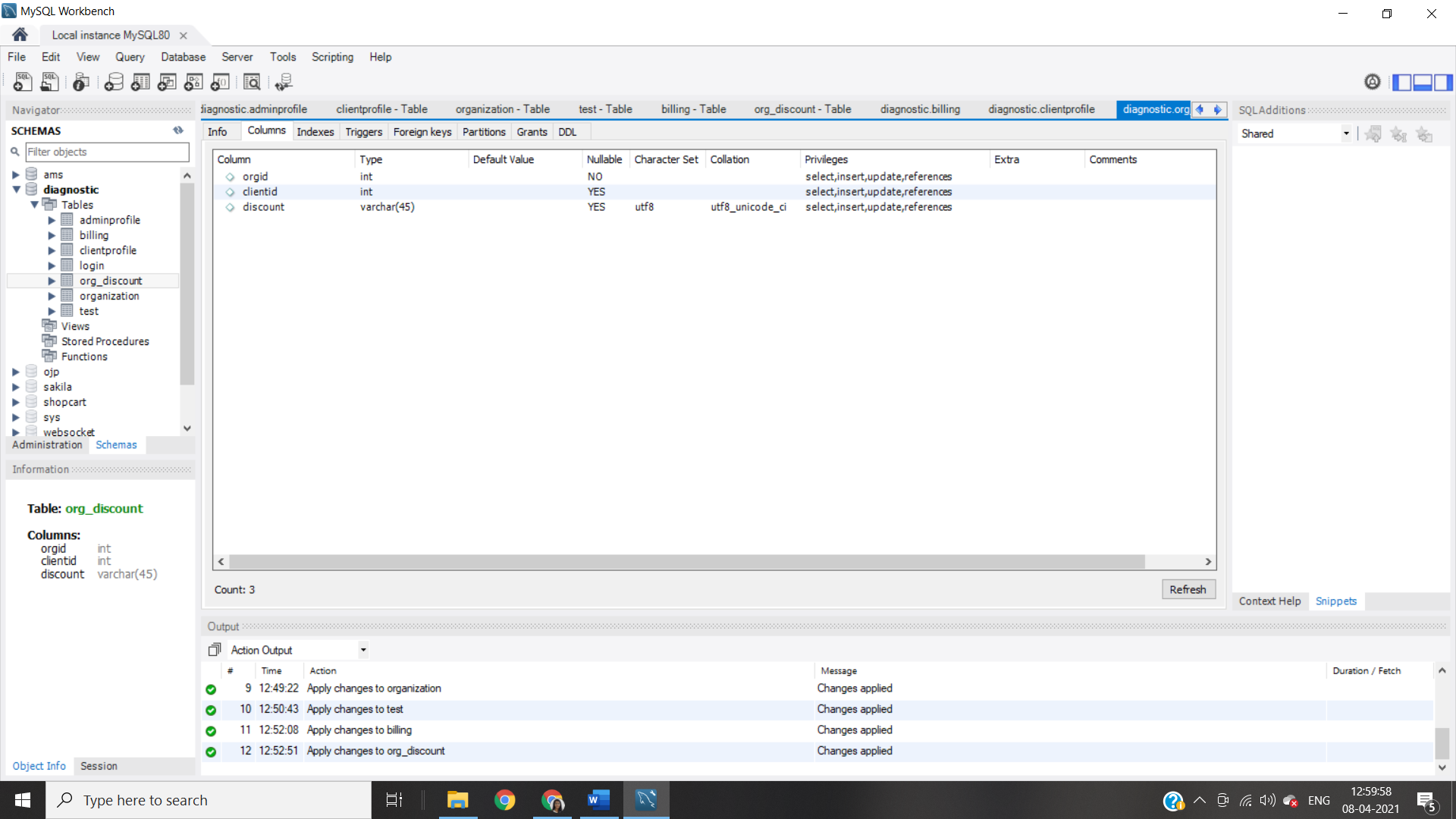
****

**Admin Profile**

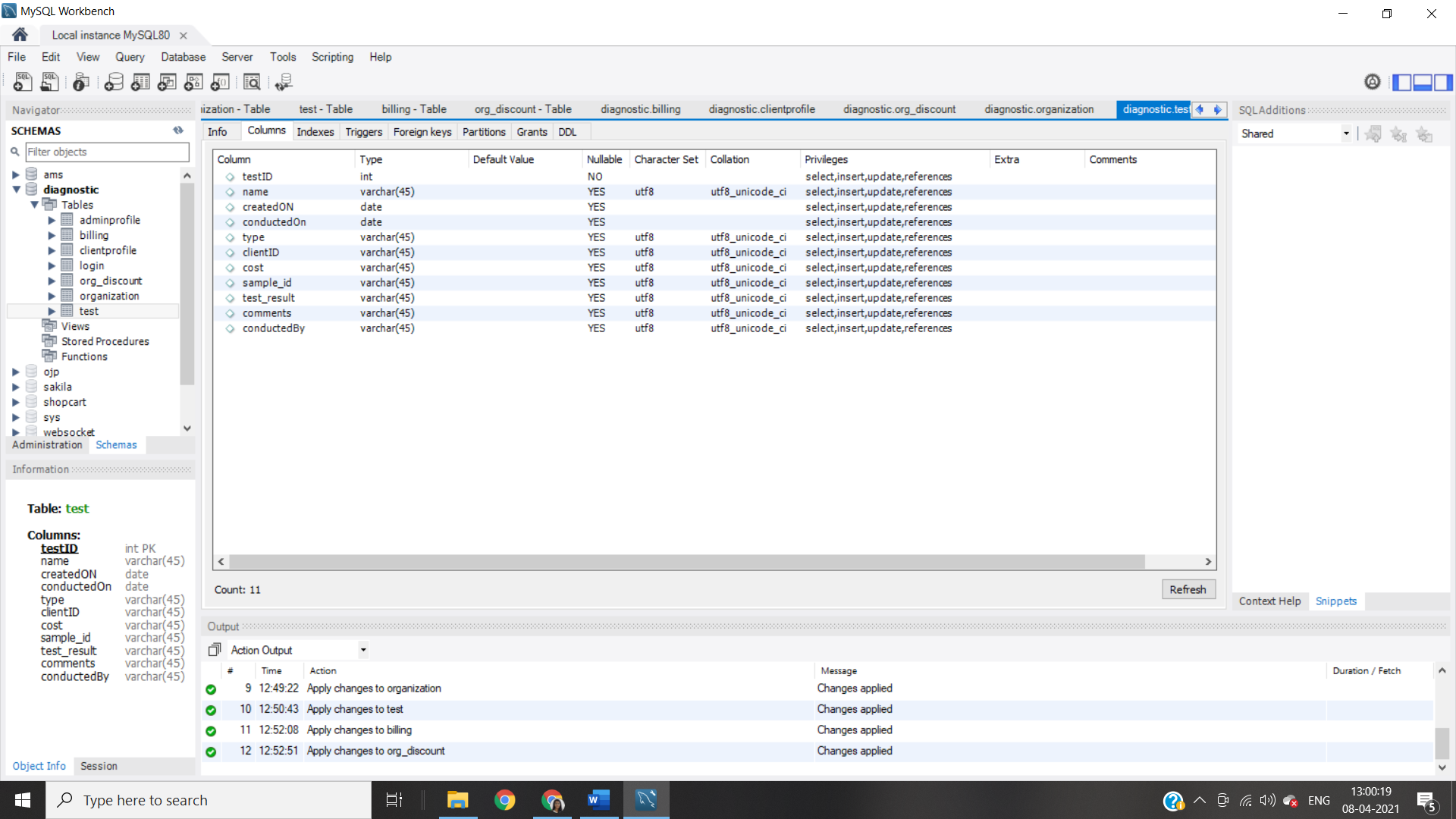
****

**ClientProfile**

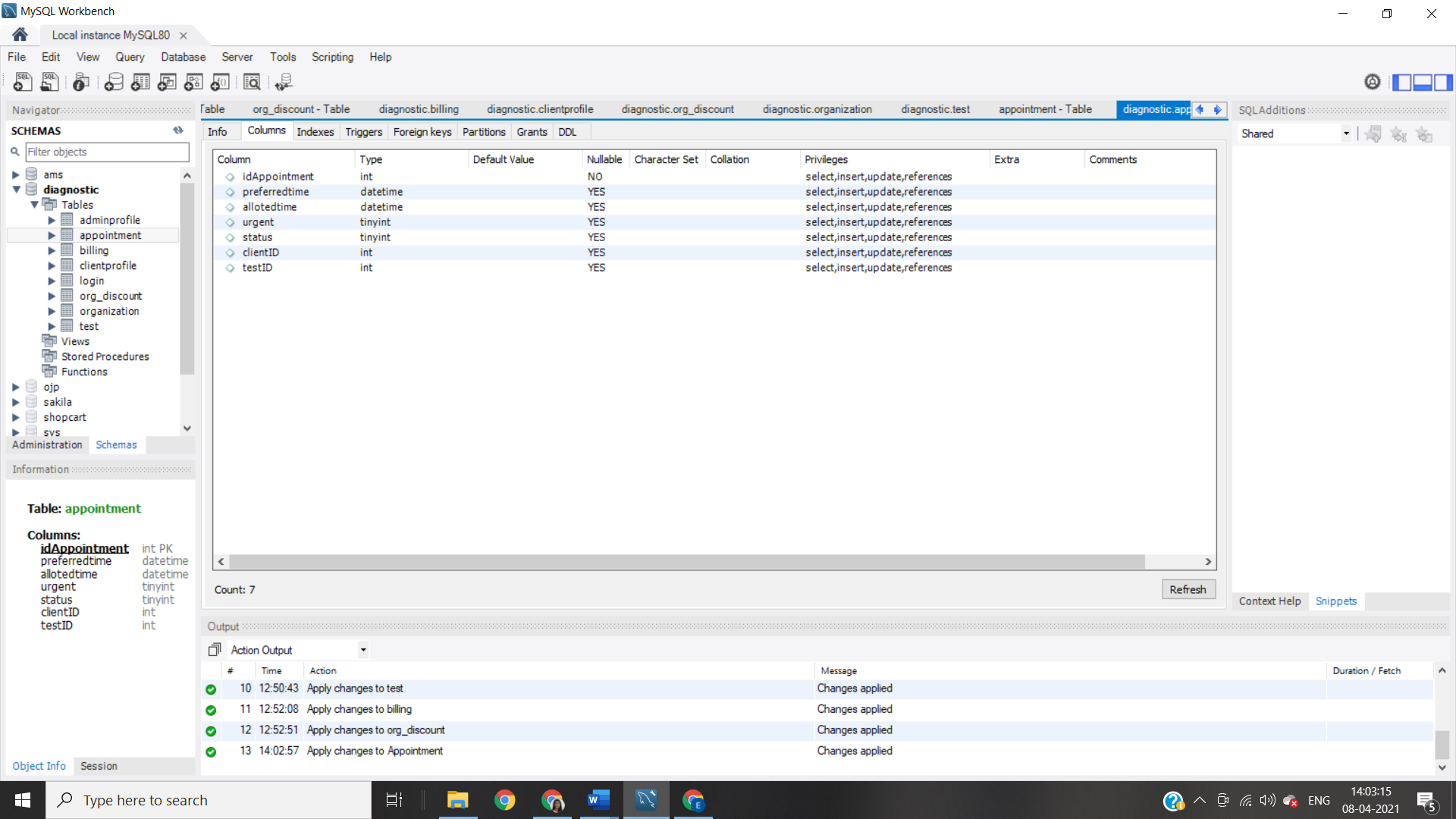
****

**OrganizationDiscount**

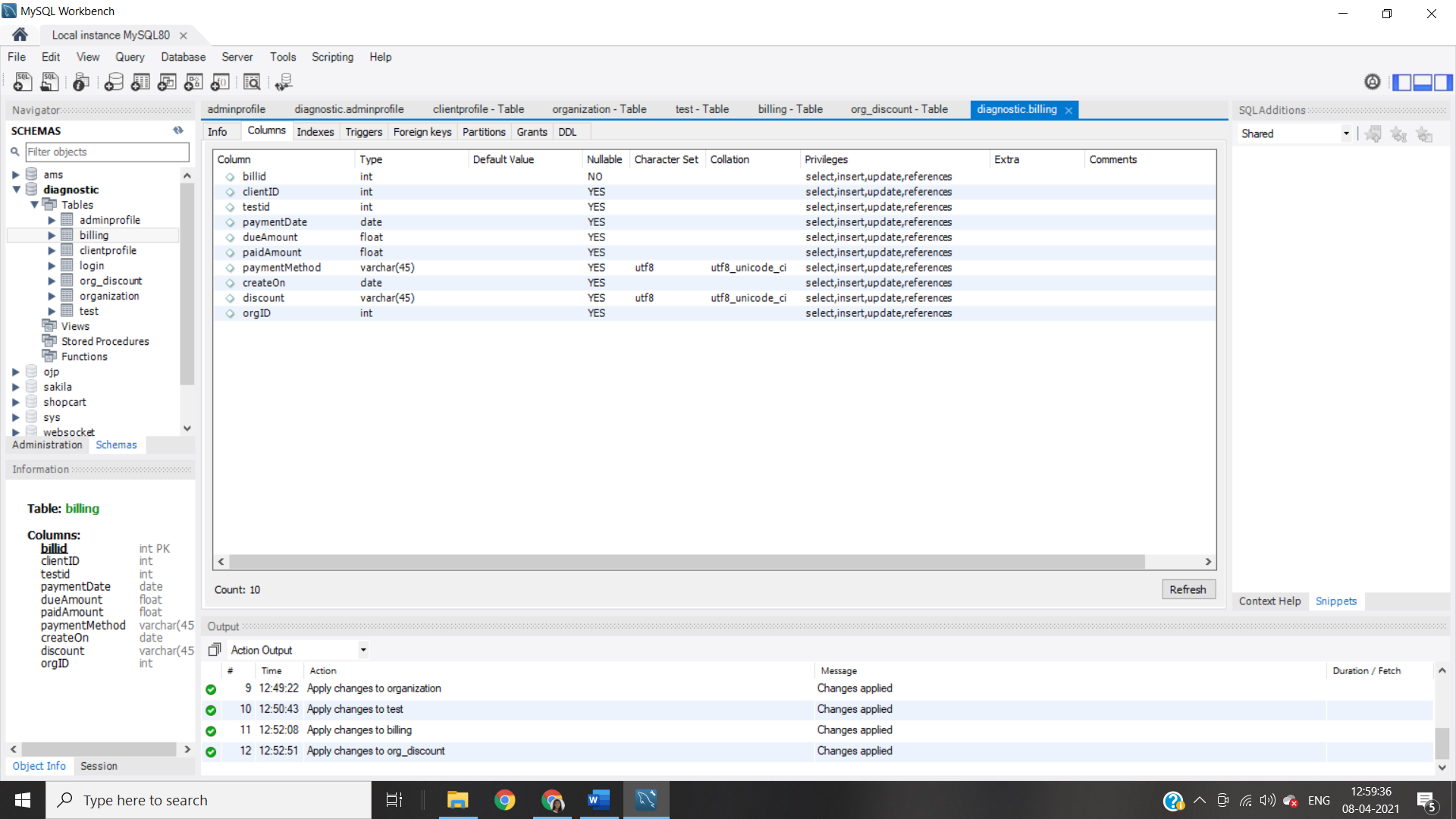
**Test**

****

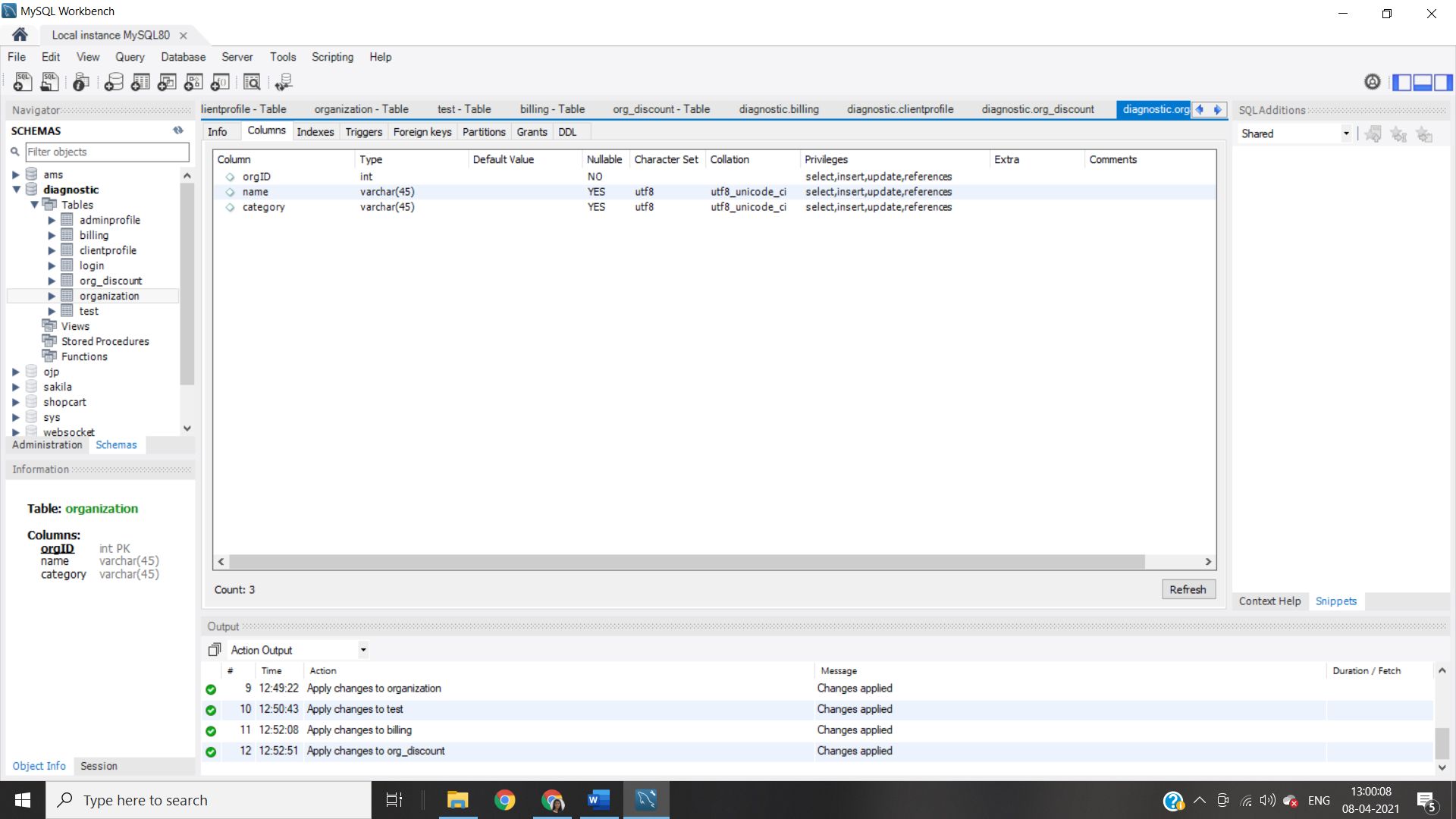
**Appointment**

****

**Blling**

****

**Organization**



***ALGORITHM DESIGN***

1. ***Register Client on System***

|  |  |
| --- | --- |
| Name | Register |
| Summary | A new user (a user which doesn’t have an account) should first register into the system to use it. This feature will allow the user to enroll into the system if the user is a new user and doesn’t already have an existing account. |
| Rationale | The user can access the system |
| Actor | Client |
| Pre-Condition | None |
| Basic course of Event | 1. User opens the desired module of the system. 2. The system displays GUI the registration. 3. User enters the valid details required to join the system 4. Submits the form |
| Post- Condition | User is registered |
| Alternate Flow | * + - 1. User goes back to the Home Page |

**Algorithm:**

**Input:** username, password, usertype;

**Output:**  user added to the system

**Pseudo Code:**

* + 1. Start
    2. Go to Registration Page
    3. Enter the username and usertype
    4. Enter password and confirm password
    5. If username!=NULL && usertype!=NULL &&password!=NULL && confirm\_password!=NULL then

If password==confirm\_password

If username already exists in table then enter another username

1. Else: Enter the entry into login table of database

Else

Alert: Password and confirm password do not match

Try again

1. Else:
   * + - 1. The field is required, do not leave it empty
2. ***Login***

|  |  |
| --- | --- |
| Name | Login |
| Summary | The user can use the features of the system |
| Rationale | After registration of the user, the valid user can login into the system |
| Actor | Admin, Client |
| Pre-Condition | The login name and password should match with the login name and password provided while registering.  If the username and or password do not match, the user cannot login successfully into the system. |
| Basic course of Event | 1. User opens the desired module of the system. 2. The system displays GUI the login form 3. The user writes its username, password and type 4. If valid, user can login into the system. If the username and or password do not match, the user cannot login successfully into the system. |
| Post- Condition | User goes to his/her dashboard |
| Alternate Flow | 1. The username and password are blank 2. The type of user is not specified 3. The username and password do not match |

**Algorithm:**

**Input:** username, password, usertype;

**Output:**  allow/denied access to home page

**Pseudo Code:**

1. Start
2. Login Page open
3. Enter the username, password, type input
4. If password==NULL | username==NULL|| type==NULL
   1. The field is required
5. Retrieve the entry from database for the given username
   1. If username=username(database) && password=password(database)&& type=usertype
   2. Go to Dashboard
6. Else
   1. Alert: Invalid entry, please try again
7. End
8. ***Book Test Appointment***

|  |  |
| --- | --- |
| Name | Book Test Appointment |
| Summary | The patient requests to book test and admin assigns a slot |
| Rationale | The appointment is booked |
| Actor | Client and Admin |
| Pre-Condition | 1. The client is a valid user 2. The slot he asks for is free |
| Basic course of Event | 1. The client (patient) logs into the system 2. The patient requests for an appointment using test name, preferrable date and time 3. The admin checks if the slot is available 4. The admin assigns the slot if free to the patient (client) |
| Post- Condition | The client is given the free slot for test |
| Alternate Flow | * + 1. The admin denies the appointment as he finds the user malicious |
| Alternate Flow 2 | * + - 1. The slot is not available   1. The admin sends a message to request for another slot   2. The admin assigns a nearby time slot |
| Exceptions | 1. The appointment is marked as urgent   The admin adjusts the time slot given to someone else   1. System fails to given appointment   Patient has to request again |

**Algorithm:**

1. ClientRequestAppointment()
   1. The client goes to the Appointment Section
   2. Choose the test
   3. Choose preferred time and fate
   4. Wait for confirmation from Admin
2. BookAppointment()
   1. Check requests from Client
   2. If the time slot==free
      1. Confirm the appointment for the particular test
      2. Enter in test database
   3. Else if urgent==true
      1. Give the time slot and adjust the previous appoinment
   4. Else
      1. The admin asks client for another slot
3. ***Generate Bill***

|  |  |
| --- | --- |
| Name | Generate Bill |
| Summary | The client is has to pay the bill for the corresponding test/s |
| Rationale | The amount to be payed is generated |
| Actor | Client and Admin |
| Pre-Condition | The appointment is booked |
| Basic course of Event | 1. The appointment for a particular test is booked 2. The system generated the Amount of the test to be paid by client as stored in its database |
| Post- Condition | The bill is printed and given to the client |
| Alternate Flow | * + - 1. If number\_of\_test of patient is 11          1. Provide 5% discount          2. Calculate the final amount |
| Alternate Flow 2 | * + - 1. The client belongs to a organization who has a contract with the system       2. Provide corresponding discount to the client       3. Calculate Total |

**Algorithm:**

**Input:** testID, patientID

**Output:**  printable bill

**Pseudo Code:**

1. Start
2. Go to Billing section of Admin
3. Enter the testID and patientID
4. If testID==NULL || patientID==NULL
   * 1. Please enter valid data
     2. Go to step 3
5. Generate the Base Amount
6. If no\_of\_test=11: then Amount:=Base Amount-5%of Base Amount
7. If client belongs to a organization
   1. Adjust amount accordingly
8. PrintBill()
9. Return Amount
10. ***Generate Test Report***

|  |  |
| --- | --- |
| Name | Generate Test Report |
| Summary | After giving the test, the client can check the report of the test |
| Actor | Client |
| Pre-Condition | The test is conducted |
| Basic course of Event | 1. Go to the Test section 2. Check the report of the test you want |
| Post- Condition | The list of report is given. |

***Algorithm***

1. Start
2. Go to the test section of Job Seeker
3. Check the result in the Report column
4. End
5. ***Search Patient History***

|  |  |
| --- | --- |
| Name | Search Patient History |
| Summary | The administrator can check the search and check the record of patient |
| Actor | Admin |
| Pre-Condition | The list of record is displayed |
| Basic course of Event | 1. The admin logs into the system 2. The admin goes to the search tab 3. Enters the name of patient/patient ID 4. The details and history of the patient are displayed |
| Post- Condition | The patient history is displayed |
| Alternate Flow | 1. There is no patient with the given patient name/ID 2. No results displayed |

**Algorithm:**

**Input:** patientID, name, testTaken

**Output:**  details of patient

**Pseudo Code:**

1. Start
2. Go to the Patient Tab of Admin
3. Enter the name of client or the patientID
4. If client name or patientID is not found
   1. Print invalid ID, try again
5. Display the details of the patient
6. End
7. ***Store/Update Patient Details***

|  |  |
| --- | --- |
| Name | Store/Update Patient History |
| Summary | The administrator can store or update the record of patient |
| Actor | Admin |
| Pre-Condition | The patient/client is added to the system |
| Basic course of Event | 1. The admin logs into the system 2. The admin searches for the patient using patient ID/name 3. The patient is selected 4. The option to add/update the details is given 5. The details are entered/edited 6. The data is stored/updated into the database |
| Post- Condition | The database is updated |
| Alternate Flow | 1. There is no patient with the given patient name/ID 2. No results dispalyed 3. A new patient is added |

**Algorithm:**

**Input:** patientID, name,

**Output:**  details of patient

**Pseudo Code:**

1. Start
2. Go to the Patient Tab of Admin
3. Enter the name of client or the patientID
4. If client name or patientID is not found
   1. Print invalid ID, try again
5. Display the details of the patient
6. Click on Edit
   1. Update the fields you want
   2. Apply the changes to update the database
7. End
8. ***Sort the tests and manage frequency***

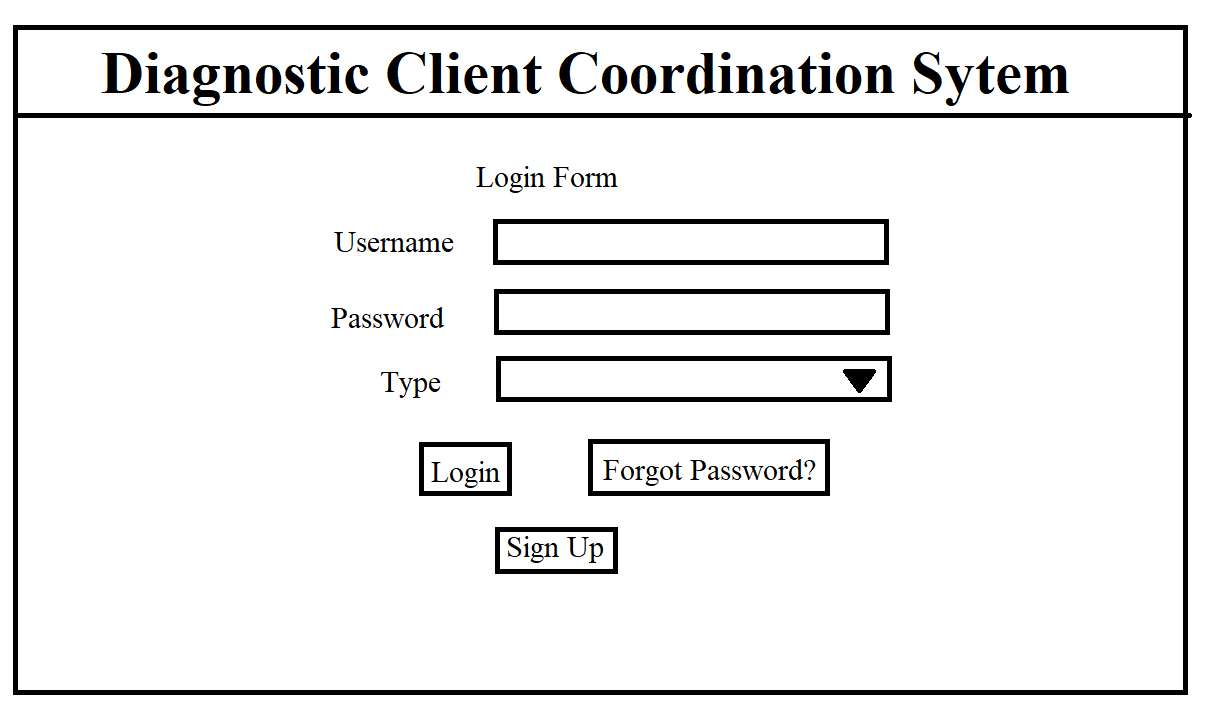
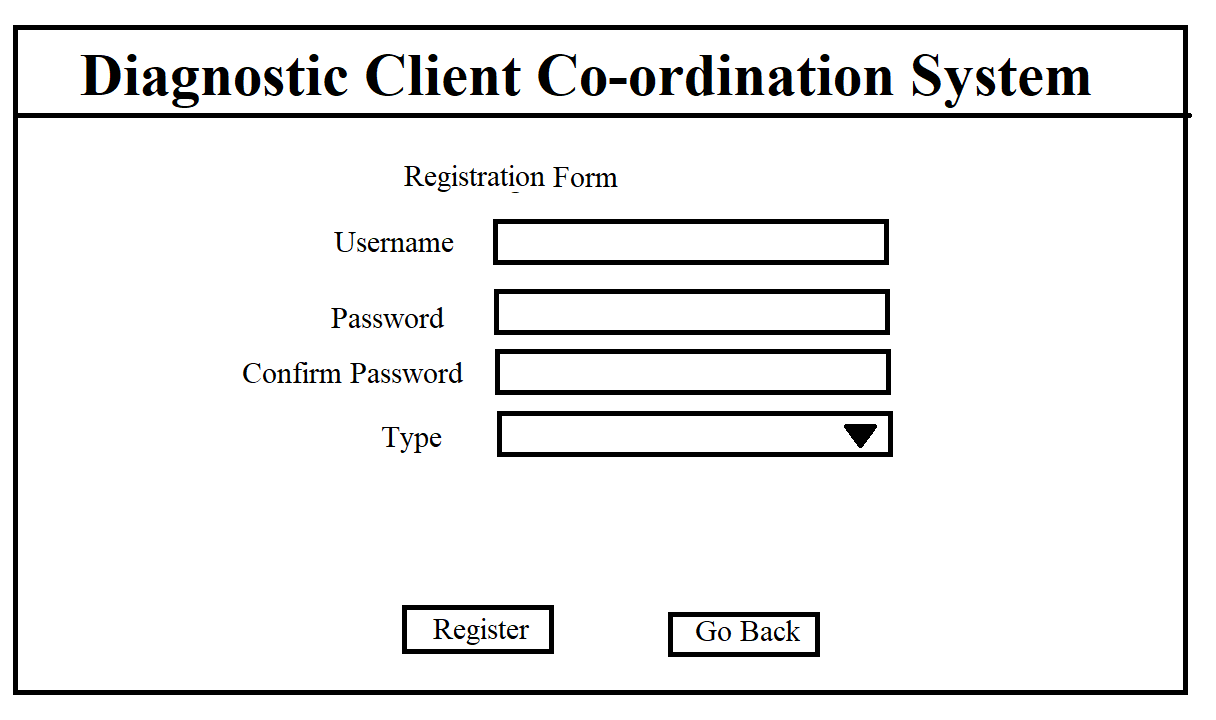
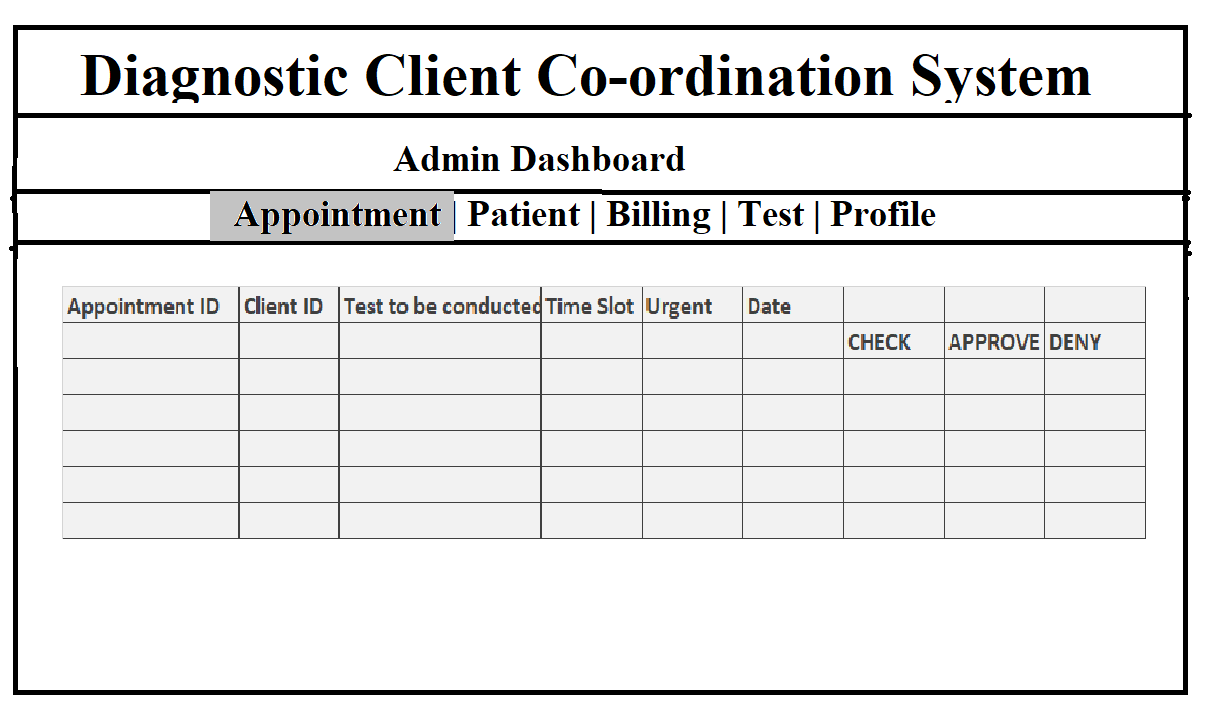
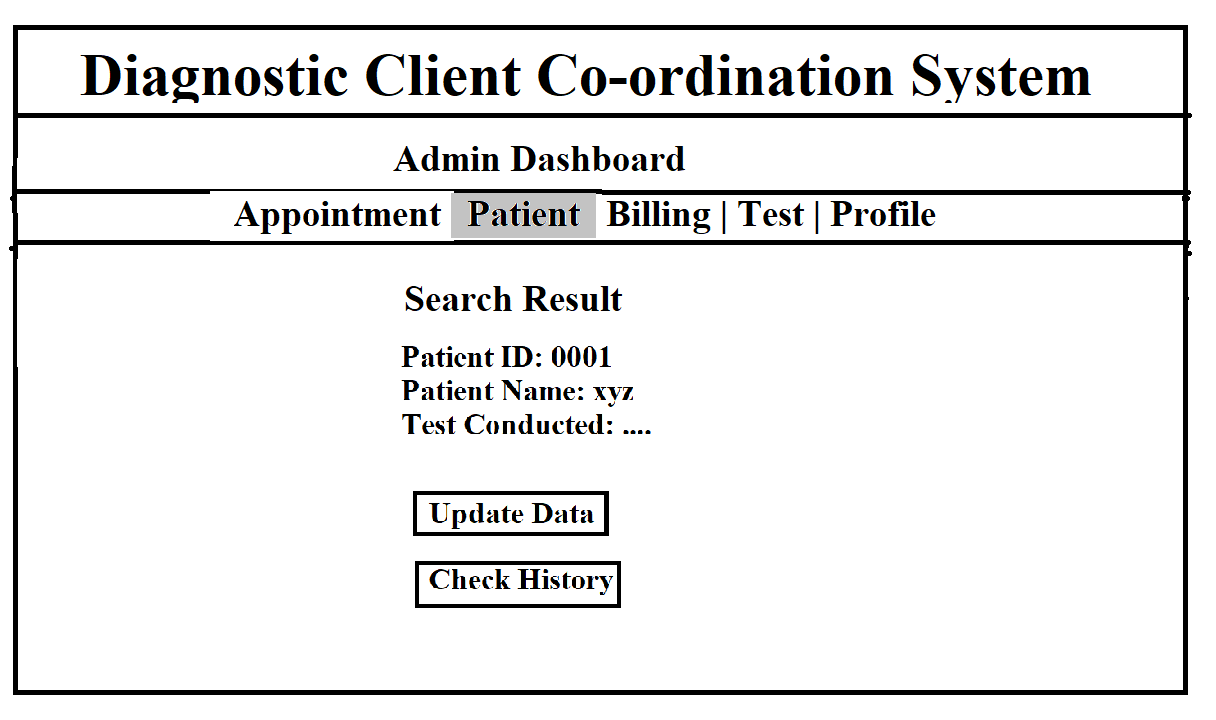
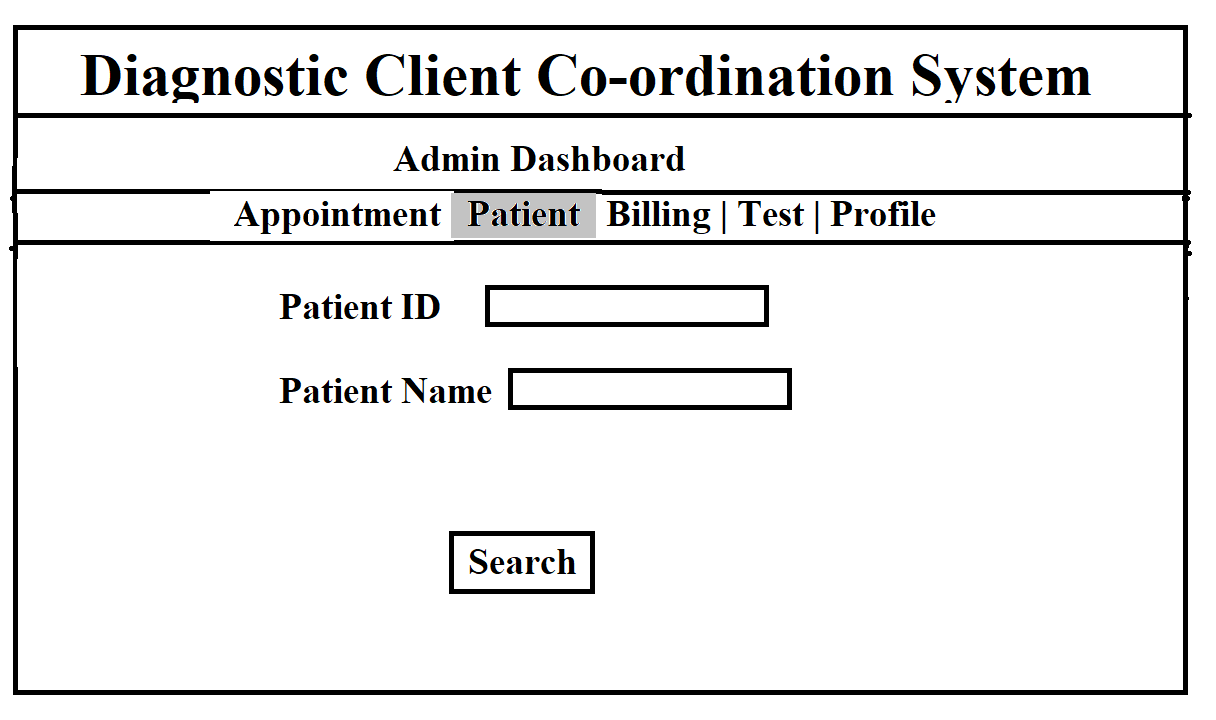
|  |  |
| --- | --- |
| Name | Manage frequency |
| Summary | The client when completed a test it is added to his list |
| Actor | Client |
| Pre-Condition | The client is registered to the system |
| Basic course of Event | 1. The test appointment is given to client 2. Client finishes the test 3. Update the no\_of\_tests in the database |
| Post- Condition | The database is updated |

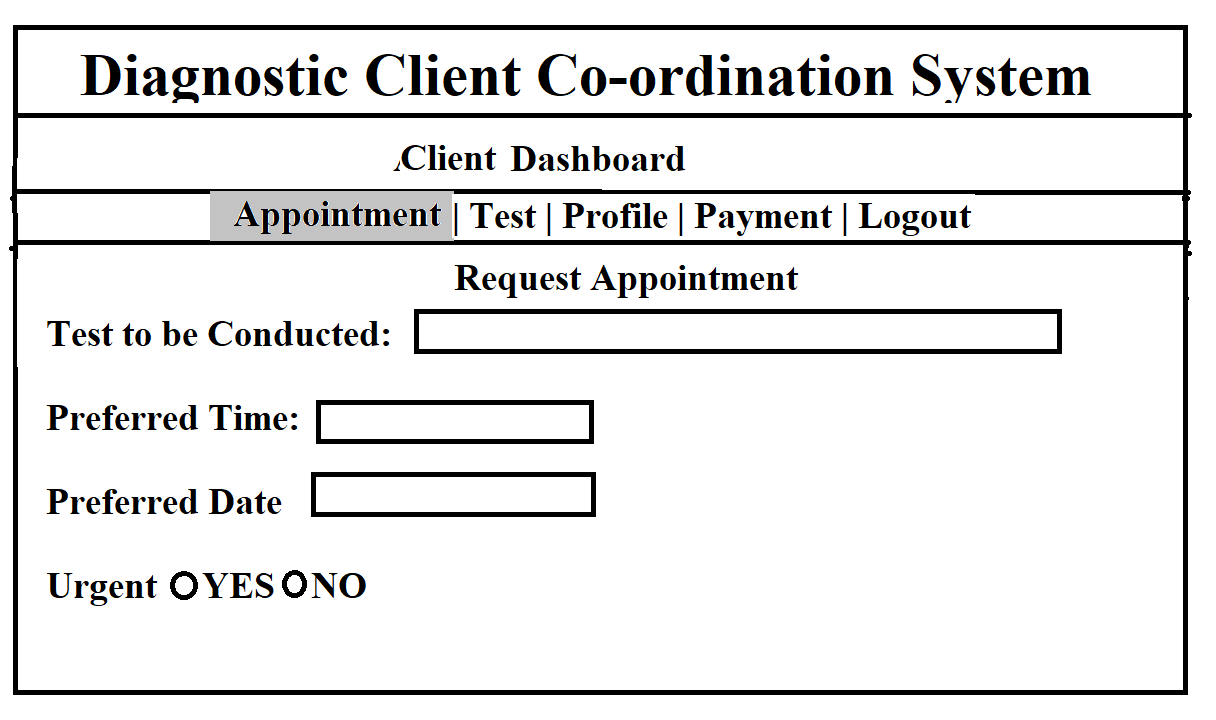
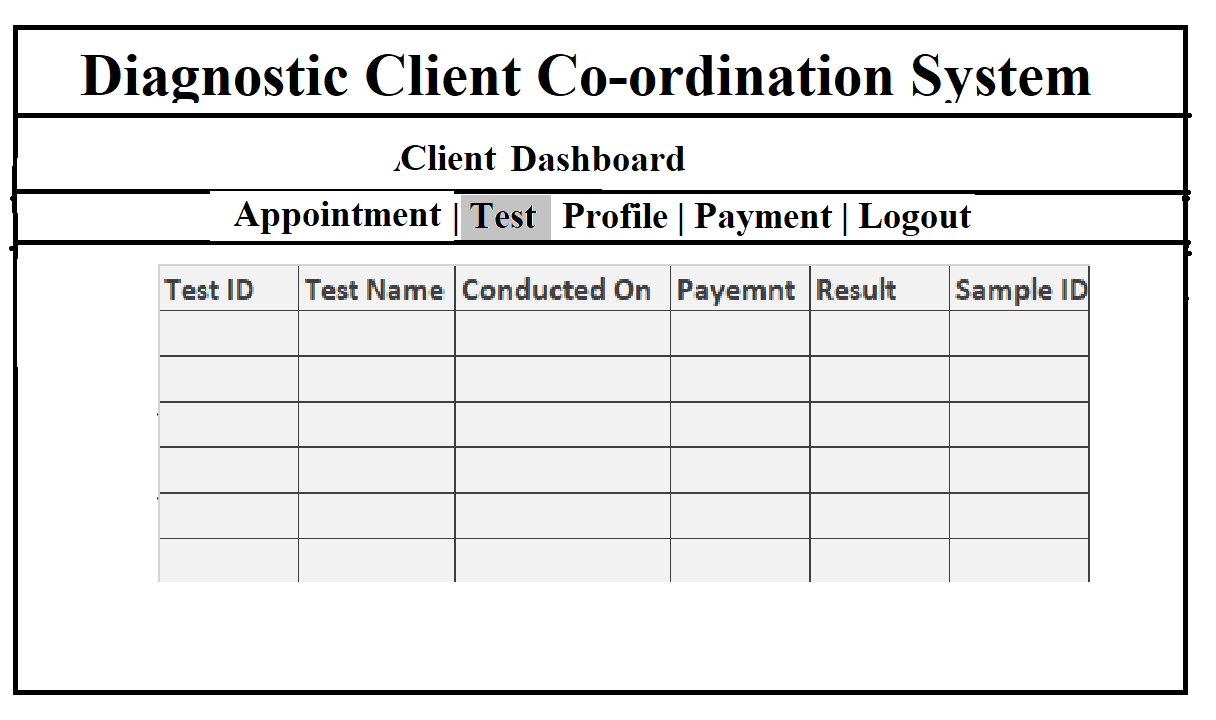
**Algorithm:**

**Pseudo Code:**

1. Start
2. Go to the Test tab
3. Once, the test is done update the no\_of\_test=no\_of\_tests+1
4. Sort the test and view details
5. End

***GUI Design***

1. Login
2. Regsiter
3. Admin Dashboard
   1. Book Apointment
   2. Search/Update/Store/Check history
4. Client Dashboard
   1. Request Appointment

b. Test Details. c. Payment