**COSC 5370 ADVANCED SOFTWARE ENGINEERING**

**TERM PROJECT**

CHOCOHOLICS ANONYMOUS

Instructor: Dr. Longzhuang Li

Team Members:

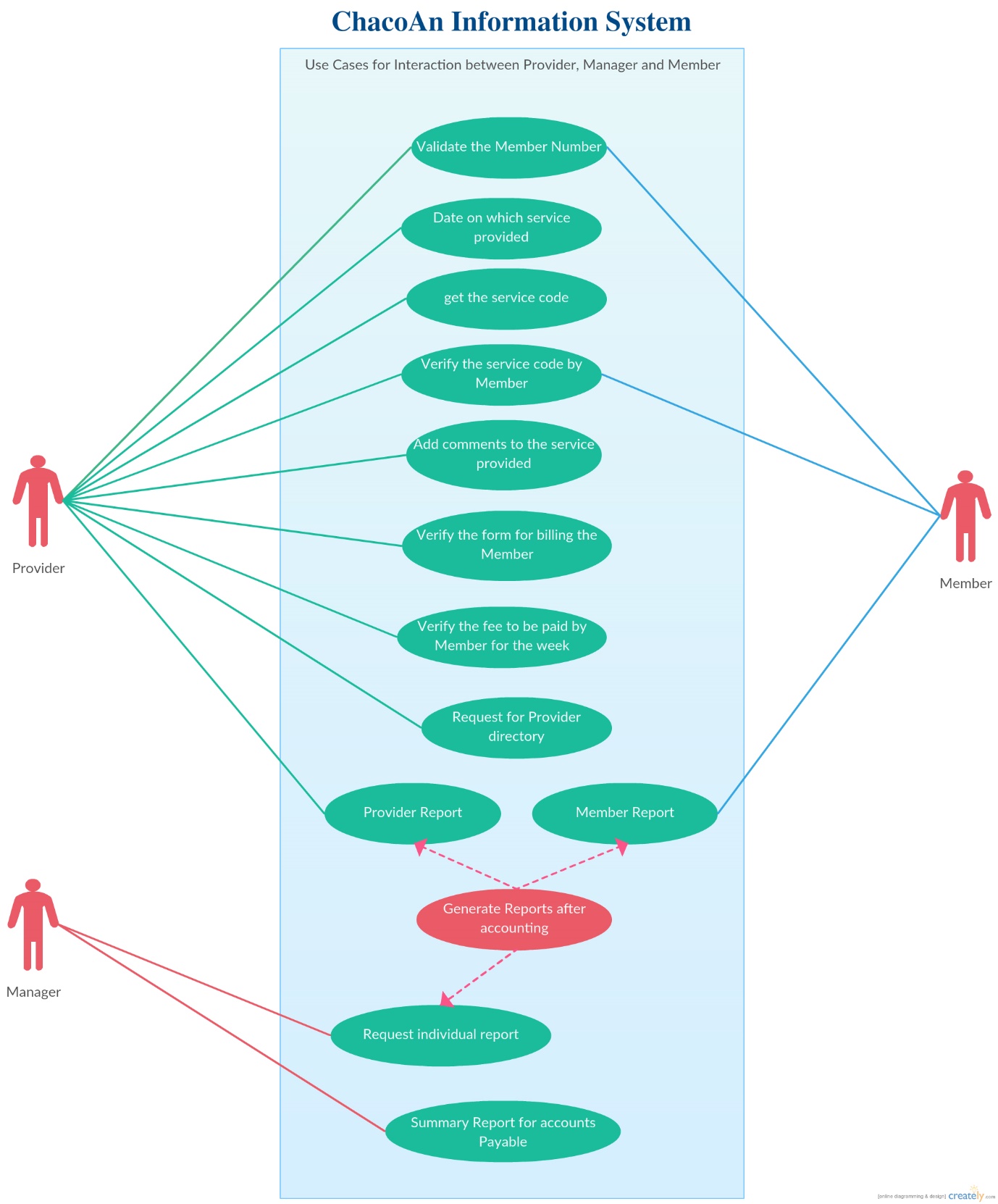
VINAY DATTA PINNAKA

SMARAN AERRAMSETTY

VENKATA MANTHENA

**Requirement Phase:**

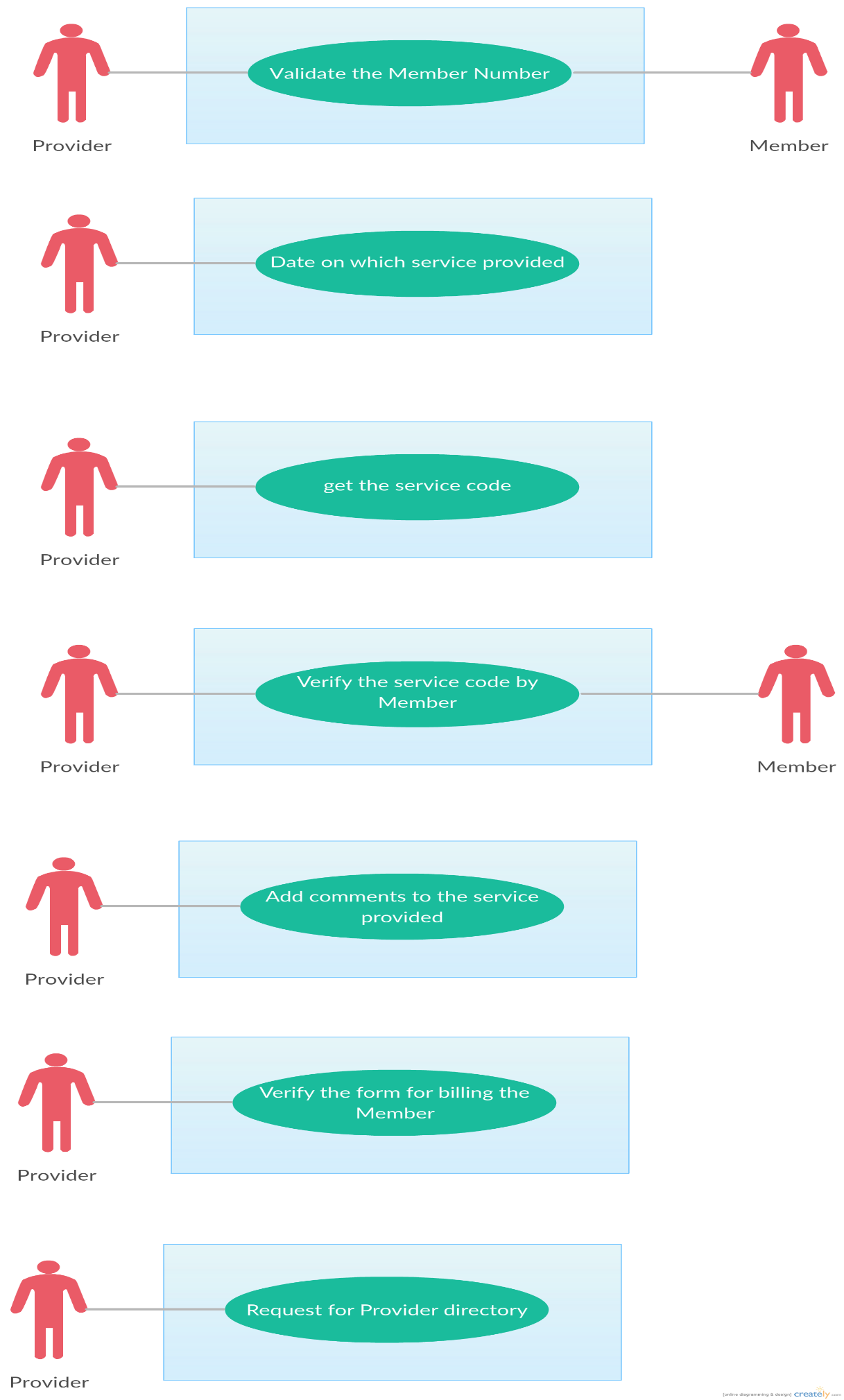
**Use Cases between Provider, Manager and Member:**



**Use Cases between Operator and Member:**



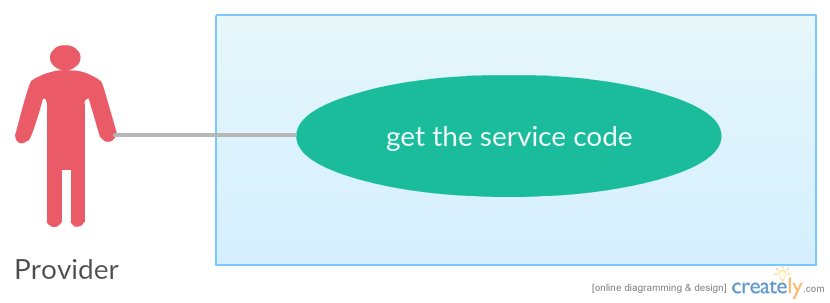
**USECASE: Validate the Member Information**



**Description:** When the Member approaches the provider, first his membership has to be validated with central database 9 digit code is validated if valid provider see a Valid Member message.

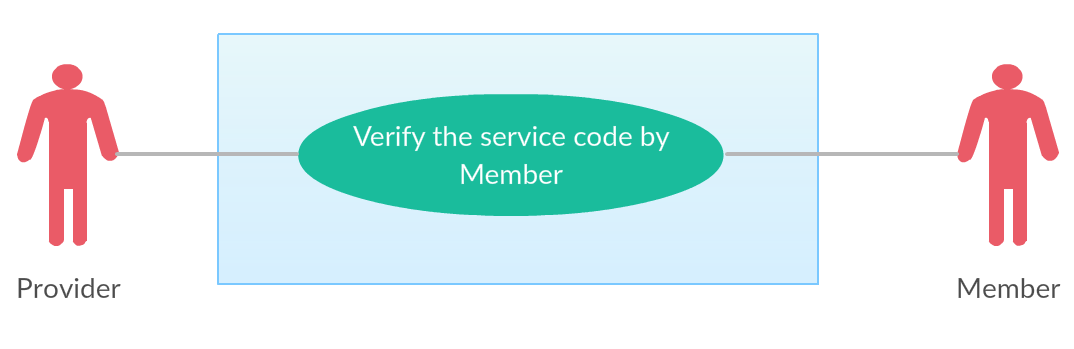
**USECASE: Date on which service provided**

**Description**: Provider enters the date on which the service is provided to the member.

**USECASE: Get the service code**

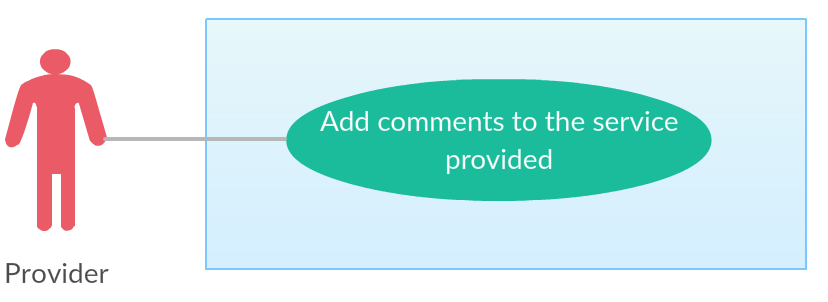
**Description**: Provider enters the 5 digit service code which is fetched from the database and displayed on the screen, if the code is invalid an invalid message is displayed.

**USECASE: Verify the service code by Member**

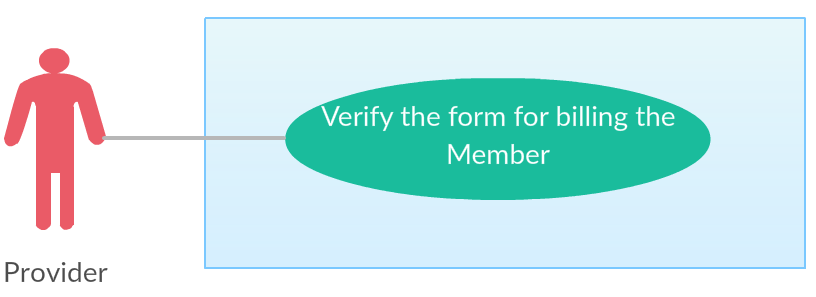


**Description**: The service obtained by the member is displayed and should be validated by the member.

**USECASE: Add comments to the service provided**

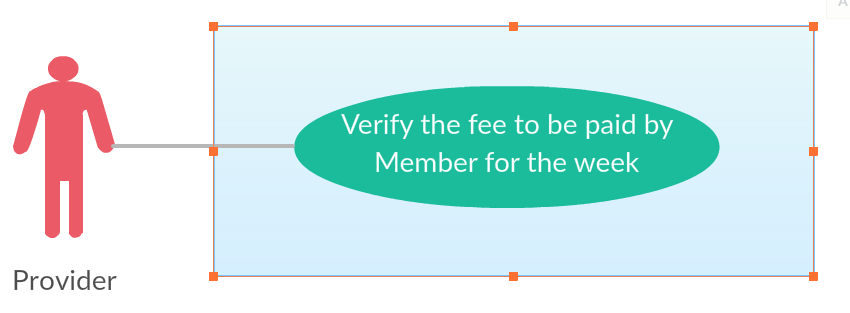


**Description**: Provider adds the comment to the service provided to the member. This may involve an extra service provided or remarks from the member.

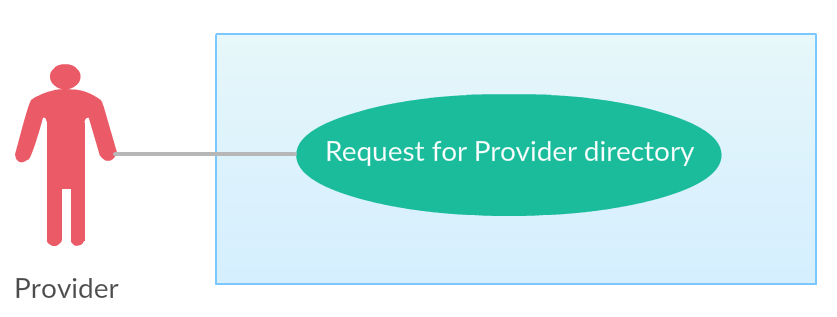
**USECASE: Verify the form for billing the member.**

**Description**: After submitting the service code for a member then Provider verifies the data that is stored, this avoids wrong billing.

**USECASE: Verify the fee to be paid by Member for the week**

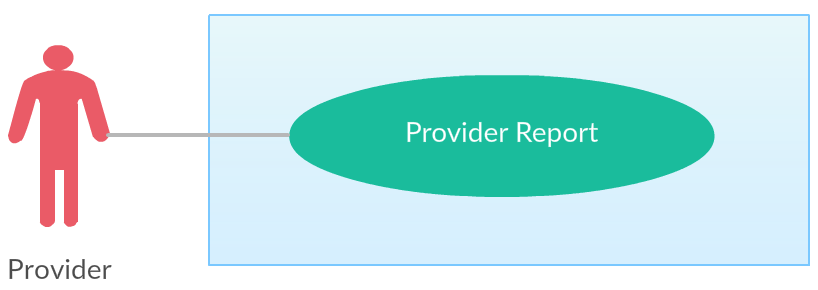


**Description**: Provider can verify the fee that should be paid by the member for the entire week. Fee for the entire week is calculated based on the services taken by the Member.

**USECASE: Request for Provider directory**

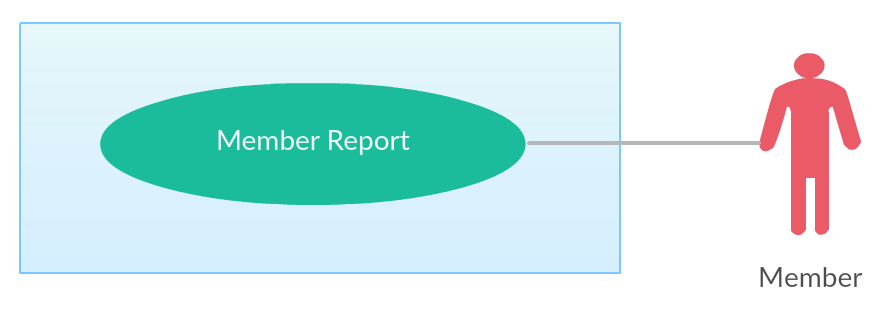
**Description**: At any time of the week provider can request for directory where all the member who have services with him are displayed.

**USECASE: Provider Report**



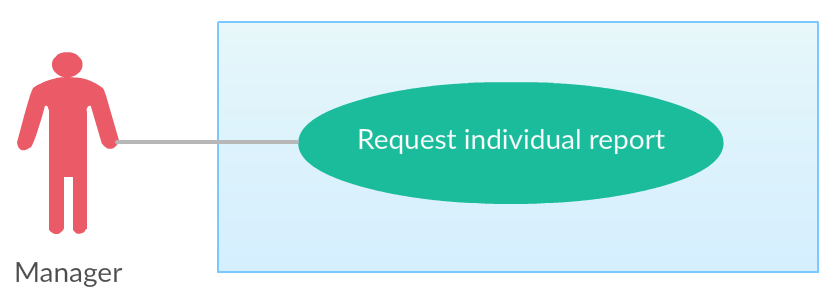
**Description**: After the accounting is done on the week end the reports are generated, Provider report contains his details, date on which he offered services, to whom he offered services, Fee paid by each member and the total consulting fee he gets for the week.

**USECASE: Member Report**



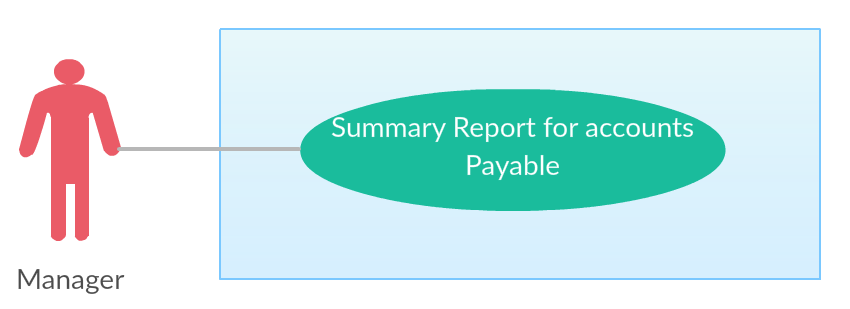
**Description**: In Member report his details, Services that he has taken on which date, Service provider name and the type of service.

**USECASE: Request Individual Report**



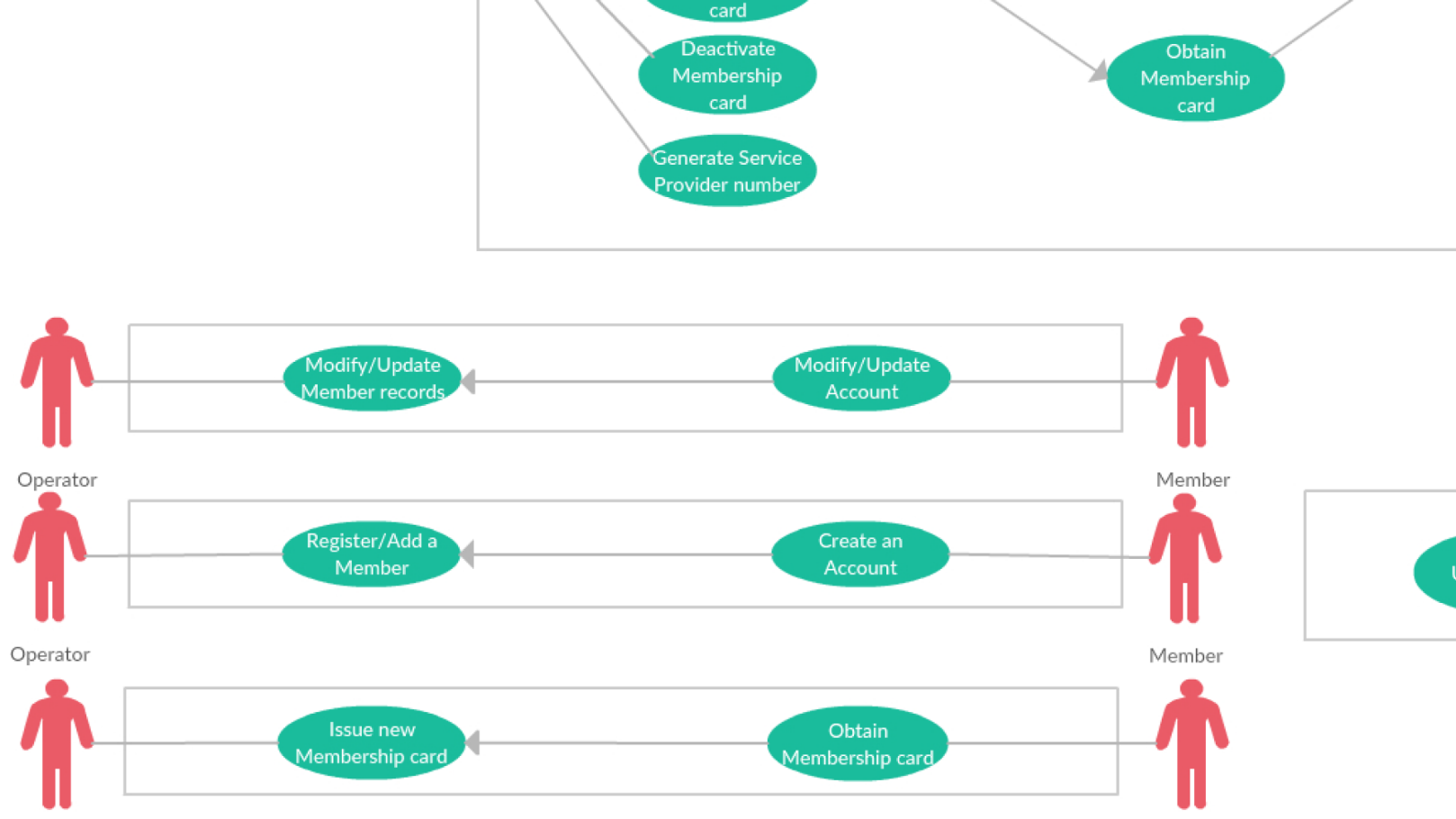
**Description**: Manager can request individual report either for the Provider or the Member.

**USECASE: Summary Reports for accounts Payable**



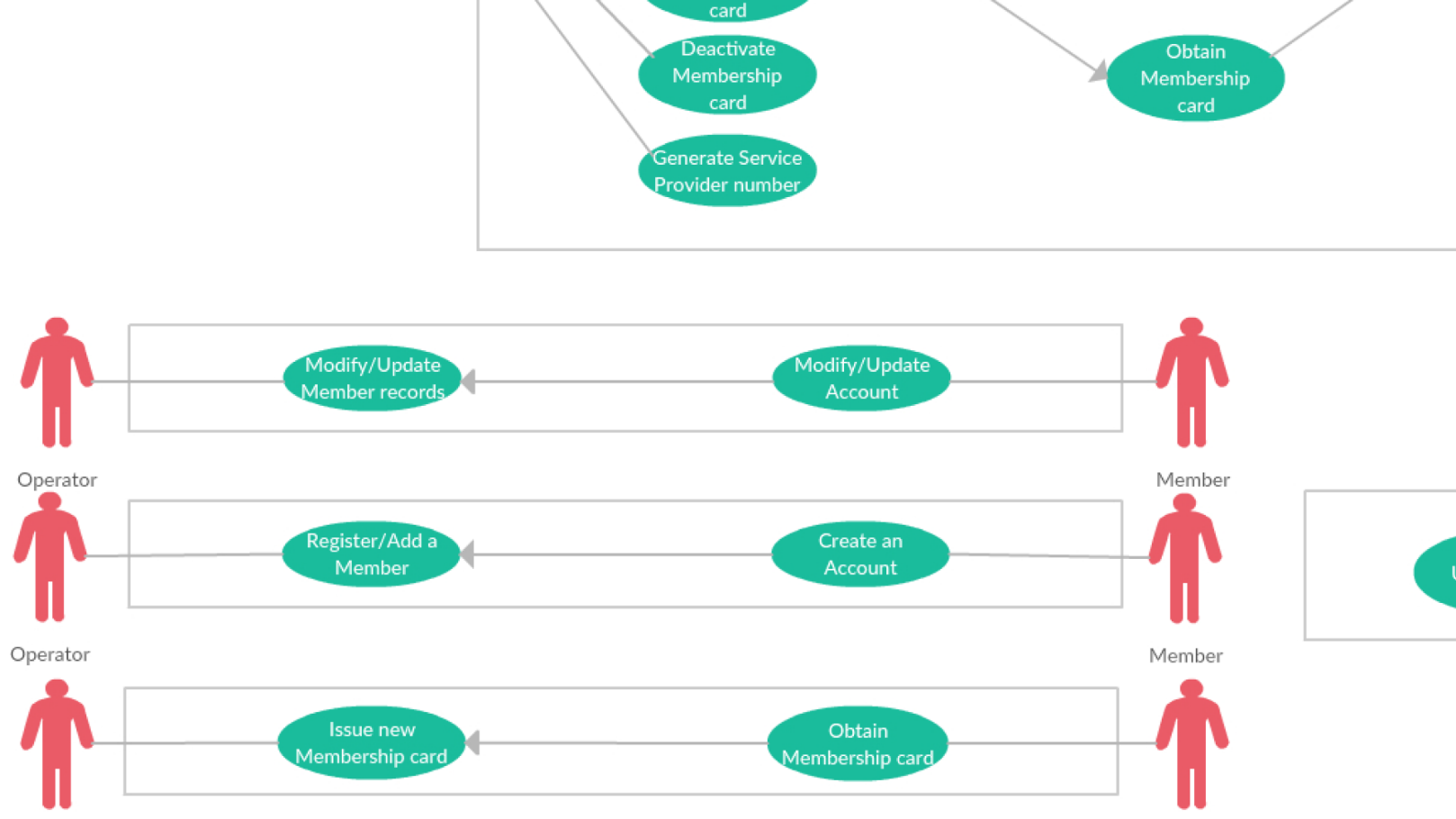
**Description**: At the end of each week Manager gets the Summary of reports for all the accounts payable mostly the Providers.

**USECASE: Modify/Update Member records**



**Description**: This use case enables the member to modify and update his account information which the operator can update accordingly.

**USECASE: Register/Add Member**



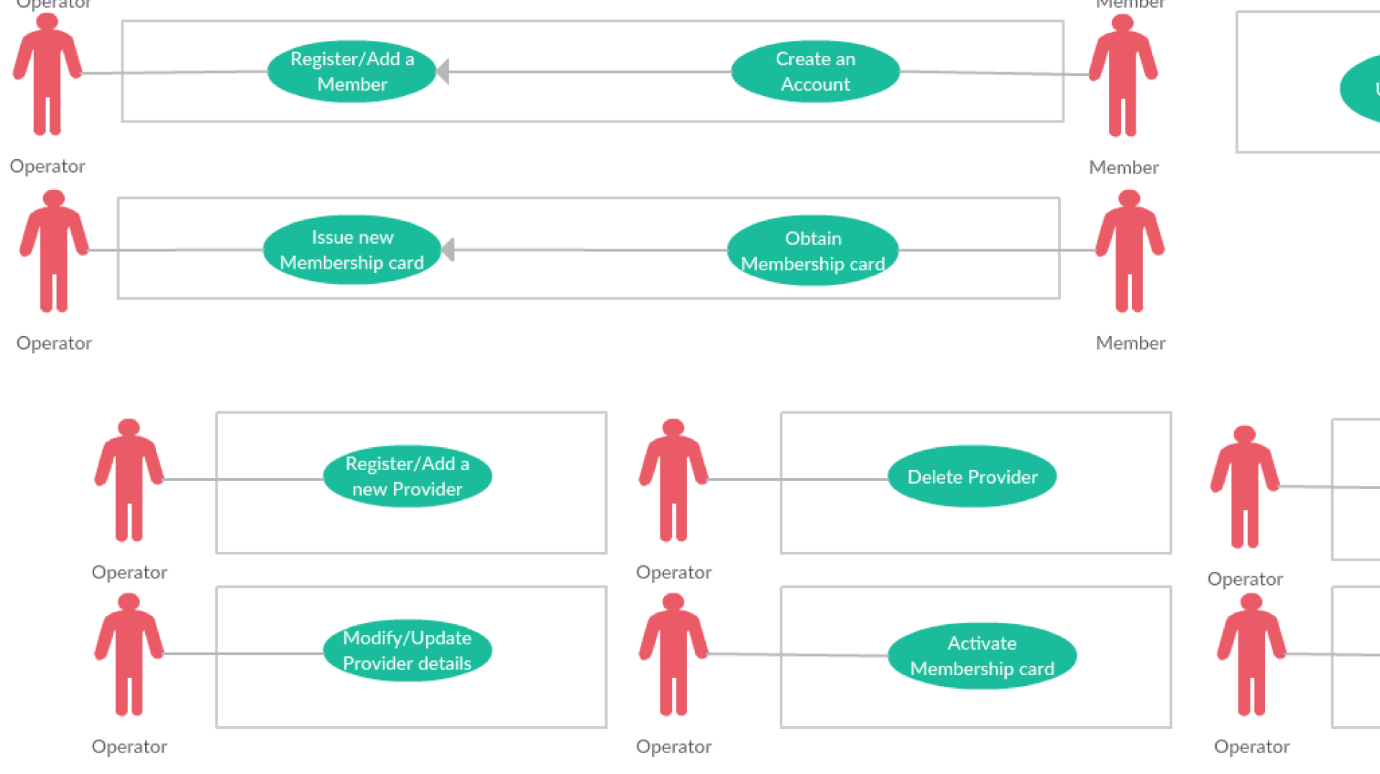
**Description**: This use case enables the member to create an account and the operator to register the member to avail the services.

**USECASE: Issue Membership Card**



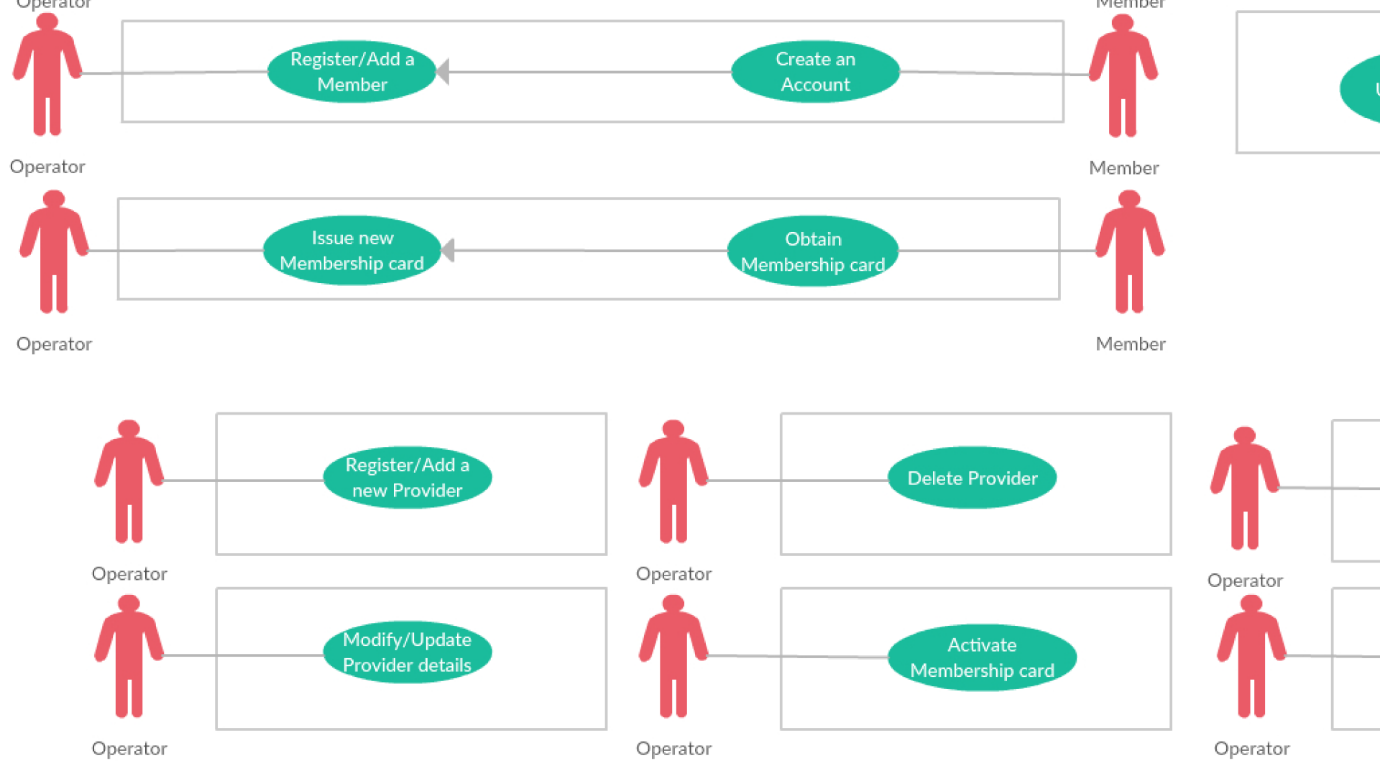
**Description**: This use case enables the member to obtain a new or replaced Membership card when issued by the operator.

**USECASE: Register/Add a Provider**



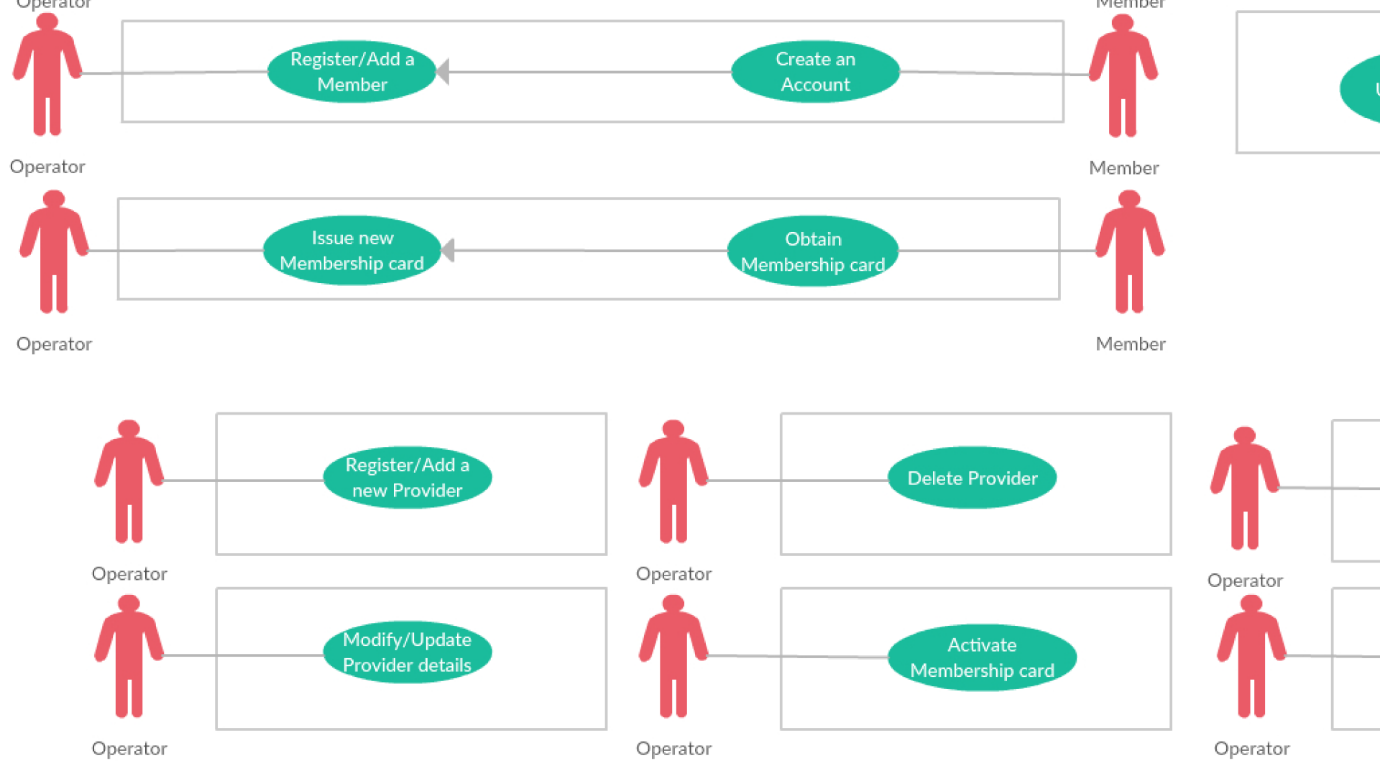
**Description**: This use case enables an Operator to Register or add a new Provider.

**USECASE: Modify/Update Provider Details**



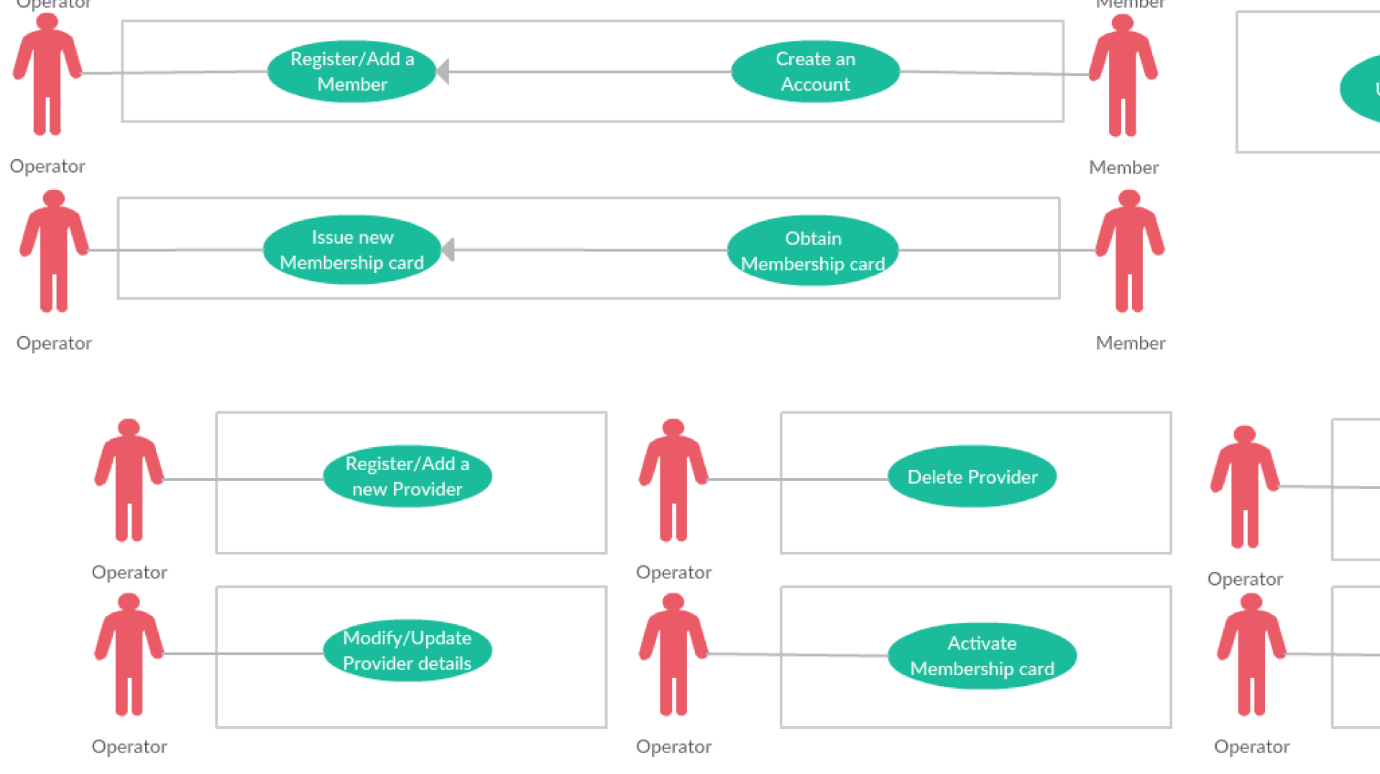
**Description**: This use case enables an Operator to Modify/Update the Provider details.

**USECASE: Delete Provider**



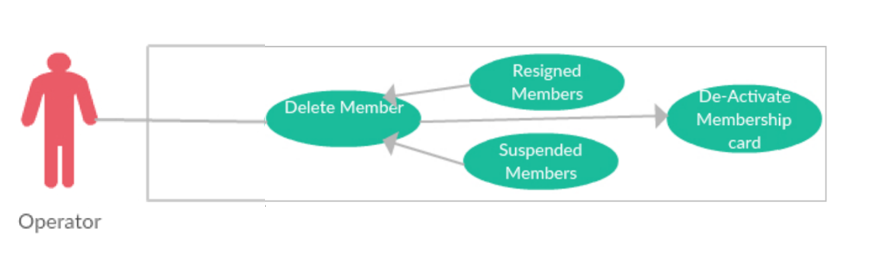
**Description**: This use case enables an Operator to delete the Provider from the records.

**USECASE: Activate Membership card**



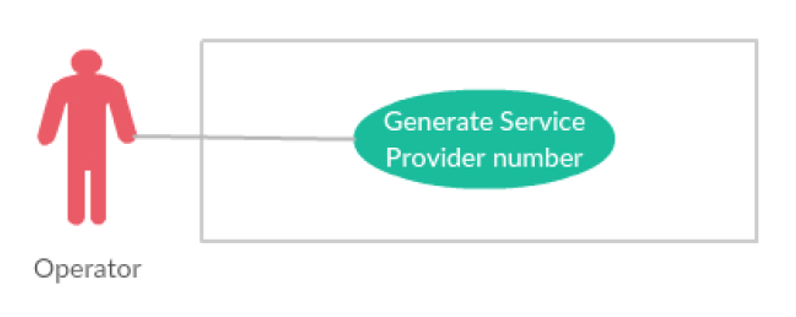
**Description**: This use case enables an Operator to Activate a Membership card.

**USECASE: Delete Member**



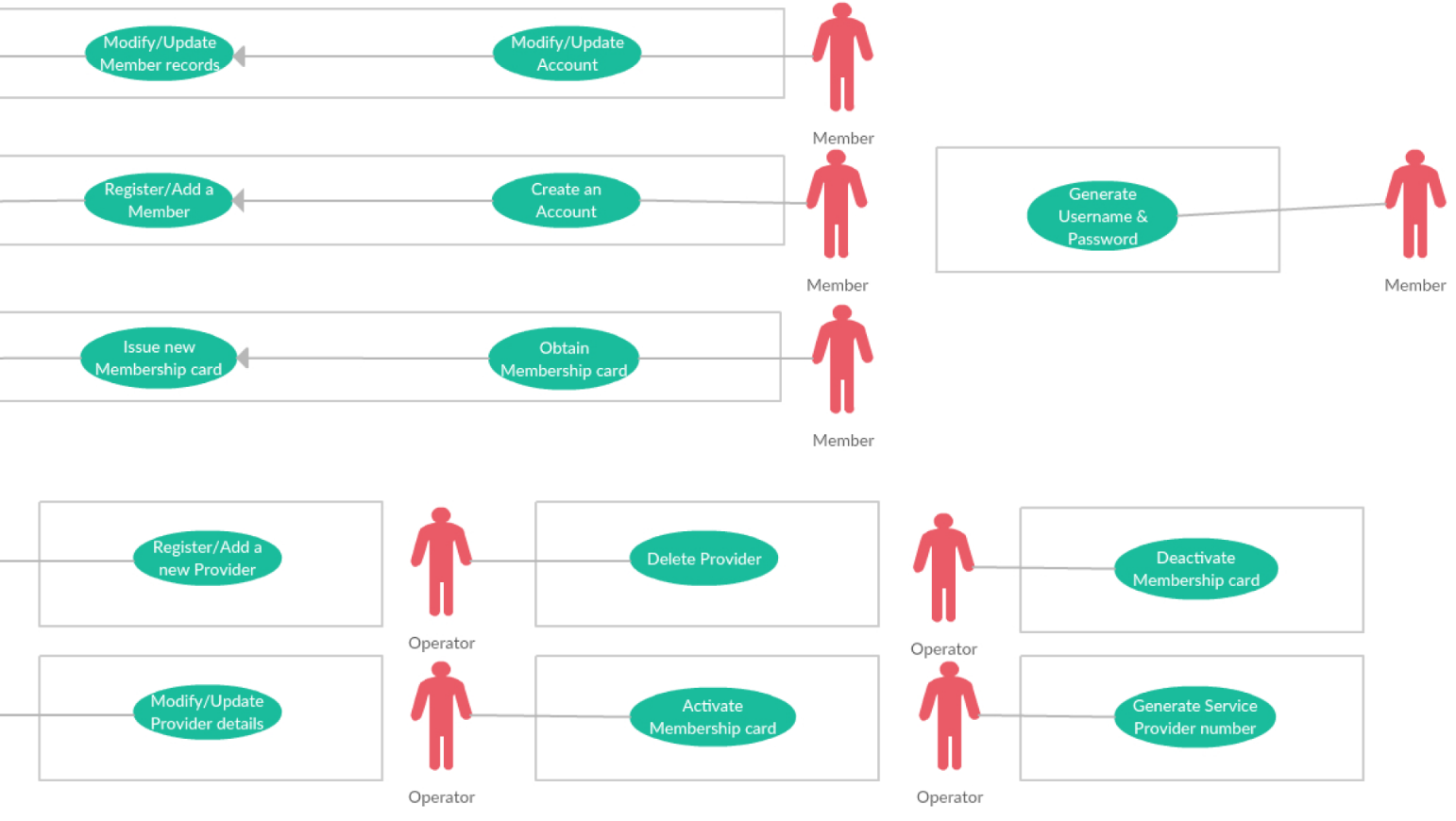
**Description**: This use case enables an operator to delete a member and deactivate his Membership Card.

**USECASE: Generate a Service Provider Number**



**Description**: This use case enables an operator to generate Service Provider Number

USECASE: Generate Username & Password



Description: This use case enables a Member to generate username and password for his/her account.

**Analysis Phase**

TEAM WORK – 2

ANALYSIS PHASE



**STEP BY STEP DESCRIPTION:**

**Provider Maintenance**:

In this use case, operator will create a new account(if new) for the provider to render his services. Once, an account is created the provider can update his registration information as and when required. The Provider maintenance can be divided into 3 use cases:

1. Register/Add a new provider
2. Modify/Update Provider Details
3. Delete Provider

A new account creation will require the provider to give the following details:

1. Name of service provider owner
2. Name of service
3. Type of service provided
4. Address
5. Contact Info

Once the account is created he can update the registration information by:

1. Entering the Service provider number
2. Update the name of Service provided
3. Update the address
4. Update contact info
5. Delete account option

**Member Maintenance**:

In this use case, the operator will create a new account for a new member to join. Once an account has been generated, the member can update his account information. This can be categorized into different use cases:

1. Register/ Add a new Member
2. Update member records
3. Delete Member

A new account generation will require the following details:

1. First Name of Member
2. Last Name of Member
3. Username of account
4. Address
5. Contact Info
6. Status of Member (Resigned, Suspended, Inactive)

Once a new account has been generated, the member can update his account information by the entering the following details:

1. Username of account
2. Update First Name
3. Update Last Name
4. Update Address
5. Update Contact Info
6. Delete Account

**Membership Card Maintenance**:

The operator has to keep a track of the membership card that is being used by the member. This serves as an ID card to the member. This can be categorized into 3 use cases:

1. Issue a new membership card
2. Activate Membership card
3. Deactivate Membership card

For issuing a membership card the details that are to be entered are:

1. Card No:
2. Issued Member Name:
3. Member ID:
4. Status: <set to Inactive by default>

For Activating Membership Card the details that are to be set are:

1. Card No:
2. Status : <Set to active>
3. Activation Date: <Current Date>

For Deactivating Membership card, the details that are to be set are:

1. Card No:
2. Status : <Set to Inactive>
3. Date of Expired : <Current Date>

**Generate Service Provider Number**:

Once a service provider has registered to render his services for the Chocoholics, the operator will issue him a new Service provider number. This number will be generated automatically on the basis of the Type of service provided and the number of service providers in that classification. For example if the new service provider wishes to provide “Dental” service and there are other 5 such providers, then the service provider number will be SP-DE-006 which can be understood as <**S**ervice **P**rovider> - <**DE**ntal>-<Number in order of service provider>.

**Generate Username and Password:**

Once the user has a new account he has to provide a username and password using which he can access his account information. For this he has to provide the following details:

1. Member Last Name:
2. Username:
3. Password:
4. Confirm Password:

**Generate Report:**

In this use case the entire report of the services rendered by whom and to whom details will be carefully stored. This can be further divided into

1. Service details
2. Payment details
3. Request for providing directory

As per the request of the user the recorded details should be displayed as a file or a table.

**Team Maintenance:**

This use case deals with the management of the team of ChocoAn system. It basically includes the details of the manager and the operator. This is sub-divided into :

1. Manager details
2. Operator details

These are not updatable by members or the providers. Thus doesn’t require any special updating classes.

**Verify Member:**

This use case is very essential as it authenticates the member when a person tries to access the service provided by the service provider. The user has to use his Membership card to verify at the service provider’s terminal.

**Step 2: Functional Modelling**

**Provider Maintenance:**

USE CASE: Register/Add a new Provider

Normal Scenario: Once a provider wants to create a new Account. He will be directed to a screen where he has to enter the following information and click submit:

1. Name of the service provider owner
2. Name of the service
3. Type of service provided
4. Address
5. Contact info

Exceptional Scenario: If the user leaves a blank field. There can be a pop up message to denote that he shouldn’t leave that particular field empty.

USE CASE: Modify/Update Provider details

Normal Scenario: Once a new account has been created the provider can modify his account information any time. He can log in to his account and click on update where he will be guided to update page. He can update the required fields as above and click on submit. The fields that have been filled will alone be updated and the rest will be left as it is. However if the user clicks on Delete account option. There will be a confirmation message which asks him to think again and if he stills stands on his choice, his account will be deleted.

Exceptional Scenario: If the provider mistakenly pressed the delete button, the confirmation dialog will handle him to go safely get back to his last page.

USE CASE: Delete Provider

Normal Scenario: If the user wants to delete his account. He can do so by clicking on the delete button. Here his account will be deactivated.

Exceptional Scenario: If the provider mistakenly pressed the delete button, the confirmation dialog will handle him to go safely get back to his last page.

**Member Maintenance:**

USE CASE: Register/Add a new member

Normal Scenario: If a new member wants to join the service he will be directed to a new account creation page which requires the following details:

1. First Name of Member
2. Last Name of Member
3. Username of account
4. Address
5. Contact Info
6. Status of Member (Resigned, Suspended, Inactive)

Exceptional Scenario: If the user leaves a blank field. There can be a pop up message to denote that he shouldn’t leave that particular field empty.

USE CASE: Update member records

Normal Scenario: Once a new account has been created the member can modify his account information any time. He can log in to his account and click on update where he will be guided to update page. He can update the required fields as above and click on submit. The fields that have been filled will alone be updated and the rest will be left as it is. However if the user clicks on Delete account option. There will be a confirmation message which asks him to think again and if he stills stands on his choice, his account will be deleted.

Exceptional Scenario: If the member mistakenly pressed the delete button, the confirmation dialog will handle him to go safely get back to his last page.

USE CASE: Delete Member

Normal Scenario: If the user wants to delete his account. He can do so by clicking on the delete button. Here his account will be deactivated.

Exceptional Scenario: If the member mistakenly pressed the delete button, the confirmation dialog will handle him to go safely get back to his last page.

**Membership Card Maintenance:**

USE CASE: Issue a new Membership card

Normal Scenario: A new membership card can be issued to the member once he registers for the service. For this he needs to enter the following information:

1. Card No:
2. Issued Member Name:
3. Member ID:
4. Status: <set to Inactive by default>

Exceptional Scenario: If the user leaves a blank field. There can be a pop up message to denote that he shouldn’t leave that particular field empty.

USE CASE: Activate Membership Card

Normal Scenario: Once new membership card is assigned it should be directly activated by the operator. The operator has to set the following properties for the card:

1. Card No:
2. Status : <Set to active>
3. Activation Date: <Current Date>

Exceptional Scenario: If there was some error in updating the card then it should be notified for operator as error in activation so that he can look into what happened.

USE CASE: Deactivate Membership card

Normal Scenario: If the operator wants to deactivate member’s card then he can do so by setting the status of the card to deactivated. These properties are to be set:

1. Card No:
2. Status : <Set to Inactive>
3. Date of Expired : <Current Date>

Exceptional Scenario: If there was some error in updating the card then it should be notified for operator as error in de-activation so that he can look into what happened.

USE CASE: Generate service provider number

Normal Scenario: Once a service provider has registered to render his services for the Chocoholics, the operator will issue him a new Service provider number. This number will be generated automatically on the basis of the Type of service provided and the number of service providers in that classification. For example if the new service provider wishes to provide “Dental” service and there are other 5 such providers, then the service provider number will be SP-DE-006 which can be understood as <**S**ervice **P**rovider> - <**DE**ntal>-<Number in order of service provider>.

Exceptional Scenario: If there was some error in generating the number then it should be notified for operator as error in generating service provider number so that he can look into what happened.

USE CASE: Generate Username and password

Normal Scenario: Once the user has a new account he has to provide a username and password using which he can access his account information. For this he has to provide the following details:

1. Member Last Name:
2. Username:
3. Password
4. Confirm Password

Exceptional Scenario: If the username already exists then there should be a dialog displayed asking the user to select another username. If password and confirm password fields do not match, then a warning should be displayed for the same.

**Generate Report:**

USE CASE: Service details

Normal Scenario: Here the details regarding the service rendered is saved. It has the following details:

1. Type of service rendered
2. Date on which service was given
3. Service code
4. Cost of the service

Exceptional Scenario: If there was an error in updating the records of any details mentioned above. The operator may immediately contact the corresponding provider about the issue.

USE CASE: Payment details

Normal Scenario: This use case deals with the payment details of the member when he requests a service. The user receives the bill and that must be updated in the records.

Exceptional Scenario: Payment generated may not be updated or might not have been processed correctly. The issue should be handled immediately by the operator and the provider.

USE CASE: Manager Details

Normal Scenario: This use case deals with storing the information of the manager such as his name and other related info. This is a static data and need less frequent updating.

Exceptional Scenario: There will be no exceptional situations in this use case

USE CASE: Operator Details

Normal Scenario: This use case deals with storing the information of the manager such as his name and other related info. This is a static data and need less frequent updating.

Exceptional Scenario: There will be no exceptional situations in this use case.

USE CASE: Verify Member

Normal Scenario: This use case deals with verifying the member details as he tries to use a service offered by the service provider. For this, as he uses his Membership Card the details to be verified are:

1. Member Name
2. Member status

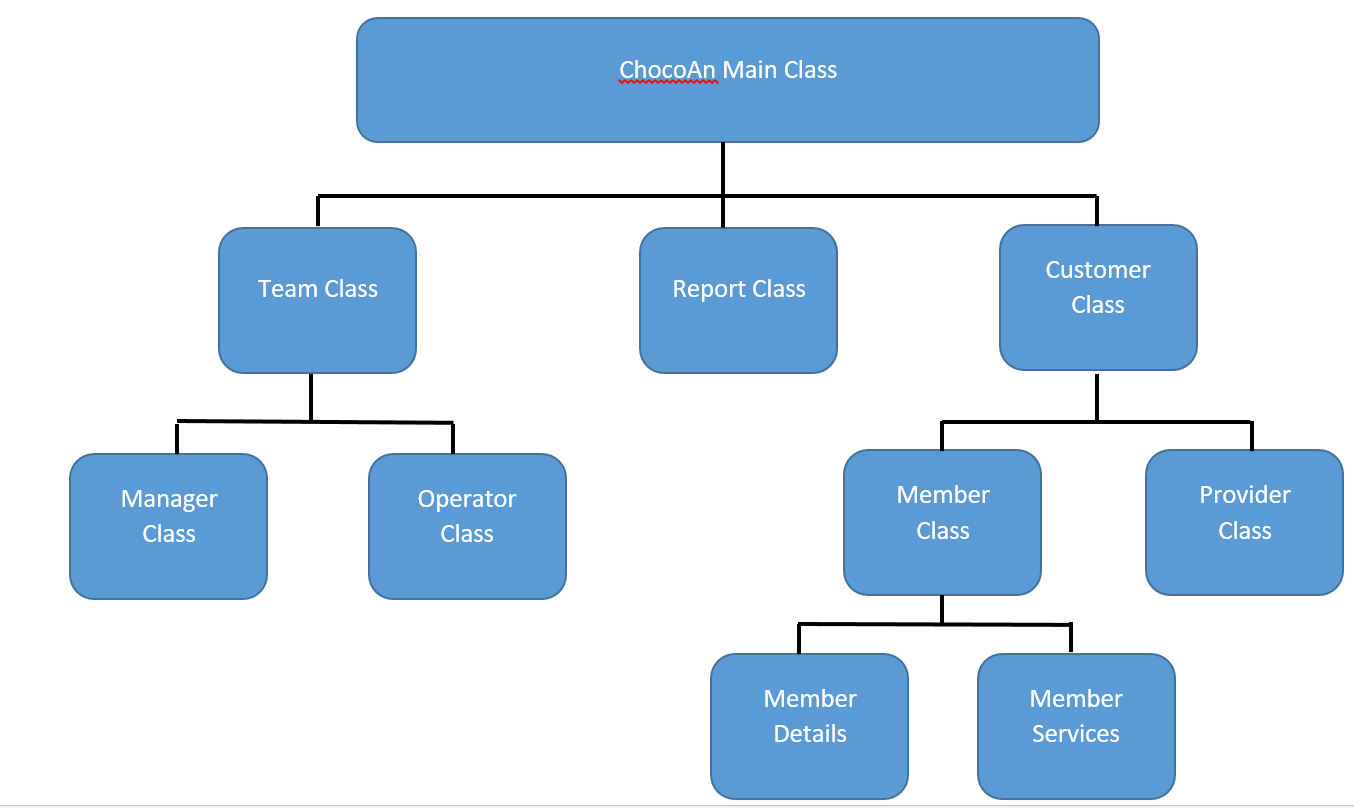
Exceptional Scenario: If the member status is obtained to be inactive or suspended, an immediate notice to the provider should be given not to provide services to that member.

**Step 3: Noun Extraction Details**

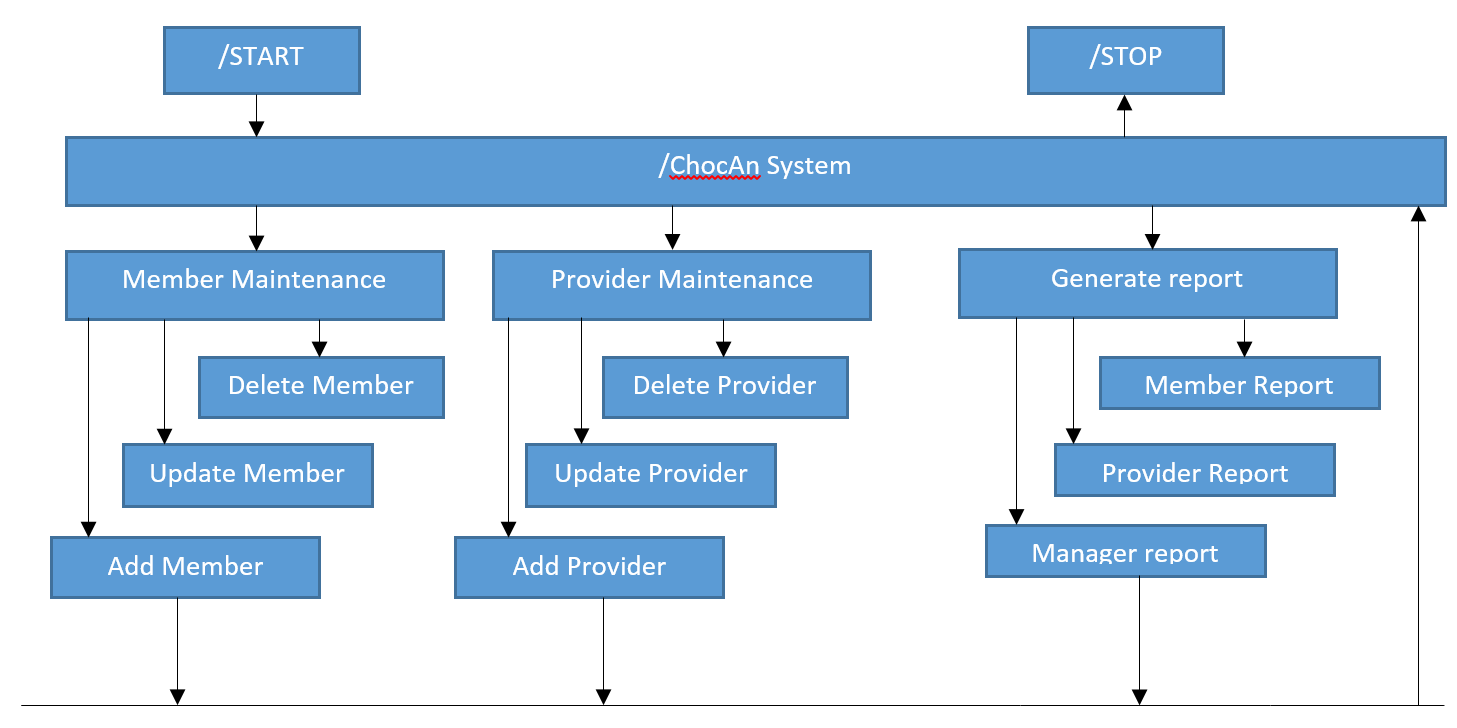
Chocoholics Anonymous is an association devoted to individuals who are dependent on chocolate. This associations has numerous providers and members enrolled. Members pay a charge for every service that the part used. The ChocAn system framework helps in creating reports like printing EFT report, Printing Members report, Printing Provider reports. Each visit of member is recorded and member can plan visits preceding the visits. Each visit is recorded with the subtle elements like date, type of service supplier. The administrations of the suppliers incorporate those of Dietitians and Exercise Specialist. The member and provider records are put away and can recovered by the manager and operators.

Nouns: Dietitian, Report, Exercise specialists, Members, Providers, Operators, Manager

**Class Diagram**

****

**DYNAMIC MODELLING**



**IDENTIFYING BOUNDARY CLASS AND CONTROL CLASS**

BOUNDAY CLASSES:

* Main User Interface
* Member’s Detailed Report
* Provider’s Detailed Report
* Manager’s Detailed Report
* User Input Interface

Control Classes:

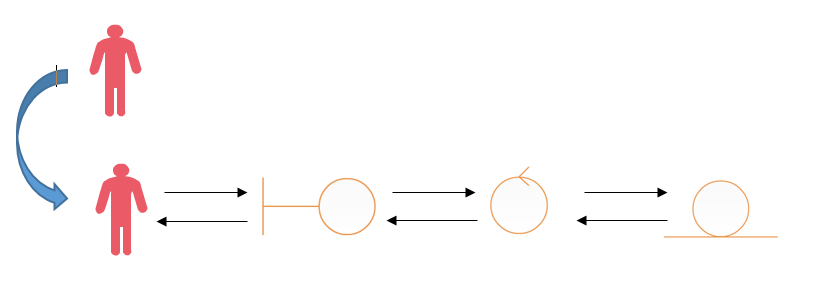
* Membership Maintenance
* Member Maintenance
* Provider Maintenance
* Visits Records
* Create Report
* Service Request

Entity Classes:

* Member database
* Provider database
* Service Information
* Visit Information

**COMMUNICATION DIAGRAMS:**

**ISSUE A NEW MEMBERSHIP CARD:**



Add membership

card

Update info

Details

Main User Interface

Card info

Database

Membership

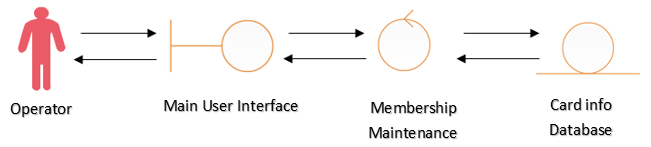
Maintenance

Class

Operator

Member

**ACTIVATE MEMBERSHIP CARD:**

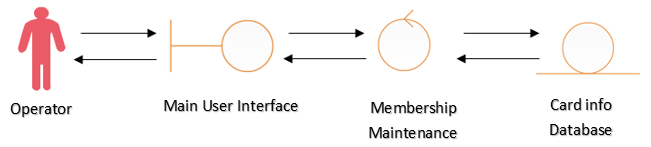


Update info

Activate

card

**DE-ACTIVATE MEMBERSHIP CARD:**

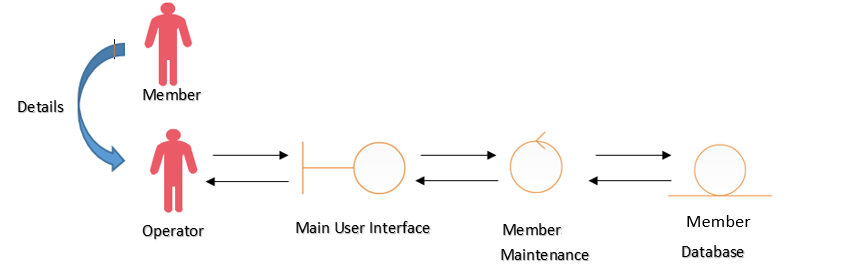


De-Activate

card

Update info

**ADD A MEMBER:**

****

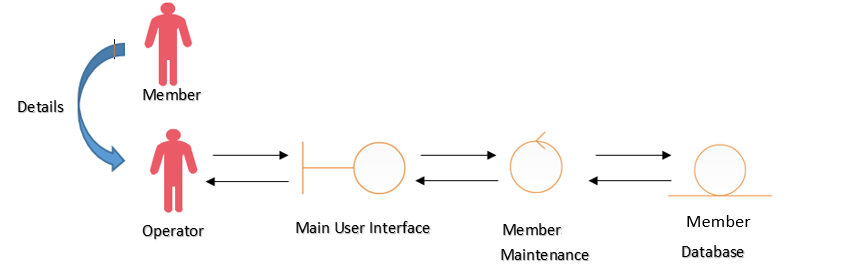
Update member

details

Select Add

member

**UPDATE MEMBER:**

****

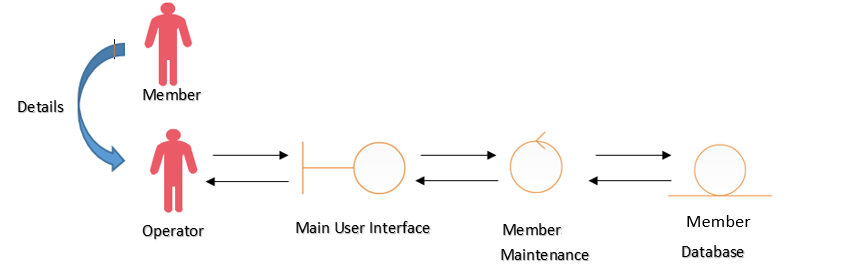
Select Update

member

Update member

details

**DELETE MEMBER:**

****

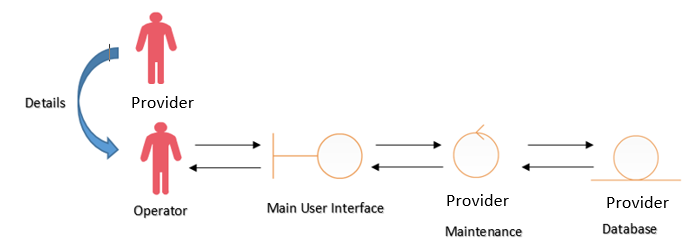
Select Delete

member

Update member

details

**ADD PROVIDER:**

****

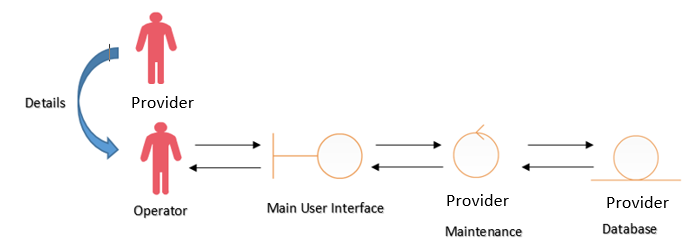
Update Provider

details

Select Add

Provider

**DELETE PROVIDER:**

****

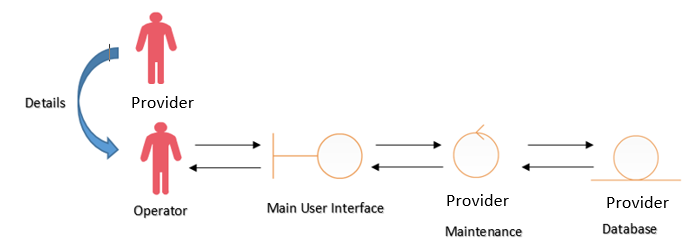
Update Provider

details

Select Delete

Provider

**UPDATE PROVIDER:**

****

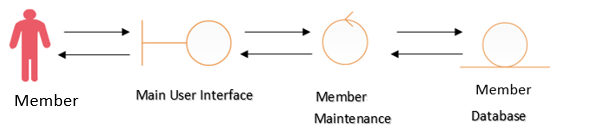
Update Provider

details

Select Update

Provider

**GENERATE USERNAME AND PASSWORD:**

****

Update member

details

Create Username

& Password

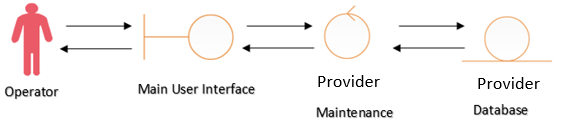
**GENERATE SERVICE PROVIDER NUMBER:**

Update

details

Generate

Number

****

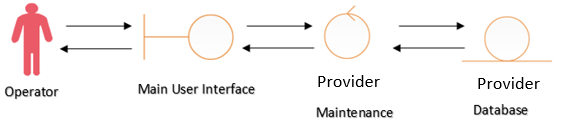
**SERVICE DETAILS:**

Update

details

Request

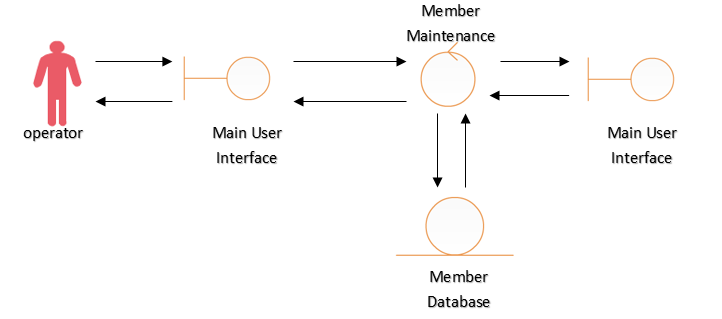
service

****

Service

Request

**VERIFY MEMBER:**

****

Show

Result

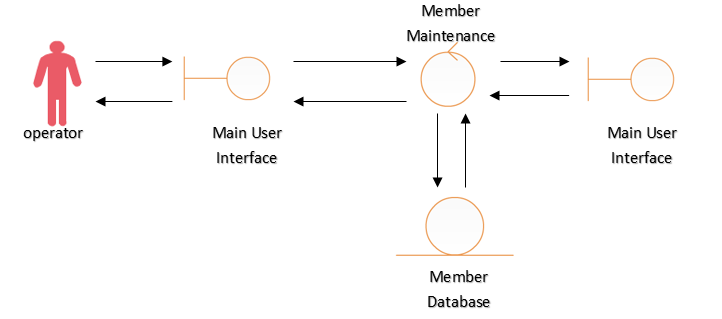
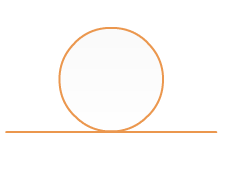
Authenticate

Send

Credentials

**GENERATE REPORT:**

Report Class

****

Display

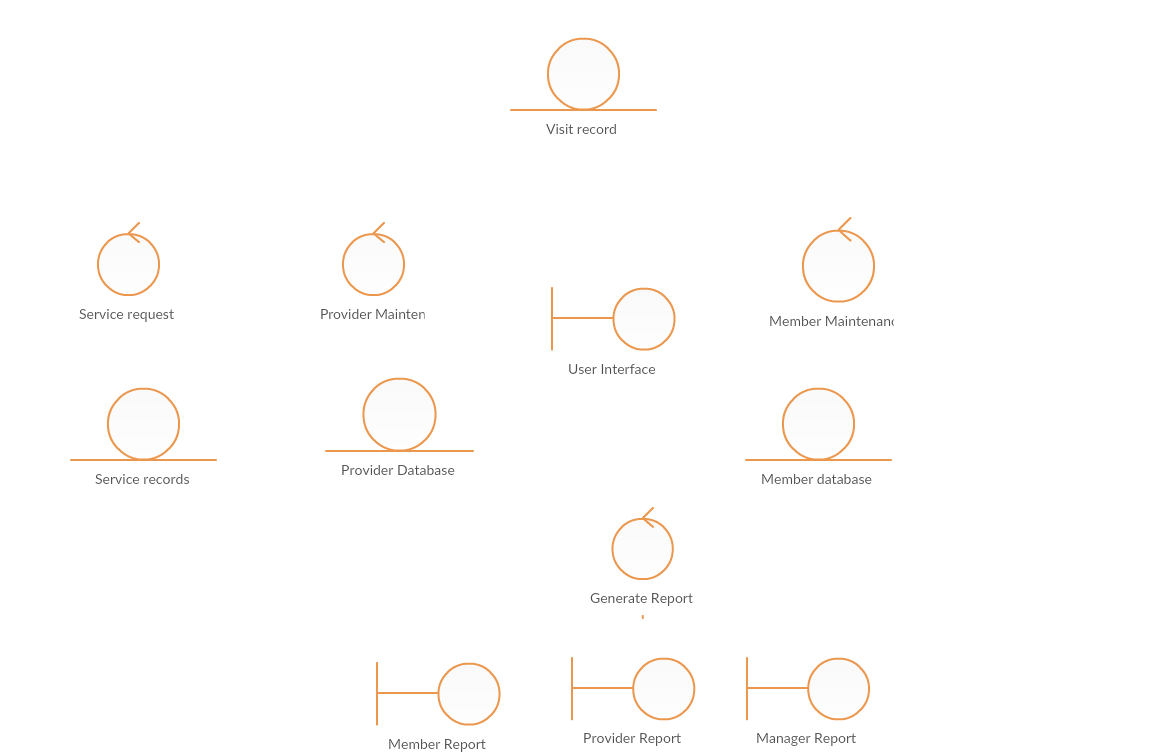
Report

Request

Report

Member report class

**REALIZATION CLASS DIAGRAM:**

****

**Design Phase :**

1. **Provider.java**

**public class Provider {**

**private Connection connection;**

**public Provider() {**

**connection = MemberUtil.getConnection();**

**}**

**public void addProviderDb(ProviderDb ProviderDb) {**

**try {**

**PreparedStatement preparedStatement = connection**

**.prepareStatement("INSERT INTO Provider\_new(firstname,lastname,streetadd,city,state,date,type,status) values(?,?,?,?,?,?,?,?)");**

**// Parameters start with 1**

**preparedStatement.setString(1, ProviderDb.getFirstName());**

**preparedStatement.setString(2, ProviderDb.getLastName());**

**preparedStatement.setString(3,ProviderDb.getStreetAdd());**

**preparedStatement.setString(4, ProviderDb.getCity());**

**preparedStatement.setString(5, ProviderDb.getState());**

**preparedStatement.setString(6, ProviderDb.getJoinDate());**

**preparedStatement.setString(7, ProviderDb.getType());**

**preparedStatement.setString(8, ProviderDb.getStatus());**

**preparedStatement.executeUpdate();**

**} catch (SQLException e) {**

**e.printStackTrace();**

**}**

**}**

**public void deleteProviderDb(int proid) {**

**try {**

**PreparedStatement preparedStatement = connection**

**.prepareStatement("delete from Provider\_new where proid=?");**

**// Parameters start with 1**

**preparedStatement.setInt(1,proid);**

**preparedStatement.executeUpdate();**

**} catch (SQLException e) {**

**e.printStackTrace();**

**}**

**}**

**public void updateProviderDb(ProviderDb ProviderDb) {**

**String sql = "update Provider\_new set streetadd=?,city=?,state=?,type=?,status=? where proid=?";**

**System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");**

**System.out.println(ProviderDb.getStreetAdd());**

**System.out.println(ProviderDb.getCity());**

**System.out.println(ProviderDb.getState());**

**System.out.println(ProviderDb.getproid());**

**System.out.println(ProviderDb.getStatus());**

**System.out.println("i am here");**

**try**

**{**

**PreparedStatement preparedStatement = connection.prepareStatement(sql);**

**preparedStatement.setString(1,ProviderDb.getStreetAdd());**

**preparedStatement.setString(2, ProviderDb.getCity());**

**preparedStatement.setString(3, ProviderDb.getState());**

**preparedStatement.setString(4, ProviderDb.getType());**

**preparedStatement.setString(5, ProviderDb.getStatus());**

**preparedStatement.setInt(6, ProviderDb.getproid());**

**preparedStatement.executeUpdate();**

**} catch (SQLException e) {**

**e.printStackTrace();**

**}**

**}**

**public List<ProviderDb> getAllProviderDbs() {**

**List<ProviderDb> ProviderDbs = new ArrayList<ProviderDb>();**

**try {**

**Statement statement = connection.createStatement();**

**ResultSet rs = statement.executeQuery("select \* from Provider\_new");**

**while (rs.next()) {**

**ProviderDb ProviderDb = new ProviderDb();**

**ProviderDb.setproid(rs.getInt("proid"));**

**ProviderDb.setFirstName(rs.getString("firstname"));**

**ProviderDb.setLastName(rs.getString("lastname"));**

**ProviderDb.setJoinDate(rs.getString("date"));**

**ProviderDb.setStreetAdd(rs.getString("streetadd"));**

**ProviderDb.setCity(rs.getString("city"));**

**ProviderDb.setState(rs.getString("state"));**

**ProviderDb.setType(rs.getString("type"));**

**ProviderDb.setStatus(rs.getString("status"));**

**ProviderDbs.add(ProviderDb);**

**}**

**} catch (SQLException e) {**

**e.printStackTrace();**

**}**

**return ProviderDbs;**

**}**

**public ProviderDb getProviderDbById(int ProviderDbId) {**

**ProviderDb ProviderDb = new ProviderDb();**

**try {**

**PreparedStatement preparedStatement = connection.**

**prepareStatement("select \* from Provider\_new where proid=?");**

**preparedStatement.setInt(1, ProviderDbId);**

**ResultSet rs = preparedStatement.executeQuery();**

**if (rs.next()) {**

**ProviderDb.setproid(rs.getInt("proid"));**

**ProviderDb.setFirstName(rs.getString("firstname"));**

**ProviderDb.setLastName(rs.getString("lastname"));**

**ProviderDb.setJoinDate(rs.getString("date"));**

**ProviderDb.setStreetAdd(rs.getString("streetadd"));**

**ProviderDb.setCity(rs.getString("city"));**

**ProviderDb.setState(rs.getString("state"));**

**ProviderDb.setType(rs.getString("type"));**

**ProviderDb.setStatus(rs.getString("status"));**

**}**

**} catch (SQLException e) {**

**e.printStackTrace();**

**}**

**return ProviderDb;**

**}**

**}**

1. **ProviderDB.java**

**public class ProviderDb {**

**private int proid;**

**private String firstName;**

**private String lastName;**

**private String joindate;**

**private String streetadd;**

**private String city;**

**private String state;**

**private String type;**

**private String status;**

**public int getproid() {**

**return proid;**

**}**

**public void setproid(int proid) {**

**this.proid = proid;**

**}**

**public String getFirstName() {**

**return firstName;**

**}**

**public void setFirstName(String firstName) {**

**this.firstName = firstName;**

**}**

**public String getLastName() {**

**return lastName;**

**}**

**public void setLastName(String lastName) {**

**this.lastName = lastName;**

**}**

**public String getJoinDate() {**

**return joindate;**

**}**

**public void setJoinDate(String joindate) {**

**this.joindate = joindate;**

**}**

**public String getStreetAdd() {**

**return streetadd;**

**}**

**public void setStreetAdd(String streetadd) {**

**this.streetadd = streetadd;**

**}**

**public String getCity() {**

**return city;**

**}**

**public void setCity(String city) {**

**this.city = city;**

**}**

**public String getState() {**

**return state;**

**}**

**public void setState(String state) {**

**this.state = state;**

**}**

**public String getType() {**

**return type;**

**}**

**public void setType(String type) {**

**this.type = type;**

**}**

**public String getStatus() {**

**return status;**

**}**

**public void setStatus(String status) {**

**this.status = status;**

**}**

**@Override**

**public String toString() {**

**return "Provider [proid=" + proid + ", firstName=" + firstName**

**+ ", lastName=" + lastName + ", street address=" + streetadd + ",city=" + city + ", state=" + state +",joindate=" + joindate + ",type=" + type + ", status="**

**+ status + "]";**

**}**

**}**

1. **Member.java**

**public class Member {**

**private Connection connection;**

**public Member() {**

**connection = MemberUtil.getConnection();**

**}**

**public void addMemberDb(MemberDb memberDb) {**

**try {**

**PreparedStatement preparedStatement = connection**

**.prepareStatement("INSERT INTO member\_new(firstname,lastname,streetadd,city,state,joindate,status) values(?,?,?,?,?,?,?)");**

**// Parameters start with 1**

**preparedStatement.setString(1, memberDb.getFirstName());**

**preparedStatement.setString(2, memberDb.getLastName());**

**preparedStatement.setString(3,memberDb.getStreetAdd());**

**preparedStatement.setString(4, memberDb.getCity());**

**preparedStatement.setString(5, memberDb.getState());**

**preparedStatement.setString(6, memberDb.getJoinDate());**

**preparedStatement.setString(7, memberDb.getStatus());**

**preparedStatement.executeUpdate();**

**} catch (SQLException e) {**

**e.printStackTrace();**

**}**

**}**

**public void deleteMemberDb(int memid) {**

**try {**

**PreparedStatement preparedStatement = connection**

**.prepareStatement("delete from member\_new where memid=?");**

**// Parameters start with 1**

**preparedStatement.setInt(1,memid);**

**preparedStatement.executeUpdate();**

**} catch (SQLException e) {**

**e.printStackTrace();**

**}**

**}**

**public void updateMemberDb(MemberDb memberDb) {**

**String sql = "update member\_new set streetadd=?,city=?,state=?,status=? where memid=?";**

**System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");**

**System.out.println(memberDb.getStreetAdd());**

**System.out.println(memberDb.getCity());**

**System.out.println(memberDb.getState());**

**System.out.println(memberDb.getMemid());**

**System.out.println(memberDb.getStatus());**

**System.out.println("i am here");**

**try**

**{**

**PreparedStatement preparedStatement = connection.prepareStatement(sql);**

**preparedStatement.setString(1,memberDb.getStreetAdd());**

**preparedStatement.setString(2, memberDb.getCity());**

**preparedStatement.setString(3, memberDb.getState());**

**preparedStatement.setString(4, memberDb.getStatus());**

**preparedStatement.setInt(5, memberDb.getMemid());**

**preparedStatement.executeUpdate();**

**} catch (SQLException e) {**

**e.printStackTrace();**

**}**

**}**

**public List<MemberDb> getAllMemberDbs() {**

**List<MemberDb> MemberDbs = new ArrayList<MemberDb>();**

**try {**

**Statement statement = connection.createStatement();**

**ResultSet rs = statement.executeQuery("select \* from member\_new");**

**while (rs.next()) {**

**MemberDb MemberDb = new MemberDb();**

**MemberDb.setMemid(rs.getInt("memid"));**

**MemberDb.setFirstName(rs.getString("firstname"));**

**MemberDb.setLastName(rs.getString("lastname"));**

**MemberDb.setJoinDate(rs.getString("joindate"));**

**MemberDb.setStreetAdd(rs.getString("streetadd"));**

**MemberDb.setCity(rs.getString("city"));**

**MemberDb.setState(rs.getString("state"));**

**MemberDb.setStatus(rs.getString("status"));**

**MemberDbs.add(MemberDb);**

**}**

**} catch (SQLException e) {**

**e.printStackTrace();**

**}**

**return MemberDbs;**

**}**

**public MemberDb getMemberDbById(int MemberDbId) {**

**MemberDb MemberDb = new MemberDb();**

**try {**

**PreparedStatement preparedStatement = connection.**

**prepareStatement("select \* from member\_new where memid=?");**

**preparedStatement.setInt(1, MemberDbId);**

**ResultSet rs = preparedStatement.executeQuery();**

**if (rs.next()) {**

**MemberDb.setMemid(rs.getInt("memid"));**

**MemberDb.setFirstName(rs.getString("firstname"));**

**MemberDb.setLastName(rs.getString("lastname"));**

**MemberDb.setJoinDate(rs.getString("joindate"));**

**MemberDb.setStreetAdd(rs.getString("streetadd"));**

**MemberDb.setCity(rs.getString("city"));**

**MemberDb.setState(rs.getString("state"));**

**MemberDb.setStatus(rs.getString("status"));**

**}**

**} catch (SQLException e) {**

**e.printStackTrace();**

**}**

**return MemberDb;**

**}**

**}**

1. **MemberDB.java**

**public class MemberDb {**

**private int memid;**

**private String firstName;**

**private String lastName;**

**private String joindate;**

**private String streetadd;**

**private String city;**

**private String state;**

**private String status;**

**public int getMemid() {**

**return memid;**

**}**

**public void setMemid(int memid) {**

**this.memid = memid;**

**}**

**public String getFirstName() {**

**return firstName;**

**}**

**public void setFirstName(String firstName) {**

**this.firstName = firstName;**

**}**

**public String getLastName() {**

**return lastName;**

**}**

**public void setLastName(String lastName) {**

**this.lastName = lastName;**

**}**

**public String getJoinDate() {**

**return joindate;**

**}**

**public void setJoinDate(String joindate) {**

**this.joindate = joindate;**

**}**

**public String getStreetAdd() {**

**return streetadd;**

**}**

**public void setStreetAdd(String streetadd) {**

**this.streetadd = streetadd;**

**}**

**public String getCity() {**

**return city;**

**}**

**public void setCity(String city) {**

**this.city = city;**

**}**

**public String getState() {**

**return state;**

**}**

**public void setState(String state) {**

**this.state = state;**

**}**

**public String getStatus() {**

**return status;**

**}**

**public void setStatus(String status) {**

**this.status = status;**

**}**

**@Override**

**public String toString() {**

**return "Member [memid=" + memid + ", firstName=" + firstName**

**+ ", lastName=" + lastName + ", street address=" + streetadd + ",city=" + city + ", state=" + state +",joindate=" + joindate + ", status="**

**+ status + "]";**

**}**

**}**

1. **AddMem.java**

**public class AddMem extends HttpServlet {**

**private static final long serialVersionUID = 1L;**

**/\*\***

**\* @see HttpServlet#HttpServlet()**

**\*/**

**public AddMem() {**

**super();**

**// TODO Auto-generated constructor stub**

**}**

**/\*\***

**\* @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)**

**\*/**

**protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {**

**// TODO Auto-generated method stub**

**System.out.println("i am here doGet");**

**}**

**/\*\***

**\* @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)**

**\*/**

**protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {**

**// TODO Auto-generated method stub**

**System.out.println("i am here doPOst");**

**System.out.println("firstname:"+request.getParameter("firstname"));**

**System.out.println("lastname:"+request.getParameter("lastname"));**

**System.out.println("Street Address:"+request.getParameter("stadd"));**

**System.out.println("City:"+request.getParameter("city"));**

**System.out.println("State:"+request.getParameter("state"));**

**System.out.println("Date:"+request.getParameter("date"));**

**System.out.println("Status:"+request.getParameter("status"));**

**String firstname = request.getParameter("firstname");**

**String lastname = request.getParameter("lastname");**

**String stadd = request.getParameter("stadd");**

**String city = request.getParameter("city");**

**String state = request.getParameter("state");**

**String date = request.getParameter("date");**

**String status = request.getParameter("status");**

**Connection c = null;**

**response.setContentType("text/html");**

**PrintWriter out = response.getWriter();**

**out.println("<title>Example<title>" +**

**"<body bgcolor=FFFFFF>");**

**System.out.println("i am here");**

**out.println("<h2>Record added</h2>");**

**// DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());**

**//Establish database connection**

**try{**

**Class.forName("com.mysql.jdbc.Driver");**

**// Step 1: Allocate a database Connection object**

**c = DriverManager.getConnection(**

**"jdbc:mysql://localhost:3306/test", "root", "root"); // <== Check!**

**// database-URL(hostname, port, default database), username, password**

**// Step 2: Allocate a Statement object within the Connection**

**//Code to write to database**

**Statement stmt = c.createStatement();**

**String updateString = "INSERT INTO member\_new(firstname,lastname,streetadd,city,state,joindate,status) " +**

**"VALUES ('" + firstname + "','"+lastname+"','"+stadd+"','"+city+"','"+state+"','"+date+"','"+status+"')";**

**int count = stmt.executeUpdate(updateString);**

**System.out.println("count:"+count);**

**//Code to read from database**

**ResultSet results = stmt.executeQuery(**

**"SELECT memid FROM member\_new ");**

**while(results.next()){**

**String s = results.getInt("memid")+"";**

**System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"+s);**

**out.println("<BR><STRONG>Member id is</STRONG>");**

**out.println(s);**

**request.setAttribute("msgid" , s);**

**request.getRequestDispatcher("/Result.jsp").forward(request, response);**

**// RequestDispatcher dispatcher = request.getRequestDispatcher("Result.jsp");**

**// dispatcher.forward(request,response);**

**// getServletConfig().getServletContext()**

**//.getRequestDispatcher("/Result.jsp")**

**//.forward(request, response);**

**}**

**stmt.close();**

**}catch(java.sql.SQLException e){**

**System.out.println(e.toString());**

**}/\* catch (ClassNotFoundException e) {**

**// TODO Auto-generated catch block**

**e.printStackTrace();**

**} \*/ catch (ClassNotFoundException e) {**

**// TODO Auto-generated catch block**

**e.printStackTrace();**

**}**

**out.close();**

**}**

**public void view() throws ClassNotFoundException**

**{**

**try**

**{**

**String sql="select \* from member\_new";**

**Class.forName("com.mysql.jdbc.Driver");**

**Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/test", "root", "root");**

**Statement stmt=con.createStatement();**

**ResultSet rs=stmt.executeQuery(sql);**

**while(rs.next())**

**{**

**rs.getInt("memid");**

**rs.getString("firstname");**

**rs.getString("lastname");**

**rs.getString("streetadd");**

**rs.getString("city");**

**rs.getString("state");**

**rs.getString("joindate");**

**rs.getString("status");**

**System.out.println(rs.getString("name"));**

**}**

**}**

**catch (SQLException e)**

**{**

**e.printStackTrace();**

**}**

**return;**

**}**

**public String insertuser() throws SQLException**

**{**

**String sql = "INSERT INTO member\_new(firstname,lastname,streetadd,city,state,joindate,status) values(?,?,?,?,?,?,?)";**

**Connection con = null;**

**PreparedStatement prep = null;**

**try**

**{**

**Class.forName("com.mysql.jdbc.Driver");**

**con = DriverManager.getConnection("jdbc:mysql://localhost:3306/test", "root", "root");**

**prep = con.prepareStatement(sql);**

**prep.setString(1, "firstname");**

**prep.setString(2, "lastname");**

**prep.setString(3, "city");**

**prep.setString(4, "state");**

**prep.setString(5, "joindate");**

**prep.setString(6, "status");**

**prep.executeUpdate();**

**prep.close();**

**}**

**catch (ClassNotFoundException e)**

**{**

**// TODO Auto-generated catch block**

**e.printStackTrace();**

**}**

**return "successfully added";**

**}**

**public String update() throws SQLException**

**{**

**String sql = "update member\_new set streetadd=?,city=?,state=?,status=? where firstname=?";**

**Connection con = null;**

**PreparedStatement prep = null;**

**try**

**{**

**Class.forName("com.mysql.jdbc.Driver");**

**con = DriverManager.getConnection("jdbc:mysql://localhost:3306/test", "root", "root");**

**prep = con.prepareStatement(sql);**

**prep.setString(1, "firstname");**

**prep.setString(2, "streetadd");**

**prep.setString(3, "city");**

**prep.setString(4, "state");**

**prep.setString(5, "status");**

**prep.executeUpdate();**

**prep.close();**

**}**

**catch (ClassNotFoundException e)**

**{**

**// TODO Auto-generated catch block**

**e.printStackTrace();**

**}**

**return "successfully update";**

**}**

**public String delete() throws SQLException**

**{**

**String sql = "delete form member\_new where memid=(select memid from member\_new where firstname=? and lastname=?";**

**Connection con = null;**

**PreparedStatement prep = null;**

**try**

**{**

**Class.forName("com.mysql.jdbc.Driver");**

**con = DriverManager.getConnection("jdbc:mysql://localhost:3306/test", "root", "root");**

**prep = con.prepareStatement(sql);**

**prep.setString(1, "firstname");**

**prep.setString(2, "lastname");**

**prep.executeUpdate();**

**prep.close();**

**}**

**catch (ClassNotFoundException e)**

**{**

**// TODO Auto-generated catch block**

**e.printStackTrace();**

**}**

**return "successfully delete";**

**}**

**}**

1. **MemberUtil.java**

**public class MemberUtil {**

**private static Connection connection = null;**

**public static Connection getConnection() {**

**if (connection != null)**

**return connection;**

**else {**

**try {**

**Properties prop = new Properties();**

**String driver="com.mysql.jdbc.Driver";**

**String url="jdbc:mysql://localhost:3306/test";**

**String user="root";**

**String password="root";**

**Class.forName(driver);**

**connection = DriverManager.getConnection(url, user, password);**

**} catch (ClassNotFoundException e) {**

**e.printStackTrace();**

**} catch (SQLException e) {**

**e.printStackTrace();**

**}**

**return connection;**

**}**

**}**

**}**