+86

CAIS Project for CSC 470

Team Members:

Bryce Herring

Cameron Mitchell

Edward Stephens

Table of Contents

Requirements elicitation for the CAIS system

- **Functional Requirements:**
  - **FR1: Datacenter Manager Adds, Updates, and Removes Members for the ChocAn Datacenter.**
  - **FR2: Datacenter Manager Adds, Updates, and Removes Providers for the ChocAn Datacenter.**
  - **FR3: Member Validates Membership for Provider.**
  - **FR4: Provider sends bill to the ChocAn Datacenter.**
  - **FR5: Software looks up and displays fee for services rendered to the Provider.** Provider does it as the actor.
  - **FR6: Provider verifies services provided to Member for ChocAn Datacenter.**
  - **FR7: Software provides weekly reports on Accounts Payable and Member services provided to Datacenter Managers.** Manager does it.
  - **FR8: Software records data to a disk of EFT (Electronic Fund Transfer) data for the current week.**

Good!

- **Nonfunctional Requirements:**
  - **NFR1: Provider having access to the Provider Directory.**
  - **NFR2: Provider having access to a weekly report of services rendered.**
  - **NFR3: ChocAn datacenter managers have access to multiple weekly reports for services rendered by all providers.**
  - **NFR4: ChocAn datacenter managers have access to weekly account payable reports.** These are functional ones. Add correct NF ones.

-1

<u>Scenario-Based Modeling</u>

Use Case Scenarios:

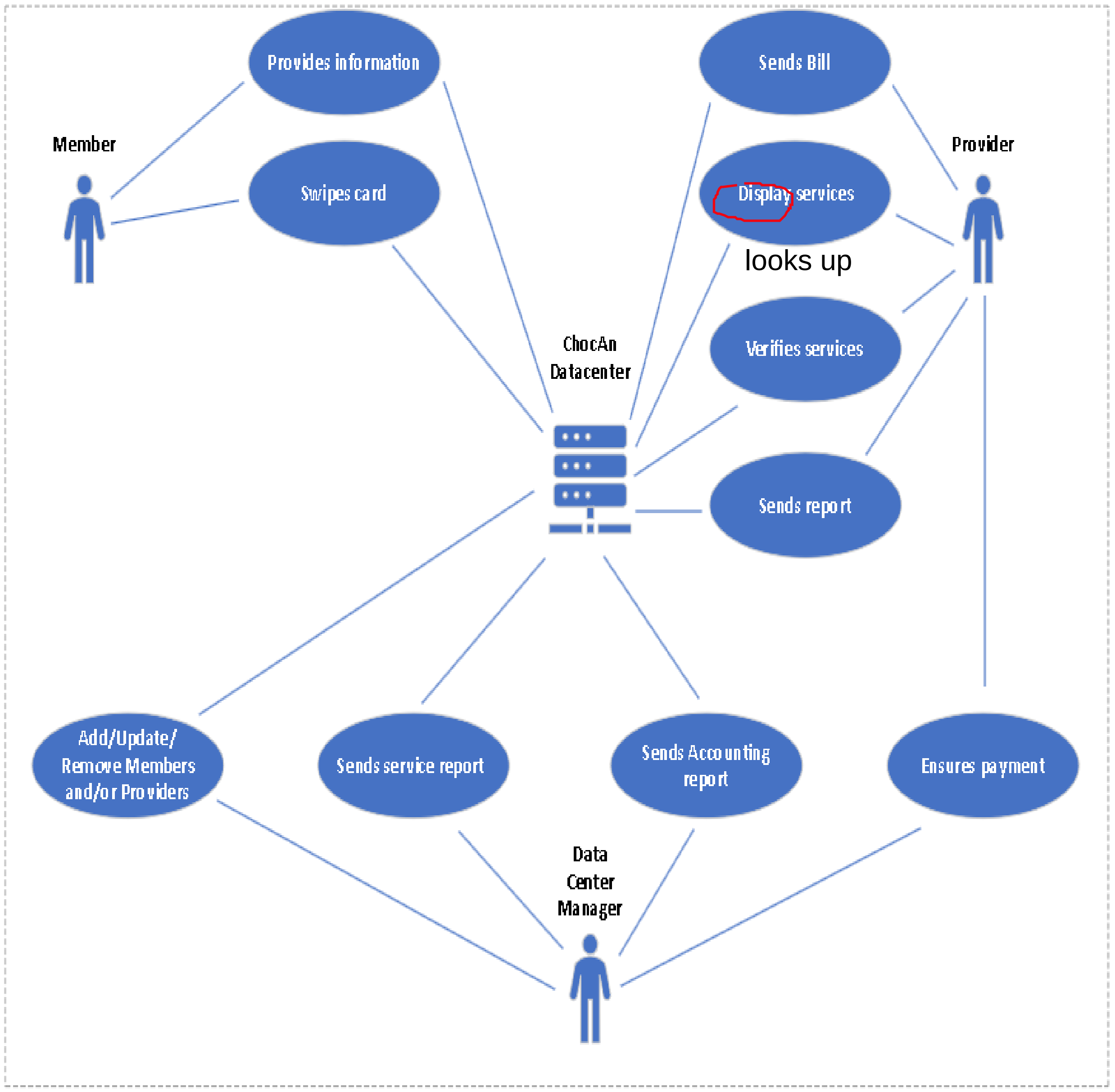
Actors for Scenarios: Member, Provider, ChocAn Datacenter Manager, ChocAn Datacenter

1. **ChocAn Datacenter Manager adds a new Member to the service.**
2. **ChocAn Datacenter Manager updates Member information if it has changed.**
3. **ChocAn Datacenter Manager removes Member no longer using the service.**
4. **ChocAn Datacenter Manager adds a new Provider to the service.**
5. **ChocAn Datacenter Manager updates Provider information if it has changed.**
6. **ChocAn Datacenter Manager removes Provider no longer using the service.**
7. **Member swipes membership card to validate membership for Provider.**
8. **Provider sends bill to ChocAn Datacenter.**
9. **ChocAn Datacenter displays services rendered and fee for the Provider on their terminal.**
10. **Provider verifies services rendered for ChocAn Datacenter.**
11. **Provider receives weekly report of services rendered from ChocAn Datacenter.**
12. **ChocAn Datacenter Manager receives weekly reports of services rendered by all Providers for the current week.**
13. **ChocAn Datacenter Manager receives a weekly report on Accounts Payable for the week.**
14. **ChocAn Datacenter Manager ensures each Provider's bank account is credited with the appropriate amount for services rendered for the week.**

A scenario is a description of detailed steps for ONE task (or use case). You put multi

Please rewrite them. -3

Use Case Diagram



Good!

Class-Based Modeling

Potential Classes:

- **Member**
- **Provider**
- **Datacenter Manager**
- **System(ChocAn Datacenter)**
- **Verification Display**
- **Weekly Accounts Payable**
- **Weekly Services Rendered**
- **Member Validation**
- **Master Password**
- **Telephone Number**

Accepted Classes:

- **Member**
- **Provider**
- **Datacenter Manager**
- **System(ChocAn Datacenter)**

Add classes of Service, Claim, Report, and so on, to hold the

too broad

Member Class:

- **Attributes: Name, ID_Number, Street, City, State, Zip_Code**
- **Functions: Pay_Fees()**

Provider Class:

Put these attributes and methods in a clas

- **Attributes: Name, ID_Number, Street, City, State, Zip_Code**
- **Functions: Swip_Member_Card(), Bill_Service(), Access_Provider_Directory(), Enter_Comments(), Verify_Services_Rendered()**

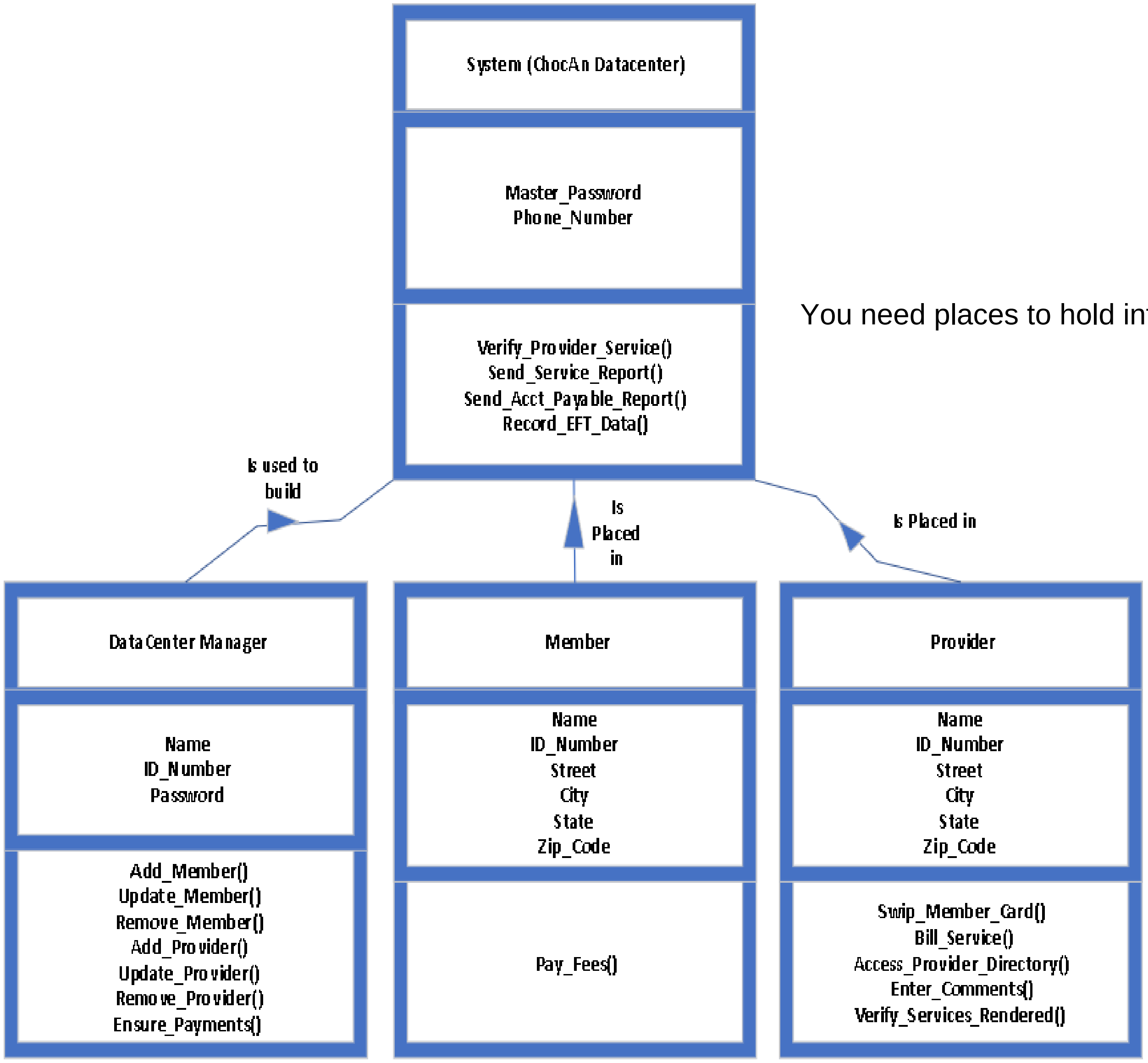
Datacenter Manager Class:

- **Attributes: Name, ID_Number, Password**
- **Functions: Add_Member(), Update_Member(), Remove_Member(), Add_Provider(), Update_Provider(), Remove_Provider(), Ensure_Payments()**

System(ChocAn Datacenter) Class:

- **Attributes: Master_Password, Phone_Number**
- **Functions: Verify_Provider_Service(), Send_Service_Report(), Send_Acct_Payable_Report(), Record_EFT_Data()**

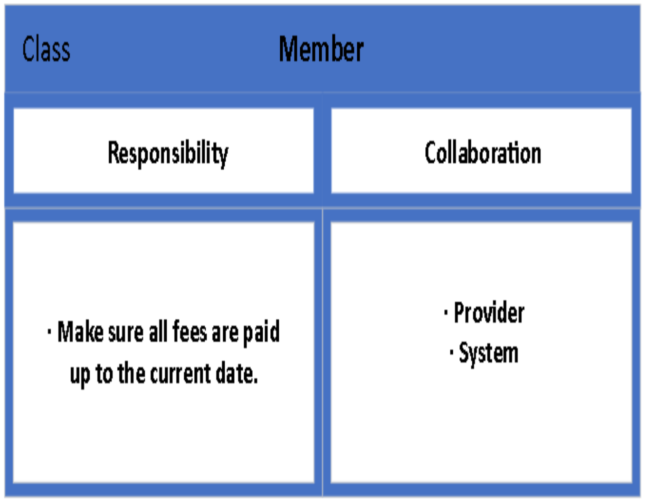
Class Diagram



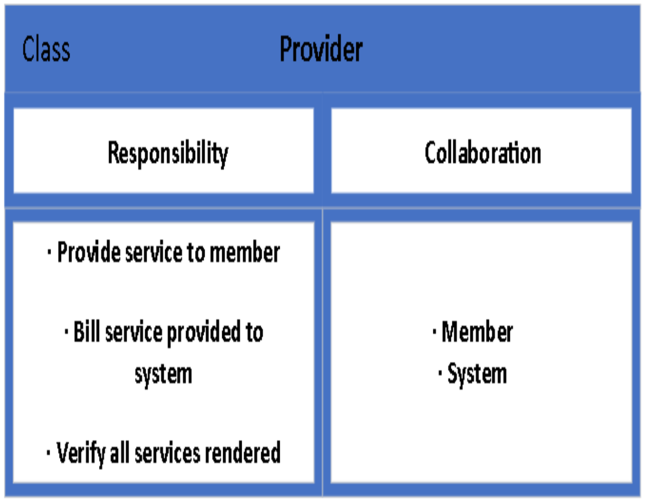
You need places to hold information a

-5

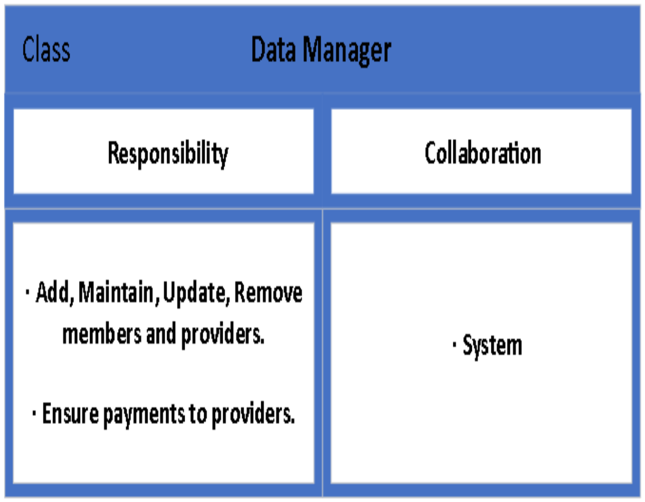
CRC Model Index Cards



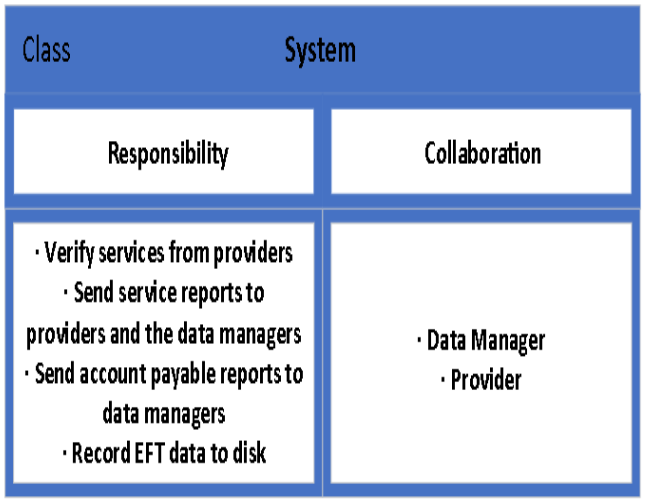
Class	Member
Responsibility	Collaboration
· Make sure all fees are paid up to the current date.	· Provider · System



Class	Provider
Responsibility	Collaboration
· Provide service to member · Bill service provided to system · Verify all services rendered	· Member · System



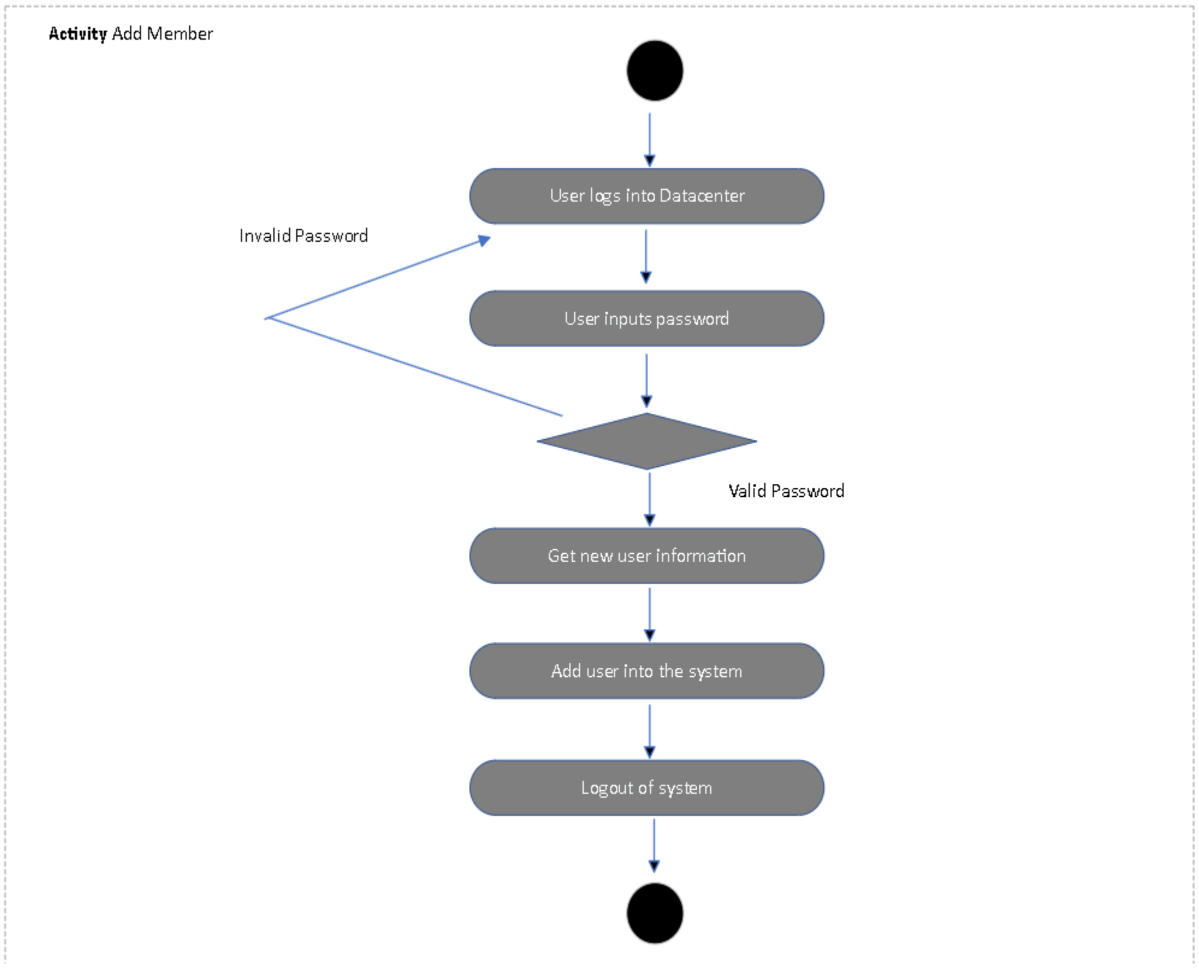
Class	Data Manager
Responsibility	Collaboration
· Add, Maintain, Update, Remove members and providers. · Ensure payments to providers.	· System

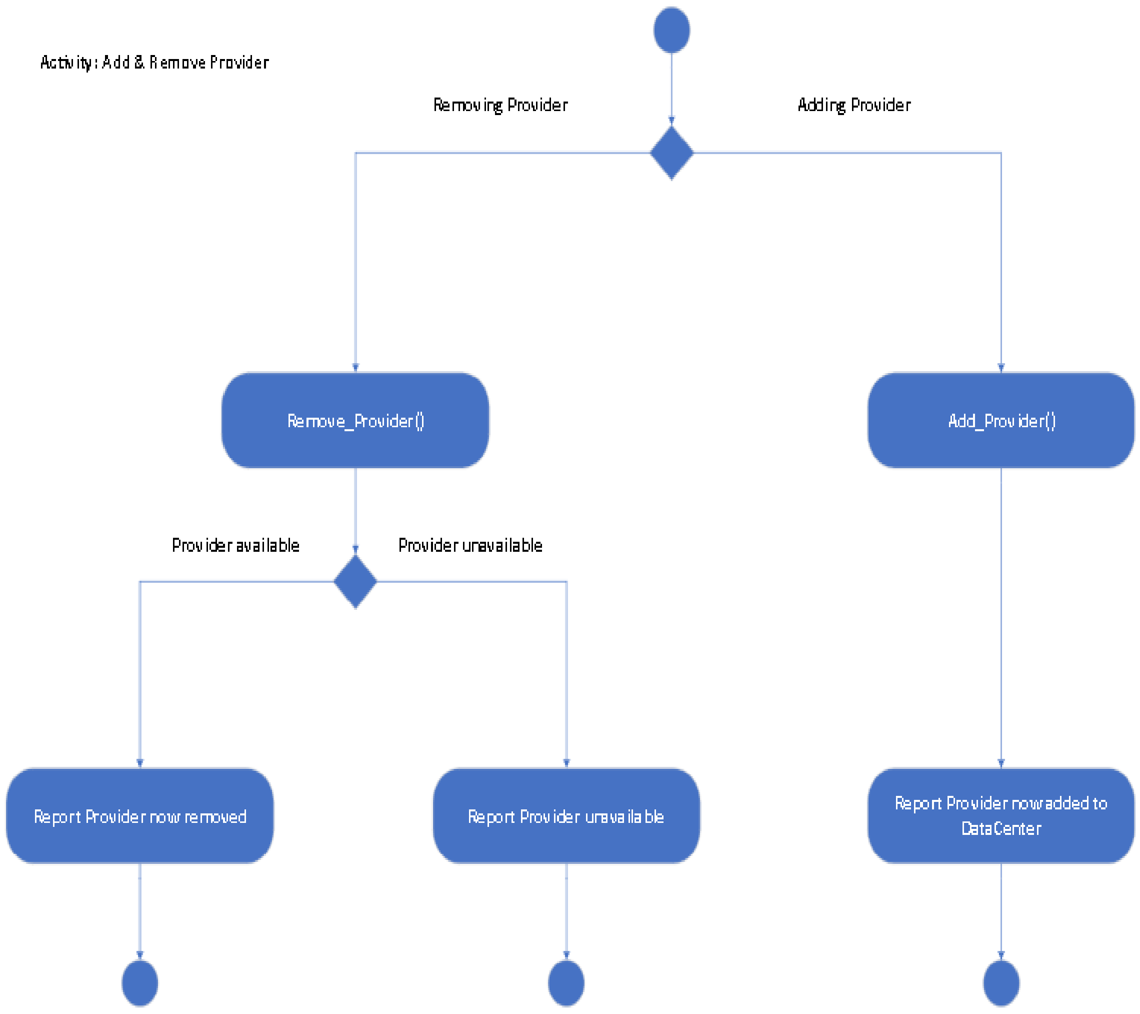


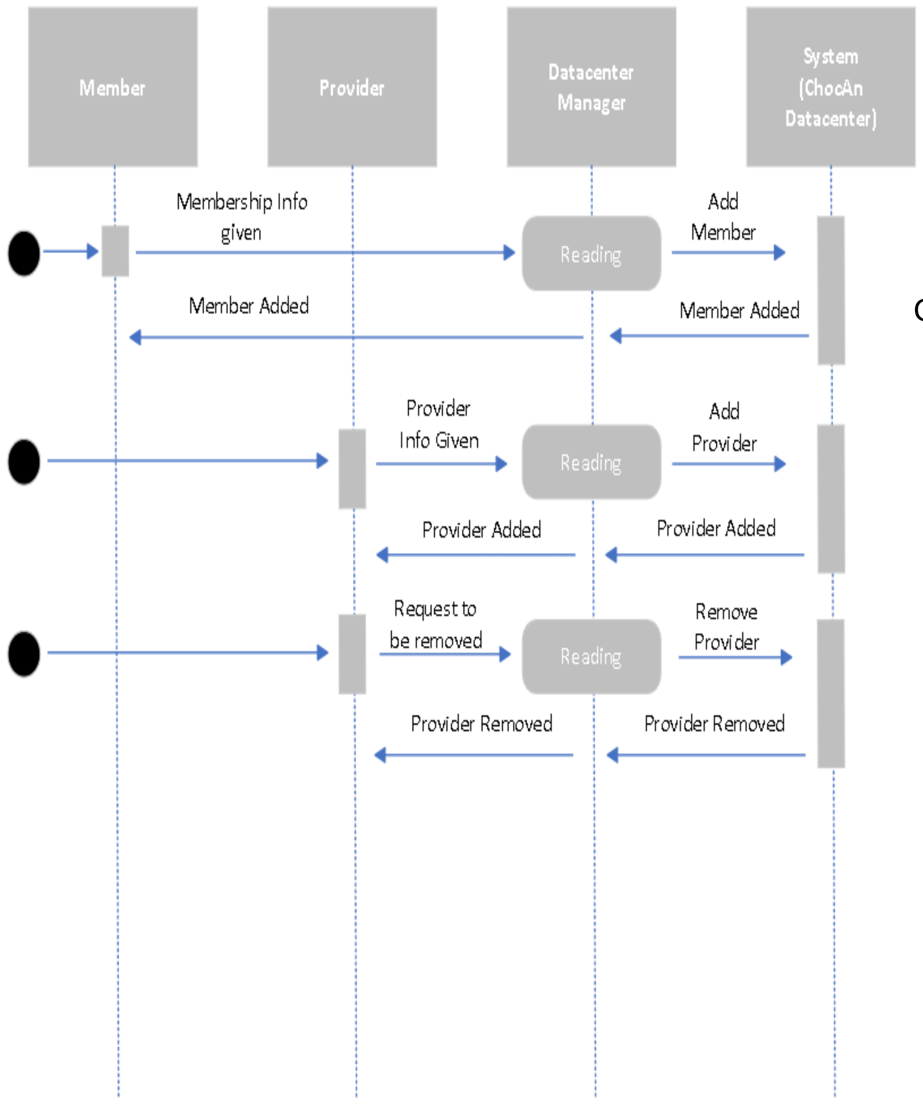
Class	System
Responsibility	Collaboration
· Verify services from providers · Send service reports to providers and the data managers · Send account payable reports to data managers · Record EFT data to disk	· Data Manager · Provider

Revise accordingly

Functional Modeling




Activity: Add & Remove Provider
Removing Provider
Adding Provider
Remove_Provider()
Add_Provider()
Provider available
Provider unavailable
Report Provider now removed
Report Provider unavailable
Report Provider now added to DataCenter



Ok

One sequence diagram for one use case. Need to show more sequence diagrams.

-5

Review Questions

1. **Is each requirement consistent with the overall objectives for the system or product?**

   The functional requirements are consistent with the system due to them being required for the system to function, and the nonfunctional requirements are consistent with the system because they improve the user experience for all participating parties.

2. **Have all requirements been specified at the proper level of abstraction?**

   The requirements are specific to the proper level of abstraction they should be at.

3. **Is the requirement really necessary or does it represent an add-on feature that may not be essential to the objective of the system?**

   The functional requirements we have are necessary for the system to properly function. The nonfunctional are not required for the system to function, but they do improve the overall user experience and ease of access to the system.

4. **Is each requirement bounded and unambiguous?**

   The requirements we defined are bounded in order to make a well-made final product. The requirements are unambiguous to avoid confusion in the coding and production of the final software.

5. **Does each requirement have attribution?**

   Yes, each of our requirements is carefully assigned to one of our actors to perform. Yes

6. **Do any requirements conflict with other requirements?**

   No, all our requirements are separated so they do not have overlapping functions in the final product.

7. **Is each requirement achievable in the technical environment that will house the system or product?**

   Yes, All the requirements will be achievable in the technical environment since they are made specifically for the product.

8. **Is each requirement testable, once implemented?**

   Yes, once the requirements are implemented the end user will be able to test them.

9. **Does the requirements model properly reflect the information, function, and behavior of the system to be built?**

The requirements model does reflect the information, function, and behavior of the system to be built by giving clear functional and nonfunctional requirements to build the software to work and have a good user-based experience.

10. **Has the requirements model been “partitioned” in a way that progressively exposes more detailed information about the system?**

    Yes, as our requirements model has been further and further partitioned more detailed information about the system is exposed.

11. **Have requirements patterns been used to simplify the requirements model?**

    As we have reviewed our requirements model the requirements pattern has made it easy and clear for anyone to read over and work from. The requirements patterns we have established are based off the needs of the final product for the product to be successful.

Work Log

- **CSC 470 Group Meeting #1, Sept4th, 2022**
  - Facilitator: Bryce
  - Recorder: Edward
  - Moderator: Cameron
  - Team Meetings: Friday and Saturdays 8:00 pm meet time.
  - UML: Microsoft Visio
  - Followed each other on GitHub.
- **CSC 470 Group Meeting #2, Sept. 9th, 2022**
  - Facilitator: Bryce
  - Recorder: Edward
  - Summarizer: Cameron
  - Going over part 1 Requirements elicitation for the CAIS system.
  - Differentiating Functional Requirements from Non-Functional Requirements, and then prioritizing them.
  - Next meeting work on Scenario Based Modeling portion of the project.
- **Group Meeting #3 CSC 470, Sept. 11th, 2022**
  - Facilitator: Bryce
  - Recorder: Edward
  - Moderator: Cameron
  - Began and finished Scenario-Based Modeling Objective.
- **Group Meeting #4 CSC 470 September 16th, 2022**
  - Facilitator: Cameron
  - Recorder: Bryce
  - Moderator: Edward
  - Reworked Part 1 and Part 2 of the project by correcting our functional and nonfunctional requirements, making more use case scenarios, and changing our diagram to a use case diagram with a UML.
- **Group Meeting #5 CSC 470 September 17th, 2022**
  - Facilitator: Cameron
  - Recorder: Bryce
  - Moderator: Edward
  - Worked on Part 3 of the project class-based modeling identified analysis classes, made a class diagram, made CRC model index cards.
- **Group Meeting #6 CSC 470 September 18th, 2022**
  - Facilitator: Cameron
  - Recorder: Bryce
  - Moderator: Edward

- Worked on part 4 Functional Modeling, made activity diagrams for use cases, and made a sequential diagram for the use cases. Next meeting will be September 19th to answer review questions and add work log to final document.

- **Group Meeting #7 CSC 470 September 19th, 2022**
    - Facilitator: Edward
    - Recorder: Bryce
    - Moderator: Cameron
    - For our final meeting we revised the project where advise was given by our professor. We then worked on the review questions as a group. The last thing we had to do was include our work log for the final touches to our project.

Good job, especially in developing functional requirements and use cases. However, the c