LAKEHEAD UNIVERSITY

FACULTY OF ENGINEERING

----- ꕥ 🕮 ꕥ -----

TERM PROJECT 1

***Topic*: Chocoholics Anonymous**

Instructor: **Dr. Ayman Diyab**

Lab Instructor: **Mr. Mohammed AbuFoul**

Student group 2:

Number	Full name	Student ID
1	Joshua Whitlock	1272349
2	Thanh Vu Nguyen	1244225
3	Kenneth Shahi	1261283

Thunder Bay, 2025

TABLE OF CONTENTS

ACKNOWLEDGEMENT

Acknowledgement is given to Dr. Ayman Diyab for providing slides and required knowledge during lectures for the Term Project and to Mr. Mohammed AbuFoul for explanation and guides during laboratory session.

ASSIGNMENT QUESTIONS

1. Question 1.19

Question: Suppose that the product for Chocoholics Anonymous of Appendix A has been implemented exactly as described. Now the product has to be modified to include endocrinologists as providers. In what ways will the existing product have to be changed? Would it be better to discard everything and start again from scratch?

Answer:

To integrate endocrinologists into the ChocAn system, the primary change would be updating the database to include new service codes, names, and fees for the treatments they provide. The provider directory (both backend and frontend) may also need to be updated so that providers can view these new services, and the validation system must recognize and accept the new codes when services are entered. If the system already gets this data from the database dynamically, then the change would be simple because the software will automatically show the new information. However, if the terminal is hardcoded with services, some refactoring might be required, but there is no need to start all over again from scratch. Even if rebuilding the system is considered an option, if the product has already been launched, a significant amount of time (developing, building, testing, launching/marketing) and money (hiring employees, paying for software and database service subscriptions, planning new marketing campaigns, and launching them) are required.

2. Question 2.22

Question: Which software life-cycle model would you use for the Chocoholics Anonymous product described in Appendix A? Give your reasons for your answer.

Answer:

The best software life-cycle model for the ChocAn system would be an iterative and incremental model, meaning we would build the system step by step, prioritize critical features or tasks first, and handle less important ones later, test each part, and refine based on the client's response until the final product is satisfactory. Since ChocAn connects to external systems like Acme Accounting and the bank's EFT system, flexibility is required to handle new requirements or updates without breaking existing functions while maintaining a high level of security, auditability, and data privacy. With this approach, developers can start with core features like validating members and generating reports, then tackle new ones like updating providers and processing payments later. The team can consistently maintain product quality while promptly addressing issues and releasing updates based on user feedback. These advantages of the iterative and incremental model make the system easier to maintain, evolve, and reliable and functional throughout the development process.

3. Question 3.15

Question: What differences would you expect to find if the Chocoholics anonymous product of Appendix A were developed by an organization at CMM level 1, as opposed to an organization at level 5?

Answer:

If ChocAn were made by a company at CMM Level 1, the project would be unorganized, rushed, and full of errors. Work would be unpredictable and testing would be weak, causing problems like incorrect payments or missing reports. A CMM Level 5 company would have a planned and controlled process with strict quality checks, proper testing, and continuous improvement. The result would be a reliable and accurate system that is easier to maintain and trust.

4. Question 11.24

Question: Perform the requirements workflow for the Chocoholics Anonymous project in Appendix A.

Answer:

4.1. Outline

1. Develop an initial understanding of the target domain:
   - Research basic information about ChocoAn's purpose, services, target market.
   - Build a glossary of terms based on information found during the initial research.
   - Refine the glossary as needed, like when the team encounters a new technical term.
2. Build Business Model:
   - Create UML diagrams to represent the client's process and system structure.
   - Include the use case diagrams to show interaction with the user and system.
   - Include the use case description diagrams to examine relationships between data.
3. Iterate the UML/Logic:
   - Review and improve diagrams through multiple iterations: Perform this by re-reading the Appendix entry and walking through our documentations.
   - Check whether the final model matches all client needs before final approval.
4. Define Requirements:
   - List what the system should do (functional) like validating members or generating reports.
   - List how the system should perform (non-functional) like response time, security and reliability.
5. Testing and validation:
   - Test the system/logic against the requirements.
   - Check if the model behaves correctly in different types of cases (valid and invalid inputs).

4.2. Glossary

Term	**Meaning**
Accounts Payable	Amounts of money that ChocAn must pay to providers for services rendered to ChocAn members.
Acme Accounting Services	An independent, external organization (not affiliated with ChocAn) who conducts financial operations like suspending/reinstating members of ChocAn and handling payment records from these members.
Card Reader	A device that reads member card and send the data on the card to the system/terminal
ChocAn Data Center	The remote database responsible for tracking transactions, tracking and reporting member status. Central system where member, provider, and service records are stored and reports are processed.
Chocoholics Anonymous (ChocAn)	An organization established to assist people with chocolate addition in all its glorious forms.
Chocolate	Produced from roasted and ground cocoa beans, this energy-dense food exists in many forms and can be directly consumed or used to make other products/food.
Electronic Funds Transfer	A method of electronically transferring funds between accounts (e.g. Chocoholics Anonymous banking account to provider's banking account).
Invalid (member number)	A member number that is reported as not found in the ChocAn Data Center database.
Member	An individual who pays a monthly fee to ChocAn for which they are entitled to unlimited consultations and treatments with health care professionals. Possess a Member Card.
Member Card	A magnetic stripe card with member data like name and nine-digit member number engraved in the front and the same data stored in a black magnetic stripe in the back.
Member Status	The status of a member who registered with ChocAn. "Valid" means that the member has paid all their service fees, while "Suspended" indicates that the member still owes service fees.
Monthly Fee	The amount paid every month to maintain membership with ChocAn.
Providers	A health care professional responsible for providing treatment to validated ChocAn

	members. Mainly dietitians, internists, and exercise experts. Receive payment from ChocAn.
Provider Directory	A catalog or list of all possible services, their codes, and fees; like a menu of treatments for members.
Services Rendered	Services provided by ChocAn providers to members that match a six-digit code listed in the Provider Directory.
Terminal	The hardware responsible for running the software.
Valid (member number)	A member number belonging to a ChocAn Member who is currently with their monthly fee.

Table 1: Term Glossary

To appropriately determine the definition of terminal/card reader we need to know if the card reader is bundled with the terminal or sold separately. This will be important for determining how we receive the card number from the terminal.

We need a way to actually conduct a local transaction in our UML diagram, something like "record service" that stores the member number, time, service code on the terminal.

4.3. Diagram

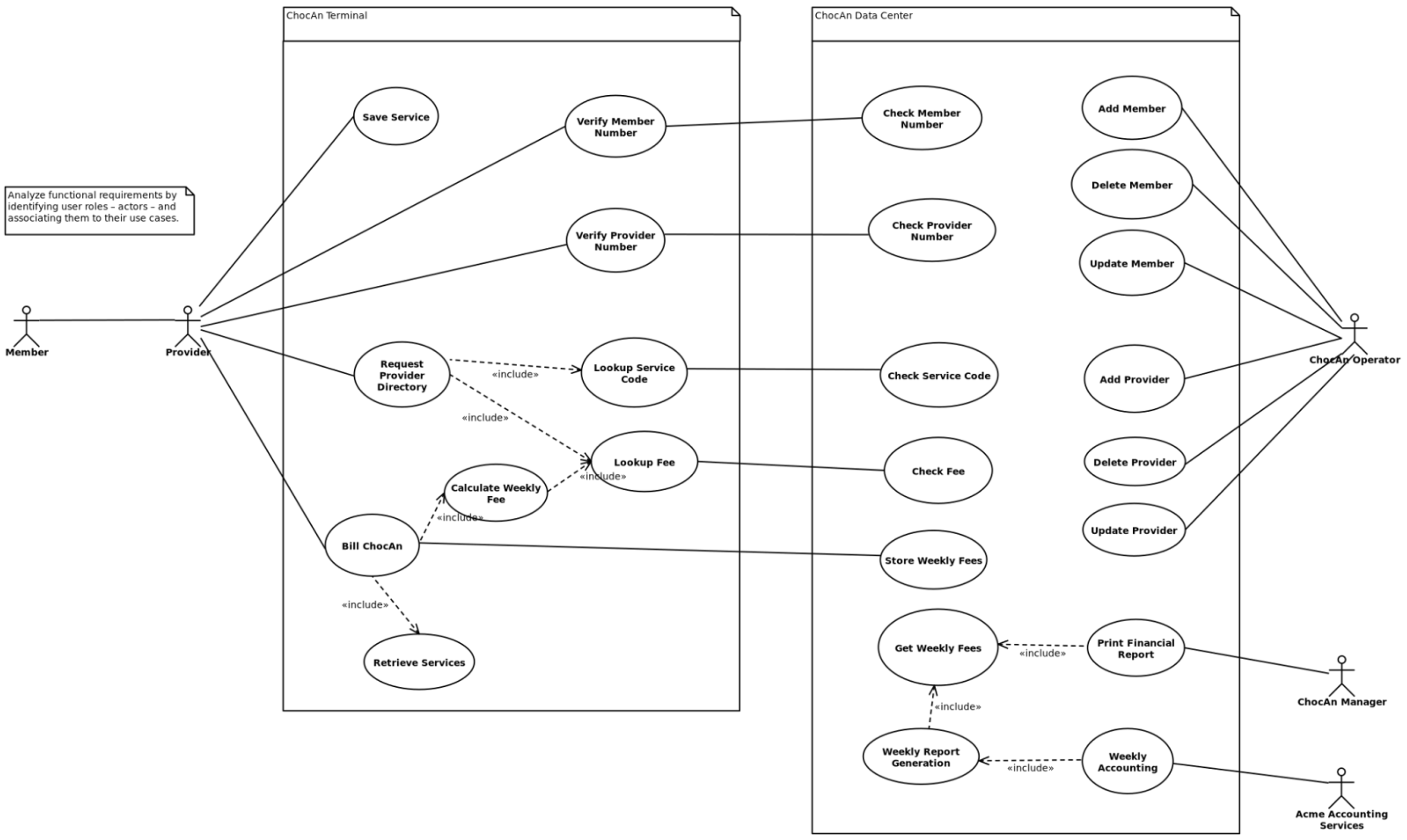

Figure 1: Diagram for ChocAn

4.4. Defining Requirements

Functional Requirements (what the system must do)

1. Check Member:
    - The provider types in a 9-digit member number.
    - The system checks the number and shows one of these: Validated, Invalid number or Member suspended.
2. Record a Service:
    - After the member is confirmed, the provider enters:
    - Date the service was given following ISO 8601 format (YYYY-MM-DD).
    - Provider number (9 digits).
    - Member number (9 digits).
    - Service code (6 digits).
    - Comment if needed (up to 100 letters).
    - The system saves this record with today's date and time following ISO 8601 format (YYYY-MM-DDTHH:MM:SSZ).
3. Find a Service Code:
    - The provider looks in the Provider Directory for the right six-digit code.
    - The system shows the name of the service so the provider can check it.

    If the code is wrong the system shows an error message.
4. Show the Fee: The system looks up the fee for the chosen service and shows it on the screen.
5. Weekly Provider Report:
    - Every Friday at midnight, the system makes a report for each provider who gave services that week.

- The report lists the provider's name, number, address and all services done that week with the date, member name, member number, code, fee, total visits and total amount.

6. Weekly Member Report:
   - Each member who had a service that week gets a report.
   - It lists the services in date order and shows the date, provider name and service name, plus the member's name, number and address.
7. EFT File for Payment: The system makes a payment file that shows the provider's name, number and total amount to be paid.
8. Summary Report:
   - The system makes a summary for the accounts manager.
   - It lists every provider, how many visits they had, each provider's total and the overall totals for the week.
9. Update Member and Provider Info
   - Staff at the Data Center can add, delete or update member records.
   - They can also add, delete or update provider records.
10. Provider Directory File
    - A provider can ask for a list of all services with their six-digit codes and fees
    - The system saves this list as a file

<ins>Non-Functional Requirements</ins> (how the system should behave)

1. Time
   - The main accounting job runs every Friday at 12 am
   - Reports can also be made any time if needed
2. Data Rules and Formats
   - Member number – 9 digits

- Provider number – 9 digits
- Service code – 6 digits
- Member name – 25 letters
- Provider name – 25 letters
- Comments – up to 100 letters
- Fee per service – max $999.99
- Total weekly fee – max $99 999.99
- Use ISO 8601 format YYYY-MM-DDTHH:MM:SSZ

3. Simulation and Limits

- Other companies handle the terminals, EFT system and fee payments
- This system only uses the keyboard for input and shows results on the screen
- Reports and directories are saved as files not actually sent by email
- The ETF file only needs the provider's name, provider number and amount to pay

4.5. Use Case Description

4.5.1. Verify Member Number

Brief Description

The `Verify Member Number` use case enables providers to ensure that a person's ChocAn membership is currently valid.

Step-by-Step Description

1. Send a query to the ChocAn data center containing a member number.
2. Receive a response of either `valid`, `member suspended`, or `invalid`.
3. Print the response from the ChocAn data center.

4.5.2. Verify Provider Number

Brief Description

The `Verify Provider Number` use case verifies providers are registered with the ChocAn system for proper accounting.

Step-by-Step Description

1. Run once when the provider's terminal is powered on.
2. Send a request to the ChocAn data center containing the provider's number.
3. Print the return value of `valid`, or `invalid`.
4. If `invalid`, re-prompt for the `providerId`.

4.5.3. Request Provider Directory

Brief Description

The `Request Provider Directory` use case allows the provider to update and view all services and their service numbers and associated fees.

Step-by-Step Description

1. Get a list of `serviceName` and `serviceCodes` by calling the `Lookup Service Code` use case.
2. Get a list of associated fees by calling the `Lookup Fee` use case with a list of `serviceCodes`.
3. Send an e-mail from the provider's terminal to the provider containing a list of `serviceNames`, `serviceCodes`, and their associated serviceFees.

4.5.4. Lookup Service Code

Brief Description

The `Lookup Service Code` use case calls from the terminal to the ChocAn Data center to retrieve an updated list of service codes and their accompanying descriptions.

Step-by-Step Description

1. Take a request containing a service code or get a service list.
2. Send a request to the ChocAn data center requesting service code validation or the table containing `serviceNames` and `serviceCodes`.
3. Return either service valid or the list of `serviceNames` and `serviceCodes`.

4.5.5. Lookup Fee

Brief Description

The `Lookup Fee` use case takes a service code and looks up its associated fee.

Step-by-Step Description

1. Receive a request containing a `serviceCode` or list of `serviceCodes`.
2. Use the `Check Fee` use case to retrieve the associated fees from the ChocAn database.
3. Return the `serviceCode` or `serviceCodes` and their `serviceFees`.

4.5.6. Bill ChocAn

Brief Description

The `Bill ChocAn` use case runs weekly from the provider's terminals to send a bill to the ChocAn database containing all the services, their codes, and associated fees, as well as the provider's number.

Step-by-Step Description

1. Get a list of `serviceCodes` for the week by utilizing the `Retrieve Services` use case.
2. Sends the list of `serviceCodes` retrieved in step 1 to the `Calculate Weekly Fee` use case.
3. Receive a total fee and a list of `serviceCodes` and `serviceFees` from the Calculate Weekly Fee use case.
4. Send the `totalFee`, and list of `serviceCodes`, `dateOfServices`, `providerName`, `memberNames`, `memberNumbers`, and `serviceFees`, `totalConsultations`, to the ChocAn Data center.

4.5.7. Calculate Weekly Fee

Brief Description

The `Calculate Weekly Fee` use case takes the list of completed service codes on the provider's terminal for the week and totals the fees for them.

Step-by-Step Description

1. Receive a list of `serviceCodes` from the `Bill ChocAn` use case.
2. Use the `Lookup Fee` use case to get `serviceFees` for each `serviceCode`.
3. Total the `serviceFees` from the `Lookup Fee` use case in step 2.
4. Returns the `totalFee`, `serviceFees`, and their associated `serviceCodes`.

4.5.8. Check Member Number

Brief Description

The `Check Member Number` use case takes a member number, queries the database and returns if it's valid or invalid.

Step-by-Step Description

1. Receives a member number from the `Verify Member Number` use case.
2. Queries the ChocAn database with the member number received from the `Verify Member Number` use case.
3. Returns the current member status.

4.5.9. Check Provider Number

Brief Description

The `Check Provider Number` use case takes a provider number, queries the database and returns valid or invalid.

Step-by-Step Description

1. Receives a member number from the `Verify Provider Number` use case.
2. Queries the ChocAn database with the provider number.
3. Returns either `valid` or `invalid`.

4.5.10. Check Service Code

Brief Description

The `Check Service Code` use case queries the server for all currently available service codes and their associated descriptions, and returns them to the terminal.

Step-by-Step Description

1. Receives a list of service codes from the `Lookup Service Code` use case.
2. Queries the ChocAn database with the code or list of service codes.
3. Returns the `serviceCodes` and their associated `serviceNames`.

4.5.11. Check Fee

Brief Description

The `Check Fee` use case takes a list of service codes and returns their associated fees.

Step-by-Step Description

1. Receives a list of `serviceCodes` from the `Lookup Fee` use case.
2. Queries the ChocAn database with the `serviceCodes` received in step 1.
3. Returns the `serviceCodes` and their associated `serviceFees`.

4.5.12. Store Weekly Fees

Brief Description

The `Store Weekly Fees` use case takes a list of services and their associated fees and fee total from a provider for the week.

Step-by-Step Description

1. Receives a list of `serviceCodes`, `serviceFees`, and a `totalFee` from the `Bill ChocAn` use case.
2. Stores the list in the ChocAn database.

4.5.13. Weekly Report Generation

Brief Description

The `Weekly Report Generation` gathers all the provider's fees from the past week that are currently in the db and totals them up.

Step-by-Step Description

1. Receives a request from the `Weekly Accounting` use case with a range of dates.
2. Sends a request to the `Get Weekly Fees` use case to retrieve all fees that were delivered in the range of dates provided in step 1.
3. Receives a response from the `Get Weekly Fees` use case containing a list of dates and total fees in that date range.
4. Generates a report containing all of the weekly fees for the week, the total of weekly fees.
5. Returns the total of weekly fees to the `Weekly Accounting` use case.

4.5.14. Weekly Accounting

Brief Description

The `Weekly Accounting` use case takes the weekly fees from the weekly report generation and sends them to Acme Accounting Services.

Step-by-Step Description

1. Runs on a schedule every Friday at 12am.
2. Sends a request to the `Weekly Report Generation` use case.
3. Receives a total of weekly fees from the `Weekly Report Generation` use case.
4. Sends the total of weekly fees to Acme Accounting.

4.5.14. Print Financial Report

Brief Description

The `Print Financial Report` use case gets the weekly fees in the database for the week and prints them.

Step-by-Step Description

1. Sends a request containing the date range from last Friday at 12am to present to the `Get Weekly Fees` use case.
2. Receives a response from the `Get Weekly Fees` use case containing a list of all the weekly fees currently in the ChocAn database in the date range.
3. Prints a Financial Report containing all the Weekly Fees and their associated provider numbers.

4.5.15. Add Provider

Brief Description

The `Add Provider` use case allows a ChocAn operator to add a provider to the ChocAn database.

Step-by-Step Description

1. Receives a provider number, and a provider name from a ChocAn operator.
2. Adds the provider to the ChocAn database.

4.5.16. Delete Provider

Brief Description

The `Delete Provider` use case allows a ChocAn operator to remove a provider from the ChocAn database.

Step-by-Step Description

1. Receives a provider number from a ChocAn operator.
2. Removes the provider from the ChocAn database.

4.5.17. Update Provider

Brief Description

The `Update Provider` use case allows a ChocAn operator to change details about a provider.

Step-by-Step Description

1. Receives a provider number and optionally a provider name, street address, city, state, zip to update for the given provider number.
2. Updates the database entry for the provider number with the new information.

4.5.18. Add Member

Brief Description

The `Add Member` use case allows a ChocAn operator to add a new member to the ChocAn database.

Step-by-Step Description

1. Receives a member number and member name from a ChocAn operator.
2. Adds the member number and associated member name to the ChocAn database.

4.5.19. Add Member

Brief Description

The `Delete Member` use case allows a ChocAn operator to remove a member from the ChocAn database.

Step-by-Step Description

1. Receives a member number from a ChocAn operator.
2. Removes the member associated with the member number from the ChocAn database.

4.5.20. Update Member

Brief Description

The `Update Member` use case allows a ChocAn operator to edit details about a member in the ChocAn database.

Step-by-Step Description

1. Receives a member number and optionally a member name, member street address, member city, member state, member zip from a ChocAn operator.
2. Updates the database entry for the given member number with any optionally included entries from step 1.

4.5.21. Get Weekly Fees

Brief Description

The `Get Weekly Fees` use case gets the fees from the DB for the current or previous week.

Step-by-Step Description

1. Receives a range of dates.
2. Queries the ChocAn database for all `totalFees` in the range of dates.
3. Returns the list of `totalFees` for the given date range.

4.5.22. Retrieve Services

Brief Description

The `Retrieve Services` use case fetches the services the provider has rendered in the past week.

Step-by-Step Description

1. Receives a request from the `Bill ChocAn` use case for a list of services.
2. Queries the terminal for all `unsubmitted` services.
3. Returns a list of all `unsubmitted` services, their associated `serviceIds`.
4. Marks the services in the terminal as `submitted`.

4.5.23. Save Service

Brief Description

The `Save Service` use case allows a provider to store the services they render on their terminal, to be retrieved and uploaded to the ChocAn data center.

Step-by-Step Description

1. Receives a service from a provider containing `dateOfService`, `providerName` and `serviceName`.
2. Stores the data in the terminal as unsubmitted.

REFERENECES

[1] Schach, Stephen R, Object-Oriented and Classical Software Engineering, 8th ed (McGraw Hill, 2010)