LAKEHEAD UNIVERSITY

FACULTY OF ENGINEERING

Requirements

For the ChocAn Project

Instructor:	Dr. Ayman Diyab
Lab Instructor:	Mr. Mohammed AbuFoul

North Light Software

Full name	**Student ID**
Joshua Whitlock	1272349
Thanh Vu Nguyen	1244225
Kenneth Shahi	1261283

Thunder Bay, 2025

Contents

List of Figures

List of Tables

1 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.

2 Workflow Steps

The workflow was derived from *The Unified Process*[Schach2010].

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).

3 Additional Considerations

3.1 Formatting

Inline code, meant to denote potential classes, methods, and attributes, is formatted through this document. `This is an example of inline code formatting used in this document.`

3.2 Tool Use

Gaphor was used for the creation of UML images. GitHub was utilized for collaboration and version tracking and management. Pandoc was used in the creation of the documentation thus far. Other tools used include Google Drive/Docs for collabaration, as well as Microsoft Office for formatting and docment generation.

3.3 Communication

The group has communicated primarliy through e-mail. Discord and in-person communication have also been utilized. Google drive was utilized initally for organizing documentation before a proper Git repository was created and access was distributed to each member.

4 *Glossary*

Table 1: A glossary of terms to better understand the ChocAn business model.

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.
Invalid (member number)	A member number that is reported as not found in the ChocAn Data Center database.
Electronic Funds Transfer	A method of electronically transferring funds between accounts (e... Chocoholics Anonymous banking account to provider's banking account).
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.
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.
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.
Suspended (member state)	Suspended indicates that the fees haven't been paid in at least one month. Tracked by Acme Accounting Services.
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.

5 Interview

The interview process was skipped as there was not practical way to interview the client (Appendix A). In lieu of an interview a rigorous examination of the appendix was conducted.

6 UML Use Case Diagram

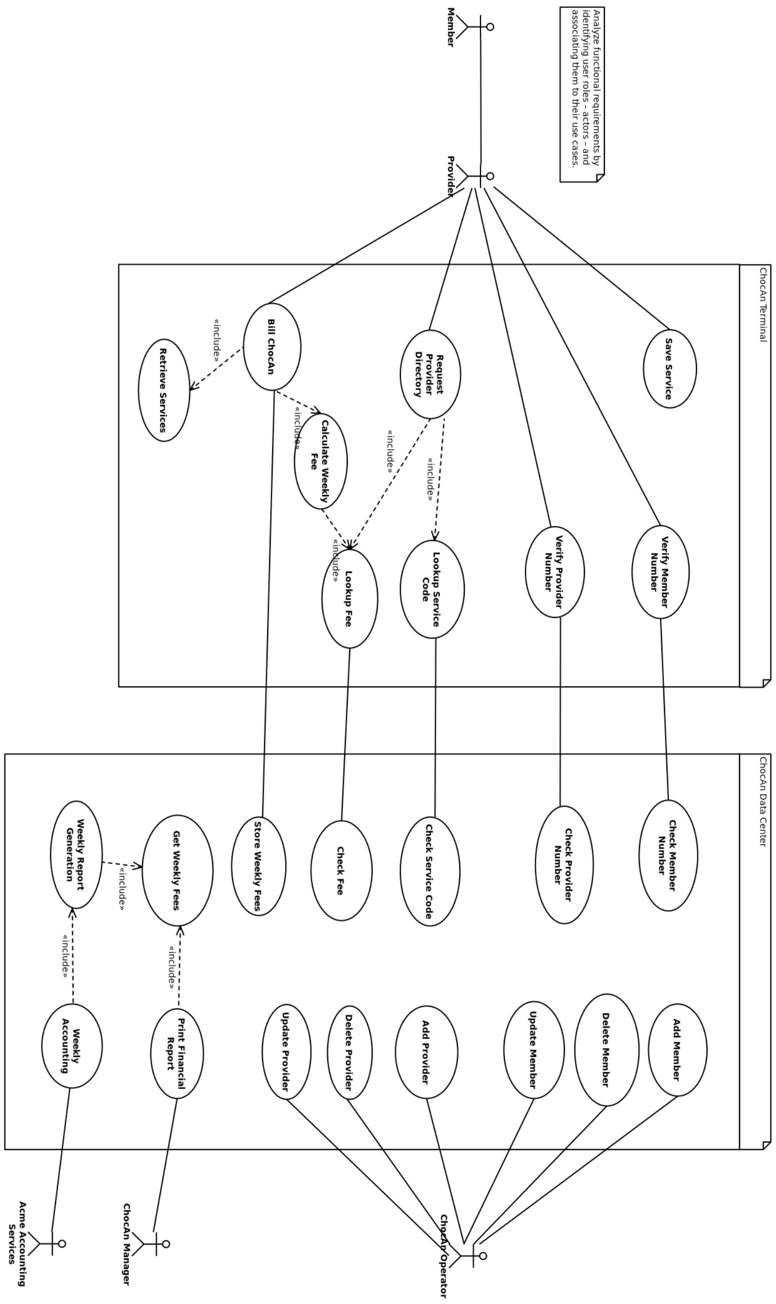

Figure 1: Use Case Diagram

7 Defining Requirements (Functional and Non-Functional)

7.1 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 (MM-DD-YYYY)
     - 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 (MM-DD-YYYY HH:MM:SS)
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 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.

7.2 Non-Functional Requirements (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 date format MM-DD-YYYY and time HH:MM:SS
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 name, provider number and amount to pay.

8 Use-Case Descriptions

Table 2: The Verify Member Number use case description.

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
Sends a query to the ChocAn data center containing a member number. Receives a response of either valid, member suspended, or invalid. Prints the response from the ChocAn data center.

Table 3: The Verify Provider Number use case description.

Brief Description
The `Verify Provider Number` use case verifies providers are registered with the ChocAn system for proper accounting.
Step-by-Step Description
Runs once when the provider's terminal is powered on. Sends a request to the ChocAn data center containing the provider's number. Prints the return value of valid, or invalid. If invalid, re-prompts for the providerId.

Table 4: The Request Provider Directory use case description.

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
Gets a list of serviceName and serviceCodes by calling the Lookup Service Code use case. Gets a list of associated fees by calling the Lookup Fee use case with a list of service Codes. Sends an e-mail from the provider's terminal to the provider containing a list of serviceNames, serviceCodes, and their associated serviceFees.

Table 5: The Lookup Service Code use case description.

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
Take a request containing a service code or get a service list. Sends a request to the ChocAn data center requesting service code validation or the table containing serviceNames and serviceCodes. Returns either service valid or the list of serviceNames and serviceCodes.

Table 6: The Lookup Fee use case description.

Brief Description
The `Lookup Fee` use case takes a service code and looks up its associated fee.
Step-by-Step Description
Receives a request containing a serviceCode or list of serviceCodes. Uses the Check Fee use case to retrieve the associated fees from the ChocAn database. Returns the serviceCode or serviceCodes and their serviceFees.

Table 7: The Bill ChocAn use case description.

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** Gets a list of serviceCodes, `dateOfService`, and `providerName` for the week by utilizing the Retrieve Services use case. Sends the list of serviceCodes retrieved in step 1 to the Calculate Weekly Fee use case. Receives a total fee and a list of serviceCodes and serviceFees from the Calculate Weekly Fee use case. Sends the `totalFee`, and list of serviceCodes, `dateOfServices`, `providerName`, `memberNames`, `memberNumbers`, and serviceFees, `totalConsultations`, to the ChocAn Data center.

Table 8: The Calculate Weekly Fee use case description.

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** Receives a list of serviceCodes from the Bill ChocAn use case. Uses the Lookup Fee use case to get serviceFees for each serviceCode. Total the serviceFees from the Lookup Fee use case in step 2. Returns the totalFee, serviceFees, and their associated serviceCodes.

Table 9: The Check Member Number use case description.

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

Table 10: The Check Provider Number use case description.

Brief Description
The `Check Provider Number` use case takes a provider number, queries the database and returns valid or invalid. **Step-by-Step Description** Receives a member number from the Verify Provider Number use case. Queries the ChocAn database with the provider number. Returns either valid or invalid.

Table 11: The Check Service Code use case description.

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

Table 12: The Check Fee use case description.

Brief Description
The `Check Fee` use case takes a list of service codes and returns their associated fees.
Step-by-Step Description
Receives a list of serviceCodes from the Lookup Fee use case. Queries the ChocAn database with the serviceCodes received in step 1. Returns the serviceCodes and their associated serviceFees.

Table 13: The Store Weekly Fees use case description.

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

Table 14: The Weekly Report Generation use case description.

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
Receives a request from the `Weekly Accounting` use case with a range of dates. 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. Receives a response from the `Get Weekly Fees` use case containing a list of dates and total fees in that date range. Generates a report containing all of the weekly fees for the week, the total of weekly fees. Returns the total of weekly fees to the `Weekly Accounting` use case.

Table 15: The Weekly Accounting use case description.

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

Table 16: The Print Financial Report use case description.

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
Sends a request containing the date range from last Friday at 12am to present to the `Get Weekly Fees` use case. 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. Prints a Financial Report containing all the Weekly Fees and their associated provider numbers.

Table 17: The Add Provider use case description.

Brief Description
The `Add Provider` use case allows a ChocAn operator to add a provider to the ChocAn database.
Step-by-Step Description
Receives a provider number, and a provider name from a ChocAn operator. Adds the provider to the ChocAn database.

Table 18: The Delte Provider use case description.

Brief Description
The `Delete Provider` use case allows a ChocAn operator to remove a provider from the ChocAn database.
Step-by-Step Description
Receives a provider number from a ChocAn operator. Removes the provider from the ChocAn database.

Table 19: The Update Provider use case description.

Brief Description
The `Update Provider` use case allows a ChocAn operator to change details about a provider.
Step-by-Step Description
Receives a provider number and optionally a provider name, street address, city, state, zip to update for the given provider number. Updates the database entry for the provider number with the new information.

Table 20: The Add Member use case description.

Brief Description
The `Add Member` use case allows a ChocAn operator to add a new member to the ChocAn database.
Step-by-Step Description
Receives a member number and member name from a ChocAn operator. Adds the member number and associated member name to the ChocAn database.

Table 21: The Delete Member use case description.

Brief Description
The `Delete Member` use case allows a ChocAn operator to remove a member from the ChocAn database.
Step-by-Step Description
Receives a member number from a ChocAn operator. Removes the member associated with the member number from the ChocAn database.

Table 22: The Update Member use case description.

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

Table 23: The Get Weekly Fees use case description.

Brief Description
The `Get Weekly Fees` use case gets the fees from the DB for the current or previous week. **Step-by-Step Description** Receives a range of dates. Queries the ChocAn database for all `totalFees` in the range of dates. Returns the list of `totalFees` for the given date range.

Table 24: The Retrieve Services use case description.

Brief Description
The `Retrieve Services` use case fetches the services the provider has rendered in the past week. **Step-by-Step Description** Receives a request from the `Bill ChocAn` use case for a list of services. Queries the terminal for all `unsubmitted` services. Returns a list of all `unsubmitted` services, their associated `serviceIds`. Marks the services in the terminal as `submitted`.

Table 25: The Save Service use case description.

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

References

Gaphor Developers. 2025. “Gaphor Documentation.” 2025. https://docs.gaphor.org/en/latest/getting_started.html.
Pandoc. 2025. “Pandoc User’s Guide.” 2025. https://pandoc.org/MANUAL.html.
Schach, Stephen R. 2010. *Object-Oriented and Classical Software Engineering*. 8th ed. McGraw-Hill.