Stamford Chat Project Documentation

Project Title: Stamford Chatbot

Course Code: ITE222

Course Name: Programming II

Student (1) Name: Lynn Thant

Student (1) ID: 2406120001

Student (2) Name: Zwe Pyi Phyo

Student (2) ID: 2406120005

Course: ITE:222

Instructor: Dr Nay Myo Sandar

Date: 4 Jun 2025

Contents

Introduction

In today’s digitally driven educational landscape, universities are increasingly adopting chatbot technology that helps many of its students to complete their respective tasks. Likewise, Stamford Chatbot is a Java-based chatbot application that aims to assist students at Stamford University by providing quick and easy access to essential information that is related to campus. This chatbot will be able to offer instant responses to common queries such as course information, the academic calendar, map of a campus, contact details of staff and allows student to calculate their budgets.

Initially, this project was developed to address the common challenges students face in accessing important university information quickly and efficiently. By creating a user-friendly chatbot using Java, we aim to streamline communication between students and the services from the university. To achieve this, the chatbot applies core programming techniques such as conditional logic, menu-driven interaction and modular design. As a result, Stamford Chatbot not only improves convenience of a student but also helps reduce the workload of administrative staff by handling frequently asked questions. This report outlines the design, functionality, and technical implementation of Stamford Chatbot.

Objectives

This section outlines the key goals of the Stamford Chatbot project. The main purpose is to design a functional and accessible chatbot application that enhances the way students interact with university services. The objective focuses on both user experience and technical design to ensure the system is helpful, reliable and easy to maintain.

Objectives of the project are:

- To develop a Java-based chatbot system tailored for university student support
- To provide quick and accurate access to key information such as:
    - Course details
    - Academic Calendar

  - Campus Map
  - Contact information of university staff
  - Calculate budget
- To ensure the chatbot interface is simple, user-friendly and intuitive.
- And, to write a modular and maintainable code that supports scalability and future updates among developers.

Tools and Technologies Used

This section presents the tools, technologies and development environments used in the implementation of the Stamford Chatbot. The tools selected for this project were based on their compatibility, ease of use and the ability to support efficient development and testing of a console-based chatbot system.

Category	**Tools / Technologies**
Programming Language	Java (JDK) – for core development and execution
Development Environment	Eclipse IDE – for writing, debugging, and managing Java code.
Execution Interface	Command Line / Terminal – for compiling and running the application
Java Packages Used	- java.util.Scanner– for reading user input - java.lang – String manipulation.

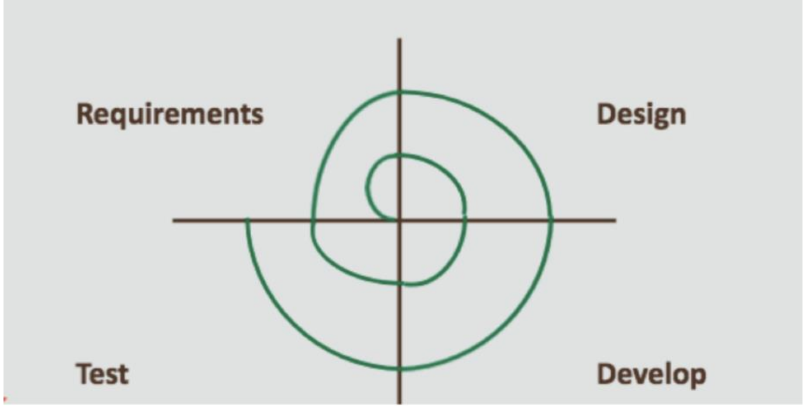
System Overview

The Stamford Chatbot system is a console program built on Java that helps users with a variety of university-related questions. It offers features like searching for course information, browsing the academic calendar, navigating the campus map, and finding staff contact information. The system is modular with separate classes handling each specific task. Users interact with the chatbot through a simple text interface, entering commands like "help," "course info," or "exit" to navigate the options. The program isn’t case sensitive, allowing users to enter correct commands as they prefer.

The system uses a layered and organized structure to handle user commands. When a user types a command, the main class checks the input and passes it to the system class. This system class then creates and uses the right feature class to do the task. For example, if the user wants campus map details, the system calls showMap(), which creates a Stamford_Chatbot_CampusMap object to handle that request. This layered approach ensures proper separation of concerns, with each class handling specific responsibilities. The benefit of developing with this design pattern allows centralized control over a program that standardize request handling and managing error. Additionally, when adding new features, the existing work will be not be disrupted, encouraging scalability to the application. Overall, the chatbot achieves to be a clean, organized structure while maintaining flexibility for future enhancements.

Spiral Modal

The development of this project follows the Spiral Modal that provides a systematic and iterative approach to a software development.



Requirement

In this phase we identified the need for a chat bot to assist Stamford University students and allows them to access important campus-related information quickly. Main function of the chat bot includes:

- Providing Course Information
- Displaying the Academic Calendar
- Offering Campus Map details
- Providing contact details for staff
- Calculating student budgets.

Additionally, we also considered potential challenges, such as user input errors and the complexity of the logic. Moreover, the decision was made to use a command-line interface due to its simplicity and our current knowledge limitations with graphical user interfaces.

Design

Based on the requirements, we planned the architecture of the Stamford Chatbot as a console program. The system was designed to be modular, with separate classes handling specific tasks.

- Stamford_Chatbot.java (Main class): designed as an entry point
- Stamford_Chatbot_System.java: designed as a central control class.
- Stamford_Chatbot_CourseFinder.java: Designed to provide detailed course information.
- Stamford_Chatbot_Schedule.java: Designed to display academic calendar.
- Stamford_Chatbot_Contact.java: Design to provide contact information.

- Stamford_Chatbot_CampusMap.java: Designed to display location information.

- Stamford_Chatbot_Budget.java: Designed to help students manage their finance.

Develop

In the product development phase, the chatbot was built according to the planned design. This involved coding the main chatbot structure first, followed by the development of each feature module (course information, academic calendar, campus map, contact details, and budget calculation) incrementally.

- Java code was written for each class and their respective methods, implementing the logic for user interaction, data processing, and output display.

- Looping constructs like while loops were used to maintain program execution, and for loops were used for iterations, such as calendar generation.

- Conditional statements (if/else if, nested if) were extensively used for menu navigation and handling hierarchical selections.

- Input and output were managed using Scanner and Object instantiation was used extensively to access functionality across classes.

- Stamford_Chatbot_System is a class for creating objects of specialized classes to handle specific requests.

- Lastly, syntax errors were addressed during this coding phase.

Test

The final phase involved rigorous review and evaluation to ensure the chatbot met all project requirements and functioned as expected.

- Manual Testing: The program was run multiple times with various inputs to check the responsiveness and accuracy of each module.
- Input Testing: Different inputs were tested to verify how well the system responded, including handling valid and potentially invalid user entries.
- Output Testing: The program's outputs were checked for correctness and clarity, ensuring messages were informative and summaries were accurate.
- Code Quality Review: The overall structure and design were reviewed to confirm that the code was clean, modular, and easy to understand, contributing to the chatbot's reliability and user-friendliness. Small bugs were identified and fixed, and message clarity was improved during this phase.

In summary, the system development of the Stamford Chatbot was carried out through a structured and modular approach. To begin, the main chatbot framework was built to establish the core functionality and interaction flow. While, individual feature modules were developed and integrated into the system. Additionally, the development process was following principle of Spiral Modal, where all phases were revisited iteratively to refine the system and reduce potential error. This approach allowed continuous improvement throughout the development process. And as a result, this project successfully produced a well-organized and functional chatbot that promotes scalability and maintainability.

Class Description

The Stamford Chatbot project consists of 7 classes that is designed to fulfill a specific role. By organizing features into separate classes, the system ensures a clear separation of concerns. This allows the development process to be more modular and maintainable. Ultimately, making the code easier to understand, test and expand.

The chatbot leverages object-oriented programming principles, where each class encapsulates its logic and user interaction processes. From handling basic user input to delivering course information the following class descriptions highlight the purpose of each component within the chatbot system.

Stamford_Chatbot.java (Main Class)

The Stamford_Chatbot.java class serves as the entry point for the Stamford Chatbot system. It acts as a bootstrapper that initializes the application and manages user input. The class utilizes a Scanner package to read user commands and while-loop to keep the program running until the user chooses to exit. Furthermore, the class consists of a main method, and it creates an instance of the Stamford_Chatbot_System class, which provides access to other specialized classes. Based on the user's input, conditional checks, if-else statements determine the appropriate action by invoking the corresponding method from the system class to handle the request.

Method	**Purpose**	**Example Output**
main(String[] args)	Entry point that initializes and runs the chatbot system	(No direct output - launches system)
Key Feature: Act as an entry point and instantiate system class.		

Stamford_Chatbot_System.java

The Stamford_Chatbot_System.java function as a central control class. The class consists of a constructor that display a welcoming message and several methods that invoke an instance of a specialized tasks.

Method	Purpose	Example Output
Constructor	Initializes the chatbot system and displays welcome message	"STAMFORD CHATBOT" Type 'help' for options"
Help()	Displays the main help menu	"1. Course Info 2. Academic Calendar 3. Campus Map 4. Contact Staff 5. Exit"
findCourse()	Launches course information module	"What area are you interested in? - Computing - Art - Business"
showSchedule()	Starts the academic calendar system	"Enter the year (2025 or 2026) to view the academic calendar…"
showMap()	Initiates campus location services	"Where? (Library/Gym/Cafeteria):"
contactStaff()	Provides department contact information	"Department: (Admissions/IT/Finance):"
Key Feature: Act as a central command between main class and module classes		

Stamford_Chatbot_CourseFinder.java

The Stamford_Chatbot_CourseFinder.java class is a specialized class designed to provide detailed course information based on user queries. It interacts with users through a structured menu system, guiding them to select an academic area and then a specific subject within that area. Its findCourse() is a main method that orchestrate the user inteaction and logic behind the course lookup. Lastly, its modular design ensures clean separation from other chatbot functionalities, promoting maintainability and scalability.

Method	Purpose	Example Output
findCourse()	Interacts with the user via console to suggest relevant courses based on input area and subject	User enters "Computing" → then "network" → gets info about ITE 554: Computer Networks.
Key Feature: The Stamford_Chatbot_CourseFinder class interactively guides users to discover course information based on their area of interest and subject preferences in Computing, Art, or Business.		

Stamford_Chatbot_Schedule.java

The Stamford_Chatbot_Schedule class is one of core components of the Stamford Chatbot system, designed to display academic calendars and key dates for specified years (2025 or 2026). It generates a monthly calendar view with annotations for important events and supports user interaction through a console interface. The class consists of constructor and several methods that allow users to select year, view a formatted calendar, check important dates and get detailed event summaries. It is built with a 6x7 2D array to represent weeks and days.

Method	Purpose	Example Output
Constructor	Starts user input loop to display the calendar for year 2025 or 2026.	"Enter the year (2025 or 2026) to view the academic calendar (or 0 to exit):"
displayCalendar(int year)	Displays the full 12-month calendar for the given year, marking holidays and exam dates.	Calendar grid with symbols (e.g., 15X, 01*) + summary at the end.

resetCalendarGrid()	Resets the 2D calendar array before rendering each month's calendar.	Internal operation; clears the grid. No direct output.
printEventSummary(int year)	Prints a summary of holidays and fake exam dates for the selected year.	Shows: 2025-01-01 - New Year's Day (Holiday), etc.
isLeapYear(int year)	Checks whether a given year is a leap year.	isLeapYear(2024) → true, isLeapYear(2025) → false
getDaysInMonth(int year, int month)	Returns the number of days in a specific month of a year.	getDaysInMonth(2025, 2) → 28, getDaysInMonth(2024, 2) → 29
getMonthName(int month)	Converts a numeric month to its full name.	getMonthName(3) → "March"
getDayOfWeek(int year, int month, int day)	Calculates day of the week (0 = Sunday, ..., 6 = Saturday).	getDayOfWeek(2025, 1, 1) → 3 (Wednesday)
isFakeHoliday(int year, int month, int day)	Determines if a given date is marked as a fake holiday.	isFakeHoliday(2025, 2, 12) → true (Fake Winter Break)
isFakeExamDate(int year, int month, int day)	Determines if a given date falls in a fake exam period.	isFakeExamDate(2025, 5, 18) → true (Midterm)
Key features: The Stamford_Chatbot_Schedule class dynamically generates and displays a year-long academic calendar for 2025 or 2026, highlighting holidays and exam periods within a formatted monthly grid.		

Stamford_Chatbot_Contact.java

The Stamford_Chatbot_Contact class is a specialized module within the Stamford Chatbot system that handles contact information requests for university departments. It provides users with email addresses and phone numbers for key departments such as Admissions, IT and Finance departments through an interactive console menu.

Method	Purpose	Example Output
Constructor Stamford_Chatbot_Contact()	Initializes the contact system and prompts the user for a department choice.	"Department: (Admissions/IT/Finance) : "
displayAdmissions()	Shows Admissions contact info.	"✉ admission@stamford.edu ☏ 02-765-4321 (9AM-5PM)"
displayIT()	Shows IT Helpdesk contact info.	"✉ it-support@stamford.edu ☏ 02-123-4567 (9AM-5PM)"
displayFinance()	Shows Finance Office contact info.	"✉ cashier_rm9@stamford.edu ☏ 02-769-4000 (8:30AM-5:30PM)"
Key Feature: The Stamford_Chatbot_Contact class allows users to quickly retrieve contact information for specific university departments such as Admissions, IT, and Finance.		

Stamford_Chatbot_CampusMap.java

The Stamford_Chatbot_CampusMap class is also one of the specialized modules within the Stamford Chatbot system that provides location-based information about key campus facilities. It offers users quick access to details such as building locations, operating hours, and amenities for predefined campus spots.

Method	Purpose	Example Output
Constructor Stamford_Chatbot_CampusMap()	Initializes the campus map system and prompts the user for a location of choice.	"Where? (Library/Gym/Cafeteria): "
displayLibrary()	Shows Library location and hours.	“Building A, 1st Floor Open 7AM – 7PM”
displayGym()	Informs users about gym availability (currently not at Rama 9 campus).	“Sorry, Not Available in Rama 9 Campus”
displayCafeteria()	Displays Cafeteria details, including today’s menu.	"Building B, 1st Floor Open 7AM-7PM Today's Menu: Pad Thai, Salad"
Key Feature: The Stamford_Chatbot_CampusMap class provides users with location and availability details for key campus facilities such as the library, gym, and cafeteria based on their input.		

Student_Chatbot_Budget.java

The Student_Chatbot_Budget class is an interactive Java tool that helps students understand their monthly financial situation. Its key methods focus on collecting data, calculating totals, and clearly presenting the results.

Method	Purpose	Example Output
Student_Chatbot_Budget()	Initializes chatbot and runs the budget tool	Runs the budget interaction
StudentBudgetCalculator()	Initializes calculator with income	Creates a calculator object
setTuitionFees(double)	Set tuition fee value	void (no output)
setAccommodation(double)	Set accommodation expense	void (no output)
setFood(double)	Set food expense	void (no output)

setTransportation(double)	Set transportation cost	void (no output)
setPersonalExpenses(double)	Set personal expenses	void (no output)
setOtherExpenses(double)	Set miscellaneous/other expenses	void (no output)
calculateRemainingBudget()	Perform remaining budget calculation	double (e.g., 450.00)
displayBudgetSummary()	Display a breakdown of income and expenses	Prints budget summary
Key Feature: It provides an interactive console-based tool for students to input their income and expenses, calculate their remaining budget, and display a detailed financial summary.		

In short, the Stamford Chatbot project is built on a well-structured, object-oriented design that divides functionality into multiple specialized classes. The Stamford_Chatbot.java serves as the main entry point while the Stamford_Chatbot_System.java acts as a controller. This modular design promotes maintainability and scalability of the program while keeping clear separation of concerns.

Core Concepts Demonstrated

This section highlights the core Java concepts that were applied during the development of the Stamford Chatbot system. Concepts such as loops, conditions, constructors and object interaction within this project showcases our abilities and solid understanding of core object-oriented programming principles. The following points summarize the technical concepts integrated into the chatbot's design and implementation.

1. Loops

while loop: Used in Stamford_Chatbot.java to keep the program running until the user exits. This keeps the chatbot interactive without having to restart the program.

for loop: In Stamford_Chatbot_Schedule.java to iterate through months/days for calendar generation. It is ideal for this case because the length of duration is predefined. For example, looping through 12 months or 7 days.

if/else if: For menu navigation (e.g., checking user input for "course info" or "calendar"). It allows decision making based on user input.

nested if: In Stamford_Chatbot_CourseFinder.java to handle multiple level of choices (area → subject). This ensures clear and structured interaction flow.

2. Constructors

Constructors: Each class initializes its own state. This allows the chatbot to set up default values and prompt the user immediately. For example, new Stamford_Chatbot_CampusMap() displays a prompt.

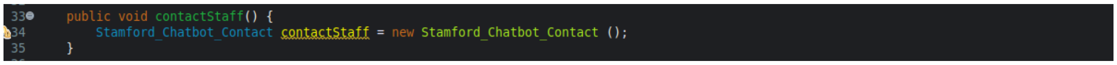
Method Overloading: Not explicitly used because constructors laid the foundation for flexible class-based architecture.

3. Inheritance

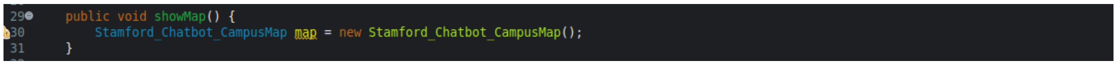
Not Applied: Not applied because each class serves a unique and specific purpose, and the design of a project favors composition over inheritance to reduce tight coupling between classes and maintain modularity.

4. Object Instantiation

Used extensively to access functionality. This promotes separation of concerns and allows each class to handle its own functionality independently.



```
    public void contactStaff() {
        Stamford_Chatbot_Contact contactStaff = new Stamford_Chatbot_Contact ();
    }
```



```
    public void showMap() {
        Stamford_Chatbot_CampusMap map = new Stamford_Chatbot_CampusMap();
    }
```

5. this and Dot Operator

This: Used in StudentBudgetCalculator to distinguish between instance variables and method parameters with the same name.

Dot Operator: Used frequently (e.g., chatBot.findCourse()) to call methods from instantiated objects, enabling interaction between classes.

6. Java Packages

Java.util.Scanner: Used for reading user input in all interactive classes. This is essential for console-based interaction and allows real-time communication between the chatbot and users.

7. Encapsulation

Private Fields: Applied in StudentBudgetCalculator to restrict direct access to sensitive data like expenses and income.

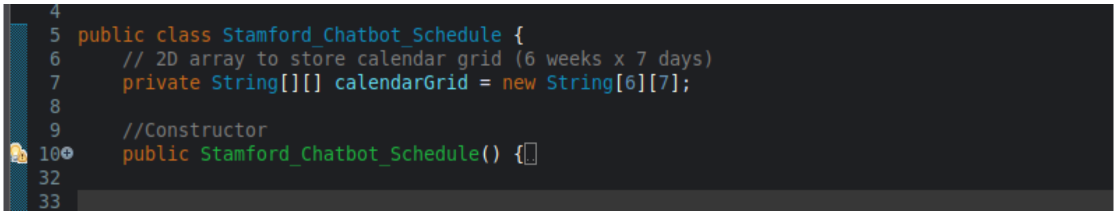
Getters/Setters: Implemented in StudentBudgetCalculator for all expense categories. Used to safely access and modify private fields, promoting data security and control.

8. Object Interaction

Collaboration: Stamford_Chatbot_System.java coordinates interactions between the main class and specalized classes (e.g., CourseFinder.java, CampusMap.java). It acts as a controller to coordinate other feature-specific classes. This allows the system to respond dynamically to user input by invoking the right class and method.

9. 2D Arrays

6x7: Used in Stamford_Chatbot_Schedule.java to model the calendar structure (6 weeks × 7 days). A 2D array is ideal for displaying a grid format like a monthly calendar.



```java

public class Stamford_Chatbot_Schedule {
    // 2D array to store calendar grid (6 weeks x 7 days)
    private String[][] calendarGrid = new String[6][7];

    //Constructor
    public Stamford_Chatbot_Schedule() {


```

10. Static vs. Non-Static Methods

Static: Used for utility-like functions such as getDayOfWeek() where no object state is needed.

Non-Static: Used for instance-based behavior (e.g., findCourse()) to allow each object to maintain its own state and logic.

11. Commenting & Naming

Comments: Added throughout the code to explain the logic, especially for complex parts like the calendar generation or input handling. This improves readability and maintainability.

Naming: Followed Java standards (camelCase for methods, PascalCase for classes) to make the code professional and easier to understand for other developers.

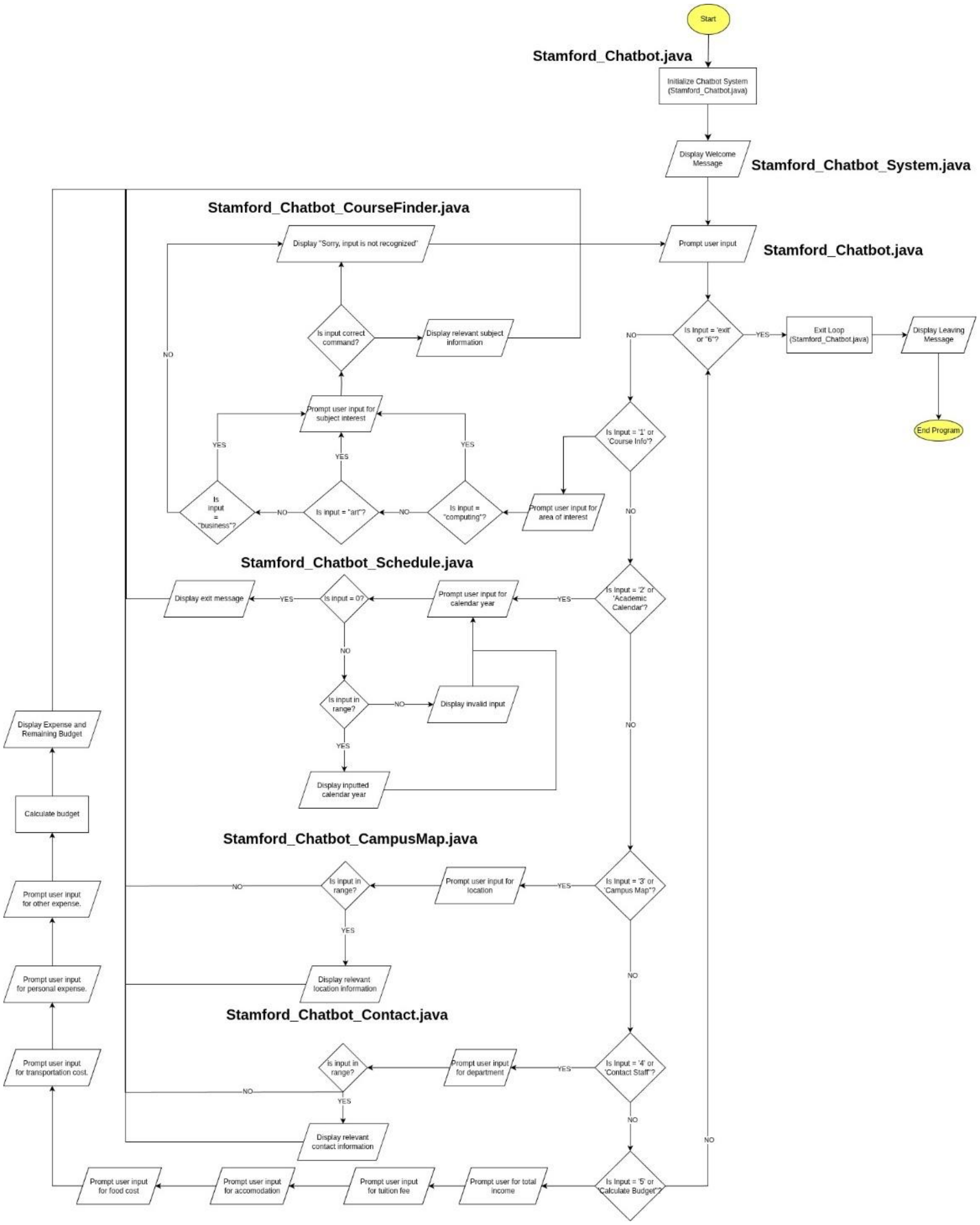
12. Access Specifiers

public: Used for methods that need to be accessed across different classes (e.g., showSchedule()).

private: Used to enforce encapsulation and prevent external classes from modifying sensitive data directly.

Flowchart

According to GeeksforGeeks (2021), flowcharts represent graphically on how data, methods or processes flow in a program. The following flowchart illustrates the logical structure and interaction flow of the Stamford Chatbot System. This visual representation helps in understanding how the chatbot maintains an interactive loop while ensuring that each user request is processed correctly.


Start
Stamford_Chatbot.java
Initialize Chatbot System (Stamford_Chatbot.java)
Display Welcome Message
Stamford_Chatbot_System.java
Prompt user input
Stamford_Chatbot.java
Is Input = 'exit' or "6"?
YES
Exit Loop (Stamford_Chatbot.java)
Display Leaving Message
End Program
NO
Is Input = '1' or 'Course Info'?
Prompt user input for area of interest
Stamford_Chatbot_CourseFinder.java
Display "Sorry, input is not recognized"
Is input correct command?
Display relevant subject information
Prompt user input for subject interest
Is input = "computing"?
Is input = "art"?
Is input = "business"?
YES
NO
Is Input = '2' or 'Academic Calendar'?
Stamford_Chatbot_Schedule.java
Prompt user input for calendar year
Is input = 0?
Display exit message
Is input in range?
Display invalid input
Display inputted calendar year
Is Input = '3' or 'Campus Map"?
Stamford_Chatbot_CampusMap.java
Prompt user input for location
Is input in range?
Display relevant location information
Is Input = '4' or 'Contact Staff"?
Stamford_Chatbot_Contact.java
Prompt user input for department
Is input in range?
Display relevant contact information
Is Input = '5' or 'Calculate Budget"?
Prompt user for total income
Prompt user input for tuition fee
Prompt user input for accomodation
Prompt user input for food cost
Prompt user input for transportation cost.
Prompt user input for personal expense.
Prompt user input for other expense.
Calculate budget
Display Expense and Remaining Budget

Algorithm

Unlike Flowchart, algorithm illustrates the step-by-step procedure to solve a problem by using a plain text (GeeksForGeeks, 2019). The algorithm represents the logical flow of the chatbot from the moment it starts to when it exits, ensuring a consistent and user-friendly experience. Therefore, it is crucial to breakdown the process and understand the chatbot's internal logic and control structure.

```
Step 1: Start
Step 2: Create an instance of Stamford_Chatbot_System
Step 3: Display Welcome Message.
Step 4: Prompt user for input
Step 5: Read the input using a Scanner.
Step 6: Check input against commands (help, course info, academic calendar, campus map, contact staff and exit).
    Step 6.1: If user enter course info.
        Step 6.1.1: Prompt user to select an area of study.
        Step 6.1.2: Read user's area of choice.
        Step 6.1.2: Prompt for a subject keyword.
        Step 6.1.3: If Match, display course details.
        Step 6.1.4 Return to main loop (Step 4)
    Step 6.2: If user enter academic calendar
        Step 6.2.1: Prompt user for a year
        Step 6.2.2: If year is valid, print calendar grind.
        Step 6.2.3: Else if year = 0, return to the main loop (Step 4)
```

```
                Step 6.2.4: Else print “Invalid Year”
        Step 6.3: If user enter campus map
                Step 6.3.1: Prompt user for location (Library, Gym, Cafeteria).
                Step 6.3.2: If valid location, display appropriate result.
                Step 6.3.3: Return to main loop (Step 4)
        Step 6.4: If user enter contact staff
                Step 6.4.1: Prompt user for department
                Step 6.4.2: Check department, if correct → display appropriate result.
                Step 6.4.3: Return to main loop (Step 4)
        Step 6.5: If user enter calculate budget
                Step 6.5.1: Prompt user for total income and wait for input.
                Step 6.5.2: Prompt user for tuition fee and wait for input.
                Step 6.5.3: Prompt user for accommodation and wait for input.
                Step 6.5.4: Prompt user for estimated food cost and wait for input.
                Step 6.5.5: Prompt user for estimated transportation cost.
                Step 6.5.6: Prompt user for personal expenses and wait for input.
                Step 6.5.7: Prompt user for other expenses and wait for input.
                Step 6.5.8: Calculate Expense.
                Step 6.5.9: Display expenses and remaining budget.
                Step 6.5.10: Return to main loop (Step 4)
Step 7: Repeat from Step 4 until “exit” is entered.
```

Step 8: Break While Loop

Step 9: Close Scanner.

Step 10: Display “Bye! See you again”

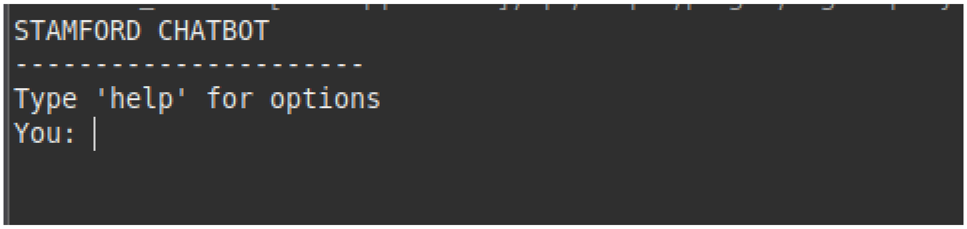
Step 11: End

As shown in above, the algorithm delivers a precise and step-by-step breakdown of the chatbot's internal logic using plain text. This procedural clarity is essential for understanding how user inputs are processed and how the system responds accordingly. Therefore, this approach supports the development of a robust and modular chatbot that is easy to maintain and expand.

Screenshots / Sample Outputs

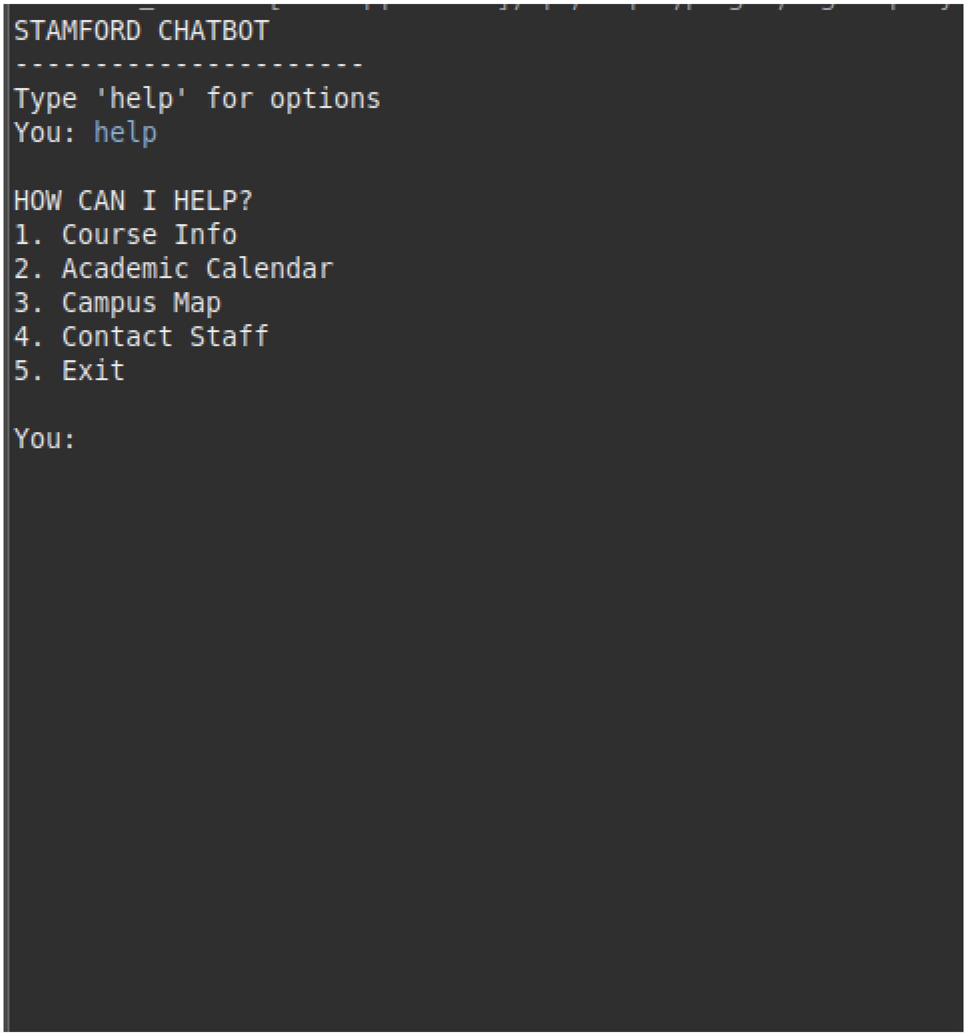
Screenshots and sample outputs showcase the functionality and how a user will interact with the Stamford Chatbot System. It provides a clear representation of how the chatbot responds to user commands and highlighting features that are implemented in the program.

1. Start



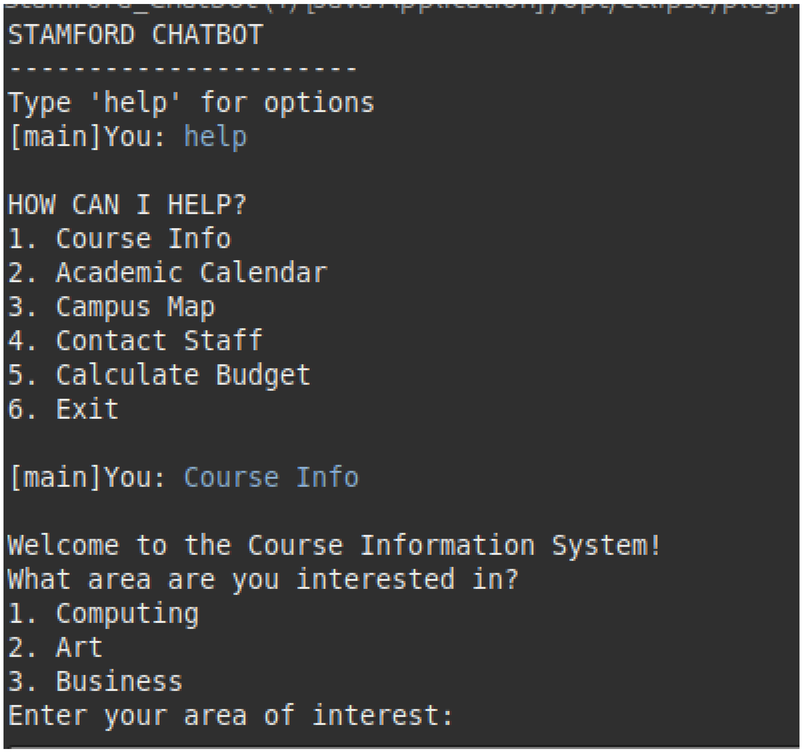
```
STAMFORD CHATBOT
----------------------
Type 'help' for options
You: |
```

2. Users enter ‘help’



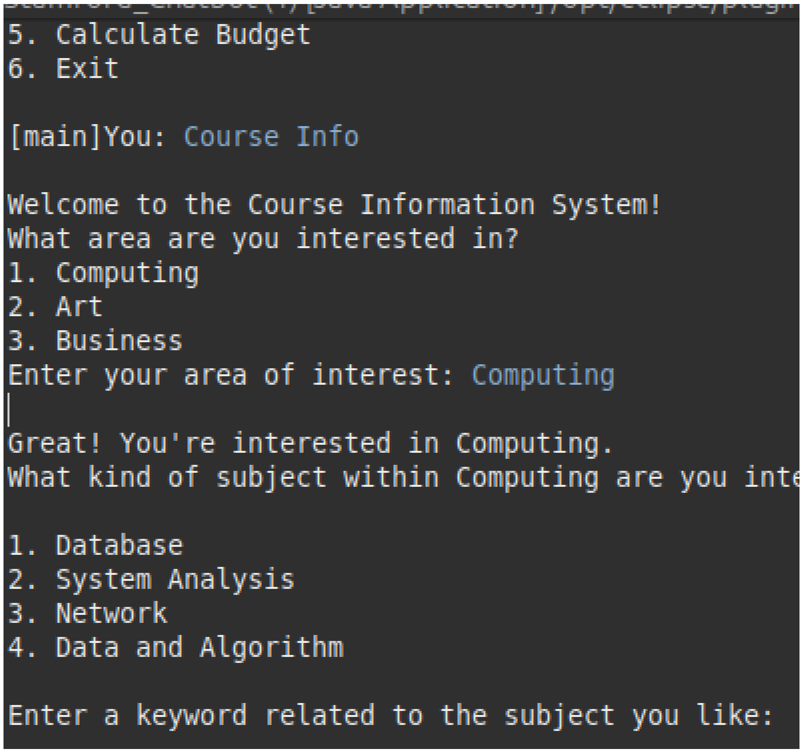
```
STAMFORD CHATBOT
----------------------
Type 'help' for options
You: help

HOW CAN I HELP?
1. Course Info
2. Academic Calendar
3. Campus Map
4. Contact Staff
5. Exit

You:
```

3. Users enter ‘course info’.



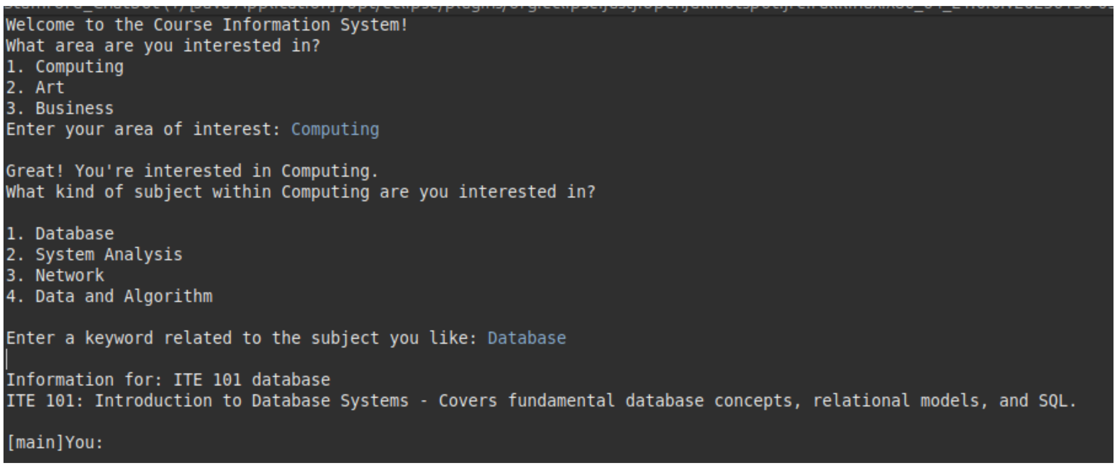
```
STAMFORD CHATBOT
----------------------
Type 'help' for options
[main]You: help

HOW CAN I HELP?
1. Course Info
2. Academic Calendar
3. Campus Map
4. Contact Staff
5. Calculate Budget
6. Exit

[main]You: Course Info

Welcome to the Course Information System!
What area are you interested in?
1. Computing
2. Art
3. Business
Enter your area of interest:
```

3.1 User chose ‘computing’ as an area of interest.



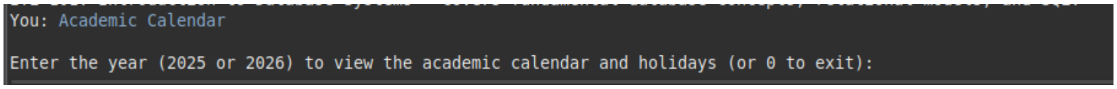
```
5. Calculate Budget
6. Exit

[main]You: Course Info

Welcome to the Course Information System!
What area are you interested in?
1. Computing
2. Art
3. Business
Enter your area of interest: Computing

Great! You're interested in Computing.
What kind of subject within Computing are you inte

1. Database
2. System Analysis
3. Network
4. Data and Algorithm

Enter a keyword related to the subject you like:
```

3.2 User chose ‘Database’ as a subject.



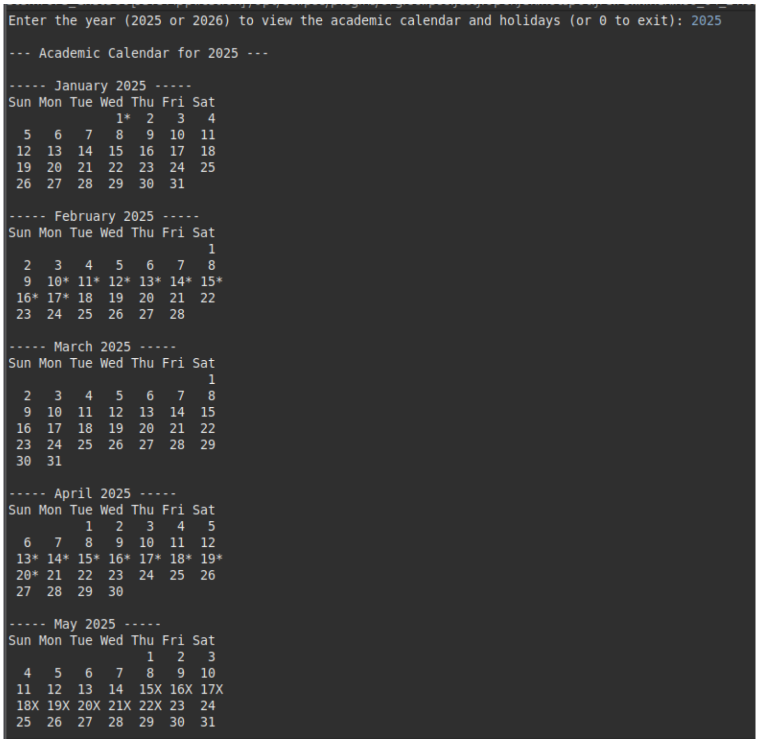
```
Welcome to the Course Information System!
What area are you interested in?
1. Computing
2. Art
3. Business
Enter your area of interest: Computing

Great! You're interested in Computing.
What kind of subject within Computing are you interested in?

1. Database
2. System Analysis
3. Network
4. Data and Algorithm

Enter a keyword related to the subject you like: Database

Information for: ITE 101 database
ITE 101: Introduction to Database Systems - Covers fundamental database concepts, relational models, and SQL.

[main]You:
```

4. Users enter ‘Academic Calendar’.

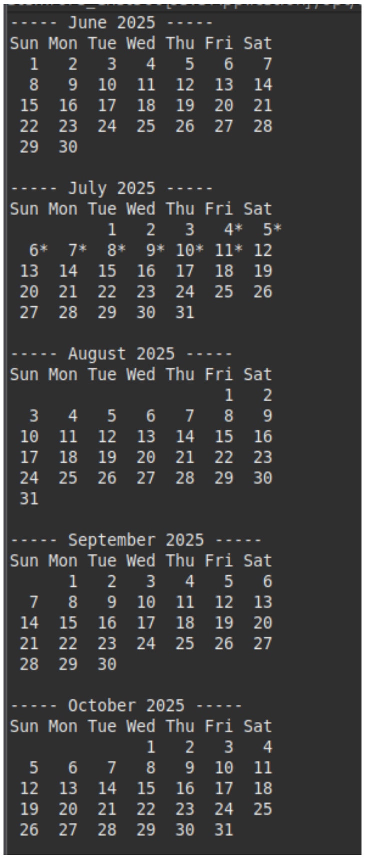


```
You: Academic Calendar

Enter the year (2025 or 2026) to view the academic calendar and holidays (or 0 to exit):
```

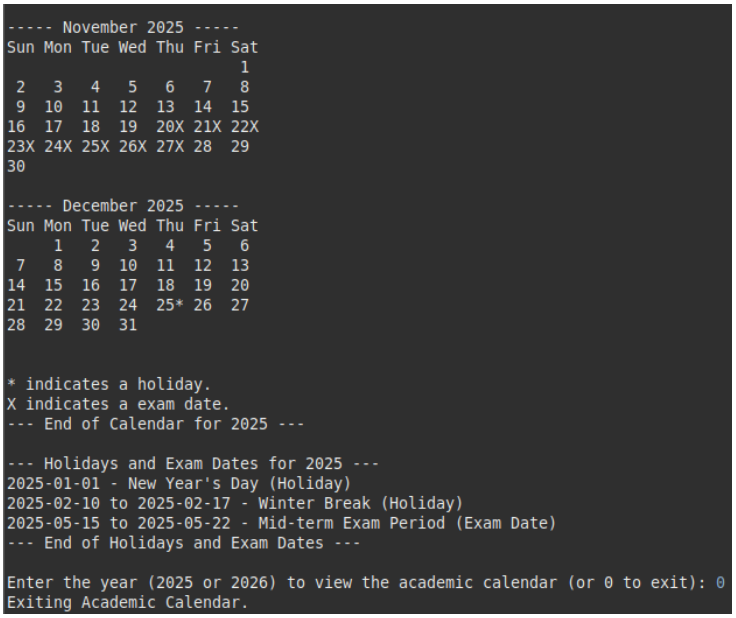
4.1 User enters year ‘2025’.



```
Enter the year (2025 or 2026) to view the academic calendar and holidays (or 0 to exit): 2025

--- Academic Calendar for 2025 ---

----- January 2025 -----
Sun Mon Tue Wed Thu Fri Sat
              1*  2   3   4
  5   6   7   8   9  10  11
 12  13  14  15  16  17  18
 19  20  21  22  23  24  25
 26  27  28  29  30  31

----- February 2025 -----
Sun Mon Tue Wed Thu Fri Sat
                          1
  2   3   4   5   6   7   8
  9  10* 11* 12* 13* 14* 15*
 16* 17* 18  19  20  21  22
 23  24  25  26  27  28

----- March 2025 -----
Sun Mon Tue Wed Thu Fri Sat
                          1
  2   3   4   5   6   7   8
  9  10  11  12  13  14  15
 16  17  18  19  20  21  22
 23  24  25  26  27  28  29
 30  31

----- April 2025 -----
Sun Mon Tue Wed Thu Fri Sat
          1   2   3   4   5
  6   7   8   9  10  11  12
 13* 14* 15* 16* 17* 18* 19*
 20* 21  22  23  24  25  26
 27  28  29  30

----- May 2025 -----
Sun Mon Tue Wed Thu Fri Sat
                  1   2   3
  4   5   6   7   8   9  10
 11  12  13  14  15X 16X 17X
 18X 19X 20X 21X 22X 23  24
 25  26  27  28  29  30  31
```

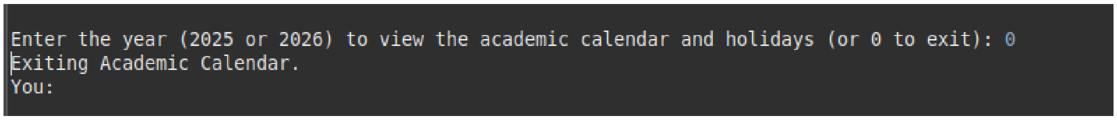


```
----- June 2025 -----
Sun Mon Tue Wed Thu Fri Sat
  1   2   3   4   5   6   7
  8   9  10  11  12  13  14
 15  16  17  18  19  20  21
 22  23  24  25  26  27  28
 29  30

----- July 2025 -----
Sun Mon Tue Wed Thu Fri Sat
          1   2   3   4*  5*
  6*  7*  8*  9* 10* 11* 12
 13  14  15  16  17  18  19
 20  21  22  23  24  25  26
 27  28  29  30  31

----- August 2025 -----
Sun Mon Tue Wed Thu Fri Sat
                      1   2
  3   4   5   6   7   8   9
 10  11  12  13  14  15  16
 17  18  19  20  21  22  23
 24  25  26  27  28  29  30
 31

----- September 2025 -----
Sun Mon Tue Wed Thu Fri Sat
      1   2   3   4   5   6
  7   8   9  10  11  12  13
 14  15  16  17  18  19  20
 21  22  23  24  25  26  27
 28  29  30

----- October 2025 -----
Sun Mon Tue Wed Thu Fri Sat
              1   2   3   4
  5   6   7   8   9  10  11
 12  13  14  15  16  17  18
 19  20  21  22  23  24  25
 26  27  28  29  30  31
```



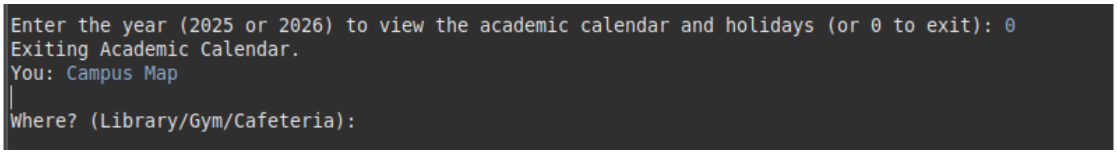
```
----- November 2025 -----
Sun Mon Tue Wed Thu Fri Sat
                          1
 2   3   4   5   6   7   8
 9  10  11  12  13  14  15
16  17  18  19  20X 21X 22X
23X 24X 25X 26X 27X 28  29
30

----- December 2025 -----
Sun Mon Tue Wed Thu Fri Sat
     1   2   3   4   5   6
 7   8   9  10  11  12  13
14  15  16  17  18  19  20
21  22  23  24  25* 26  27
28  29  30  31


* indicates a holiday.
X indicates a exam date.
--- End of Calendar for 2025 ---

--- Holidays and Exam Dates for 2025 ---
2025-01-01 - New Year's Day (Holiday)
2025-02-10 to 2025-02-17 - Winter Break (Holiday)
2025-05-15 to 2025-05-22 - Mid-term Exam Period (Exam Date)
--- End of Holidays and Exam Dates ---

Enter the year (2025 or 2026) to view the academic calendar (or 0 to exit): 0
Exiting Academic Calendar.
```

4.2 User enters ‘exit’ to get back to menu.



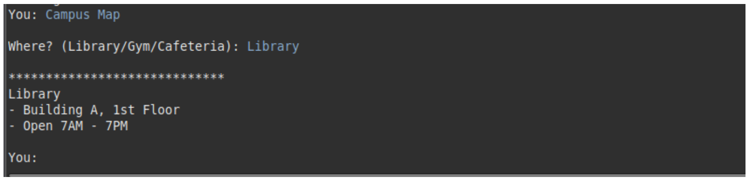
```
Enter the year (2025 or 2026) to view the academic calendar and holidays (or 0 to exit): 0
Exiting Academic Calendar.
You:
```

5. Users enter ‘Campus Map’.



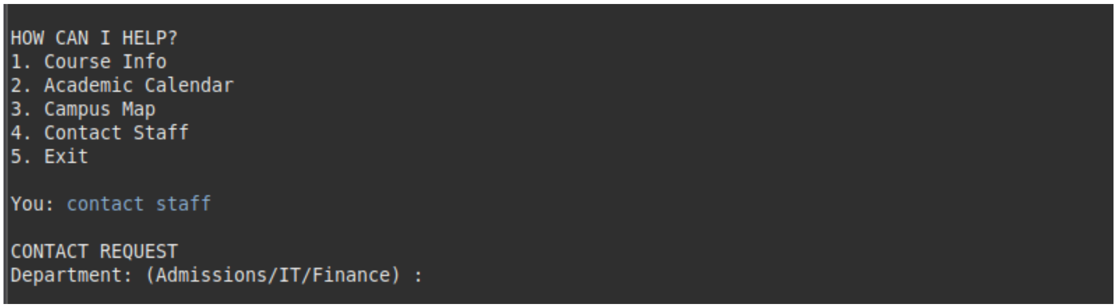
```
Enter the year (2025 or 2026) to view the academic calendar and holidays (or 0 to exit): 0
Exiting Academic Calendar.
You: Campus Map

Where? (Library/Gym/Cafeteria):
```

5.1 Users find ‘Library’.



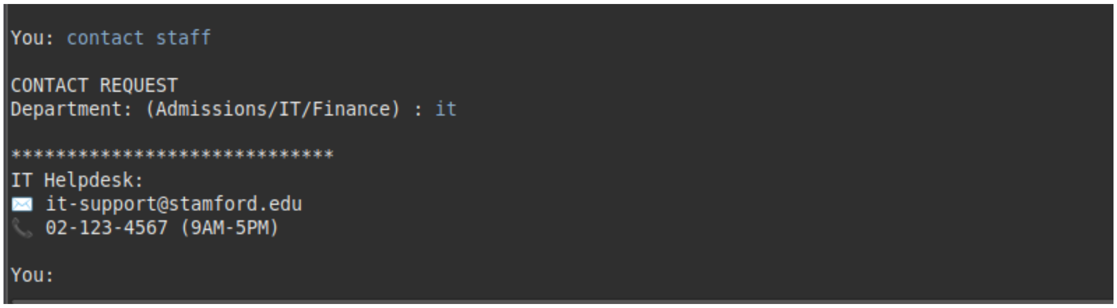
```
You: Campus Map

Where? (Library/Gym/Cafeteria): Library

*****************************
Library
- Building A, 1st Floor
- Open 7AM - 7PM

You:
```

6. Users enter ‘Contact Staff’.



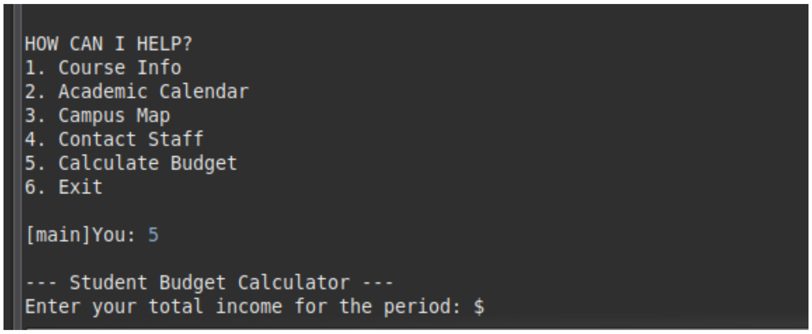
```
HOW CAN I HELP?
1. Course Info
2. Academic Calendar
3. Campus Map
4. Contact Staff
5. Exit

You: contact staff

CONTACT REQUEST
Department: (Admissions/IT/Finance) :
```

6.1 Users find ‘IT Department’.



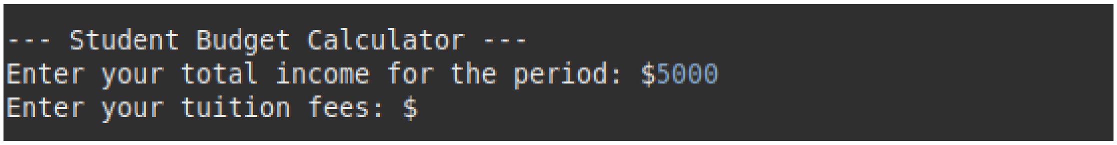
```
You: contact staff

CONTACT REQUEST
Department: (Admissions/IT/Finance) : it

*****************************
IT Helpdesk:
✉ it-support@stamford.edu
📞 02-123-4567 (9AM-5PM)

You:
```

7.0 User want to calculate budget



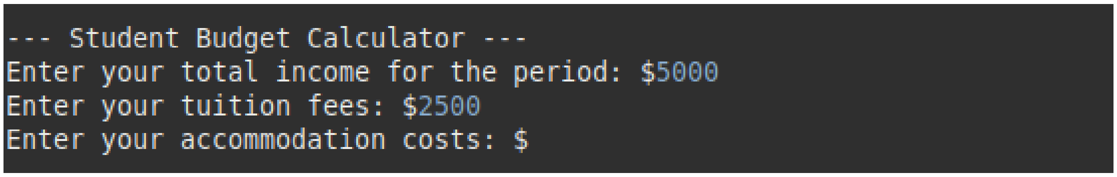
```
HOW CAN I HELP?
1. Course Info
2. Academic Calendar
3. Campus Map
4. Contact Staff
5. Calculate Budget
6. Exit

[main]You: 5

--- Student Budget Calculator ---
Enter your total income for the period: $
```

7.1 Prompt user income.



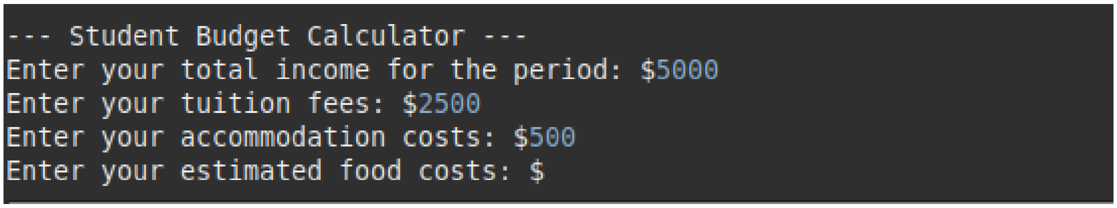
```
--- Student Budget Calculator ---
Enter your total income for the period: $5000
Enter your tuition fees: $
```

7.2 Prompt user for tuition fee.



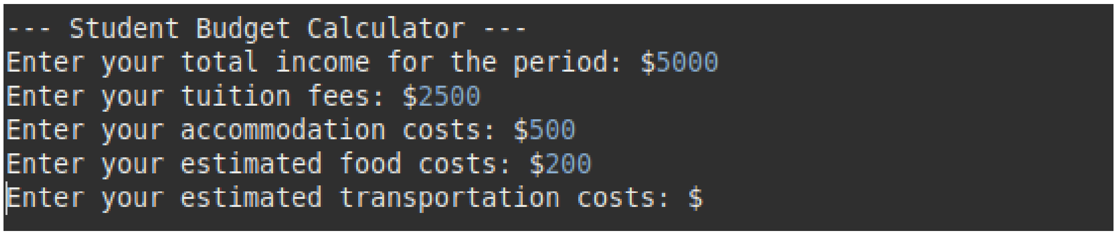
```
--- Student Budget Calculator ---
Enter your total income for the period: $5000
Enter your tuition fees: $2500
Enter your accommodation costs: $
```

7.3 Prompt user accommodation.



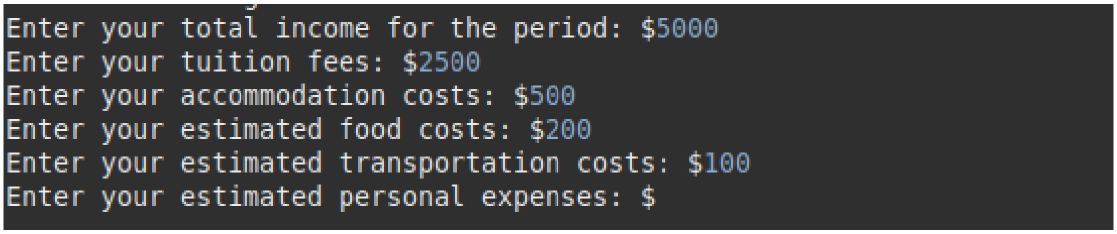
```
--- Student Budget Calculator ---
Enter your total income for the period: $5000
Enter your tuition fees: $2500
Enter your accommodation costs: $500
Enter your estimated food costs: $
```

7.4 Prompt user for estimated food costs.



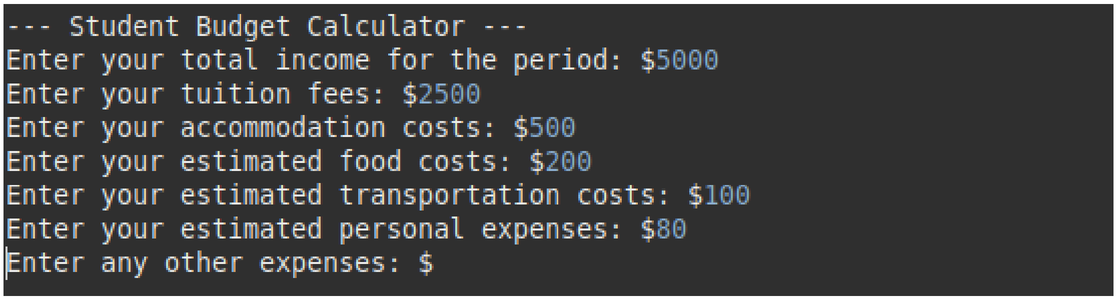
```
--- Student Budget Calculator ---
Enter your total income for the period: $5000
Enter your tuition fees: $2500
Enter your accommodation costs: $500
Enter your estimated food costs: $200
Enter your estimated transportation costs: $
```

7.5 Prompt user for transportation costs.



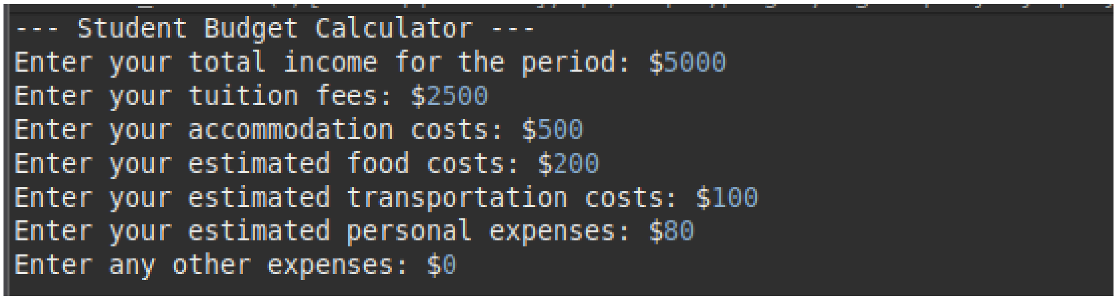
```
Enter your total income for the period: $5000
Enter your tuition fees: $2500
Enter your accommodation costs: $500
Enter your estimated food costs: $200
Enter your estimated transportation costs: $100
Enter your estimated personal expenses: $
```

7.6 Prompt user for transportation costs.



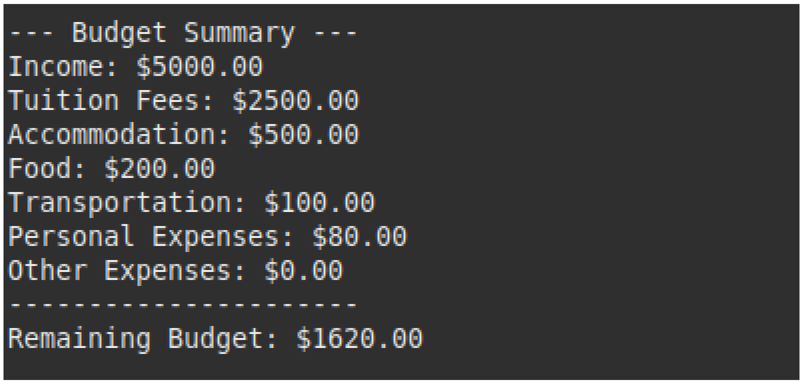
```
--- Student Budget Calculator ---
Enter your total income for the period: $5000
Enter your tuition fees: $2500
Enter your accommodation costs: $500
Enter your estimated food costs: $200
Enter your estimated transportation costs: $100
Enter your estimated personal expenses: $80
Enter any other expenses: $
```

7.7 Prompt user for transportation costs.



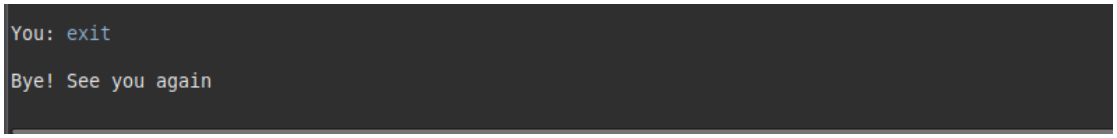
```
--- Student Budget Calculator ---
Enter your total income for the period: $5000
Enter your tuition fees: $2500
Enter your accommodation costs: $500
Enter your estimated food costs: $200
Enter your estimated transportation costs: $100
Enter your estimated personal expenses: $80
Enter any other expenses: $0
```

7.8 Print Budget Summary and Remaining Budget.



```
--- Budget Summary ---
Income: $5000.00
Tuition Fees: $2500.00
Accommodation: $500.00
Food: $200.00
Transportation: $100.00
Personal Expenses: $80.00
Other Expenses: $0.00
----------------------
Remaining Budget: $1620.00
```

8. User Exit the Program



```
You: exit

Bye! See you again
```

Conclusion

In conclusion, the Stamford Chatbot project stands as a successful Java-based console application that assist students with essential campus-related information. By adopting modular and layered architecture, it ensured a clear separation of concerns throughout the system. Additionally, the development process was carried out in Spiral Modal where each phase was carefully planned, analyzed and tested. The project also effectively applied a variety of core Java programming concepts such as object instantiation, conditional logic, loops, method calls, 2D arrays, and the usage of both static and non-static methods. Object-Oriented Programming (OOP) principles such as encapsulation were implemented through the use of private fields and getter/setter methods, promoting data security and modularity. These features collectively contributed to a clean, organized, and interactive application that successfully met its intended goals. However, despite its strengths, the current chatbot has a few limitations. Due to the absence of advanced Natural Language Processing (NLP), user input must strictly follow predefined keywords, which can reduce flexibility and overall user engagement. Additionally, the isolated structure of each module limits the ability of a chatbot to maintain conversational context or memory across different features. These limitations present opportunities for future development to enhance interactivity and user experience. Overall, the Stamford Chatbot not only provided a practical solution for student assistance but also served as a valuable learning platform for real-world Java application development.

References

List any resources or references you used during the development.

GeeksForGeeks. (2019, May 28). *Difference Between Algorithm and Flowchart*. GeeksforGeeks. https://www.geeksforgeeks.org/difference-between-algorithm-and-flowchart/

GeeksforGeeks. (2021, September 20). *What is a Flowchart and its Types?* GeeksforGeeks. https://www.geeksforgeeks.org/computer-science-fundamentals/what-is-a-flowchart-and-its-types/