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

Table of Contents

Introduction

In today's digitally-driven educational landscape, universities are increasingly adopting chatbot technology—particularly using robust programming languages like Java to enhance student engagement, helping many students complete their respective tasks. 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. It is able to offer instant responses to common queries such as course information, the academic calendar, map of a campus, and the contact details of staff. This report outlines the design, functionality, and technical implementation of Stamford Chatbot.

Objectives

The aim of this project is to develop a Java-based chat bot system that automates student support services within a university setting. The chat bot aims to provide necessary information such as course information, academic calendar, map of a campus and contact of any respective personnel in the university. The program should be easy to use as a user and consist of maintainable codes that allow smooth collaboration between developers.

Tools and Technologies Used

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 classchecks 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 main class for input/output, the system class for request routing, and specialized classes for domain-specific operations. 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 follow the Spiral Modal that provides a systematic and iterative approach to a software development.

Planning & Requirement Analysis

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
- Academic Calendar
- Campus Map
- Contact Details

Risk Analysis

During risks analysis phase, we took a consideration of potential challenges such as user input errors or over-complicating a logic. This allow us to create a program that mitigates unnecessary errors from occurring. Hence, increasing a user experience and keeping a project simple. Additionally, we chose the command line interface rather than graphic user interface due to its simplicity and our limited knowledge on the tool.

Product Development

In the product development phase, we focused on building the chatbot based on our planned design. We started by coding the main chatbot structure, then developed each feature module course information, academic calendar, campus map, and contact details one at a time.

Review and Evaluation

Lastly, In the review and evaluation phase, we carefully tested the chatbot to make sure it met all the project requirements and functioned as expected. We ran the program multiple times, trying different inputs to check how well each module responded. This helped us identify and fix small bugs, improve message clarity, and ensure smoother user interaction. We also looked at the overall structure and design to confirm that the code was clean, modular, and easy to understand. This phase ensured that the chatbot was not only working but also reliable and user-friendly.

Class Description

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 consist of 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	"What area are you interested

	module	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 component 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 module 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 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.		

Core Concepts Demonstrated

1. Loops

while loop: Used in Stamford_Chatbot.java to keep the program running until the user exits.

for loop: In Stamford_Chatbot_Schedule.java to iterate through months/days for calendar generation.

if/else if: For menu navigation (e.g., checking user input for "course info" or "calendar").

nested if: In Stamford_Chatbot_CourseFinder.java to handle hierarchical selections (area → subject).

2. Constructors

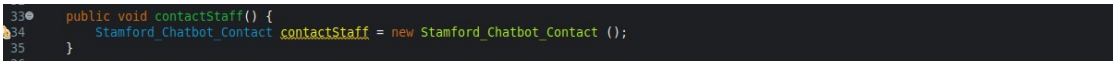
Constructors: Each classes initializes its own state.

Method Overloading: Not explicitly used.

3. Inheritance

Not Applied: The project uses composition instead of inheritance.

4. Object Instantiation



5. this and Dot Operator

This: not heavily used.

Dot Operator: Frequently used to access methods (eg. chatBot.findCourse()).

6. Java Packages

Java.util.Scanner: For user input in all interactive classes.

7. Encapsulation

Private Fields: Limited (most classes rely on methods, not fields).

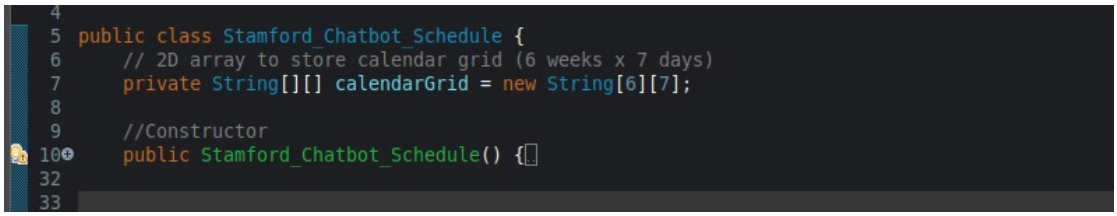
Getters/Setters: Not explicitly needed (data is processed via methods).

8. Object Interaction

Collaboration: Stamford_Chatbot_System.java coordinates interactions between the main class and specalized classes (e.g., CourseFinder.java, CampusMap.java).

9. 2D Arrays

6x7: 6x7 2D Array is implemented in Calendar Logic to present weeks and days.



10. Static vs. Non-Static Methods

Static: Helper methods like getDayOfWeek() in Stamford_Chatbot_Schedule.

Non-Static: Most methods (e.g., findCourse(), displayLibrary()) are instance-based.

11. Commenting & Naming

Comments: Minimal but descriptive (e.g., // Checks if a date is a holiday).

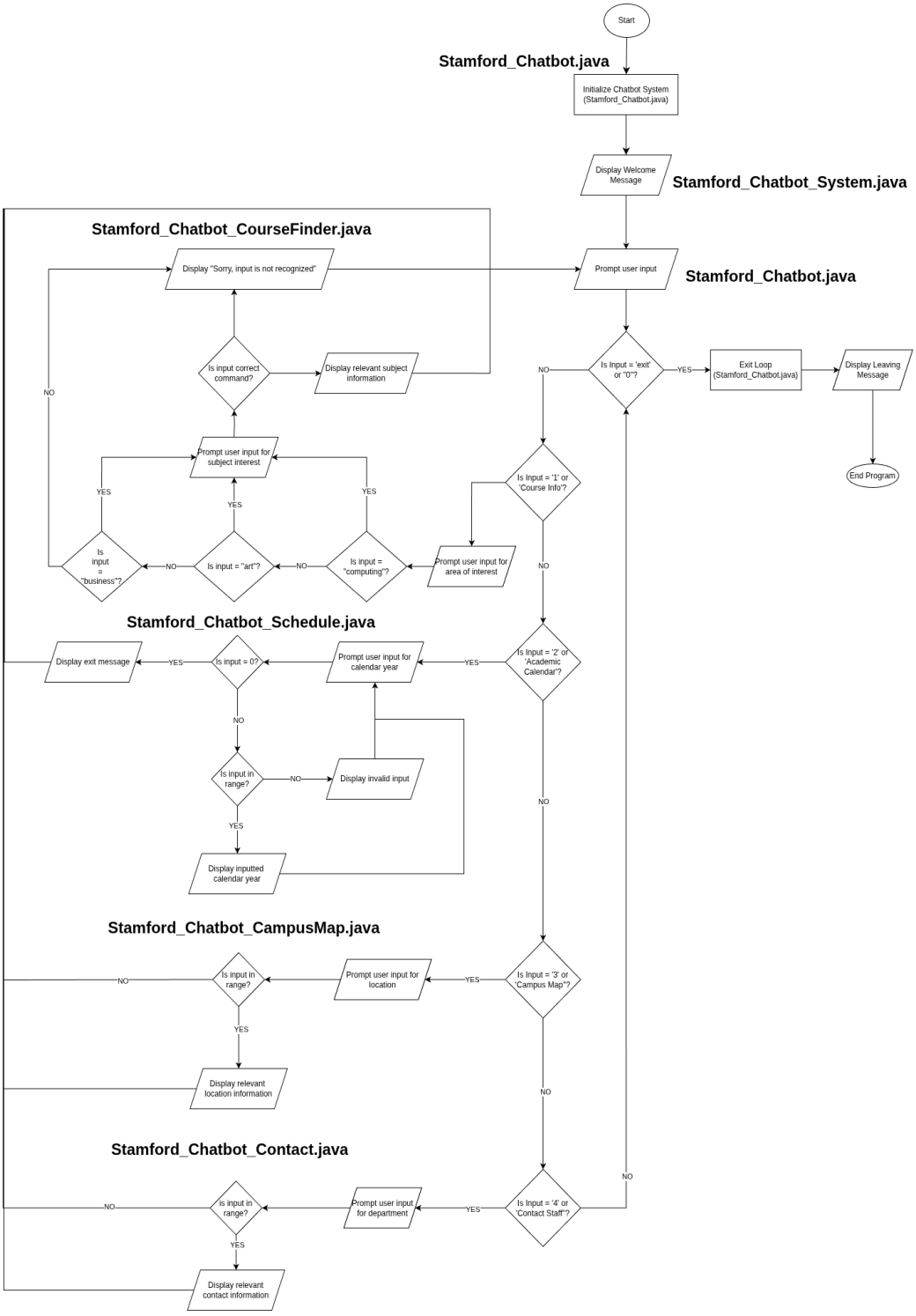
Naming: Follows conventions (e.g., displayCafeteria(), isFakeExamDate).

12. Access Specifiers

public: Methods called across classes (e.g., showSchedule()).

private: Rare (project focuses on method-based encapsulation).

Flowchart


Start
Stamford_Chatbot.java
Initialize Chatbot System (Stamford_Chatbot.java)
Display Welcome Message
Stamford_Chatbot_System.java
Stamford_Chatbot_CourseFinder.java
Display "Sorry, input is not recognized"
Prompt user input
Stamford_Chatbot.java
Is input correct command?
Display relevant subject information
Is Input = 'exit' or '0'?
Exit Loop (Stamford_Chatbot.java)
Display Leaving Message
End Program
Prompt user input for subject interest
Is Input = '1' or 'Course Info'?
Is input = "art"?
Is input = "computing"?
Prompt user input for area of interest
Is input = "business"?
Stamford_Chatbot_Schedule.java
Is Input = '2' or 'Academic Calendar'?
Prompt user input for calendar year
Is input = 0?
Display exit message
Is input in range?
Display invalid input
Display inputted calendar year
Stamford_Chatbot_CampusMap.java
Is Input = '3' or 'Campus Map'?
Prompt user input for location
Is input in range?
Display relevant location information
Stamford_Chatbot_Contact.java
Is Input = '4' or 'Contact Staff'?
Prompt user input for department
is input in range?
Display relevant contact information
YES
NO

Algorithm

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 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 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 7: Repeat from Step 4 until “exit” is entered.

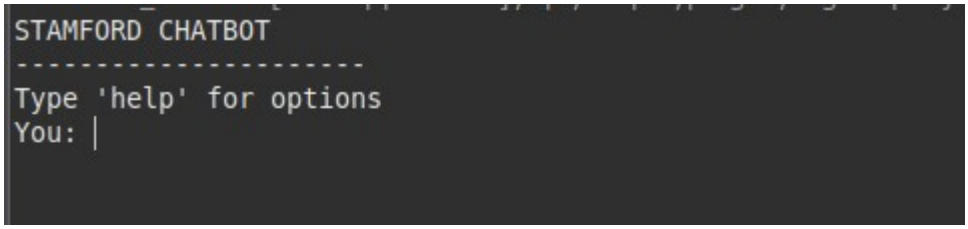
Step 8: Break While Loop

Step 9: Close Scanner.

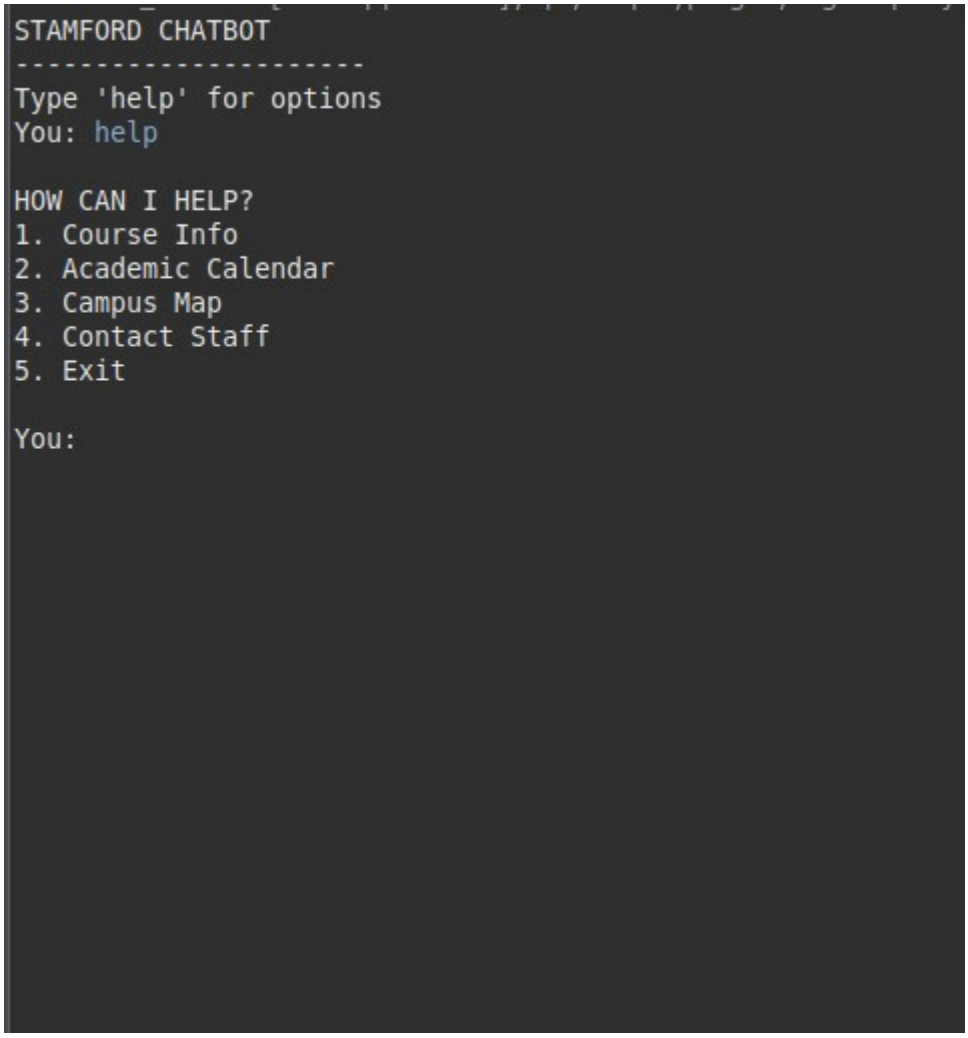
Step 10: Display “Bye! See you again”

Screenshots / Sample Outputs

1. Start



2. User enter ‘help’



3. User enter ‘course info’.

```
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: course info

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

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

```
You: course info

Welcome to the Course Information System!
What area are you interested in?
- Computing
- Art
- Business
Enter your area of interest: computing

Great! You're interested in Computing.
What kind of subject within Computing are you interested in
- Database
- System Analysis
- Network
- Data and Algorithm
Enter a keyword related to the subject you like:
```

3.2 User chose ‘Database’ as a subject.

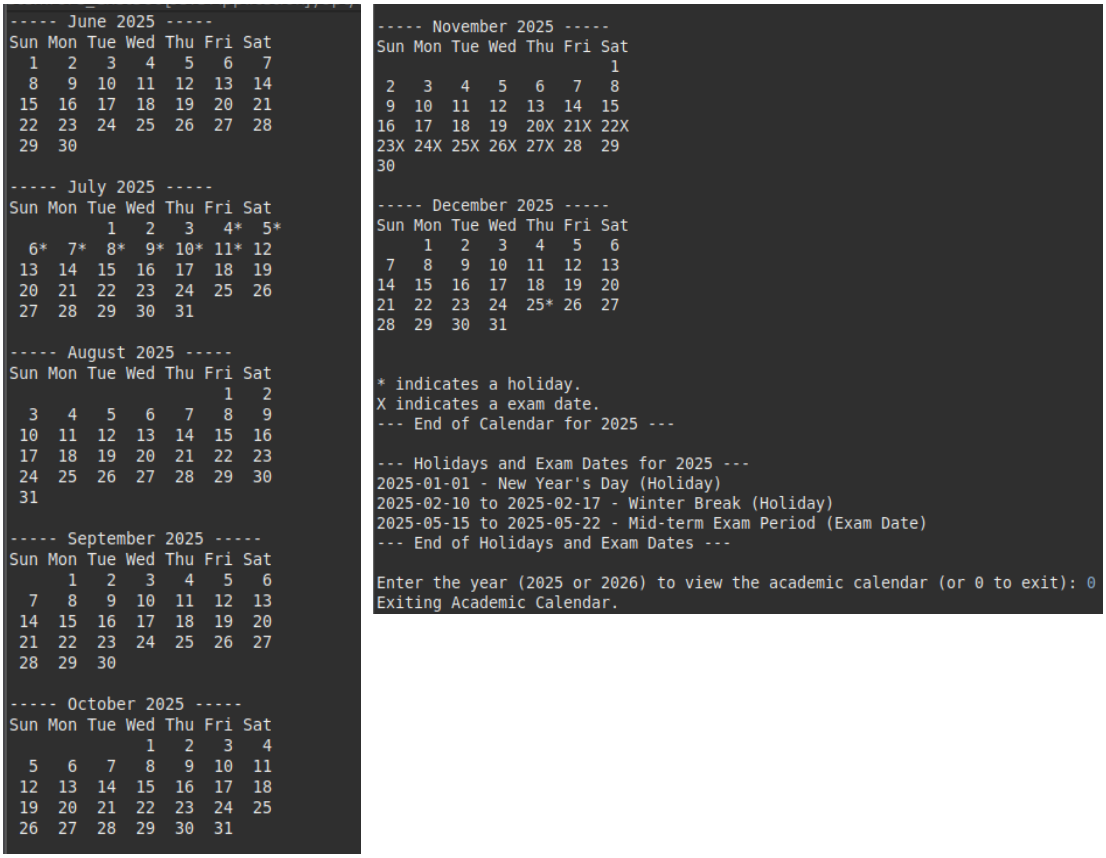
```
Welcome to the Course Information System!
What area are you interested in?
- Computing
- Art
- Business
Enter your area of interest: computing

Great! You're interested in Computing.
What kind of subject within Computing are you interested in?
- Database
- System Analysis
- Network
- 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.
You:
```

4. User enter ‘Academic Calendar’.

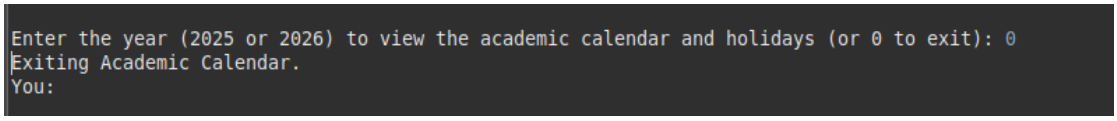
```
You: Academic Calendar

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

4.1 User enter year ‘2025’.

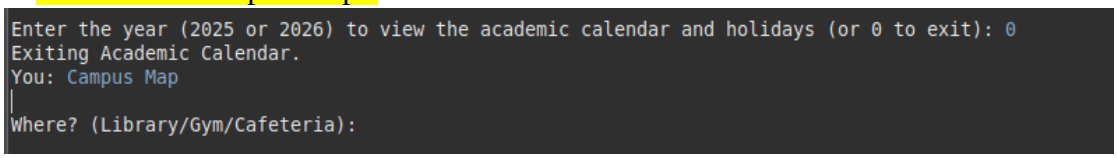
```
Enter the year (2025 or 2026) to view the academic calendar and holidays (or 0 to exit): 2025

--- 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
```



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



5. User enter ‘Campus Map’.



5.1 User finds about ‘Library’.

```
You: Campus Map

Where? (Library/Gym/Cafeteria): Library

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

You:
```

6. User enter about ‘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 User finds about ‘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. User Exit the Program

```
You: exit

Bye! See you again
```

Conclusion

In conclusion, The Stamford Chatbot project successfully demonstrated the design and development of a Java-based console application that aims to enhance student support services. Each component of the chat bot were created in its own dedicated class using a modular and layered design approach. This guarantees clear separation of concerns and allows scalability easily. Additionally, the use of the Spiral Model allowed us to carefully plan, analyze risk, build and test the program thoroughly at each phase. The project consists of core Java concepts such as object instantiation, conditional logic, 2D arrays, static methods and encapsulation. Throughout the development process, we focused on writing clean, maintainable code and structured the system in a way that makes future improvements straightforward. During the review and evaluation phase, we tested each module thoroughly and refined the chatbot based on functionality, user interaction, and code quality. Overall, the Stamford Chatbot not only meets its intended goals but also serves as a strong example of applying object-oriented programming principles to a real-world problem. This project has helped us strengthen our understanding of Java and software design, while also giving us valuable hands-on experience in collaborative development and systematic problem-solving.

References

List any resources or references you used during the development.