Module #3 Report | CSE 310 – Applied Programming

Name	Date	Teacher
Ethan Trent	11/01/25	Bro McGary

Project Repository Link

GitHub Repository

Video Demonstration

[Watch the demonstration video](<https://www.loom.com/share/a13376cfd6e049c3b8882f4050cb6c44)

Module

Mark an **X** next to the module you completed

Module	Language	
Cloud Databases	Java	X
Data Analysis	Kotlin	
Game Framework	R	
GIS Mapping	Erlang	
Mobile App	JavaScript	
Networking	C#	
Web Apps	TypeScript	
Language – C++	Rust	
SQL Relational Databases	Choose Your Own Adventure	

Fill Out the Checklist

Complete the following checklist to make sure you completed all parts of the module. Mark your response with **Yes** or **No**. If the answer is **No** then additionally describe what was preventing you from completing this step.

Question	Your Response	Comments
Did you implement the entire set of unique requirements as described in the Module Description document in I-Learn?	Yes	Implemented chatbot with HashMap knowledge base, ArrayList chat history, TreeSet topic tracking, multiple bot personalities (JokeBot, GameBot, AdviceBot), and all required Java concepts (variables, conditionals, loops, functions, classes, Collections).

Question	Your Response	Comments
Did you write at least 100 lines of code in your software and include useful comments?	Yes	Wrote approximately 738 lines of code across 7 Java files. All classes include comments explaining methods, parameters, and functionality.
Did you use the correct README.md template from the Module Description document in I-Learn?	Yes	README.md follows standard format with project description, features, requirements, usage instructions, project structure, and Java concepts demonstrated.
Did you completely populate the README.md template?	Yes	README.md is fully populated with all sections including project description, features, requirements, how to run, usage examples, project structure, data persistence, and author information.
Did you create the video, publish it on YouTube, and reference it in the README.md file?	Yes	Video is created. Please see README.md for the link or top of this file
Did you publish the code with the README.md (in the top-level folder) into a public GitHub repository?	Yes	Code published to GitHub.

Did you complete a Stretch Challenge

If you completed a stretch challenge, describe what you completed.

Yes, I completed multiple stretch challenges:

1. **File I/O for Persistence**: Implemented file operations to save and load the bot's learned responses to `chatbot_data.txt`. The implementation includes:
    - Error handling with try-catch blocks
    - Chat history saving to `chat_history.txt` with metadata (timestamp, message count)
2. **Object-Oriented Design Patterns**:
    - **Inheritance**: Created a base `Bot` class with core functionality and extended it with specialized bot classes (`JokeBot`, `GameBot`, `AdviceBot`) that inherit shared functionality while adding unique behaviors
    - **Interfaces**: Implemented `Respondable` interface to define a contract for bot personalities, enabling extensible design where new bot types can be easily added
3. **Enhanced Features Beyond Requirements**:
    - GameBot includes progress tracking, guess counting, and duplicate guess prevention
    - Improved input validation with length limits and better error messages

    - Better user experience with detailed feedback and helpful messages

Record your time

How many hours did you spend on this module and the team project this Sprint?
Include all time including planning, researching, implementation, troubleshooting, documentation, video production, and publishing.

	Hours
Individual Module	15
Team Project	5

Retrospective

- What learning strategies worked well in this module?

  Following the planned schedule systematically - starting with core features (HashMap, ArrayList, basic conversation loop) before adding complexity (inheritance, interfaces, file I/O) helped maintain momentum. **Breaking down the project into small, manageable tasks** - one feature per day made the work feel achievable. **Testing incrementally** - after each major feature helped catch issues early. **Reviewing Java Collections documentation** - when implementing HashMap, ArrayList, and TreeSet clarified proper usage and best practices. **Using JavaDoc comments** - from the start made the code more maintainable and helped clarify design decisions.

- What strategies (or lack of strategy) did not work well?

  Underestimating file I/O complexity - handling edge cases (pipe characters in data, file paths, error scenarios) took more time than expected. The original plan assumed simpler file operations. **Not allocating buffer time** - some integration issues between classes took longer to resolve than anticipated. The schedule was somewhat optimistic, leaving little margin for troubleshooting or refactoring. **Waiting until the end to document** - while a README was created, some implementation details could have been documented as features were built rather than all at once.

- How can you improve in the next module?

  Prepare for more buffer time - debugging, testing, and unexpected challenges. **Document as I code** - rather than documenting everything at the end to help catch design issues earlier. **Create video demonstrations earlier** - rather than leaving it to the end. **Start with file I/O earlier** - understanding data persistence constraints earlier would have influenced some design decisions.