Southern Luzon State University
College of Engineering
Computer Engineering Department

CPE18L Software Design

Activity 4: Software Data Modelling

Name: Dave Jhared G. Paduada Date: March 4, 2025

Section: IF Score:_____

1.1 Introduction

Data modeling is the process of **creating a simplified diagram** of a software system and the data elements it contains, using text and symbols to represent the data and how it flows. Data models provide a blueprint for designing a new database or reengineering a legacy application. Overall, data modeling helps an organization use its data effectively to meet business needs for information.

A data model can be thought of as a **flowchart** that illustrates data entities, their attributes, and the relationships between entities. It enables data management and analytics teams to document data requirements for applications and identify errors in development plans before any code is written.

A use case diagram is a dynamic or behavior diagram in UML. Use case diagrams **model the functionality of a system using actors and use cases**. Use cases are a set of actions, services, and functions that the system needs to perform. In this context, a "system" is something being developed or operated, such as a web site. The "actors" are people or entities operating under defined roles within the system.

A flowchart is a **visual representation of the sequence of steps and decisions** needed to perform a process. Each step in the sequence is noted within a diagram shape. Steps are linked by connecting lines and directional arrows. This allows anyone to view the flowchart and logically follow the process from beginning to end. A flowchart is a powerful business tool. With proper design and construction, it communicates the steps in a process very effectively and efficiently.

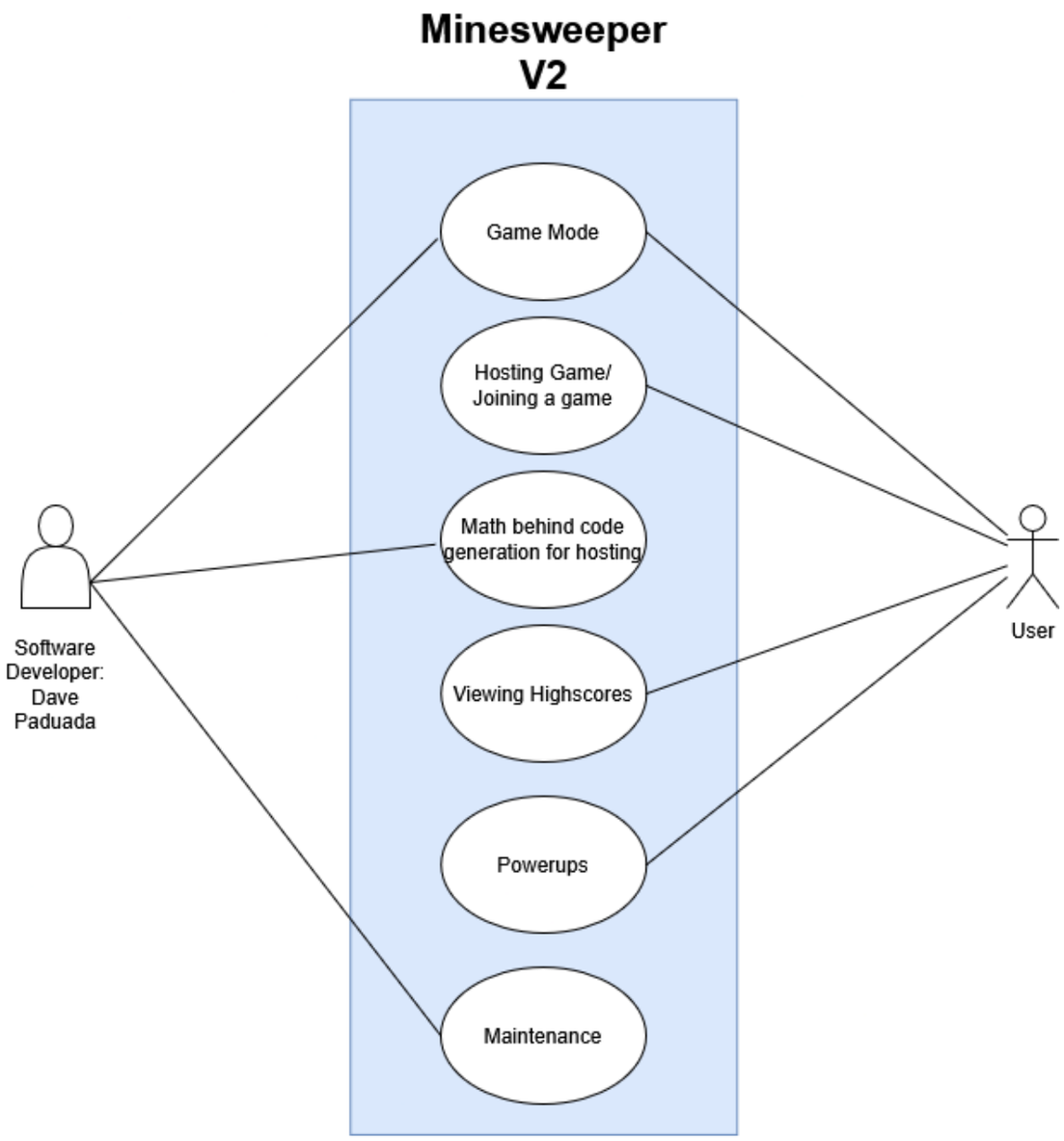
1.2 Objective

- To determine the necessary data to be included in creating an application.
- To create a data model based from previous activities.
- To refine the requirements of the application.

1.3 Activity

Create a Use Case Diagram, Flowchart and Data Flow Diagram of your respective Software.

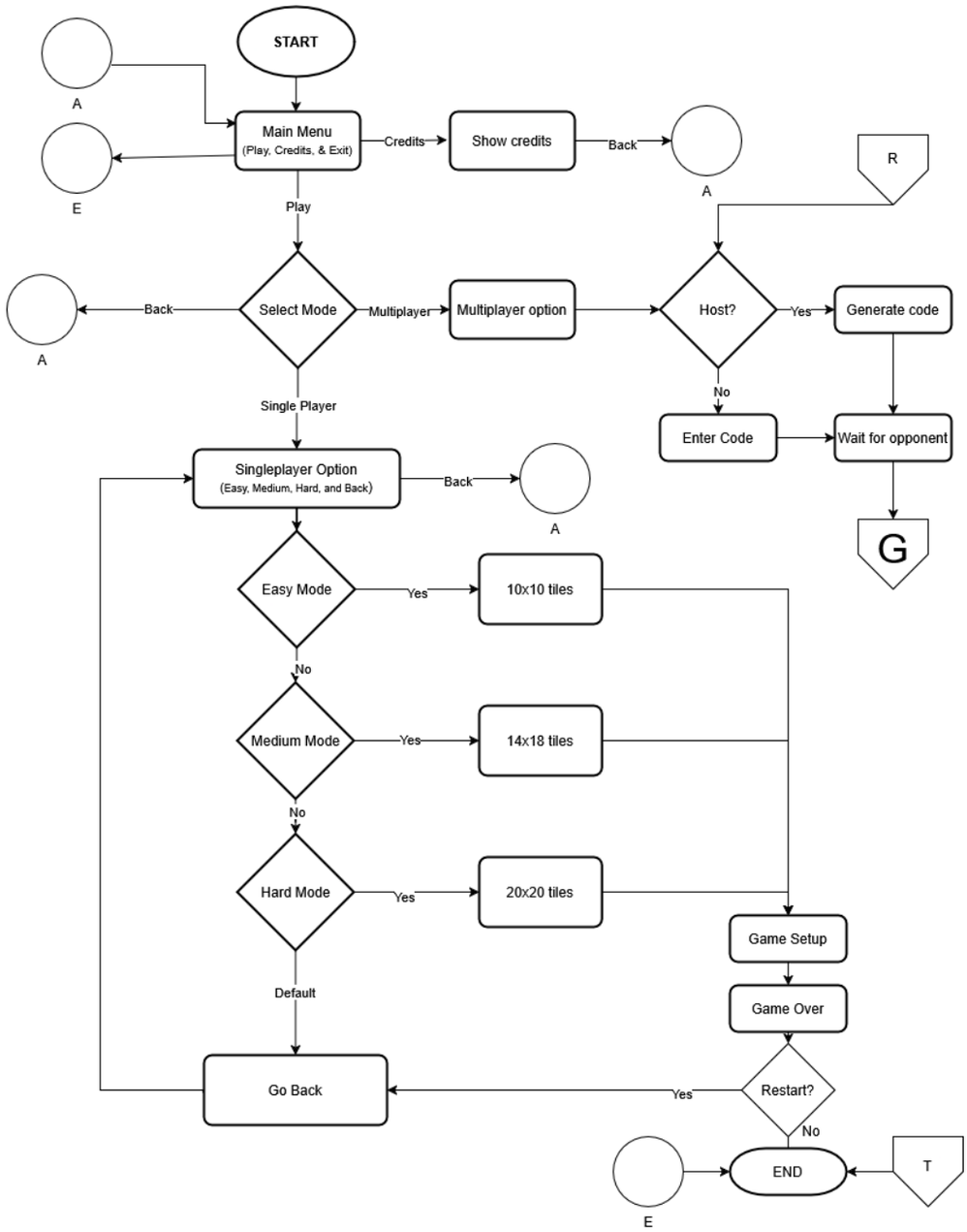
Use Case Diagram:

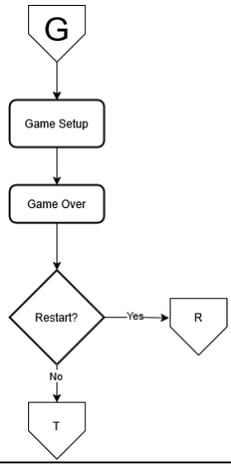


Discussion:

The use case diagram provides a clear visual map of how different users interact with the system, highlighting core functionalities like game mode selection, hosting, and joining games. It outlines the roles of various actors, such as the player and potentially an admin, and details the main actions they can perform. This diagram helps developers understand system requirements and ensures that all user interactions are accounted for in the design process. It also simplifies complex processes into an accessible format, bridging the gap between technical details and user experience. Overall, it serves as a foundational tool in both designing and communicating the software's intended behavior.

Flowchart:


START
A
E
Main Menu
(Play, Credits, & Exit)
Credits
Show credits
Back
A
R
Play
Select Mode
Back
A
Multiplayer
Multiplayer option
Host?
Yes
Generate code
No
Enter Code
Wait for opponent
G
Single Player
Singleplayer Option
(Easy, Medium, Hard, and Back)
Back
A
Easy Mode
Yes
10x10 tiles
No
Medium Mode
Yes
14x18 tiles
No
Hard Mode
Yes
20x20 tiles
Default
Game Setup
Game Over
Go Back
Yes
Restart?
No
E
END
T



Discussion:
The "Start" node at the top of the flowchart opens a Main Menu process with the choices to Play, see Credits, or Exit. A decision node prompts the user to pick Single Player, Multiplayer, or Exit when the Play option is selected. A switch-case selection decides the difficulty—Easy (10x10 tiles), Medium (14x18 tiles), or Hard (20x20 tiles)—in Single Player mode. The player then enters a "Single Player Options" procedure, which leads to the Game Setup process and ultimately a Game Over. The player is asked to restart after the game finishes in Single Player mode; an affirmative answer takes them back to the Single Player Options, and a negative answer kills the application. A decision node in Multiplayer mode asks the player if they want to host a game. If they say yes, the flowchart generates a code and waits for an opponent; if they don't, they can enter an existing code. Both choices result in Multiplayer Game Setup, Game Over, and a final decision about whether to restart or quit.

1.4 Questions

1. What information and ideas does a flowchart brings in a software development?
Flowcharts visually outline the control flow and logic of a program, representing processes, decision points, and the sequence of operations. They help developers understand, design, and communicate complex algorithms and workflows. This visual representation makes it easier to identify inefficiencies, potential bugs, or areas for optimization. Additionally, flowcharts serve as a common language for both technical and non-technical team members during early development stages.

2. What are the significance of having a Use Case Diagram?
Use Case Diagrams illustrate how users (or other systems) interact with the software, capturing functional requirements and defining system boundaries. They help stakeholders visualize the interactions, roles, and functionalities of the system in a simple, intuitive manner. By highlighting key system features and actor interactions, these diagrams provide a roadmap for what the software should accomplish, ensuring all user needs are addressed. They also serve as a foundation for further detailed design and documentation.

3. Do you think that flowchart and use case diagram is enough in modelling your software? Why or why not?
Although they include high-level procedures and user interactions, flowcharts and use case diagrams offer a strong basis; nonetheless, they are usually insufficient for comprehensive software modeling on their own. To properly depict system architecture, object interactions, dynamic behaviors, and data flow, more intricate diagrams—such as class diagrams, sequence diagrams, state diagrams, and activity diagrams—are typically needed. By eliminating ambiguity and enhancing the overall quality of the design, these extra models aid in making sure that the system's behavior and structure are thoroughly described.

1.5 Conclusion
In conclusion, this project has enabled us to determine the essential data required for creating Minesweeper V2, ensuring that all game mechanics and user interactions are accurately captured. By drawing on insights from our previous activities, we developed a robust data model that maps out the relationships between various game elements, from tile interactions to player actions. This model serves

as a blueprint for the application's architecture, aligning closely with our refined requirements. Ultimately, the process of identifying, modeling, and refining these requirements not only streamlines the development process but also enhances the overall design, ensuring a comprehensive and engaging user experience.