**PROJECT for Software Engineering, Summer 2022**

**Wandering in the Woods Game**

Sandeep Perikala

Naveen Madupoju

Arshiya Nooreen

Praneeth Somisetty

Khaja Ahmed Hussain Syed

Table of Contents

[**1.0 Introduction** 3](#_Toc112192332)

[**1.1 Purpose** 3](#_Toc112192333)

[**1.2 Wandering in the Woods Game** 3](#_Toc112192334)

[**2.0 Process Model** 3](#_Toc112192335)

[**3.0 Use Case** 5](#_Toc112192336)

[**3.1 Use Case 1: Start Game** 6](#_Toc112192337)

[**3.2 Use Case 2: End Game** 6](#_Toc112192338)

[**3.3 Use Case 3: Wander** 7](#_Toc112192339)

[**3.4 Use Case 4: Set up size of grid** 7](#_Toc112192340)

[**3.5 Use Case 5: Run Experiments** 8](#_Toc112192341)

[**3.6 Use Case 6: Exploring Different Protocols** 9](#_Toc112192342)

[**3.7 Use Case 7: Display Statistics** 10](#_Toc112192343)

[**3.8 Use Case 8: Play Music** 10](#_Toc112192344)

[**4.0 UML Model** 11](#_Toc112192345)

[**4.1** **Use Case Diagram** 11](#_Toc112192346)

[**4.2** **Deployment Diagram** 12](#_Toc112192347)

[**4.3** **State Diagram** 13](#_Toc112192348)

[**4.4** **Activity Diagram** 14](#_Toc112192349)

[**5.0** **Customer Journey Map** 16](#_Toc112192350)

[**6.0** **Personas** 16](#_Toc112192351)

# **1.0 Introduction**

## **1.1 Purpose**

This document will serve as basis for software design of Wandering in Woods Game (WWG). The specifications given for case study will be analyzed and will be presented in documentation. We will adopt agile way for purpose of this documentation. Agile documentation, according to SEPA, is the smallest amount of descriptive data needed to ensure that the software development team, management, and other stakeholders have a complete grasp of the requirements and the program to be created. In keeping with agile principles, only the content that the design team believed required to be retained and maintained is included in this document. We consider this document to be an ongoing work and a version of the ultimate design. The document and framework are open to criticism and recommendations.

We believe that this document will be helpful for educators who want to learn more about the concepts behind computer simulations (e.g., game). The goal of this prototype is to teach K–8 kids computational thinking, arithmetic principles, and computer programming. Additionally, this paper will be helpful for students who want to examine a more in-depth version of the simulation.

## **1.2 Wandering in the Woods Game**

Wandering in the woods is a game which is a rather straightforward visualization intended to explain the fundamental concept to kids in grades K-2 (and hopefully to amuse kids); kids would select and watch what happens. We intend for levels 3 to 5 kids to deal with individual information generating activities, or "little data," in a somewhat more challenging form in which they decide things, view information, and face extra challenges. There will be a more complicated edition for grades 6 to 8, where children participate with small and large data, charts are produced, and learners decide things that mostly transform the data.

# **2.0 Process Model**

For the development of game, the software development team will use Feature-Driven Development (FDD) process model. It is an agile software development process that focuses on the needs of the customer and emphasizes frequent and fast delivery of working software. In Agile, FDD supports regular status monitoring to focus attention on development and evaluate performance. FDD enables teams to rapidly detect flaws and implement fixes with frequent updates to the project. In addition, information and significant outcomes may be made available to customers at any time. However, features in the FDD context may not always correspond to the conventionally known notion of a product's characteristics. In fact, they more closely resemble Scrum's user stories. What this means is that "deliver the registration process" might be a feature in the Feature Driven Development (FDD) framework. The FDD method is structured in five stages, each of which is based on individual "feature" projects. This is the project lifecycle:

1. Create a comprehensive plan
2. Create a list of functionalities
3. Create a strategy based on features
4. Feature-based design
5. Systematical feature-based design

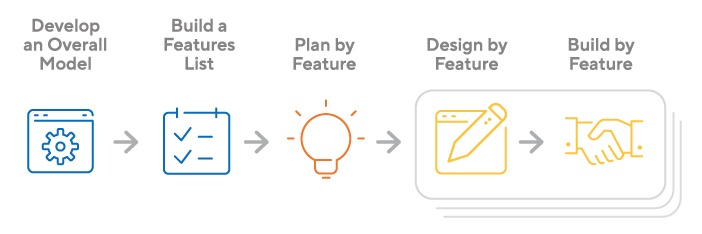


Figure 1: WWG system FDD process model diagram.

# **3.0 Use Case**

The term "use case" in the fields of software and systems engineering, have two distinct meanings.

* One or more possible ways in which a piece of software might be put to work; usually spoken about in the plural.
* A possible case when an internal system processes an external request for example taking input from user and acts accordingly.

For the purposes of the Unified Modeling Language (UML), a role (sometimes called an actor) interacts with a system via a series of use cases to accomplish a certain task. Writing use cases is a common task for business analysts, and these documents are useful throughout the software development lifecycle. They may be used for anything from laying the groundwork for a system's needs to verifying the design to conducting quality assurance tests. Use cases are a way to communicate the needs of users and developers alike clearly and concisely about the functionality of a system.

The fundamental required data for the delivery of a functioning prototype were identified by the team and the customer as the following six use cases. It's important to remember that as the project proceeds, more use cases may be readily included into the existing functionality to provide a more robust foundation.

There will be four types of actors.

1. K-2 grade student
2. Grade 3-5 student
3. Grade 6-8 student
4. System

There are three types of students, so the use cases for each student will be different. There are 3 use cases which are same for each student. These use cases are described below:

## **3.1 Use Case 1: Start Game**

|  |  |
| --- | --- |
| **Use Case Name** | U001: Start Game |
| **Summary** | The student will launch the game and select the level of game from three options. The options are grade k-2, grade 3-5, grade 6-8. Then student will start playing game. |
| **Actor** | Student |
| **Pre-Condition** | The game must be installed and show the main menu to the student. |
| **Flow of events** | 1. Student installs the game. 2. Student opens the game 3. Student selects the level. 4. Student plays the game. |
| **Post- Condition** | The system must navigate the student to the playing screen after selecting the level. |
| **Acceptance Criteria** | The student can play game with different levels. |

## **3.2 Use Case 2: End Game**

|  |  |
| --- | --- |
| **Use Case Name** | U002: End Game |
| **Summary** | The student will end game depending upon the situation whether the game is over, or the student quits the game. |
| **Actor** | Student |
| **Pre-Condition** | The game must be installed and show the main menu to the student. |
| **Flow of events** | 1. Student installs the game. 2. Student opens the game 3. Student selects the level. 4. Student plays the game. 5. Student ends the game. |
| **Post- Condition** | The student must play the game. |
| **Acceptance Criteria** | The student must install and play the game to select end game option. |

The use cases for student of grade K-2 are describe below:

## **3.3 Use Case 3: Wander**

|  |  |
| --- | --- |
| **Use Case Name** | U003: Wander |
| **Summary** | The student will wander in diagonally opposite corners of the grid. They can wander randomly. Each move will be counted. In case they bump into another student, there will be a happy display and statistics will be displayed. |
| **Actor** | K-2 Grade Student |
| **Pre-Condition** | The student must be in a K-2 grade. |
| **Flow of events** | 1. Student installs the game. 2. Student opens the game 3. Student selects the level. 4. Student plays the game. 5. Student start moving position in diagonally opposite corners of grid. 6. Student bumps into another student. 7. The system displays happy graphics. 8. The system displays the statistics. 9. The game resets. 10. The student may restart game. |
| **Post- Condition** | The system must display happy graphics in case of bumping of two students in a grid. The system must display the statistics. |
| **Acceptance Criteria** | There must be two students and students must be in K-2 grade. |

The use cases for grade 3-5 are described below:

## **3.4 Use Case 4: Set up size of grid**

|  |  |
| --- | --- |
| **Use Case Name** | U004: Set up size of grid |
| **Summary** | The student will set up the size of grid which can be rectangular. The student will place the players anywhere on the grid. The statistics will be shown by system. |
| **Actor** | Grade 3-5 Student |
| **Pre-Condition** | The student must be in a grade 3-5. |
| **Flow of events** | 1. Student installs the game. 2. Student opens the game 3. Student selects the level. 4. Student plays the game. 5. Student set up the size of grid. 6. Student places the players on the grid. 7. Student replays the game. 8. Student bumps into another student. 9. The system displays happy graphics. 10. The system displays the statistics including long run, short run, and average run. 11. The game resets. 12. The student may restart game. |
| **Post- Condition** | The system must display happy graphics in case of bumping of two students in a grid. The system must display the statistics. |
| **Acceptance Criteria** | There must be at least two students and students must be in 3-5 grade for changing size of grid or placing players anywhere on the grid. |

The use cases for grade 6-8 are described below:

## **3.5 Use Case 5: Run Experiments**

|  |  |
| --- | --- |
| **Use Case Name** | U005: Run Experiments |
| **Summary** | The student will set up the size of grid which can be rectangular. The student will run experiments to see how the size and shape of grid affects the average run. |
| **Actor** | Grade 6-8 Student |
| **Pre-Condition** | The student must be in a grade 6-8. |
| **Flow of events** | 1. Student installs the game. 2. Student opens the game 3. Student selects the level. 4. Student plays the game. 5. Student set up the size of grid. 6. Student places the players on the grid. 7. Student changes size of grid each time. 8. Student changes position of players each time. 9. Student replays the game. 10. Student bumps into another student. 11. The system displays happy graphics. 12. The system displays the statistics of average run according to size and shape of grid. 13. The game resets. 14. The student may restart game. |
| **Post- Condition** | The system must display happy graphics in case of bumping of two students in a grid. The system must display the statistics according to size and shape of grid. |
| **Acceptance Criteria** | There must be at least two students and students must be in 6-8 grade for changing size of grid or doing different experiments for determining the average run. |

## **3.6 Use Case 6: Exploring Different Protocols**

|  |  |
| --- | --- |
| **Use Case Name** | U006: Exploring Different Protocols |
| **Summary** | The student will explore various protocols for wandering in game and will decide which path to wander to shorten the amount of time it takes to bump each other. |
| **Actor** | Grade 6-8 Student |
| **Pre-Condition** | The student must be in a grade 6-8. |
| **Flow of events** | 1. Student installs the game. 2. Student opens the game 3. Student selects the level. 4. Student plays the game. 5. Student set up the size of grid. 6. Student places the players on the grid. 7. Student changes size of grid each time. 8. Student changes position of players each time. 9. Student replays the game. 10. Student choses different protocols. 11. Student bumps into another student. 12. The system displays happy graphics. 13. The system displays the statistics of average run according to size and shape of grid. 14. The game resets. 15. The student may restart game. |
| **Post- Condition** | The system must display happy graphics in case of bumping of two students in a grid. The system must display different protocols to wander. The system must display the statistics according to size and shape of grid. |
| **Acceptance Criteria** | There must be at least two students and students must be in 6-8 grade for selecting different protocols to wander. |

## **3.7 Use Case 7: Display Statistics**

|  |  |
| --- | --- |
| **Use Case Name** | U007: Display Statistics |
| **Summary** | The system will display statistics when two students bump into each other. |
| **Actor** | System |
| **Pre-Condition** | At least two students bump into each other while wandering. |
| **Flow of events** | 1. The students play game 2. The students bump into each other. 3. The system displays statistics according to grade of students. |
| **Post- Condition** | The system must display happy graphics in case of bumping of two students in a grid. The system must display statistics. |
| **Acceptance Criteria** | Two students must bump into each other. |

## **3.8 Use Case 8: Play Music**

|  |  |
| --- | --- |
| **Use Case Name** | U008: Play Music |
| **Summary** | The system will play music as game starts. |
| **Actor** | System |
| **Pre-Condition** | The student must start the game. |
| **Flow of events** | 1. The students play game 2. The students wander in the game. 3. The system plays music |
| **Post- Condition** | The system must play music as students wander in the game. |

# **4.0 UML Model**

## **Use Case Diagram**

What the system would do to meet a business goal is defined by use cases. To achieve this goal, the use-case details the external actors (or entities) that interact with the system and how they do so. When creating a new piece of software, the main method of documenting the system/software requirements is using a UML use case diagram. As opposed to describing how something should be done, use cases describe what ought to work out that way instead (how). After been defined, use cases may be represented either in text or graphic form (i.e., use case diagram). One of the basic ideas behind use case modelling is that it facilitates system design from the user's point of view.

Use case diagram for the six use cases is shown in Figure 2. The student who interacts with the system to play the game is the lone player. The student 's grade level determines the difficulty of the game they may access.

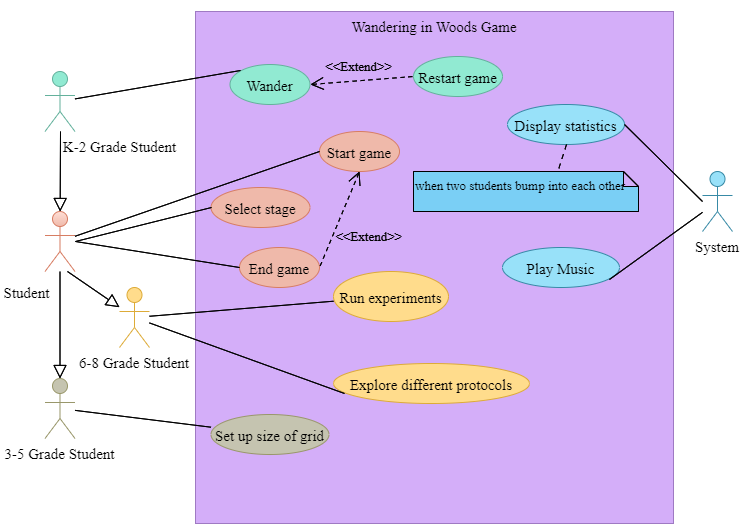


Figure 2: Use case diagram for WWG system.

## **Deployment Diagram**

The deployment diagram is shown in Figure 3. A single server, a single client for the instructor, and a single client for the student comprise up the system. The server is responsible for keeping score, controlling the movement through the game's stages, and communicating to and from the clients. Java SE and MySQL will be used to build the server side of the system, which was then seamlessly linked with an Apache web server.

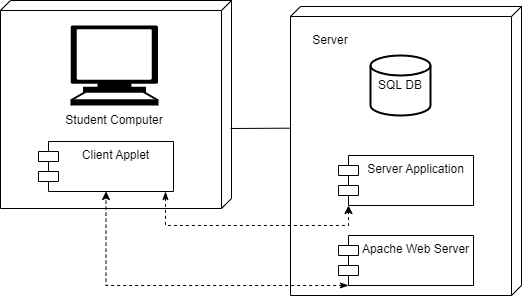


Figure 3:Deployment diagram for WWG system.

## **Class Diagram**

Class diagram is used to show the static structure or architecture of a system in UML. The classes, properties and operation which are also called methods are listed and how the classes relate to other classes are shown. The class of the system is shown in Figure 4.

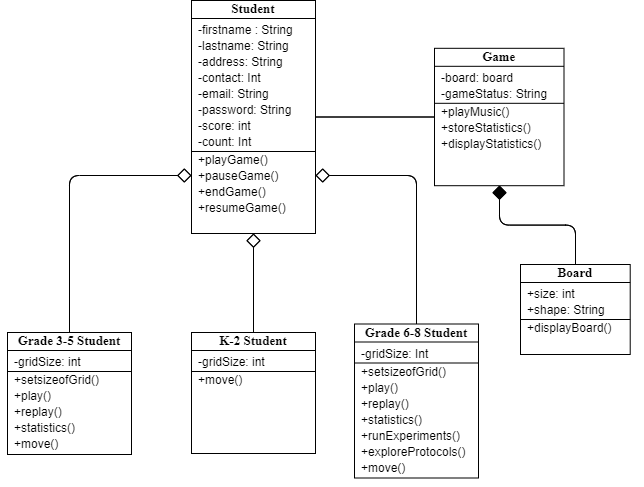


Figure 4: Class Diagram of WWG system.

**Classes:**

1. **2Student:** the main class of system.

**Attributes:**

* **firstname:** the first name of the student
* **lastname:** the last name of the student
* **address:** the address of the student
* **contact:** the contact of student
* **email:** the email id of student for authentication
* **password:** a password for protection of student’s information
* **score:** the score of students after playing game which is statistics of each student.
* **count:** the count for each move of student during game

**Operations:**

* playGame(): used to invoke method of playing game.
* pauseGame(): used to invoke method of pause game.
* endgame(): used to invoke method of end game.
* resumeGame(): used to invoke method of resume game.

1. **Grade K-2 student:** the students who are in grade 2. As there are different functionality for each player, so the class is made separately.

**Attributes:**

As the class is inherited from Student class so attributes will be same.

**Operations:**

**move():** as students in K-2 level can move only in diagonal direction so this method will be used to store the dimension.

1. **Grade 3-5 student:** the students who are in grade 3-5. As there are different functionality for each player, so the class is made separately.

**Attributes:**

As the class is inherited from Student class so attributes will be same.

**Operations:**

* **setsizeofGrid():** the students of 3-5 level grade can change the size of grid.
* **replay():** the students in 3-5 level grade can replay game multiple times:
* **move():** the students in 3-5 grade can move in any direction and can place players anywhere on grid.

1. **Grade 6-8 student:** the students who are in grade 6-8. As there are different functionality for each player, so the class is made separately.

**Attributes:**

As the class is inherited from Student class so attributes will be same.

**Operations:**

* **setsizeofGrid():** the students of 6-8 level grade can change the size of grid.
* **replay():** the students in 6-8 level grade can replay game multiple times:
* **move():** the students in 6-8 grade can move in any direction and can place players anywhere on grid.
* **runExperiments():** the students in grade 6-8 can run multiple experiments for average run estimation.
* **exploreProtocols():** the students in grade 6-8 can explore multiple protocols for wandering for selecting best path to shorten time.

1. **Board**: the class which is used to set size and shape of grid.

**Attributes:**

* **Size:** the size of grid which will depend on the grade level of students.
* **Shape:** the shape of grid which will depend on the grade level of students.

**Operations:**

* **displayboard():** the method used to display grid to students.

1. **Game**: the class for storing the attributes and functions related to the game.

**Attributes:**

* **board:** the object type attribute which will retrieve the values from board class.
* **gameStatus:** the attribute used to store status of game such as play, pause, resume, and end.

**Operations:**

* **playMusic ():** the method used to play music during game.
* **storeStatistics():** the method used to store statistics of each student.
* **playMusic ():** the method used to display statistics of each student.

## **State Diagram**

Figure 5 is a state diagram of the system, depicting its overall behavior. Within its usual operating range, the system will transition between eight distinct phases.

The system will start in **Start State** where a student selects the game. This state will transition from **Start State** to **Initialize State**. The game will load. Then student starts a game. The state will be changed to **Ready State.** The student will select the level and will start playing. The state will change to **Playing State**. Here student can either stop the game or pause the game. If student pauses the game the state will change to **Pause State**. The student can resume the game and the state will again change to **Ready State.** If student selects to stop game the state will change to **End Game** State. During the pause, student can also end the game. When the student closes the window, the game will be destroyed, and state will change to **Destroy State**. The system will store all the statistics automatically.

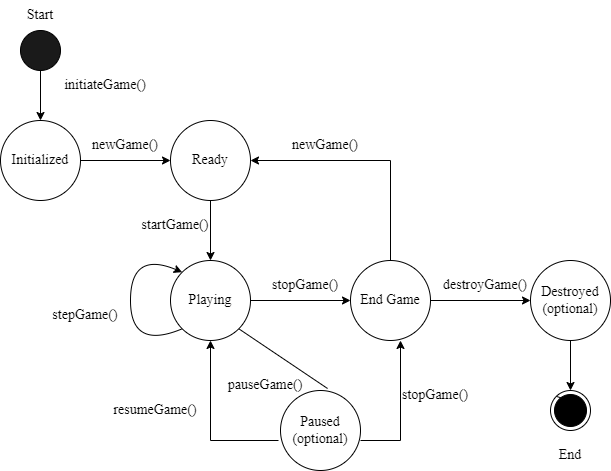


Figure 5:State diagram for WWG system.

## **Activity Diagram**

Like a flowchart or data flow diagram, an activity diagram graphically depicts the steps involved in a process or the direction in which data is being processed inside a system. In the realm of business process modelling, activity diagrams are a common technique. As an alternative, a use case diagram may be used to explain the process. Both sequential and concurrent processes may be described.

The activity diagram of system is shown in Figure 6. The student will open the game and system will show the main menu. The student will select the play menu. The system will show game play menu. There will be three options for student. Start game, select level, and continue game. The start game will start the game and student will start playing. If student will select the level, then the requested level will be shown, and student will play that level. If student will select continue game option, the game will start from the pause state where student had left the game. When student plays game, system will check whether the students have completed the game then system will show the happy graphic displays and will ask the student whether he/she want to play game again. If student will select yes, then system will show levels to student again to select and play. If student selects no, the system will end the game and return the student to main screen.

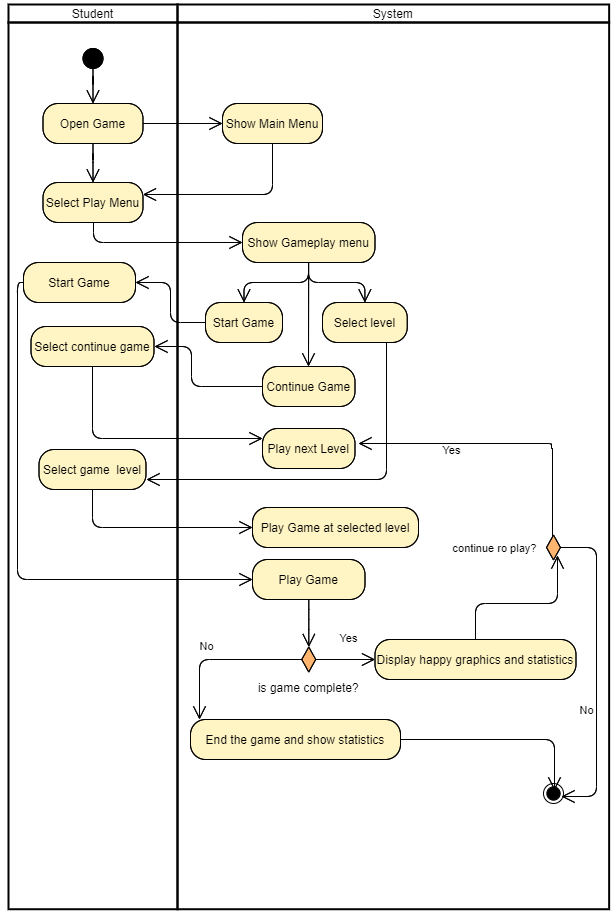


Figure 6:Activity diagram for WWG system.

# **Customer Journey Map**

A customer journey map is a basic concept: it's a graphic that shows the many interactions a client has with our business, whether they're purchasing something from us online, at a store, or using one of our services. The goal of student is to enhance computational thinking and basic concepts of math, and computer science. The game will be designed as a s simulation to provide the students an opportunity to select the level depending upon the grade and play the game while system will present different challenges.

We must remember that when the prototype undergoes additional testing and development, it may reveal previously unknown dangers to the client, which may need revisions to this document.



Figure 7: Customer Journey Map for WWG system.

# **Personas**

Figure 8 is an example of a typical user profile for the WWG system. In this case, the team has given a single identity with the expectation of creating more ones later. The team has selected a representative sample of prospective customers for use in the project's evaluation. For example, there is student named John try to play the game for first time, but he finds It difficult to play as he is new to technology. The designers should consider about including the user manuals and suggestions in the game which may pop up on screen from time to time to guide John while playing.

|  |  |  |
| --- | --- | --- |
|  | **Name:** John  **Age**: 18  Profession: Student  **About:** Student in a college just after passing school in a small town.  **Goals:**  Wants to enhance computer skills. | Knows to run pc in home but has no experience in playing math- based game. Wants to pass the college with good marks in math and computer science. |

Figure 8: Sample persona of a student for evaluating WWG system.