

DUNGEONS & DRAGONS ADVENTURE MANAGER

Created by: Alessandro Perini (S1104314), Luca Severini(S1107253), Tommaso Pieroni.

SOFTWARE DESCRIPTION:

DUNGEONS AND DRAGONS MANAGER:

WHAT IS THE SYSTEM ABOUT?

The System provides the User with a Software capable of providing the functionalities required for playing the famous role play game "Dungeons & Dragons". Through the User Interface built for providing flexibility users will be able to create their own story. They will be provided with a selection of different fields, which will allow them to craft their own experience and progress through the story. To better understand why flexibility is important it is crucial to first explain what Dungeons & Dragons is about. It is a role play game played with multiple people where every single player makes his own choices and engages in fights. The story itself develops thanks to the players' choices and, more importantly thanks to the choices of the Dungeon Master, who is the main director of the story and the one who hosts the game. He has the power to add new Monsters and NPCs in the map and make them interact with the other characters.

HOW DOES THE SYSTEM WORK?

Now that the System functionalities have been explained it's time to better understand what the system does and how the user experience is crafted. Every user will be able to play "Dungeons & Dragons" only after having made a successful Login. If the User is not logged and does not have a correct user name and password the System will not allow them to Create or Join any Adventure. If the User does not have any login credentials it means he has not registered yet. The User can register by clicking on a specific button and he will be redirected to a page where he can insert his Username, Password and Email. After registering or logging in the User will be able to either create or join an Adventure, which is basically a Dungeons & Dragons' match.

Every player (except the Dungeon Master) will be able to create his own character with a specific Character Creation tab. After doing so he will finally be playing in the adventure. In the game the Dungeon Master will be able to see the map, allocate new Monsters or NPCs in the map, and decide their states.

Characters, NPCs and Monsters can engage in combat thanks to the DM who provides a targeting functionality.

Normal Players and the Dungeon Master will be provided with a chat that allows them to communicate with each other.

All these functionalities put together will allow players to craft and progress through their own story, creating an organic and dynamic environment that suits the Dungeon Master narration and leaves players with certain degrees of freedom.

DETAILED SOFTWARE DESCRIPTION

In this section all the pages and interfaces the User finds will be explained in Detail.

PAGES AND BUTTONS:

LOGIN INTERFACE:

It is the first page the User sees after starting the Software. It is the interface where the User can Login thanks to the insertion of Username and Password Credentials. If the User does not have any credentials he can be redirected to the Registration Page.

REGISTRATION PAGE:

It is the page where the User can put his credentials for the first time (Username, Email and Password), after which they will be registered in the Database. Some of these credentials must be inserted by the User during the login process to be able to play the Game.

MENU PAGE:

In the Menu page the User will have the possibility to use certain buttons to Create a new Adventure, load an adventure he previously played on, delete an adventure, join an existing adventure created by another user, Logout or Exit the Game.

CREATE NEW ADVENTURE PAGE:

In this page the User will be able to give a name to the adventure he wants to create.

LOAD ADVENTURE BUTTON:

In the Menu page the User can select one adventure he previously played on and then load it by clicking on the load button.

DELETE ADVENTURE BUTTON:

In the Menu page the User can select one adventure he previously played on and then delete it by clicking on the Delete button.

JOIN ADVENTURE:

The User can join an existing adventure created by another User by inserting the name of the Adventure he wants to join.

CHARACTER CREATION SECTION:

In this series of pages a User who joins an adventure can create his own character by filling all the requested fields.

D&D PLAYABLE INTERFACE:

The playable interface is composed of different TABs. What certain players can and cannot do depends on whether they are Dungeon Master or not.

MAIN TAB: In this interface Players can see the Chat, send messages, throw dice and see the result displayed in the chat and view the Map.

CHARACTERS TAB:

In the Characters Tab the User can see his own character and the Character of other players in the Adventure. Every character has a Propic (Profile Picture), a Name, an Alignment, a Size, a Class, a Level, a Race, a Player Name, an Armor Class Level (AC), Speed Level, Hp, MAX Hp, Strength, Dexterity, Constitution, Intelligence, Wisdom, Charisma, Attacks, Equipment, Skills and Extras (some additional information about the character which is functional for narrative purposes).

The characters list appears on the left and every character on the list can be potentially edited by everyone or deleted only by the DM.

Characters are placeable in the map.

BESTIARY TAB:

In the Bestiary Tab the Dungeon Master can delete, create or edit a Monster. To do so he has to edit or to insert the following fields: Propic (Profile Picture), Name, Alignment, Size, Hp, AC, Speed, Strength, Dexterity, Constitution, Intelligence, Wisdom, Charisma, Attacks, Equipment, Skills, Extras.

Every monster is selectable and listed on the left.

Monsters are placeable in the map.

NPCs TAB:

In the NPCs Tab the Dungeon Master can Delete, Create or Save a new NPC. To do so he has to edit or to insert the following fields: Propic, Name, Alignment, Size, Hp, Armor Class, Speed, Strength, Dexterity, Constitution, Intelligence, Wisdom, Charisma, Attacks, Equipment, Skills, Extras.

Every NPC is selectable and listed on the left.

NPCs are placeable in the map.

ITEMS TAB:

In the Items Tab the Dungeon Master can delete, create and save new Items. To do so he has to edit or to insert the following fields: Propic, Name, Description, Potion, Spell, Armor, Weapon, Effect, Casting Time, Casting Type, Damage Type, Damage Dice, CA, Extras, Resistances.

Every Item is selectable and listed on the left.

Items are placeable in the map.

MAPS:

In the Maps Tab the Dungeon Master can Delete, Create, Save and Load Maps. To do so he has to edit or to insert the following fields: Name, Width, Height, Weather.

Every map is selectable and listed on the left.

Once a Map is loaded the DM can decide to put Monsters and NPCs in it.

GLOSSARY:

TERMS	DESCRIPTION
Player	User registered to the game and who is allowed to Create or take part in Adventures along with other players.
Battle	Specific moment during the adventure when Characters and Monsters fight each other.
Character	Creatable Entity made by the Player at the start of the adventure and usable by such player throughout the course of the game.
Adventure	Dungeons and Dragons match that can be created by a player and which other players can take part in. The player who creates the adventure Automatically becomes "Dungeon Master".
DND	Abbreviation for "Dungeons and Dragons": well known role-playing analog game, reinterpreted in digital form by our Software "D&D Adventure Manager".
Dungeon Master (DM)	Player who creates the Adventure, who decides how the story progresses and when certain events take place.
Map	Representation of the Gameworld viewable on players' interface and made of a certain number of tiles of different types. The map gives a general overview of the development of the adventure and of the way players interact with the entities and events represented on it.
Non Playable Character (NPC)	Entity controlled only by the Dungeon Master. NPCs can interact with players and with the world around them in a dynamic way.
Monster	Entity characterized by its hostility towards players. Whenever possible monsters will try to start a battle against players in order to kill them.
Chat	Viewable section of the players' interface designated to contain the list of messages sent by players during the Adventure.
Whisper	Message readable only by players with whom

	the sender wishes to communicate.
Global Message	Chat message viewable by all Players.
Dice	Elements viewable on players' interface which symbolically represent the random luck factor that manifests itself in different ways throughout the Adventure.
Inventory	Section used by players to keep track of money and other objects which are gathered throughout the adventure.
Bestiary	Section where it is possible to read all Monster's Stats and features.
NPC List	List of all NPCs present in the Gameworld.
Tile	Measurement Unit which defines a limited plot of land in the Map. It's useful to determine with precision specific areas and their coordinates.
HP	Abbreviation for Health Points. They are numbers associated with all entities and whose depletion causes the entity to die.
XP	Abbreviation for "Experience Points". They are numbers associated with all Characters. They can be gained in various ways and they determine the leveling up and the upgrade of the Characters.
Launch	Launch of the listed dice, which gives in output the sum of the dice results.
Launch List	List of dice that are thrown during Launch.
Stats	It's the set of parameters that define each entities' characteristics and behaviours.
Message Box	Section of players' interface which displays all viewable sent messages and allows players to write their own ones.
Gameworld	World where the D&D Adventure takes place. It is filled with entities and its representation is displayed by the map.
Role Play Game	It is the genre of games D&D belongs to. It is characterized by a plot players take part in as characters and where they make their own choices.
Players' Interface	Portion of screen displayed to players and with which they can interact.

Section	Specific Area or Tab of players' interface with certain information about the Game.
NPC List	List that contains all the NPCs present in the adventure.
Character List	List of Characters present in an Adventure.
Warning Alert	Special Alert the User sees when a certain Process fails to be completed due to an error.
Item	Findable object in the gameworld, which can be bought or picked up by the Characters during their adventure.
Item's List	List of Items present in an adventure, available to the DM for consultation.
Entity	Monster or NPC present in the gameworld and with whom the Characters can interact in various ways.
State	Specific condition the NPCs or Monsters find themselves in and which influences fights and interactions.
Combat	Action that takes place when an entity inflicts damage to another. When an entity is in combat is decided by the Dungeon master.
GUI	Graphical User Interface. It is the user interface which allows the interaction between user and Software.
Login	Process after which the User is able to join the D&D experience. It is characterized by the insertion of the User's credentials.
Registration	It's the process by which the User's credentials get registered in the Database.
Database	Virtual space where the Data the Software requires is stored.
Logout	Process after which the User's previously selected credentials are temporarily not used anymore by the system for matching the current User to all the Data the user has stored in his profile.
Tab	Section of the Playable D&D interface which the user can select when he needs to know specific information about something.
Alignment	It is a categorization of the ethical and moral

	perspective of player characters.
Armor Class	It is the measure of how hard it is for an attacker to hit you when rolling to attack.
Propic	Abbreviation for Profile Picture, which is associated with every Character, Monster, NPC and Item.
Casting Time	It's a value that defines the time it takes to cast a spell. The more it takes the more turns have to be spent casting that spell.
Damage Dice	Type of dice which determine the amount of damage that will be inflicted by a spell.
Resistances	Defines which sources of damage deal less damage than normal when using the specific item.

FUNCTIONAL REQUIREMENTS:

- **Monster Manager**
 - **FR1** - Create Monster
 - *Type: Requirement*
 - *Status: Proposed. Version 1.0. Phase 1.0.*
 - *Package: Monster Manager*
 - The system must allow the DM to create a new Monster
 - **FR2** - Edit Monster
 - *Type: Requirement*
 - *Status: Proposed. Version 1.0. Phase 1.0.*
 - *Package: Monster Manager*
 - The system must allow the DM to edit all the stats, attacks, skills and name of a pre-existing Monster selected from the Bestiary
 - **FR3** - Save Monster
 - *Type: Requirement*
 - *Status: Proposed. Version 1.0. Phase 1.0.*
 - *Package: Monster Manager*
 - The system must allow the DM to save a valid newly created or the newly edited Monster inside the Bestiary
 - **FR4** - Remove Monster
 - *Type: Requirement*
 - *Status: Proposed. Version 1.0. Phase 1.0.*
 - *Package: Monster Manager*
 - The system must allow the DM to delete the selected Monster from the Bestiary
 - **FR5** - Search Monster
 - *Type: Requirement*

- *Status: Proposed. Version 1.0. Phase 1.0.*
 - *Package: Monster Manager*
 - The system must allow the DM to search in the Bestiary for a specific Monster
- **FR6** - View Monsters
 - *Type: Requirement*
 - *Status: Proposed. Version 1.0. Phase 1.0.*
 - *Package: Monster Manager*
 - The system must allow the DM to visualize all the Monsters inside the Bestiary
- **NPC Manager**
 - **FR7** - Create NPC
 - *Type: Requirement*
 - *Status: Proposed. Version 1.0. Phase 1.0.*
 - *Package: NPC Manager*
 - The system must allow the DM to create a new NPC
 - **FR8** - Save NPC
 - *Type: Requirement*
 - *Status: Proposed. Version 1.0. Phase 1.0.*
 - *Package: NPC Manager*
 - The system must allow the DM to save a valid newly created or edited NPC to the NPC list
 - **FR9** - Edit NPC
 - *Type: Requirement*
 - *Status: Proposed. Version 1.0. Phase 1.0.*
 - *Package: NPC Manager*
 - The system must allow the DM to edit a pre-existing NPC selected from the NPC list
 - **FR10** - Search NPC
 - *Type: Requirement*
 - *Status: Proposed. Version 1.0. Phase 1.0.*
 - *Package: NPC Manager*
 - The system must allow the DM to search the NPC List for a specific NPC
 - **FR11** - Remove NPC
 - *Type: Requirement*
 - *Status: Proposed. Version 1.0. Phase 1.0.*
 - *Package: NPC Manager*
 - The system must allow the DM to remove a selected NPC from the NPC List
 - **FR12** - View NPC
 - *Type: Requirement*
 - *Status: Proposed. Version 1.0. Phase 1.0.*
 - *Package: NPC Manager*
 - The system must allow the DM to visualize all the NPCs inside the NPC List
- **Chat Manager**
 - **FR13** - Send Message

- *Type: Requirement*
 - *Status: Proposed. Version 1.0. Phase 1.0.*
 - *Package: Chat Manager*
 - The system must send all the players partaking in the current adventure the message typed in the Message Box once the User clicks "Send".
- **FR14** - Send Whisper
 - *Type: Requirement*
 - *Status: Proposed. Version 1.0. Phase 1.0.*
 - *Package: Chat Manager*
 - The system must send the message typed in the Message Box once the User clicks "Send" only to the sender User and the receiving User
- **FR15** - Show Dice Results
 - *Type: Requirement*
 - *Status: Proposed. Version 1.0. Phase 1.0.*
 - *Package: Chat Manager*
 - The system must send a message in Chat whenever a Player throws it's dice list containing the sum of the dice results
- **Adventure Manager**
 - **FR16** - Add Adventure
 - *Type: Requirement*
 - *Status: Proposed. Version 1.0. Phase 1.0.*
 - *Package: Adventure Manager*
 - The system must allow a logged in User to create a new Adventure, the creator User will become the DM of that Adventure
 - **FR17** - Generate Adventure Code
 - *Type: Requirement*
 - *Status: Proposed. Version 1.0. Phase 1.0.*
 - *Package: Adventure Manager*
 - The system link an ID to the newly created Adventure
 - **FR18** - Remove Adventure
 - *Type: Requirement*
 - *Status: Proposed. Version 1.0. Phase 1.0.*
 - *Package: Adventure Manager*
 - The system must allow the DM of an Adventure to delete that Adventure
 - **FR19** - Search Adventure
 - *Type: Requirement*
 - *Status: Proposed. Version 1.0. Phase 1.0.*
 - *Package: Adventure Manager*
 - The system must allow searching in the Database for a given Adventure
 - **FR20** - Load Adventure
 - *Type: Requirement*
 - *Status: Proposed. Version 1.0. Phase 1.0.*
 - *Package: Adventure Manager*
 - The system must allow for the loading of an Adventure in the Database.

- **FR21** - Save Adventure
 - *Type: Requirement*
 - *Status: Proposed. Version 1.0. Phase 1.0.*
 - *Package: Adventure Manager*
 - The system must allow for the saving of an Adventure in the Database.
- **Item Manager**
 - **FR22** - Add Item
 - *Type: Requirement*
 - *Status: Proposed. Version 1.0. Phase 1.0.*
 - *Package: Item Manager*
 - The system must allow the DM to add a new Item to the Items List
 - **FR23** - Remove Item
 - *Type: Requirement*
 - *Status: Proposed. Version 1.0. Phase 1.0.*
 - *Package: Item Manager*
 - The system must allow the DM to remove a selected pre-existing Item from the Items List
 - **FR24** - Search Item
 - *Type: Requirement*
 - *Status: Proposed. Version 1.0. Phase 1.0.*
 - *Package: Item Manager*
 - The system searches through the Items List for a specific Item
 - **FR25** - Save Item
 - *Type: Requirement*
 - *Status: Proposed. Version 1.0. Phase 1.0.*
 - *Package: Item Manager*
 - The system must allow the DM to save a valid item to the Items List
 - **FR26** - Edit Item
 - *Type: Requirement*
 - *Status: Proposed. Version 1.0. Phase 1.0.*
 - *Package: Item Manager*
 - The system must allow the DM to edit a selected Item
 - **FR27** - View Items
 - *Type: Requirement*
 - *Status: Proposed. Version 1.0. Phase 1.0.*
 - *Package: Item Manager*
 - The system must allow the DM to visualize all the Items inside the Items List
- **Map Manager**
 - **FR28** - Add Map
 - *Type: Requirement*
 - *Status: Proposed. Version 1.0. Phase 1.0.*
 - *Package: Map Manager*
 - The system must allow the DM to load a new map in the Maps List
 - **FR29** - Remove Map
 - *Type: Requirement*
 - *Status: Proposed. Version 1.0. Phase 1.0.*

- *Package: Map Manager*
 - The system must allow the DM to remove a Map from the Maps List
- **FR30** - Search Map
 - *Type: Requirement*
 - *Status: Proposed. Version 1.0. Phase 1.0.*
 - *Package: Map Manager*
 - The system must allow the DM to search for a specific Map inside the Maps List
- **FR31** - Save Map
 - *Type: Requirement*
 - *Status: Proposed. Version 1.0. Phase 1.0.*
 - *Package: Map Manager*
 - The system must allow the DM to save a newly generated Map to the Maps List
- **FR32** - Load Map
 - *Type: Requirement*
 - *Status: Proposed. Version 1.0. Phase 1.0.*
 - *Package: Map Manager*
 - The system must allow the DM to choose a pre-existing Map in the Maps List
- **FR33** - Add Entity to Map
 - *Type: Requirement*
 - *Status: Proposed. Version 1.0. Phase 1.0.*
 - *Package: Map Manager*
 - The system must allow the DM to add an Entity to the Map.
- **FR34** - Attack Entity
 - *Type: Requirement*
 - *Status: Proposed. Version 1.0. Phase 1.0.*
 - *Package: Map Manager*
 - The system must allow the Users (both DM and non-DM) to decide the target of their attack
- **FR35** - Remove Entity from Map
 - *Type: Requirement*
 - *Status: Proposed. Version 1.0. Phase 1.0.*
 - *Package: Map Manager*
 - The system must allow the DM to remove an Entity from the Map
- **FR36** - Set Entity State
 - *Type: Requirement*
 - *Status: Proposed. Version 1.0. Phase 1.0.*
 - *Package: Map Manager*
 - The system must allow the DM to set the State of an entity
- **FR37** - Search Entity
 - *Type: Requirement*
 - *Status: Proposed. Version 1.0. Phase 1.0.*
 - *Package: Map Manager*
 - The system must allow the System to search for an entity in the map.
- **FR38** - Change Tile
 - *Type: Requirement*

- *Status: Proposed. Version 1.0. Phase 1.0.*
 - *Package: Map Manager*
 - The system must allow the DM to change the image of a specific Tile
 - **FR39** - View Maps
 - *Type: Requirement*
 - *Status: Proposed. Version 1.0. Phase 1.0.*
 - *Package: Map Manager*
 - The system must allow the DM to visualize all the Maps inside the Maps List
- **Character Manager**
 - **FR40** - Edit Characters
 - *Type: Requirement*
 - *Status: Proposed. Version 1.0. Phase 1.0.*
 - *Package: Character Manager*
 - The system must allow Players to edit their own Character Sheet
 - **FR41** - View Characters
 - *Type: Requirement*
 - *Status: Proposed. Version 1.0. Phase 1.0.*
 - *Package: Character Manager*
 - The system must allow the Players to visualize their own Character and other player's Characters.
 - **FR42** - Save Character
 - *Type: Requirement*
 - *Status: Proposed. Version 1.0. Phase 1.0.*
 - *Package: Character Manager*
 - The system must allow the Players to save their own Character.
 - **FR43** - Add Character
 - *Type: Requirement*
 - *Status: Proposed. Version 1.0. Phase 1.0.*
 - *Package: Character Manager*
 - The system must allow a logged in User to subscribe to a pre-existing Adventure though the Adventures' name
 - **FR44** - Remove Character
 - *Type: Requirement*
 - *Status: Proposed. Version 1.0. Phase 1.0.*
 - *Package: Character Manager*
 - The system must allow a logged in User to remove it's own Character from an Adventure in which it is participating.
 - **FR45** - Search Character
 - *Type: Requirement*
 - *Status: Proposed. Version 1.0. Phase 1.0.*
 - *Package: Character Manager*
 - The system must allow searching in the Database for a given Character
- **Dice Manager**
 - **FR46** - Throw Dice
 - *Type: Requirement*
 - *Status: Proposed. Version 1.0. Phase 1.0.*

- *Package: Dice Manager*
 - The system must allow the Players to throw their own Dice List and show the result as a Message in Chat
- **FR47** - Add Dice to List
 - *Type: Requirement*
 - *Status: Proposed. Version 1.0. Phase 1.0.*
 - *Package: Dice Manager*
 - The system must allow Players to add a Dice to their own Dice List
- **Database Manager**
 - **FR48** - Save To Database
 - *Type: Requirement*
 - *Status: Proposed. Version 1.0. Phase 1.0.*
 - *Package: Database Manager*
 - The system must allow the Users to save modifications, additions and removals made to the Database
 - **FR49** - Load from Database
 - *Type: Requirement*
 - *Status: Proposed. Version 1.0. Phase 1.0.*
 - *Package: Database Manager*
 - The system must allow Users to read from the Database the Bestiary, NPCs List, Items List and Information about Users
 - **FR50** - Authenticate
 - *Type: Requirement*
 - *Status: Proposed. Version 1.0. Phase 1.0.*
 - *Package: Database Manager*
 - The system must be able to authenticate with the Database incoming Users

NON FUNCTIONAL REQUIREMENTS

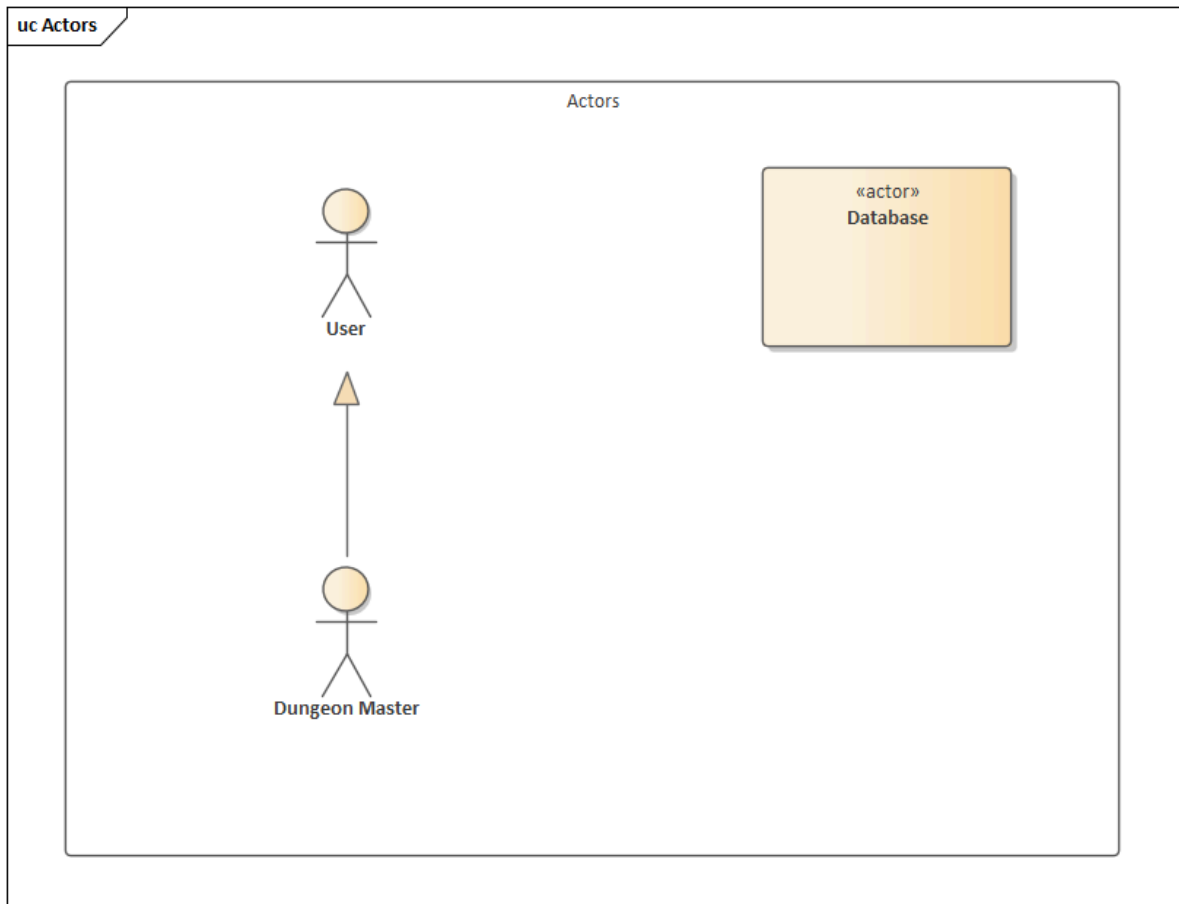
- **NFR1** - Implementation
 - *Type: Requirement*
 - *Status: Proposed. Version 1.0. Phase 1.0.*
 - *Package: Non Functional Requirements*
 - The system must be implemented using Python 3
- **NFR2** - User Interface
 - *Type: Requirement*
 - *Status: Proposed. Version 1.0. Phase 1.0.*
 - *Package: Non Functional Requirements*
 - The system must interact with the User with a GUI
- **NFR3** - DataBase
 - *Type: Requirement*
 - *Status: Proposed. Version 1.0. Phase 1.0.*
 - *Package: Non Functional Requirements*
 - The system must be connected to a DataBase at all times
- **NFR4** - Signin And Login
 - *Type: Requirement*

- *Status: Proposed. Version 1.0. Phase 1.0.*
- *Package: Non Functional Requirements*
- The User must be authenticated before accessing the system
- **NFR5** - Data Sync
 - *Type: Requirement*
 - *Status: Proposed. Version 1.0. Phase 1.0.*
 - *Package: Non Functional Requirements*
 - The system must provide Data to the Users synchronously, all the Users must have the same Data at all times

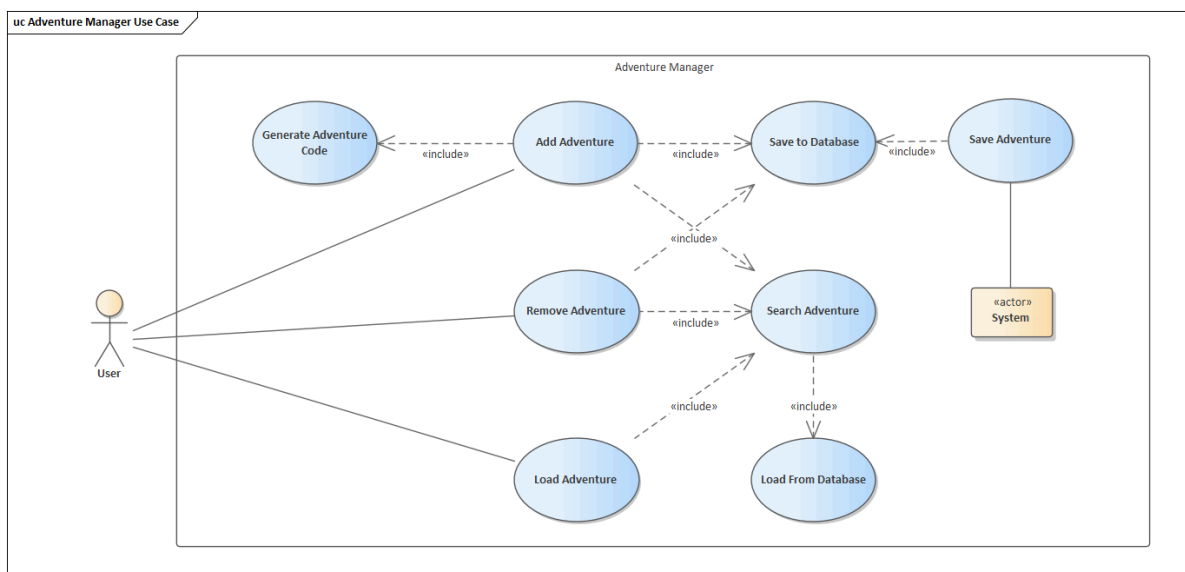


USE CASE DIAGRAMS

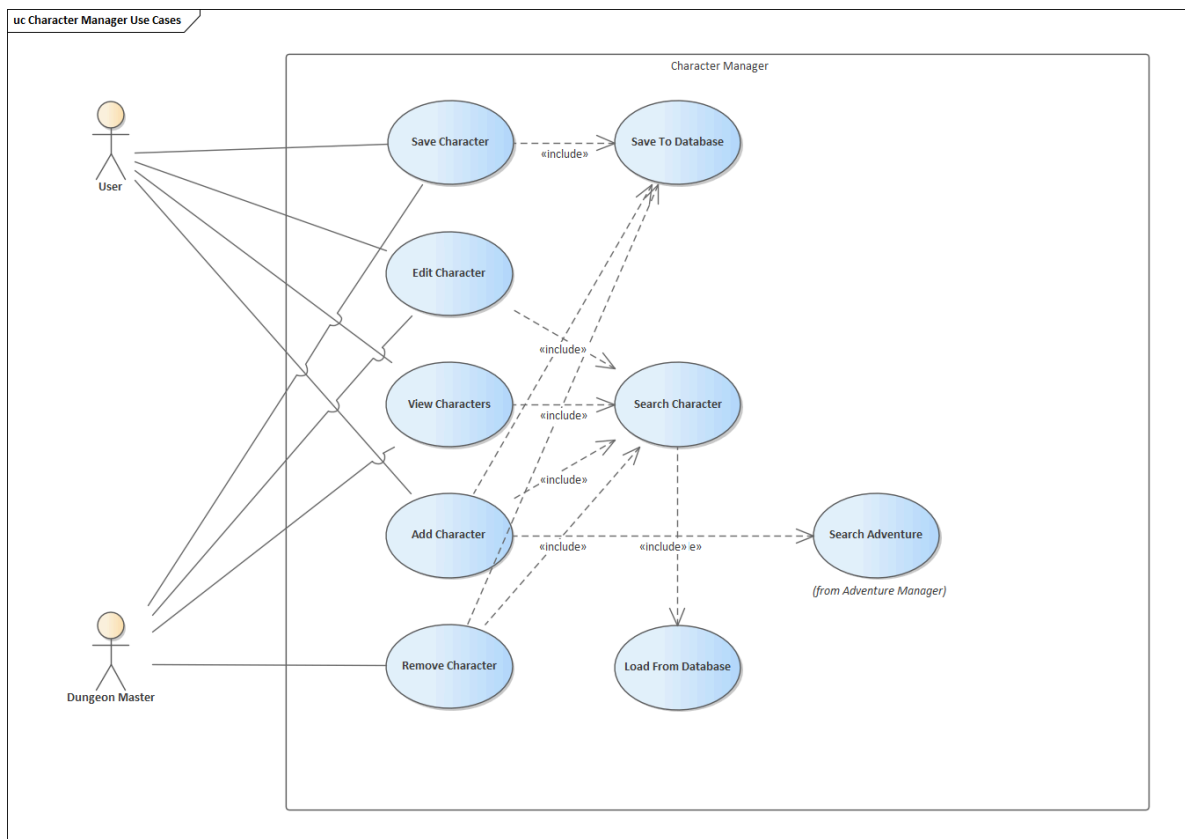
ACTORS:



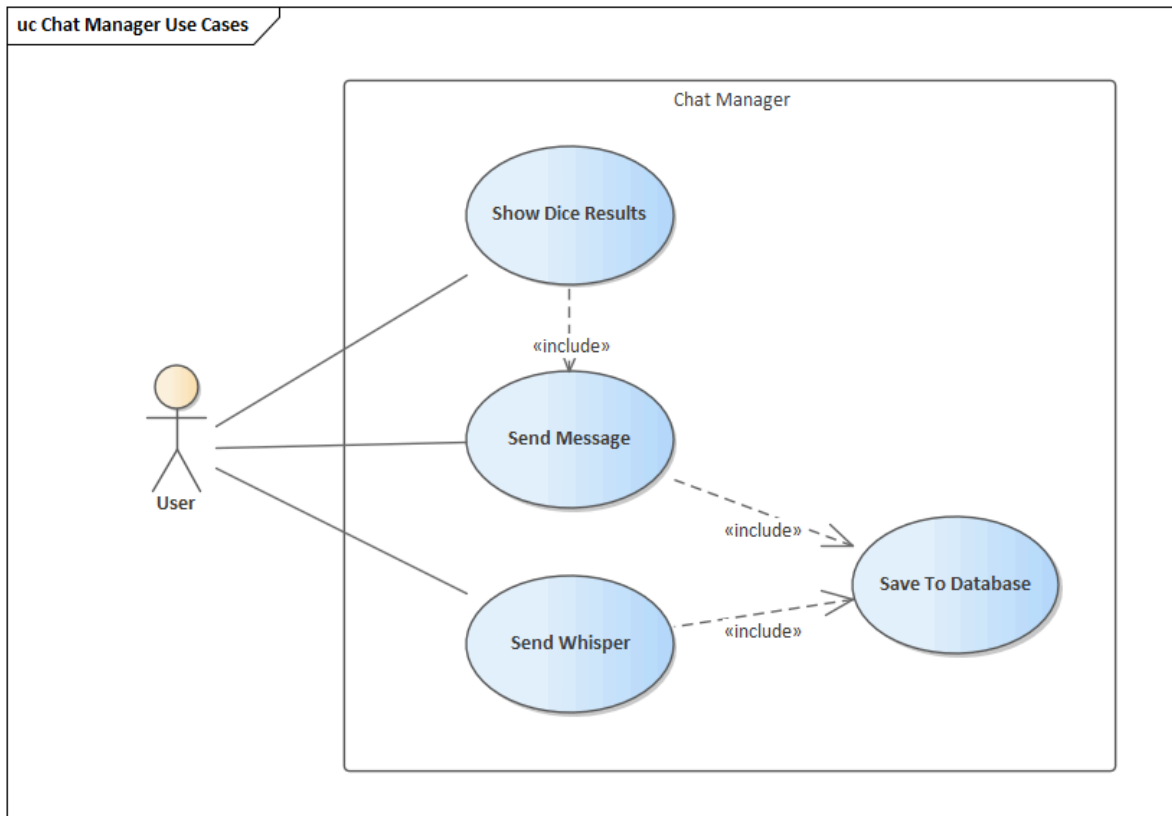
ADVENTURE MANAGER:



CHARACTER MANAGER:

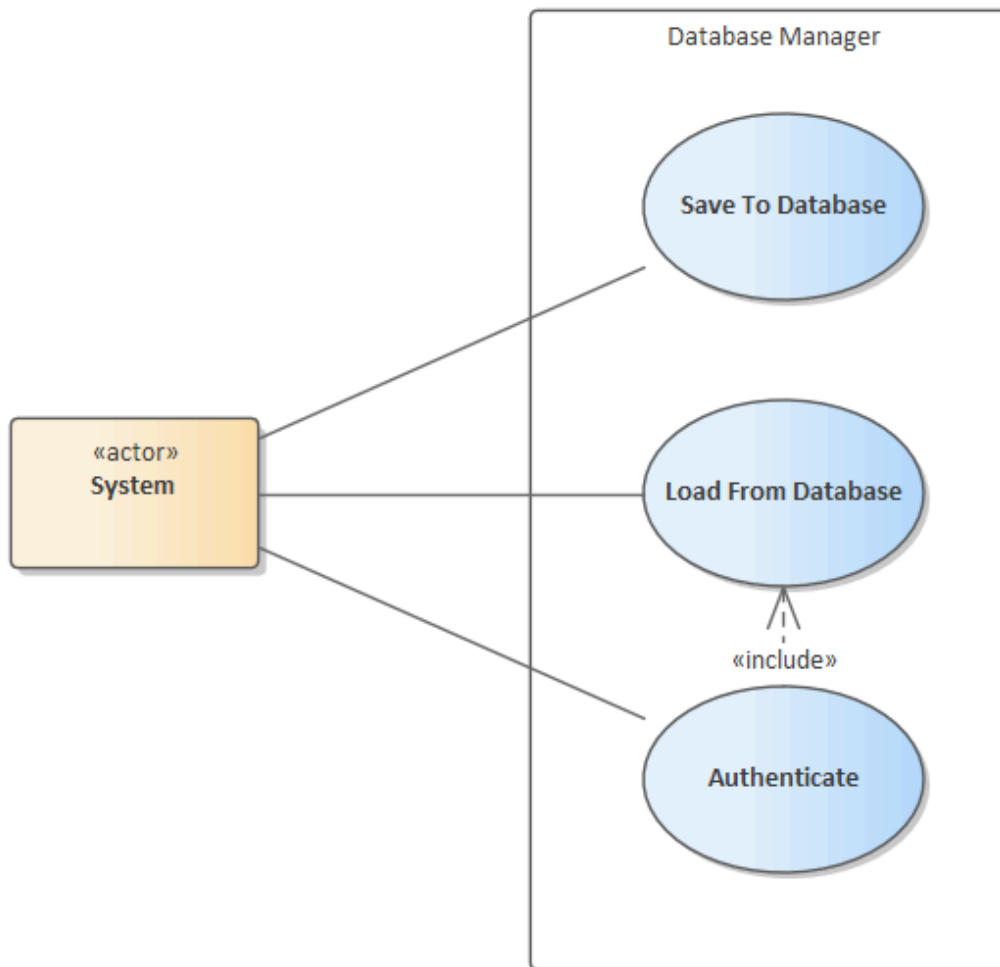


CHAT MANAGER:

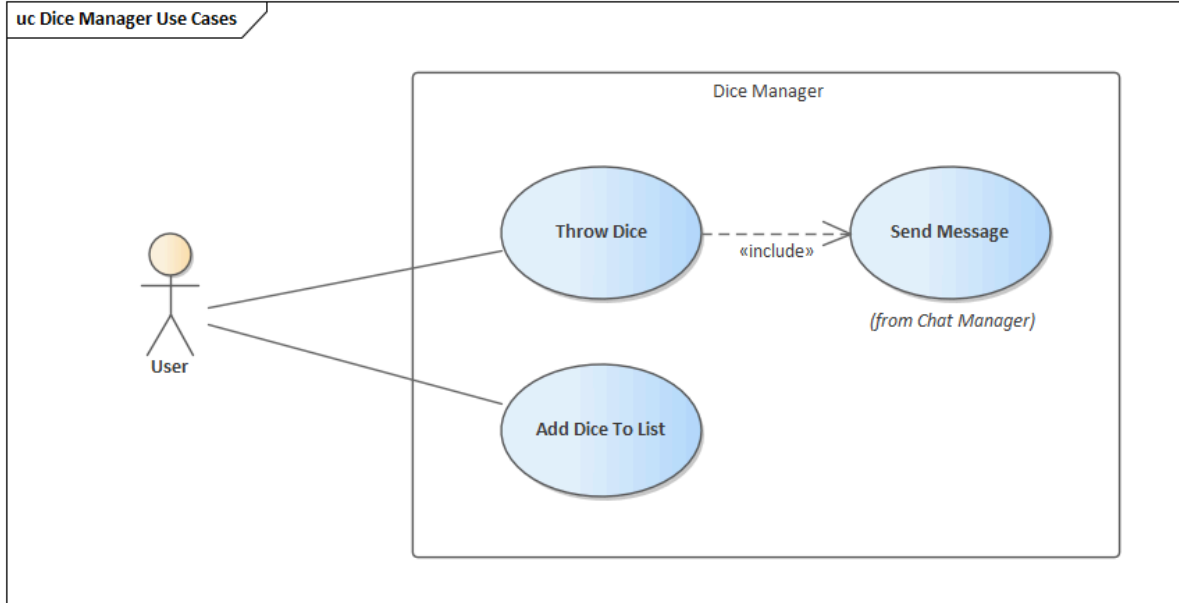


DATABASE MANAGER:

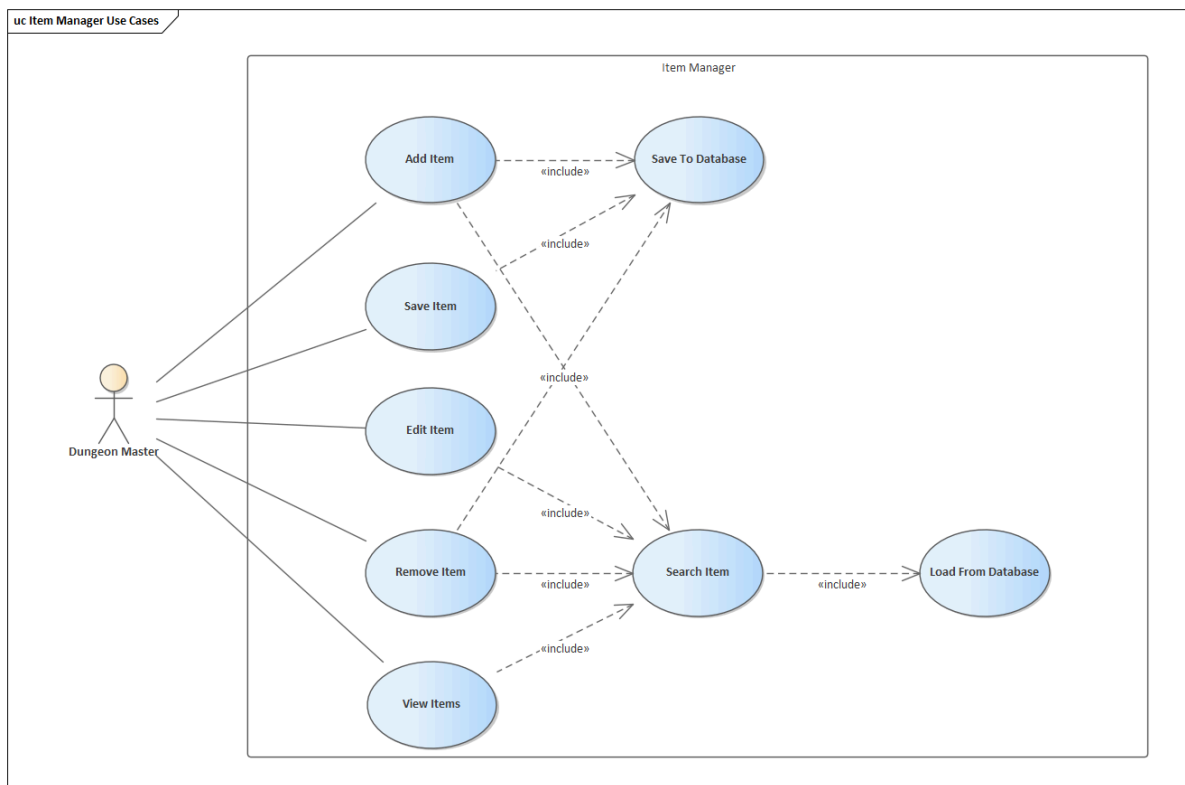
uc Database Manager Use Cases



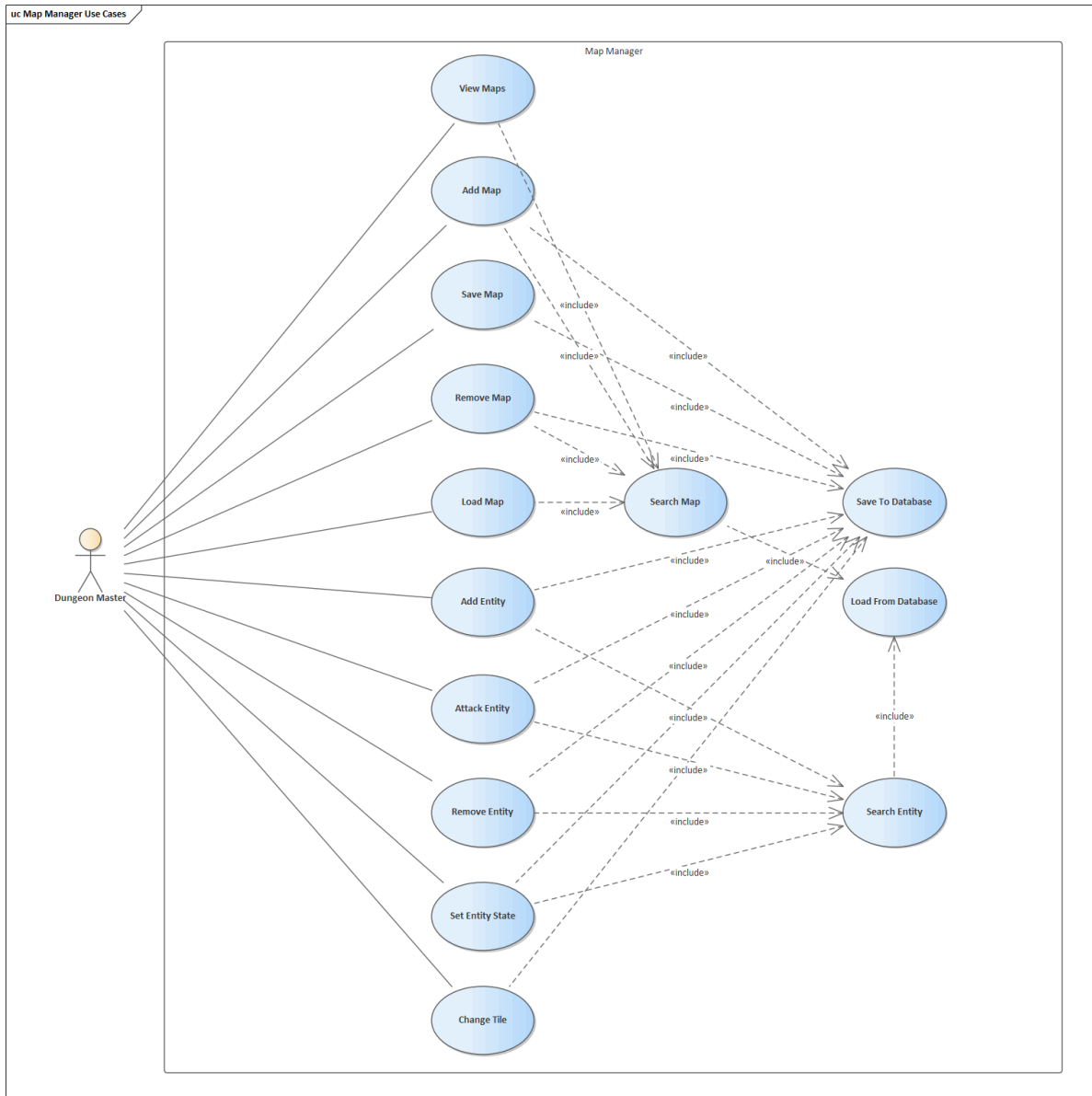
DICE MANAGER:



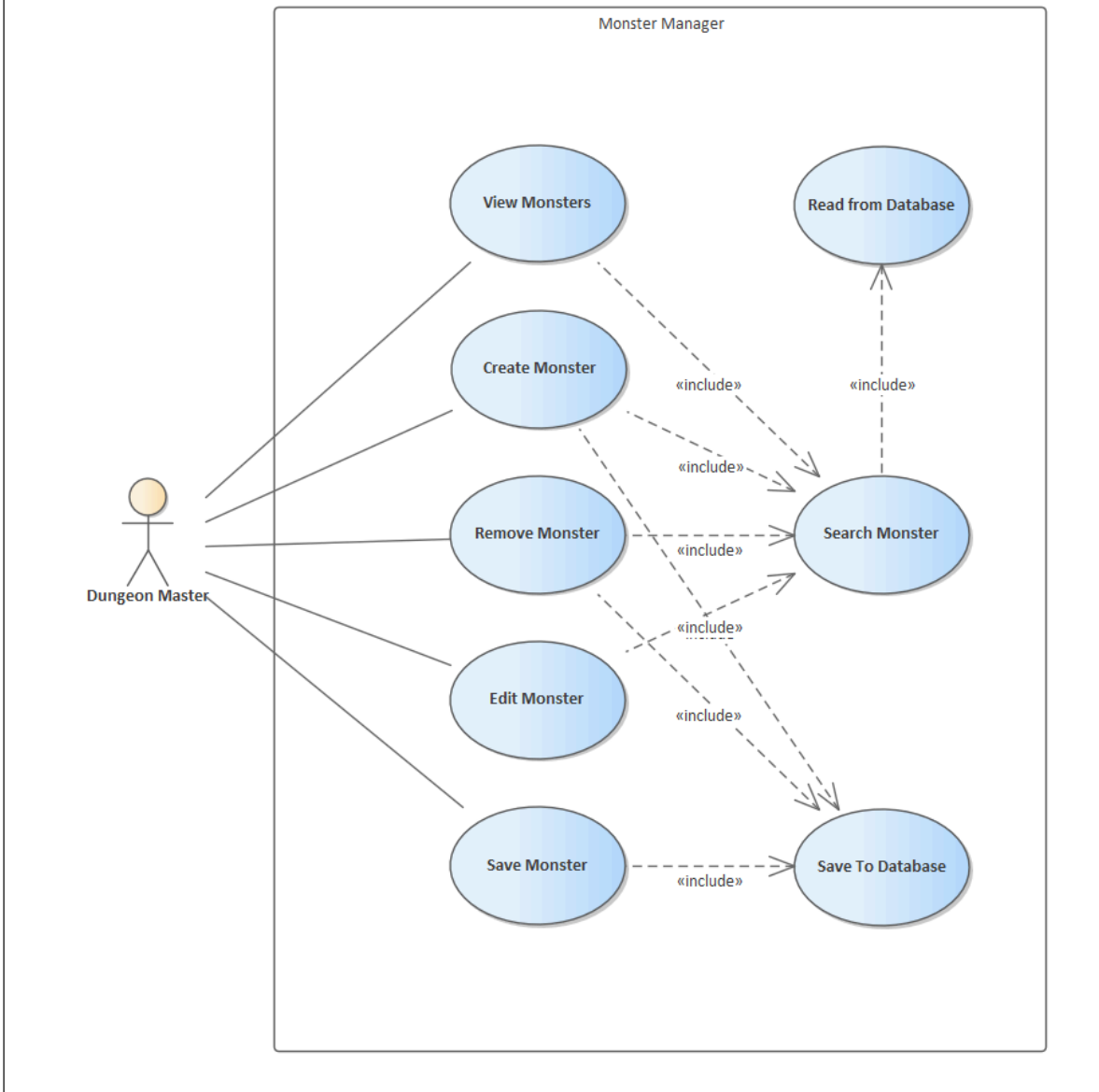
ITEM MANAGER:



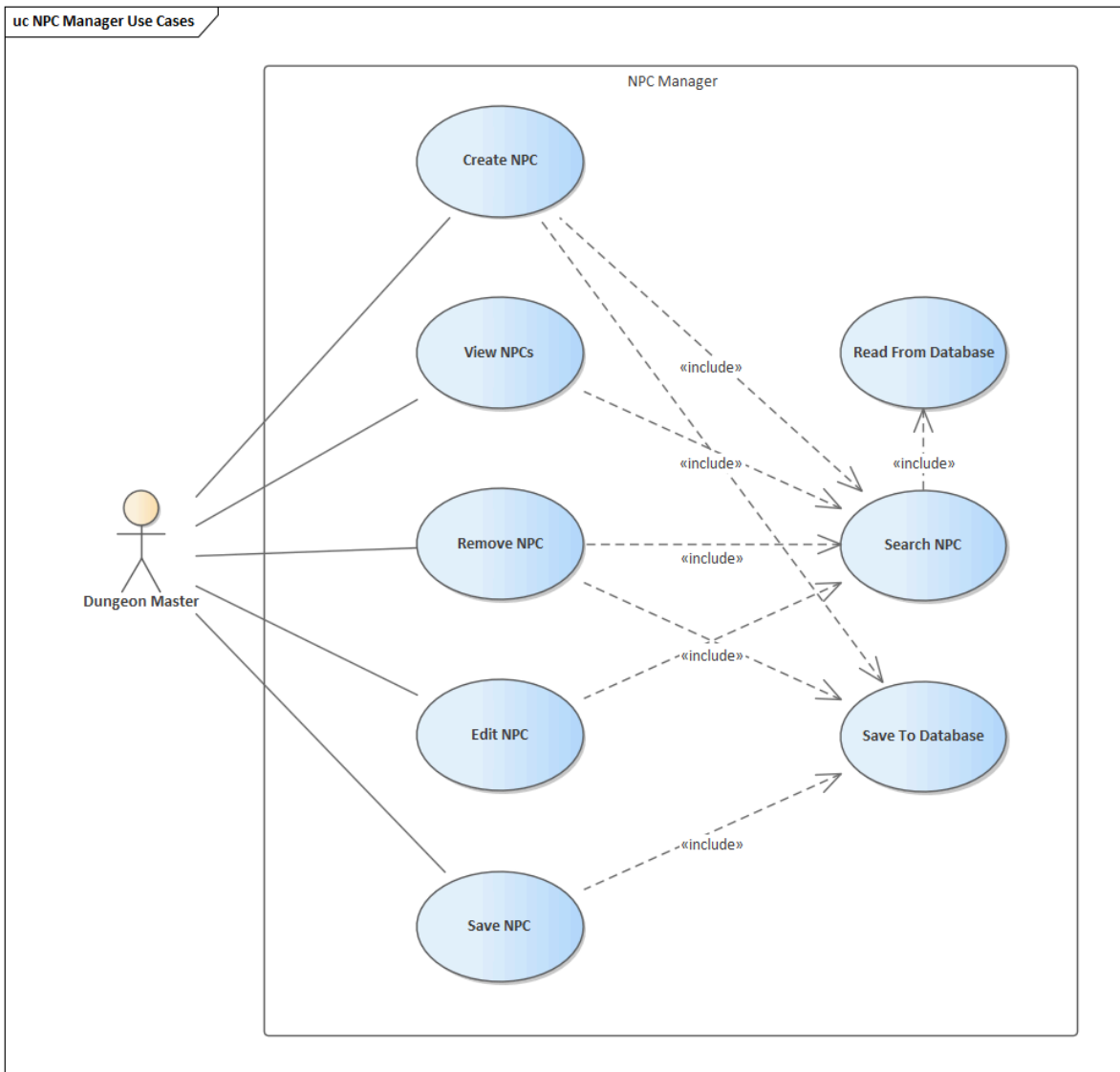
MAP MANAGER:



MONSTER MANAGER:



NPC MANAGER:



USE CASES DESCRIPTION:

USE CASE: CREATE MONSTER (FR1)

CreateMonster

The Use case "CreateMonster" allows for the generation in the Gameworld of an entity type "Monster" which has all the features and the stats the Dungeon Master decides for it.

Main Actors: DM.

Secondary Actors: None.

Preconditions:

1. The user must have selected an Adventure
2. The DM must be logged in
3. The DM must own the Adventure

Main Sequence of Events:

1. The Use Case begins when the Dungeon Master decides to create a new Monster
2. include (SearchMonster)
3. IF the monster exists
 - 3.1. The System shows a warning alert
4. include (SaveToDatabase)

Postconditions: None

Alternative Sequence of Events: None.

USE CASE: EDIT MONSTER (FR2)**EditMonster**

The use case “EditMonster” allows the Dungeon Master to edit Name, Size, Alignment, Armor Class, HP, Speed, Strength, Dexterity, Constitution, Intelligence, Wisdom, Charisma, Abilities, Resistances/Weaknesses, Extras and Actions/Attacks of a selected Monster

Main Actors: DM.

Secondary Actors: Database.

Preconditions:

1. The Bestiary must not be empty.
2. The DM must be logged in
3. The DM must own the Adventure

Main Sequence of Events:

1. The use case begins when the DM decides to edit a pre-existing Monster from the Bestiary.
2. include (SearchMonster)
3. The DM edits all the parameters it wants

Postconditions: None

Alternative Sequence of Events: None.

USE CASE: SAVE MONSTER (FR3)**SaveMonster**

The use case ‘SaveMonster’ allows the Dungeon Master to save a newly edited monster.

Main Actors: DM.

Secondary Actors: Database.

Preconditions:

1. The DM must be logged in
2. The DM must own the Adventure

Main Sequence of Events:

1. The use case begins when the DM saves the Monster
2. IF the Monster is NOT Valid:
 - 2.1. The system shows a Warning alert
3. ELSE include (SaveToDatabase)

Postconditions:

1. The Monster must have been saved to the Database.

Alternative Sequence of Events: None.

USE CASE: REMOVE MONSTER (FR4)**RemoveMonster**

The use case "RemoveMonster" allows the DM to remove an entity "Monster" from the Bestiary.

Main Actors: DM.

Secondary Actors: Database.

Preconditions:

1. The Bestiary must not be empty.
2. The DM must be logged in
3. The DM must own the Adventure

Main Sequence of Events:

1. The use case begins when the DM decides to remove a selected Monster from Bestiary.
2. include(SearchMonster).
3. IF the Monster exists in the Bestiary.
 - 3.1. Monster is removed from the Bestiary.
4. IF the monster does not exist in the Bestiary
 - 4.1. System shows a warning alert.
5. include (SaveToDatabase).

Postconditions:

1. The selected Monster must have been eliminated from Bestiary.

Alternative Sequence of Events: none.

USE CASE: SEARCH MONSTER (FR5)**SearchMonster**

The use case "SearchMonster" allows the System find a specific Monster in the Database.

Main Actors: DM.

Secondary Actors: Database.

Preconditions:

1. The Monster must exist in the Database

Main Sequence of Events:

1. The use case begins when the System is requested to search for a Monster using the Name of the monster.
2. Include (LoadFromDatabase)
3. IF the Monster does NOT exist
 - 3.1 The System returns an error message.
4. ELSE the System returns the Monster.

Postconditions: None.

Alternative Sequence of Events: None.

USE CASE: VIEW MONSTERS (FR6)**ViewMonsters**

The use case "View Monsters" allows the Dungeon Master to visualize the entire Bestiary inside a scroll view.

Main Actors: DM.

Secondary Actors: Database.

Preconditions: None

Main Sequence of Events:

1. The Use Case begins when the Dungeon Master enters the Bestiary tab.
2. Include (SearchMonster)
3. The system populates the scroll view with the results from the Database

Postconditions:

1. The scroll view must be populated.

Alternative Sequence of Events: none.

USE CASE: CREATE NPC (FR7)**CreateNPC**

The use case "CreateNpc" allows for the creation of an entity type "NPC" which has all the features and the stats the Dungeon Master decides for it.

Main Actors: DM.

Secondary Actors: none.

Preconditions:

1. The user must have selected an Adventure
2. The DM must be logged in
3. The DM must own the Adventure

Main Sequence of Events:

1. The Use Case begins when the Dungeon Master decides to create a new NPC.
2. include (SearchNPC)
3. IF the NPC exists
 - a. The System shows a warning alert
4. include (SaveToDatabase)

Postconditions: None

Alternative Sequence of Events: None.

USE CASE: SAVE NPC (FR8)**SaveNPC**

The Use case "SaveNpc" allows the DM to save a valid newly created or edited NPC to the NPC list.

Main Actors: DM.

Secondary Actors: Database.

Preconditions:

1. The DM must be logged in
2. The DM must own the Adventure

Main Sequence of Events:

1. The use case begins when the DM saves the NPC
2. IF the NPC is NOT Valid:
 - 2.1. The system shows a Warning alert
3. ELSE include (SaveToDatabase)

Postconditions:

1. The NPC must be saved in the Database.

Alternative Sequence of Events: None.

USE CASE: EDIT NPC (FR9)**EditNPC**

The Use case “EditNPC” allows the DM to edit an NPC which was previously created and saved in the NPC list.

Main Actors: DM.

Secondary Actors: Database.

Preconditions:

1. The user must have a saved NPC to edit.
2. The DM must be logged in
3. The DM must own the Adventure

Main Sequence of Events:

1. The Use Case begins when the Dungeon Master decides to edit a pre-existing NPC.
2. include (SearchNPC)
3. The DM edits all the parameters it wants

Postconditions: None

Alternative Sequence of Events: None.

USE CASE: SEARCH NPC (FR10)

SearchNPC

The Use case “EditNPC” allows the System to search for a specific NPC using the Name.

Main Actors: DM.

Secondary Actors: Database.

Preconditions:

1. The NPC must exist in the Database

Main Sequence of Events:

1. The Use Case begins when the System needs to search for a pre-existing NPC.
2. include (ReadFromDatabase)
3. IF the System finds the NPC in the Database
 - 3.1. The NPC is returned
4. ELSE the System shows an Error message

Postconditions: None

Alternative Sequence of Events: None.

USE CASE: REMOVE NPC (FR11)

RemoveNPC

The use case “RemoveNPC” allows the DM to remove an entity “NPC” from the NPC List.

Main Actors: DM.

Secondary Actors: Database.

Preconditions:

1. The NPC List must not be empty.
2. The DM must be logged in
3. The DM must own the Adventure

Main Sequence of Events:

1. The use case begins when the DM decides to remove an NPC from the NPC List.
2. include(SearchNPC)
3. IF the NPC exists in the NPC List
 - 3.1. NPC is removed from the NPC List and from the Adventure.
4. IF the NPC does not exist in the NPC List
 - 4.1. System shows a warning Alert
5. include (SaveToDatabase)

Postconditions:

1. The selected NPC must have been eliminated from the NPC List.

Alternative Sequence of Events: None.

USE CASE: VIEW NPC (FR12)

ViewNPC

The Use case “ViewNpc” allows the DM to visualize all the NPCs inside the NPC List in a scroll view.

Main Actors: DM.

Secondary Actors: None.

Preconditions: None

Main Sequence of Events:

1. The Use Case begins when the Dungeon Master enters the NPCs tab.
2. include (SearchNPC)
3. The System shows all the NPCs in the NPC list.

Postconditions:

1. The scroll view must be populated

Alternative Sequence of Events: None.

USE CASE: SEND MESSAGE (FR13)

SendMessage

The use case “SendMessage” allows the player to send a message in the Chat.

Main Actors: User.

Secondary Actors: Database.

Preconditions:

1. An Adventure must be loaded.

Main Sequence of Events:

1. The use case begins when a Player decides to write a Message in the Chat.
2. The Player writes the Message in the Message Box and clicks “Send”.
3. IF the sender sends a message in Global Chat
 - 3.1. All the Users in the same Adventure will receive the message
4. include (SaveToDatabase)

Postconditions:

1. The sent messages must be displayed in the Chat.

Alternative Sequence of Events: None.

USE CASE: SEND WHISPER (FR14)

SendWhisper

The use case “SendWhisper” allows the Player to send a Private Message to another Player, which can be visualized in the Chat.

Main Actors: User.

Secondary Actors: Database.

Preconditions:

1. An Adventure must be loaded.

Main Sequence of Events:

1. Use Case begins when a Player decides to write a Private Message to another player without letting other players read it in the Global Chat.
2. Player writes the message in the Message Box and specifies the recipient to whom he intends to send the message using “/w receiverName”.

3. Player clicks on "Send Message".
4. The recipient and sender visualizes the message in the Chat displayed.
5. include (SaveToDatabase)

Postconditions:

1. The sent messages must be displayed in the Chat.

Alternative Sequence of Events: None.

USE CASE: SHOW DICE RESULTS (FR15)

ShowDiceResults

The use case "ShowDiceResults" allows the Player to visualize the results of his Dice Launch, which are displayed as a Message in Chat, containing the sum of the dice results

Main Actors: User.

Secondary Actors: None.

Preconditions: None

Main Sequence of Events:

1. The Use Case begins when the User clicks on "Throw".
2. The System sends a message, showing the Sum of all thrown Dice.
3. include (SendMessage)

Postconditions: None.

Alternative Sequence of Events: None.

USE CASE: ADD ADVENTURE (FR16)

Add Adventure

The Use case "AddAdventure" allows the User to create a new Adventure. The creator becomes the Dungeon Master of the Adventure.

Main Actors: User.

Secondary Actors: None.

Preconditions:

1. The User must be logged in.

Main Sequence of Events:

1. The use case begins when a registered User creates a new Adventure
2. The System asks the User the Name of the Adventure
3. include (SearchAdventure)

4. IF the name already exists
 - 4.1. The system shows the User an error message
5. ELSE the System promotes the User to Dungeon Master of the new Adventure
6. include (GenerateAdventureCode)
7. include (SaveToDatabase)

Postconditions:

1. The Adventure must have been created.

Alternative Sequence of Events: None.

USE CASE: GENERATE ADVENTURE CODE (FR17)

Generate Adventure Code

The Use case "GenerateAdventureCode" allows the System to create an invitation code for an Adventure.

Main Actors: System.

Secondary Actors: None.

Preconditions:

1. The adventure must exist in the Database

Main Sequence of Events:

1. The use case begins when a new Adventure is created
2. The system generates a unique ID for the Adventure

Postconditions:

1. The generated code must be unique

Alternative Sequence of Events: None.

USE CASE: REMOVE ADVENTURE(FR18)

Remove Adventure

The Use case "RemoveAdventure" allows the Dungeon Master to delete an existing Adventure from the Database, unsubscribing all Users subscribed to that Adventure.

Main Actors: DM.

Secondary Actors: Database.

Preconditions:

1. The adventure must exist in the Database
2. The DM must be logged in
3. The DM must own the Adventure

Main Sequence of Events:

1. The use case begins when a DM decides to delete an Adventure.

2. include (SearchAdventure)
3. The System unsubscribes all subscribed Users to the Adventure
4. The System deletes the Adventure from the Database
5. include (SaveToDatabase)

Postconditions:

1. All the subscribed users must be unsubscribed from the deleted Adventure.
2. The Adventure must be deleted.

Alternative Sequence of Events: None.

USE CASE: SEARCH ADVENTURE(FR19)

Search Adventure

The Use case "SearchAdventure" allows the System to find an Adventure in the Database

Main Actors: System.

Secondary Actors: Database.

Preconditions:

1. The adventure must exist in the Database.

Main Sequence of Events:

1. The use case begins when the System is requested to search for an Adventure using the Name.
2. include (LoadFromDatabase)
3. IF the Adventure does NOT exist
 - a. The system returns an error message
4. ELSE the System returns the Adventure

Postconditions: None

Alternative Sequence of Events: None.

USE CASE: LOAD ADVENTURE (FR20)

Load Adventure

The use case "LoadAdventure" allows for the loading of an Adventure already present in the Database and which can be accessed by a user that has already taken part in the same Adventure.

Main Actors: User.

Secondary Actors: Database.

Preconditions:

1. At least one adventure must have been saved in the Database.
2. The User must be logged in.

Main Sequence of Events:

1. The Use Case begins when a User selects an Adventure where he is a Player or the DM.
2. The System retrieves the selected Adventure from the Database.
3. include (SearchAdventure)
4. The System loads the Adventure and Allows the User to play in it.

Postconditions: None.

Alternative Sequence of Events: None.

USE CASE: SAVE ADVENTURE (FR21)**Add Adventure**

The Use case "SaveAdventure" allows the System to save a new Adventure in the Database.

Main Actors: System, User.

Secondary Actors: Database.

Preconditions: None

Main Sequence of Events:

1. The use case begins when a registered User has created a new Adventure or when data in the Adventure Changes
2. include (SaveToDatabase)

Postconditions:

1. The Adventure must have been saved in the database.

Alternative Sequence of Events: None.

USE CASE: ADD ITEM (FR22)**AddItem**

The use case "AddItem" allows the DM to add a new Item to the Items List and in the Adventure.

Main Actors: DM.

Secondary Actors: Database

Preconditions:

1. The User must have selected the Adventure.
2. The DM must be logged in
3. The DM must own the Adventure

Main Sequence of Events:

1. The use case begins when the DM decides to add a new Item to the Items List.
2. include (SearchItem)
3. IF the Item already exists:
 - 3.1. The System shows an Error message
4. include (SaveToDatabase)

Postconditions: None

Alternative Sequence of Events: None.

USE CASE: REMOVE ITEM (FR23)

RemoveItem

The use case “RemoveItem” allows the DM to remove a pre-existing Item from the Items List.

Main Actors: DM.

Secondary Actors: Database.

Preconditions:

1. The Item’s List must not be empty.
2. The DM must be logged in
3. The DM must own the Adventure

Main Sequence of Events:

1. The use case begins when the DM decides to remove a pre-existing item from the Items List.
2. include (RemoveItem)
3. include (SaveToDatabase)

Postconditions: None

Alternative Sequence of Events: None.

USE CASE: SEARCH ITEM (FR24)

SearchItem

The use case “SearchItem” allows the System to search for a specific Item through the Items List by using the name.

Main Actors: System.

Secondary Actors: Database.

Preconditions:

1. The Item must exist in the Database.

Main Sequence of Events:

1. The use case begins when the System needs to search for a specific Item in the Items List.
2. include (LoadFromDatabase)
3. IF the Item does Not Exist
 - 3.1. The System returns an error message.
4. ELSE the System returns the Item.

Postconditions: None.

Alternative Sequence of Events: None.

USE CASE: SAVE ITEM (FR25)**Saveltem**

The use case "Saveltem" allows the DM to save a valid item to the Items List.

Main Actors: DM.

Secondary Actors: None.

Preconditions:

1. The DM must be logged in
2. The DM must own the Adventure

Main Sequence of Events:

1. The use case begins when the DM decides to save a created item.
2. IF the item is NOT Valid:
 - a. The system shows a Warning message
3. ELSE include (SaveToDatabase)

Postconditions:

1. The Item must be saved in the Items List.

Alternative Sequence of Events: None.

USE CASE: EDIT ITEM (FR26)**EditItem**

The use case "EditItem" allows the DM to edit a selected Item.

Main Actors: DM.

Secondary Actors: Database.

Preconditions:

1. The Item must be in the Item List.
2. The DM must be logged in
3. The DM must own the Adventure

Main Sequence of Events:

1. The use case begins when the DM decides to edit a pre-existing item.
2. include (SearchItem)
3. The DM edits everything he wants.

Postconditions: None

Alternative Sequence of Events: None.

USE CASE: VIEW ITEMS (FR27)

ViewItems

The Use case "ViewItems" allows the DM to visualize all the Items inside the Items List in a scroll view.

Main Actors: DM.

Secondary Actors: None.

Preconditions: None

Main Sequence of Events:

1. The Use Case begins when the Dungeon Master enters the Items tab.
2. include (SearchItem)
3. The System shows all the Items in the Items list.

Postconditions:

1. The scroll view must be populated

Alternative Sequence of Events: None.

USE CASE: ADD MAP (FR28)

AddMap

The use case "AddMap" allows the DM to add a new map in the Maps List.

Main Actors: DM.

Secondary Actors: None.

Preconditions:

1. The DM must be logged in
2. The DM must own the Adventure

Main Sequence of Events:

1. The use case begins when the DM decides to add a new map in the Maps List.

2. include (SearchMap)
3. IF the Map already exists
 - 3.1. The System shows a Warning message
4. ELSE include (SaveToDatabase)

Postconditions: None

Alternative Sequence of Events: None.

USE CASE: REMOVE MAP (FR29)

RemoveMap

The use case “RemoveMap” allows the DM to remove a pre-existing Map from the Maps List.

Main Actors: DM.

Secondary Actors: Database.

Preconditions:

1. The DM must be logged in
2. The DM must own the Adventure

Main Sequence of Events:

1. The use case begins when the DM decides to remove a Map from the Maps List.
2. include (SearchMap)
3. include (SaveToDatabase)

Postconditions: None

Alternative Sequence of Events: None.

USE CASE: SEARCH MAP (FR30)

SearchMap

The use case “SearchMap” allows the DM to search for a specific Map inside the Maps List.

Main Actors: DM.

Secondary Actors: Database.

Preconditions:

1. The Maps List must not be empty.

Main Sequence of Events:

1. The use case begins when the DM decides to search for a Map from the Maps List.
2. include (ReadFromDatabase)
3. IF the Map is found

- 3.1. The Map is shown from the Maps List.
4. ELSE
 - 4.1. The System shows a Warning message.

Postconditions: None

Alternative Sequence of Events: None.

USE CASE: SAVE MAP (FR31)

SaveMap

The use case “SaveMap” allows the DM to save a Map to the Maps List.

Main Actors: DM.

Secondary Actors: None.

Preconditions:

1. The DM must be logged in
2. The DM must own the Adventure

Main Sequence of Events:

1. The use case begins when the DM decides to save a Map into or from the Map List.
2. IF the Map is valid
 - 2.1. Include (SaveToDatabase)
3. ELSE
 - 3.1. The System shows a Warning Alert.

Postconditions: None

Alternative Sequence of Events: None.

USE CASE: LOAD MAP (FR32)

LoadMap

The use case “LoadMap” allows the DM to choose a pre-existing Map in the Maps List.

Main Actors: DM.

Secondary Actors: Database.

Preconditions:

1. The DM must have a map to load in the Maps list.
2. The DM must be logged in
3. The DM must own the Adventure

Main Sequence of Events:

1. The use case begins when the DM choses a Map from the Maps List to be loaded.
2. include (SearchMap)
3. The selected Map gets shown in the Main tab.

Postconditions: None

Alternative Sequence of Events: None.

USE CASE: ADD ENTITY (FR33)**AddEntity**

The use case "AddEntity" allows the DM to add an Entity to the Map.

Main Actors: DM

Secondary Actors: Database.

Preconditions:

1. The Entity must exist in the Database.
2. The DM must be logged in
3. The DM must own the Adventure

Main Sequence of Events:

1. The use case begins when the DM decides to add an Entity to the Map.
2. include (SearchEntity)
3. IF the Entity is Not Valid
 - 3.1. System shows a Warning message.
4. IF the Entity is Valid
 - 4.1. The selected entity is added to the Map.
 - 4.2. include (SaveToDatabase)

Postconditions: None.

Alternative Sequence of Events: None.

USE CASE: ATTACK ENTITY (FR34)**AttackEntity**

The use case "AttackEntity" allows the Users to decide the target of their attack during Combat.

The User communicates their chosen target to the DM and the DM executes the Attack.

Main Actors: DM

Secondary Actors: Player, Database.

Preconditions: None

Main Sequence of Events:

1. The User selects an Entity from the Entities that are in the map.
2. include (SearchEntity)
3. IF User wants to damage the target
 - 3.1. The DM damages the Entity
4. ELSE
 - 4.1. Include(SetEntityState)
5. include(SaveToDatabase)

Postconditions: None.

Alternative Sequence of Events: None.

USE CASE: REMOVE ENTITY (FR35)

RemoveEntity

The use case "RemoveEntity" allows the removal of an Entity from the Map when the DM decides so.

Main Actors: DM

Secondary Actors: Database.

Preconditions:

1. The DM must be logged in
2. The DM must own the Adventure

Main Sequence of Events:

1. The use case begins when an Entity has to be removed from Combat due to DM choice.
2. include (SearchEntity)
3. IF the DM chooses to remove the Entity from Combat.
 - 3.1. The Entity is removed from Combat.
4. include(SaveToDatabase)

Postconditions: None.

Alternative Sequence of Events: None.

USE CASE: SET ENTITY STATE (FR36)

SetEntityState

The use case "SetEntityState" allows the DM to set the State of an entity.

Main Actors: DM

Secondary Actors: Database.

Preconditions:

1. The Entity must be selected.
2. The DM must be logged in
3. The DM must own the Adventure

Main Sequence of Events:

1. The use case begins when the DM decides to set the State of an entity.
2. include (SearchEntity)
3. IF the Entity is Not Valid
 - 3.1. System shows a Warning Alert.
4. IF the Entity is Valid
 - 4.1. The DM changes or adds the selected entity state
 - 4.2. The state of the entity is changed and saved
5. include(SaveToDatabase)

Postconditions: None

Alternative Sequence of Events: None.

USE CASE: SEARCH ENTITY (FR37)**SearchEntity**

The use case "SearchEntity" must allow the System to search for an Entity in the map.

Main Actors: System.

Secondary Actors: Database.

Preconditions: None.

Main Sequence of Events:

1. The use case begins when the System needs to search for a specific Entity in combat.
2. Include (LoadFromDatabase)
3. IF the Entity is in the Map
 - 3.1. The System Returns the Entity.

Postconditions: None.

Alternative Sequence of Events: None.

USE CASE: CHANGE TILE(FR38)**ChangeTile**

The use case "ChangeTile" must allow the DM to change the image of a specific Tile.

Main Actors: DM.

Secondary Actors: Database.

Preconditions:

1. A Map must be loaded
2. The DM must be logged in
3. The DM must own the Adventure

Main Sequence of Events:

1. The Use Case begins when the DM decides to change the background image of a specific Tile of the loaded Map.
2. The System cycles through the images available.
3. include (SaveToDatabase)

Postconditions: None.

Alternative Sequence of Events: None.

USE CASE: VIEW MAPS(FR39)

ViewMaps

The use case "ViewMaps" must allow the DM to visualize all the Maps saved in the Maps List

Main Actors: DM.

Secondary Actors: Database.

Preconditions:

1. The DM must be logged in
2. The DM must own the Adventure

Main Sequence of Events:

1. The Use Case begins when the Dungeon Master enters the Maps tab.
2. Include (SearchMap)
3. The system populates the scroll view with the results from the Database

Postconditions: None.

Alternative Sequence of Events: None.

USE CASE: EDIT CHARACTER (FR40)

EditCharacter

The use case "EditCharacter" allows Players/DM to edit Characters.

Main Actors: Player, DM.

Secondary Actors: Database.

Preconditions:

1. The User must be logged in an adventure.

Main Sequence of Events:

1. The use case begins when the Player or DM decides to edit a Character Sheet.
2. include (SearchCharacter)
3. The Player/DM edits whatever they want.

Postconditions: None

Alternative Sequence of Events: None.

USE CASE: VIEW CHARACTERS (FR41)

ViewCharacters

The use case "ViewCharacters" allows the Players/DM to visualize their own and other Players' Characters.

Main Actors: Player, DM.

Secondary Actors: Database.

Preconditions:

1. The User must be logged in an adventure.

Main Sequence of Events:

1. The Use Case begins when a Player/DM enters the Characters tab.
2. include (SearchCharacter)
3. The Character gets shown.

Postconditions: None

Alternative Sequence of Events: None.

USE CASE: SAVE CHARACTER (FR42)

SaveCharacter

The use case "SaveCharacter" allows the System to save a newly edited Character.

Main Actors: User.

Secondary Actors: Database.

Preconditions:

1. The User must be logged in

Main Sequence of Events:

1. The use case begins when a User clicks on “Save” to save a newly edited Character
2. IF the Character is NOT Valid:
 - 2.1. The system shows a Warning alert
3. ELSE include (SaveToDatabase)

Postconditions: None

Alternative Sequence of Events: None

USE CASE: ADD CHARACTER (FR43)

AddCharacter

The use case “AddCharacter” allows a User to create a new Character to use in a pre-existing Adventure

Main Actors: User, Player.

Secondary Actors: Database.

Preconditions:

1. The User must be logged in

Main Sequence of Events:

1. The use case begins when the User decides to participate in a pre-existing new adventure.
2. The User inserts the name of the Adventure he wants to participate in.
3. include (SearchAdventure)
4. IF the Adventure exists
 - 4.1. The User creates a character
 - 4.1.1. include (SearchCharacter)
 - 4.1.2. IF the character is valid
 - 4.1.2.1. include (SaveToDatabase)
 - 4.1.3. ELSE
 - 4.1.3.1. The System shows an Error message

Postconditions: None

Alternative Sequence of Events: None

USE CASE: REMOVE CHARACTER (FR44)

RemoveCharacter

The use case “RemoveCharacter” allows the DM to remove a character from the adventure.

Main Actors: DM.

Secondary Actors: Database.

Preconditions:

1. The character must be present in the Characters List.

2. The DM must be logged in
3. The DM must own the Adventure

Main Sequence of Events:

1. The use case begins when the DM decides to remove a Character from his current Adventure by clicking "Delete"
2. The Character gets removed
3. include (SaveToDatabase)

Postconditions: None

Alternative Sequence of Events: None.

USE CASE: SEARCH CHARACTER(FR45)

Search Character

The Use case "SearchCharacter" allows the System to find a Character in the Database

Main Actors: System.

Secondary Actors: Database.

Preconditions:

1. The Character must exist in the Database.

Main Sequence of Events:

1. The use case begins when the System is requested to search for a Character using the Name.
2. include (LoadFromDatabase)
3. IF the Character does NOT exist
 - a. The system returns an error message
4. ELSE the System returns the Character

Postconditions: None

Alternative Sequence of Events: None.

USE CASE: THROW DICE(FR46)

Throw Dice

The Use case "ThrowDice" allows the User to obtain a random value based on the number of dice thrown

Main Actors: User.

Secondary Actors: System.

Preconditions:

1. The User must be logged in

Main Sequence of Events:

1. The use case begins when the User clicks on “Throw”
2. IF the Dice list is NOT empty
 - 2.1. The System virtually throws all the dices contained in the Dice list
 - 2.2. The System sends a message in the Chat containing the result
 - 2.3. include (SendMessage)

Postconditions: None

Alternative Sequence of Events: None.

USE CASE: ADD DICE TO LIST (FR47)

AddDiceToList

The use case “AddDiceToList” allows Players to add a Dice to their own Dice List.

Main Actors: Players.

Secondary Actors: System.

Preconditions:

1. The Player who performs the action must be logged in.

Main Sequence of Events:

1. The use case begins when a Player decides to add a Dice to his own Dice List.
2. The selected dice gets added to the Dice list.

Postconditions: None

Alternative Sequence of Events: None.

USE CASE: SAVE TO DATABASE(FR48)

SaveToDatabase

The use case “SaveToDatabase” allows the System to write data to the Database

Main Actors: System, Database.

Secondary Actors: None.

Preconditions:

1. The data that needs to be saved must be valid.

Main Sequence of Events:

1. The use case begins when the System needs to save data to the Database
2. The System dumps the data into the Database

Postconditions: None

Alternative Sequence of Events: None.

USE CASE: LOAD FROM DATABASE(FR49)

LoadFromDatabase

The use case "LoadFromDatabase" allows the System to read data from the Database

Main Actors: System, Database.

Secondary Actors: None.

Preconditions: None

Main Sequence of Events:

1. The use case begins when the System needs to read data from the Database
2. IF the requested data does NOT exists
 - 2.1. The system shows an Error message.
3. ELSE
 - 3.1. The System returns the requested data

Postconditions:

1. The System must have received the requested data or no result.

Alternative Sequence of Events: None.

USE CASE: AUTHENTICATE(FR50)

Authenticate

The use case "Authenticate" allows the Users to authenticate with the System

Main Actors: User.

Secondary Actors: Database.

Preconditions: None

Main Sequence of Events:

1. The use case begins when the User tries to login
2. include (ReadFromDatabase)
3. IF The credentials are incorrect
 - 3.1. The System shows an error and aborts the request
4. ELSE The System lets the User in

Postconditions:

1. The User must be correctly authenticated with the System

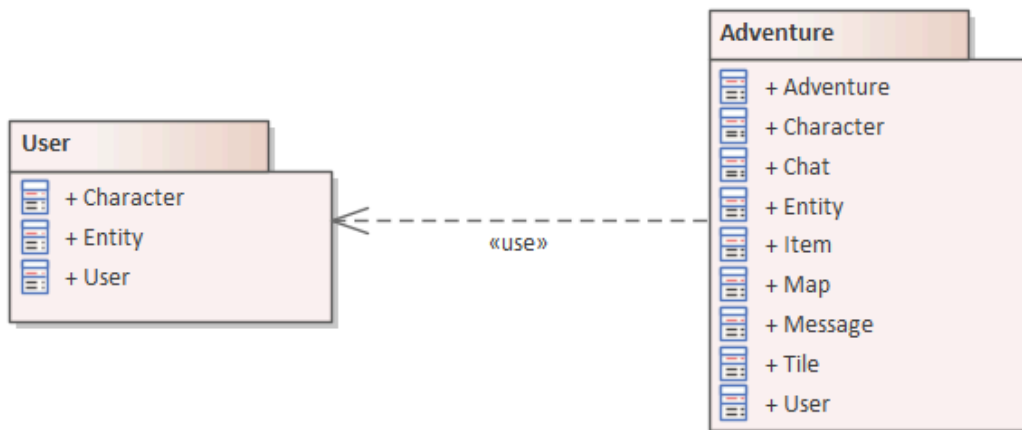
Alternative Sequence of Events: None.

REQUIREMENTS MATRIX:

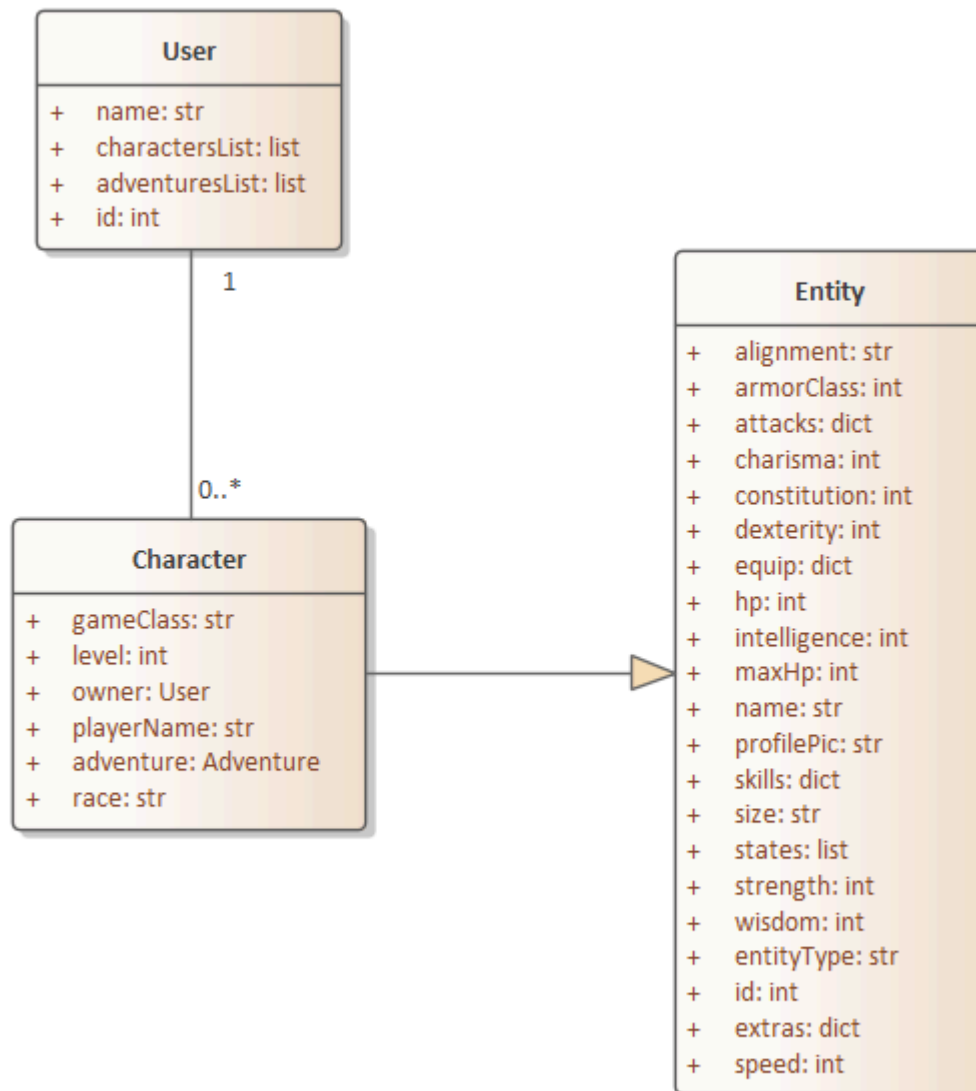
[illegible]

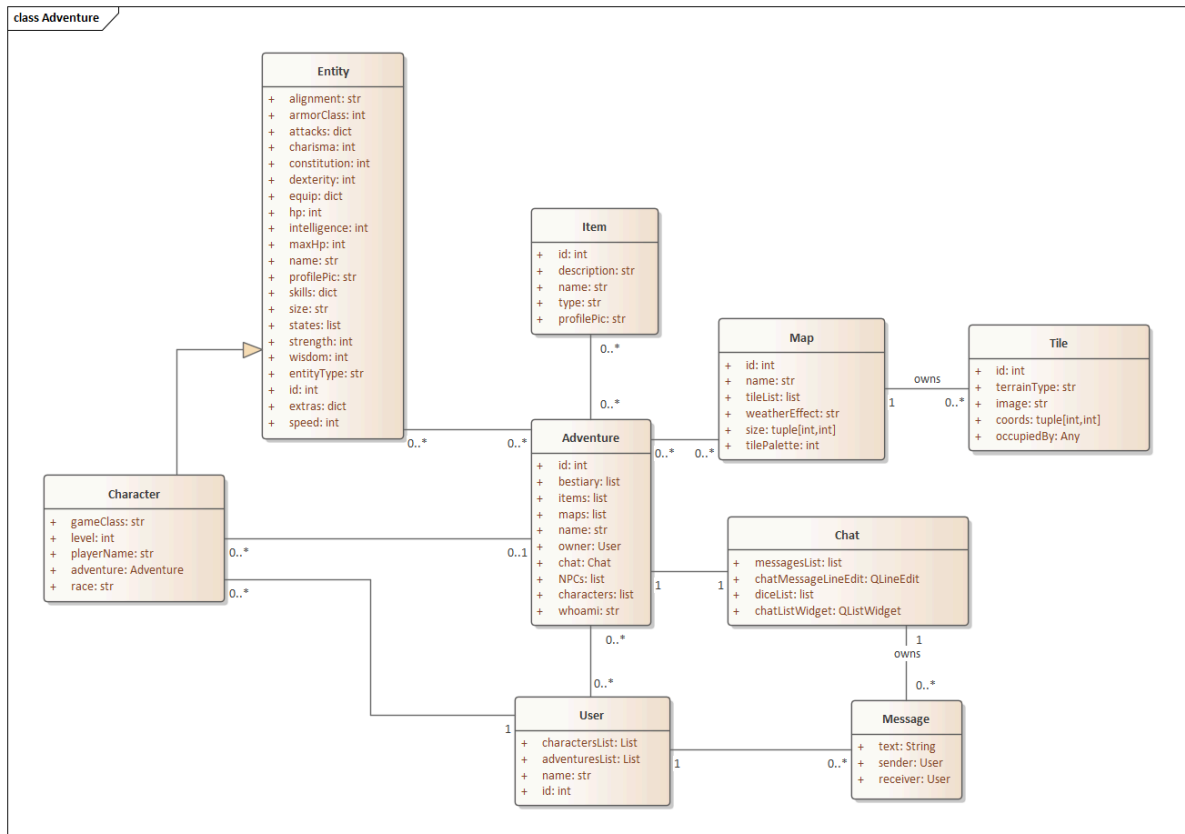
CLASS ANALYSIS DIAGRAMS:

pkg Analysis Classes



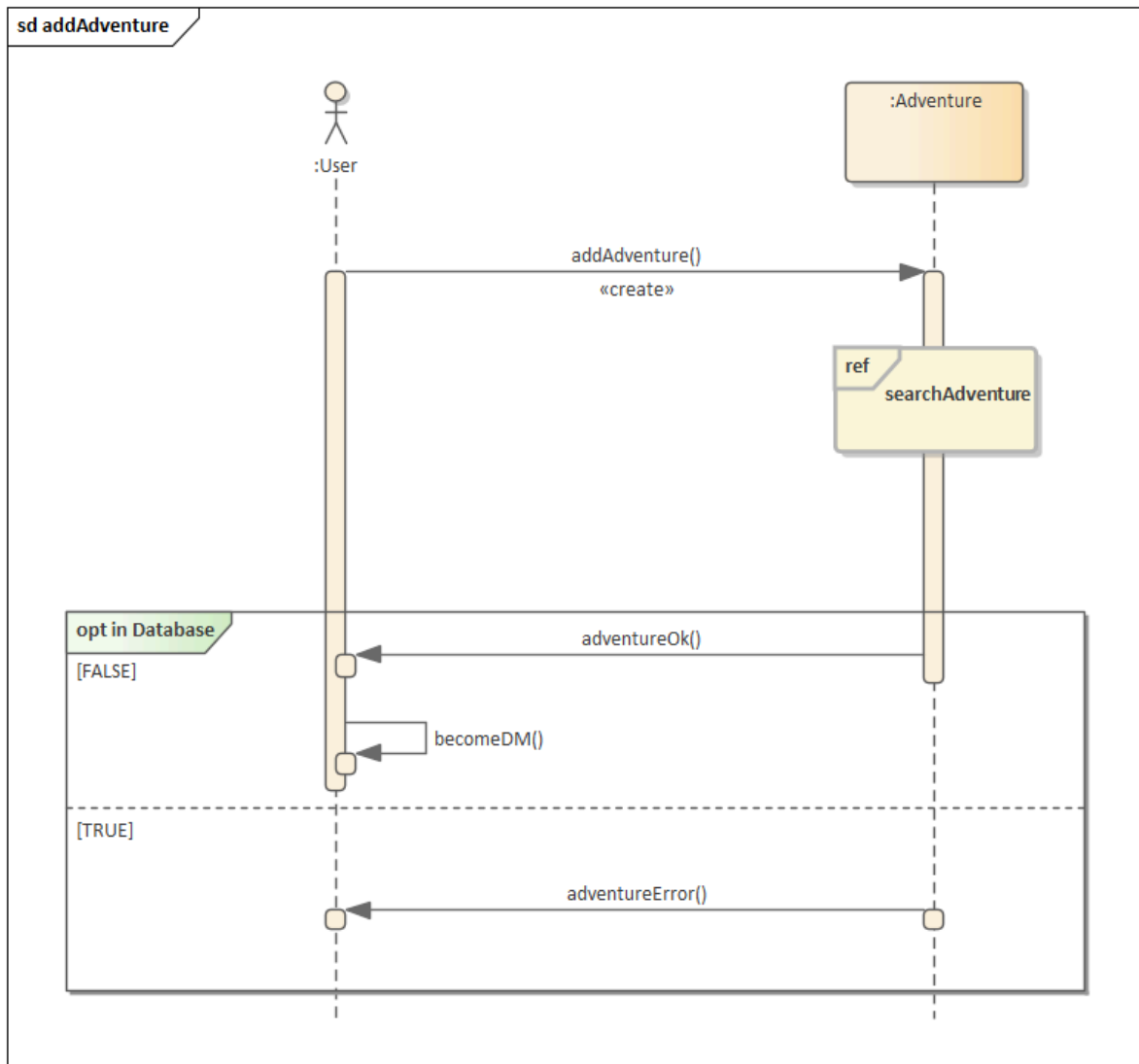
class User



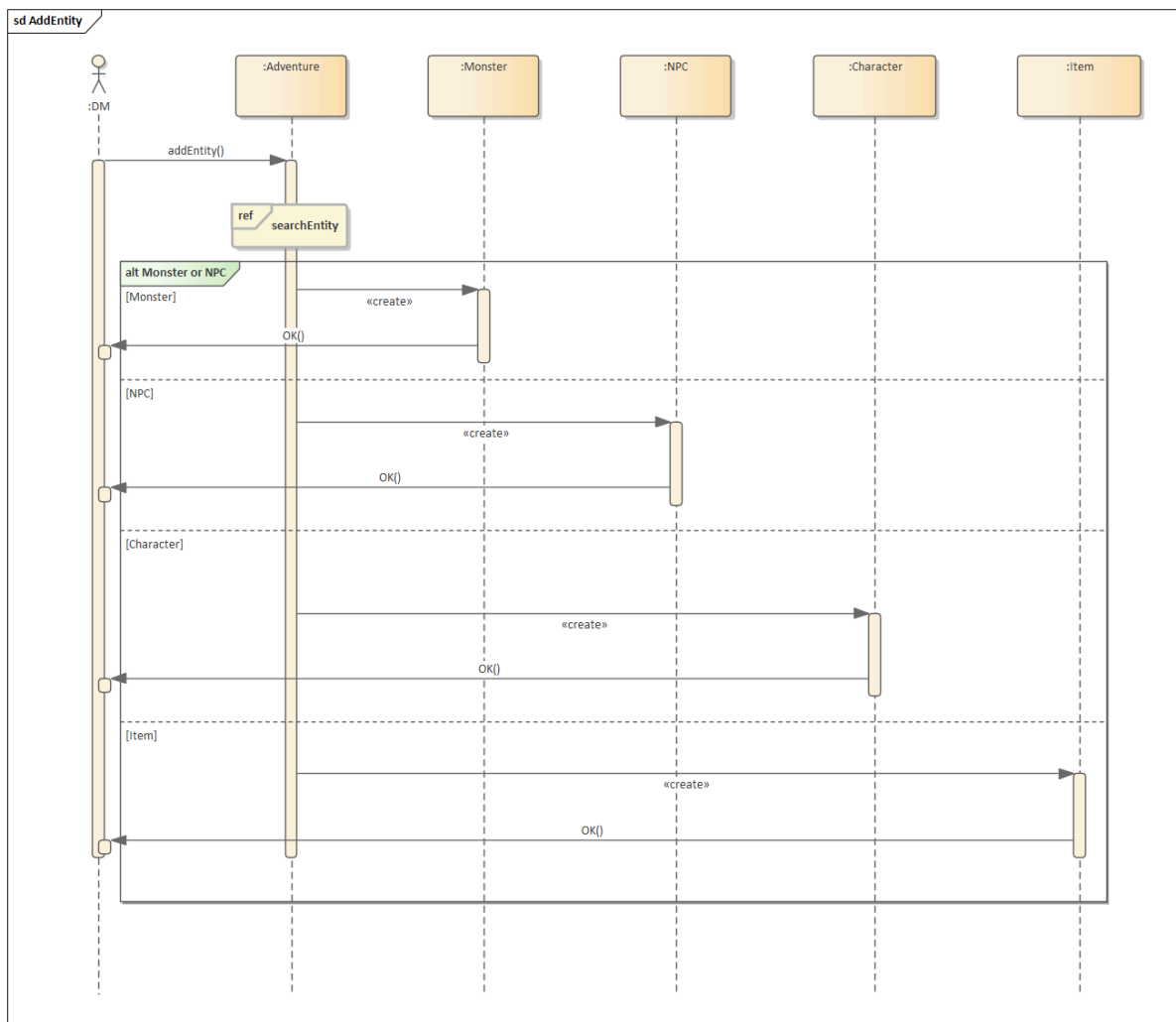


SEQUENCE DIAGRAMS:

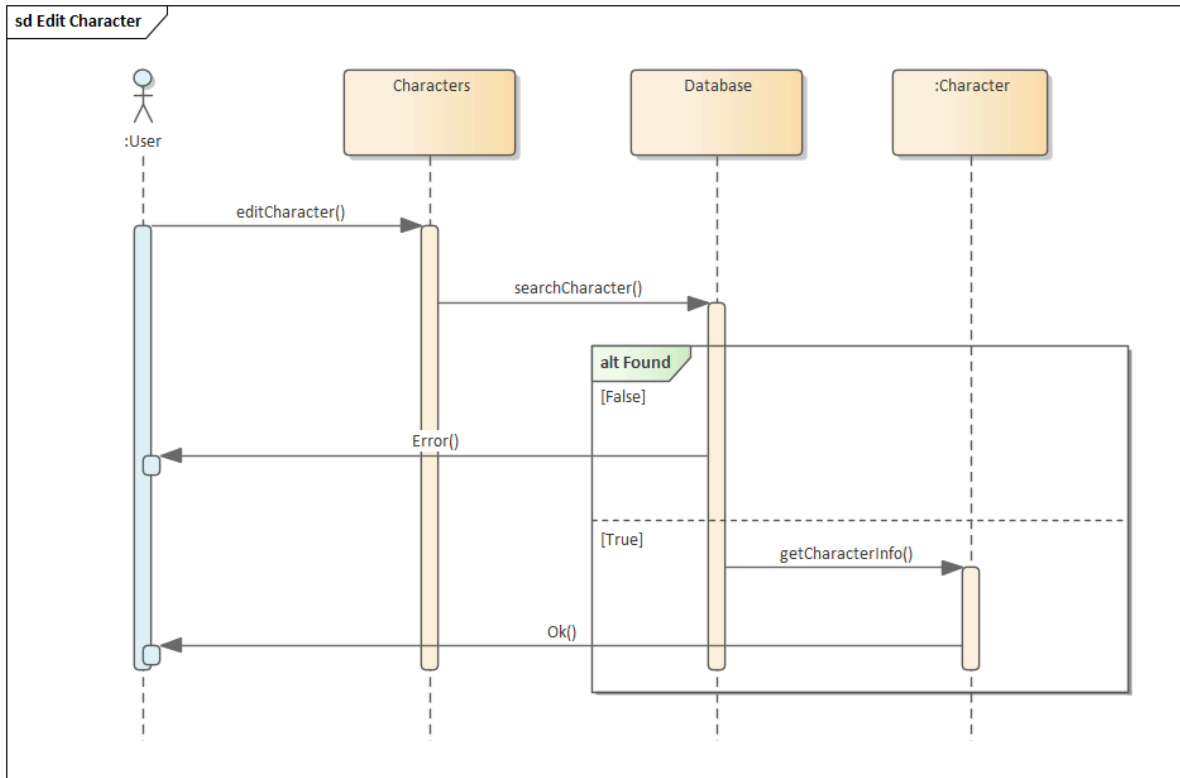
addAdventure SEQUENCE DIAGRAM:



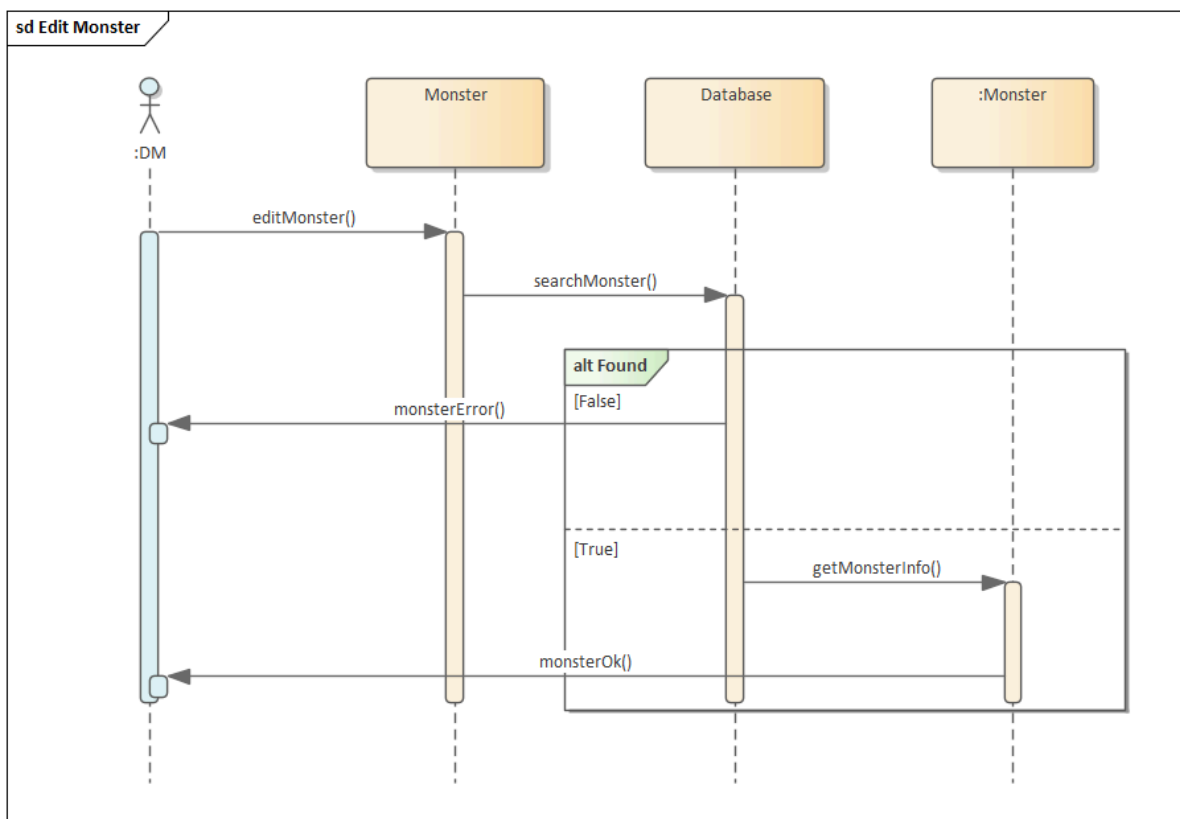
addEntity SEQUENCE DIAGRAM:



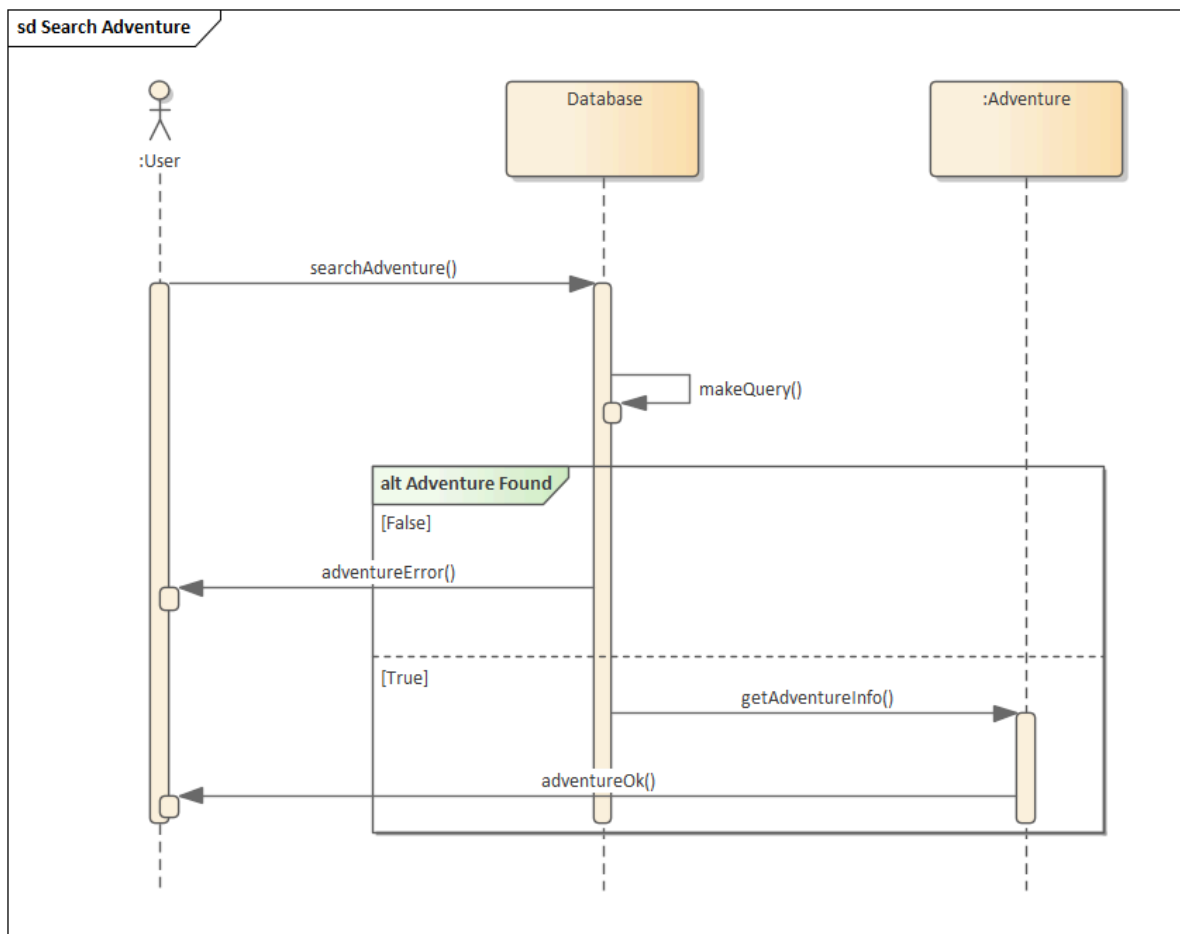
editCharacter SEQUENCE DIAGRAM:



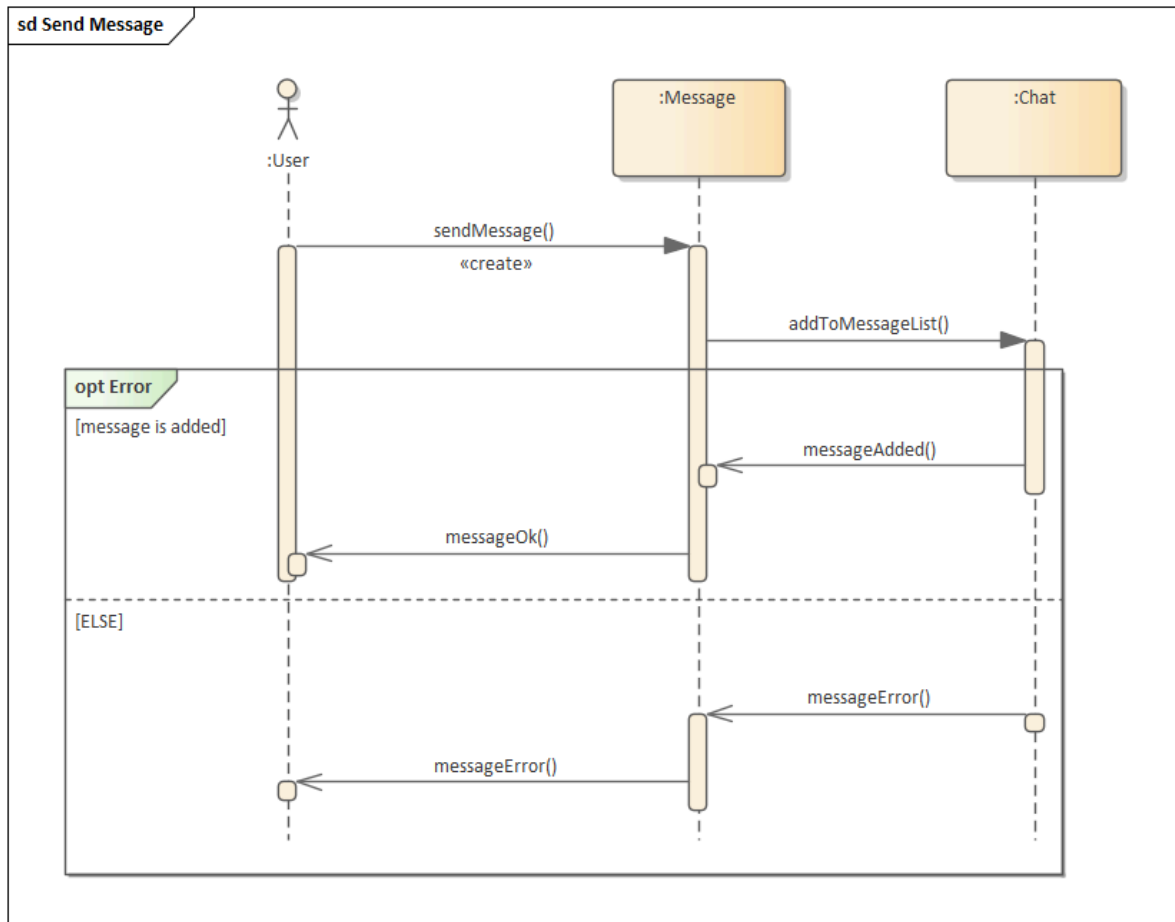
editMonster SEQUENCE DIAGRAM:



searchAdventure SEQUENCE DIAGRAM:



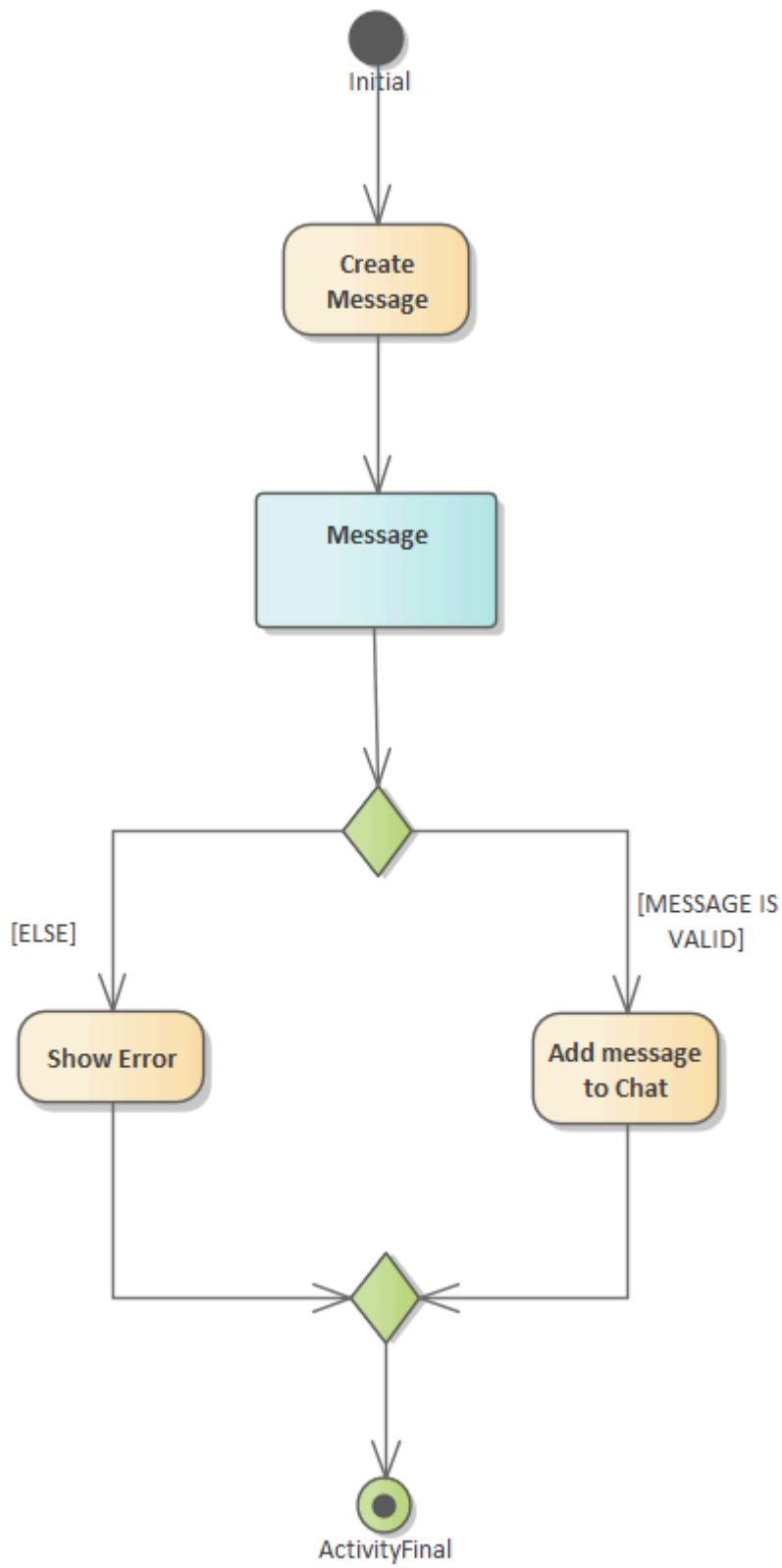
sendMessage SEQUENCE DIAGRAM:



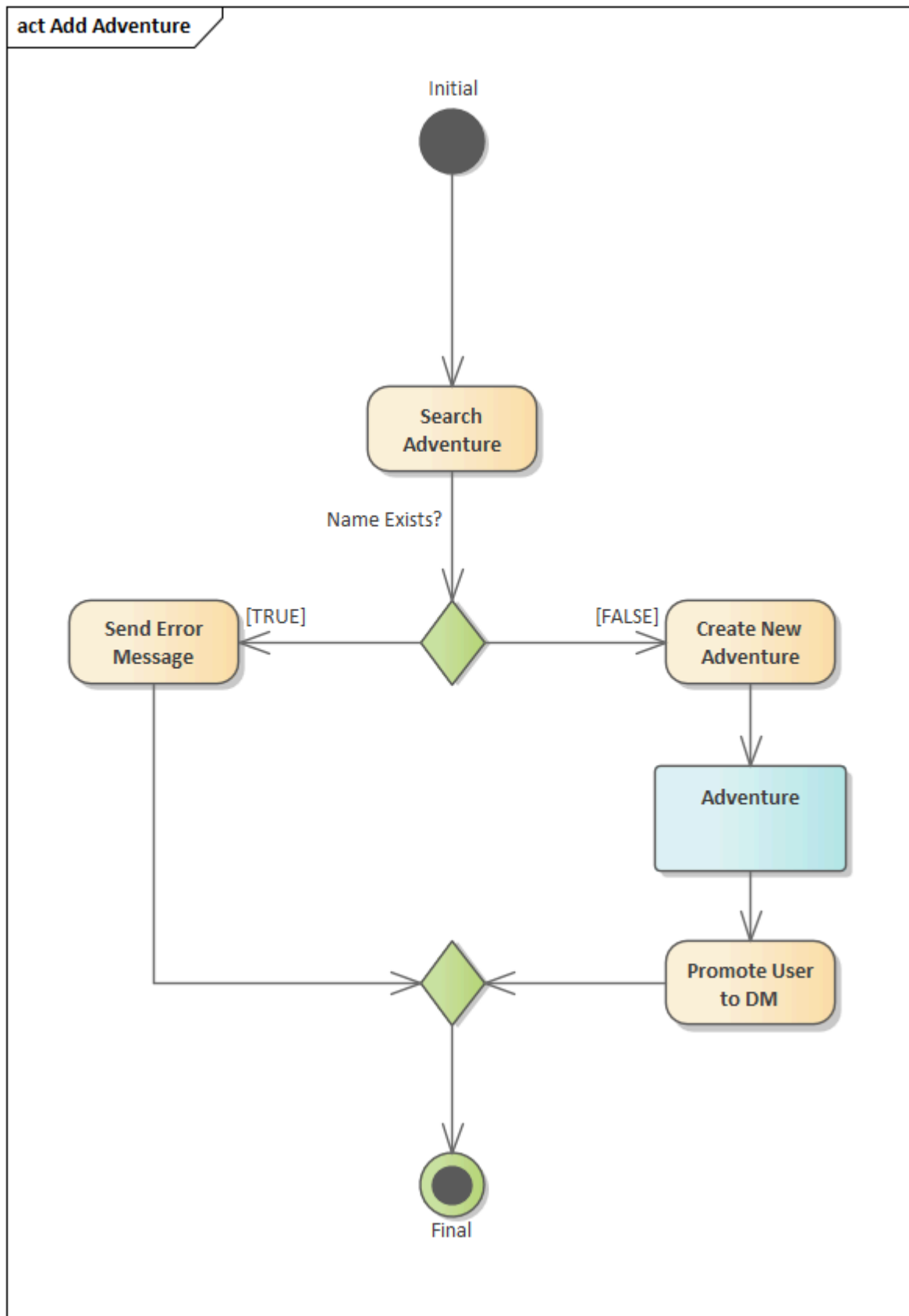
ACTIVITY DIAGRAMS:

sendMessage ACTIVITY DIAGRAM:

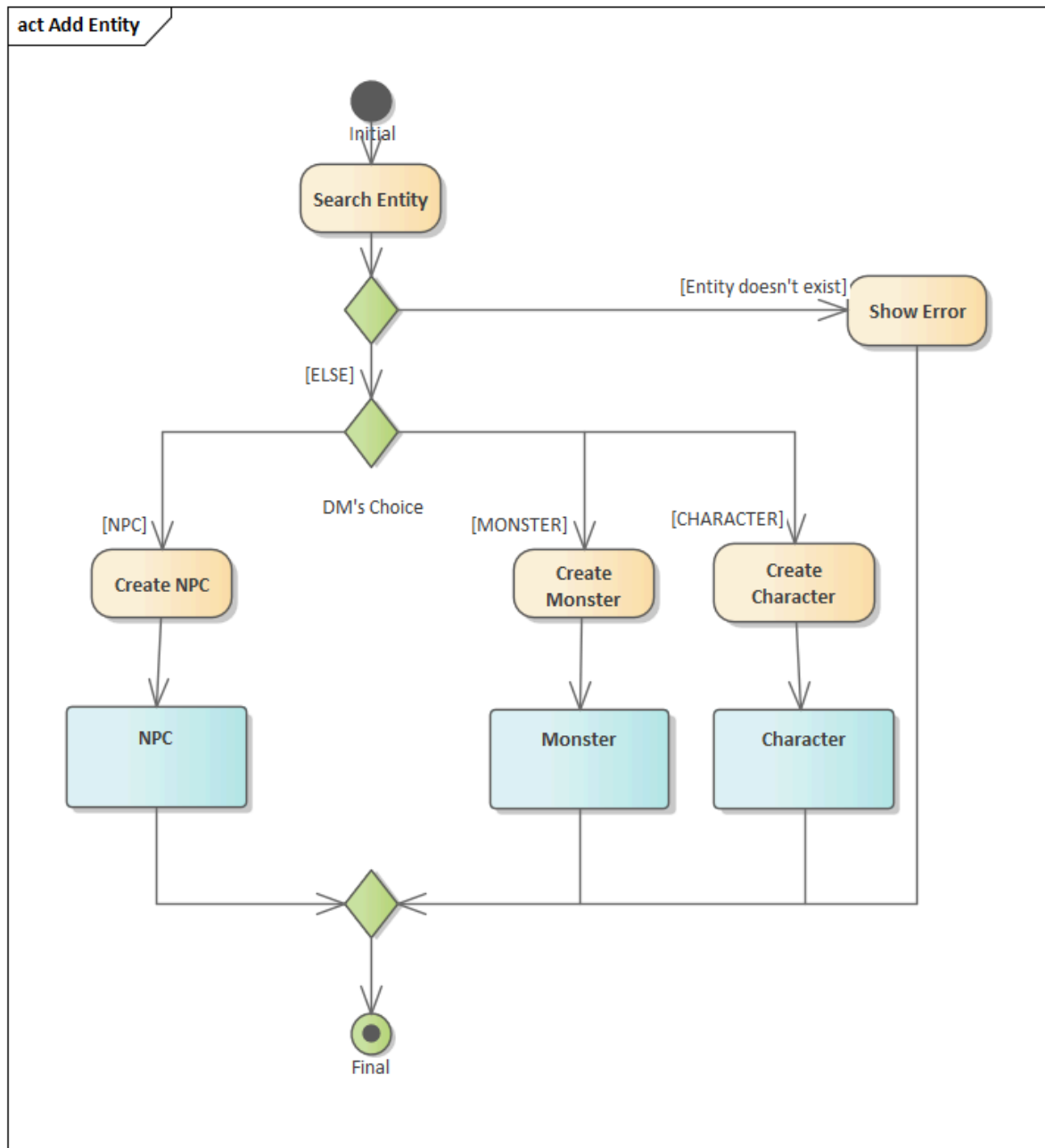
act Send Message



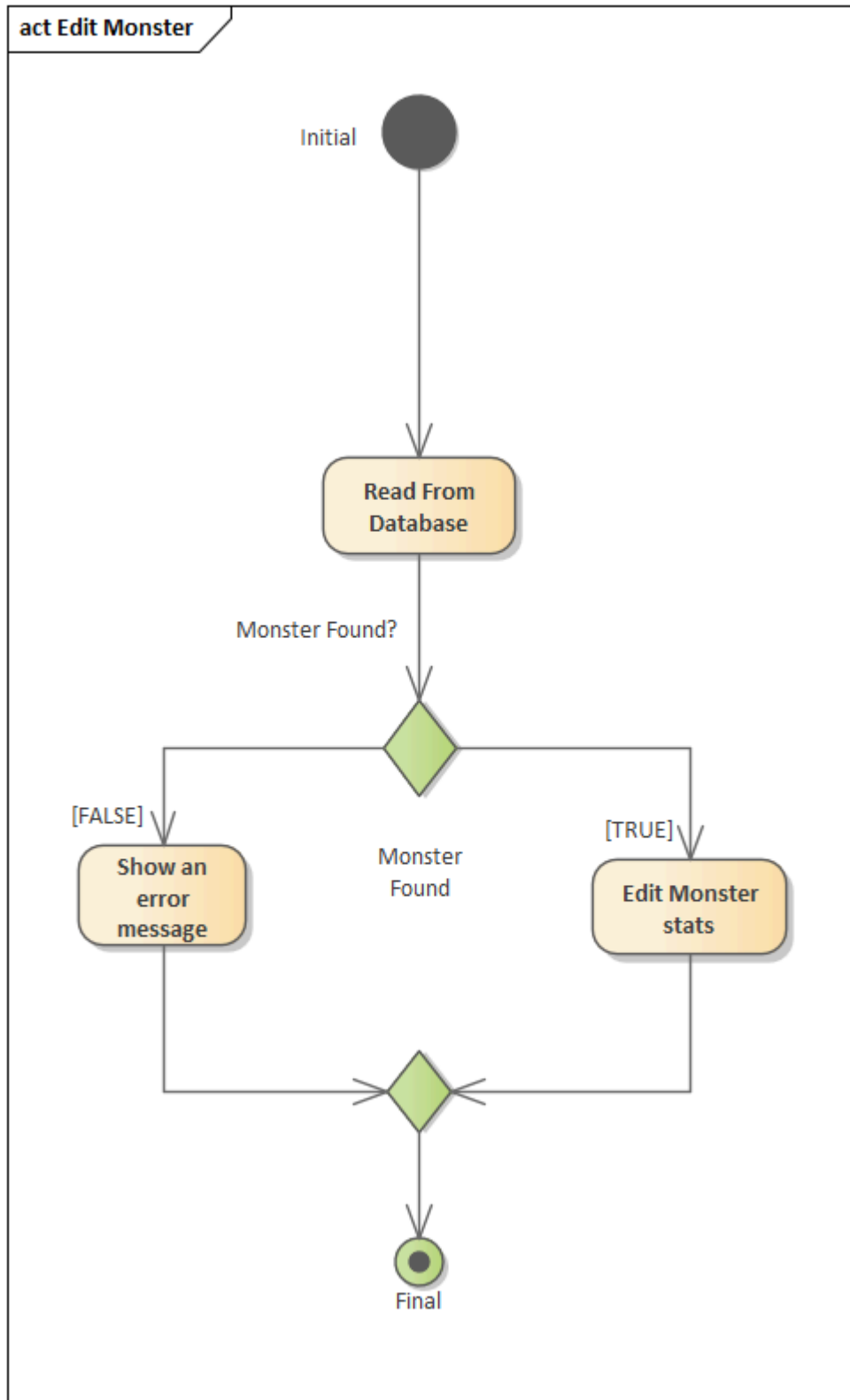
addAdventure ACTIVITY DIAGRAM:



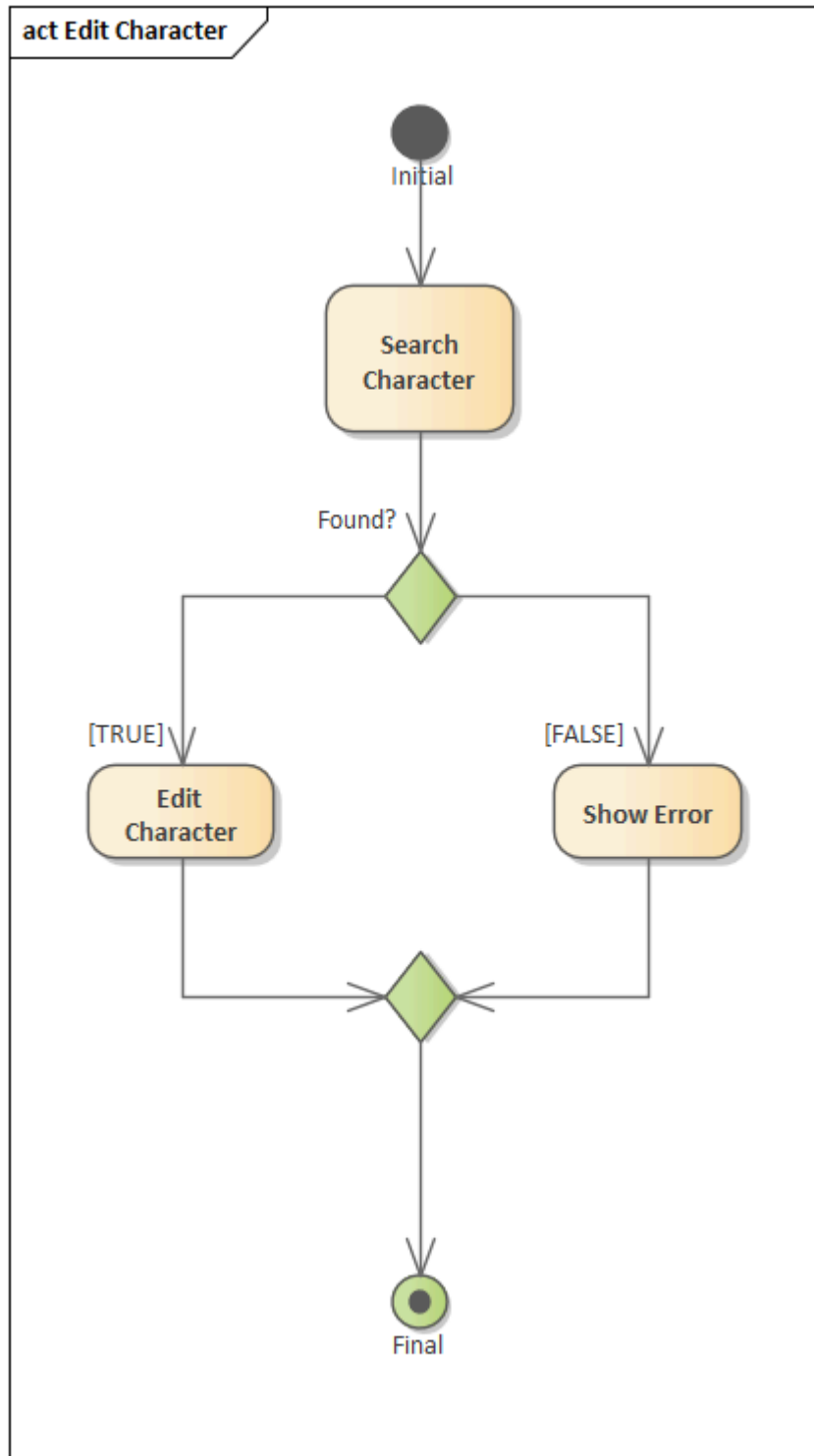
addEntity ACTIVITY DIAGRAM:



editMonster ACTIVITY DIAGRAM:

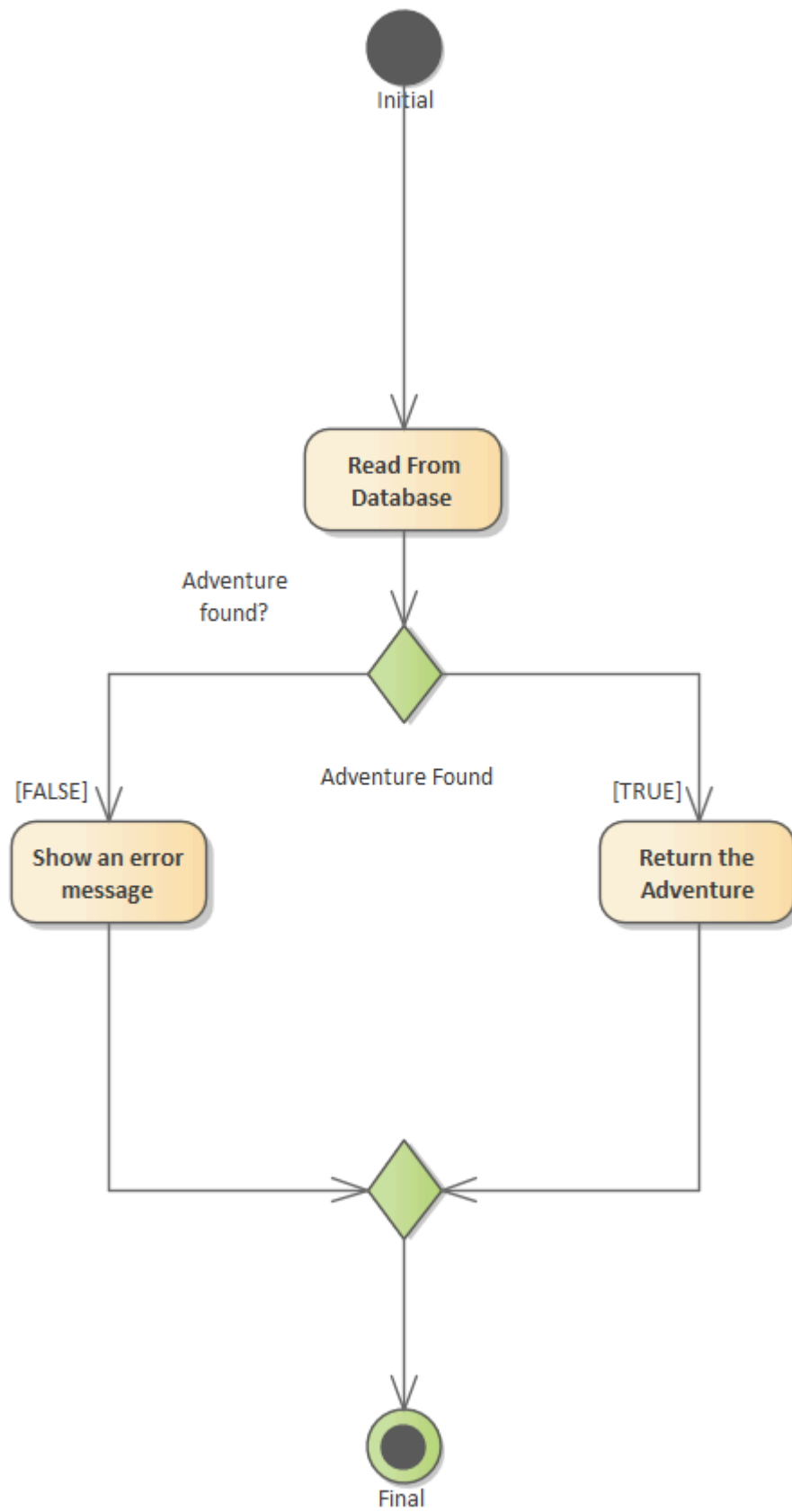


editCharacter ACTIVITY DIAGRAM:

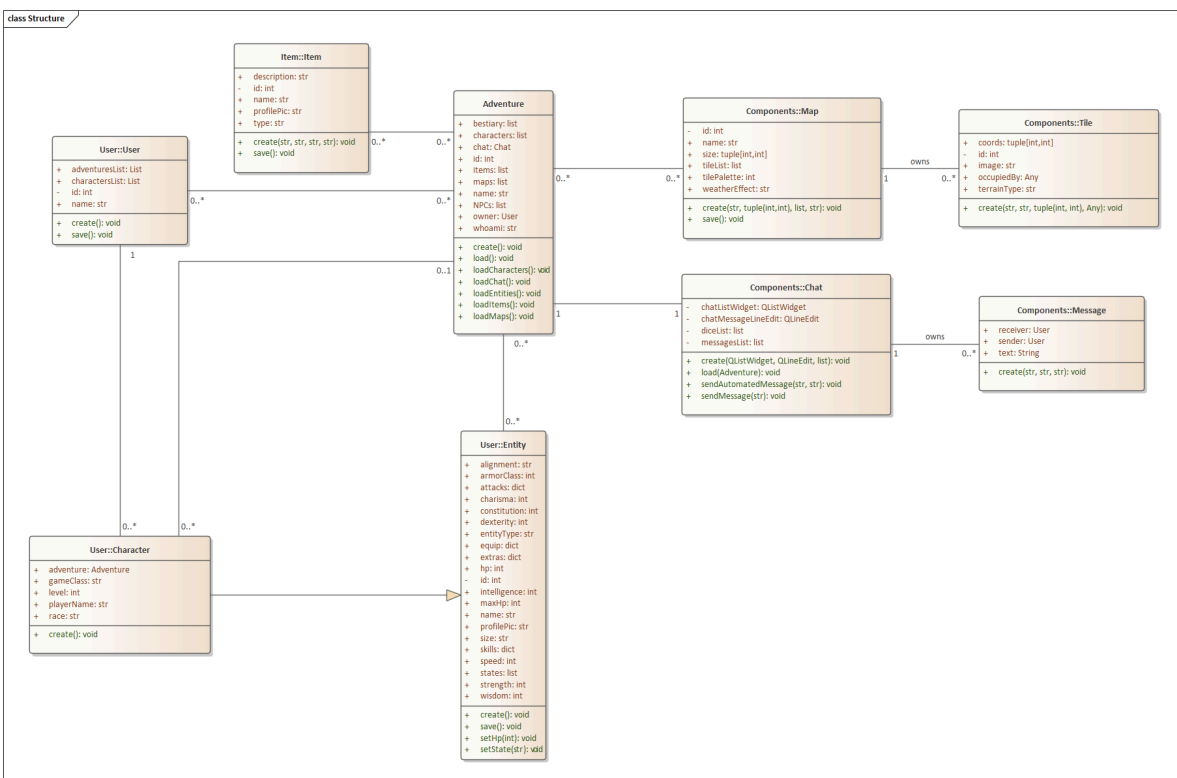
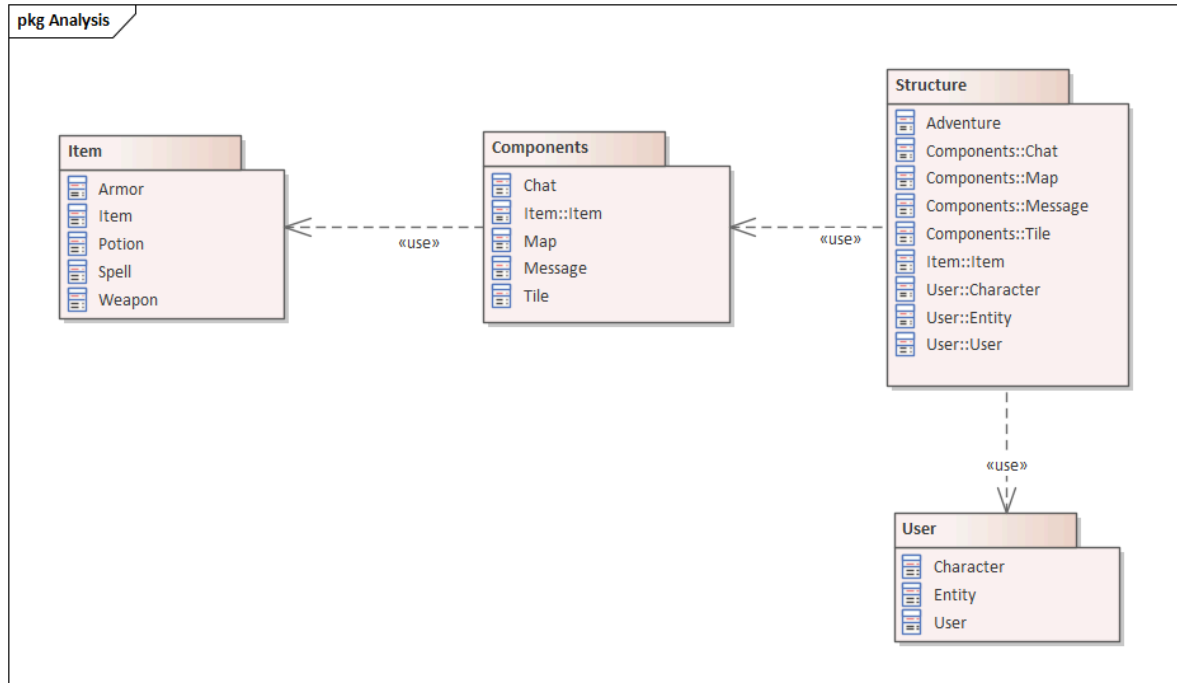


searchAdventure ACTIVITY DIAGRAM:

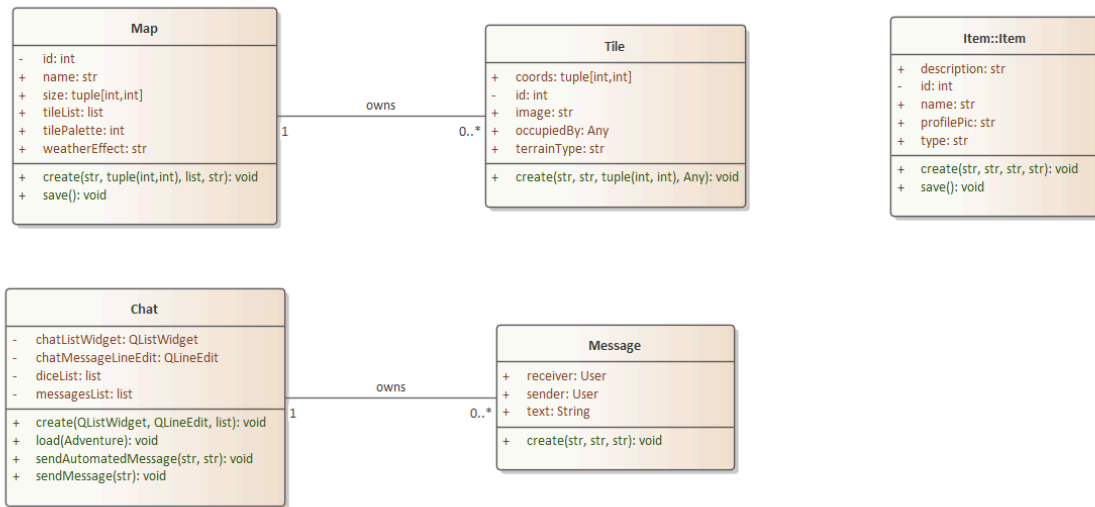
act Search Adventure



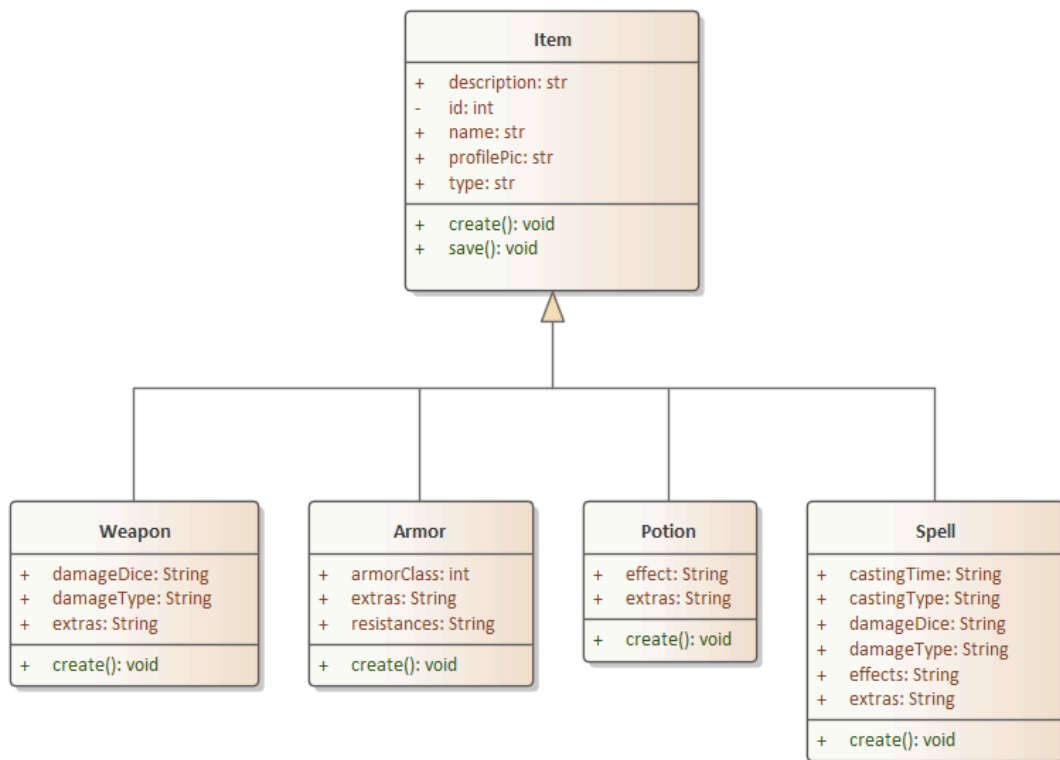
DESIGN CLASSES DIAGRAMS:



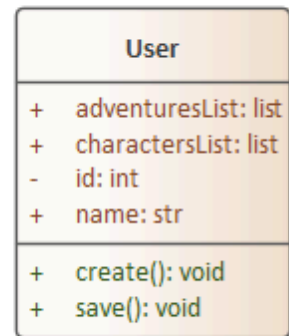
class Components



class Item

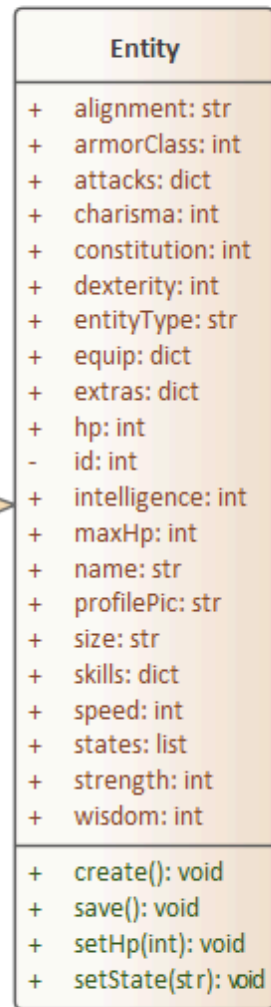
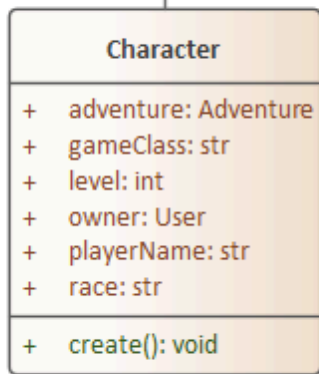


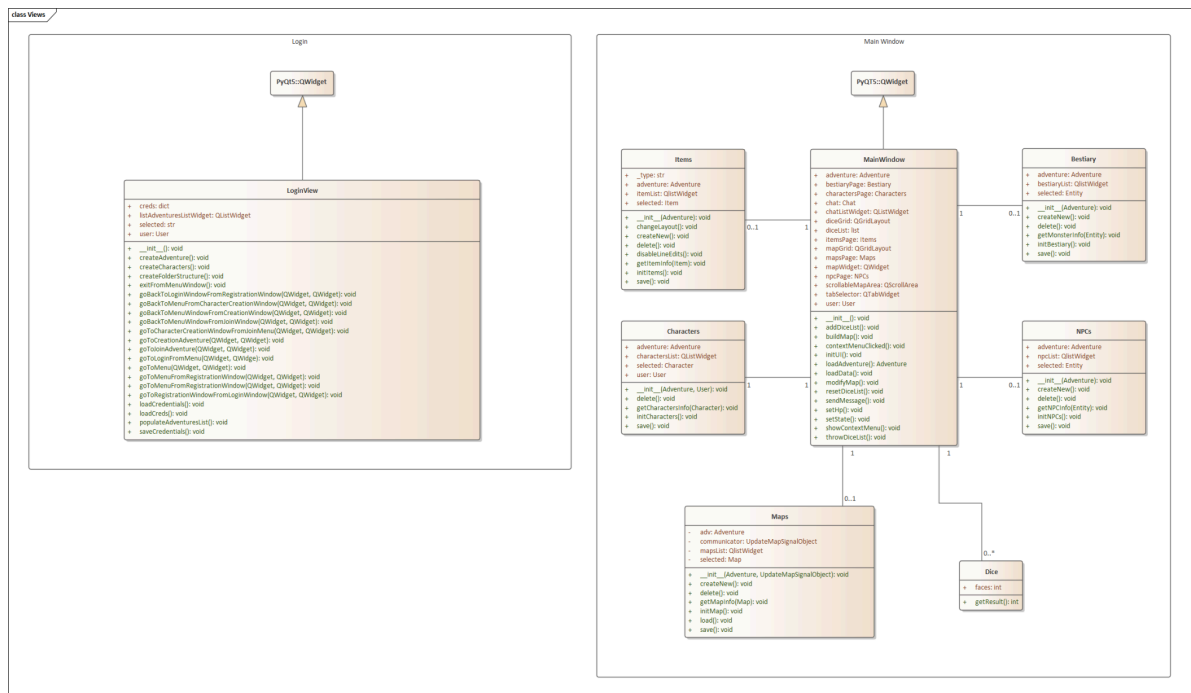
class User



1

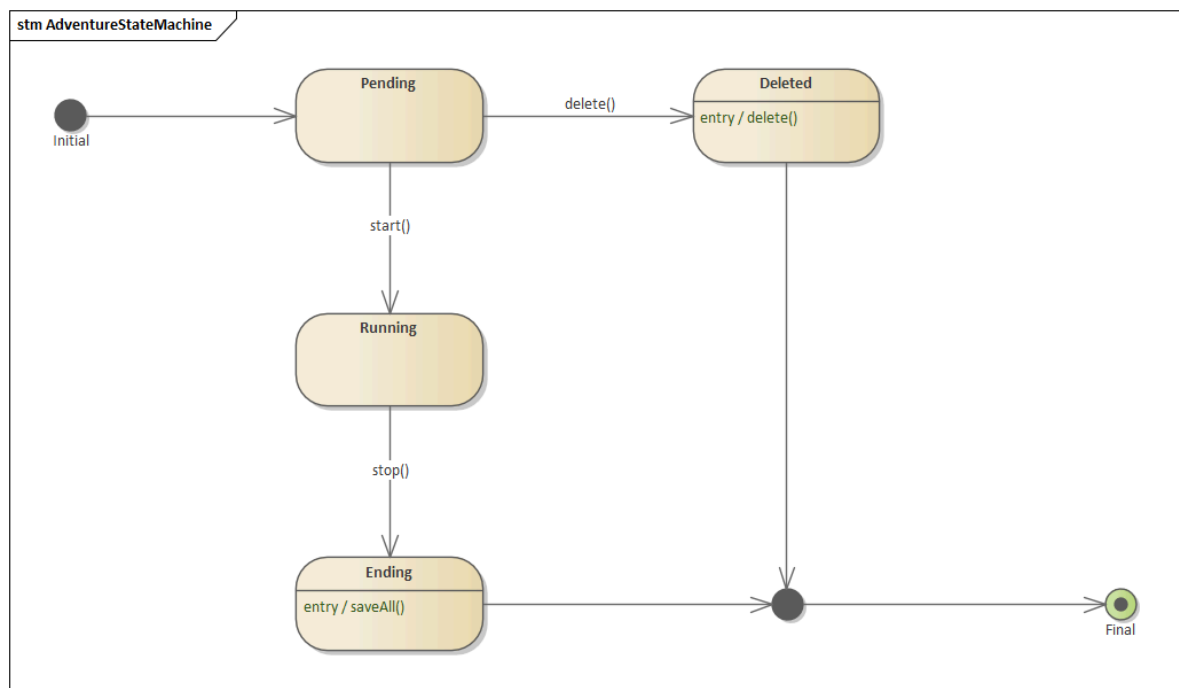
0..*



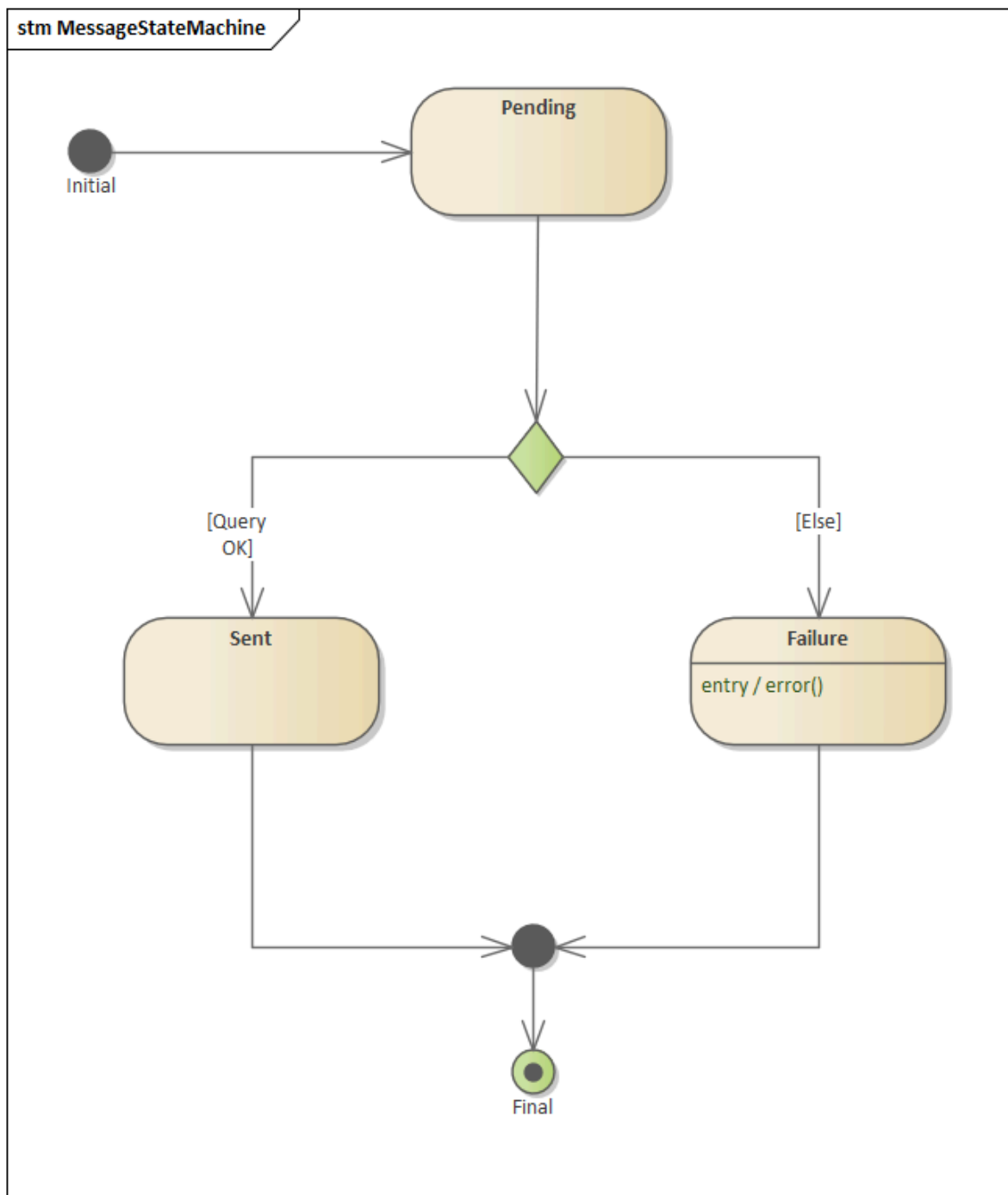


MACHINE STATES DIAGRAM:

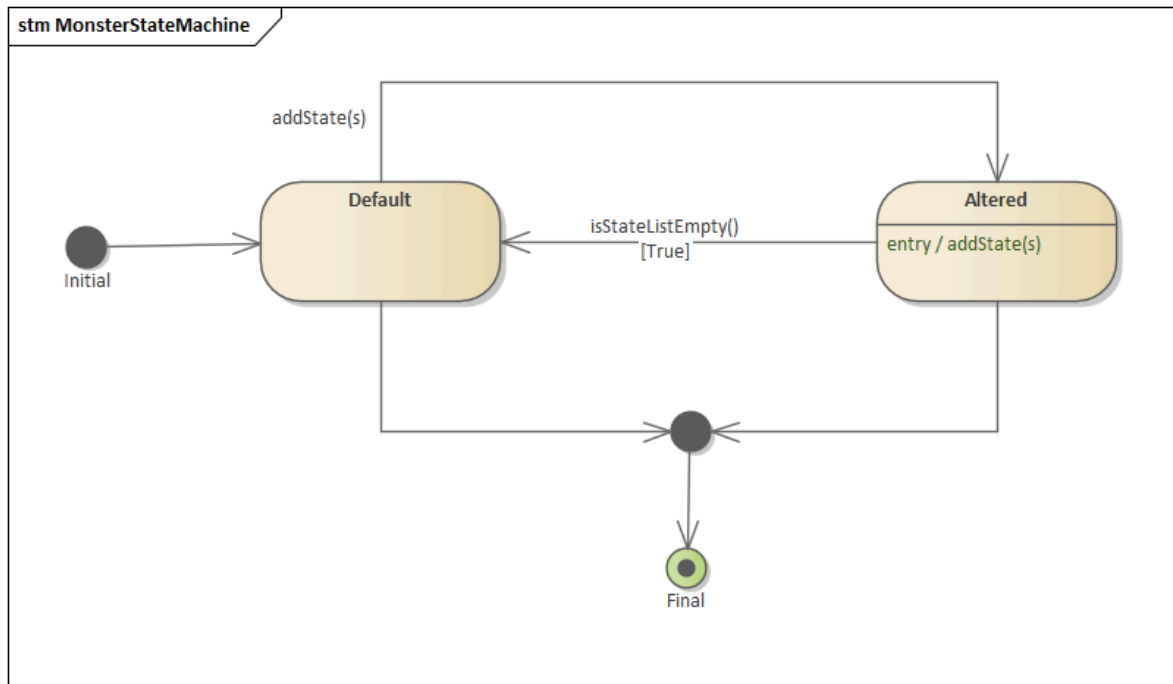
Adventure Machine State Diagram:



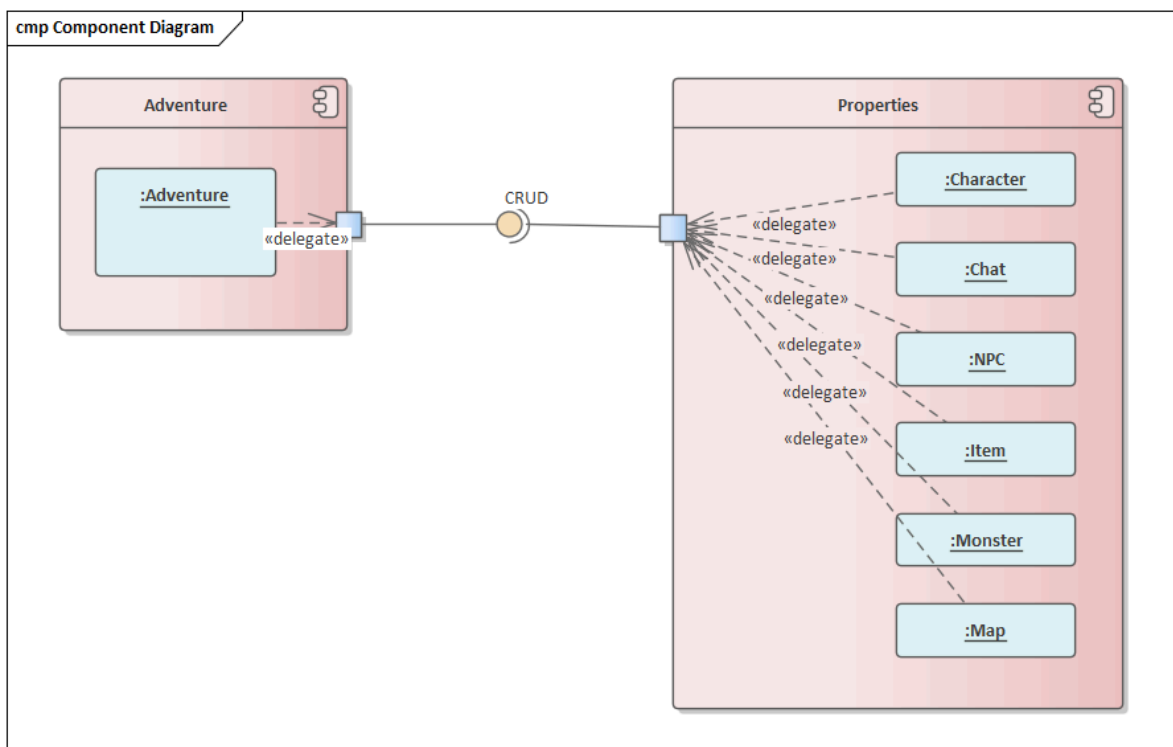
Message Machine State Diagram:



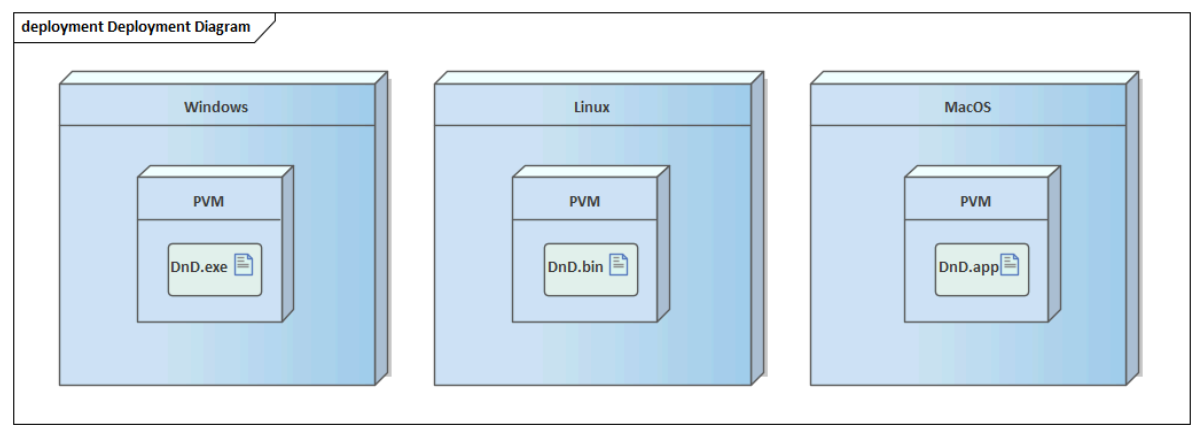
Monster Machine State Diagram:



COMPONENTS DIAGRAM:



DEPLOYMENT DIAGRAM:



MOCKUPS:

Bestiary Mockup:

D&D Manager

Main

Characters

Bestiary

NPCs

Items

Maps

Monster 1 name

Monster 2 name

Monster 3 name

Propic

Name

Alignment

Size

AC

Speed

Hp

Str

Dex

Con

Int

Wis

Cha

Attacks

Equipment

Skills

Extras

Delete

Save

Characters Mockup:

D&D Manager

Main

Characters

Settings

NPCs

Items

Maps

Character 1 name

Character 2 name

Character 3 name

Propic

Name

Alignment

Size

Class

Level

Race

Player Name

ACSpeedHpMaxHp

StrDexConIntWisCha

Attacks

Equipment

Skills

Extras

Delete

Save

Create Adventure Mockup:

Create Adventure

Create Adventure

Adventure Name:

Create

Back

Create Character Mockup:

Create Character

Name

Alignment

Size

Class

Level

Race

AC

Speed

Max/hp

Str

Dex

Con

Int

Wis

Cha

Attacks

Equipment

Skills

Extras

Return to Menu

Next

Items Mockup:

D&D Manager

Main

Characters

Bestiary

NPCs

Items

Maps

Item 1 name

Item 2 name

Item 3 name

Propic

Name

Description

Potion

Spell

Armor

Weapon

Effect

Casting Time

Casting Type

Damage Type

Damage Dice

CA

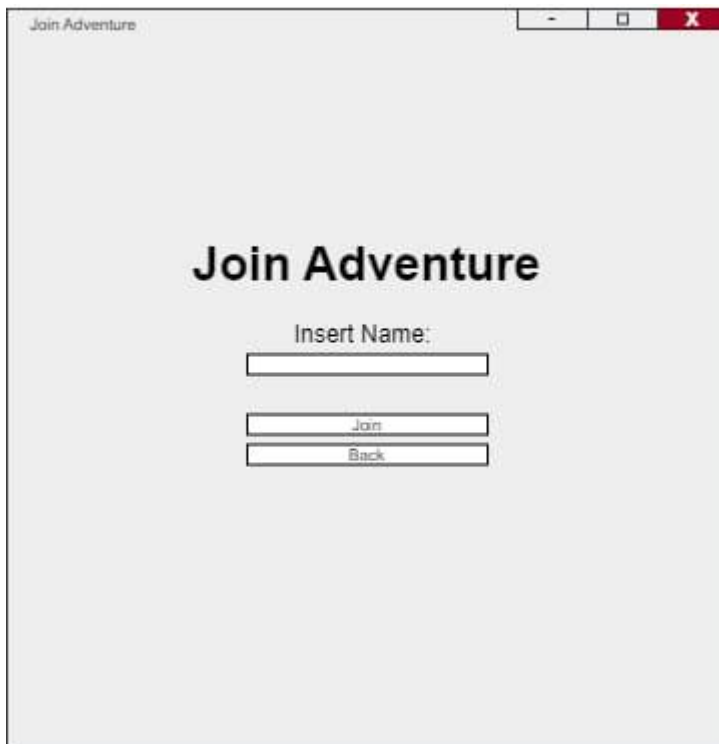
Extras

Resistances

Delete

Save

Join Adventure Mockup:



Join Adventure

Join Adventure

Insert Name:

Login Mockup:



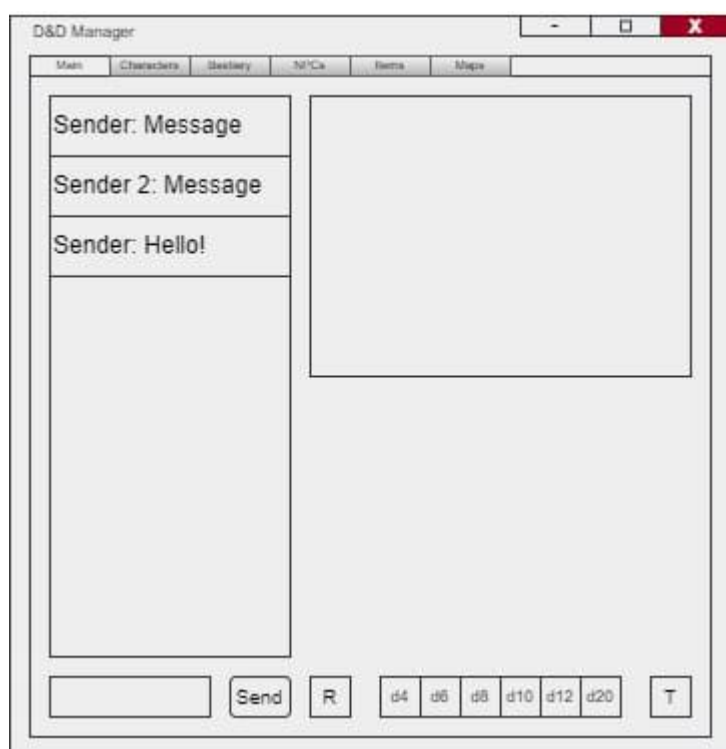
Login

D&D

Username:

Password:

Main Playable Page Mockup:



Maps Mockup:

D&D Manager

Main

Characters

Bestiary

NPCs

Items

Maps

Map 1 name

Map 2 name

Map 3 name

Name

Width

Height

Weather

Delete

Save

Menu Mockup:



NPC Creation Mockup:

D&D Manager

Main

Characters

Bestiary

NPCs

Items

Maps

NPC 1 name

NPC 2 name

NPC 3 name

Propic

Name

Alignment

Size

ACSpeedHp

StrDexConIntWisCha

Attacks

Equipment

Skills

Extras

Delete

Save

Registration Page Mockup:

Register

D&D

Username:

Password:

Confirm Password:

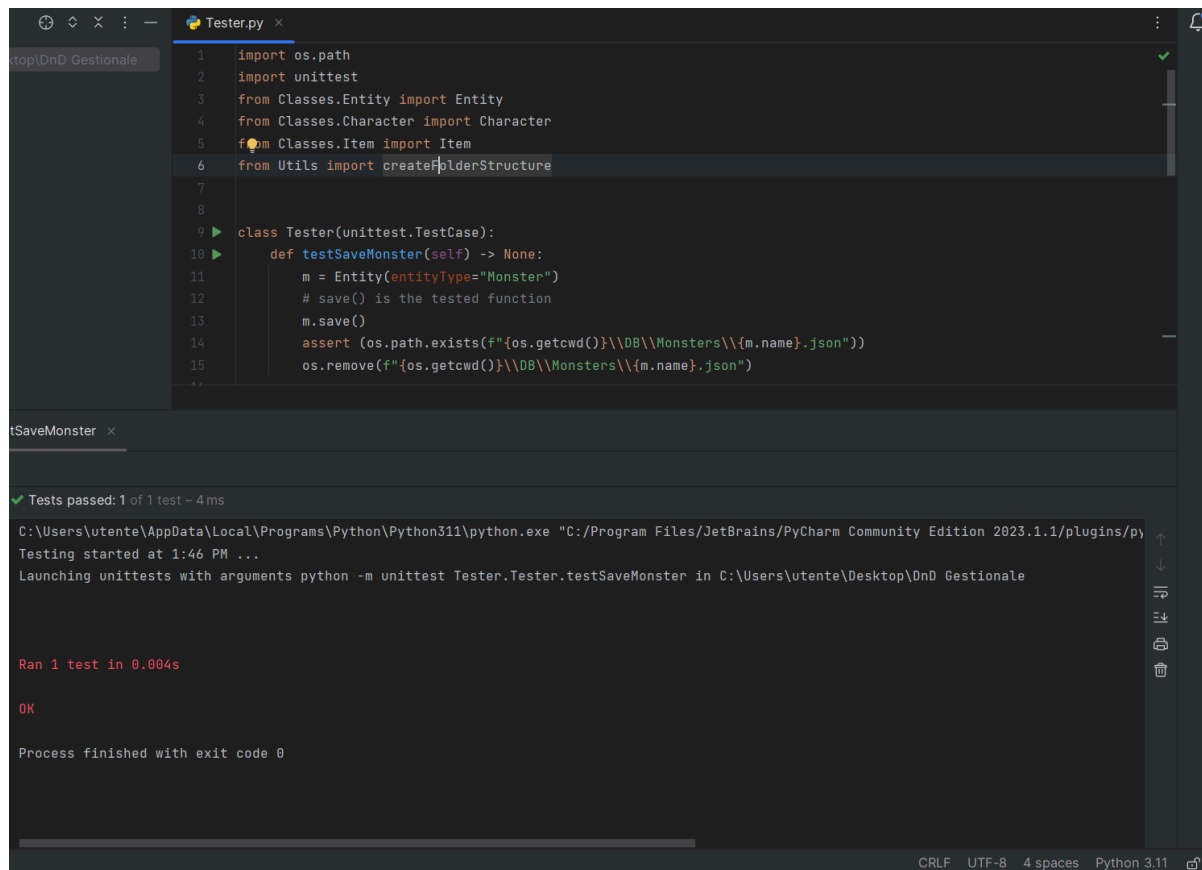
[Already Registered? Click here](#)

UNIT TESTS DESCRIPTION:

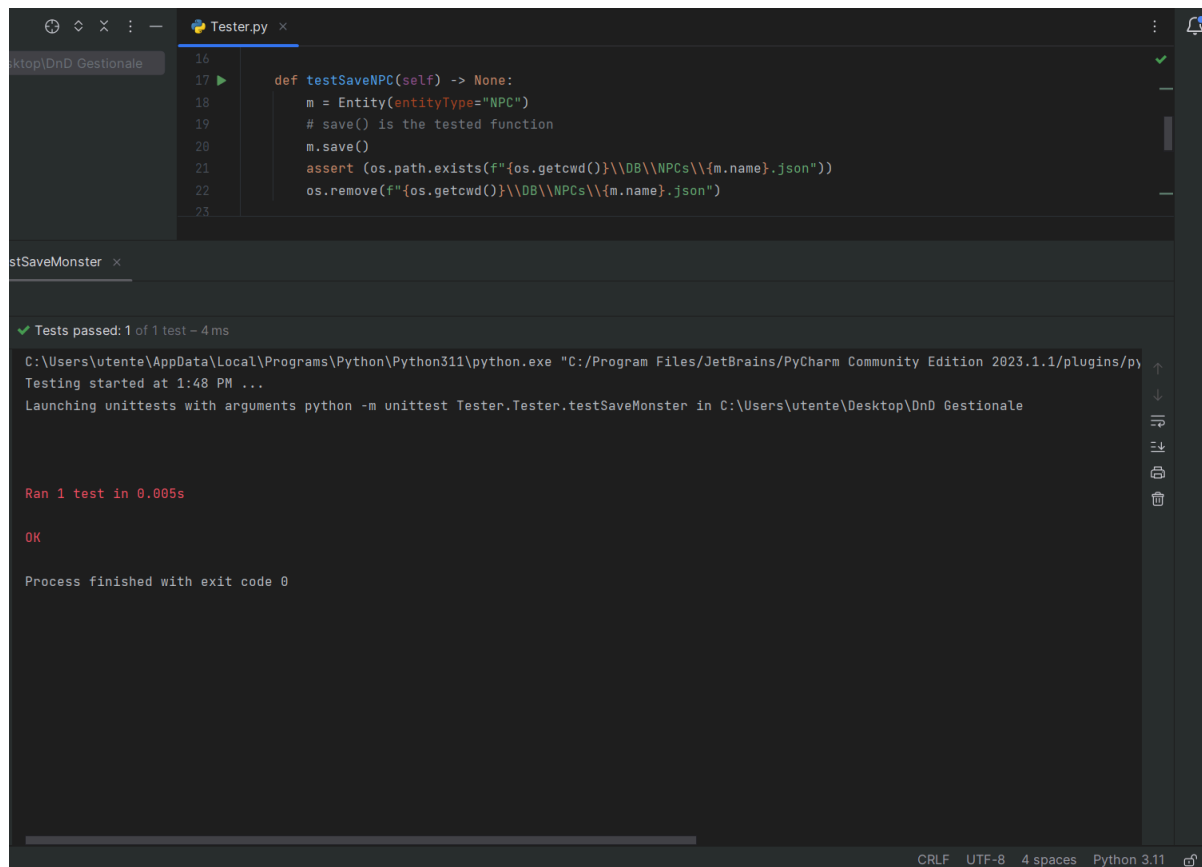
The Unit Tests made have tested the correctness of some Software functionalities critical for the right maintenance and usage of the Database. After a new entity, Item or character is created eventually it needs to be saved in the Database and to do so the functions “saveEntity()”, “saveItem()” and “saveCharacter()” have been implemented and used. These functions have successfully been tested through the usage of the unittest module and in the following section these Test cases will be explained.

testSaveMonster() TEST:

This test takes an entity type Monster and tests the save function to make sure the Monster Entity has been saved in the Database. After completing the test the created monster is successfully removed from the Database.



The same process has been repeated with an entity type NPC:



The screenshot shows the PyCharm IDE interface. The top pane displays a Python file named `Tester.py` with the following code:

```
16
17
18     def testSaveNPC(self) -> None:
19         m = Entity(entityType="NPC")
20         # save() is the tested function
21         m.save()
22         assert (os.path.exists(f"{os.getcwd()}\\DB\\NPCs\\{m.name}.json"))
23         os.remove(f"{os.getcwd()}\\DB\\NPCs\\{m.name}.json")
24
```

The bottom pane shows the test execution results:

```
✓ Tests passed: 1 of 1 test - 4 ms
C:\Users\utente\AppData\Local\Programs\Python\Python311\python.exe "C:/Program Files/JetBrains/PyCharm Community Edition 2023.1.1/plugins/py
Testing started at 1:48 PM ...
Launching unittests with arguments python -m unittest Tester.Tester.testSaveMonster in C:\Users\utente\Desktop\DnD Gestionale

Ran 1 test in 0.005s

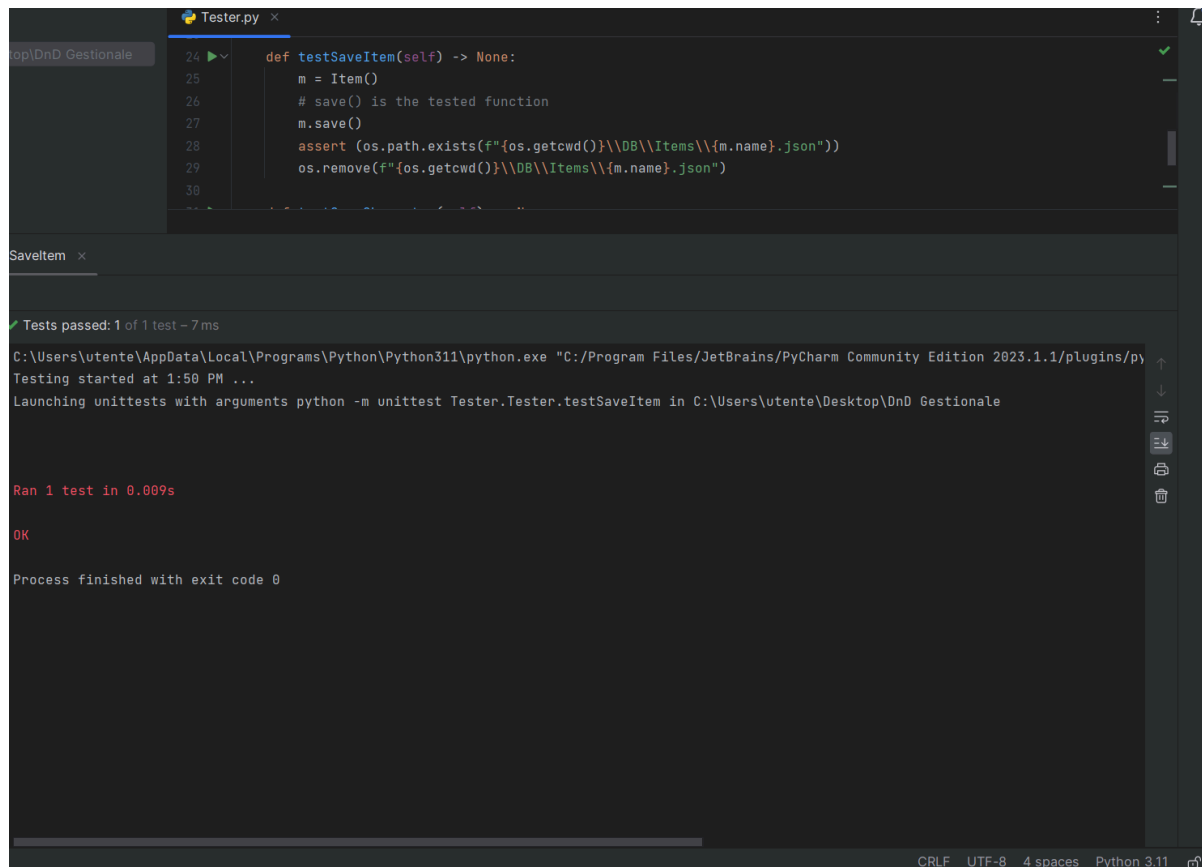
OK

Process finished with exit code 0
```

The status bar at the bottom indicates the file encoding is CRLF, UTF-8, with 4 spaces, using Python 3.11.

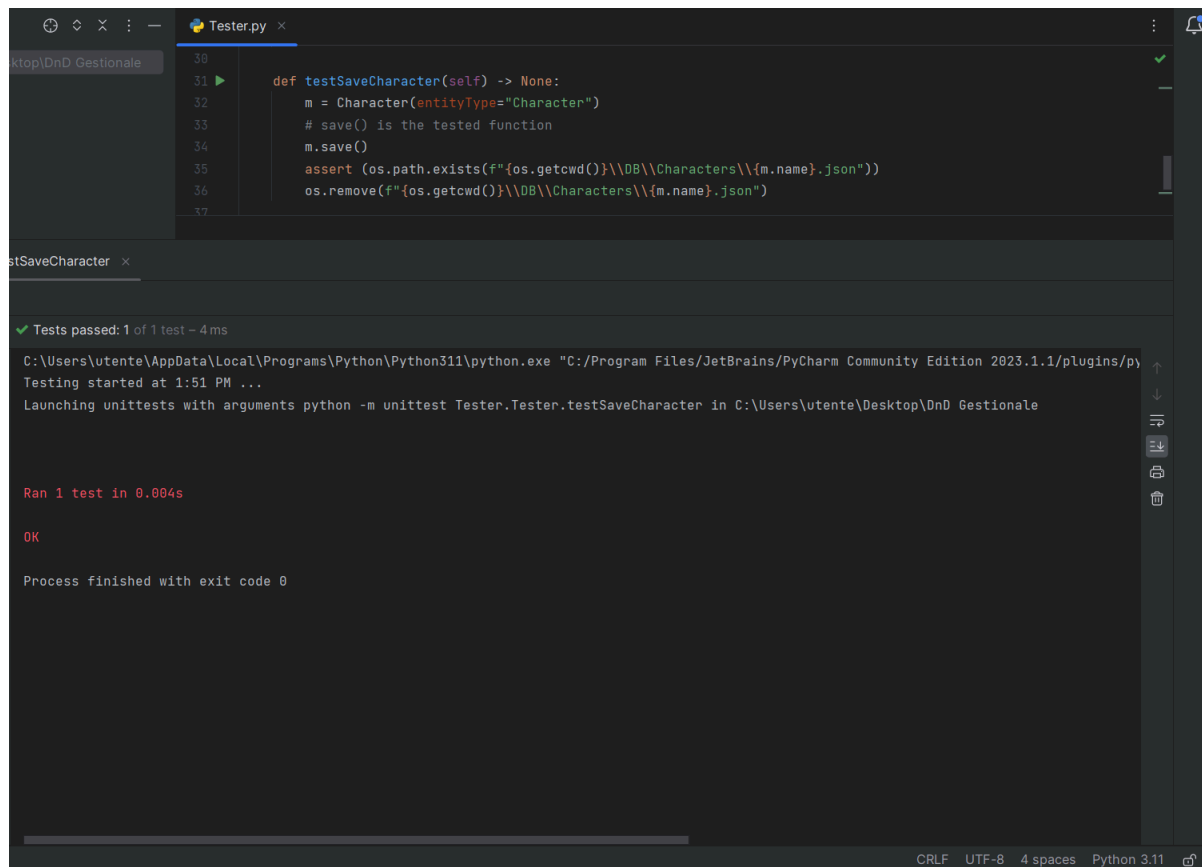
testSaveItem() TEST:

The saving process has also been tested with the Items, making sure these were correctly saved in the Database.



testSaveCharacter() TEST:

Lastly the save function has been tested when used on the Character Entities.



The screenshot shows the PyCharm IDE interface. The top pane displays a Python file named `Tester.py` with the following code:

```
38  
31 ▶ def testSaveCharacter(self) -> None:  
32     m = Character(entityType="Character")  
33     # save() is the tested function  
34     m.save()  
35     assert os.path.exists(f"{os.getcwd()}\\DB\\Characters\\{m.name}.json")  
36     os.remove(f"{os.getcwd()}\\DB\\Characters\\{m.name}.json")  
37
```

The bottom pane shows the test execution results:

```
✓ Tests passed: 1 of 1 test - 4 ms  
C:\Users\utente\AppData\Local\Programs\Python\Python311\python.exe "C:/Program Files/JetBrains/PyCharm Community Edition 2023.1.1/plugins/py  
Testing started at 1:51 PM ...  
Launching unittests with arguments python -m unittest Tester.Tester.testSaveCharacter in C:\Users\utente\Desktop\DnD Gestionale  
  
Ran 1 test in 0.004s  
  
OK  
  
Process finished with exit code 0
```

The status bar at the bottom indicates the file encoding is UTF-8, uses 4 spaces for indentation, and is running on Python 3.11.

We also Tested a function whose main prerogative in ensuring the correctness of the overall folder system.

testFolderStructure() TEST:

This test makes sure the folder structure of the System is correctly functioning and able to support all the other operations taking place during the course of the Adventure. All folders must behave in a coherent way so that every folder is able to contain the requested Data it was made to contain whenever a new object is created.

This test also runs in the main code to make sure the game itself behaves correctly thanks to all the folders performing their tasks correctly.

