Analysis document

Thomas van der Molen
IPS3-DB03

Project Information	
Project members	Thomas van der Molen (4168003)
Project name	Text Adventure
Version	1.5

Table of Contents

Version History

Version	Date	Change
1.0	01-09-2021	Created file and made: Introduction, requirements, conceptual model & UI Concepts.
1.1	02-09-2021	Added more requirements and made Requirements into MoSCoW, added use cases.
1.2	15-09-2021	Added Game Content.
1.3	16-09-2021	Added ERD.
1.4	17-09-2021	Added User stories.
1.5	11-01-2022	Added Class diagram and new versions of UI and ERD

Introduction

In this application you can play a text adventure game that will be procedurally generated while playing. You will also be able to save your last played adventure to resume later. Furthermore, there will be a leaderboard to show finished adventures with the highest achieved scores.

Requirements

Requirements (FR-/NFR-) for the application including possible restrictions (R-) and qualitative demands (Q-).

Functional requirements

Must Have

FR-01 The user can start an adventure.

FR-02 The user can register an account.

R-02.1 The user needs to enter a unique username and email.

Q-02.1 The username can’t contain any offensive words.

FR-03 The user can see a leaderboard of adventurers.

Should Have

FR-04 The user can change his password.

R-04.1 The user must fill in correct current password.

FR-05 The user can choose to not show their adventures on the leaderboards.

FR-06 The user can save and resume an adventure.

R-06.1 The user needs to be logged in.

FR-07 The user can see his own adventurers.

FR-08 The user can find an overview of all useable commands.

FR-09 The user can see information on the current adventure.

Could Have

FR-10 The user can access all saved adventures and user information.

R-10.1 The user must be an admin.

Q-10.1 The user cannot see passwords.

FR-11 The user can see a map of the explored dungeon.

FR-12 The user can post comments on the leaderboards.

R-12.1 The user needs to be logged in.

FR-13 The user can request an automated password reset.

FR-14 The user can play with other users in one dungeon.

Won’t Have

Non-functional requirements

NFR-01 The user will get a feedback notification when sending data.

NFR-02 Actions submitted in games will give feedback that it has been executed.

NFR-03 The game will have fast response times.
    Q-03.1 The response delay cannot be longer than 500 milliseconds

User stories

US-01 As a user I can easily create an account and login.

US-02 As a user I can clearly see a leaderboard of the top users.

US-03 As a user I can easily start or resume an adventure.

US-04 As a user I can clearly see all useable commands during the game.

US-05 As a user I can easily see and use any items I have.

US-06 As a user I have a map showing all rooms I have discovered.

US-07 As a user I can explore rooms and interact with object in the room.

US-08 As a user I can engage in combat against enemies.

US-09 As a user I can quit an adventure and resume at any time.

US-10 As a user I want to have a unique experience from other users.

US-11 As a user I can customize my account.

Use Cases

Name:	**UC01: Starting new adventure**
Summary	A user can start a new adventure
Actors	User
Assumptions	Actor is on the home page
Scenario	1. Actor wants to start a new adventure 2. Actor presses new game button 3. System creates a new game 4. System redirects actor to game page
Exceptions	1. User has a saved adventure. (Q-01.1) Show message warning user that this action will delete the saved adventure, if confirmed go to 3. else bring user to homepage
Result	A new adventure has been created for user.

Name:	**UC02: Registering account**
Summary	A user can register an account
Actors	User
Assumptions	Actor is on the home page
Scenario	1. Actor wants to register a new account 2. Actor chooses the register account option 3. System shows form with username, email and password 4. Actor fills in information 5. Actor clicks on create account 6. System creates the account
Exceptions	1. Actor does not fill in all fields properly. (R-02.1) Show message telling user what field was not filled in properly, go to 3 2. Actor fills in username or email that is already in use. (Q-02.1) Show message telling user that the filled in information is already in use, go to 3
Result	A new account has been created.

Name:	**UC03: Viewing leaderboards**
Summary	User wants to view leaderboards
Actors	User
Assumptions	Actor is on the home page
Scenario	1. Actor wants to view leaderboards 2. Actor presses the leaderboards button 3. System get's all viewable finished runs and orders them based on score 4. System redirects actor to leaderboard
Exceptions	none
Result	Actor is shown an up-to-date leaderboard.

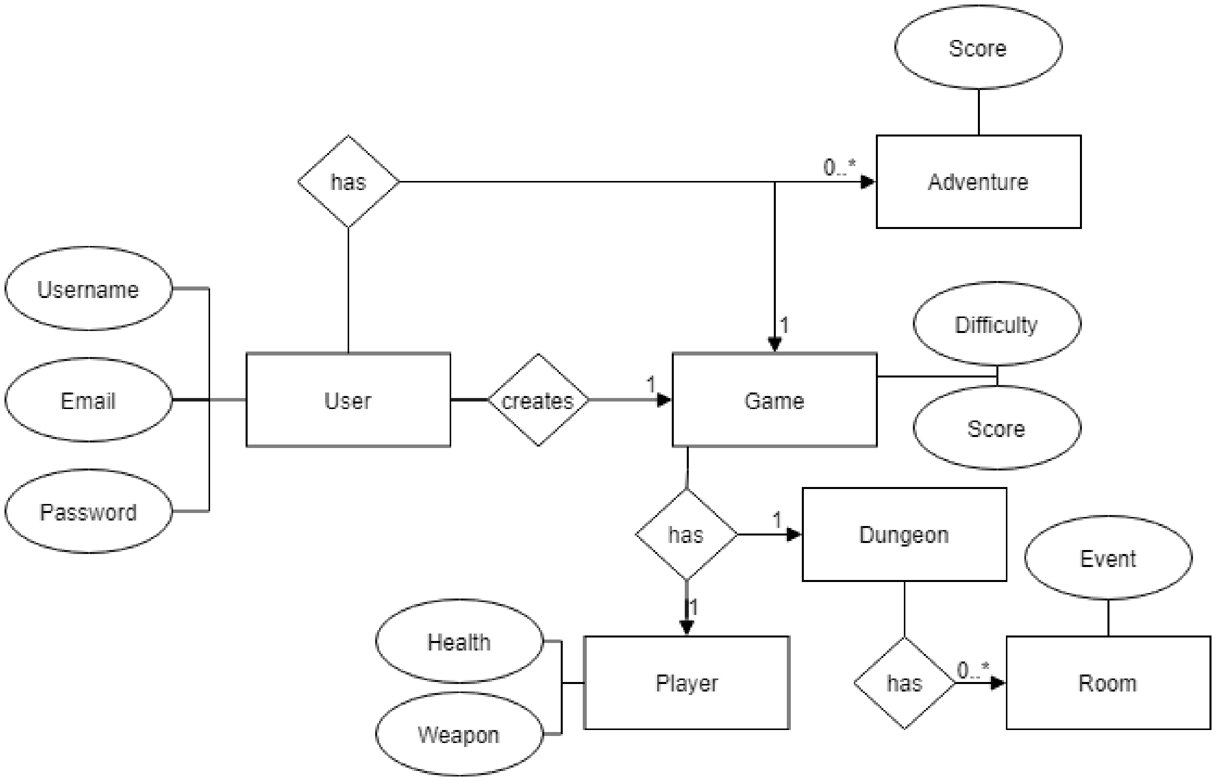
Name:	**UC04: User turns off adventure visibility**
Summary	User wants it’s finished adventures not to show up on the leaderboards.
Actors	User
Assumptions	Actor is on the home page and is logged in
Scenario	1. Actor wants to change visibility of finished adventures 2. Actor presses the account button 3. System redirects actor to account page 4. Actor unchecks the show adventures on leaderboards option 5. System will change actor’s account to not show adventures on leaderboards
Exceptions	None
Result	Actor’s adventures will no longer show up on the leaderboards.

Name:	**UC05: User changes password**
Summary	User wants to change password
Actors	User
Assumptions	Actor is on the home page and is logged in
Scenario	1. Actor wants to change password 2. Actor presses the account button 3. System redirects actor to account page 4. Actor clicks on change password button 5. System shows input fields with current password and new password 6. Actor fills in the input fields 7. Actor clicks on change password button 8. System checks if fields are valid 9. System changes actor’s password
Exceptions	1. Actor fills in the wrong current password. (R-04.1) Show actor that filled in password is incorrect go to 5 2. Actor fills in the same current and new password. Show actor that filled in passwords are the same go to 5
Result	Actor’s password is changed.

Name:	**UC06: User wants to save adventure**
Summary	User would like to save current adventure to resume later
Actors	User
Assumptions	Actor is playing an adventure
Scenario	1. Actor wants to save adventure 2. Actor presses the save adventure button 3. System saves all current adventure data 4. System shows message saying adventure saved
Exceptions	1. Actor is not logged in. Show player he has to be logged in to save adventures
Result	Actor’s current adventure has been saved

Name:	**UC07: User wants to resume adventure**
Summary	User would like to resume a previously saved adventure.
Actors	User
Assumptions	Actor is on the home page
Scenario	1. Actor wants to resume adventure 2. Actor logs in 3. System checks if actor has a saved adventure 4. System shows user a resume adventure button 5. Actor clicks resume adventure button 6. System loads saved adventure to game 7. System redirects actor to the game's page
Exceptions	1. User does not have a saved adventure. System does not show the resume adventure button.
Result	Actor's saved adventure has been resumed.

Conceptual Model



In my application is the user a person, that can create and play a game. The user can also store a game to resume. Every user can have many finished runs also called adventures, these adventures are stored and can be viewed and sorted by score.

A user has a username, email, password and can have many adventures and a game.

An adventure has a score.

A game has a difficulty, score and includes a player and dungeon.

A player has health and a weapon.

A dungeon has many rooms.

A room has an event.

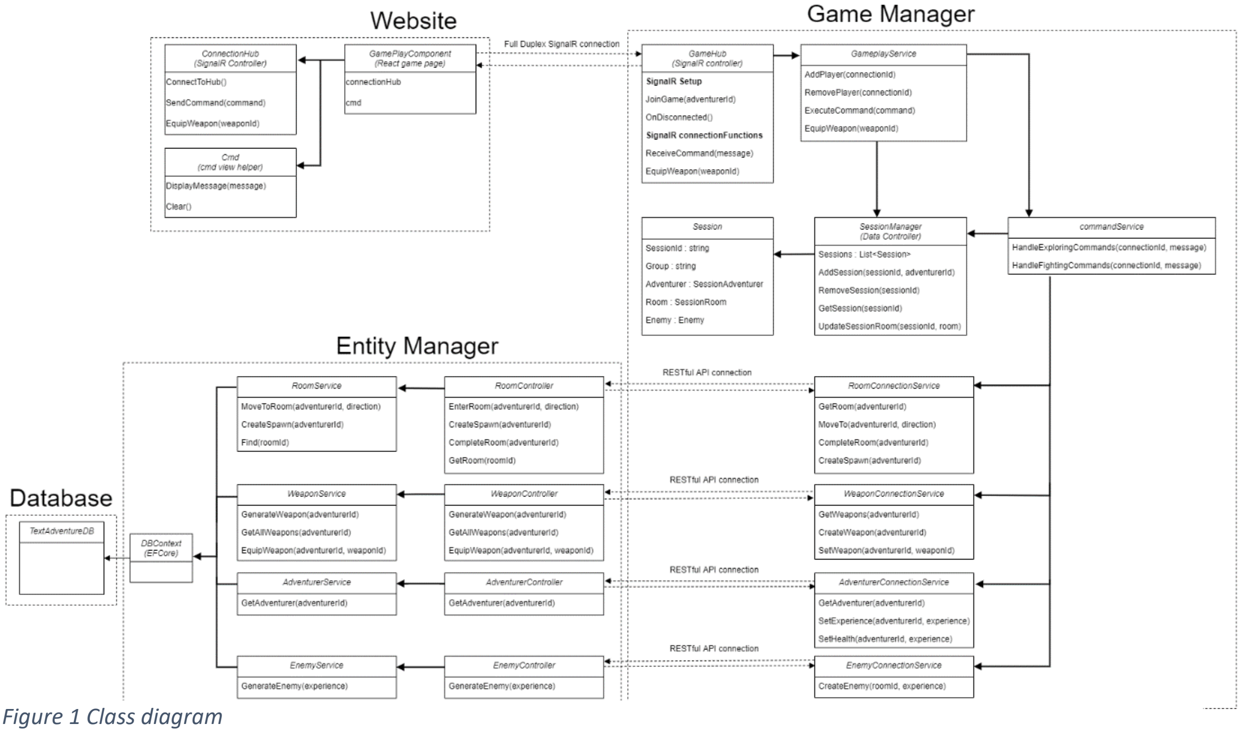
Game Content

The game will contain many different mechanics these mechanics have been split up in individual requirements. This is to both keep in mind the scope of the project and the progress of the game's functionality.

1. The game will have a player who has health, damage and experience.
2. The player will have an inventory with items such as potions, keys and weapons.
3. The weapons will have a durability check making them break over time.
4. The game spawn the player in a massive dungeon that all players traverses.
5. The player will appear in a random in the dungeon that has not been explored yet.
6. Every room will have 1 special event that can happen when entering the room like an ambush, chest, trap
7. The game will not overlap rooms.
8. Every room will have a north, east, south and west side. These sides can all contain 1 special occurrence such as a door, person, item.
9. Every player will have a map of what they have explored.
10. Game events will be based on the player's level.

Class Diagram

During the semester I started to notice that my project was getting very messy and not OOP/SOLID anymore, for this reason I decided to setup a class diagram of what my project should be structured like. I used this diagram afterwards to restructure the entire project.



Website
ConnectionHub
(SignalR Controller)
ConnectToHub()
SendCommand(command)
EquipWeapon(weaponId)
GamePlayComponent
(React game page)
connectionHub
cmd
Cmd
(cmd view helper)
DisplayMessage(message)
Clear()
Full Duplex SignalR connection
Game Manager
GameHub
(SignalR controller)
SignalR Setup
JoinGame(adventurerId)
OnDisconnected()
SignalR connectionFunctions
ReceiveCommand(message)
EquipWeapon(weaponId)
GameplayService
AddPlayer(connectionId)
RemovePlayer(connectionId)
ExecuteCommand(command)
EquipWeapon(weaponId)
Session
SessionId : string
Group : string
Adventurer : SessionAdventurer
Room : SessionRoom
Enemy : Enemy
SessionManager
(Data Controller)
Sessions : List<Session>
AddSession(sessionId, adventurerId)
RemoveSession(sessionId)
GetSession(sessionId)
UpdateSessionRoom(sessionId, room)
commandService
HandleExploringCommands(connectionId, message)
HandleFightingCommands(connectionId, message)
Entity Manager
RoomService
MoveToRoom(adventurerId, direction)
CreateSpawn(adventurerId)
Find(roomId)
RoomController
EnterRoom(adventurerId, direction)
CreateSpawn(adventurerId)
CompleteRoom(adventurerId)
GetRoom(roomId)
RESTful API connection
RoomConnectionService
GetRoom(adventurerId)
MoveTo(adventurerId, direction)
CompleteRoom(adventurerId)
CreateSpawn(adventurerId)
Database
TextAdventureDB
DBContext
(EFCore)
WeaponService
GenerateWeapon(adventurerId)
GetAllWeapons(adventurerId)
EquipWeapon(adventurerId, weaponId)
WeaponController
GenerateWeapon(adventurerId)
GetAllWeapons(adventurerId)
EquipWeapon(adventurerId, weaponId)
RESTful API connection
WeaponConnectionService
GetWeapons(adventurerId)
CreateWeapon(adventurerId)
SetWeapon(adventurerId, weaponId)
AdventurerService
GetAdventurer(adventurerId)
AdventurerController
GetAdventurer(adventurerId)
RESTful API connection
AdventurerConnectionService
GetAdventurer(adventurerId)
SetExperience(adventurerId, experience)
SetHealth(adventurerId, experience)
EnemyService
GenerateEnemy(experience)
EnemyController
GenerateEnemy(experience)
RESTful API connection
EnemyConnectionService
CreateEnemy(roomId, experience)

Figure 1 Class diagram

UI Concepts

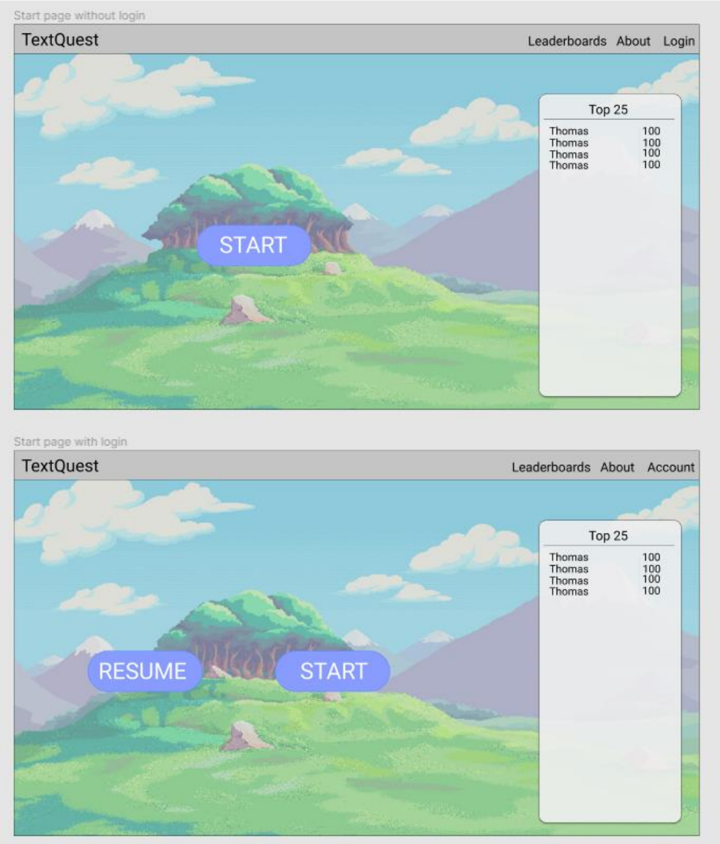
Version 1

The homepage will give the user easy access to starting or resuming a new run depending on if they are logged in. The homepage will also include a navigation bar at the top giving links to the leaderboard, an about section and login.

The homepage will also include a short list of the top players on the leaderboard.

On the game page there will be a command-line style window, in this window all dialogue will show up. Under the command-line window will be a text input box, the user will type the commands they want to execute in here.

On the right side of the screen there will be helpful information. This includes a button to give the user an overview of all commands and a save button. There is also an information box showing useful data like the player's health or its inventory.



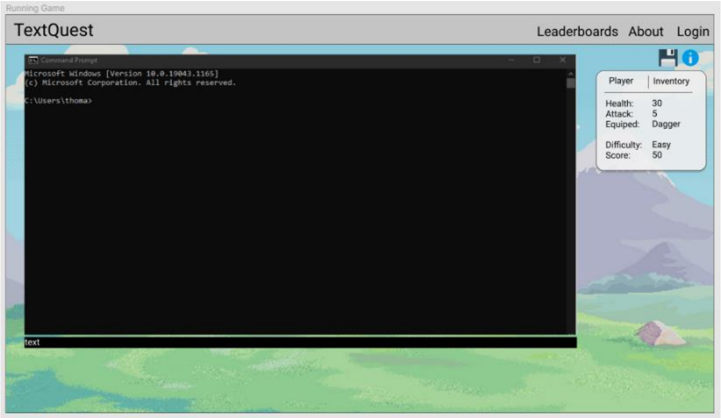
Top 25

Thomas	100
Thomas	100
Thomas	100
Thomas	100

Top 25

Thomas	100
Thomas	100
Thomas	100
Thomas	100

Figure 2 homepage



```
Microsoft Windows [Version 10.0.19043.1165]
(c) Microsoft Corporation. All rights reserved.

C:\Users\thoma>
```

Health: 30
Attack: 5
Equiped: Dagger

Difficulty: Easy
Score: 50

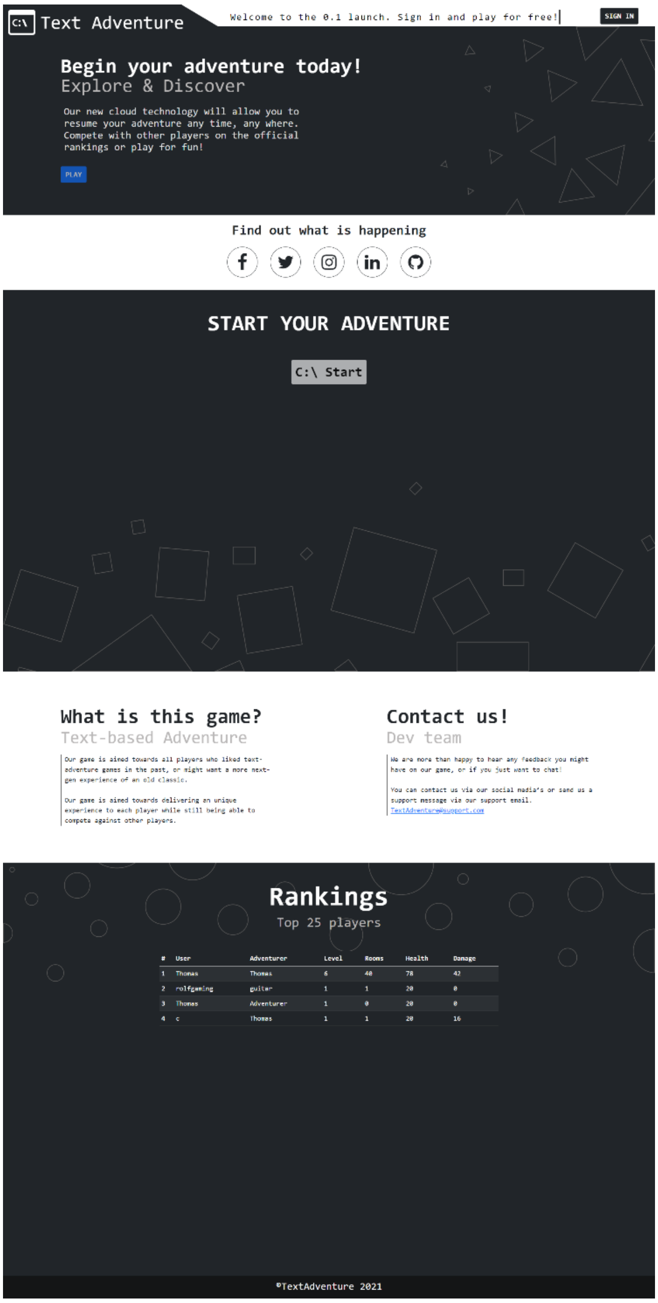
Figure 3 Game

Version 2

After spending a lot of time working on the UI/UX and getting feedback from users, I have decided to redesign the website and go in a different direction. Feedback and advise given by people about my project has given me a better idea of what people expect from a modern-day text adventure game.

For my redesign I have chosen to go for a more simplistic and darker toned look, this will tie the whole website in with the common designs of command line systems.

The pages have also been condensed. There is now a homepage with all the needed information and a page for the game itself.



Welcome to the 0.1 launch. Sign in and play for free!

Begin your adventure today!
Explore & Discover

Our new cloud technology will allow you to resume your adventure any time, any where. Compete with other players on the official rankings or play for fun!

Find out what is happening

START YOUR ADVENTURE

What is this game?
Text-based Adventure

Our game is aimed towards all players who liked text-adventure games in the past, or might want a more next-gen experience of an old classic.

Our game is aimed towards delivering an unique experience to each player while still being able to compete against other players.

Contact us!
Dev team

We are more than happy to hear any feedback you might have on our game, or if you just want to chat!

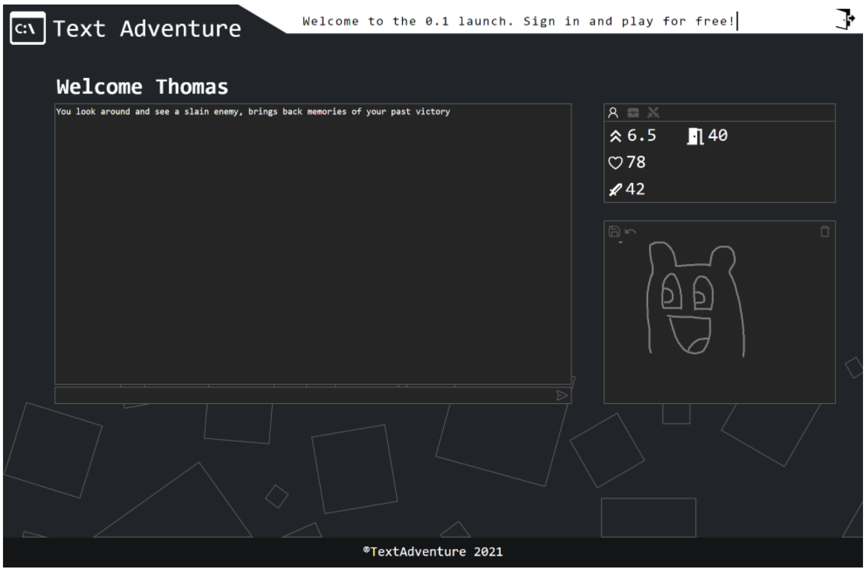
You can contact us via our social media's or send us a support message via our support email.
TextAdventure@support.com

Rankings
Top 25 players

#	User	Adventurer	Level	Rooms	Health	Damage
1	Thomas	Thomas	6	40	78	42
2	rolfgaming	guitar	1	1	20	0
3	Thomas	Adventurer	1	0	20	0
4	c	Thomas	1	1	20	16

©TextAdventure 2021

Figure 4 Home page



Welcome to the 0.1 launch. Sign in and play for free!

Welcome Thomas

You look around and see a slain enemy, brings back memories of your past victory

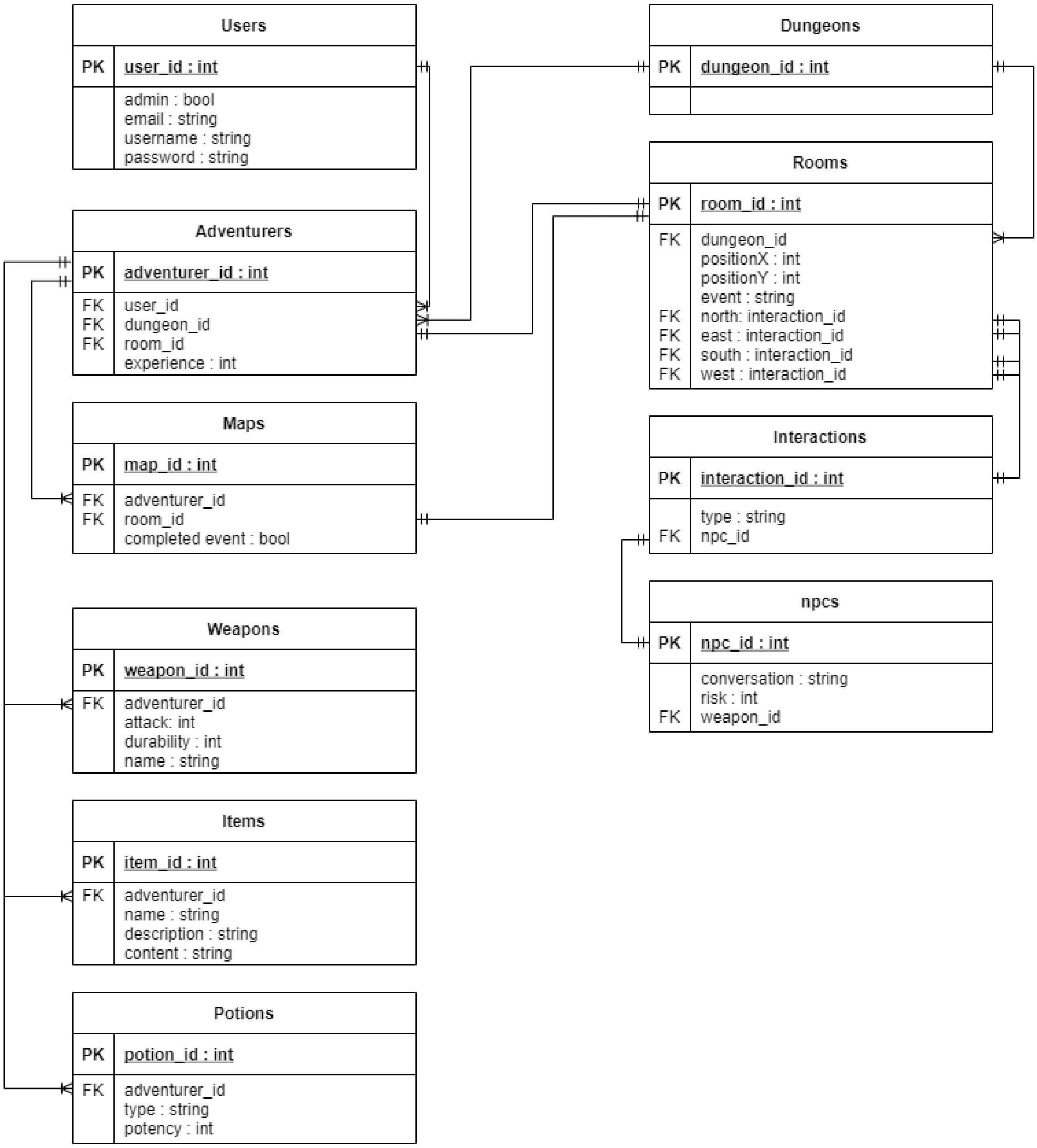
©TextAdventure 2021

Figure 5 Game page

ERD

Version 1

For this project I have also created an ERD to easily visualize the database structure.



Users	
PK	user_id : int
	admin : bool email : string username : string password : string

Adventurers	
PK	adventurer_id : int
FK FK FK	user_id dungeon_id room_id experience : int

Maps	
PK	map_id : int
FK FK	adventurer_id room_id completed event : bool

Weapons	
PK	weapon_id : int
FK	adventurer_id attack: int durability : int name : string

Items	
PK	item_id : int
FK	adventurer_id name : string description : string content : string

Potions	
PK	potion_id : int
FK	adventurer_id type : string potency : int

Dungeons	
PK	dungeon_id : int

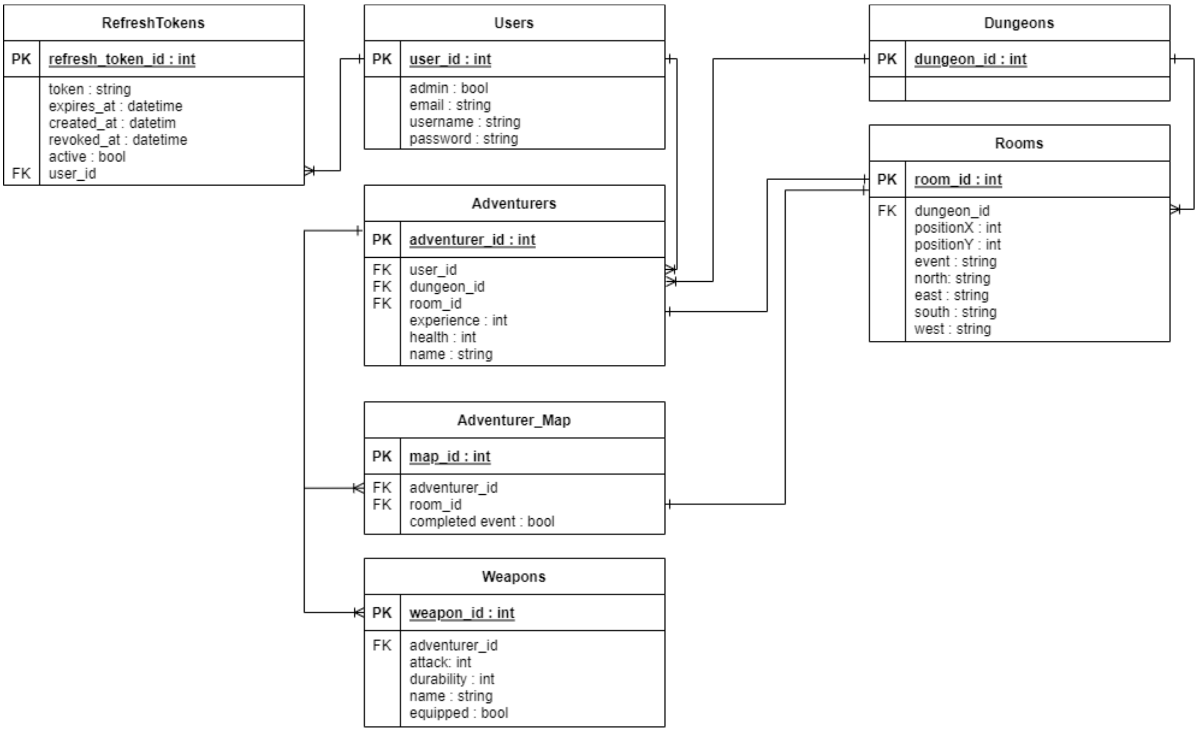
Rooms	
PK	room_id : int
FK FK FK FK FK	dungeon_id positionX : int positionY : int event : string north: interaction_id east : interaction_id south : interaction_id west : interaction_id

Interactions	
PK	interaction_id : int
 FK	type : string npc_id

npcs	
PK	npc_id : int
  FK	conversation : string risk : int weapon_id

Version 2

While working on the project I realized that the scope of my game was going to be too large to finish in just one semester, for this reason I decided to simplify my game and focus on the base game loop and thus I have simplified my current ERD while still allowing for expansion in the future.


RefreshTokens
PK refresh_token_id : int
token : string
expires_at : datetime
created_at : datetim
revoked_at : datetime
active : bool
FK user_id
Users
PK user_id : int
admin : bool
email : string
username : string
password : string
Dungeons
PK dungeon_id : int
Rooms
PK room_id : int
FK dungeon_id
positionX : int
positionY : int
event : string
north: string
east : string
south : string
west : string
Adventurers
PK adventurer_id : int
FK user_id
FK dungeon_id
FK room_id
experience : int
health : int
name : string
Adventurer_Map
PK map_id : int
FK adventurer_id
FK room_id
completed event : bool
Weapons
PK weapon_id : int
FK adventurer_id
attack: int
durability : int
name : string
equipped : bool