

DAT602 – ASSESSMENT TWO

PROJECT

Milestone Three

Table of Contents

Milestone One.....	5
General Game Overview.....	6
Login.....	6
Quest Selection.....	6
Gameplay or Questing.....	6
Administration.....	7
Storyboarding.....	8
Storyboard One - Login.....	8
Storyboard Two – Delete Account.....	9
Storyboard Three – Incorrect Password.....	9
Storyboard Four – Username Incorrect.....	10
Storyboard Five – User Registration.....	10
Storyboard Six – Game Selection.....	11
Storyboard Seven – New Quest.....	12
Storyboard Eight – Game Play.....	13
Storyboard Nine – Game Play Notifications.....	14
Storyboard Ten – Administration Portal.....	15
Storyboard Eleven – User Administration.....	16
Screen Design Rationale.....	17
Login.....	17
Quest Selection.....	17
Game Play or Questing.....	17
Administration.....	17
CRUD Table.....	18
CRUD Table Analysis.....	19
User Registers.....	19
User Login.....	19
Delete Account.....	19
Account Locked.....	19
Quest Selection.....	20
New Quest.....	20
User Moves in a Quest.....	20
User Chat.....	20
Quest Ends.....	20
User Logout.....	20

Administration Portal.....	21
Kill in Progress Quest	21
Delete User	21
Create User	21
Modify User.....	21
Entity Relationship Diagram	22
Entity Relationship Diagram Rationale	23
Players and their Quest's	23
Quest and Quest Map.....	23
Quest Map, Map Tiles, and Assets.....	23
Quest Score and Inventory	23
Quest Chat	23
SQL	24
tblUser.....	24
tblQuest	24
tblChat.....	25
tblQuestPlay.....	25
tblQuestScore	25
tblAsset	26
tblQuestPlayInventory	26
tblQuestMap	26
tblQuestMapTile	26
tblQuestTileAsset.....	27
Milestone Two	28
Entity Relationship Diagram – Iteration Two.....	29
CRUD Table – Iteration Two.....	30
ERD and CRUD Iteration Two Discussion	31
CURD Activities MySQL Procedures	31
User Registration	31
User Login	31
Delete Account.....	31
Lock Account	32
Join Quest	32
New Quest	32
User Moves	32
User Leaves Quest.....	32

Quest Ends	32
User Logout.....	32
Kill In Progress Quest	32
Delete User	32
Create User	32
Modify User.....	32
ACID and Multi-Player Game Support	33
Atomicity	33
Consistency	33
Isolation	33
Durability.....	34
References	35

Milestone One

General Game Overview

The game is Wizard Quest. Players take turns moving around the map; as they do so, more of the world will be uncovered. The main aim is to collect gold to gain points. The map has several other items for players to collect and store in their inventory to assist in their quest. Players beware! Your quest is full of danger at every turn with traps that could lose you points or even end your quest! Players can set up an account, play, and communicate with other players in real-time.

Additionally, players will be able to save their games, delete their accounts, and administrators will be able to assist players with account and game issues.

Login

The game will initially begin with a login window. Users can enter their credentials:

- Existing users with credentials matching those stored on the database will proceed to the Wizard Quest, quest selection window, or they can delete their account.
- If the user is existing, but there is a credential mismatch, the user will be prompted to re-enter their password. The user has five attempts to enter the correct password before their account being locked and requiring administrator intervention.
- If credentials are not on the database, the user is asked if they would like to register. The user is prompted for their email address and password.

Quest Selection

The Quest Selection window appears after a user has successfully logged in. Users can see currently online players, a list of Quests (games) in play, and their saved quests from past interactions. The user can:

- Join a Quest that is in progress with other user(s).
- Re-join a saved Quest.
- Begin a new Quest.
- Logout.
- If the user is an administrator, they will be able to access the administration portal.

Gameplay or Questing

Once the user has made their selection and begins a quest (game), they are taken to the Questing window, where the gameplay commences. The player begins on a tile, and only a tiny amount of the map is uncovered. As they move around, more of the map will become visible, showing:

- **Gold** – the main aim of the game is to collect the gold and score points. Points will be added to the players game score, and they will be able to view this in the quest along with the score of any other player they are playing. The player that has the highest quest score after the quest ends wins that game.
- **Chances** – These will be some type of question mark tile graphic which will award the player some kind of item for their inventory. Examples include a spell gem or a wand with additional points. Or some detracting object is encountered like a cursed dragon's egg or a magic dead area which the player then loses points. This is the risk of taking the chance.

- **Traps** – Traps are unmarked and in a small allocation of tiles in the game. If a player chooses a tile where a trap has been set, the player can either:
 - Lose some points or
 - Lose all their points scored in that quest and die.

As players move around the map, more of the world is uncovered, and they can collect the gold they see. The quest ends once all the gold is collected or all players have died. Players do not have to choose a chance tile if they do not want to take a risk. However, chance tiles can be beneficial as the potential points could secure that player's quest victory.

While playing the game, users can chat via the chat console. Entering a message into the type to chat field and clicking the send button will send their message to other users in the same game.

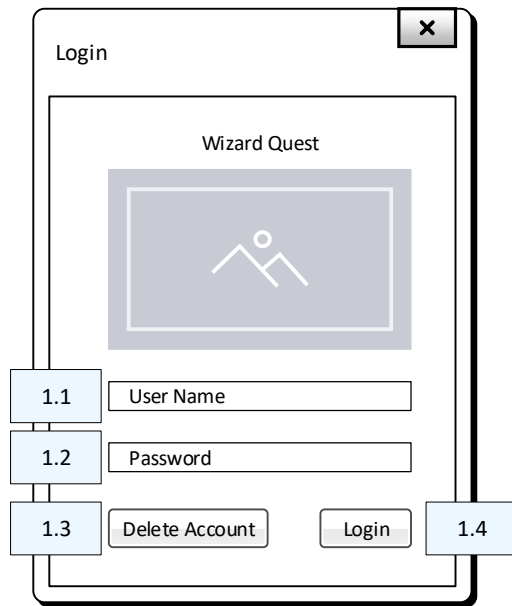
Administration

Users that have administrative credentials will have access to the Wizard Quest Administration Portal. Administrators can:

- End quests, add, modify, and delete users.
- They also can unlock accounts, grant administrative privileges, and adjusting user scores.

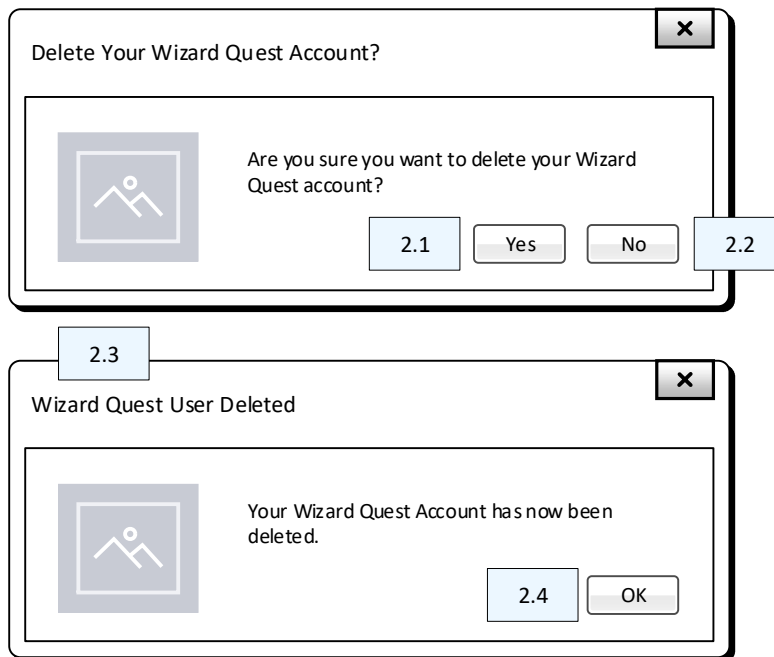
Storyboarding

Storyboard One - Login



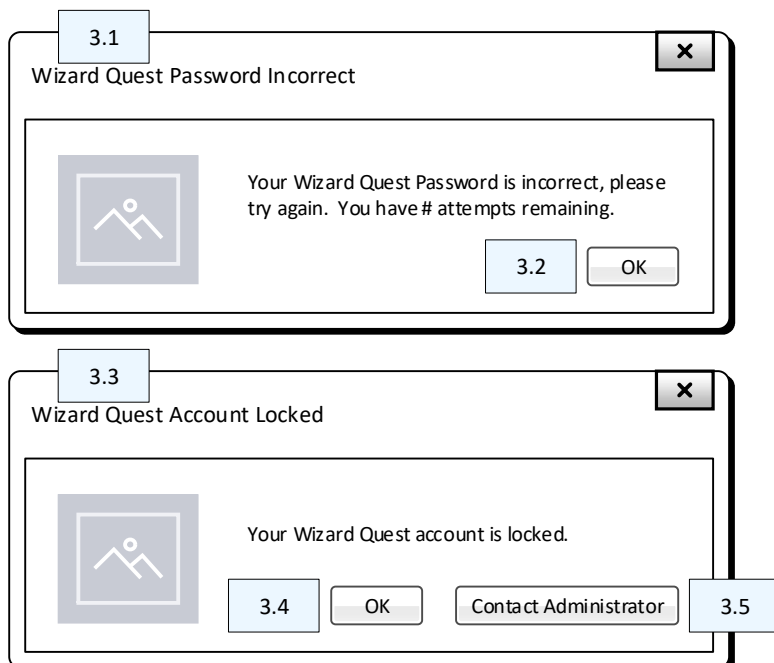
- 1.1 Textbox for username.
- 1.2 Textbox for password.
- 1.3 Delete Account button, if clicked and username and password are correct, go to storyboard two if password incorrect, go to storyboard three.
- 1.4 Login Button, if clicked and username and password are correct, go to storyboard six. If the username is valid and the password incorrect, go to storyboard three. If the username is not found, go to storyboard four.

Storyboard Two – Delete Account



- 2.1 Yes button confirms account deletion and displays (2.3) confirmation dialogue box.
- 2.2 No button returns the user to login. Go to storyboard one.
- 2.3 Account Delete confirmation dialogue box.
- 2.4 OK button. Returns to log in, go to storyboard one.

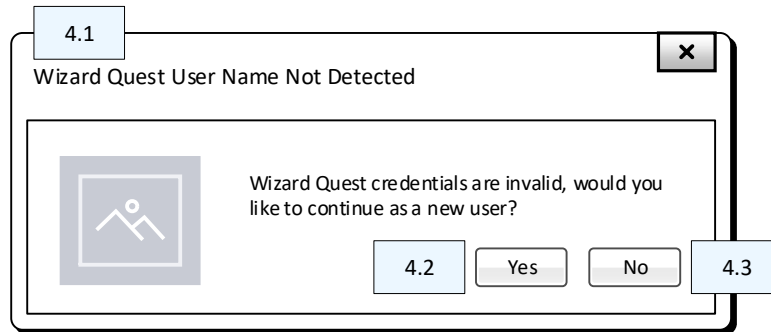
Storyboard Three – Incorrect Password



- 3.1 Incorrect password dialogue box displayed. Informs the user of how many attempts they have before their account is locked.
- 3.2 OK button returns the user to login. Go to storyboard one.

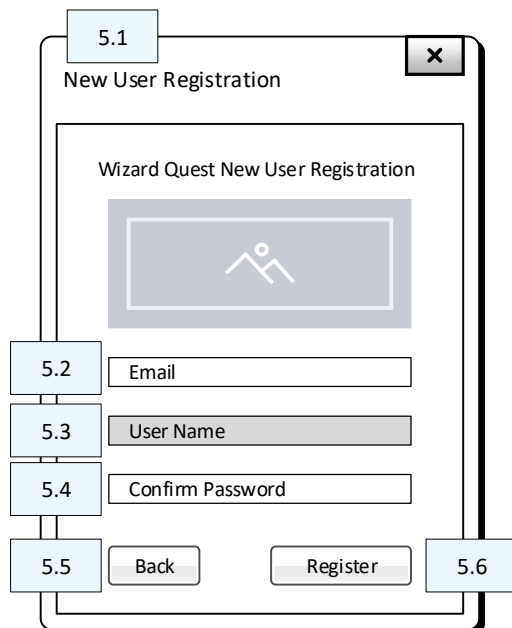
- 3.3 Account locked dialogue box displayed after five unsuccessful password attempts.
- 3.4 OK button returns the user to login. Go to storyboard one.
- 3.5 Gives user administrator email to contact for account reactivation assistance.

Storyboard Four – Username Incorrect



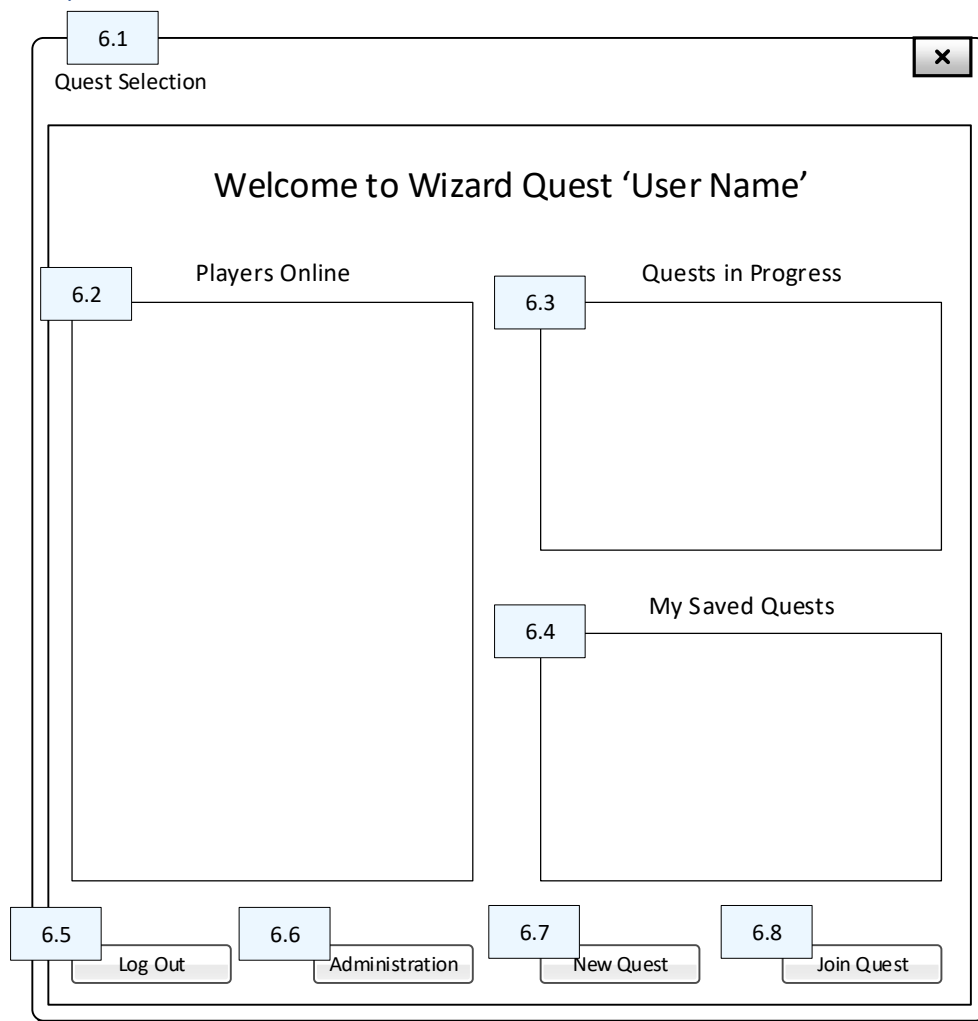
- 4.1 Username not detected dialogue box displayed.
- 4.2 Yes button, if clicked, the user can register. Go to storyboard five.
- 4.3 No button, if clicked, the user is returned to login. Go to storyboard one.

Storyboard Five – User Registration



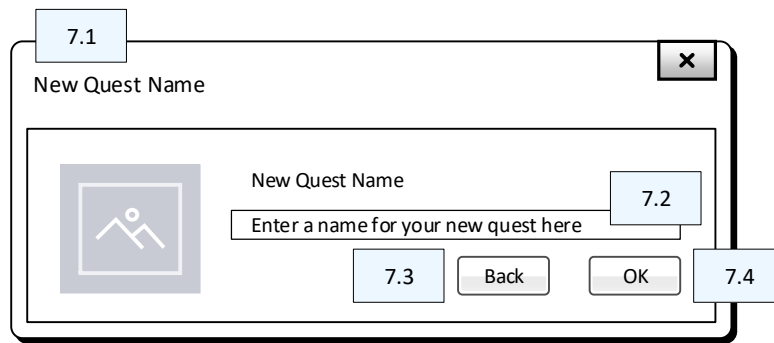
- 5.1 New user registration dialogue box displayed.
- 5.2 Textbox for the user to enter an email address.
- 5.3 Username textbox populates with username from the login screen.
- 5.4 Password textbox for user password.
- 5.5 Back button, if clicked, returns to login. Go to storyboard one.
- 5.6 Register button if clicked go to storyboard six.

Storyboard Six – Game Selection



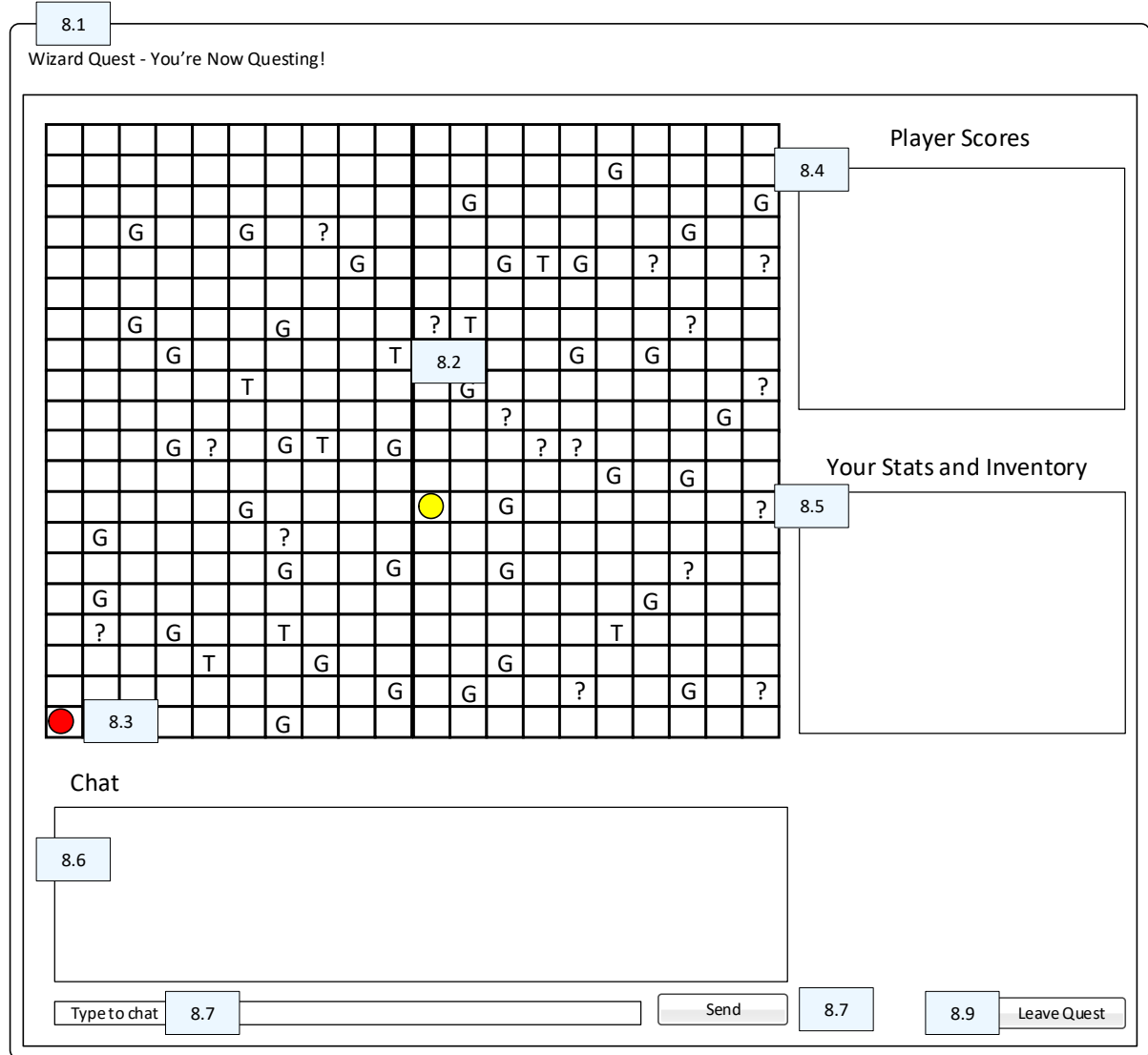
- 6.1 Quest selection dialogue box displayed.
- 6.2 Lists all players online and their scores, listed in high score order.
- 6.3 Lists all quests currently being played by other online players.
- 6.4 List all quest the user has previously started and are now saved.
- 6.5 Log out button if clicked returns to login. Go to storyboard one.
- 6.6 Administration button only visible If the user has administrative privileges. If clicked displays the administration portal. Go to storyboard ten.
- 6.7 New Quest button, if clicked, starts a new game. Go to storyboard seven.
- 6.8 Join Quest button clickable only when a quest is in progress of saved quest has been selected from lists 6.3 or 6.4 respectively. Go to storyboard eight.

Storyboard Seven – New Quest



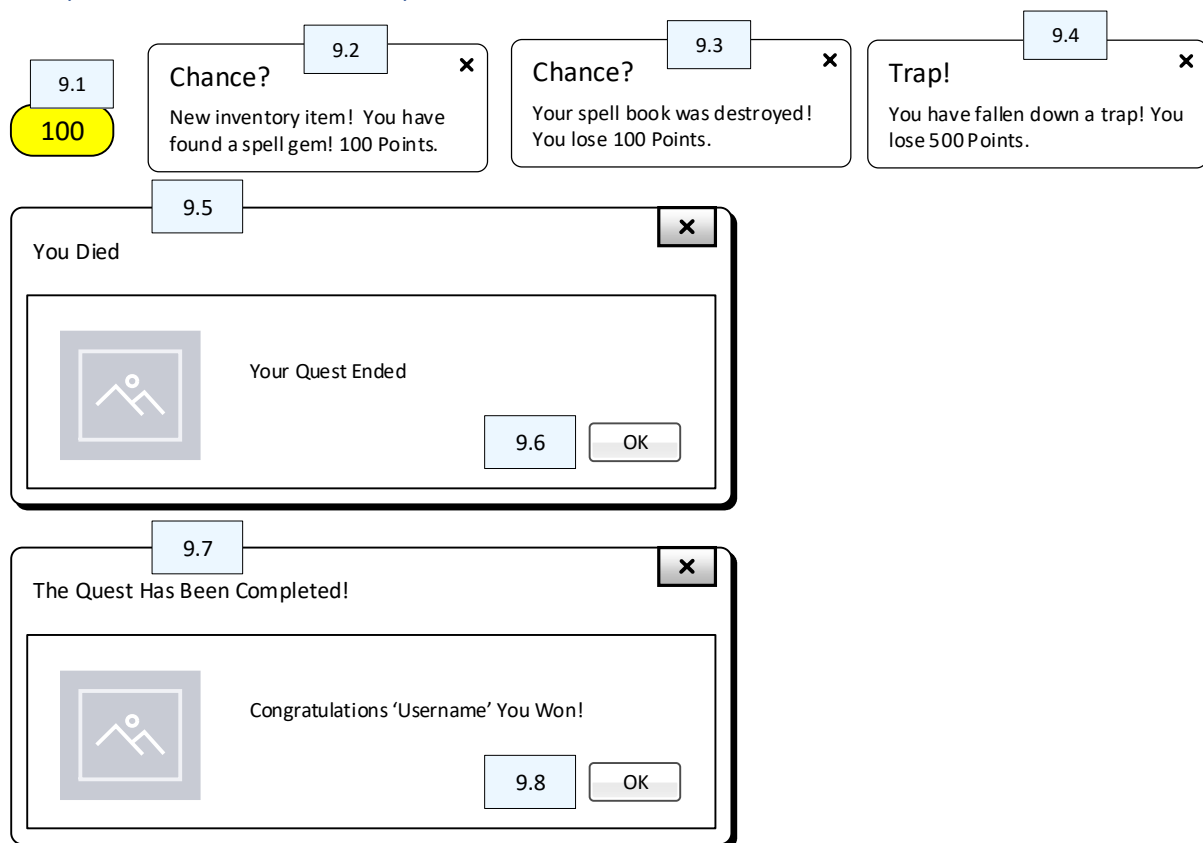
- 7.1 New Quest dialogue box displayed.
- 7.2 Textbox for the user to enter a name for their new quest.
- 7.3 Back button, if clicked, returns to Quest Selection. Go to storyboard six.
- 7.4 OK button if clicked starts gameplay. Go to storyboard eight.

Storyboard Eight – Game Play



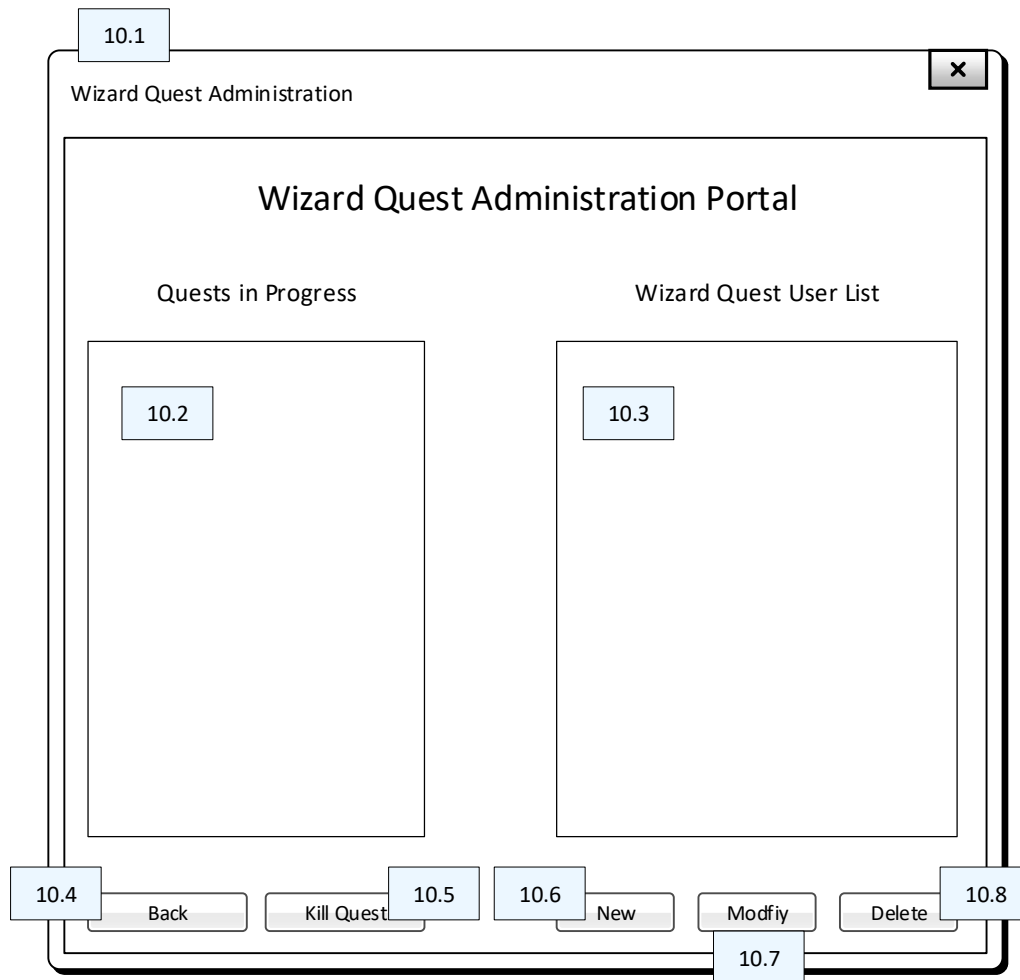
- 8.1 Game dialogue box displays quest.
- 8.2 Tilemap shows the world as the player moves around the map tiles with gold (G) give points, tile with chances (?) can provide inventory or cost player points, and traps (T) aren't shown, but if the tile is selected by the user, they can lose points or die. All notifications will appear as pop-up notifications and the final win/lose notification as a dialogue box. Go to storyboard nine.
- 8.3 Counters shows players current tile location.
- 8.4 Each player's score is listed in the game in ascending score order.
- 8.5 Show user's complete stats and inventory collected.
- 8.6 Chat window displays all messages sent in the game.
- 8.7 Chat textbox where the user can enter a new chat message.
- 8.8 Send button, the message entered will be seen by all players in the chat window (8.6).
- 8.9 Leave Quest button. If clicked, the user returns to Quest Selection, where this game will now appear on the user's My Saved Quests list. Go to storyboard six.

Storyboard Nine – Game Play Notifications



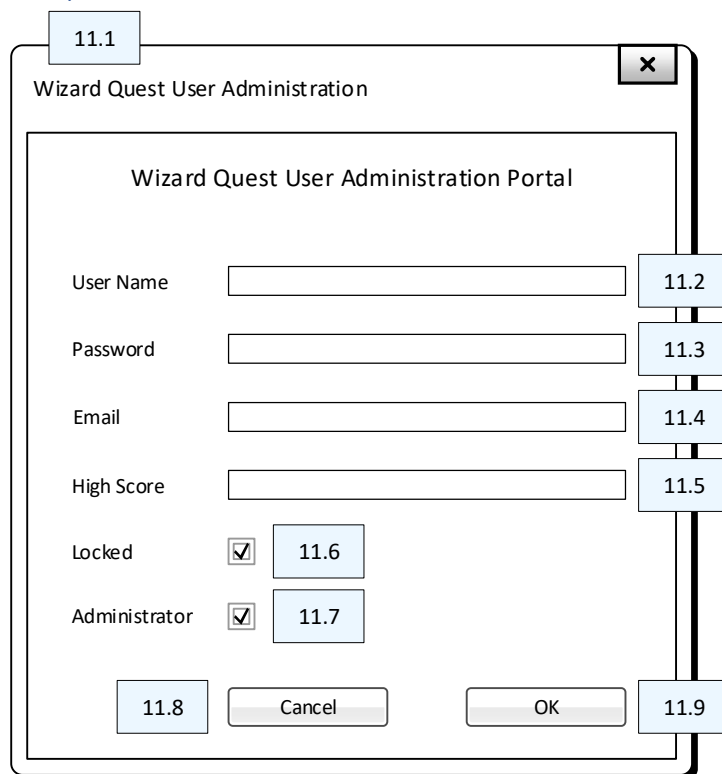
- 9.1 Bubble appears above player tile when gold is present, and player scores points.
- 9.2 Bubble appears when a chance tile is selected, inventory is acquired, and points are scored.
- 9.3 Bubble appears when a chance tile is selected and a player loses points.
- 9.4 Bubble appears when trap tile is selected, and a player loses points.
- 9.5 You Died dialogue box appears when the player dies during a quest by either trap or losing their points.
- 9.6 OK button returns the user to Quest Selection. Go to storyboard six.
- 9.7 Completed Quest dialogue box appears when all the gold has been found on the quest map, and the user with the most points acquired during the game wins.
- 9.8 OK button returns the user to Quest Selection. Go to storyboard six.

Storyboard Ten – Administration Portal



- 10.1 Administration Portal dialogue box.
- 10.2 List quests (games) currently in progress.
- 10.3 List all users.
- 10.4 Back button in clicked returns the user to Quest Selection. Go to storyboard six.
- 10.5 Kill Quest button kills selected quest from the Quests in Progress list (10.2).
- 10.6 New button, if clicked, opens a create user dialogue. Go to storyboard eleven.
- 10.7 Modify button if clicked opens modify user administration dialogue. Go to storyboard eleven.
- 10.8 Delete button if clicked deleted selected user from Wizard Quest User List (10.3).

Storyboard Eleven – User Administration



The image shows a storyboard for a 'Wizard Quest User Administration' dialog box. The dialog box has a title bar with a close button (X) and a label 'Wizard Quest User Administration'. Inside the dialog, there is a section titled 'Wizard Quest User Administration Portal'. Below this title, there are several input fields and checkboxes, each with a storyboard label:

- User Name**: A text input field labeled 11.2.
- Password**: A text input field labeled 11.3.
- Email**: A text input field labeled 11.4.
- High Score**: A text input field labeled 11.5.
- Locked**: A checkbox labeled 11.6, which is currently checked.
- Administrator**: A checkbox labeled 11.7, which is currently checked.
- Buttons**: At the bottom, there are two buttons: 'Cancel' labeled 11.8 and 'OK' labeled 11.9.

- 11.1 User Administration dialogue box.
- 11.2 Textbox for username.
- 11.3 Textbox for password.
- 11.4 Textbox for email.
- 11.5 Textbox for the high score.
- 11.6 Checkbox for account locked (checked) and unlocked (unchecked) status.
- 11.7 Checkbox for administrative privileges.
- 11.8 Cancel button returns the user to the administration portal. Go to storyboard ten.
- 11.9 OK button adds/modifies user details and returns administrator to administration portal. Go to storyboard ten.

Screen Design Rationale

Login

A simple login screen built to the brief and designed to reflect good UX and UI specifications. Using object-oriented principles, player registration could inherit from.

Quest Selection

The Quest Selection window acts as a collection point for all the user's information, gives them an overview of current online game and score activity, and is a central hub to interact with games in progress or start a new game. The design is easy to follow and understand while displaying the required deliverables.

Game Play or Questing

The questing game screen is a basic grid formation that follows map game layout conventions to assist the user with an intuitive and easy-to-use interface. Using 20 x 20 tiles at this stage in the design gives 400 tiles for players to move around on during their quest. Questing is achieved on the map, with player score overview and inventory information appearing on the right-hand side.

Additionally, users can chat with others playing the same game by utilising the chat terminal at the bottom of the quest window.

Administration

Administrators' requirements include ending running quests, adding new users, updating user details. The administration portal design meets these requirements with a simple to use dual window game and user management interface.

As with login, the modify and add user window is reused to align with object-oriented design principles.

CRUD Table

Entity/Attribute	User Registers	User Login	Delete Account	Lock Account	Quest Selection	New Quest	User Moves	User Chat	Quest Ends	User Logout	Administration Portal	Kill in Progress Quest	Delete User	Create User	Modify User
tblUser	C	R	D		R	R		R	RU	RU	R		D	RC	RU
UserID	C		D		R	R		R			R		D	C	R
UserName	C	R	D		R			R	R		R		D	RC	RU
UserPassword	C	R	D										D	C	RU
Email	C		D										D	C	RU
LoginAttempts	C	RU	D										D	C	RU
UserLocked	C	R	D	U									D	C	RU
UserOnline	C		D		RU					U			D	C	RU
Administrator	C		D		R								D	C	RU
TotalScore	C		D		R				RU	RU		RU	D	C	RU
tblQuest					R	C	U	R	U	U	R	U			
QuestID					R	C		R		U	R				
QuestName					R	C					R				
MaxUsers						C									
ActivePlayer						C	U			U					
QuestStatus									U	U	R	U			
tblQuestPlay			D		R	C		R		U	R		D		
QuestPlayID			D		R	C		R		U	R		D		
UserID			D		R	C					R		D		
TileID			D		R	C	U			U			D		
ChatID			D										D		
QuestID			D							U			D		
PlayerNumber			D			C				U			D		
tblQuestPlayInventory			D			C									
InventoryID			D			C									
QuestPlayID			D			C									
AssetID			D			C									
Quantity			D			C									
tblQuestScore			D		R	C						R	D		
ScoreID			D		R	C							D		
CurrentQuestScore			D		R	C	U		R	R		R	D		
QuestPlayID			D		R	C							D		
QuestID			D		R	C							D		
tblChat			D					C					D		
ChatID			D					C					D		
UserID			D					C					D		
Message			D					C					D		
tblQuestMap						C	R								
MapID						C									
QuestID						C									
xMax						C									
yMax						C									

Entity/Attribute	User Registers	User Login	Delete Account	Lock Account	Quest Selection	New Quest	User Moves	User Chat	Quest Ends	User Logout	Administration Portal	Kill in Progress Quest	Delete User	Create User	Modify User
tblQuestMapTile						C	R								
TileID						C	RU								
MapID						C	R								
xPosition						C	R								
yPosition						C	R								
TileActive						C	RU								
tblQuestTileAsset						C	RU								
TileAssetID						C	R								
TileID						C	R								
AssetID						C	R								
tblAsset						R	R								
AssetID						R	R								
AssetName						R	R								
AssetDescription						R	R								
AssetValue						R	R								

CRUD Table Analysis

User Registers

A user registering in the system would trigger an insert statement populating entries in the tblUser table.

User Login

A user logging in would require retrieving their username, password, and an account locked check. Successful logins would require an update to login attempts to be reset to zero.

Delete Account

Executed either by user or administrator would delete all records from tblUser and cascade to any referential child keys from tblQuestPlay, tblQuestPlayInventory, tblQuestScore, and tblChat.

Account Locked

With five unsuccessful attempts at password validation, the account locked attribute of tblUser would be updated to true for the selected username.

Quest Selection

Once logged in, the programme will need to retrieve the user information, other users online, games in progress and saved game data. User needs to be updated to online status (UserOnline = True) and retrieve their details UserID, UserName, Administrator, and TotalScore.

Retrieval of quests in progress where QuestStatus = True. Attributes required include QuestID, QuestName to display in the Quest Selection window.

Retrieval of QuestPlayID, UserID, TileID from tblQuestPlay and corresponding QuestScoreID from tblQuestScore will enable the quest selection window to populate online users, show a list of current high scores, and allow the user to join.

New Quest

When a user starts a new quest, they will name it, and this will be assigned to the QuestName attribute when created in the tblQuest table. This will also create dependent table entries in tblQuestPlay, tblQuestPlayInventory, tblQuestScore, tblQuestMap, tblQuestMapTile, tblQuestTileAsse, and tblAsset to initialise a new game.

User Moves in a Quest

When a user can move ActivePlayer = PlayerNumber, the TileID is updated to the new location. The CurrentQuestScore is refreshed to add any points the user may have scored by retrieving asset values for score additions from tblAsset when moving to the new tile location. tblQuestMapTile is retrieved and updated with new location information.

User Chat

Chat requires the retrieval of the players UserID and UserName, along with their quest information QuestPlayID. The user sent a chat that created a new chat record showing the username and their message to all current players.

Quest Ends

When a quest ends, the QuestStatus in tblQuest = False. The CurrentQuestScore of each user in the quest is retrieved to ensure it is ultimately added to TotalScore.

User Logout

User status is set to offline by updating UserOnline = false. Any quests the player was on are updated to reflect the user has left and offline and stores play information to re-join the quest in the future if still available.

Administration Portal

The administration button will only be visible if Administrator = true for the user. The administration portal will retrieve a list of users displaying UserID and UserName. Additionally, in-progress quests by name. Therefore QuestID, QuestName, QuestPlayID, and UserID will all be retrieved where QuestStatus = True.

Kill in Progress Quest

To kill an in-progress quest, the QuestStatus will be updated to false. Additionally, the retrieval of users and their current scores in the quest will be retrieved to ensure they are added to TotalScore.

Delete User

The administrator can delete a user. This will remove their attributes from tblUser and cascade to any dependent records in tblQuestPlay, tblQuestPlayInventory, tblQuestScore, and tblChat.

Create User

When the administrator adds a new user, the system must first check to see if the username is unique by retrieving UserName then inserting new user details in tblUser.

Modify User

The administrator modifies a user by selecting their username for the list in the portal. Details can then be modified and updated in tblUser.

Entity Relationship Diagram Rationale

Players and their Quest's

The player's information is stored in tblUser. Details include their login credentials and total score data. Each quest is uniquely identified and named in tblQuest. Whilst each player that plays in a quest generates their own QuestPlayID in tblQuestPlay, respectively. This enables single and multiplayer options for each quest.

Quest and Quest Map

Each quest in tblQuest has its own unique map in tblQuestMap. This ensures that each game can be played with an independent map.

Quest Map, Map Tiles, and Assets

Each unique map populates with its own tiles and adds assets for players to collect. Each Quest has its own MapID in tblQuestMap, which in turn has its own tile configuration in tblQuestMapTile. Tiles with assets assigned a TileAssetID in tblQuestTileAsse, which enables players to score or lose point based on the asset value from AssetValue in tblAsset.

Quest Score and Inventory

Each player with their own unique QuestPlayID for each quest they are playing has their score calculated by tblQuestScore. This table calculates each player's current quest score as they accumulate assets. Any asset of positive value the user scores points, and the asset is added to their inventory in tblQuestPlayInventory. Their CurrentQuestScore is added to their TotalScore in tblUser to reflect their overall lifetime quest score.

Quest Chat

Each quest has a chat that all players in that quest can communicate. Chats are generated in tblChat and are displayed in the quest(QuestID) in which they are generated.

SQL

The SQL to build the database as per the entity-relationship diagram for the game creates a database call WizardQuest. Additionally, all DDL statements are compiled into one procedure call ddlWizardQuest, whilst all DML test statements exist in a procedure call dmlWizardQuest. Notes have been added to the DML statements to indicate what function they may be used for.

See repository file Milestone One – SQL.sql

tblUser

The user table contains all user information for account verification, privileges, and total score.

- **UserID**
 - Integer, auto incremental, not null, and the primary key
 - Used to uniquely identify a user in different games with scores and chats
- **UserName**
 - Varchar, unique, not null
 - Used to identify the user in gameplay, chat, verification, and administration
- **UserPassword**
 - Varchar, not null
 - Used to verify user at login
- **Email**
 - Varchar, not null
 - Contact details for each user
- **LoginAttempts**
 - Integer, not null
 - Counts unsuccessful logins up to five for account lock requirement
- **UserLocked, UserOnline, Administrator**
 - All Boolean, not null.
 - Used to show if the account is locked, user online, and user administrator, respectively.
- **TotalScore**
 - Integer, not null
 - Holds the user's overall score for all their games played so far

tblQuest

the quest table contains a primary record of each quest that is being played.

- **QuestID**
 - Integer, auto incremental, not null, and the primary key
 - Used to uniquely identify the game being played
- **QuestName**
 - Varchar, not null
 - Initiating player can give the game a unique name.
- **MaxUsers**
 - Integer, not null
 - Maximum players in a game at one time

- **ActivePlayer**
 - Integer, not null
 - Holds the value of the player number currently taking a turn
- **QuestStatus**
 - Boolean, not null
 - Value to indicate if the game is currently being played

tblChat

The chat table contains all in-game chat data.

- **ChatID**
 - Integer, auto incremental, not null, and the primary key
 - User to uniquely identify each individual chat in each game
- **UserID**
 - Foreign key constraint connects chat to individual user
- **Message**
 - Varchar, not null
 - Contains the message that will be displayed in the chat terminal during gameplay. Not null ensures no blank messages can be sent

tblQuestPlay

The play table connects each individual player to a game/quest.

- **QuestPlayID**
 - Integer, auto incremental, not null, and the primary key
 - Used to identify each instance of a player in a quest
- **UserID, TileID, ChatID, QuestID**
 - Foreign key constraints to complete individual player instance in a quest
- **PlayerNumber**
 - Integer, not null
 - Contains the number of the player currently in play. If PlayerNumber is equal to ActivePlayer in the quest, that is, the player currently in play

tblQuestScore

The score table calculates each score for a user in every quest they are currently in.

- **ScoreID**
 - Integer, auto incremental, not null, and the primary key
 - Used to identify each score for a user in every quest they are in
- **QuestPlayID, QuestID**
 - Foreign key constraints linking individual score to play instance and quest instance.
- **CurrentQuestScore**
 - Integer, not null
 - Contains the user's current score in the quest.

tblAsset

The asset table stores all assets that can potentially be on a tile for each quest.

- **AssetID**
 - Integer, auto incremental, not null, and the primary key
 - Used to identify each unique asset and its value
- **AssetName**
 - Varchar, not null
 - Short name used for inventory and scoring notifications during quest play
- **AssetDescription**
 - Varchar, not null
 - Short description used for scoring notifications during quest play
- **AssetValue**
 - Integer, not null
 - Value used for scoring points during quest play

tblQuestPlayInventory

The inventory table keeps a total of each player's accumulated inventory in each quest instance.

- **InventoryID**
 - Integer, auto incremental, not null, and the primary key
 - Used for a unique inventory for each quest the user is playing
- **QuestPlayID, AssetID**
 - Foreign key constraints linking the inventory to a user's quest instance and their current accumulated assets
- **Quantity**
 - Integer, not null
 - Value to indicate assets in inventory

tblQuestMap

The map table has a unique map for each instance of a quest.

- **MapID**
 - Integer, auto incremental, not null, and the primary key
 - Identifies each map for each quest.
- **QuestID**
 - Foreign key constraint, each map has one and only one quest.
- **xMax, yMax**
 - Integer, not null
 - Holds a value for the maps total size on the x and y-axis.

tblQuestMapTile

The map tile table has each tile for each map in every quest.

- **TileID**
 - Integer, auto incremental, not null, and the primary key
 - Identifies each tile for each map

- **MapID**
 - Foreign key constraint, each tile has one and only one map.
- **xPosition**
 - Integer, not null
 - Indicates tile row position
- **yPosition**
 - Integer, not null
 - Indicates tile column position
- **Active**
 - Boolean, not null
 - Indicates if a tile is active.

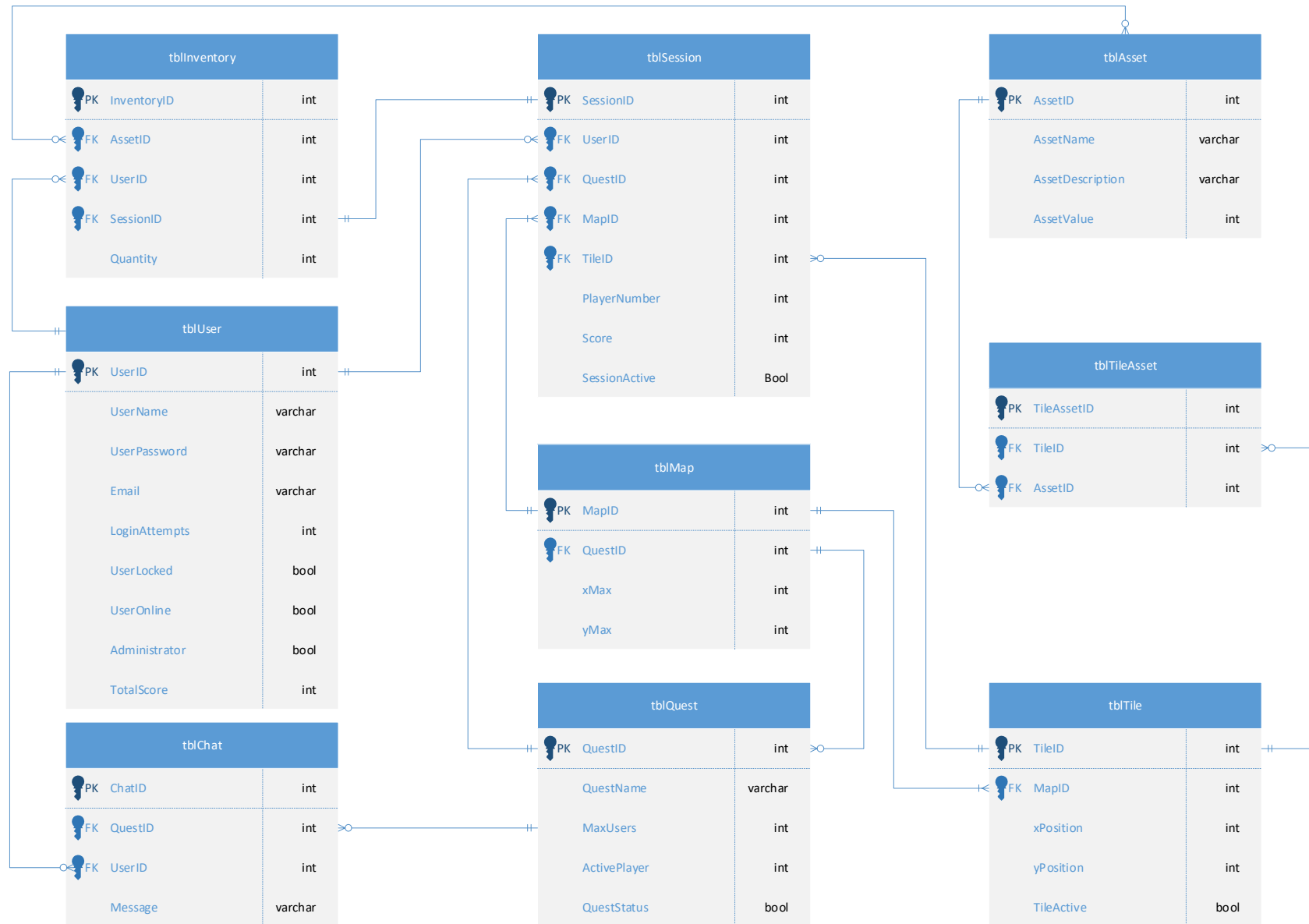
[tblQuestTileAsset](#)

The tile asset table assigns asset to various locations on each map in every quest.

- **TileAssetID**
 - Integer, auto incremental, not null, and the primary key
 - Identifies each instance of an asset of a tile
- **TileID, AssetID**
 - Foreign key constraints, a unique combination of asset and tile used for location and scoring.

Milestone Two

Entity Relationship Diagram – Iteration Two



CRUD Table – Iteration Two

Entity/Attribute	User Registers	User Login	Delete Account	Lock Account	Join Quest	New Quest	User Moves	User Chat	User Leaves Quest	Quest Ends	User Logout	Kill in Progress Quest	Delete User	Create User	Modify User
tblUser	C	R	RD		R	R	R	R	R	RU	RU		RD	RC	RU
UserID	C		D	R	R	R	R	R	R	R	R		RD	C	R
UserName	C	R	RD					R		R			D	RC	RU
UserPassword	C	R	RD										D	C	RU
Email	C		D										D	C	RU
LoginAttempts	C	RU	D										D	C	RU
UserLocked	C	RU	D	RU									D	C	RU
UserOnline	C	RU	D								RU		D	C	
Administrator	C	R	D										D	C	RU
TotalScore	C		D							RU			D	C	RU
tblQuest					R	C	RU	R	RU	RD		RD			
QuestID					R	C	R	R	R	RD		RD			
QuestName						RC				RD		D			
MaxUsers						C				RD		D			
ActivePlayer						C	RU		RU	RD		D			
QuestStatus						C				RD		D			
tblSession			D		RC	C	RU		R	RD		RD	RD		
SessionID			D		RC	C	R		R	RD		D	D		
UserID			D		RC	C	R		R	RD		D	RD		
QuestID			D		R	C	R		R	RD		RD	D		
MapID			D		R	C	R		R	RD		D	D		
TileID			D		C	C	R		R	RD		D	RD		
PlayerNumber			D		C	C	R			D		D	D		
Score			D		C	C	RU			RD		D	D		
SessionActive			D		C	C			RU	D		D	D		
tblInventory			D		C	C	RU			RD		D	RD		
InventoryID			D		C	C	R			RD		D	D		
AssetID			D		C	C	R			D		D	D		
UserID			D		C	C	R			RD		D	RD		
SessionID			D		C	C	R			RD		D	D		
Quantity			D		C	C	RU			D		D	D		
tblChat			D					C		RD		D	RD		
ChatID			D					C		D		D	D		
QuestID			D					C		RD		D	D		
UserID			D					C		D		D	RD		
Message			D					C		D		D	D		
tblMap					R	C	R		R	RD		D			
MapID					R	C	R		R	RD		D			
QuestID					R	C	R		R	RD		RD			
xMax						C				D		D			
yMax						C				D		D			

Entity/Attribute	User Registers	User Login	Delete Account	Lock Account	Join Quest	New Quest	User Moves	User Chat	User Leaves Quest	Quest Ends	User Logout	Kill in Progress Quest	Delete User	Create User	Modify User
tblTile					R	C	RU		RU	RD		D	RU		
TileID					R	C	R		R	RD		D	R		
MapID					R	C	R		R	RD		D			
xPosition					R	C	R			D		D			
yPosition					R	C	R			D		D			
TileActive					R	C	RU		RU	D		D	RU		
tblTileAsset						C	RD			RD		D			
TileAssetID						C	RD			RD		D			
TileID						C	RD			RD		D			
AssetID						C	RD			RD		D			
tblAsset						R	R								
AssetID						R	R								
AssetName						R	R								
AssetDescription						R	R								
AssetValue						R	R								

ERD and CRUD Iteration Two Discussion

Minor changes to ERD and CRUD have been made based on feedback and learning from creating MySQL procedures during milestone two development. Specific changes include:

- Minor changes to ERD relationship notations based on feedback.
- Additional SessionActive attribute to tblSession to indicate if a player has left the quest.
- The removal of the scoring table and the addition of a Score attribute in tblSession.
- Refactor the chat table to have both the QuestID and UserID as foreign keys.
- Minor changes to table names for shorter and terms and better understanding when coding.

CURD Activities MySQL Procedures

Each CRUD activity has been developed into a MySQL procedure and tested in the console application provided with this document.

User Registration

See the userRegistration procedure that creates a user record in tblUser, respectively.

User Login

See the userLogin procedure that verifies user credentials and login activity.

Delete Account

See the userDelete procedure that verifies the user credentials and deletes the user records.

Lock Account

Included as part of the login procedure.

Join Quest

See the joinQuest procedure that verifies the chosen quest and inserts the user's details into the game if successful.

New Quest

See the newQuest procedure that verifies chosen quest name and builds a new quest, map, tiles, assets, and user details.

User Moves

See the userMove procedure that verifies the user's input, scores point, update tiles, and inventory.

User Leaves Quest

See the leaveQuest procedure that updates the user's quest status and ensure the tile they were on is now available for other players.

Quest Ends

See the questCheck procedure that runs at the end of each userMove procedure. It validates the current quest to see if there are any players with a winning score and that there are still assets available. If the game is over, the questCheck procedure ends the game, updates the user's total score and informs them of the winner.

User Logout

See the userLogout procedure that updates online status.

Kill In Progress Quest

See the administratorKill procedure that kills any in-progress quest and removes all records.

Delete User

See administratorDelete procedure that deletes selected user and their records.

Create User

See administratorAdd that creates new user records.

Modify User

See administratorModify that modifies existing user records.

ACID and Multi-Player Game Support

ACID, according to Jepson (2001), is an excellent way to test overall quality. ACID is an acronym that describes four properties of a robust database system: atomicity, consistency, isolation, and durability. These features are scoped to a transaction, a unit of work that the programmer can define. A transaction can combine one or more database operations.

Atomicity

Jepson (2001) states that atomicity is an all-or-none proposition. Suppose you define a transaction that contains an UPDATE, an INSERT, and a DELETE statement. These statements are treated as a single unit with atomicity, and thanks to consistency (the C in ACID), there are only two possible outcomes: either they all change the database, or none of them do. Furthermore, Kaur (2021) states that atomicity is also known as the 'All or nothing rule', meaning transactions must be executed in their entirety to ensure the correctness of the database state.

With Wizard Quest, the procedures are designed to complete single functions for users. If the process does not complete, the database will not be updated. For example, the userMove procedure handles a player moving, scoring any points associated with that move, and setting the following players turn all in the same procedure. If the database failed at any point during the move, none of these changes would happen, keeping the database in a stable state. Additionally, the game structure is designed to ensure only one player at a time can move in a quest. This adds a further constraint to the database, ensuring only one user is making changes in a game.

Consistency

Consistency links into the previous discussion about users moving in the game, and Jepson (2001) discusses, consistency guarantees that a transaction never leaves your database in a half-finished state. If one part of the transaction fails, all of the pending changes are rolled back, leaving the database as it was before you initiated the transaction. For instance, when you delete a customer record, you should also delete all of that customer's records from associated tables.

In Wizard Quest, all quest data is deleted, and players scores are calculated when the quest ends, including any map entries, assets, and user quest details. If the game ends successfully, MariaDB (2018) states, new data will be added to the database and the resulting state will be consistent with existing rules.

Isolation

Jepson (2001) discusses isolation as keeping transactions separated from each other until they're finished. Furthermore, Tech School (2020) explains that when working with database transactions, one crucial thing we must do is choose the appropriate isolation level.

- The lowest isolation level is **read uncommitted**. Transactions in this level can see data written by other uncommitted transactions, thus allowing dirty read phenomenon to happen.
- The next isolation level is **read committed**, where transactions can only see data that other transactions have committed. Because of this, dirty read is no longer possible.
- A bit more strict is the **repeatable read** isolation level. It ensures that the same select query will always return the same result, no matter how many times it is executed, even if some other concurrent transactions have committed new changes that satisfy the query.

- The highest isolation level is **serializable**. Concurrent transactions running in this level are guaranteed to yield the same result as if they're executed sequentially in some order, one after another without overlapping.

Importantly Oracle (2021) state that isolation level read uncommitted and serializable change processing behaviour so drastically that they are rarely used.

It is important to note that these isolation level behaviours as described are specific to MySQL. Given these insights and further discussions in class, a global isolation level of read committed has been applied to the Wizard Quest database. Isolation at this level protects the integrity within each transaction and prevents dirty reads. This level, combined with the nature of the game being a turn-based game, is the best solution ensuring each transaction is completed with the latest data available, system performance is maintained, and accurately supports a multiplayer transaction environment.

Durability

Jepson (2001) discusses that durability guarantees that the database will keep track of pending changes in such a way that the server can recover from an abnormal termination. Hence, even if the database server is unplugged in the middle of a transaction, it will return to a consistent state when it's restarted. Furthermore, Kaur (2021) state durability ensures that once the transaction has completed execution, the updates and modifications to the database are stored in and written to disk, and they persist even if a system failure occurs. These updates now become permanent and are stored in non-volatile memory. The effects of the transaction, thus, are never lost.

The ACID properties, in totality, provide a mechanism to ensure correctness and consistency of a database in a way such that each transaction is a group of operations that acts as a single unit, produces consistent results, acts in isolation from other operations and updates that it makes are durably stored.

References

- Jepson, B. (2001). *Building better databases*. Web Techniques.
<https://people.apache.org/~jim/NewArchitect/webtech/2001/09/jepson/index.html>
- Kaur, A. (2021). *ACID Properties in DBMS*. Geeks for Geeks. <https://www.geeksforgeeks.org/acid-properties-in-dbms/>
- MariaDB. (2018). *ACID Compliance: What It Means and Why You Should Care*. MariaDB
<https://mariadb.com/resources/blog/acid-compliance-what-it-means-and-why-you-should-care/>
- Oracle. (2021). *MySQL 8.0 Reference Manual*. MySQL.
https://dev.mysql.com/doc/refman/8.0/en/glossary.html#glos_isolation_level
- Tech School. (2020). *Deeply understand Isolation levels and Read phenomena in MySQL & PostgreSQL*. Tech School Guru. <https://dev.to/techschoolguru/understand-isolation-levels-read-phenomena-in-mysql-postgres-c2e>