March 1, 2021

Project Report Snake Game

DAT602 Frank Schulz

Contents

[MILESTONE ONE 2](#_Toc73129737)

[1. Game Description 3](#_Toc73129738)

[1.1 BASE IDEA 3](#_Toc73129739)

[1.2 LOGGING IN 3](#_Toc73129740)

[1.3 MAIN MENU 3](#_Toc73129741)

[1.4 GAMEPLAY 3](#_Toc73129742)

[2. Storyboards 4](#_Toc73129743)

[STORYBOARD 1 – LOGIN SCREEN 4](#_Toc73129744)

[STORYBOARD 2 – INVALID USER 4](#_Toc73129745)

[STORYBOARD 3 – REGISTER A NEW ACCOUNT 4](#_Toc73129746)

[STORYBOARD 4 – ACCOUNT LOCKED 5](#_Toc73129747)

[STORYBOARD 5 – LOGGED IN 5](#_Toc73129748)

[STORYBOARD 6 - GAME 6](#_Toc73129749)

[storyBOARD 7 – GAME OVER 6](#_Toc73129750)

[STORYBOARD 8 – WIN 6](#_Toc73129751)

[STORYBOARD 9 – EDIT USER DETAILS 7](#_Toc73129752)

[STORYBOARD 8 – CONFIRMATION DIALOG 7](#_Toc73129753)

[Conceptual ERD 8](#_Toc73129754)

[Use Case Diagram 9](#_Toc73129755)

[Logical ERD 10](#_Toc73129756)

[C.R.U.D table 11](#_Toc73129757)

[SQL 12](#_Toc73129758)

[DDL 12](#_Toc73129759)

[DML 12](#_Toc73129760)

[MILESTONE TWO 13](#_Toc73129761)

[SQL Procedures 14](#_Toc73129762)

[Multiplayer Support and ACID 14](#_Toc73129763)

[Isolation Levels and Locking 14](#_Toc73129764)

[Bibliography 15](#_Toc73129765)

# MILESTONE ONE

## 1. Game Description

### 1.1 BASE IDEA

“Snake” is a multiplayer, two-dimensional, top-down game in which the player controls a snake that gets longer as it eats more food, which is how points are earned.

### 1.2 LOGGING IN

Upon first opening the game users will be given the option of either logging in or playing on the guest account, which does not keep a score specific to each player and rather just keeps the highest score achieved on the account. This makes it like all other accounts with the exception of not requiring the user to input any login details.

### 1.3 MAIN MENU

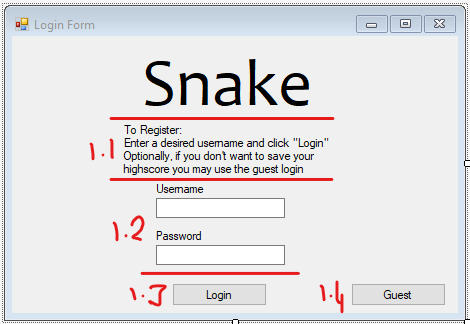
Once an account has been logged in to, the user will be taken to the main menu, which will display their username and high score, as well as other online players and their high score. Here the user will have the option of logging out, editing the details, or starting a new game. If the user is an admin, they may also open an admin window.

### 1.4 GAMEPLAY

The more you eat, the longer the longer the snake will get, however, if the snake bumps into any obstacles (game boundary, itself or otherwise) it will be game over. Your score for any given game is determined by whatever score you manage to achieve before reaching game over. Survive longer than your opponent to win.

## 2. Storyboards

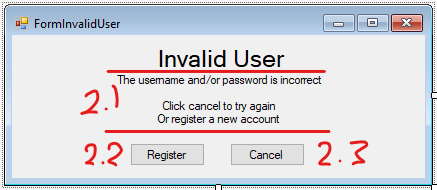
### STORYBOARD 1 – LOGIN SCREEN



The login screen provides the user with two textboxes where a username and password can be entered.

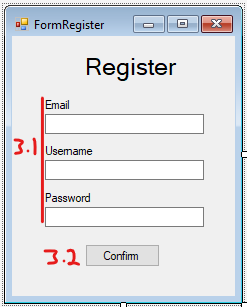
* 1. Instructions for how to register and guest login notice.
  2. Fields for username and password entry
  3. Login button, if the username and password are successfully validated, then move to “Storyboard 4”. If not, then move to “Storyboard 2”.  
     To many login attempts will result in that user being temporarily locked out of their account. (Storyboard 4)
  4. Guest login that doesn’t require a personal login and saves its own high score.

### STORYBOARD 2 – INVALID USER



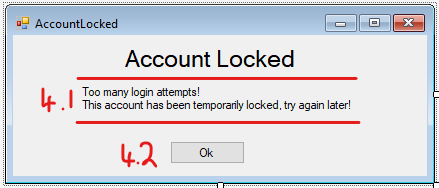
* 1. Information to guide the user.
  2. “Register” will take the user to the registration screen. (Storyboard 3)
  3. “Cancel” will close the message and allow the user to re-attempt the login (Storyboard 1)

### STORYBOARD 3 – REGISTER A NEW ACCOUNT



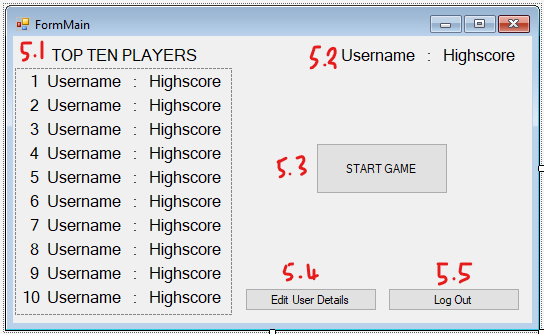
* 1. Enter user details.
  2. Confirm account creation. Move to Storyboard 5.

### STORYBOARD 4 – ACCOUNT LOCKED



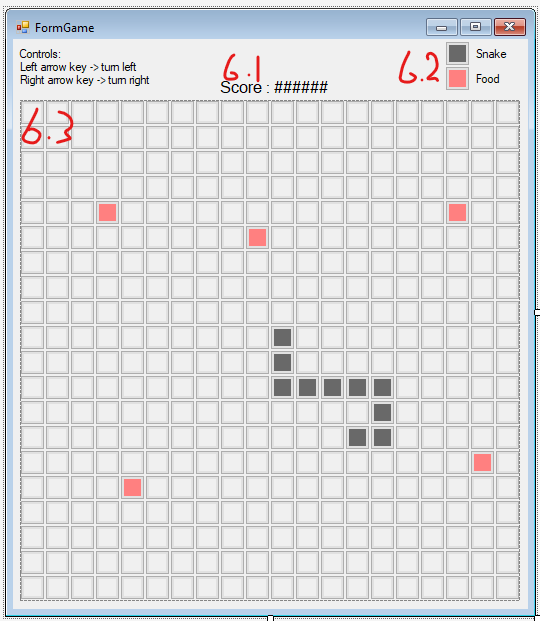
* 1. Information for the user, letting them know their account has been temporarily locked.
  2. Takes the user back to the login screen.

### STORYBOARD 5 – LOGGED IN



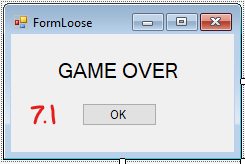
* 1. A list that shows the current top ten players
  2. Shows the logged in players username and highscore
  3. Start a game. Move to Storyboard 6.
  4. Edit the details of the currently logged in user. (Storyboard 9)
  5. Log out the current user. (Storyboard 10 -> Storyboard 1)

### STORYBOARD 6 - GAME



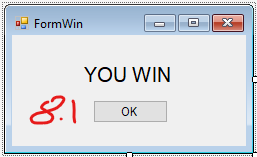
1. Displays the score that has been achieved in the current game.
2. A key that explains the entities on the grid.
3. The game grid.

### storyBOARD 7 – GAME OVER



1. Return to Storyboard 6

### STORYBOARD 8 – WIN

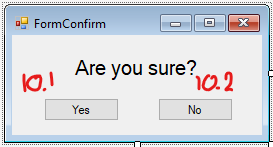


1. Return to Storyboard 6

### STORYBOARD 9 – EDIT USER DETAILS

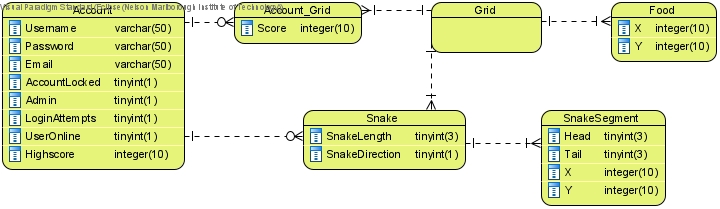
1. Optional field for changing the username.
2. Optional field for changing the email.
3. Optional field for changing the password.
4. Field to enter current password as security measure.
5. Buttons to either confirm or cancel account edits. (Return to Storyboard 6)

### STORYBOARD 8 – CONFIRMATION DIALOG

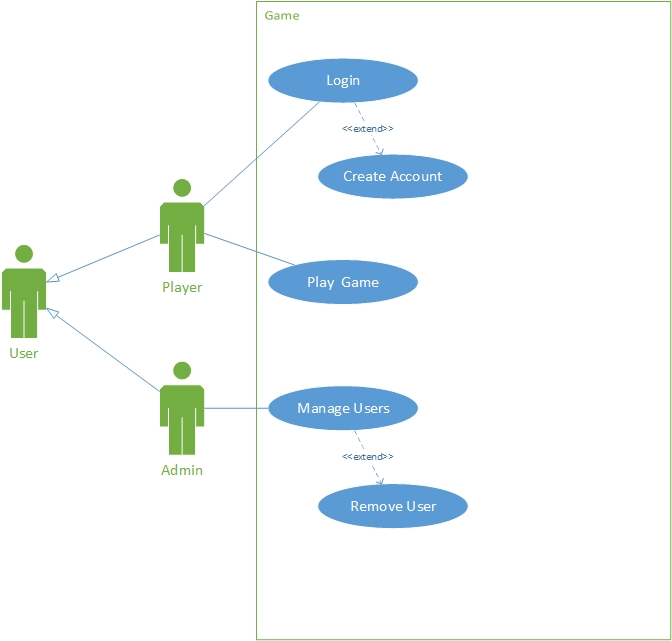


1. Confirm action (go to intended Storyboard)
2. Cancel action (return to previous Storyboard)

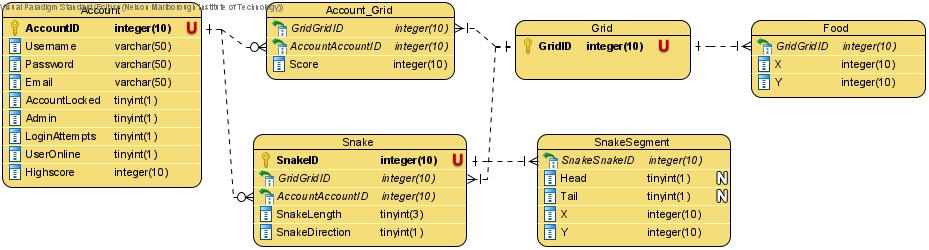
## Conceptual ERD



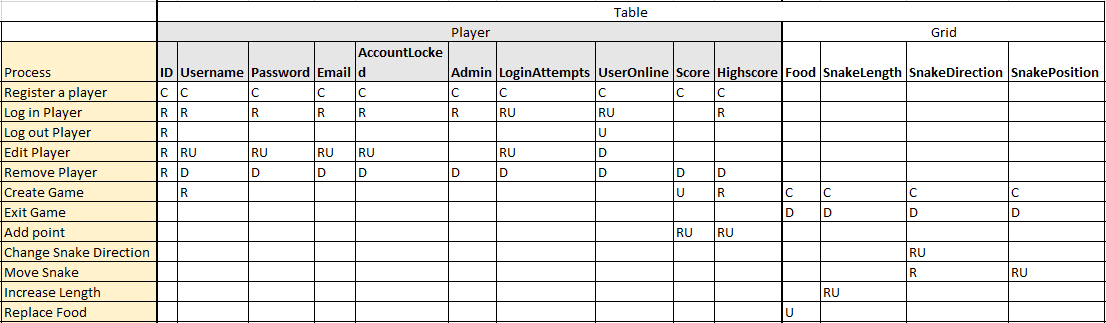
## Use Case Diagram



## Logical ERD



## C.R.U.D table



## SQL

### DDL

See attached file: DAT602 - Frank Project DDL-DML.sql

### DML

See attached file: DAT602 - Frank Project DDL-DML.sql

# MILESTONE TWO

### SQL Procedures

### Multiplayer Support and ACID

Describe how your system supports multi-player game play, in terms of transactions and database table specification in the target DBMS. Discuss how the transactions are reliable in terms of the acronym ACID.

My system supports multiplayer gameplay through specifying which player is affected in relevant procedures such as my “turnSnake” procedure, which requires a specific player to be given as an input. Because of this, multiple players can exist in my database at once and can be used concurrently due to my procedures always having a specific target.

I have used the Transaction Isolation Level: Repeatable Read (or Read Repeat).

This means that while a procedure is running it will only be able to read the database as it was when the procedure started. This works well for my game as, while it is live gameplay, it updates on a regular basis rather than being real-time, negating the need for something like the Read Committed Isolation Level.

The transactions in my game use atomicity through the use of checks that determine whether the transaction would complete in the way it is intended to. If a transaction were to run with bad data or other fault, it would instead pick up on the error and send an error message leaving the database unchanged. In this way I have included:

* Atomicity

All or nothing, meaning it either completes successfully or fails.

* Consistency

Ensures that a transaction can only bring the database from one valid state to another.

* Isolation

Guarantee that a transaction is not affected by another transaction yet to complete.

However the database remains volatile for now as I have not yet implemented a method of saving and retrieving it, which falls under:

* Durability

Once the transaction is completed, it remains in the database. i.e. it is non-volatile.

## Bibliography