DAT602 Project Corvus

Hamish Drogemuller

13515109

Contents

[1. Introduction 3](#_Toc106287341)

[2. Redesign Decision 4](#_Toc106287342)

[3. Game Description 5](#_Toc106287343)

[3.1 Game Premise 5](#_Toc106287344)

[3.2 Login 5](#_Toc106287345)

[3.3 Home 5](#_Toc106287346)

[3.4 Gameplay 5](#_Toc106287347)

[3.5 Objective and Scoring 5](#_Toc106287348)

[3.6 Administration 6](#_Toc106287349)

[4. Storyboards 7](#_Toc106287350)

[4.1 Login 7](#_Toc106287351)

[4.2 Register 7](#_Toc106287352)

[4.3 Home 8](#_Toc106287353)

[4.4 Gameplay 8](#_Toc106287354)

[4.5 Game End 9](#_Toc106287355)

[4.6 Admin Screen 9](#_Toc106287356)

[5. Screen Design Decisions 10](#_Toc106287357)

[6. Logical ERD 10](#_Toc106287358)

[7. C.R.U.D Table 11](#_Toc106287359)

[7.1 C.R.U.D Analysis 11](#_Toc106287360)

[8. Procedures 13](#_Toc106287361)

[8.1 Registration 13](#_Toc106287362)

[8.2 Login 13](#_Toc106287363)

[8.3 Game Creation 13](#_Toc106287364)

[8.4 Gameplay 13](#_Toc106287365)

[8.5 Game End 13](#_Toc106287366)

[8.6 Administration 13](#_Toc106287367)

[9. A.C.I.D 14](#_Toc106287368)

[9.1 Atomicity 14](#_Toc106287369)

[9.2 Consistency 14](#_Toc106287370)

[9.3 Isolation 14](#_Toc106287371)

[9.4 Durability 14](#_Toc106287372)

[10. Visual Studio Implementation 15](#_Toc106287373)

[11. Procedure Implementation Class 16](#_Toc106287374)

# Introduction

This document will serve as the reference documentation for the updated Corvus game. This document aims to list and explain the development of the Corvus game, presenting information on the design and reasoning of elements of the game.

# Redesign Decision

The decision to redesign the Corvus game was not one taken lightly, I came to the conclusion that a redesign was required as the initial design I used to approach the development of Corvus were not working smoothly together which made development harder than required. I decided that a simplification of the database the game operates from was required. The ideal conclusion of this redesign is to provide a more consistent enjoyable experience for those who wish to play Corvus.

# Game Description

Corvus aims to provide a player vs player experience where upon the players will compete against each other to collect crows(corvus) while avoiding traps.

## Game Premise

The game outlined within this document referred to as “Corvus” will revolve around the user trying to catch crows while avoiding placed traps. The aim is for the user to catch more crows within the set time limit than their opponent, a different form of scoring could be implemented in which the first player to reach ‘x’ amount of crows wins.

## Login

When the game is run, players will be prompted to log in with an email/username and password. If the email isn’t found in the database then the player will be asked if they wish to create an account. When registering an account the player will be required to enter a unique email and a password. However, if the email/username is already in the database and the password does not match what is stored against the email, the app will log the wrong attempts. The user will become locked out upon 5 wrong entries. Once an account is created the user will be prompted to log in again.

## Home

When a successful login happens the user will be taken to the Corvus home page. The home page is the users hub for experiencing Corvus. This home screen will display a variety of information such as players in the current lobby, players highscores and your current username. From the home page the user will be able to navigate to other screens with the use of buttons. These will be “Account”, “Admin”, “Join Game”, “New Game” and “Logout”. Admin will only be present for those players that are admins.

Account – Provides general account information, the ability to edit username as well as the option to delete their account.

Join/New Game – These buttons provide the user with the ability to Join a game in progress or start a fresh new game.

Admin – If the user has the correct permissions they will be able to access the admin functions. These provide the admin with the ability to use the admin functions.

Logout – The logout button will log the user out of Corvus, taking them back to the login screen.

## Gameplay

The gameplay of Corvus will take place upon a ten by twenty grid where the players will race to collect as many crows as possible within a set period. At the same time players will also have to avoid traps, traps may influence the player in 1 of 2 ways. Either the trap will deduct points from the user (Thematically: Make them drop crows) or remove crows from tiles in a radius of 1 from the trap (Thematically: Scaring crows off with noise).

## Objective and Scoring

The objective of Corvus is to collect as many crows as possible within the time limit. Each crow adds 1 to the players score, however if a player trips a trap they will lose 3 points.

## Administration

Administration functions are required to maintain the integrity of the Corvus game space. Providing users with the ability to get into their account if they forget their password. Providing admins with the ability to remove players that are detrimental to the Corvus ecosystem.

# Storyboards

## Login

Graphical user interface, application

Description automatically generated

## Register

Graphical user interface

Description automatically generated

## Home

Graphical user interface, text, application

Description automatically generated

## Gameplay

Graphical user interface, application, Word

Description automatically generated

## Game End

Graphical user interface, application, Word

Description automatically generatedGraphical user interface, application, Word

Description automatically generated

## Admin Screen

Graphical user interface, Word

Description automatically generated

# Screen Design Decisions

**Overall:**

The overall design I wanted to encapsulate with the entire game application is a minimalistic, simple-to-follow design.

**Login:**

The login screen was designed to be minimal and easy to follow, Providing easy-to-follow steps that allow users to log in as well as register if it’s their first time on the app. Since the user has to sign up if their name/email isn’t in the system the login screen will also function as a registration screen. The user can put in an email and password, then be prompted for a username and any more information the app requires.

**Home:**

For the home screen, I wanted to maintain the minimal aesthetic but also provide a simple interface that doesn’t overwhelm the user with options, to this end I made sure that big groups of data were put behind headers. Such as hiding all of the available lobbies under the join game header. I also wanted this screen to be the “hub” for the game, allowing the user to do anything they need to do before getting into a game. This includes creating a game, joining a game, editing options, running the administration window if you have the privileges, and logging out of the game.

**Gameboard:**

I wanted the gameboard to feel like it was an open expanse that the user had to defend. To achieve this I decided to make the game space a 20x10 board. This would provide each player with a 10x5 space to defend, the size of each side means that the players can place traps and decoys but still have to move around to scarecrows.

**Admin:**

The admin screen is all about maintaining the game, to this extent I wanted to make it a concise screen with all the necessary tools ready to use upon screen boot. To this end, I included all available commands compactly on this screen. Entity Relational Diagram

# Logical ERD

Diagram

Description automatically generated

# C.R.U.D Table

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Entity/Attribute | **Check For Username** | **Register Account** | **Check For Correct Password** | **Lock Account** | **Successful Login** | **Start New Game** | **Quick Join/Join Lobby** | **Player Moves** | **Scoring** | **Game Ends** | **Player Logs Off** | **Open Admin Console** | **Admin Kill Running Game** | **Admin Delete Player** | **Admin Creates New Player** | **Admin Edits Existing Player Info** | **Player sends a Chat Message** |
| **Account** | R | C |  |  | R | R |  | R | R, U | U |  | R |  | D | C | R, U | R |
| Username | R | C |  |  | C |  |  |  |  |  |  |  |  | D | C | R, U |  |
| Password |  | C | R |  | C |  |  |  |  |  |  |  |  | D | C | R, U |  |
| Email |  | C |  |  |  |  |  |  |  |  |  |  |  | D | C | R, U |  |
| Locked User |  | C |  | U |  |  |  |  |  |  |  |  |  | D | C |  |  |
| Admin |  | C |  |  |  |  |  |  |  |  |  | R |  | D | C |  |  |
| Login Attempts |  | C |  | R |  |  |  |  |  |  |  |  |  | D | C |  |  |
| Online |  | C |  |  | U |  |  |  |  |  | U |  |  | D | C |  |  |
| Wins |  | C |  |  |  |  |  |  |  | U |  |  |  | D | C |  |  |
| Losses |  | C |  |  |  |  |  |  |  | U |  |  |  | D | C |  |  |
| Score |  | C |  |  |  |  |  |  | R, U | U |  |  |  | D | C |  |  |
| HighScore |  | C |  |  |  |  |  |  |  | U |  |  |  | D | C |  |  |
| **Tile** |  |  |  |  |  | C | R | R |  | D |  |  | D |  |  |  |  |
| **tileID** |  |  |  |  |  | C | R | R |  | D |  |  | D |  |  |  |  |
| **row** |  |  |  |  |  | C | R | R |  | D |  |  | D |  |  |  |  |
| **Column** |  |  |  |  |  | C | R | R |  | D |  |  | D |  |  |  |  |
| **crow** |  |  |  |  |  | C | R | R |  | D |  |  | D |  |  |  |  |
| **trap** |  |  |  |  |  | C | R | R |  | D |  |  | D |  |  |  |  |
| **hometile** |  |  |  |  |  | C | R |  |  | D |  |  | D |  |  |  |  |
| **playerLocation** |  |  |  |  |  | R | R | R,U |  | U |  |  | U |  |  |  |  |
| **username** |  |  |  |  |  | R | R | R |  | U |  |  | U |  |  |  |  |
| **tileID** |  |  |  |  |  | R | R | R,U |  | U |  |  | U |  |  |  |  |
| **Chat** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| chatID |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | C |
| username |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | R |
| message |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | C |

## C.R.U.D Analysis

To develop the CRUD table above I used the entities from my ERD as the left hand column and the storyboards further up in this report as the actions along the top.

**Check For Username, Register account, Check for Password, Lock account and Successful Login:**

I decided to merge these events as it made the most sense to me, this event will be achieved by using a simple select where statement that looks for a specific username and password in the account table, if neither is found the user will be prompted to fill in extra information to register, if the username is found but the password isn’t/is incorrect then the user will be requested to try again and the login attempts column of the Account table will be incremented by one. The procedure will then check to see if the login attempts column is greater than or equal to 5 and if it is the account will become locked. If both username and password are correct the user will be logged in and taken to the home screen.

**Quick Join/Join Lobby:**

When a player joins a lobby the database will need to create new data as well as pull existing data from session and game tables. This will be more complex than just creating a game.

**Create Game:**

The database will need to populate the session and game tables with data so that someone wanting to join can pull the data from those tables.

**Player Move, Scoring:**

When the player moves the tile will be checked to see if there is a crow within a 1 tile space of it or if there is an item on the tile, if there is a crow present the crow will be “scared” and moved to a random tile on the opposing half, if there is an item it will be picked up. Score will be calculated at select intervals. When this is done the system will count the amount of tiles with a Crow status on the opponents half and add that many points to the points count, after this count, if a player has equal or greater to 100 points the game will be over.

**Game End:**

Upon game end the session will be concluded, the player with the most points or who reached 100 first will receive a you win message and the opponent will receive a you lose message. Both players Win, Loss and scoreperminute columns in the account table will be updated with new information.

**Log Off:**

Upon logout the players online status will be change to false and the ActiveTile column will be updated with with the tileid of the players last location.

**Admin:**

When the admin window is booted the database will retrieve the online players as well as session and game info. This will allow the admin to kill active games. The admin also has the power to create new records or edit records in the account table. This facilitates the addition of new players.

**Chat:**

When a player wants to send a chat it will update the chat table with their playerID and the message they want to type.

# Procedures

## Registration

## Login

## Game Creation

## Gameplay

## Game End

## Administration

# A.C.I.D

ACID is a set of principles that help to insure that your database processes transactions reliably and without incident.

## Atomicity

Atomicity ensures that all transactions execute correctly or fail but do not impact upon further transactions. The theme of atomicity is to make sure all transactions are individually wrapped and can’t be affected by previous transactions failing.

Atomicity is necessary for a database because it ensures that the database tables and entries will maintain integrity and all necessary functions will be carried out without interruption from the others. An example from our database would be when a player moves to a new tile, once the player moves there are several procedures that need to be run. Surrounding squares must be checked for crows, crows need to be added to opposing boards and scoreboards need to be updated. Without atomicity the game could become stuck if a crash happened before the opposing boards were updated with crows. Therefore no crows would appear on the opposing board once the game was resumed. This would then lead to a stalemate as nobody would be able to gain points.

## Consistency

The Consistency principle revolves around maintaining the integrity of the database before and after transactions are run. Ensuring that invalid data is not inserted during transactions makes sure our database remains stable throughout any process that may be executed.

## Isolation

The Isolation Principle states that every transaction committed in the database is self contained and not impactful on another transaction that may be running at the same time. Isolation is controlled through the use of Lock-based Concurrency Control or Multi-Versioned Concurrency Control.

## Durability

The Durability Principle works to ensure that changes made with transactions remain a part of the database and that these additions are not lost. Durability revolves around adequate storage and backup options to ensure no data is lost.

# Visual Studio Implementation

# Procedure Implementation Class