Title Page

Project Title: Noverca
Student Name: Joshua Eum
Date: 3rd July 2025
Course: Software Engineering Stage 6
GitHub URL: https://github.com/sunghaeum/Noverca

Table Of Contents

1. Identifying and Defining

1.1. Problem Statement

Currently, there is a lack of engagement between students and their schoolwork that prevents students from learning effectively. This loss of motivation is often the result of repetitive, tedious school tasks, as well as a decrease in the students' attention span. Without interactive or otherwise meaningful learning experiences, students can find it hard to retain concepts and focus in school environments.

1.2 Project Purpose and Boundaries

What is the project trying to achieve? What is in and out of scope?

This project aims to spark an interest in learning. By introducing basic physics concepts through an interactive 2D platformer, the game hopes to create curiosity about learning these concepts in more depth, which students can seek out in their own schools. As a result, creating an entire physics curriculum, or integrating a multiplayer system into the game would be out of scope, and the game would rather be accessible, engaging and functional to students.

1.3 Stakeholder Requirements

Who are your stakeholders? What do they need from this software?

As an educational game, the primary stakeholders would consist of students and teachers. Being a game for learning, students would be playing the game as an alternative to other mundane school activities, whilst teachers would be distributing the software to their students, in an attempt to spark a curiosity for learning.

1.4 Functional Requirements

- The player is able to move around, jump and interact with physics-based objects.
- Signs to provide context on physics concepts
- Working main menu and levels system
- Respawn or reset system in case of player failure

1.5 Non-Functional Requirements

- Intuitive and/or easy to navigate UI
- Code follows secure software guidelines
- Prioritised user security
- Game should run smoothly on all devices

1.6 Constraints

While developing my project, I encountered some constraints that limited me from further improving my game. These included time, where I needed to finish before the deadline and thus restricted me from implementing more levels and features, as well as budget constraints, meaning that only free assets and tools could be used.

2. Research and Planning

2.1 Development Methodology

I used an Agile development methodology, where projects are broken down into smaller dynamic phases called sprints. After every sprint, developers look back and reflect to identify potential improvements for the next iteration, enabling continuous improvement and flexibility. This method suited the experimental nature of game development, where rather than everything being planned, mechanics are tested as they are made. For example, after implementing the initial player movement, I noticed that some surfaces caused the player to stick unnaturally. Using Agile principles, I adjusted friction settings to fix the issue.

2.2 Tools and Technology

For my project, I used Unity as the main game engine, with C# as the programming language and Visual Studio Code as the integrated development environment (IDE). I also used inbuilt Unity structures, including TextMeshPro, for UI text rendering, Animator for sprite animation, RigidBody2D for physics handling and Cinemachine to track the player and for other dynamic camera effects.

2.3 Gantt Chart / Timeline

	Week									
Task	1	2	3	4	5	6	7	8	9	10
Planning (researching ideas, identifying problem)										
Design (creating level concepts, UI layouts)										
Development (player mechanics, UI systems)										
Testing (unit testing, peer feedback, bugfixing)										
Evaluation (writing documentation, reflection)										

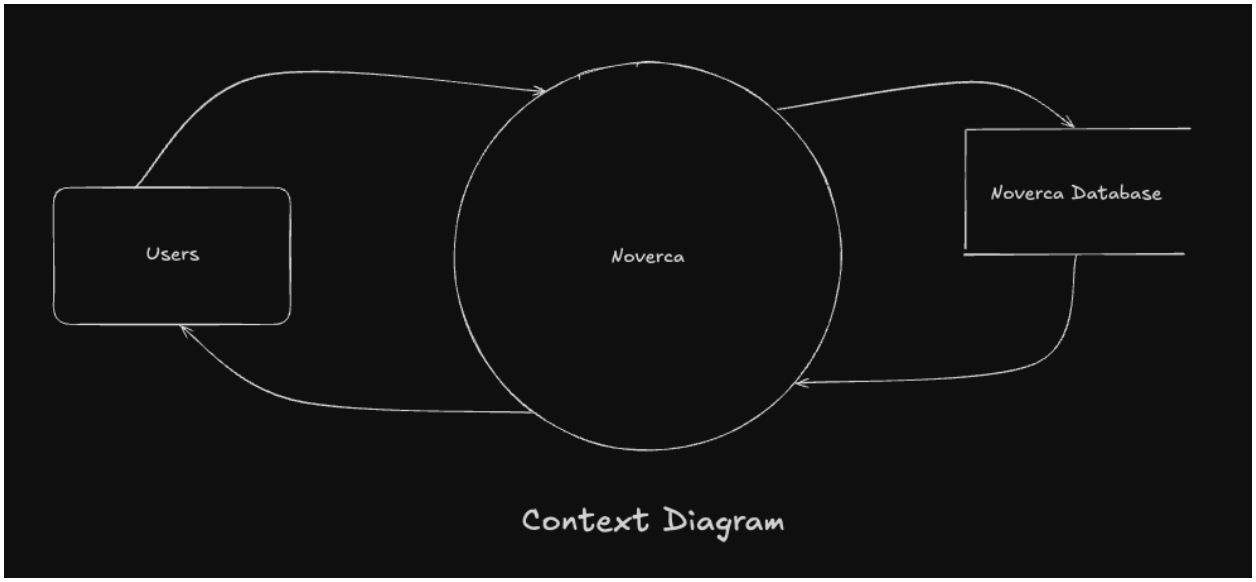
2.4 Communication Plan

How will you engage with your client or simulate feedback?

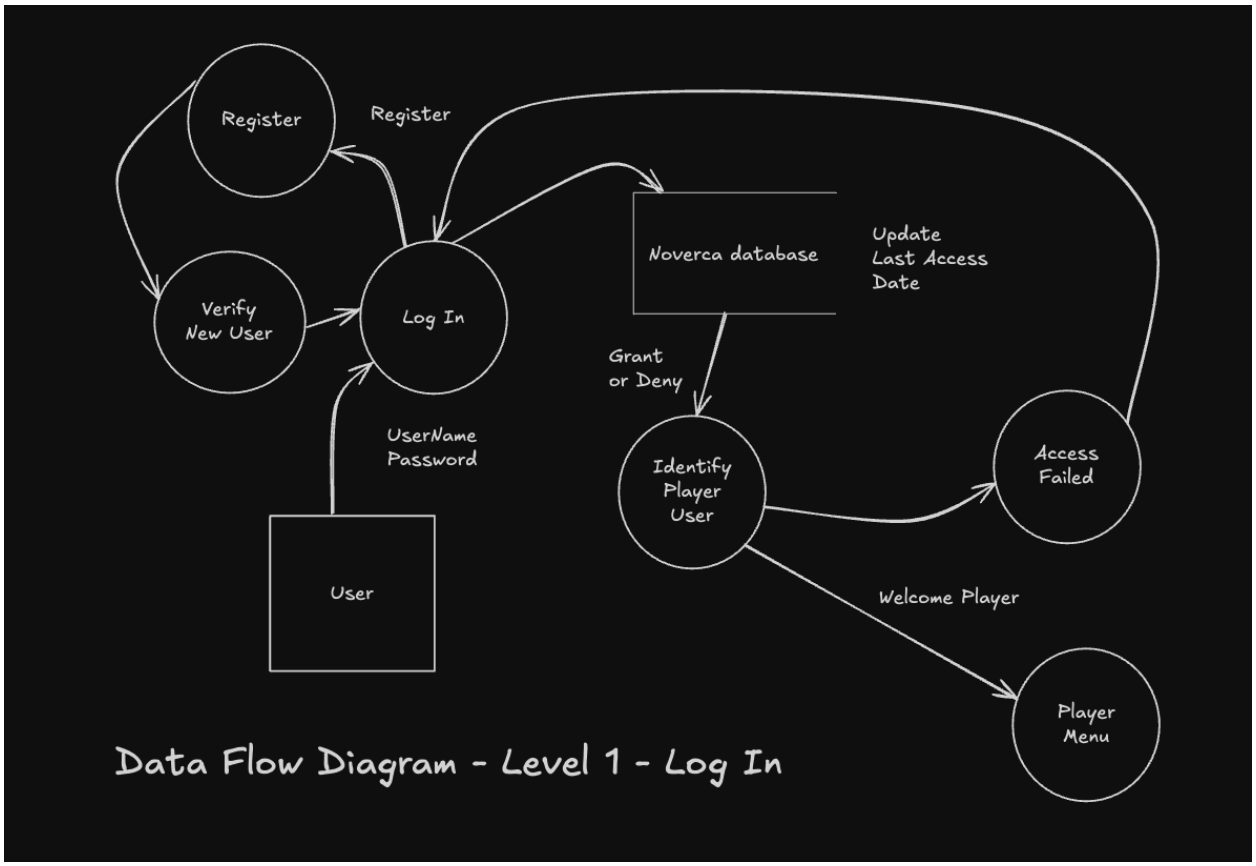
To simulate feedback for my project, I would use peer testing and observation by inviting classmates to try the game after it has been developed. To gather feedback, I would ask testers to identify any bugs or areas for improvement, which would be recorded informally and used to guide the game's future direction. This could be extended to a real-world setting by using surveys or interviews.

3. System Design

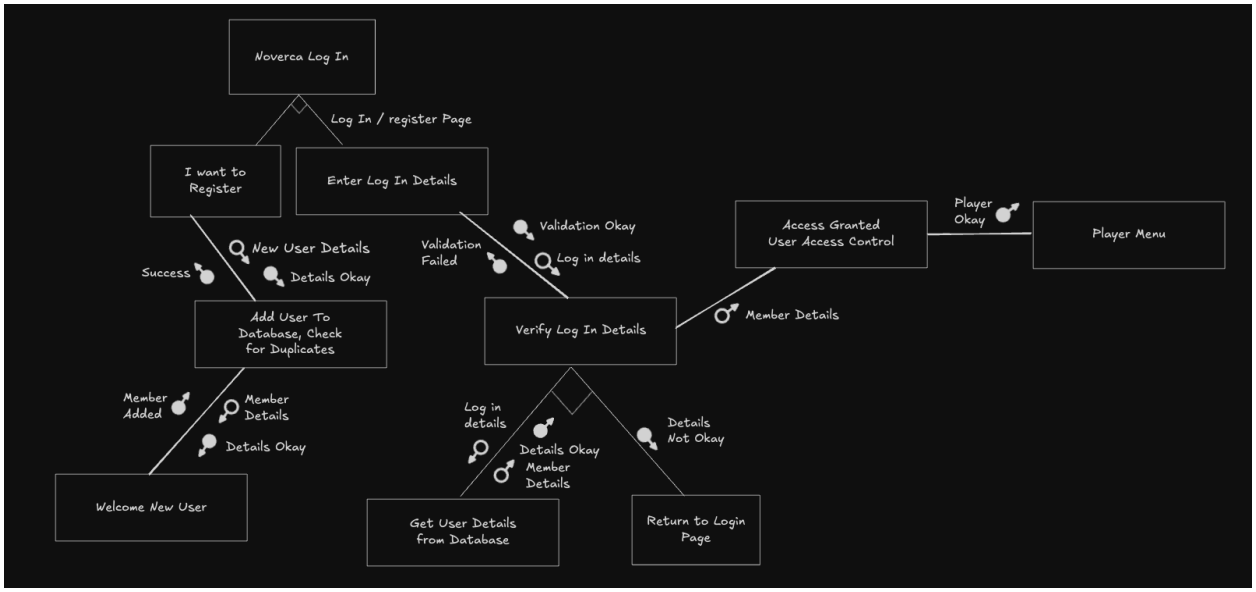
3.1 Context Diagram



3.2 Data Flow Diagram (Level 1)



3.3 Structure Chart



3.4 IPO Chart

Script Name	Input	Process	Output
MainMenuManager	Username, password, button clicks	Login/signup logic, file save/load, UI switching, password visibility, fade transition	Access to play scene, user messages
PlayerController	Horizontal/Jump input, ground detection	Handles player physics, animation, direction, and sound	Movement, jumping, sound and animation
MusicController	Audio clip, start delay	Skips intro, plays clip, loops audio from set point	Seamless partial music loop
PulleyManager	Rigidbody2D positions	Simulates pulley physics between a basket and platform	Basket and platform react to tension
GameTimer	Game time, delay	Displays a stopwatch with milliseconds and can stop on demand	On-screen timer

VirtualCameraShake	Shake trigger from other scripts	Applies short screen shake effect to camera using random offset	Visual screen shake

3.5 Data Dictionary

Variable Name	Type	Purpose
loginUsernameField	TMP_InputField	Username input (login)
loginPasswordField	TMP_InputField	Password input (login)
loginMessageText	TMP_Text	Displays login errors
loginSuccessMessageText	TMP_Text	Shows “Sign up successful” after redirect
signupUsernameField	TMP_InputField	Username input (signup)
signupPasswordField	TMP_InputField	Signup password input
signupConfirmPasswordField	TMP_InputField	Confirmation field for signup password
signupMessageText	TMP_Text	Displays signup errors
userDatabase	Dictionary	Stores usernames and hashed passwords
moveSpeed	float	Horizontal speed of player
jumpVelocity	float	Upward force applied on jump
gravity	float	Custom downward force
groundCheckBoxSize	Vector2	Size of ground detection zone
walkSound, jumpSound	AudioClip	Sound effects on player action
loopEndTime	float	End point of the looping music section

delayBeforeStart	float	Initial delay before music begins
audioSource	AudioSource	Plays and loops music
basket, platform	Rigidbody2D	Pulley system objects being affected
pulleyA, pulleyB	Transform	Anchors for simulating pulley paths
ropeLength	float	Desired rope length constraint
stiffness, damping	float	Force tuning for pulley spring behavior
timerText	TextMeshProUGUI	Displays formatted elapsed time
elapsedTime	float	Time since timer started
StartTimer(), StopTimer()	Method	Starts/stops in-game stopwatch
ShakeCamera(intensity)	Method	Triggers short camera shake
shakeDuration, shakeMagnitude	float	Controls camera shake effect

4. Producing and Implementing

4.1 Development Process

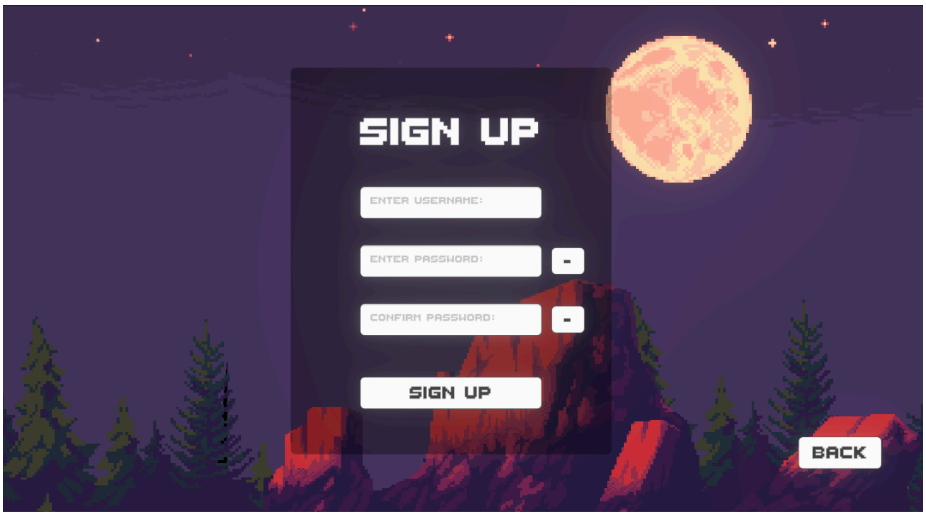
My development process used an Agile methodology to suit the experimental nature of game development. Rather than planning the whole project from the beginning, I worked in short development sprints, which allowed me to gradually build and improve the game. This also allowed for more freedom, where I could implement any game features that I wanted, rather than sticking to a strict plan. Through this procedure, each feature, such as player movement, UI systems and interactions were first added, then tested and refined before continuing with development. As a result, issues were easier to respond to and take care of based on unit tests and feedback, and the development process was more flexible, allowing me to experiment with ideas and features.

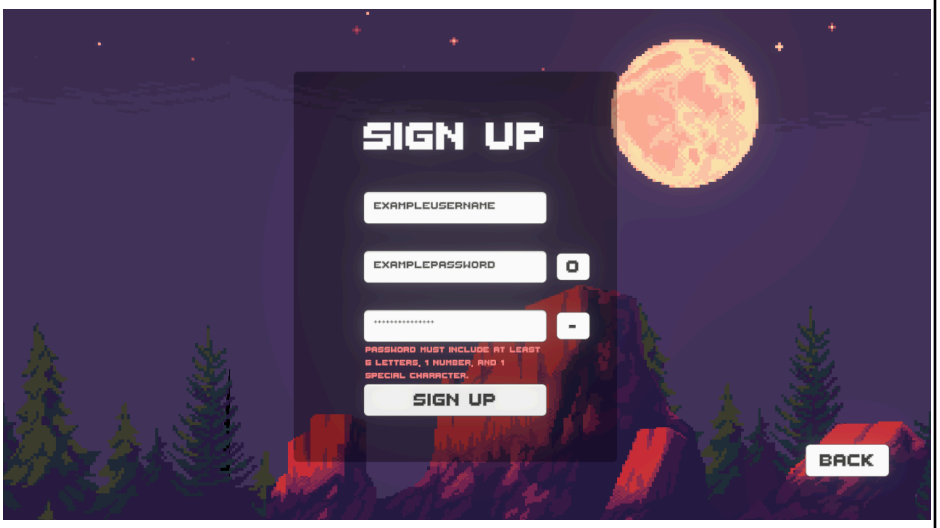
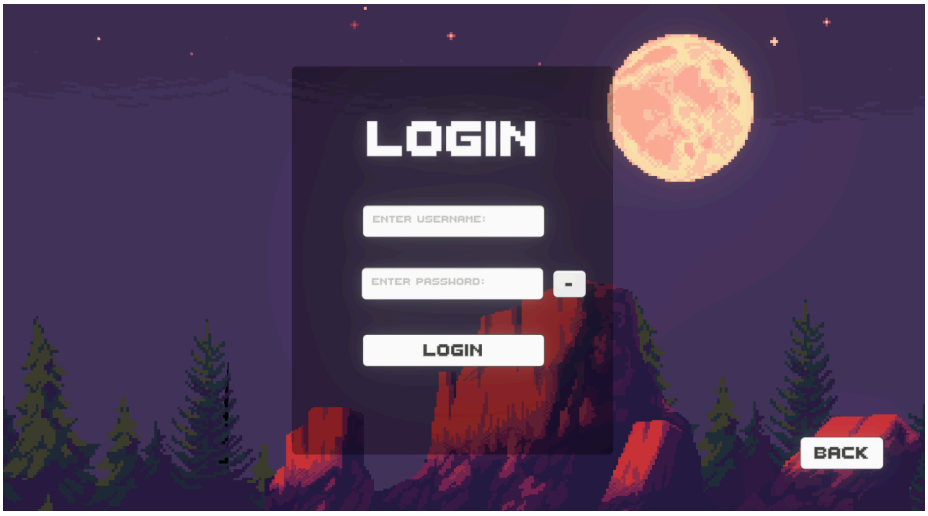
4.2 Key Features Developed

Describe and justify the core features.

- Login/signup system: The main menu includes login and signup functionality and encryption with password hashing, ensuring that user details remains secure. To comply with other secure software practices, passwords must be confirmed when signing up, display error messages when not in proper format, and have toggleable visibility
- Player movement system: This system allows for basic movement and jumping with the WASD and space keys or the arrow keys, depending on the user preference.
- Sign system: The signs in the game play a major role in actually teaching students about relevant physics concepts and providing context on physical reactions

4.3 Screenshots of Interface

Interfaces	Description
	The main menu interface enables users to log in to the game and sign up. Playing the game isn't enabled before first logging in.
	From this interface, the users are able to sign up using login and password credentials. To ensure that users don't mistakenly enter incorrect passwords, the sign-up interface provides a confirm password feature.

	The interface is also equipped with a toggle password visibility option, indicated by either the “o” or a “-”, as well as a password error system, where if the user does not create a password suiting the requirements, an error message appears, prompting the user to remake their password. The interface also has a back button, allowing users to return to the title page. User created passwords are hashed and stored in a secure database, ensuring malicious attackers are prevented from accessing user accounts or private information.
	Similarly to the sign-up interface, the login interface accommodates a back button, a username field and a password field. The password field has toggleable visibility to enhance privacy and ensure that software security is integrated into every stage of the sign-up process.

5. Testing and Evaluation

5.1 Testing Methods Used

Testing is the process of identifying and fixing bug issues to ensure that software functions as intended. In my project, I used a combination of unit testing, system testing, and user testing,

aligned with my Agile development approach. Each sprint concluded with testing specific features, enabling regular validation and improvement before moving forward.

I primarily used unit testing after each module (e.g. login system or movement controls) was implemented. Integration testing was also performed when combining these modules (e.g. login to play panel to scene changing) to ensure that they work together, and system testing was used near the end of development to verify the entire program.

5.2 Test Cases and Results

Test ID	Description	Expected Result	Actual Result	Pass/Fail
TC01	Login with valid user (Unit Test)	Success message	Success message	Pass
TC02	Invalid user login (Validation Test)	Error message	Error message	Pass
TC03	Sign-up with existing username (Validation)	Show "username already taken" message	Show "username already taken" message	Pass
TC04	Password visibility toggle (Unit Test)	Password field switches between masked/unmasked	Works as expected	Pass
TC05	Scene loads correctly after pressing Play (System Test)	Scene loads with fade transition	Fade transition completes and Level_1 loads	Pass
TC06	Wall sticking bug (before fix)	Player should slide off of walls	Clings to wall	Fail
TC07	Wall sticking bug fix (Unit Test / Validation)	Player should no longer cling to walls	Player no longer clings after friction adjustment	Pass

TC08	Integration: Sign-up to login flow	Player signs up and logs in successfully	Seamless transition to login, then play panel	Pass
TC09	Validation: Empty username during sign-up	Error: "Username and password required"	Error appears as expected	Pass
TC10	System: Fade transition during scene load	Black overlay fades in smoothly before scene loads	Overlay fade and scene transition work correctly	Pass

One test that failed was TC06, where the player could stick to walls due to high friction. To fix the bug, I lowered the material friction in Unity's physics settings. The result of this fix was correct player behaviour on wall contact.

5.3 Evaluation Against Requirements

My solution primarily meets the project goals, but still has a few limiting factors. Within the game, there is a secure login system, where passwords are hashed to protect user data, and other core gameplay features, such as player movement and jumping, have been implemented successfully. In addition, the user interface is clean and easy to understand, and the application loads quickly and runs without noticeable lag. However, the level of difficulty and structure could use some improvement, and beyond player data, currently, no other data is saved (e.g. level completion or times), which can lead to user dissatisfaction.

5.4 Improvements and Future Work

If I had more time, I might have reworked my game as a whole, but I would have mainly focused on expanding the game by adding more levels, as well as incorporating more physics-based content, such as utilising light and magnets. I would also have implemented player animations to make the game feel more dynamic, or added other features to the player movement, such as dashing or wall climbing. Data-saving could also use improvement, where level completion is not saved, and neither is the time taken to beat levels.

6. (Client) Feedback and Reflection

6.1 Summary of Client Feedback

Peer Feedback:

Name	Feedback
Andrew	Good game overall, but could maybe use more variety in the platforming and maybe needs more levels
Vincent	Smooth gameplay, maybe needs to be complex due to several existing platformers with a similar feel in gameplay

6.2 Personal Reflection

From this project, I mostly learned how to use Unity and how to structure my development process using Agile methodology. In particular, I learned the skills to operate Unity, animate, add effects, manipulate cameras, and incorporate sound effects, among other things. In addition, I learnt how to use Gantt charts to structure my workload and schedule to complete the project, which ties into Agile's methodology where developers do sprints, rather than working on the project as a whole. This helped me with my time management skills as well.

7. Appendices

Full Gantt Chart

	Week									
Task	1	2	3	4	5	6	7	8	9	10
Identifying Needs/Opportunities										
Researching Project Ideas										
Brainstorming level concepts										
Experimenting with UI layouts										

Implementing Player Mechanics										
Creating UI systems										
Creating Game Levels										
Implementing game systems										
Unit Testing										
Peer Feedback										
Bugfixing										
Writing documentation										
Reflection										

Complete Data Dictionary:

Name	Type	Description / Purpose
titlePanel	GameObject	UI panel containing title screen
loginPanel	GameObject	UI panel for login form
signupPanel	GameObject	UI panel for sign-up form
playPanel	GameObject	UI panel shown after login
infoPanel	GameObject	UI panel displaying information/help
openLoginButton	Button	Button to open login panel
openSignupButton	Button	Button to open sign-up panel

loginUsernameField	TMP_InputField	Field to enter login username
loginPasswordField	TMP_InputField	Field to enter login password
loginButton	Button	Confirms login attempt
loginBackButton	Button	Returns to title from login screen
loginToggleVisibilityButton	Button	Toggles visibility of login password
loginMessageText	TMP_Text	Shows login errors
loginSuccessMessageText	TMP_Text	Shows success message after sign-up
signupUsernameField	TMP_InputField	Field to enter new username
signupPasswordField	TMP_InputField	Field to enter new password
signupConfirmPasswordField	TMP_InputField	Field to confirm entered password
signupButton	Button	Submits sign-up form
signupBackButton	Button	Returns to title from sign-up
signupToggleVisibilityButton	Button	Toggles visibility of signup password

signupConfirmToggleVisibilityButton	Button	Toggles visibility of confirm password field
signupMessageText	TMP_Text	Shows sign-up form validation messages
playButton	Button	Starts the game
infoButton	Button	Opens info/help panel
infoBackButton	Button	Returns from info panel
fadeDuration	float	Time for UI fade-out effect
fadePauseDuration	float	Wait time after fade before loading scene
playPanelFadeOverlay	Image	Overlay image for screen fading
fadeableBackgroundSprites	List<SpriteRenderer>	Sprites to fade with UI
loginToggleIcon	TMP_Text	Icon shown on login password toggle
signupToggleIcon	TMP_Text	Icon for signup password toggle
signupConfirmToggleIcon	TMP_Text	Icon for confirm password toggle
showIcon	string	Icon for password visible state

hideIcon	string	Icon for password hidden state
savePath	string	Path to save user data file
userDatabase	Dictionary<string, string>	Stores usernames and hashed passwords
currentUser	string	Stores currently logged-in user
isLoginPasswordVisible	bool	Whether login password is visible
isSignupPasswordVisible	bool	Whether sign-up password is visible
isSignupConfirmPasswordVisible	bool	Whether confirm password is visible
OnLoginButtonPressed()	Method	Validates login credentials
OnSignupButtonPressed()	Method	Validates sign-up form and adds user to database
IsPasswordSecure(string)	Method	Checks password security (length, number, special char)
SaveUserDatabase()	Method	Saves current user database to file

LoadUserDatabase()	Method	Loads user database from file
FadeOutPlayPanelAndOverlayAndLoadScene(string)	Coroutine	Fades UI and loads game scene
HashPassword(string)	Method	Hashes password using SHA256
moveSpeed	float	Horizontal movement speed
jumpVelocity	float	Initial upward velocity on jump
gravity	float	Custom gravity applied manually
terminalVelocity	float	Maximum downward speed
groundCheck	Transform	Transform used to check if grounded
groundLayer	LayerMask	Layer mask to identify ground
groundCheckBoxSize	Vector2	Size of ground check area
jumpSound	AudioClip	Sound played on jump
landSound	AudioClip	Sound played on landing
walkSound	AudioClip	Sound played while walking

walkSoundInterval	float	Cooldown time between walking sounds
walkSoundVolume	float (0.0–1.0)	Volume of walking sound
horizontalInput	float	Player’s current movement input
isGrounded	bool	Whether player is grounded
Flip()	Method	Flips player sprite direction
ApplyMovement()	Method	Handles manual movement physics
HandleWalkingSound()	Method	Triggers walking sound with cooldown
Instance	VirtualCameraShake	Singleton reference
camTransform	Transform	Cached camera transform
initialLocalPosition	Vector3	Default local position of camera
shakeDuration	float	How long to shake for
shakeMagnitude	float	Strength of the shake
dampingSpeed	float	Speed to reduce shaking
ShakeCamera(float)	Method	Public method to trigger camera shake

Full Test Logs:

Test ID	Description	Test Type	Input/Action	Expected Result	Actual Result	Pass/Fail
TC01	Login with valid credentials	Unit Test	Username: testuser, Password: Test123!	Game progresses to Play Panel	Game progresses to Play Panel	Pass
TC02	Login with invalid credentials	Unit Test	Username: wronguser, Password: wrongpass	Error message shown	"Invalid username or password" displayed	Pass
TC03	Sign-up with weak password	Validation Test	Password: abc	Message: Password not secure	"Password must include at least 6 letters..."	Pass
TC04	Password and confirmatio n mismatch	Validation Test	Password: Test123!, Confirm: Test456!	Message: Passwords do not match	Error message appears correctly	Pass
TC05	Password already in use	Validation Test	Username: testuser	Message: Username is already taken	Works as expected	Pass
TC06	Toggle login password visibility	UI Test	Click visibility toggle on login screen	Password becomes visible	Icon changes and visibility toggles	Pass

TC07	Jump and land controls	System Test	Use Space/W while grounded	Character jumps and lands with sound	Sound plays on jump/land	Pass
TC08	Walk sound triggers correctly	Unit Test	Hold Left/Right on ground	Walk sound plays in intervals	Walk sound plays every 0.4s	Pass
TC09	Camera shake activates	Integration Test	Call ShakeCamera(0.2f) method during impact	Camera shakes briefly and stops	Shakes for expected duration	Pass
TC10	Fade-to-black scene transition	Integration Test	Press Play Button after login	UI fades and new scene loads	UI fades correctly and Level_1 loads	Pass
TC11	UI returns to Title from login	UI Navigation Test	Click "Back" from login/signup	Title Panel becomes active	Works as expected	Pass
TC12	Signup success message shown	UI Feedback Test	Sign up with valid data	"Sign up successful!" message shows on login screen	Message shown in green at bottom of login panel	

Exemplar Code Snippets:

Player Movement

```
private void Update()
{
   bool wasGrounded = isGrounded;
```

```
    HandleInput();         // Check input
    ApplyMovement();       // Apply movement physics
    UpdateAnimation();     // Update animation states

    // Landing sound when player touches ground again
    if (!wasGrounded && isGrounded && landSound)
    {
        audioSource.PlayOneShot(landSound);
    }

    walkSoundTimer -= Time.deltaTime;
    HandleWalkingSound(); // Optional walking sound
}

private void HandleInput()
{
    horizontalInput = Input.GetAxisRaw("Horizontal");
    isGrounded = Physics2D.OverlapBox(groundCheck.position, groundCheckBoxSize, 0f,
groundLayer);

    // Jump if grounded
    if ((Input.GetButtonDown("Jump") || Input.GetKeyDown(KeyCode.W) ||
Input.GetKeyDown(KeyCode.UpArrow)) && isGrounded)
    {
        rb.velocity = new Vector2(rb.velocity.x, jumpVelocity);
        if (jumpSound) audioSource.PlayOneShot(jumpSound);
    }
}
```

Password Hashing

```
// Hashes the password using SHA-256 before storing or comparing it.
string HashPassword(string password)
{
    using (SHA256 sha256Hash = SHA256.Create())
    {
        // Convert the input string to a byte array and compute the hash
        byte[] bytes = sha256Hash.ComputeHash(Encoding.UTF8.GetBytes(password));

        // Convert byte array to a hexadecimal string
        StringBuilder builder = new StringBuilder();
```

```
        foreach (byte b in bytes)
        {
            builder.Append(b.ToString("x2"));
        }

        return builder.ToString(); // Final hashed password
    }
}
```

Password Validation

```
bool IsPasswordSecure(string password)
{
    int letterCount = 0;          // Count of alphabetic characters
    bool hasDigit = false;        // Flag for at least one number
    bool hasSpecial = false;      // Flag for at least one special character

    foreach (char c in password)
    {
        if (char.IsLetter(c)) letterCount++;       // Count each letter
        else if (char.IsDigit(c)) hasDigit = true; // Check for digits
        else if (!char.IsWhiteSpace(c)) hasSpecial = true; // Check for special
characters
    }

    // Password is valid if it has:
    // - At least 6 letters
    // - At least one digit
    // - At least one special character
    return letterCount >= 6 && hasDigit && hasSpecial;
}
```