EcoFriend Project Submission and Documentation Report

Group Name: Tiny Muffins
Group Number: 17
Project Title: EcoFriend
Website Github URL: https://github.com/unit0113/EcoFriend
Website Github URL: https://github.com/lr2120/G17_Eco_Friend
Game File (backup):
https://drive.google.com/file/d/1ycJ_hqWT-LuBAN5F-CDdH2Tj_hsLYGch/view?usp=sharing
Team Members: Christian Carey, Kyle Lund, Lily Reyes, Ed Lang

Table of Contents:

Section 1

Project Description

Climate change ranks among today's most pressing challenges. While many are eager to make a difference, they often lack clear guidance. Although software products exist to gauge individual ecological footprints and the internet is full of varied advice, there is a need for a single software that seamlessly combines these elements. There's a need for an engaging platform that not only estimates a user's ecological footprint but also offers actionable steps in its reduction and fosters collaboration among peers and communities to collectively make a difference.

EcoFriend is an immersive web-app/game combination where we invite the user to participate in an enjoyable eco-educational experience. We wanted the user to come for the fun and captivating experience our website and game has to offer and stay for the educational resources we provide toward promoting an ecological lifestyle. To do this, we built an engaging platform focused on environmental awareness and sustainability. In this project documentation report, we'll walk you through our journey and highlight key aspects of our project.

In web development, our primary goal was to make users feel welcome in our digital space. We created features like Account Creation and User Login to ensure a secure and personalized user experience. Navigating through our Home Page is designed to be intuitive, offering users easy access to different sections. The Community Blog is a designated spot for users to learn on environmental topics. Although we initially planned ambitious features like Rewards and Recognition and the Community Art Gallery, we've temporarily put those on hold to focus on essential components.

Transitioning to the game, EcoFriend aims to deliver an engaging and educational experience. We've solidified our game design to lay the foundation for an immersive journey. We collaboratively designed the world environment and level design to both invite the user in and promote an educational experience. Prototyping was an important part of our process, giving shape to Player and NPC characters and offering a tangible preview with the Level 1 Prototype. Art assets were carefully crafted to enhance the overall aesthetic appeal.

How Solution Addressed Challenge Statement

In tackling the challenge at hand, our approach involved the creation of a web app tailored for individuals ranging from curious to those who are committed to reducing their ecological footprint. This application not only simplifies learning about one's environmental impact but also introduces actions that reinforce eco-friendly habits in the gaming element. While the internet offers various informative tools and blogs focused on environmental contributions, our app stands out by infusing a gamified dimension into the contribution experience. Users can form teams with friends, embarking on environmentally beneficial experiences together. The gamified system, complete with digital points, serves as a motivational force, encouraging friendly competition and nurturing a sense of belonging and shared eco-responsibility. Furthermore, our platform places a premium on user security, employing strong user authentication measures to safeguard data security. Ultimately, our vision is to inspire widespread and active engagement in the collective effort to preserve our environment.

Features and Functionality

Website:
The project website allows users to perform all the standard account management actions that we have come to expect. Users can both create new accounts and sign in to an existing account with an email address or a password. Alternatively, they can also create their account via their google or github credentials. Our users are also able to change their username, update their password, and when they no longer intend to use this product, can delete their account. Once logged into the website, users can either play the game or interact with the other provided features. These features include the teams system, where users can create and join teams with their friends, the community blog, where curated environmental news is provided to the users, and the about page, to learn more about the goals of this project and the people behind it.

Game:
"EcoAdventure" is a 2D top-down game, accessible directly from our website, tailored for casual gamers to enjoy in brief 5 to 10 minute sessions. Upon launching the game, players are greeted with a starter screen offering a succinct, illustrative tutorial. This tutorial outlines the game's primary objective: collecting seeds and water to plant and nurture a tree. The interface presents two straightforward options: 'Start Game' and 'Quit Game'.

The game unfolds across two distinct levels, each featuring a Heads-Up Display (HUD) that tracks the player's score and a timer. Players navigate the game world using arrow keys, engaging with the environment to achieve their objectives. Upon encountering a seed, the player can collect it by pressing the spacebar. The next steps involve deciding whether to plant the seed immediately or gather water first. Once the seed is planted and watered, the player earns a point, symbolizing a successfully cultivated tree.

The second level introduces a heightened challenge: a squirrel character that pilfers seeds, reducing the number available for gathering. This added obstacle increases the game's difficulty, requiring more strategic play. The level concludes once all accessible seeds are gathered. At the end of the game, players receive their final score and time, culminating their eco-adventure.

System Models

Architectural Pattern - Website

In the creation of our website, we utilize key principles that define our goal of deploying an optimized application ensuring a seamless user experience. First, we utilized a component-based architectural approach for our web pages, emphasizing each module's distinct yet interconnected nature. This methodology was grounded in core design principles such as reusability, replaceability, encapsulation, and autonomy. We enhanced development efficiency by segmenting the website's functional and logical elements into refined abstractions. Our website's structure is a consolidation of modular elements - including grids, buttons, events, headers, layouts, and containers - each crafted for repeated use. The modular design pattern streamlined systemic updates and alterations and greatly simplified the design process. Isolated components are also beneficial in testing and debugging, allowing for precise identification of erroneous code and quick deployment of solutions. The components-based approach encouraged collaboration among our team, enabling developers to work concurrently on the website while maintaining a cohesive design vision without any overlaps or conflicts. Adopting component-based architectural patterns streamlined our development process and established a robust framework that promoted teamwork and innovation.

We opted for a monolithic architecture for our web application, which includes a client-side UI with dynamic client-side rendering, a MongoDB database, and our controller handling functionality. This monolithic approach was selected primarily for its efficiency in development and deployment. Being part of a small team with strict deadlines and considering that our project is intended for a limited number of clients, our primary focus has been rapid deployment and testing. This approach also conveniently allows us to avoid the complexities of scaling a monolithic system. Our system's independent functions served as a robust testing tool. Our goal has been to create a stable, functional product quickly and efficiently, and the monolithic architecture aligns perfectly with this objective.

The model-view-controller pattern was used to organize interactions between the client and server. Each client has their own view of the website, allowing for a dynamic and seamless user experience. MongoDB's schemas, acting as the system data and business logic, was our model. We employed client-server communication through GET, POST, and DELETE instructions to pass and retrieve JSON objects to and from our database. Finally, API routes through the Next.js framework were our controller, acting as an intermediary between the client-side view and the MongoDB model. The controller handles client requests and sends them to the model. We had a web server that communicated with clients, an application server that provided unique information to each client, and a database server to manage that communication. Utilizing an MVC architectural pattern supported unique views for our clients in seeing their level, awards, and team affiliation.

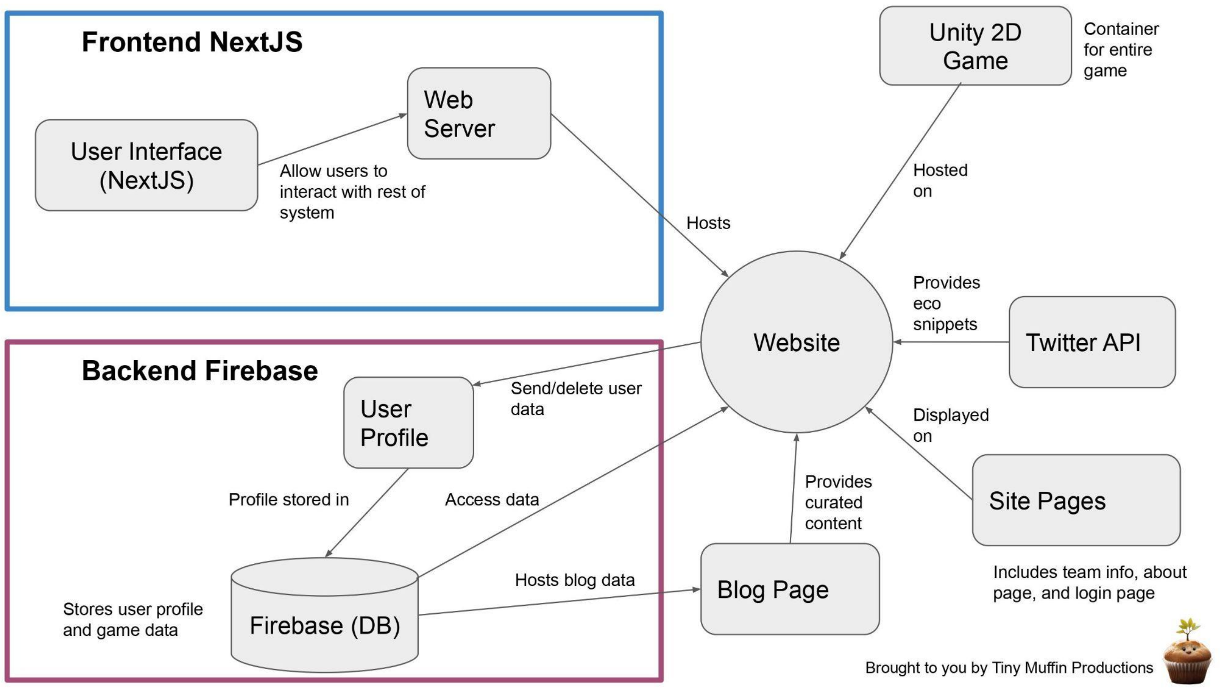
Architectural Pattern - Game

"EcoAdventure" is built on Unity 2020.3, leveraging its robust component-based architecture. This architectural pattern is inherently modular, promoting flexibility and ease of development. In Unity, game objects are containers to which various components are attached, each component giving the object specific functionalities or behaviors.

In the development of "EcoAdventure," this approach was central. Each game element, be it a player character, environmental tile object, or UI element, is a game object within Unity. To these objects, we attached custom scripts - the core components that define their behavior and interactions within the game world. These scripts were written in C#, Unity's primary programming language, and encapsulated functionalities ranging from player movement and item collection to game state management and UI control.

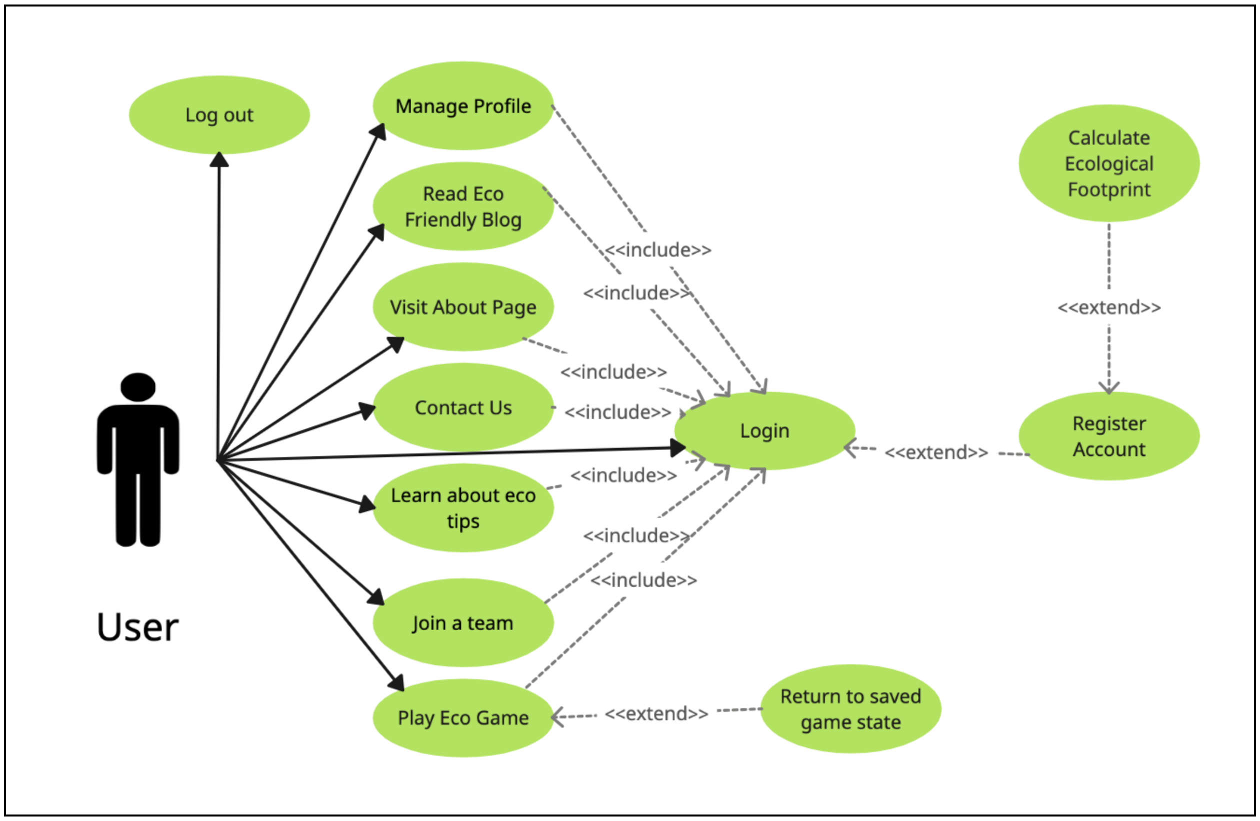
This architecture's strength is its modularity and reusability. By isolating functionalities into distinct scripts, we could easily tweak individual aspects of the game without affecting others. For example, altering the behavior of the squirrel in the second level required changes only to the squirrel’s behavior script, leaving other components unaffected.

System Context Model



Use Case Model

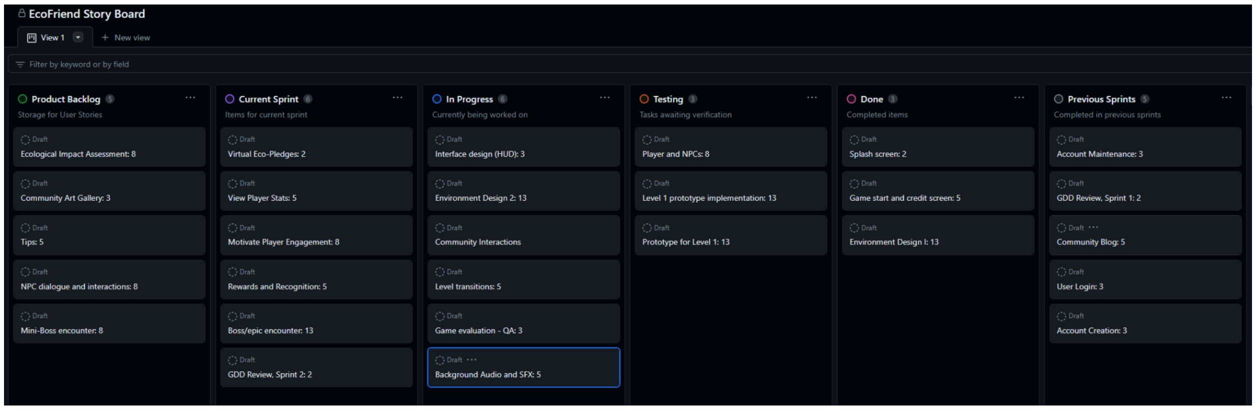
The use case model is the blueprint for our EcoFriend website, detailing the ways in which users will interact with the system. The use case model includes some features which aren't currently implemented, such as the Ecological Footprint Calculator and the Return to Saved Game State features. Each use case represents a specific functionality or feature within the app. From the initial account creation to the collaborative efforts, the use case model outlines the step-by-step processes that users will follow.


Log out
Manage Profile
Read Eco Friendly Blog
Visit About Page
Contact Us
Learn about eco tips
Join a team
Play Eco Game
User
Login
<<include>>
<<include>>
<<include>>
<<include>>
<<include>>
<<include>>
<<include>>
Calculate Ecological Footprint
<<extend>>
Register Account
<<extend>>
Return to saved game state
<<extend>>

Section 2

Code Management

In our approach to code management, we extensively utilized GitHub as our version control system. Our repositories were organized with separate ones dedicated to the website and game code, facilitating a more focused and efficient development process. Collaboratively, the team managed branches judiciously, ensuring that each branch served a specific purpose and aligning with our development workflow. Code reviews were a pivotal aspect of our collaboration, providing an opportunity for thorough examination and feedback exchange before merging. Our version control best practices were exemplified through the incorporation of two essential test applications within the website repository. Firstly, Vercel was employed to validate that any code submitted in a pull request would function seamlessly in a live deployment environment. Concurrently, Sonarcloud played a crucial role by conducting static code analysis, identifying potential bugs and code smells. These applications allowed us to impose stringent requirements on pull requests, maintaining a consistently high level of code quality throughout the development lifecycle. Additionally, we used a GitHub StoryBoard to track project progress, keeping the entire team in sync with the evolving project landscape.



EcoFriend Story Board

Product Backlog 5	Current Sprint 6	In Progress 6	Testing 3	Done 3	Previous Sprints 5
Storage for User Stories	Items for current sprint	Currently being worked on	Tasks awaiting verification	Completed items	Completed in previous sprints
Draft Ecological Impact Assessment: 8	Draft Virtual Eco-Pledges: 2	Draft Interface design (HUD): 3	Draft Player and NPCs: 8	Draft Splash screen: 2	Draft Account Maintenance: 3
Draft Community Art Gallery: 3	Draft View Player Stats: 5	Draft Environment Design 2: 13	Draft Level 1 prototype implementation: 13	Draft Game start and credit screen: 5	Draft GDD Review, Sprint 1: 2
Draft Tips: 5	Draft Motivate Player Engagement: 8	Draft Community Interactions	Draft Prototype for Level 1: 13	Draft Environment Design I: 13	Draft Community Blog: 5
Draft NPC dialogue and interactions: 8	Draft Rewards and Recognition: 5	Draft Level transitions: 5			Draft User Login: 3
Draft Mini-Boss encounter: 8	Draft Boss/epic encounter: 13	Draft Game evaluation - QA: 3			Draft Account Creation: 3
	Draft GDD Review, Sprint 2: 2	Draft Background Audio and SFX: 5			

Test Plan

The project’s test plan was created collaboratively in Google Sheets. It features a comprehensive set of tests and test suites. The test plan outlines various test cases covering account management, community features, game integration, in-game functionality, health resources, and more. The goal of each test is to ensure the integrity of a function, feature, or component. Each test case includes preconditions, test steps, test data, expected results, postconditions, and space for actual results:

ITEM NUMBER	TEST SUITE	TEST CASE ID	TEST CASE DESCRIPTION	PRECONDITION	TEST STEPS	TEST DATA	EXPECTED RESULT	POSTCONDITION	ACTUAL RESULT
1	acnt_mgt	account_creation	Verify account creation works	unique email (email not in system)	1. Fill out account with unique email and password 2. Click create	unique email should be generated for each test	Account creation is sucessful.	Account is [unique_email]@[any].com	
2	acnt_mgt	account_creation_google	Verify account can be created with google account	User has google account	1. Create google account 2. Create account with google account 3. Verify account has been created	Email: EcofriendsTest@gmail.com	Account creation is sucessful.	Account is EcofriendsTest@gmail.com	
3	acnt_mgt	account_creation_github	Verify account can be created with github account	User has github account	1. Create github account 2. Create account with github account 3. Verify account has been created	Testers github account	Account creation is sucessful.	Account is testers github account	
4	acnt_mgt	account_creation_duplicate_account	Verify account creation does not work when an associated email is already in use	non-unique email (email in system)	1. Fill out account with non-unique email and pasword 2. Click create	Email: EcofriendsTest@gmail.com Password: password	Account creation is unsucessful.	A duplicate account of EcofriendsTest@gmail.com does not exist	
5	acnt_mgt	verify_login_correct	Verify login works with correct email and password	User has a valid account (email and password)	1. Enter valid email 2. Enter valid password 3. Click Login	Email: EcofriendsTest@gmail.com Password: password	Login is successful.	Account is set to EcofriendsTest.	
6	acnt_mgt	verify_login_google	Verify login works with previously used google account	User has google account	1. Sign in with google	Email: EcofriendsTest@gmail.com	Login is successful.	Account is EcofriendsTest@gmail.com	
7	acnt_mgt	verify_login_github	Verify login works with previously used github account	User has github account	1. Sign in with github	Testers github account	Login is successful.	Account is testers github account	
8	acnt_mgt	verify_login_incorrect_email	Verify login does not work without correct email	User has an invalid account (wrong email)	1. Enter incorrect email 2. Enter any password 3. Click login	Email: invalidemailbad@gmail.com Password: password	Login is unsucessful.	No account is logged in.	
9	acnt_mgt	verify_login_incorrect_password	Verify login does not work without correct password	User has a valid account.	1. Enter correct email 2. Enter wrong password 3. Click login	Email: EcofriendsTest@gmail.com Password: wrongpassword	Login is unsucessful.	No account is logged in.	

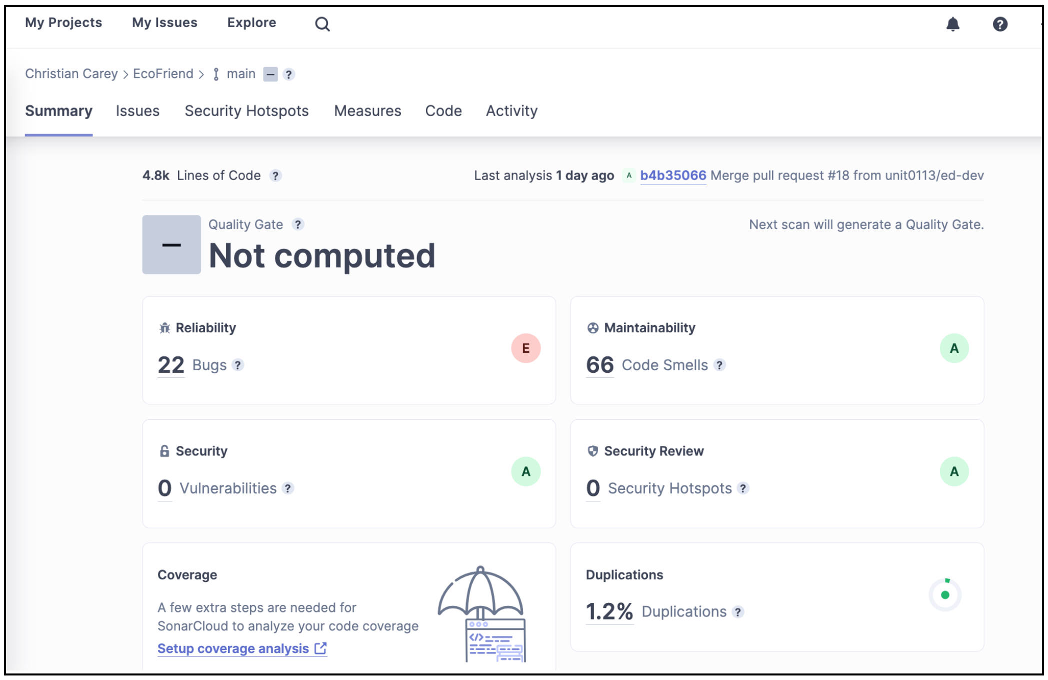
10	acnt_mgt	account_maintenance_deletion	Verify account deletion works	Must be logged into an account.	1. Navigate to the account maintenance settings. 2. Find the option to delete the account. 3. Confirm the account deletion by entering the password. 4. Click the "Delete Account" button.	Account password for confirmation.	Account deletion is successful. User cannot login with previous credentials.	Account is deleted.	
11	community	community_blog	Verify that the community blog page loads and allows users to view blog posts	Navigate to community_blog page	1. Navigate to the community blog page. 2. Observe the loading of blog posts.	N/A	The community blog page loads and displays blog posts.	Users can view blog posts.	
12	community	community_art_gallery	Verify page loads and populates	User is logged into the system.	1. Navigate to the community art gallery page. 2. Observe the loading of art and content.	N/A	The community art gallery page loads and populates with art and content.	Users can view art and content in the gallery.	
13	community	virtual_eco_pledges	Verify that users can view and take virtual eco-pledges to support environmental causes.	User is logged into the system.	1. Navigate to the virtual eco-pledges section. 2. Browse available eco-pledges. 3. Select an eco-pledge and complete the process to support an environmental cause.	Eco-pledges available for selection.	Users can view and take virtual eco-pledges to support environmental causes.	The selected eco-pledge is recorded as supported.	
14	integration	game_integration	Verify game loads and functions in website	User is logged into an account	1. Navigate to game page 2. Verify page loads 3. Verify game loads and plays	Game page URL, account login	The game loads and functions properly on the website.		
15	in-game	player_and_npc	Verify that players can interact with NPCs in the game	Player collides/triggers NPC.	1. Interact with non-player characters (NPCs) in the game. 2. Observe player-NPC interactions and in-game actions.	In-game actions and interactions with NPCs.	Events are executed when player interact with an NPC in the game.	Expected interaction with NPC.	

16	in-game	interface_design_hud	Verify that the in-game Heads-Up Display (HUD) elements, including health bars and game information, are displayed correctly	Play is clicked on game build.	1. Play the game and observe the Heads-Up Display (HUD) elements. 2. Ensure that health bars and game information are displayed correctly. 3. Ensure HUD information updates on state changes.	In-game HUD elements.	The in-game HUD elements, including health bars and game information, are displayed and updated correctly.	Players can easily access relevant game information through the HUD.	
17	in-game	game_evaluation_qa	Verify that the game is thoroughly evaluated for quality assurance, including functionality, performance, and bug testing	Game is in alpha build so all essential elements are in the game.	Conduct thorough evaluation, including functionality, performance, and bug testing.	Entire play though of game.	Identify and document issues, ensure functionality, and evaluate game performance.	A detailed evaluation report is generated for further improvements.	
18	in-game	rewards_and_recognition_functionality	Verify that players receive rewards and recognition for achieving in-game milestones and objectives	Players have achieved in-game milestones and objectives.	Observe the process of receiving rewards and recognition for achievements.	Achieved in-game milestones and objectives.	Players receive rewards and recognition for achieving in-game milestones and objectives.	Players' achievements are rewarded and recognized within the game.	
19	hlth_rsrcs	health_increase_validation	Verify that the player's (tree's) health increases when it collects water and water GameObject is destroyed.	The player collects water while not at maximum health.	1. Locate the water in the game. 2. Move the player to the water's position. 3. Collect the water.	Player's health increases by 1 and does not exceed maximum upon collecting water.	Water object should disappear and the player's health should reflect the increase.	Pre-condition water is not in game anymore.	
20	hlth_rsrcs	health_decrease_validation_mistletoe	Confirm that the player sustains damage when traveling over mistletoe.	The player is active in the game with some but not full health (1).	1. Locate mistletoe on the ground. 2. Move the player over the mistletoe.	Initial health (at least 1), amount of health decrease expected (1).	The player's health decreases by 1 after moving over mistletoe.	Player's health is updated, and the mistletoe remains in place for future damage.	
21	hlth_rsrcs	health_decrease_validation_caterpillar	Verify that the player's health decreases after colliding with a gypsy moth caterpillar.	The player is active in the game with some but not full health (1).	1. Locate mistletoe on the ground. 2. Move the player over the caterpillar.	Initial health (at least 1), amount of health decrease expected (1).	The player's health decreases by 1 after moving over mistletoe.	Player's health is updated, and the caterpillar remains in place for future damage.	
22	hlth_rsrcs	seed_collection	Ensure that the player can collect scattered seeds.	Seeds are scattered on the ground, and the player is near a seed.	1. Identify a seed on the ground. 2. Move the player to the seed's location to collect it.	Number of seeds before collection (0).	The player has one more seed in the inventory after collection (inventory = 1).	Seed disappears from the ground and is added to the player's inventory.	
23	hlth_rsrcs	tossing_seeds	Confirm that the player can toss seeds to birds, and the birds can get the seeds and fly away.	The player has at least one seed in the inventory.	1. Spot a bird in the game. 2. Aim and throw a seed at the bird.	Seeds in inventory (1), birds in range (1).	The bird takes the seed and flies off to germinate it.	Seed count decreases by one, and the bird flies away.	

Static Code Analysis and Report

We started with 22 bugs and 66 code smells when we ran the Sonarcloud report initially.

Part 1


My Projects My Issues Explore

Christian Carey > EcoFriend > main

Summary Issues Security Hotspots Measures Code Activity

4.8k Lines of Code

Last analysis 1 day ago A b4b35066 Merge pull request #18 from unit0113/ed-dev

Quality Gate
Not computed
Next scan will generate a Quality Gate.

Reliability: 22 Bugs (E)

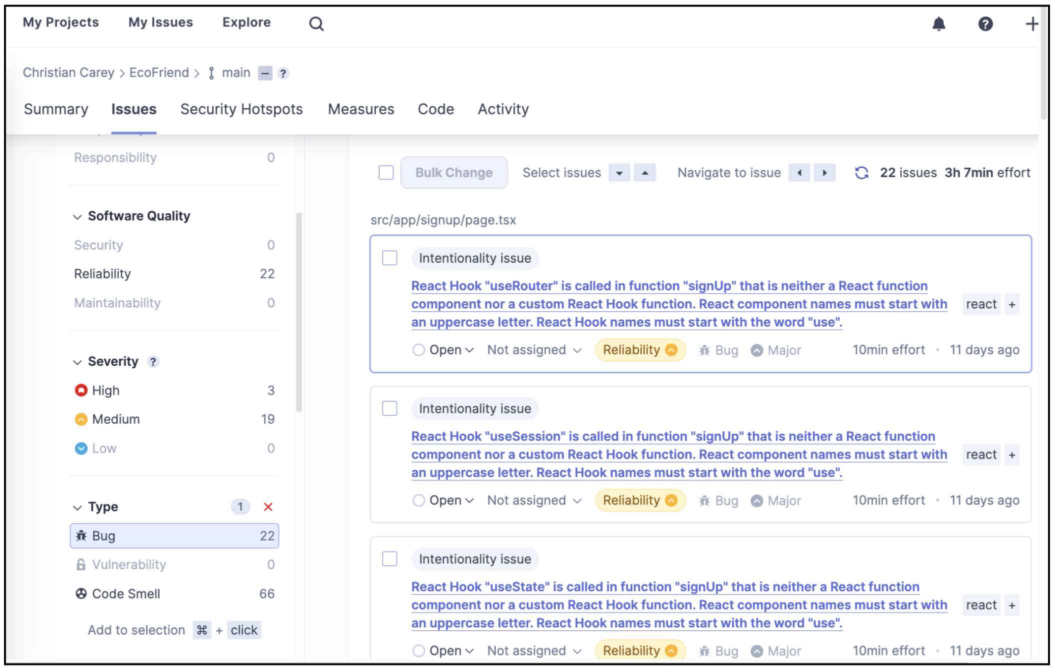
Maintainability: 66 Code Smells (A)

Security: 0 Vulnerabilities (A)

Security Review: 0 Security Hotspots (A)

Coverage: A few extra steps are needed for SonarCloud to analyze your code coverage. Setup coverage analysis

Duplications: 1.2% Duplications


My Projects My Issues Explore

Christian Carey > EcoFriend > main

Summary Issues Security Hotspots Measures Code Activity

Responsibility	0
Software Quality	
Security	0
Reliability	22
Maintainability	0
Severity	
High	3
Medium	19
Low	0
Type	1
Bug	22
Vulnerability	0
Code Smell	66

Add to selection ⌘ + click

Bulk Change Select issues Navigate to issue 22 issues 3h 7min effort

src/app/signup/page.tsx

Intentionality issue
React Hook "useRouter" is called in function "signUp" that is neither a React function component nor a custom React Hook function. React component names must start with an uppercase letter. React Hook names must start with the word "use". react
Open Not assigned Reliability Bug Major 10min effort · 11 days ago

Intentionality issue
React Hook "useSession" is called in function "signUp" that is neither a React function component nor a custom React Hook function. React component names must start with an uppercase letter. React Hook names must start with the word "use". react
Open Not assigned Reliability Bug Major 10min effort · 11 days ago

Intentionality issue
React Hook "useState" is called in function "signUp" that is neither a React function component nor a custom React Hook function. React component names must start with an uppercase letter. React Hook names must start with the word "use". react
Open Not assigned Reliability Bug Major 10min effort · 11 days ago

https://sonarcloud.io/project/issues?id=unit0113_EcoFriend&open=AYv3_5qvUe7tEvUpUXjm

Here is a description of the 88 issues (code smells and bugs) that Sonarcloud found and a description of how we will resolve them:
src/.../about/page.tsx
The element ul has an implicit role of list. Defining this explicitly is redundant and should be avoided.
Planned Smell Resolution: Remove explicit role of list

src/app/blog/wrapper.tsx
Mark the props of the component as read-only.
Planned Smell Resolution: Use typescript's read-only utility type to make the component prop

Imported JSX component wrapper must be in PascalCase
Planned Smell Resolution: Refactor component name into PascalCase

src/app/layout.tsx
Mark the props of the component as read-only.
Planned Smell Resolution: Use typescript's read-only utility type to make the component prop

src/.../learn/page.tsx
Mark the props of the component as read-only.
Planned Smell Resolution: Use typescript's read-only utility type to make the component prop

Ambiguous spacing before next element strong
Planned Smell Resolution: Insert an explicit JSX space as a string expression {' '}

Ambiguous spacing after previous element strong
Planned Smell Resolution: Insert an explicit JSX space as a string expression {' '}

Ambiguous spacing before next element strong
Planned Smell Resolution: Insert an explicit JSX space as a string expression {' '}

src/.../login/page.tsx
A fragment with only one child is redundant.
Planned Smell Resolution: Remove redundant fragment

src/app/page.tsx
Mark the props of the component as read-only.

Planned Smell Resolution: Use typescript's read-only utility type to make the component prop

src/.../page.tsx
A fragment with only one child is redundant.
Planned Smell Resolution: Remove the redundant fragment

src/.../page.tsx
A fragment with only one child is redundant.
Planned Smell Resolution: Remove the redundant fragment

src/.../page.tsx
A fragment with only one child is redundant.
Planned Smell Resolution: Remove the redundant fragment

src/.../page.tsx
A fragment with only one child is redundant.
Planned Smell Resolution: Remove the redundant fragment

src/.../profile/page.tsx
A fragment with only one child is redundant.
Planned Smell Resolution: Remove the redundant fragment

src/.../signup/page.tsx
React Hook "useRouter" is called in function "signUp" that is neither a React function component nor a custom React Hook function. React component names must start with an uppercase letter. React Hook names must start with the word "use".
Planned Bug Resolution: Rename signUp React Function to SignUp React Function

React Hook "useSession" is called in function "signUp" that is neither a React function component nor a custom React Hook function. React component names must start with an uppercase letter. React Hook names must start with the word "use".
Planned Bug Resolution: Rename signUp React Function to SignUp React Function

React Hook "useState" is called in function "signUp" that is neither a React function component nor a custom React Hook function. React component names must start with an uppercase letter. React Hook names must start with the word "use".
Planned Bug Resolution: Rename signUp React Function to SignUp React Function

React Hook "useState" is called in function "signUp" that is neither a React function component nor a custom React Hook function. React component names must start with an uppercase letter. React Hook names must start with the word "use".
Planned Bug Resolution: Rename signUp React Function to SignUp React Function

React Hook "useState" is called in function "signUp" that is neither a React function component nor a custom React Hook function. React component names must start with an uppercase letter. React Hook names must start with the word "use".
Planned Bug Resolution: Rename signUp React Function to SignUp React Function

React Hook "useState" is called in function "signUp" that is neither a React function component nor a custom React Hook function. React component names must start with an uppercase letter. React Hook names must start with the word "use".
Planned Bug Resolution: Rename signUp React Function to SignUp React Function

React Hook "useState" is called in function "signUp" that is neither a React function component nor a custom React Hook function. React component names must start with an uppercase letter. React Hook names must start with the word "use".
Planned Bug Resolution: Rename signUp React Function to SignUp React Function

React Hook "useState" is called in function "signUp" that is neither a React function component nor a custom React Hook function. React component names must start with an uppercase letter. React Hook names must start with the word "use".
Planned Bug Resolution: Rename signUp React Function to SignUp React Function

Remove this useless assignment to variable "setEmailError".
Planned Smell Resolution: Remove the redundant assignment to variable “setEmailError”

React Hook "useState" is called in function "signUp" that is neither a React function component nor a custom React Hook function. React component names must start with an uppercase letter. React Hook names must start with the word "use".
Planned Bug Resolution: Rename signUp React Function to SignUp React Function

Remove this useless assignment to variable "setPwSucksError".

Planned Smell Resolution: Remove the redundant assignment to variable "setPwSucksError"

Prefer using an optional chain expression instead, as it's more concise and easier to read.
Planned Smell Resolution: Replace with ?. optional chaining the logical expression that checks for null/undefined before accessing the property of an object.

A fragment with only one child is redundant.
Planned Smell Resolution: Remove the redundant fragment

src/.../teams/page.tsx
The element ul has an implicit role of list. Defining this explicitly is redundant and should be avoided.
Planned Smell Resolution: Remove explicit role of list

Missing "key" prop for element in iterator
Planned Smell Resolution: Use a string or a number that uniquely identifies the list item. The key must be unique among its siblings, not globally

Missing "key" prop for element in iterator
Planned Smell Resolution: Use a string or a number that uniquely identifies the list item. The key must be unique among its siblings, not globally

src/.../AddPasswordSidebar.tsx
Remove this useless assignment to variable "setPwSucksError".
Planned Smell Resolution: Remove the redundant assignment to variable "setPwSucksError"

src/.../Blockquote.tsx
Mark the props of the component as read-only.
Planned Smell Resolution: Use typescript's read-only utility type to make the component prop

Mark the props of the component as read-only.
Planned Smell Resolution: Use typescript's read-only utility type to make the component prop

Consider removing 'undefined' type or '?' specifier, one of them is redundant.
Planned Smell Resolution: Remove either undefined type or ? specifier

src/.../Button.tsx
Consider removing 'undefined' type or '?' specifier, one of them is redundant.
Planned Smell Resolution: Remove either undefined type or ? specifier

src/.../ChangePasswordSidebar.tsx
Prefer using an optional chain expression instead, as it's more concise and easier to read.
Planned Smell Resolution: Replace with ?. optional chaining the logical expression that checks for null/undefined before accessing the property of an object.

Remove this useless assignment to variable "setPwSucksError".
Planned Smell Resolution: Remove the redundant assignment to variable “setPwSucksError”

React Hook "useState" is called conditionally. React Hooks must be called in the exact same order in every component render. Did you accidentally call a React Hook after an early return?
Planned Bug Resolution: Move hook call before conditional

React Hook "useState" is called conditionally. React Hooks must be called in the exact same order in every component render. Did you accidentally call a React Hook after an early return?
Planned Bug Resolution: Move hook call before conditional

React Hook "useState" is called conditionally. React Hooks must be called in the exact same order in every component render. Did you accidentally call a React Hook after an early return?
Planned Bug Resolution: Move hook call before conditional

React Hook "useState" is called conditionally. React Hooks must be called in the exact same order in every component render. Did you accidentally call a React Hook after an early return?
Planned Bug Resolution: Move hook call before conditional

React Hook "useState" is called conditionally. React Hooks must be called in the exact same order in every component render. Did you accidentally call a React Hook after an early return?
Planned Bug Resolution: Move hook call before conditional

React Hook "useState" is called conditionally. React Hooks must be called in the exact same order in every component render. Did you accidentally call a React Hook after an early return?
Planned Bug Resolution: Move hook call before conditional

src/.../ChangeUsernameSidebar.tsx
Prefer using an optional chain expression instead, as it's more concise and easier to read.
Planned Smell Resolution: Replace with ?. optional chaining the logical expression that checks for null/undefined before accessing the property of an object.

useState call is not destructured into value + setter pair
Planned Smell Resolution: Restructure to [newUsername, setNewUsername]

React Hook "useState" is called conditionally. React Hooks must be called in the exact same order in every component render. Did you accidentally call a React Hook after an early return?
Planned Bug Resolution: Move hook call before conditional

React Hook "useState" is called conditionally. React Hooks must be called in the exact same order in every component render. Did you accidentally call a React Hook after an early return?
Planned Bug Resolution: Move hook call before conditional

src/.../Footer.tsx
The element ul has an implicit role of list. Defining this explicitly is redundant and should be avoided.
Planned Smell Resolution: Remove explicit role of list

Do not use Array index in keys
Planned Smell Resolution: Use a string or a number that uniquely identifies the list item. The key must be unique among its siblings, not globally.

The element ul has an implicit role of list. Defining this explicitly is redundant and should be avoided.

Planned Smell Resolution: Remove explicit role of list

Do not use Array index in keys
Planned Smell Resolution: Use a string or a number that uniquely identifies the list item. The key must be unique among its siblings, not globally.

src/.../GridList.tsx
Mark the props of the component as read-only.
Planned Smell Resolution: Use typescript's read-only utility type to make the component prop

The element ul has an implicit role of list. Defining this explicitly is redundant and should be avoided.
Planned Smell Resolution: Remove explicit role of list

Mark the props of the component as read-only.
Planned Smell Resolution: Use typescript's read-only utility type to make the component prop

src/.../GridPattern.tsx
Invalid property 'fill' found on tag 'pattern', but it is only allowed on: altGlyph, circle, ellipse, g, line, marker, mask, path, polygon, polyline, rect, svg, symbol, text, textPath, tref, tspan, use, animate, animateColor, animateMotion, animateTransform, set
Planned Smell Resolution: Remove fill property

src/.../Hero.tsx
Remove this unused import of 'ChevronRightIcon'.
Planned Smell Resolution: Remove the redundant import of "ChevronRightIcon"

src/.../Impact.tsx
Mark the props of the component as read-only.
Planned Smell Resolution: Use typescript's read-only utility type to make the component prop

The element ul has an implicit role of list. Defining this explicitly is redundant and should be avoided.
Planned Smell Resolution: Remove explicit role of list

src/.../List.tsx
Mark the props of the component as read-only.

Planned Smell Resolution: Use typescript’s read-only utility type to make the component prop

The element ul has an implicit role of list. Defining this explicitly is redundant and should be avoided.
Planned Smell Resolution: Remove explicit role of list

Mark the props of the component as read-only.
Planned Smell Resolution: Use typescript’s read-only utility type to make the component prop

src/.../LoginBtnHeader.tsx
A fragment with only one child is redundant.
Planned Smell Resolution: Remove the redundant fragment

A fragment with only one child is redundant.
Planned Smell Resolution: Remove the redundant fragment

src/.../PageIntro.tsx
Mark the props of the component as read-only.
Planned Smell Resolution: Use typescript’s read-only utility type to make the component prop

src/.../PageLinks.tsx
Mark the props of the component as read-only.
Planned Smell Resolution: Use typescript’s read-only utility type to make the component prop

Mark the props of the component as read-only.
Planned Smell Resolution: Use typescript’s read-only utility type to make the component prop

src/.../RootLayout.tsx
Mark the props of the component as read-only.
Planned Smell Resolution: Use typescript’s read-only utility type to make the component prop

Mark the props of the component as read-only.

Planned Smell Resolution: Use typescript's read-only utility type to make the component prop

Mark the props of the component as read-only.
Planned Smell Resolution: Use typescript's read-only utility type to make the component prop

Mark the props of the component as read-only.
Planned Smell Resolution: Use typescript's read-only utility type to make the component prop

Mark the props of the component as read-only.
Planned Smell Resolution: Use typescript's read-only utility type to make the component prop

The object passed as the value prop to the Context provider changes every render. To fix this consider wrapping it in a useMemo hook.
Planned Smell Resolution: Wrap the values in a useMemo hook. Use the useCallback() hook if the value is a function.

src/.../SocialMedia.tsx
Mark the props of the component as read-only.
Planned Smell Resolution: Use typescript's read-only utility type to make the component prop

The element ul has an implicit role of list. Defining this explicitly is redundant and should be avoided.
Planned Smell Resolution: Remove explicit role of list

src/.../StatList.tsx
Mark the props of the component as read-only.
Planned Smell Resolution: Use typescript's read-only utility type to make the component prop

src/.../TagList.tsx
Mark the props of the component as read-only.
Planned Smell Resolution: Use typescript's read-only utility type to make the component prop

The element ul has an implicit role of list. Defining this explicitly is redundant and should be avoided.
Planned Smell Resolution: Remove explicit role of list

Mark the props of the component as read-only.
Planned Smell Resolution: Use typescript's read-only utility type to make the component prop

src/.../TeamActivityFeed.tsx
The element ul has an implicit role of list. Defining this explicitly is redundant and should be avoided.
Planned Smell Resolution: Remove explicit role of list

A fragment with only one child is redundant.
Planned Smell Resolution: Remove the redundant fragment

src/.../Testimonial.tsx
Mark the props of the component as read-only.
Planned Smell Resolution: Use typescript's read-only utility type to make the component prop

src/.../api/auth/[...nextauth].ts
Refactor this function to not always return the same value.
Planned Smell Resolution: Refactor return statements to reduce redundancy

src/styles/typography.css
Unexpected unknown unit "xl"
Planned Bug Resolution: False positive, Sonarcloud fails to identify tailwind CSS class

Unexpected unknown unit "xl"
Planned Bug Resolution: False positive, Sonarcloud fails to identify tailwind CSS class

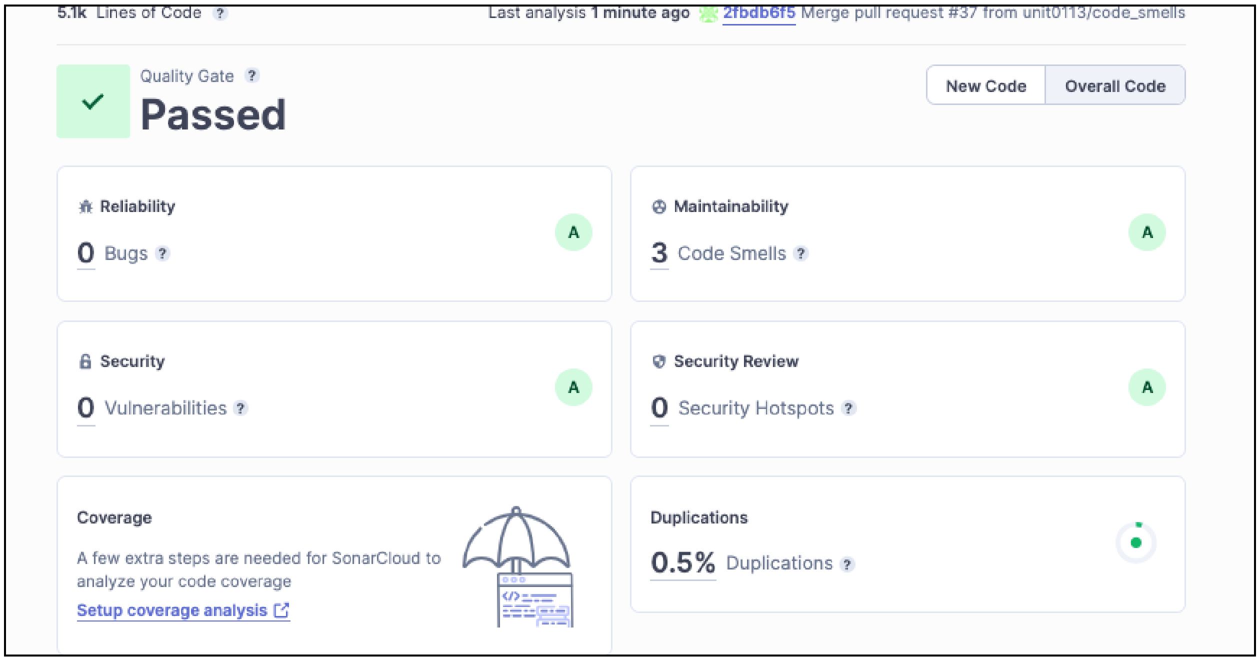
Unexpected unknown unit "xl"
Planned Bug Resolution: False positive, Sonarcloud fails to identify tailwind CSS class

Unexpected unknown at-rule "@screen"
Planned Bug Resolution: False positive, Sonarcloud fails to identify tailwind CSS class

Unexpected unknown at-rule "@screen"
Planned Bug Resolution: False positive, Sonarcloud fails to identify tailwind directive

A majority of the identified issues fell into the category of minor bugs, such as misnamed React functions or instances where React Hooks were erroneously called within functions. Our initial focus was on addressing these bugs as they directly impacted the program's functionality. Code smells, characterized as code deemed confusing or challenging to maintain by Sonarcloud, were also detected but were considered less critical. While we made efforts to address code smells where feasible, our priority was rightfully placed on resolving bugs due to their immediate impact on program performance and user experience.

Part 2


5.1k Lines of Code ? Last analysis 1 minute ago 2fbdb6f5 Merge pull request #37 from unit0113/code_smells

Quality Gate ? Passed

New Code | Overall Code

Reliability: 0 Bugs ? — A

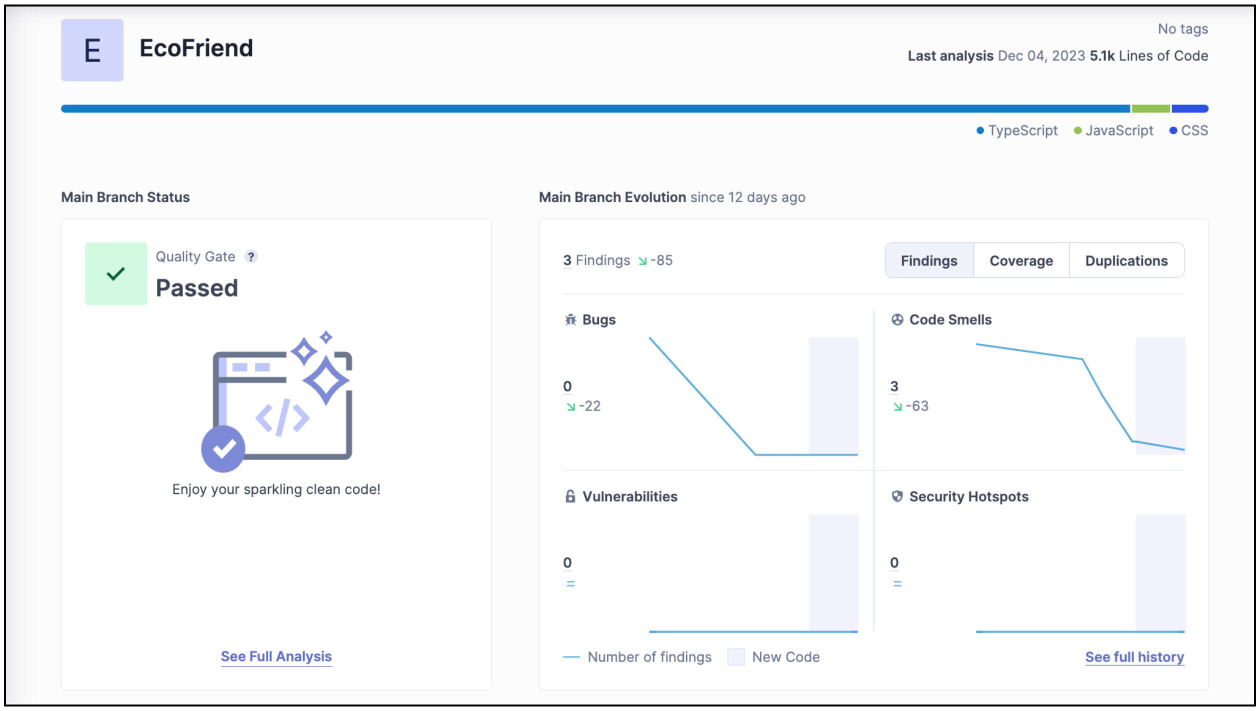
Maintainability: 3 Code Smells ? — A

Security: 0 Vulnerabilities ? — A

Security Review: 0 Security Hotspots ? — A

Coverage: A few extra steps are needed for SonarCloud to analyze your code coverage. Setup coverage analysis

Duplications: 0.5% Duplications ?



Remaining issues:

src/.../Blockquote.tsx

Consider removing 'undefined' type or '?' specifier, one of them is redundant.

Resolution: Performing the requested fix resulted in compilation errors, resolving as not an issue

src/.../Button.tsx

Consider removing 'undefined' type or '?' specifier, one of them is redundant.

Resolution: Performing the requested fix resulted in compilation errors, resolving as not an issue

src/.../GridPattern.tsx

Invalid property 'fill' found on tag 'pattern', but it is only allowed on: altGlyph, circle, ellipse, g, line, marker, mask, path, polygon, polyline, rect, svg, symbol, text, textPath, tref, tspan, use, animate, animateColor, animateMotion, animateTransform, set
src/.../RootLayout.tsx

Resolution: Fill property does affect this object and we are classifying this as a false positive

Section 3

Technical Details

The project utilizes several frameworks and packages. The website is built using the Next.js framework. It’s a powerful framework built on React and offers many improvements to the core react experience. Next.js is useful for its server-side and client-side rendering, automatic page routing, API routing within the application, typescript support, and most importantly, no configuration is required.

The website is hosted on Vercel. Vercel has a quick integration process, deploying directly from your GitHub repository. It automatically deploys and updates itself alongside the GitHub repository.

For styling, we used Tailwind CSS. Tailwind allows developers to design directly in HTML, reducing the need and bloat of traditional CSS.

The game's Unity engine utilizes a combination of 2D sprites and tilesets for its visuals, ensuring a lightweight and visually appealing experience. The programming is done in C#, with scripts handling everything from character movement to environmental interactions. Efficient use of Unity's physics engine allows for smooth and responsive gameplay. For sound, the game integrates sound effects, enhancing the game experience.

Central to this system is the use of tile-based mechanics, where individual tiles change visually and functionally based on player interactions. This is achieved through a combination of Unity’s Sprite Renderer and Animation systems. These responsive changes are driven by C# scripts that detect player actions and modify the states of the tiles.

"EcoAdventure" is optimized for web deployment. Hosted on Itch.io, the game is designed to load quickly and run smoothly across various browsers and devices, making it readily available to players directly from our website.

Installation Instructions

Before installation, ensure git and Node.js are installed on your system. Navigate to their respective sites to find the detailed installation guides.
To install the code on a local system, clone the GitHub repository on your local system, by running “git clone [repo-url]”. In a terminal window at the root of the source directory,

execute the command, “pnpm install” or “npm install” to install package dependencies. Once complete, deploy the project to your local host by executing “pnpm run dev” or “npm run dev”. Next.js automatically updates the application in the deployment environment whenever changes to the code are made, significantly reducing development time.

Login and Access Credentials and API Keys

Login and access controls utilize MongoDB as the back end database. This database contains two separate tables. The first is the user table, which stores the username, email address, a hashed password, as well as a unique id number of the team that they are a part of. The second is the teams table, which stores the team name, a hashed team password, and a list of members. These tables are connected together via a many-to-one relationship, with each user only being allowed to be a member of one team at any given time, and each team currently being allowed to have up to four members.

Access to this database, and the ability to perform the various account maintenance actions listed early in the report, are enabled by a combination of the NextAuth library, and the use of various RESTful APIs. NextAuth handles the user credentials, whether they are using a standard email and password combination, or logging in via their Google or Github credentials. Updates to the backend database are enabled by a securely stored API key, which allow the various POST, GET, and DELETE methods to perform the desired operations.

Below is the required .env file which contains the API keys that are required to run the website locally. The file should be placed in the main folder, at the same level as the src/ folder.

```
MONGODB_URI=mongodb+srv://ecofriend:kzsPhEjDxzIjBJuC@cluster0.g9as8rg.mong
odb.net/ecofriend
NEXTAUTH_SECRET=bileuthislergALFkcnbEESugh
NEXTAUTH_URL=http://localhost:3000/

GOOGLE_ID=605393816183-0iec23gnev8l2ndvm8hsbv28s4tcpmr6.apps.googleusercon
tent.com
GOOGLE_SECRET=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

GITHUB_ID=31ffe9839d4450ffb219
GITHUB_SECRET=b1f300207405c64954841a71365abdb1414963ff
```

Section 4

Risk Management Plan

Our project's risk management plan includes risks that we identified, their probabilities, effects, and responses. Risks are categorized based on their nature, probability, and potential effects. The plan covers technical challenges, including limited scalability and security vulnerabilities, with corresponding responses such as using cloud-based services and implementing encryption measures. User adoption concerns are addressed by planning an onboarding process and offering incentives for early adopters. People-related risks, particularly team members' unfamiliarity with the programming language, were mitigated through pre-sprint tutorials and careful language/framework selection. Organizational disruptions are managed by prioritizing flexible approaches and communication, while challenges in task estimation are approached through enhanced evaluation processes during product grooming and sprint planning.

Risks:	**Probability:**	**Effects:**	**Responses:**
Technical Challenge: The platform has limited scalability.	Low	Serious	Leveraging cloud-based services to easily increase resources as needed, ensuring the platform can accommodate growth.
Technical Challenge: The platform has security vulnerabilities	Moderate	Serious	Encrypting stored data and minimizing the collection of Personal Identifiable Information (PII).
User Adoption: Failing to gain user trust and active participation.	High	Serious	Launch a compelling user onboarding process and provide incentives for early adopters.
People: Significant portions of the team is unfamiliar with the programming language and framework	Moderate	Serious	Prior to the first sprint, team members new to the framework will work through several tutorials to develop familiarity with the language. Only development languages with significant documentation and/or tutorials will be chosen. Development that is complex and impacts the work of other members of the team will be well documented.

Organizational: Disruptions during the development cycle due competing priorities and other responsibilities.	Moderate	Tolerable	Prioritize flexible approaches and scheduling. Anticipate and communicate potential issues to the rest of the team.
Estimation: Difficulty in appropriately estimating time and complexity of tasks to assign necessary resources.	High	Tolerable	Extra focus will be given to evaluating task difficulty during product grooming and sprint planning. Refinements to our evaluation process will be a primary focus during sprint retrospectives.

Software Quality Attributes and Explanations

Product Operation

Correctness:
Upon check in, all software was subjected to various quality control checks, including a build viability check using Vercel, and a static code analysis from SonarCloud. We also performed our own functional testing, to ensure that new code adhered to the product requirements as laid out in the tasks and user stories.

Efficiency:
Both the website, which utilizes Next.js, and the game, which was created with the Unity Engine, are single threaded workloads and require only a single processor core to function. Because of this, significant effort was put in to minimize required API calls in the website, and limit unnecessary script functionality in the game.

Integrity:
The website has specific controls to block unauthorized users from accessing certain sections of the website. For security reasons, the user cannot access the game page without valid credentials. Additionally, the user is unable to access the various pages associated with account maintenance and control without having first logged in. Any attempt to access these parts of the site without valid credentials results in an immediate redirect to the home page.For further security, the user does not have access to links to these pages until they are logged in.

Reliability:
The website software makes extensive use of try-catch blocks when calling any function or API which has the possibility of failure. Additionally, the website will auto-redirect to the home page should it ever encounter an error. In cases of user error, warning messages are presented to the user informing them of which actions are causing errors, and how they can rectify the situation.

Usability:
A significant focus of the design effort was spent creating an intuitive design that was easy for the user to navigate and utilize. On the website, this entailed utilizing common design patterns, such as a hamburger style access menus, using standardized naming patterns, and placing links and other functionality in the areas where users would typically expect to find them. For the game, we aimed to simplify the control scheme, so that users could jump in immediately and begin playing without a steep learning curve.

Product Revision

Maintainability:
Our functional testing strategy enabled the development team to quickly detect errors as they inevitably arose during the development process. The team also made a concerted effort to work small to big, breaking down large tasks into smaller, more cohesive components, which made errors easier to isolate and also fit better into our overall testing process.

Flexibility:
The project's frontend framework is flexible by design, making extensive use of object oriented programming design principles. This enabled the development team to create the website, which, while having a large number and variety of components, nevertheless exhibited low coupling. As a result, making changes to any individual piece of the project tended to have very few negative effects on the rest of the program.

Testability:
The development team relied heavily on various forms of functional testing while developing both the website and the game. Because of the additional effort that this functional testing required, the team adapted by adopting a strategy which sought to limit the size and scope of any individual change. This way, functionality of the affected code could be tested without needing to perform a complete functional test of the entire system.

Product Transition

Portability:
The web-based nature of our program ensures remarkable portability, making it compatible with a wide array of devices accessible through web browsers. Whether users are on a desktop, laptop, tablet, or smartphone, our program adapts seamlessly, delivering a consistent and accessible experience across various platforms.

Re-usability:
While our application is purpose-built to meet specific needs, we've prioritized modularity in the game code. This deliberate design choice facilitates the easy movement of game components, allowing developers to leverage and integrate them into other projects. This emphasis on re-usability underscores our commitment to creating flexible and adaptable solutions.

Interoperability:
The integration of the game and web components exemplifies the interoperability of our program. The seamless collaboration between these two elements reflects our dedication to creating a harmonious and integrated user experience. This effortless interoperability ensures that users can navigate between the game and web interfaces with ease, contributing to a cohesive and interconnected digital environment.

References

Sommerville, Ian. “4.” *Engineering Software Products: An Introduction to Modern Software Engineering*, Pearson Education, Inc., Hoboken, NJ, 2020.

Archiveddocs. “Chapter 3: Architectural Patterns and Styles.” *Microsoft Learn*, learn.microsoft.com/en-us/previous-versions/msp-n-p/ee658117(v=pandp.10). Accessed 2 Dec. 2023.