Here’s a concise documentation draft (about 1–2 pages when formatted into PDF) that covers all the required points for the project:

Which mini-game did you choose and why?

I chose Arkanoid Brick Breaker because it has simple mechanics yet addictive gameplay.

Its core loop of bouncing a ball to break bricks is easy to understand but challenging to master, making it ideal for a prototype recreation.

What new creative feature did you add to the game?

a) Custom Paddle Design: Red ends with a white center for visual distinction.

b) Brick Fragment Animation: Animated fragments scatter above the title screen to polish the game.

c) Bonus Feature: A progress bar in the HUD that fills as bricks are destroyed, giving players visual feedback on completion.

Technical Approach and Architecture Decisions

a) Project Structure: Modular files (paddle.js, ball.js, brick.js, levels.js, ui.js, sound.js, game.js) for separation of concerns.

b) Game Loop: Implemented with requestAnimationFrame for smooth rendering at ~30 FPS.

c) UI Flow: Three main screens (Home, Level Select, Game) with fade-in transitions.

d) Responsive Design: Canvas locked to portrait 9:16 ratio, but UI grids adapt to different resolutions.

d) Performance: Preloaded sounds and lightweight assets to keep load time under 8 seconds.

Explanation of Bonus Feature Implementation

a) Progress Bar: Implemented as a simple div with a child #progressFill. Each time a brick is destroyed, the width of #progressFill increases proportionally.

b) Brick Fragment Animation: CSS keyframes animate small red blocks scattering above the title. Different delays create a natural staggered effect.

d) Level Housing: The Level Select grid is wrapped in a semi-transparent container with a border and rounded corners to visually group the buttons.

Generative AI Tool Usage

I used Copilot AI to:

a) Generate the project structure.

b) Generate the first sample of modular code for game entities and UI.

c) Curate sound integration.

d) To improve the styling of the code I made and to guide me on how to make the events of the project.

This accelerated development by providing reusable code snippets and creative polish ideas.

Challenges Faced and Solutions Implemented

a) Collision Detection: Ensuring the ball bounced correctly off paddle edges.

Solution: Adjusted bounce angle based on hit point relative to paddle center.

b) UI Transitions: Making screen changes smooth without breaking game state.

Solution: Added fade-in CSS animations and controlled visibility with .active class.

c) Level Unlocking: Designing a progression system.

Solution: Initially locked levels with padlock icons; future unlock logic can be added.

What would you improve with more time?

a) Dynamic Level Unlocking: Unlock next level after winning the current one.

b) Power-Ups: Add special bricks that drop items (multi-ball, paddle expand/shrink).

c) Background Effects: Add animated starfield or geometric patterns for a more immersive space theme.

d) Audio Depth: Layer background music with mute/unmute toggle and varied sound effects.

This documentation captures the design choices, creative additions, technical implementation, and future improvements for the Arkanoid Brick Breaker prototype.