# Game Design Document: "Cyber Blitz"

1. Game Overview

- Title: Cyber Blitz

- Genre: Top-down shooter

- Platform: PC (Java-based)

- Target Audience: Casual gamers, ages 12+

2. Story and Setting

- Setting: Futuristic cyberpunk city

- Premise: Player controls a rebel hacker fighting against corrupt AI-controlled robots

3. Gameplay Mechanics

a. Core Mechanics:

- Top-down perspective

- Twin-stick style controls (move with WASD, aim/shoot with mouse)

- Projectile-based combat

- Wave-based enemy encounters

b. Player Abilities:

- Basic shot: Rapid-fire energy bullets

- Special abilities: Shield, EMP blast, Hacking (temporarily turn enemies to allies)

- Dash movement for quick dodges

c. Enemy Types:

- Grunt: Basic melee robot

- Shooter: Ranged attack robot

- Tank: Slow, high-health robot

- Bomber: Explodes when near player

- Boss: Appears every 5 waves, unique abilities

d. Power-ups:

- Health restore

- Weapon upgrades (spread shot, piercing shot, rapid fire)

- Temporary invincibility

- Score multiplier

4. Game Progression

- Wave-based structure, increasing difficulty

- Boss fight every 5 waves

- Endless mode with high score system

5. User Interface

a. HUD Elements:

- Health bar

- Score

- Current wave

- Special ability cooldowns

- Mini-map (shows enemy positions)

b. Menus:

- Main Menu: Play, Options, High Scores, Quit

- Pause Menu: Resume, Restart, Options, Quit to Main Menu

- Game Over Screen: Final Score, Restart, Quit to Main Menu

6. Art Style

- 2D pixel art with neon cyberpunk aesthetics

- Character and enemy designs inspired by retro arcade games

- Dynamic lighting effects for weapons and explosions

7. Sound and Music

- Upbeat electronic soundtrack

- Sound effects for shooting, explosions, power-ups, and player actions

- Unique sound cues for special events (boss arrival, wave completion)

8. Technical Specifications

- Developed using Java and Swing

- Minimum resolution: 800x600

- Target frame rate: 60 FPS

9. AI Implementation Considerations

- Procedural level generation for varied gameplay

- Dynamic difficulty adjustment based on player performance

- AI-driven enemy behavior for unpredictable challenges

10. Monetization (Optional for future expansion)

- In-game currency for persistent upgrades

- Cosmetic skins for player character

11. Possible Expansions

- Multiple playable characters with unique abilities

- Co-op multiplayer mode

- Additional game modes (time attack, survival)

# the necessary steps for each phase.. focusing on the essentials:

## 1. Planning and Setup

- Finalize minimal game design document

- Set up Git repository

- Create project in IDE with necessary libraries (Swing)

2. Core Game Engine

- Implement GameLoop class with basic update() and render() methods

- Create simple Entity class for game objects (player, enemy, projectile)

- Implement basic rectangular collision detection

3. Player Mechanics and Basic Rendering

- Create GamePanel class extending JPanel

- Implement KeyListener for player input

- Create Player class with movement and shooting methods

- Implement basic Sprite class for rendering

4. Enemy AI and Spawning

- Create Enemy class with basic AI behaviors

- Implement EnemySpawner class to manage wave-based enemy creation

5. UI and Game Flow

- Create HUD class for rendering health and score

- Implement simple menu screens using Swing components

- Create WaveManager to handle game progression

6. Power-ups and Additional Features

- Create PowerUp class and basic power-up system

- Design and implement one additional enemy type

- Add basic sound effects using Java's built-in sound API

7. Polish and Extra Features

- Implement one special player ability

- Add pause functionality

- Fine-tune game parameters for basic balance

- Enhance visuals within time constraints (e.g., simple particle effects)

8. Final Testing and Packaging

- Perform playtesting and gather quick feedback

- Fix critical bugs

- Use Java's built-in packaging tools to create executable JAR

This revised plan focuses on getting a playable game with core features within the 15-day timeframe. It prioritizes essential gameplay elements while leaving room for some basic polishing and extra features if time allows.

Since you're working on the Git repo, the next step would be to set up your Java project in your IDE. Would you like guidance on creating the basic project structure or setting up the main game classes?