**Hackathon Project Phases Template** for the **AutoSage App** project.

Hackathon Project Phases Template

# Project Title:

**Flappy bird game**

# Team Name:

Meta crew

# Team Members:

* shushanth
* manispoorthy
* shiva kumar
* sai kumar
* vivek

# Phase-1: Brainstorming & Ideation

## Objective:

 Recreate the classic Flappy Bird game using web technologies.

 Improve the user experience with fluid mechanics and smooth animations.

## Key Points:

1. **Problem Statement:**

Challenge: Recreate the iconic Flappy Bird experience with added features, or use this as an opportunity to explore and apply new game development techniques

1. **Proposed Solution:**

#### ****Overview of the Solution****

Our solution is to develop the **Flappy Bird** game using **HTML5, JavaScript**, and the **Canvas API**. This approach allows for interactive and visually appealing gameplay while keeping the implementation lightweight and simple for a browser-based game.

**Key Features of the Proposed Solution**:

* **Smooth Gravity and Flap Mechanics**: The bird's motion will be affected by gravity, pulling it downward. The player will be able to control the bird's upward movement by pressing a key or clicking the mouse.
* **Dynamic Pipe Generation**: Pipes will be generated dynamically at regular intervals, with varying heights and gaps to create a challenging and unpredictable experience.
* **Collision Detection**: The game will detect when the bird collides with a pipe or the ground, triggering the end of the game.
* **Score Tracking**: The player’s score will increase each time the bird successfully navigates through a pair of pipes.

1. **Expected Outcome:**

 **Bird Movement**:

* The bird constantly falls due to gravity, which is a fundamental part of the game's challenge.
* The player can **flap** the bird upward, which temporarily counteracts gravity.

 **Pipe Generation**:

* Pipes are **generated dynamically** at regular intervals, with a random gap size.
* Pipes move from right to left, and the player must navigate through them.

 **Collision Detection**:

* The bird’s position is compared with the pipes and the ground to detect collisions.
* The game ends if the bird touches a pipe or the ground.

# Phase-2: Requirement Analysis

## Objective: The **objective** of performing a **Requirement Analysis** for the Flappy Bird game project is to ensure that the game’s design, development, and final implementation align with the needs and expectations of users, while also adhering to technical constraints. The following are the primary objectives of conducting this analysis:

## Key Points:

* **Technical Requirements:**
  + Programming Language: **Html**
  + Backend: **Google Gemini Flash API**
  + Frontend: **UI,graphics and animated**
* **Functional Requirements:**
* **Main menu**
* **Game play**
* **Scoring system**
* **Game over**
* **Constraints & challenges:**
* **Rapid development**
* **Bug fixing**
* **Game play balance**
* **Precision issues**

# Phase-3: Project Design

## Objective:

When designing a **Flappy Bird** clone for a hackathon or as a personal project, it's important to plan out both the **technical** and **artistic** aspects to ensure a smooth development process. Below is an outline for a basic design, divided into key components that you should consider:

+-------------------+

| Main Menu |

+-------------------+

|

(Click Start) or (Press Play)

v

+-------------------+

| Initialize |

| Game Variables |

+-------------------+

|

(Set Bird Position, Pipes, Score)

v

+-------------------+

| Game Loop |

+-------------------+

| | |

(Bird Movement) (Pipe Movement) (Collision Detection)

| | |

(Gravity & Player Input) (Scroll Pipes Left)

| | |

(Bird Flaps & Falls) (Random Pipe Gap)

| | |

+-------------------+

| Check for Game |

| Over Condition | <---> No Collision, Continue Game Loop

+-------------------+

| Yes |

(Bird Hits Pipe or Ground)

v |

+-------------------+

| Game Over |

| Show Final Score|

+-------------------+

| |

(Click Retry) (Click Main Menu)

v |

+-------------------+

| Restart Game |

+-------------------+

## Key Points:

1. **System Architecture:**
   * Game UI,Game core
   * Game objects like bird object,pipe object,background object
   * Collision detection
   * Game state management
   * Difficulty progression
2. **User Flow:**
   * Step 1: When the **Start Button** is pressed, the game begins
   * Step 2: The bird is controlled by the player, and the player must **tap/click** to make the bird flap and rise.
   * Step 3: The bird is controlled by the player, and the player must **tap/click** to make the bird flap and rise.
   * Step 4: The game tracks the score, which increments when the bird successfully passes through a gap between two pipes.
   * Step 5: The game tracks the score, which increments when the bird successfully passes through a gap between two pipes.
3. **UI/UX Considerations:**
   * Simplicity and Clarity
   * Consistent Visual Theme
   * Game State Indicators
   * Avoid Frustration

# Phase-4: Project Planning (Agile Methodologies)

## Objective:

Break down development tasks for efficient completion.

ere is the **project planning for Flappy Bird** game in a tabular form:

| **Phase** | **Task** | **Responsible Team Member(s)** | **Estimated Time** | **Deadline** |
| --- | --- | --- | --- | --- |
| **1. Project Setup & Initial Design** | Setup the development environment | Developer | 1-2 Days | Day 2 |
|  | Design the initial game concept (mechanics, layout) | Designer / Developer | 2-3 Days | Day 3 |
|  | Create basic game assets (bird, pipes, background) | Artist / Developer | 2-3 Days | Day 5 |
| **2. Core Game Development** | Implement bird movement (flap, gravity, physics) | Developer | 2-3 Days | Day 8 |
|  | Implement pipe generation & movement | Developer | 3 Days | Day 11 |
|  | Implement collision detection (bird with pipes/ground) | Developer | 2 Days | Day 13 |
|  | Implement scoring system (increment on passing pipes) | Developer | 1-2 Days | Day 15 |
|  | Implement game-over screen (score display, retry) | Developer | 2 Days | Day 16 |
| **3. UI/UX Design & Development** | Design and implement main menu screen | Designer / Developer | 3 Days | Day 19 |
|  | Design and implement game-over screen | Designer / Developer | 2-3 Days | Day 21 |
|  | Optimize controls and gameplay experience | Developer | 2-3 Days | Day 23 |
| **4. Sound & Visual Enhancements** | Integrate sound effects (flap, score, collision) | Sound Designer / Developer | 2-3 Days | Day 26 |
|  | Polish visual elements (animations, polished sprites) | Artist / Developer | 2-3 Days | Day 28 |
| **5. Testing & Debugging** | Conduct playtesting (on multiple devices) | Developer / Tester | 5-7 Days | Day 35 |
|  | Address feedback and bugs | Developer | 3-4 Days | Day 39 |
| **6. Deployment & Final Touches** | Package game for deployment (web/mobile) | Developer | 2-3 Days | Day 42 |
|  | Final bug fixes and testing | Developer | 1-2 Days | Day 43 |
|  | Publish game on hosting platforms (Web/Play Store) | Developer | 2 Days | Day 45 |

## 

### ****Summary of Key Milestones and Deadlines****

1. **Day 5**: **Initial Design and Setup Complete** (Environment setup and basic assets ready).
2. **Day 16**: **Core Game Development Complete** (Bird movement, pipes, collision detection, scoring, and game-over).
3. **Day 19**: **UI/UX Complete** (Main menu, game-over screen, gameplay optimization).
4. **Day 28**: **Sound & Visual Enhancements Complete** (Sound integration and visual polish).
5. **Day 35**: **Testing and Debugging Complete** (Playtesting, bug fixes).
6. **Day 45**: **Deployment and Final Touches Complete** (Game ready for deployment and final publishing).

# Phase-5: Project Development

## Objective:

These objectives guide the overall development process and help ensure that the project is successful.

## Key Points:

1. **Technology Stack Used:**
   * **Frontend:github**
   * **Backend:** google Gemini canvas API
   * **Programming Language: Html**
2. **Development Process:**
   * Implement **API key authentication** and **Gemini API integration**.
   * Develop **game comparison and maintenance tips logic**.
   * Optimize **search for performance and relevance**.
3. **Challenges & Fixes:**

**Challenge:** As the game’s graphics, animations, and physics grow more complex, performance might degrade, especially on lower-end devices.

**Fixes**:

* + **Optimize Image and Asset Sizes**: Compress images and optimize sprites. Use image formats like PNG for transparent images or JPEG for backgrounds to reduce file size.
  + **Reduce Redundant Operations**: Minimize unnecessary calculations in the game loop, such as redundant checks for collision detection or frame updates.
  + **Use RequestAnimationFrame**: Instead of using setInterval or setTimeout, which can cause lag, use requestAnimationFrame() for smoother animations and frame updates.
  + **Limit the Number of Objects**: Avoid creating too many game objects at once. Recycle and reuse objects, such as pipes, instead of creating new ones for each frame.

**Challenge:** Inaccurate collision detection (e.g., detecting whether the bird has hit As the game’s graphics, animations, and physics grow more complex, performance might degrade, especially on lower-end devices

**Fixes**:

* + **Bounding Box Collision Detection**: Implement a more efficient way of detecting collisions by using bounding boxes or hitboxes. These can be rectangular or circular, depending on your game objects.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

# Phase-6: Functional & Performance Testing

## Objective:

ensures that all features of the game work as intended

| **Test Case** | **Test Description** | **Expected Result** | **Pass/Fail Criteria** | **Responsible** |
| --- | --- | --- | --- | --- |
| **Frame Rate (FPS) Consistency** | Test if the game maintains a consistent frame rate (FPS) during gameplay. | The game should maintain a smooth frame rate, ideally 60 FPS. | Frame rate should remain stable and smooth. | Tester/Developer |
| **Memory Usage** | Test the game’s memory usage during extended gameplay (e.g., 30 minutes). | The memory usage should remain constant without significant increase. | Memory usage should not spike excessively. | Tester/Developer |
| **Load Time** | Test the time it takes for the game to load (from clicking “Start” to gameplay). | The game should load within 3-5 seconds. | Load time should be less than 5 seconds. | Tester/Developer |
| **Responsive UI** | Test if the game’s UI adapts to various screen sizes (mobile, desktop). | The UI should adjust to different screen sizes without distortion. | UI should be fully responsive across devices. | Tester/Developer |
| **Device Performance (Mobile)** | Test the game on various mobile devices to ensure it runs smoothly. | The game should run without crashes or significant lag on all tested devices. | No crashes, and the game should perform smoothly on most devices. | Tester/Developer |
| **Background Process Handling** | Test if background processes (e.g., notifications, calls) impact gameplay. | The game should pause properly when interrupted and resume without issues. | No game crashes or performance drops during interruptions. | Tester/Developer |
| **Multiple Players (Multitasking)** | Test how the game performs when switching between tabs or apps on mobile devices. | The game should pause and resume without issues when switching apps/tabs. | Game should pause when switching away, resume without issues. | Tester/Developer |
| **Battery Drain (Mobile)** | Test the battery drain during gameplay (on mobile devices). | The game should not cause excessive battery drain during gameplay. | Battery drain should be moderate during gameplay. | Tester/Developer |

# Final Submission

1. **Project Report Based on the templates**
2. **Demo Video (3-5 Minutes)**
3. **GitHub/Code Repository Link**
4. **Presentation**