# Gamification-Based Teaching Project Proposal Template

## Team Information

* **School: İstanbul Kültür Üniversitesi**
* **Semester: Spring**
* **Course Code: SEN4012**
* **Course Name: Analysis of Algorithms**
* **Team Name: Binary Blossoms**
* **Date: 30.05.2025**

### Team Members

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Name | Surname | Email | Phone Number |
|  |  |  |  |  |
| 2100005804 | Hayal İldeniz | İnanç | [2100005804@stu.iku.edu.tr](mailto:2100005804@stu.iku.edu.tr) |  |
| 2000004664 | Özge | Küçükbayram | [2000004664@stu.iku.edu.tr](mailto:2000004664@stu.iku.edu.tr) |  |
|  |  |  |  |  |

## Project Overview

* **Project Name: Learning** Algorithms With Games
* **Type:** (**Digital** / Physical / Hybrid)
* **Description:** This project consists of three interactive games designed to teach fundamental algorithm analysis concepts: Selection Sort, Greedy Algorithm, and Binary Search. Each game simulates the algorithm in an engaging, fun and physical way to reinforce comprehension of time complexity and logical structure.

## Educational Objectives

* **Primary Objectives:**
  + Teach algorithmic thinking through physical simulations
  + Solidify understanding of time complexity with real-world task analogies
* **Secondary Benefits:**
  + Encourages fun and active learning
  + Improve problem-solving and analytical thinking through experiential learning
* **Demographic Profile:** Ages 10–22, undergraduate computer science student
* **Learning Styles:** Visual, Active Participation, Collaborative
* **User Personas: A** visual learner struggling with abstract Big-O concepts; A person that wants to learn algorithms with fun games.
* **Mechanics:** Points, Game Selection, Leaderboards
* **Dynamics:** Competition (fastest completion), Collaboration (group sort tasks), Progression (increasing difficulty)
* **Rewards:**
  + Sense of mastery and accomplishment, Internal motivation to improve.
* **Narrative:** “It’s not just about solving problems; it’s about solving them efficiently.”
* **Social Interaction:** Users compare results and scores
* **Customization:** Encouraging the player to strive for a better score.

## Artifacts

* **Physical Artifacts:** Numbered cards, item tokens (e.g., Wallet, Clock), backpacks, printed task cards
* **Digital Artifacts** Leaderboard, performance tracking dashboard

## Environment Setup

* **Physical Zones:** Setup Area: distribution of materials
* Game Area: execution stations for each algorithm
* **Digital Zones:** Leaderboard display, Game Selection Interface
* **Visual Aids:** Algorithm posters, sorting diagrams, step-by-step instructions

## Game Descriptions

### Game 1: 🩷 Selection Sort – " Selection Sort Dance"

**Algorithm:** Selection Sort (O(n²))  
**Teaching Method:** Sorting 7 shuffled cards using swaps only; select the smallest remaining unsorted card and place it in the correct position.  
**Duration:** 5 minutes max  
**Scenario:** Random shuffle → swaps → check order → score  
**Randomness:** Initial card order is randomized  
**Scoring:** 100 - 10 × (Swaps - IdealSwaps), (min 0)  
**Intro Instructions:** Clear explanation of goals, algorithm, scoring, roles, success criteria

### Game 2: 💼 Greedy Algorithm – "Treasure Hunt"

**Algorithm:** Greedy (Fractional Knapsack, O(n log n))  
**Teaching Method:** Maximize item value within 15kg bag using value/weight ratio  
**Duration:** 5 minutes max  
**Scenario:** Choose items → observe weight/value updates → finish  
**Randomness:** Random item weights/values  
**Scoring:** (Player Value / Greedy Max Value) × 100, (rounded)  
**Intro Instructions:** Objectives, algorithm steps, strategy tips, scoring details

### Game 3: 🔍 Binary Search – "Guess the Number"

**Algorithm:** Binary Search (O(log n))  
**Teaching Method:** Guess number 1–100; halve search space per attempt  
**Duration:** 5 minutes max  
**Scenario:** Generate secret number → user guesses → feedback → score  
**Randomness:** Number is randomized  
**Scoring:** 100 - 10 × (Steps - IdealSteps), ((log₂(100) ≈ 7) Ideal = 7 steps)  
**Intro Instructions:** Goal, algorithm logic, efficient search examples

## Game Flow Plan

* **Pre-game Setup:** Initialize web-based game environment. Load all task modules (Selection Sort, Greedy Selection, Binary Search Guess) into the platform. User login and session setup. The system assigns the player’s name and tracks progress. User is prompted to start with the first algorithm challenge
* **Introduction Phase:** Explain the game rules and learning goals for each algorithm
* **Planning Phase:** Strategy development
* **Execution Phase:** Selection Sort: Sort shuffled number cards, select the smallest remaining unsorted card and place it in the correct position.
* Greedy: Choose highest-value items within backpack weight limit.
* Binary Search: Guess the system's hidden number in minimal steps.
* **Review Phase:** Evaluate step counts and time complexity alignment
* **Debrief Phase:** Discuss algorithm performance and possible optimizations
* **Scenarios:**
  + Main Scenario: Three interactive stations for Selection, Greedy, Binary
  + Alternative Scenario(s): Timed tournament

## Assessment and Performance Indicators

* **Performance Indicators:**
  + Task completion accuracy
  + Step count and efficiency
  + Correct algorithm classification
  + Quality of collaboration
  + Problem-Solving Ability
* **Assessment Methods:**
  + Rubric-based evaluation
  + Peer Assessment
  + Self-Assessment
* **Grading Criteria:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Game** | **Scoring Formula** | **Bonus** | **Penalty** |
| **Selection Sort** | 100 - 10 × (Swaps - IdealSwaps) (min 0) | If swaps = ideal swaps → **100 full points** | +1 swap over ideal → -10 Significantly over → 0 |
| **Greedy** | (Player Value / Greedy Max Value) × 100 (rounded) | Used all 15kg efficiently (e.g., ≥95% of max score) | Low total value or very underutilized capacity |
| **Binary Search** | 100 - 10 × (Steps - IdealSteps) ((log₂(100) ≈ 7) Ideal = 7 steps) | Found in 7 steps or fewer → **100 points** | Each step above 7 → -10 Too many steps (e.g., >14) → 0 |

## Execution and Monitoring

* **Execution Log Template:**
  + **Participants/Teams:** Each log entry is tied to a name representing an individual participant.
  + **Roles:** The instructor is assigned only the role of questioning. The player is the sole executor during the game.
  + **Tasks**: Each task attempt is logged with metadata: task type (e.g., Binary Search), time spent, and result (success/fail).
  + **Events**: Key interactions such as task start/end times, wrong moves, retries, and completion moments are tracked.
  + **Points and Rewards Tracking:** The system logs the points earned after each task.
  + **Observations and Notes:** Optional fields for auto-generated insights (e.g. “You achieved a higher score than the greedy estimate”).
* **Leaderboard:** Updated after each round based on efficiency scores

## Risks and Mitigation Strategies

* **Identified Risks:**
  + Time mismanagement
  + Rule misunderstanding
* **Mitigation Strategies:**
  + Preventive Measures: The user interface is designed to be simple and intuitive; each task is introduced with a brief explanation and a sample interaction.
  + Responsive Measures: If a user completes a task incorrectly, feedback is provided, along with the opportunity to retry.

## Tools and Technologies Used

## This project was developed using the following tools and programming languages:

HTML – For the structure of the game screens

CSS – For visual styling and responsive layout

JavaScript – For game logic, timers, and user interaction

Python (Flask) – For server-side logic, data handling, and routing

JSON – For storing and managing leaderboard data

Visual Studio Code – As the main development environment

## Data Collection and Feedback

* **Feedback Survey:**
* Did you understand the algorithm clearly?
* Did the game help solidify your learning?
* Would you recommend this approach?
* **Log Management:** Digital logs submitted by each task moderator,

## Continuous Improvement

* **Proposed Enhancements:**
* Add more algorithms (e.g., Merge Sort, Quick Sort)
* Integrate mobile tracking apps
* Organize inter-university tournaments based on scores