Feasibility Study: Code Quest - An Educative Coding Game

Mohamed Amine El Bacha and Younes Menfalouti

UM6P - Cyber Security Department

October 2025

Contents

0.1 Introduction

This document outlines a feasibility study for **Code Quest**, an educational coding game designed for children eager to learn programming. The game seeks to introduce young learners to coding concepts through an interactive experience, where players control the game using their own lines of code, seamlessly blending fun and learning.

—

0.2 Technical Feasibility

0.2.1 Tools and Technologies Required

Table 1: Technical Requirements Analysis

Category	Available/Planned	Gap Analysis
Front-end	pygame	Team has basic pygame knowledge, need to master pygame and game development
Backend	python(Flask, django), Firebase	Team has basic python experience, needs learn Firebase
Design Tools	Figma, canva	Team member has design experience
Hardware	Standard laptops	no gaps

0.2.2 Team Skills Assessment

Table 2: Skills Inventory

Skill	Team Proficiency	Required Level
Game Development	Beginner	Intermediate
Python Development	Intermediate	Intermediate
Database Design	Beginner	Intermediate
UI	Beginner	Intermediate

0.2.3 Potential Technical Challenges

- Handling errors for beginners (translating complex Python errors into kid-friendly messages)
- Displaying simple UI elements (buttons, text boxes, instructions)
- Creating simple animations (character movement, object interactions)
- Playing basic sound effects (collecting items, moving, errors)

0.2.4 Conclusion

The project is technically feasible with some learning requirements. The team has the foundational knowledge needed and access to required resources through the university.

—

0.3 Market / User Feasibility

0.3.1 Target Users

- **Primary:** Primary student (ages 13-16) want to learn how to code
- **Secondary:** All people want to learn coding

0.3.2 Competitive Analysis

Table 3: Competitive Benchmarking

Platform	Key Features	Strengths	Weaknesses
Code Combat	Real Python/JavaScript Coding, Progressive Difficulty Levels	Effective Educational Framework	Subscription Cost Barrier
Screeps	Player vs. Player Combat, Mod Support, Continuous World Simulation	Real-World Programming Application, Teaches Scalable Architecture	Extreme Learning Curve, Slow Early Progression

0.3.3 Identified Gaps and Opportunities

- **Gap:** Lack of educational coding games that teach actual Python syntax to young children through an engaging, nature-themed environment with appropriate scaffolding for different age groups.
- **Opportunity:** Create the first nature-themed Python coding game that seamlessly blends authentic programming education with engaging gameplay for children, filling the critical gap between playful visual tools (like Scratch) and intimidating adult-focused platforms (like CodeCombat).
- **Unique Value:** Kids-friendly coding
- **Target Market:** Children aged 8-14 transitioning from visual block-coding to Python syntax, supported by educators and parents seeking engaging, nature-themed programming education.

0.3.4 User Survey Results (Sample)

A quick survey of 24 students revealed the following key insights, which are illustrated in the figures below:

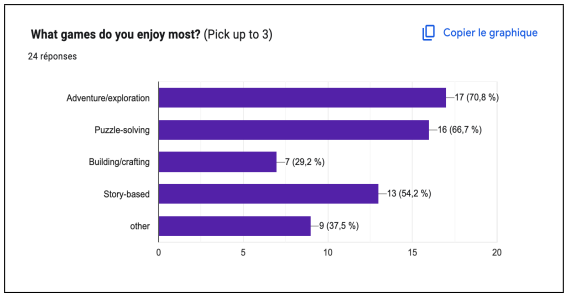

Figure 1: Gaming Preferences Distribution (70,8% prefer adventure and exploration))

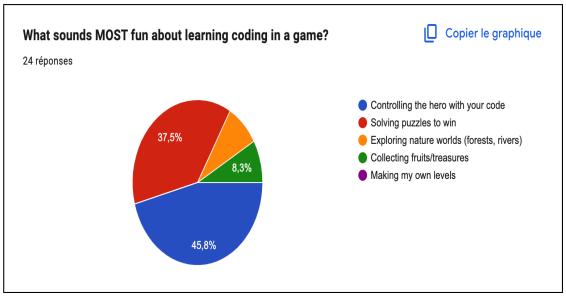

Figure 2: Motivational Factors in Educational coding games (52% prefer controling heros through codes)

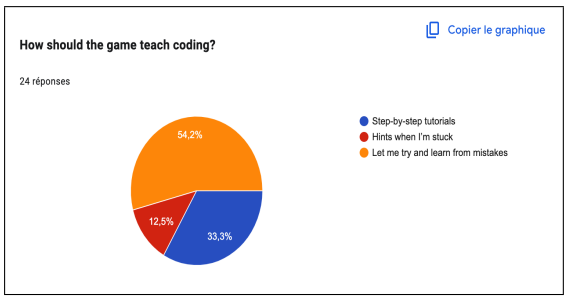

Figure 3: Preferred Coding Instruction Methods (54,2% favor learning by doing approch)

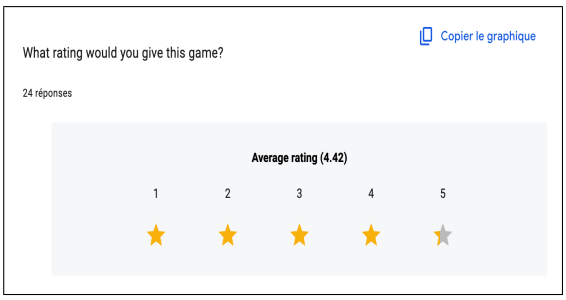

Figure 4: Overall Game rating Distribution (60% rate the game idea 4,5/5)

0.4 Schedule Feasibility

0.4.1 Project Timeline Overview

Table 4: High-Level Project Timeline (7-Week Semester)

Phase	Duration	Key Deliverables
Week 1: Planning & Design	1 week	Feasibility Study, High-Level Design
Week 2-3: Core Development	2 weeks	Game Engine Setup, Basic Level Mechanics
Week 4-5: Advanced Features	2 weeks	Python Command Integration, Advanced Levels
Week 6: Testing & Refinement	1 week	Internal Testing, Bug Fixing
Week 7: Deployment & Documentation	1 week	Final Polish, User Guide, Presentation

0.4.2 Task Breakdown and Dependencies

- **Game Development Path:**
  - Game Design → Core Game Mechanics → Code Execution System → Level Design → Testing → Polish
  - Curriculum Design → Python Command Implementation → Tutorial Creation → Progressive Difficulty System → User Testing → Refinement
- **Tight Deadlines:**
  - Core development phase (Weeks 3-6) is intensive
  - Python commands integration must be completed by Week 5 to allow sufficient testing
- **Risk Areas:**
  - Children entering arbitrary Python code could crash the game or compromise system security.
  - Failing to balance fun gameplay with effective Python instruction

0.4.3 Timeline Assessment

The 7-week timeline is aggressive but achievable given:

- Team of 2 developers (cybersecurity student, 15-20 hours/week each)
- Clear milestone definitions and regular progress reviews
- Buffer time included in testing phase for unexpected delays

—

0.5 Conclusion

The Code quest project demonstrates strong feasibility across all three dimensions:

- **Technical:** Achievable with current team skills and available resources
- **Market:** Clear user need identified with viable market opportunity
- **Schedule:** Ambitious but realistic timeline for semester completion

The project is recommended to proceed to the implementation phase with continued monitoring of the identified risks and challenges.