The Icarus Protocol

Project Solution Approach

Lincoln Middle School

Collin Nelson;

Anna Ueti;

10.3.22

TABLE OF CONTENTS

I. Introduction

This document, Project Solutions Approach, serves as a comprehensive overview of the inter-relational design of the gamified project commissioned by Lincoln Middle School. It will focus on three main designs, Architecture, Data, and User Interface. For Architecture, each subsystem within the overarching design of the project will be thoroughly explored and explained, with a focus on the subsystem's concepts, algorithms, and interface properties. For Data, any data type and database interaction and properties will be diagrammed and evaluated. And for User Interface, each page will have a mocked version and a detailed description of components and functions.

The purpose of this document is to create a guideline for the developers to follow throughout the development phase of this project. The developers will be able to perform verification tests during development based on the detailed description of

i. Subsystems descriptions

ii. Subsystem relationships

iii. Data type objects

iv. Data type and Database relationships

v. Database schema

vi. User Interface Mocked pages

Additionally, another purpose for this design document is for stakeholders. With these designs created and documented within this solution, stakeholders will be able to cross reference the prototype with the intended goals and design outlined here.

Our team aims to explore the potential of games as a tool for education by producing a fully featured game designed to teach basic programming skills to students at a middle school or early high school level with no prior programming experience. While games of a similar nature exist, they often fall into one of two traps which our team sees as pitfalls. Some, while employing the surface level appearance of a game, fail to truly embody the game design principles that make games powerful for learning; Others use a proprietary scripting language that has lessened impact in teaching actionable programming skills. In the construction of this project (working title: "Icarus Protocol") we aim to solve this problem by producing a game that is a genuinely fun and interesting experience, while also serving as an effective tool for teaching real Python programming.

II. System Overview

III. Architecture Design

III.1. Overview

This section should describe the overall architecture of your software. The architecture provides the top-level design view of a system and provides a basis for more detailed design work. This will be the initial draft of your software architecture. Next semester you will revise this draft and finalize your design.

- Provide a bird's-eye view of your software architecture. Mention the architectural pattern you adopted in your software and briefly discuss the rationale for using the proposed architecture (i.e., why that pattern fits well for your system).
- Please refer to CptS 322, CptS 487, CptS 321, and CptS 422 materials to refresh your knowledge on system decomposition and software architectural patterns.
- Briefly describe each layer/component in the architecture and explain its responsibilities.
- Provide a block diagram (e.g., UML component diagram) that illustrates the proposed architecture. The block diagram should show all major subsystems and identify the layers/components in the architecture.

III.2. Subsystem Decomposition

This section explains how you decomposed your system into subsystems. A subsystem typically corresponds to the amount of work that a single developer can tackle. You will show your system decomposition, identify the major subsystems, describe the assignment of functionality to each subsystem, and define the interfaces between them. When you decompose your system into subsystems, you need to consider the dependencies within and between the subsystems, i.e., cohesion and coupling measures.

- Briefly explain how you decomposed your system into subsystems.
- Discuss the rationale for the proposed decomposition in terms of cohesion and coupling.
- Redraw your architecture diagram (in section III.1) and show all the services each subsystem provides and requires (for example, UML component diagram that uses ball-and-socket notation to depict provided and required interfaces).
- For each subsystem in your architecture, include a sub-section.
- To improve clarity, you may provide multiple figures that show different parts of the architecture (illustrating services) and place each figure right before the corresponding subsection.

I.1.1. [Subsystem Name]

Include the following sub-sections for each subsystem.

a) Description

Describe the subsystem and identify its responsibilities.

b) Concepts and Algorithms Generated

Discuss the concepts, algorithms or solutions generated and considered for this subsystem. Report the selected solution and explain the solution selection process. Include any special considerations and/or trade-offs considered for the solution approach you have chosen.

c) Interface Description

Provide a description of the subsystem interface. Explain the provided services in detail and give the names of the required services.

Services Provided:

1. Service name:
   Service provided to: [list the receiving subsystems here]
   Description: [Describe what the service is and what it does. Provide its input and output values. Briefly describe the major functions that the service provides.]

Services Required:

Names of the required services and the subsystems that provide them.

I.1.2. [Include sections III.2, III.3, etc., for other subsystems]

IV. Data design

V. User Interface Design

[You may skip this section if your project doesn't have a GUI component] – but! If the tools is ever to be used by humans (even just starting and stopping it), there's some form of user interface design. It can be very simple, but it does exist. Make sure you document how you expect people to use your product, even if it's just:

- Installation
- Configuration file edits
- Launch daemon by running command [x]

Provide a detailed description of user interface. The information in this section should be accompanied with proper images showing how exactly you vision the interface to be like (for example mock-ups). Make sure to mention which use cases in your "Requirements Specification" document will utilize these interfaces for user interaction.

VI. Glossary

.

VII. References

VIII. Appendices