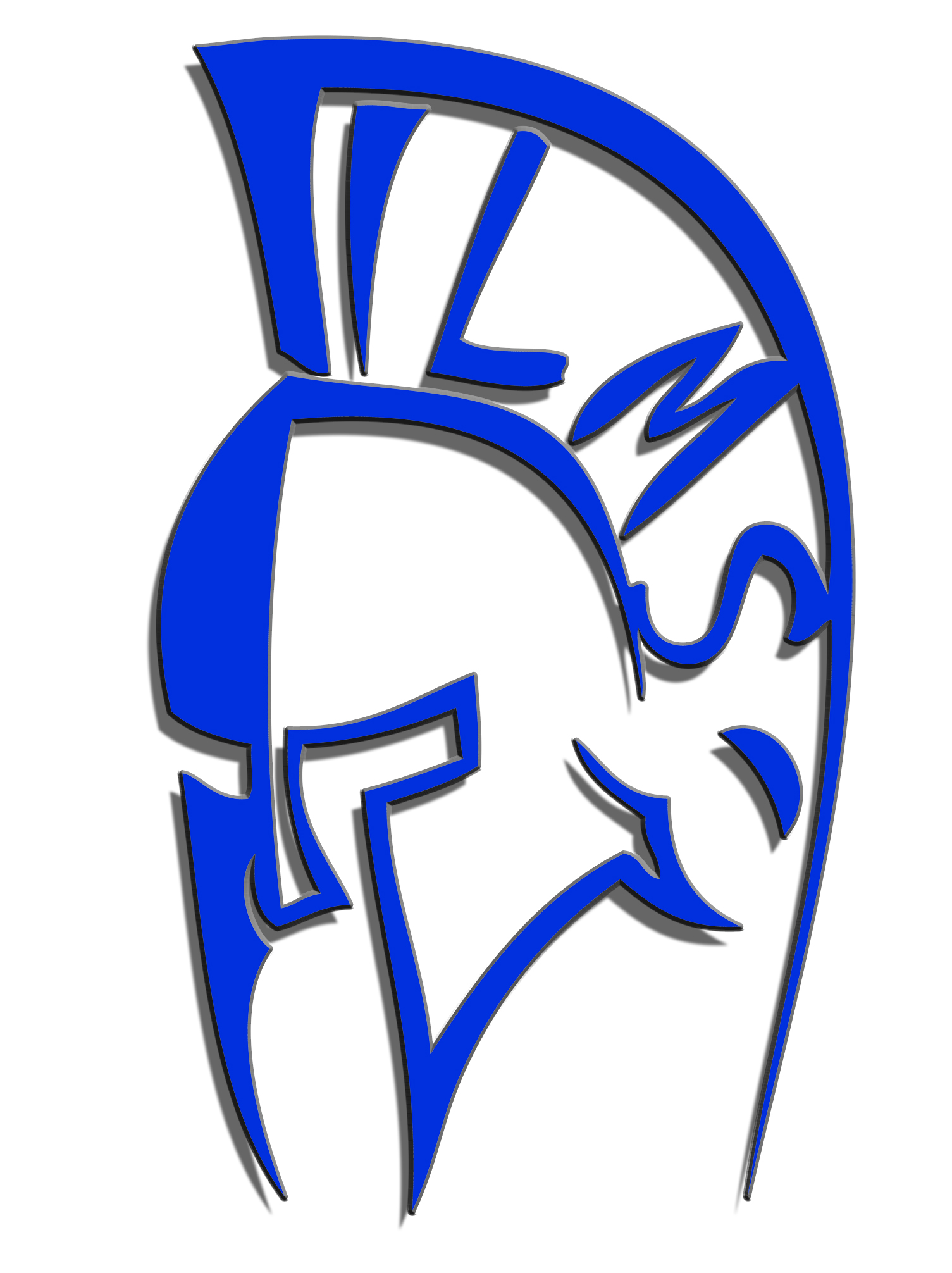
The Icarus Protocol

Project Solution Approach

Lincoln Middle School





Collin Nelson;

Anna Ueti;

10.3.22

**TABLE OF CONTENTS**

|  |  |
| --- | --- |
| **I. Introduction** | 3 |
| [**II. System Overview**](#_heading=h.3znysh7) | 3 |
| [**III. Architecture Design**](#_heading=h.2et92p0) | 3 |
| III.1 Overview | 3 |
| III.2 Subsystem Decomposition | 6 |
| I.1.1 [Subsystem Name] | 6 |
| a) Description |  |
| b) Concepts and Algorithms Generated |  |
| c) Interface Description |  |
| I.1.2 [Subsystem Name] |  |
| II.3 Non-Functional Requirements | 7 |
| [**IV. Data design**](#_heading=h.26in1rg) | 8 |
| [**V. User Interface Design**](#_heading=h.lnxbz9) |  |
| [**VI. Glossary**](#_heading=h.35nkun2) | 9 |
| [**VII. References**](#_heading=h.1ksv4uv) | 9 |
| [**VIII. Appendices**](#_heading=h.44sinio) |  |

# 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.

# System Overview

This project’s main functionality surrounds the idea of teaching python to students with a novel and engaging experience, focusing on skill and knowledge retention through the avenue of gaming. The functionalities of this project can be loosely divided into three major categories; User Interface and Python IDE, IronPython Integration, and Core Game Systems. Our design methodology is heavily influenced by the engine specific requirements caused by using the Unity Engine to drive development. The effect of this influence on our system architecture and component design patterns will be more thoroughly discussed in the Architectural design overview section. In regards to the design of our data structures, there are two major considerations that will be discussed. The first is the method of storing user progress in reliable and extensible save files. The second major consideration is the structure of our level data, which will have to be designed to be both highly flexible and easily modified without recompilation of the primary executable. The core consideration when designing our UI will be the strengths and limitations of the Unity UI system. While this system is flexible enough to be useful for our purposes, its limitations will have to be taken into account when creating UI layouts. The specifics of our component interfaces, data structures, and UI layouts will be discussed in the following sections.

# Architecture Design

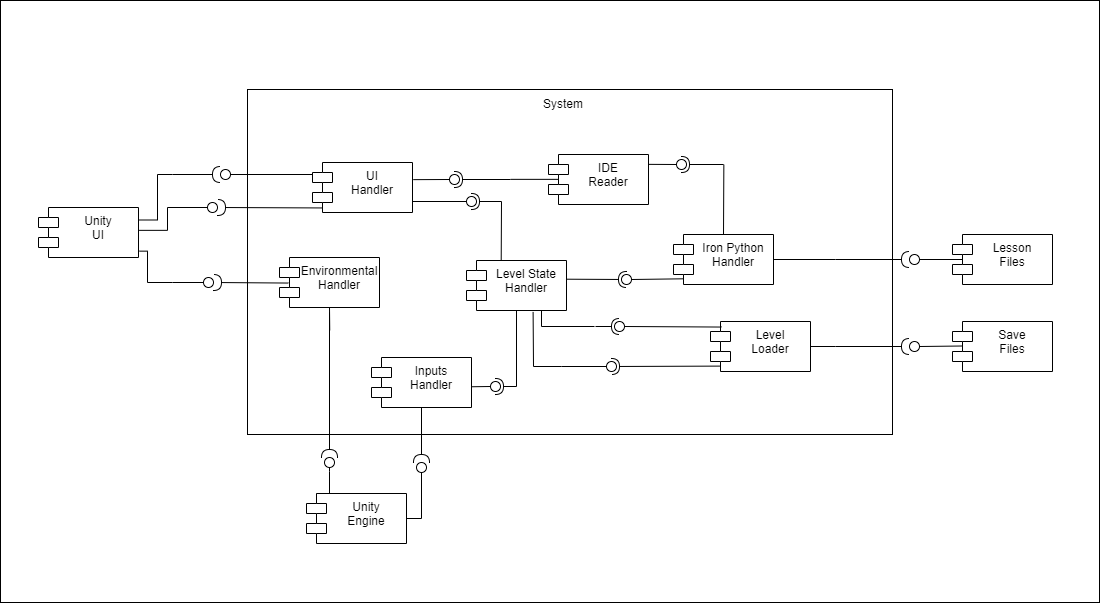
## Overview

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.

Architecture: Object Scripting Model

The Pluribus Doctrina Team has chosen to use an Object Scripting Model for this project. Since the application is developed in the Unity Engine, the architecture model must conform to the engine specifications. Given these requirements, the team has decided on an engine specific model that will utilize the strengths of the Unity Engine. Within the context of solely working on the game in the Unity engine, the team couldn’t see another pattern that would take advantage of the Unity system as well. Below is a component diagram that illustrates the overarching architecture of our game. This diagram’s purpose is two-fold, one, it serves as a guide for the developers to reference and base development around, and two, it provides a visual representation of the component architecture that is decomposed in the following section. Here is an overview of the components, as seen in the diagram below. Given the nature of working in the Unity environment, the player will start with the Unity User Interface. The Unity UI will then communicate with select system handlers. Such systems would be, one, the UI Handler, which will facilitate any UI element such as page layouts and icons, and two, the Environmental Handler, which modifies environmental properties such as camera positioning. From the UI Handler, there is a connection to the Level State Handler, which is the main subsystem that handlers the levels which the player interacts with. Level State Handler connects to the Iron Python Handler - facilitates the interpretation and simulation of the python code inputted and loads lesson files -, the Level Loader – saves and loads levels as the player progresses -, Inputs Handler – abstraction for user input from unity engine. Additional in-depth descriptions of these subsystems will be elaborated on in the following section.



## 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.

### [UI Handler]

#### Description

The UI Handler subsystem manages any user interaction with the UI elements as well as UI details such as the displayed layout and icons. It handles inputs from the Unity UI external component such as actionable functionality that is tied to UI elements or player code entered into a UI element.

#### Concepts and Algorithms Generated

Similar to the structure of Unity’s systems, the UI Handler is dispersed across many sub-classes. The most prominent of these sub-classes would be the UI-Layout class that dictates to the Unity UI which layouts should be displayed. The UI Handler also consists of the scripts tied to each UI elements’ input events. In alignment with the Unity Engine’s design philosophy, each button or UI element may have one or more atomic scripts which handle event actions. For the purposes of simplicity these are all considered to be part of the UI Handler subsystem. These classes will handle the interaction between the Unity UI elements and other subsystems.

#### Interface Description

Services Provided:

1. **Service name:** UpdateLayout

**Service provided to:** Unity UI, Level State Handler

**Description:** The UpdateLayout service will allow the Unity UI or the Level State Handler to call for an update to the page layout, this will occur for the transitions between the home page, the level select page, the manual page, and the level page, with any variations due to level phase.

1. **Service name:** BundlePlayerCode

**Service provided to:** IDE Reader

**Description:** The BundlePlayerCode service will package the code written in a UI element into a complied data object that is passed to the IDE reader, this will happen once the player has selected simulate.

Services Required:

1. **Service name:** [**Input name here**] to UI Handler

**Service provided from:** Unity UI

### [Subsystem Name]

Include the following sub-sections for each subsystem.

#### Description

Describe the subsystem and identify its responsibilities.

#### 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.

#### 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.

### [Subsystem Name]

Include the following sub-sections for each subsystem.

#### Description

Describe the subsystem and identify its responsibilities.

#### 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.

#### 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.

### [Subsystem Name]

Include the following sub-sections for each subsystem.

#### Description

Describe the subsystem and identify its responsibilities.

#### 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.

#### 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.

### [Subsystem Name]

Include the following sub-sections for each subsystem.

#### Description

Describe the subsystem and identify its responsibilities.

#### 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.

#### 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.

### [Subsystem Name]

Include the following sub-sections for each subsystem.

#### Description

Describe the subsystem and identify its responsibilities.

#### 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.

#### 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.

### [Subsystem Name]

Include the following sub-sections for each subsystem.

#### Description

Describe the subsystem and identify its responsibilities.

#### 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.

#### 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.

# Data design

# 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.

# Glossary

HUD: Heads Up Display

IDEL

# References

# VIII. Appendices