ChessEDU

Version <1.1>

[Note: The following template is provided for use with the Unified Process for EDUcation. Text enclosed in square brackets and displayed in blue italics (style=InfoBlue) is included to provide guidance to the author and should be deleted before publishing the document. A paragraph entered following this style will automatically be set to normal (style=Body Text).]

[To customize automatic fields in Microsoft Word (which display a gray background when selected), select File>Properties and replace the Title, Subject and Company fields with the appropriate information for this document. After closing the dialog, automatic fields may be updated throughout the document by selecting Edit>Select All (or Ctrl-A) and pressing F9, or simply click on the field and press F9. This must be done separately for Headers and Footers. Alt-F9 will toggle between displaying the field names and the field contents. See Word help for more information on working with fields.]

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| <dd/mmm/yy> | <x.x> | <details> | <name> |
| 25/10/2022 | 1.0 | First Draft | Adair Torres |
| 29/10/2022 | 1.1 | Reformatted old document to new document, first iteration of Article 1 and 2 created | Chinh Nguyen |
|  |  |  |  |

Table of Contents

1. Introduction 4

1.1 Purpose 4

1.2 Scope 4

1.3 Definitions, Acronyms, and Abbreviations 4

1.4 References 4

1.5 Overview 4

2. Architectural Representation 4

3. Architectural Goals and Constraints 4

4. Use-Case View 4

4.1 Use-Case Realizations 5

5. Logical View 5

5.1 Overview 5

5.2 Architecturally Significant Design Packages 5

6. Interface Description 5

7. Size and Performance 5

8. Quality 5

# Introduction

## Purpose

The purpose of this document is to a detailed architecture design of the ChessEDU application. It serves as a means of communication between the entire development team regarding the overlying architecture of ChessEDU and the decisions made. Through this document, the overall systems, architectural views, and structures in place should be clearly defined and understandable. In essence, it is a technical blueprint for the software.

## Scope

This document applies to a variety of architectures within ChessEDU which include, but are not limited to:

* Application Architecture – *how the client will interact with our product*
* Data Architecture – *handling of data such as lessons and practice modules within database*
* Integration Architecture – *the delivery of API’s and data from database to client application*

## Definitions, Acronyms, and Abbreviations

List of Terms:

* JavaScript: Primary programming language used to deliver content to client through a website
* HTML: A standard markup language for documents intended to be displayed in website
* JSP: Server-sided technology used to create web applications and dynamic web content
* Python: An interpreted, object-oriented, high-level programming language
* Flask: A web application framework in the Python programming language
* SQL: A domain-specific language, used for handling and managing data in database

Acronyms:

* SAD: Software Architecture Document
* API: Application Programming Interface – a method for two or more systems to communicate with each other

More specific terminology from ChessEDU and chess itself can be found within this project’s Glossary found within the file “chessedu\_gloss.docx” under Article 2.

## References

No references were used.

## Overview

This document contains information regarding the general architecture of ChessEDU and overlying details of the project’s organizational structure.

This document is organized through multiple articles and subsections (Articles 2 through 8) immediately after this subsection (Article 1 Subsection 5).

# Architectural Representation

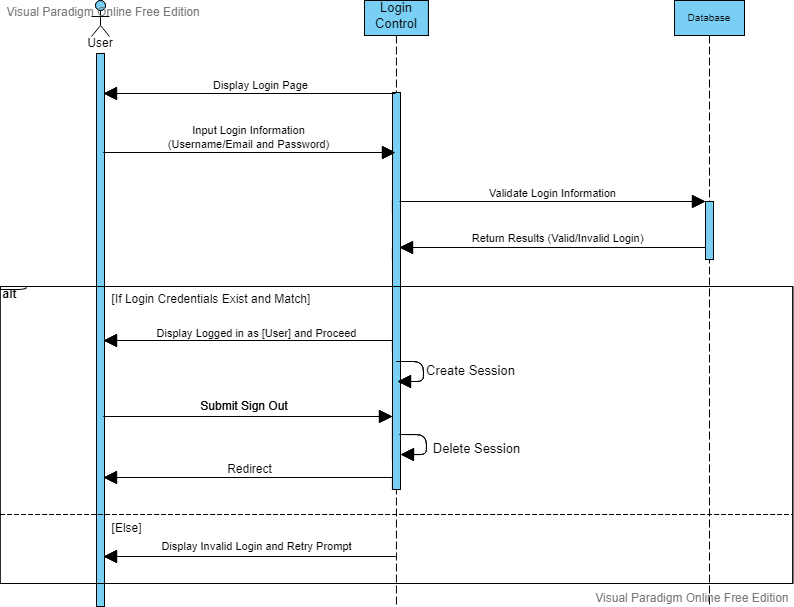


Figure 3: UML Sequence Diagram for Login/Logout System

Figure 4: UML Sequence Diagram for Requesting and Using a Lesson

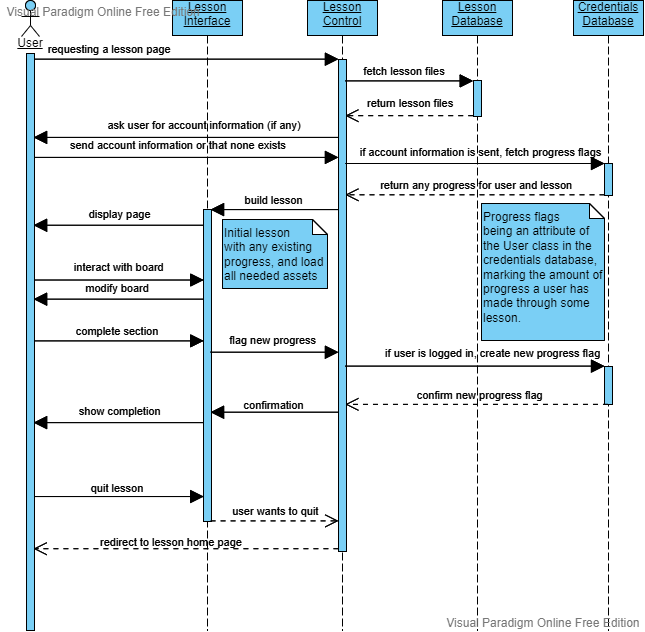


Figure 2: UML Class Diagram for Flask

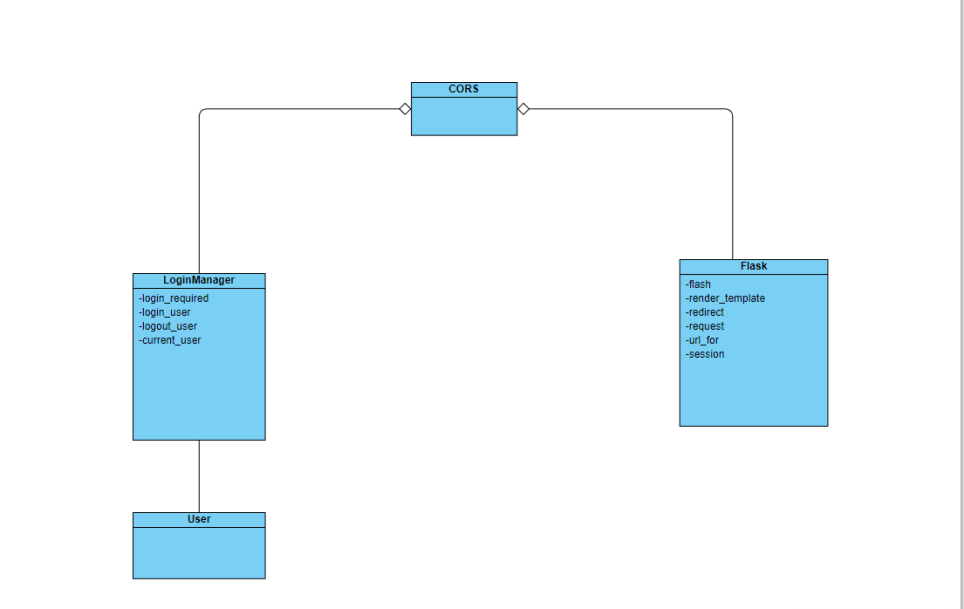
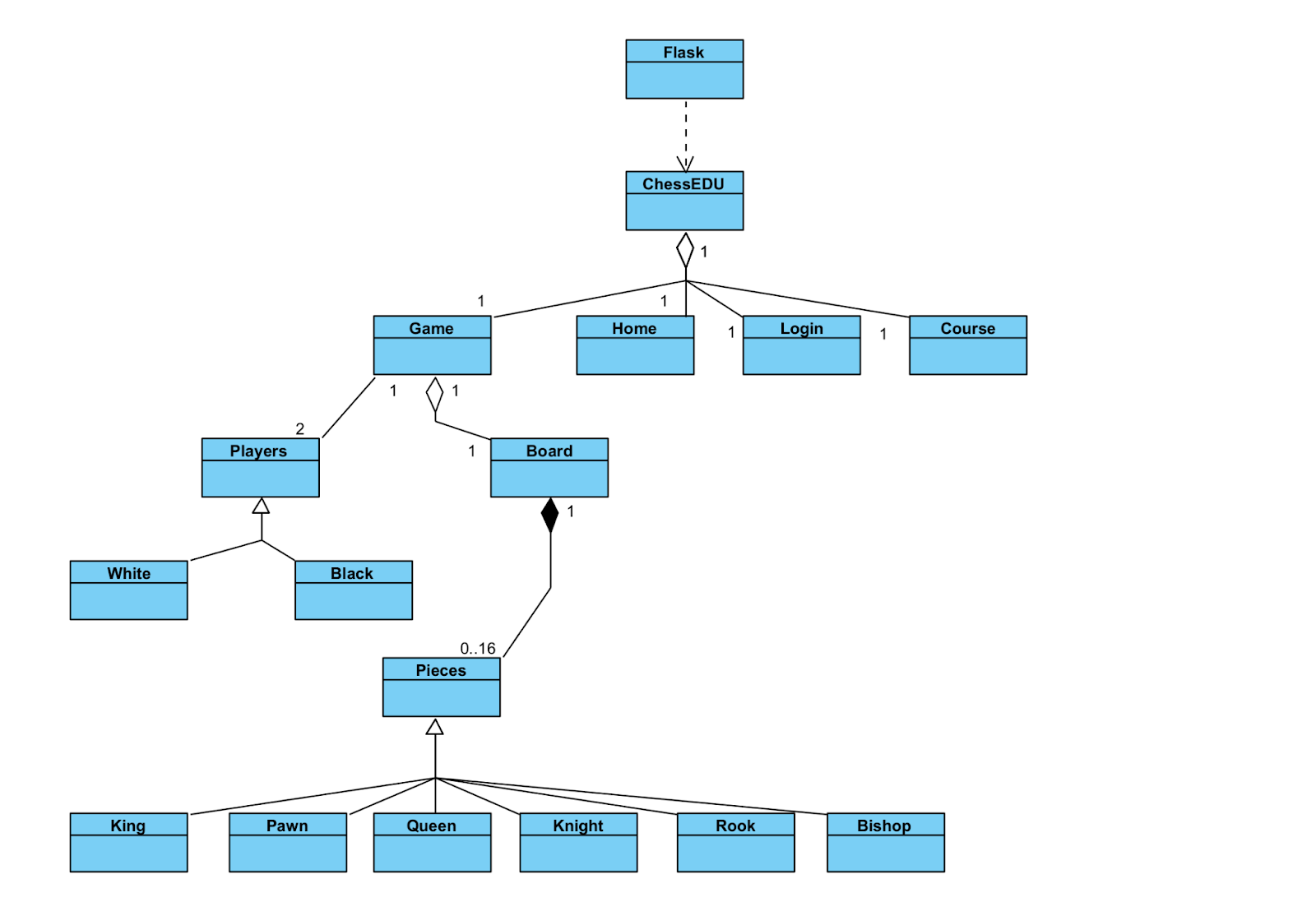


Figure 1: UML Class Diagram for ChessEDU Application



The diagrams above outline the central architecture of the system through UML diagrams. Figure 1 shows how the application will be split up into the major system components.

# Architectural Goals and Constraints

[This section describes the software requirements and objectives that have some significant impact on the architecture; for example, safety, security, privacy, use of an off-the-shelf product, portability, distribution, and reuse. It also captures the special constraints that may apply: design and implementation strategy, development tools, team structure, schedule, legacy code, and so on.]

# Use-Case View

[This section lists use cases or scenarios from the use-case model if they represent some significant, central functionality of the final system, or if they have a large architectural coverage—they exercise many architectural elements or if they stress or illustrate a specific, delicate point of the architecture.]

## Use-Case Realizations

[This section illustrates how the software actually works by giving a few selected use-case (or scenario) realizations, and explains how the various design model elements contribute to their functionality. If a Use-Case Realization Document is available, refer to it in this section.]

# Logical View

[This section describes the architecturally significant parts of the design model, such as its decomposition into subsystems and packages. And for each significant package, its decomposition into classes and class utilities. You should introduce architecturally significant classes and describe their responsibilities, as well as a few very important relationships, operations, and attributes.]

## Overview

[This subsection describes the overall decomposition of the design model in terms of its package hierarchy and layers.]

## Architecturally Significant Design Packages

[For each significant package, include a subsection with its name, its brief description, and a diagram with all significant classes and packages contained within the package.

For each significant class in the package, include its name, brief description, and, optionally, a description of some of its major responsibilities, operations, and attributes.]

# Interface Description

[A description of the major entity interfaces, including screen formats, valid inputs, and resulting outputs. If a User-Interface Prototype Document is available, refer to it in this section]

# Size and Performance

[A description of the major dimensioning characteristics of the software that impact the architecture, as well as the target performance constraints.]

# Quality

[A description of how the software architecture contributes to all capabilities (other than functionality) of the system: extensibility, reliability, portability, and so on. If these characteristics have special significance, such as safety, security or privacy implications, they must be clearly delineated.]