LearningEDU

ChessEDU
Supplementary Specifications

Version <2.0>

Revision History

Date	Version	Description	Author
25/10/2022	1.0	First Draft	Adair Torres
27/10/2022	2.0	Applied modifications as requested by the Lab Attendant	Adair Torres

Table of Contents

Supplementary Specifications

1. Introduction

1.1 Purpose

Supplementary specifications capture the requirements which aren't easily defined within the UseCase Model. Requirements such as: legal standards, quality aspects, reliability, supportability, and execution criteria of the system.

1.2 Definitions, Acronyms, and Abbreviations

Browser

A browser is a software which allows the user to visualize3 and interact with all information presented and flowing through the internet.

Engine

In a software or in a computer, Engine is the term used for smaller programs executing specific functionalities and useful tasks for other programs or software.

Flask

A python module used to establish and control a web application / Engine through HTTP queries and python code. The information and data are controlled by the Engine first and then displayed in its final shape to the end-user through HTML and Jinja templates.

Jinja

A templating engine with special placeholders that allow writing code similar to Python syntax in an HTML web page. A template is passed data to render in the final HTML document.

SQLArchive

A SQLite database specifically formatted to track file data. Entries should include the full file path to a file and data associated with the file, such as access permissions, last modification time, original file size, and any compressed content.

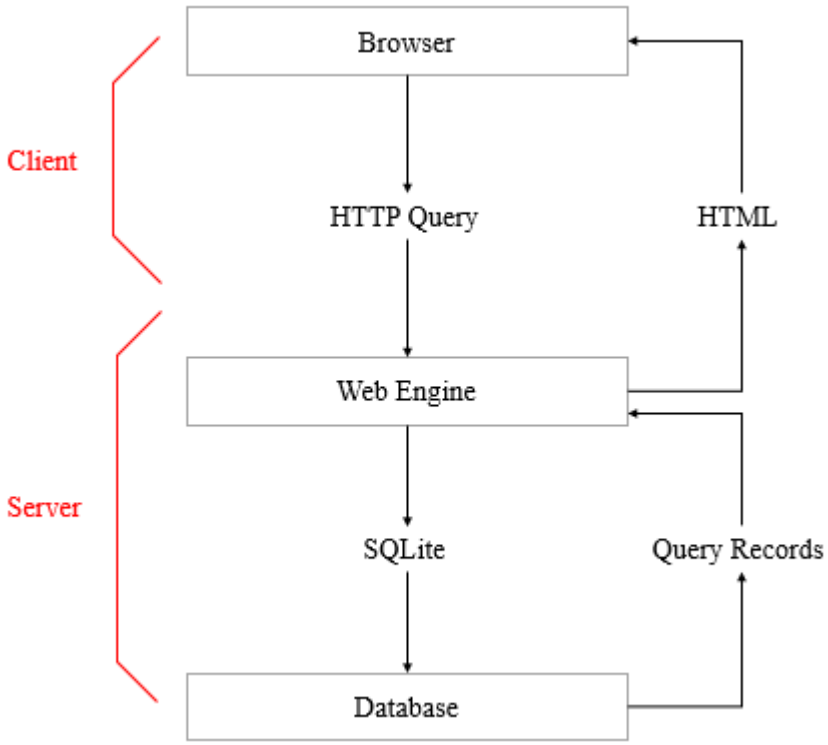
1.3 References

Course Web Page: https://people.eecs.ku.edu/~saiedian/Teaching/448/

2. Supplementary Specifications

2.1 Client / Server

The user interacts with the system through a web browser (Google Chrome, Safari, Mozilla Firefox, etc.). The browser executes HTML and JavaScript queries to the Web Engine through Flask application routing of HTTP methods. The Engine communicates with the Database using SQLite language. Finally the database returns the data to the Engine, which, in turn, gives back the requested package to the browser. The browser then shows the results to the end-user.



2.2 Usability

2.2.1 Graphical Interface

All interactions between a user and the software are made through a graphical interface. Each system functionality must be accessible by mouse or keyboard. The user must be able to choose a specific branch from the system and then obtaining the results on the screen.

2.2.2 User's Knowledge

Targeted users are familiar with the use of web Browsers on the platform / operating system they are utilizing. The tool is designed to be user-friendly, and any user should not require training. Targeted users may have varying levels of understanding and familiarity with the game of chess.

2.2.2.1 Accessibility

The software should include accessibility features for users with disabilities. This includes but is not limited to a narrator to read course modules, announce game piece movements, and board states; a magnifier to increase visibility of web pages; and adaptive controls for users with restricted motor functions.

2.3 Reliability

2.3.1 Responsiveness

The application shall be able to capture, execute, and respond to all user entries. All false entries, such as invalid or restricted HTTP queries, shall be validated and returned to the user without abnormally stopping the application.

2.3.2 Bug / Defect Tolerance

The application must have absolutely no bugs or defects when executing a move in a game of chess. All the rules of Chess must by fully implemented without bugs in order to prevent accidental illegal moves and player frustration. Move and turn logging must also have zero defects in order to maintain the accuracy of game logs.

The application may display some web pages improperly due to certain unexpected screen sizes, aspect ratios, and display formats. However, the application should display consistently across web browser windows, even after a window is resized.

The application should not return the incorrect course module or lesson to the user. The Web Engine should only attempt to locate and track module files through the SQLArchive SQLite database to reduce any chance of a bug resulting from improper file retrieval / storage.

2.4 Performance

2.4.1 Execution Speed

The application will execute itself on a Windows-based platform and response time for all tasks shall be inferior to 1.5 seconds. However, this time may vary depending on the number of users, and request load on the server.

2.4.2 Resource Use

The application will require access to a significant and possibly growing amount of storage memory as more course modules are added by contributors.

2.5 Supportability

2.5.1 Platforms / Operating Systems

Client side can be installed on any platform. While the Server side and Web Engine can be installed on any MacOS, Linux, or Windows system, the current installation targets a Windows-based platform.

2.5.2 Maintenance Access

Maintenance of the software is solely accessible by the LearningEDU developer team. Maintenance of a course / lesson module's content is accessible to Lesson Author users and Lesson Developers.

2.6 Design Constraints

2.6.1 Programming Languages Used

JavaScript and HTML for all web content; Python for Web Engine / Application implementation, SQLite for database management queries.

2.7 Security

2.7.1 Risks

The type of data with the most significant risk are passwords and credentials. As some users reuse passwords, a security breach and leakage of user credentials can lead to accounts not associated with our software becoming compromised.

2.7.2 User Data Minimization

Gather data on individual users should be minimized as much as possible to reduce the liability and risk tied to a data leakage. The bare minimum for any given user is their associated credentials and course module data, including completed and in-progress courses.

2.7.3 Encryption

At minimum, user credentials should be encrypted. This can be either encryption before being placed within the credentials database, or the complete encryption of the credential database.

2.7.4 Credential Requirements

Users should be required to use lengthy and complex passwords for their associated credentials. A user should be required to use a password that includes uppercase and lowercase letters, numbers, and symbols. A password must be at least 12 characters long. At signup, a user should be given tips for creating a strong and memorable password.

2.8 Interfaces

2.8.1 Localhost:<port#>

This interface is used for testing and debugging on developer systems. This is the basic standard for running a Client / Server system on one's own computer and prevents outside interaction so long as the individual device's IP or hostname are not shared. The application is set to use the default port 5000, but can be changed.