# PLC Editor

## Group 1

**INF19C**
**DHBW Stuttgart**

# Software Architecture Specification


Team members:
Mouaz Tabboush, Elian Yildirim, Franziska Kopp, Leonie De Santis

Author:
Mouaz Tabboush



Version 1.0

Last Edited: November 5, 2020

**Revision History**

| Version | Description of Versions / Changes | Responsible Party | Date |
|---|---|---|---|
| 1.0 | Initial version | Mouaz Tabboush | 11/03/2020 |

**Approval Block**

| Version | Comments | Responsible Party | Date |
|---|---|---|---|
| | | | |
| | | | |
| | | | |
| | | | |

## Table of Contents

# Software Architecture Document

## 1. Introduction

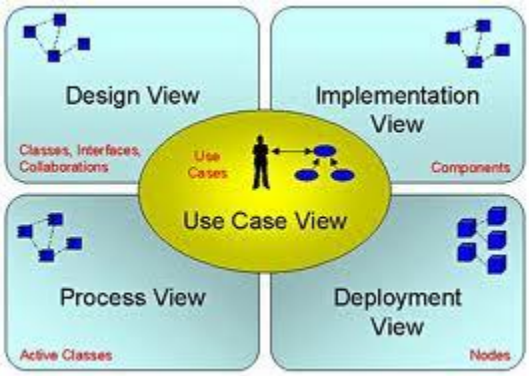
This document provides a high-level overview and explains the architecture of the PLC Editor as a Web Application

The document defines goals of the architecture, the use cases supported by the system, architectural styles and components that have been selected. The document provides a rationale for the architecture and design decisions made from the conceptual idea to its implementation.

### 1.1. Purpose

The Software Architecture Specification (SAS) provides a comprehensive architectural overview of the PLC Editor as a Web Application. It presents a number of different architectural views to depict the different aspects of the system.

In order to depict the software as accurately as possible, the structure of this document is based on Philippe Kruchten's "4+1" model view of architecture [Kruchten].




The "4+1" View Model allows various stakeholders to find what they need in the software architecture.

### 1.2. Scope

The scope of this SAS is to explain the architecture of the PLC Editor as a Web Application.

This document describes the various aspects of the PLC Editor Web Application design that are considered to be architecturally significant. These elements and behaviors are fundamental for guiding the construction of the Web Application and for understanding this project as a whole. Stakeholders who require a technical understanding of the PLC Editor are encouraged to start by reading the Project Proposal, Concept of Operations and Software Requirements Specification documents developed for this system [PP, ConOps, SRS].

### 1.3. Definitions, Acronyms, and Abbreviations

- **PLC** – Programmable Logic Controller
- **XML** – Extensible Markup Language, a file-format used to transfer data
- **SFC** - Sequence Flow Chart
- **FBD** - Function Block Diagramm
- **POU** - Project Organization Unit(s) contained inside the XML file.

### 1.4. References

[BC]: Business Case

[PSP]: Project Structure Plan

[SRS]: Software Requirements Specification

[MedBiquitous]: Document Template, https://projects.cecs.pdx.edu/attachments/download/3146/Software_Architecture_Document.docx

[Kruchten]: The "4+1" view model of software architecture, Philippe Kruchten, November 1995, http://www3.software.ibm.com/ibmdl/pub/software/rational/web/whitepapers/2003/Pbk4p1.pdf

### 1.5. Overview

In order to fully document all the aspects of the architecture, the Software Architecture Document contains the following subsections.

Section 2: describes the use of each view

Section 3: describes the architectural goals and constraints of the system

Section 4: describes the most important use-case realizations

Section 5: describes logical view of the system including interface and operation definitions.

Section 6: describes significant persistence elements.

Section 7: describes how the system will be deployed.

## 2. Architectural Representation

This document details the architecture using the views defined in the "4+1" model [Kruchten]. The views used to document the DMM system are:

### Use Case view

**Audience**: all the stakeholders of the system, including the end-users.

**Area**: describes the set of scenarios and/or use cases that represent some significant, central functionality of the system. Describes the actors and use cases for the system, this view presents the needs of the user and is elaborated further at the design level to describe discrete flows and constraints in more detail. This domain vocabulary is independent of any processing model or representational syntax (i.e. XML).
**Related Artifacts** : [BC]

### Logical view

**Audience**: Designers.
**Area**: Functional Requirements: describes the design's object model. Also describes the most important use-case realizations and business requirements of the system.
**Related Artifacts**:

### Data view

**Audience**: Data specialists, Database administrators
**Area**: Persistence: describes the architecturally significant persistent elements in the data model as well as how data flows through the system.
**Related Artifacts**:

### Deployment view

**Audience**: Deployment managers.
**Area**: Topology: describes the mapping of the software onto the hardware and shows the system's distributed aspects. Describes potential deployment structures, by including known and anticipated deployment scenarios in the architecture we allow the implementers to make certain assumptions on network performance, system interaction and so forth.
**Related Artifacts**:

## 3. Architectural Goals and Constraints

There are some key requirements and system constraints that have a significant bearing on the architecture. They are:

1. The Application is meant as an online tool for building and editing PLCopen projects. the primary stakeholders in this document and the system as a whole are end-users as is normally the case. As a result the primary Goal of the Application is to provide an intuitive and easy to use editing tool.

2. The system will be written using Angular technologies but will also use an XML for data Transfer and will be deployed as a Stand-alone Client. These special deployment requirements require additional consideration in the development of the Application.

3. The system must import and export PLCopen Projects using XML and additionally an AML wrapper. Defining how the Application work with these XML-Files is a primary concern of the architecture.

## 4. Use-Case View

The purpose of the use-case view is to give additional context surrounding the usage of the system and the interactions between its components. For the purposes of this document, each component is considered a use-case actor. Section 4.1 lists the current actors and gives a brief description of each in the overall use context of the system. In section 4.2, the most common use-cases are outlined and illustrated using UML use-case diagrams and sequence diagrams to clarify the interactions between components.

### 4.1. Actors

**User**

The user will drive all operation of the software. No distinction is made in regards to type of user. The user interacts with all available interfaces to initiate all application operations.

**Angular Client**

The Angular client serves to aid the user in general navigation of the application.

**Editor Page**

The editor page on the Angular Client is used to develop PLC Functions or programs and supports editing in SFC and FBD.
Changes made in the editor is applied to the XML file when saving

**Project Overview Page**

The Project overview page helps the user navigate through the POUs inside the XML File and edit the file on an organizational level.

**The Parser**

The Parser component reads the POUs Structure inside the XML file and generate a JSON to be interpreted by the Project Overview Page

**Xml File**

XML files are used to transfer all information related to a PLCopen Project.

## 4.2. Use-Case Realizations

### 4.2.1. Navigate Project

Users are able to upload their Projects to navigate through them and edit them on a organizational level

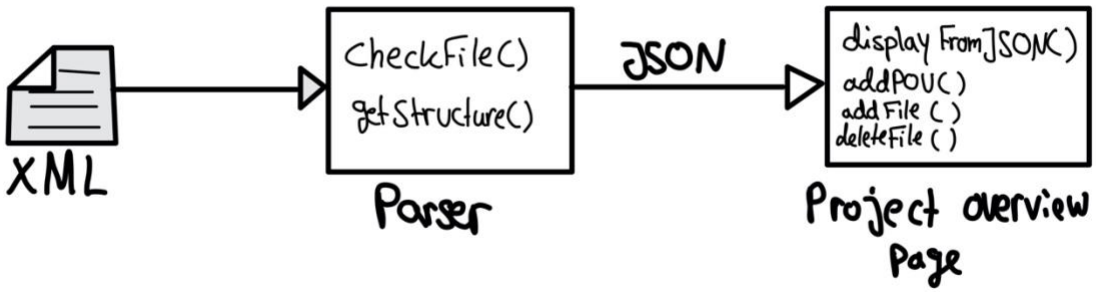



**Figure 4.1** Navigate Project Use Case Diagram

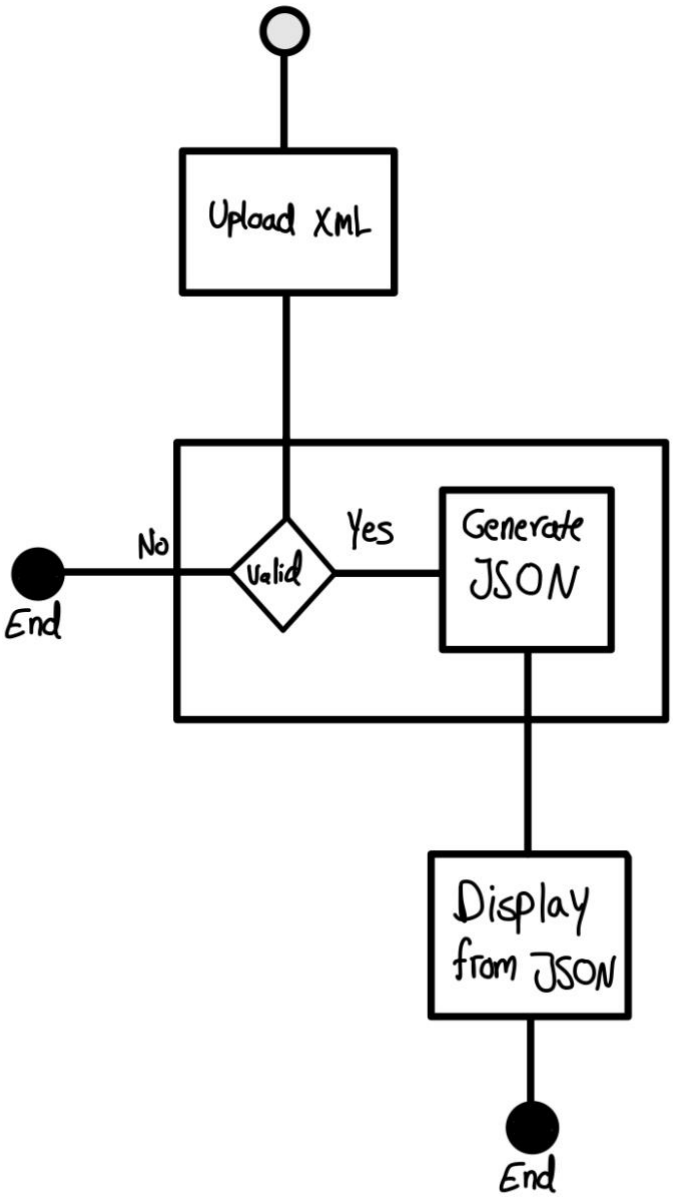



**Figure 4.2** Navigate Project Sequence Diagram

### 4.2.2. Drag & Drop Components

The User Drags a Component from a list to add it to their program at a specific place.

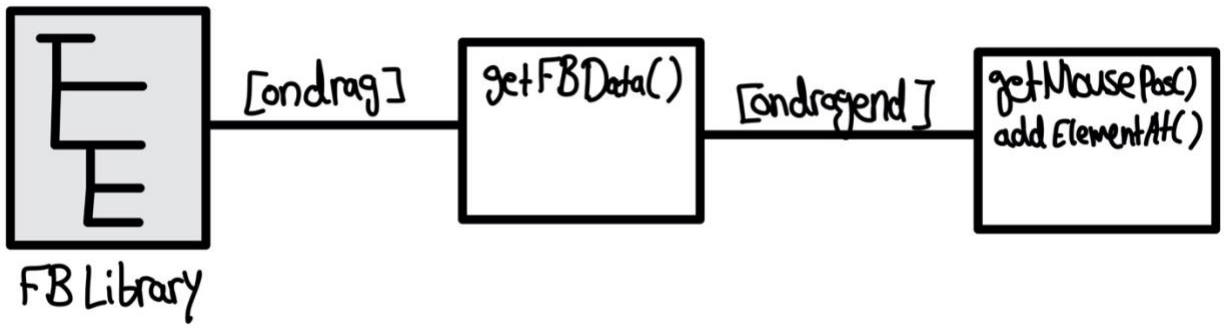



**Figure 4.3** Drag & Drop Use Case Diagram

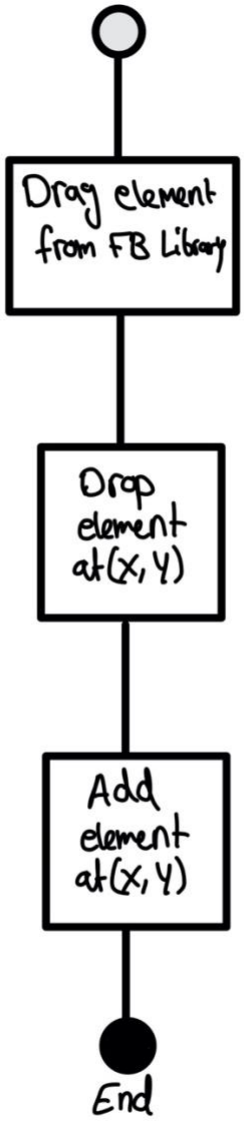



**Figure 4.4** Drag & Drop Sequence Diagram

### 4.2.3. Save Changes

The User save their changes into the XML File by clicking on the save button.


**Figure 4.5** Save Changes Use Case Diagram

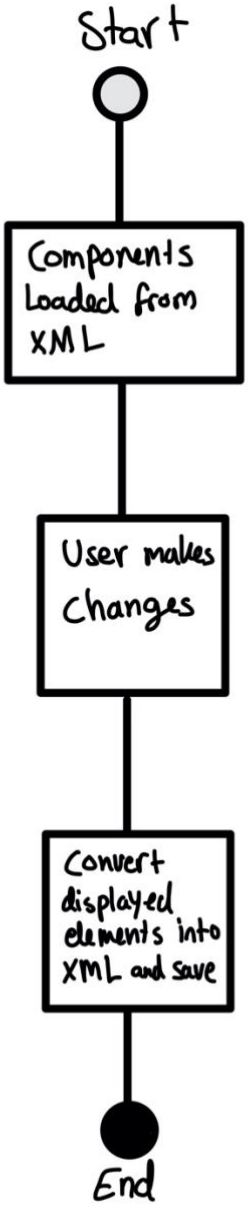



**Figure 4.6** Save Changes Sequence Diagram

### 4.2.4. Connectors

User connects components in the program.

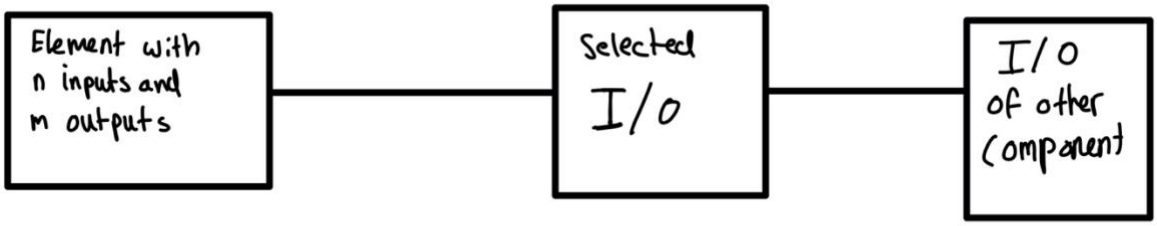



**Figure 4.9** Connectors Use Case Diagram

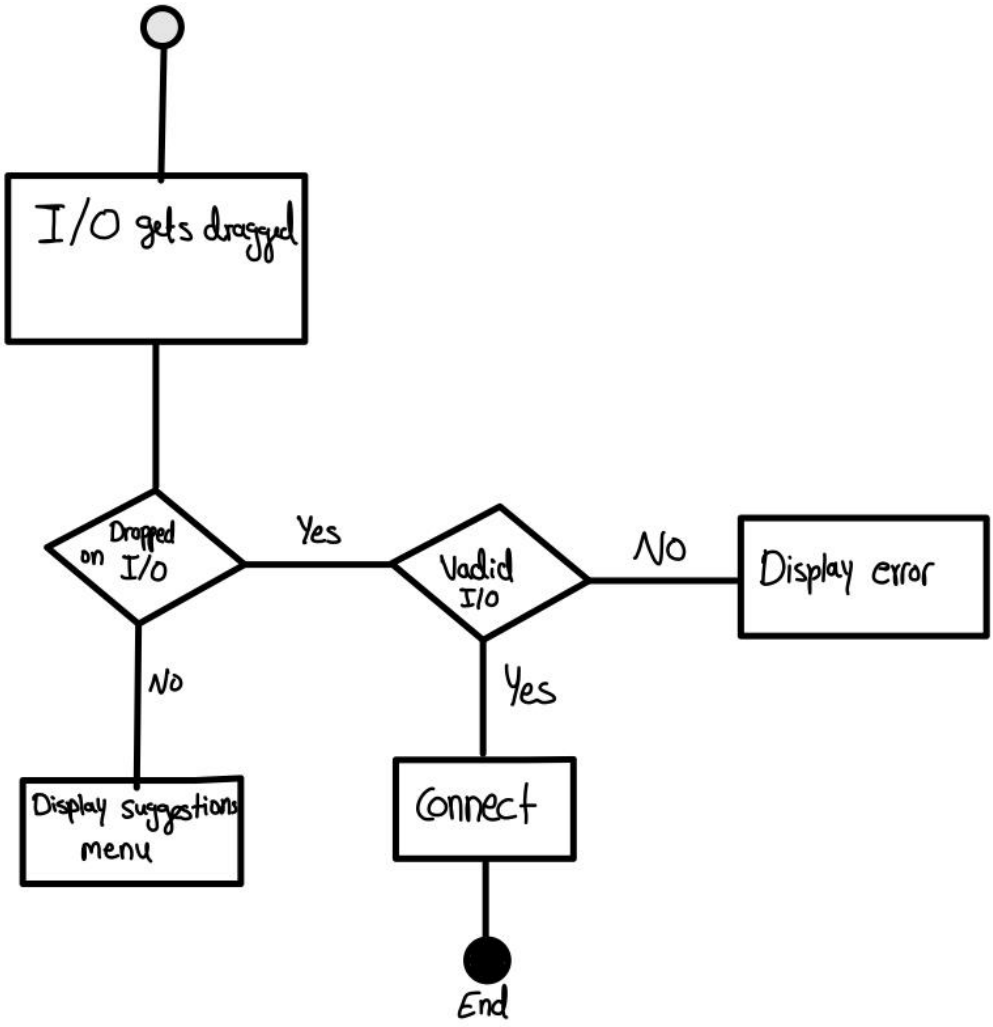



**Figure 4.10** Connectors Sequence Diagram

## 5. Logical View

### 5.1. Overview

The main goal of the logical view is to define the components that will make up the Application and to define the interfaces through which they will communicate and interact with one another. The primary decision-making factor behind defining the components is the need to isolate the components that are likely to change from the rest of the Application. By clearly defining the interfaces of these components and hiding their internal implementations from the rest of the Application, the impact of unexpected changes can be minimized.

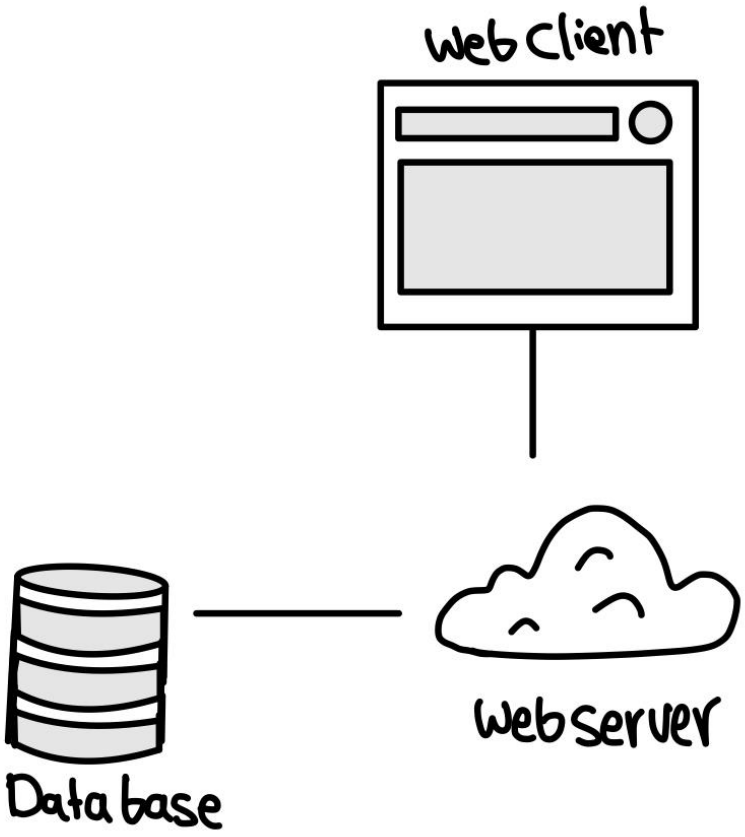



**Figure 5.1** Logical Component Diagram.

**Table 5.1** Element Responsibilities

| Element | Responsibilities |
|---|---|
| Webclient | • Upload the XML File to the Server<br>• Navigate the Project<br>• Edit in SFC and FBD |
| Webserver | • Save the XML File<br>• Save changes into the XML File<br>• Upload the XML File to the Client |
| Database | • Store the XML File |

## 6. Data View

This diagram illustrates the static data structure and relationships of the main entities that will be stored by the application during Project overview and Editing. Each element nominally represents a database table. Relationship cardinality is denoted with UML multiplicity notation.

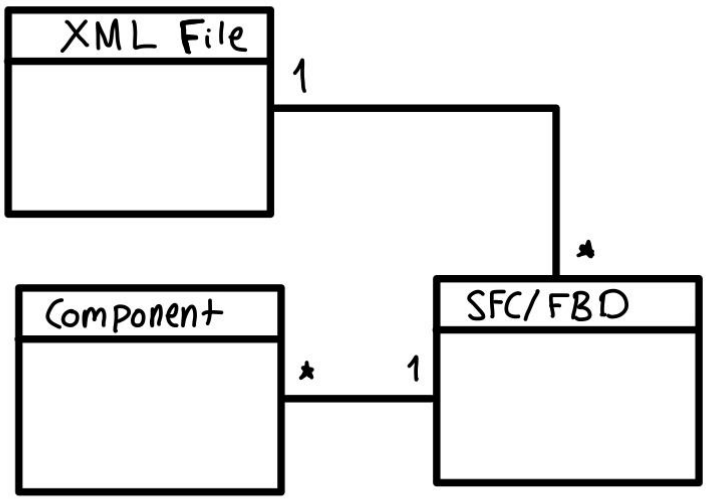



**Figure 8.1** Static Data Structure Diagram

## 7. Deployment View

The web application will be hosted on a local server. An Apache webserver running a mono module will be used to serve the application pages.
The application's deployment specifics can be seen below.

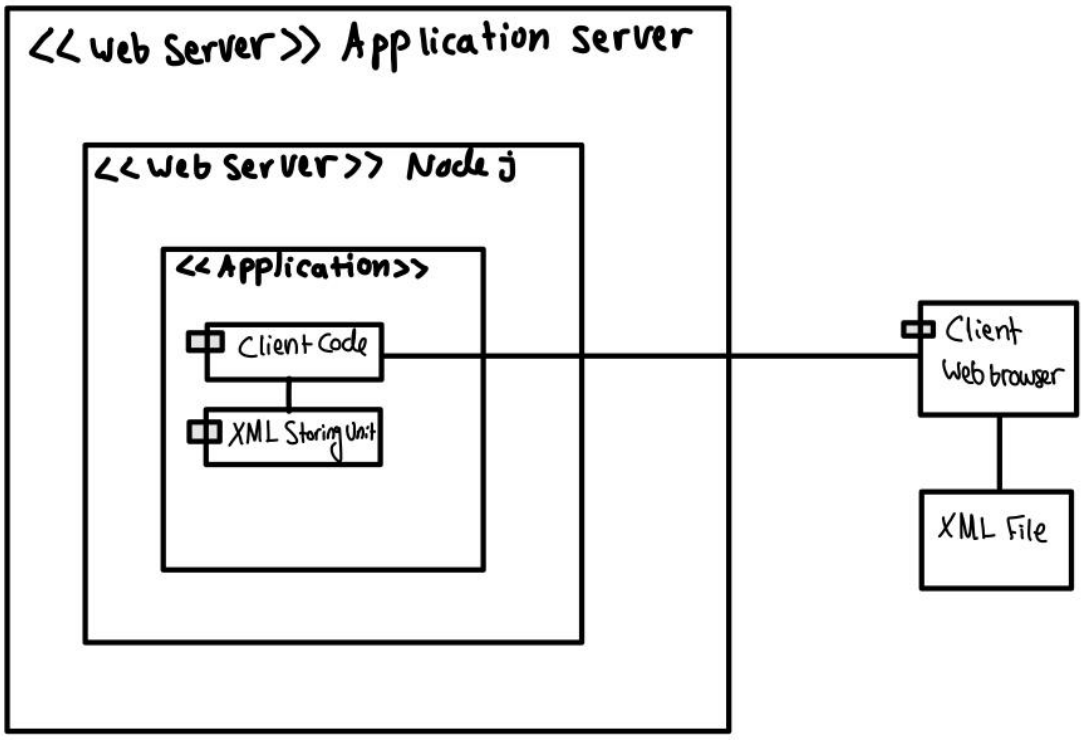



**Figure 7.1** Deployment View Diagram