Architecture and Design Specification

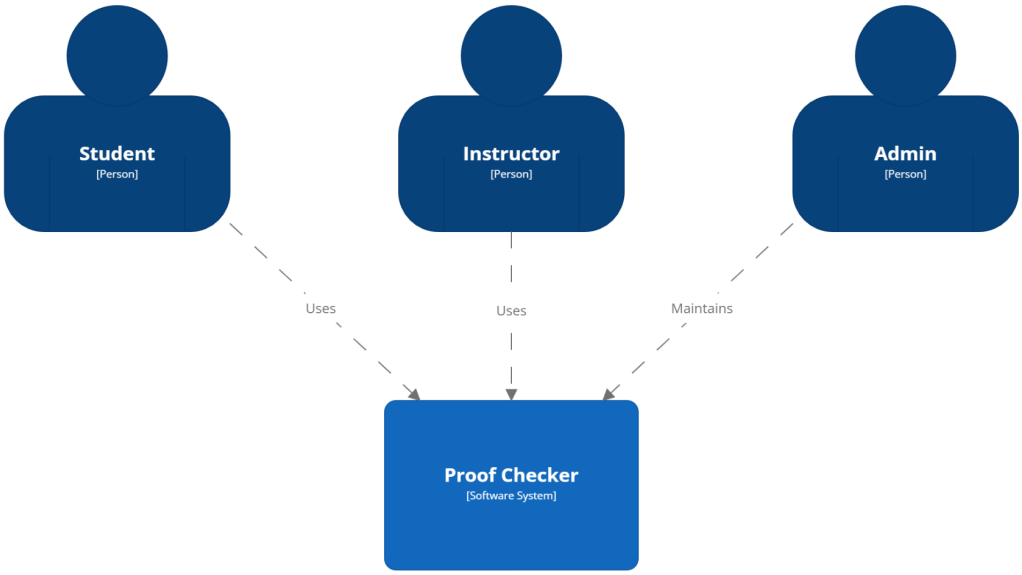
Proof Checker Tool - SE691

Rakhfa Amin, Thoman Andrews, Kersley Jatto, Colton Shoenberger

Introduction

The following architecture specification uses the C4 model (https://c4model.com/) for visualizing software architecture. The C4 model defines four levels of abstraction, ordered from highest to lowest: Context, Container, Component, and Class (Code). Diagrams have been created using the Structurizr tool (https://structurizr.com/)

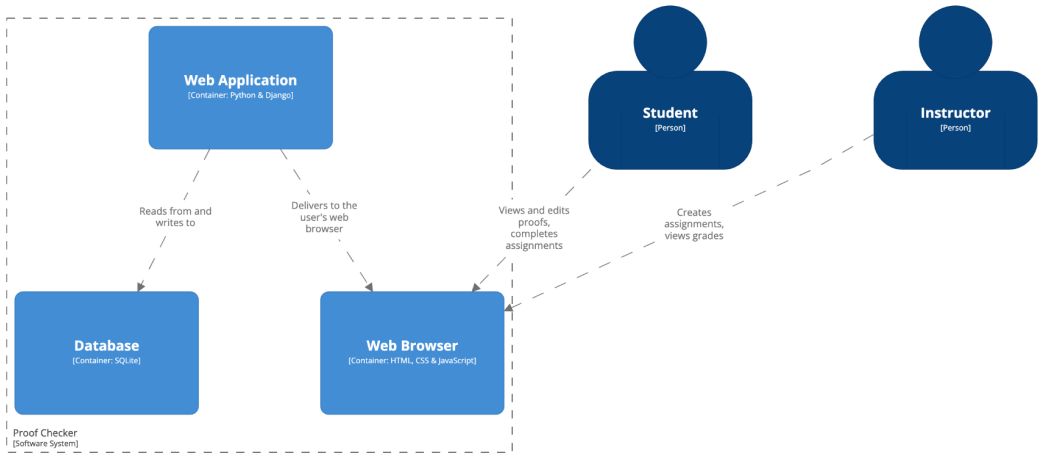
Context Diagram



System Context diagram for Proof Checker
Thursday, December 2, 2021, 10:50 AM Eastern Standard Time

Beginning with the highest level of abstraction, our Context Diagram presents a starting point for understanding how our software system fits into the world around it. The primary users (or actors) involved with our system will be students and instructors. Administrative users will also be involved for development and maintenance of the system. The current name we are utilizing for the system is “Proof Checker”.

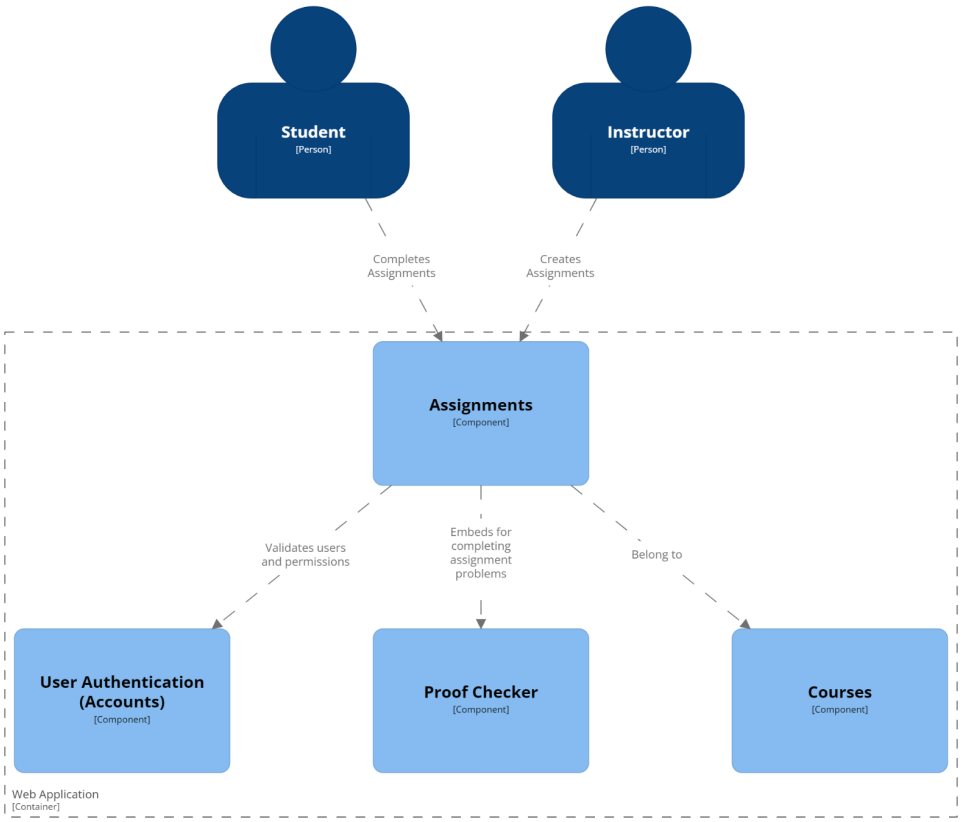
Container Diagram



Container diagram for Proof Checker
Tuesday, October 12, 2021, 7:53 PM Eastern Daylight Time

In our Container Diagram, we decompose the architecture of the software system slightly into “containers”, which are defined in the C4 model as “high-level technical building blocks”. Our primary users will interact with the system through a web browser, which is our first container. The primary technologies involved with the web browser include HTML, CSS, and JavaScript. On the server-side, the web application is delivered to users’ web browsers via our web application, our second container. The web application is developed primarily with Python and the Django framework. Lastly, for storing and retrieving data, our web application interacts with a database, our final container. The technology we are currently using for our database is SQLite, which is a lightweight database efficient for rapid development. As the scale of our project grows, we may need to migrate to a different database technology such as MySQL or PostgreSQL. Fortunately, the Django framework makes migration between different database technologies extremely simple, as it includes automatic handlers for migrating data models to various SQL schemas. Migrating to another database technology will not require us to write any SQL code whatsoever.

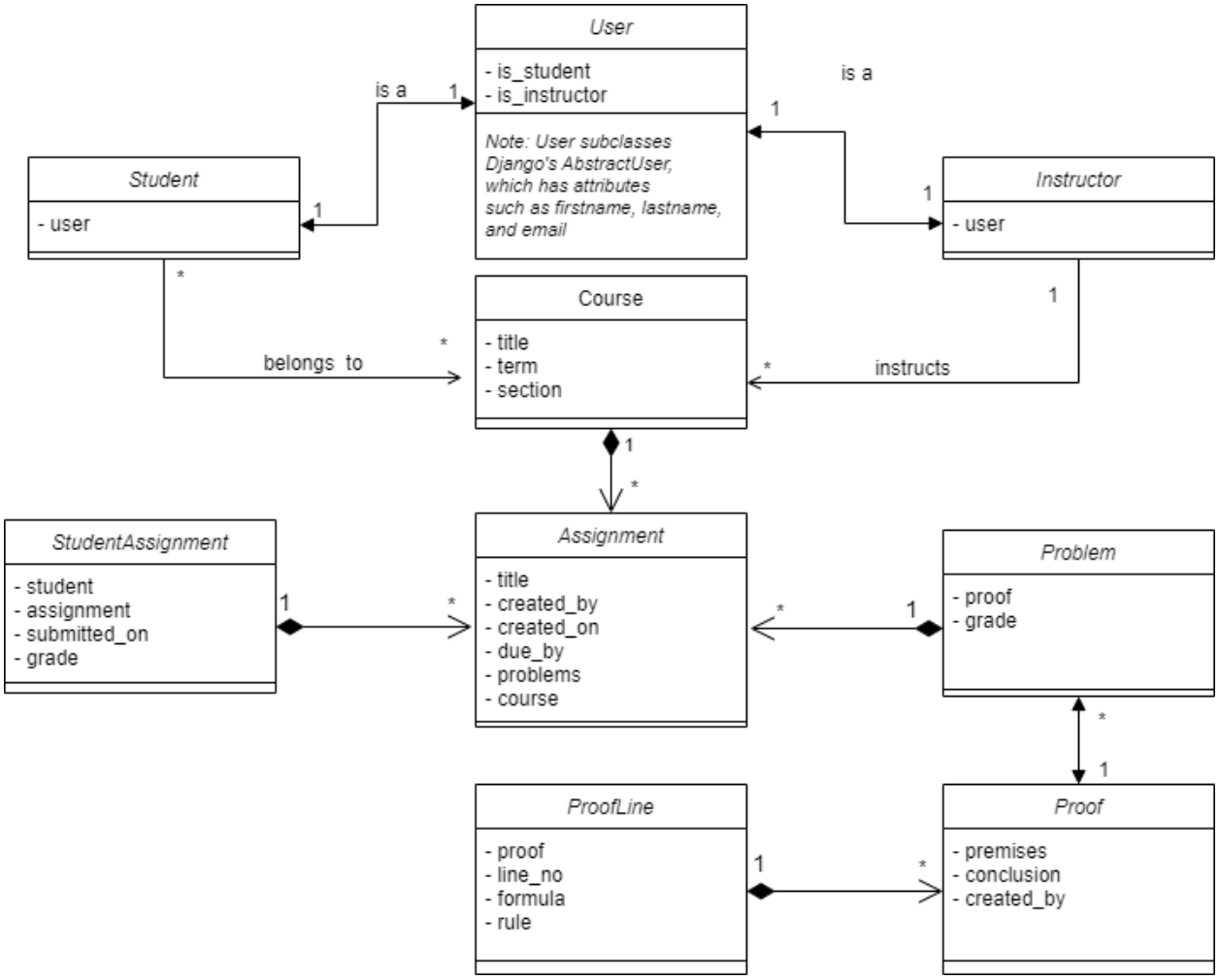
Component Diagram



Component diagram for Proof Checker - Web Application
Tuesday, December 7, 2021, 10:03 AM Eastern Standard Time

In the Component Diagram, we further decompose our web application container into various components, or internal elements. The primary component that we have been developing throughout this term is our “Proof Checker” component. The Proof Checker is essentially a web form that users can use to create, edit, save, and delete mathematical proofs. Additionally, the Proof Checker component provides feedback and validation of the proofs submitted to the tool. Our Proof Checker is intended to be an embeddable resource that will be utilized by other components of the system such as Assignments. The Assignments component is the primary component that we expect users to interact with. Instructors will upload assignments for their students, which students will complete and submit for grading. Development of the Assignments component is our primary objective for the upcoming term. Lastly, the User Authentication (or “Accounts”) component is responsible for validating users of the system and permission to access various resources within the system.

Class (Code) Diagram



Lastly, our lowest level of abstraction is the Class Diagram, which is simply a Unified Modeling Language (UML) depiction of the main classes (data models) in our system (https://www.uml.org/) . The illustration above provides all of the classes that are currently being persisted to our database. The attributes of each class are portrayed, as well as the relationships between the objects.

It is worth noting that the data objects in our system may expand in the upcoming term. Importantly, we plan to develop a “Rule” object that represents a mathematical rule that can be applied within a proof. The rule object will include a callable method to verify whether it is being applied correctly within a proof. This redesign of our current rule verification will make the system more reusable and extensible. Additionally, it will allow instructors to select which rules may be applied within an assignment, which is a requirement of our system.