Architecture and Design Specification

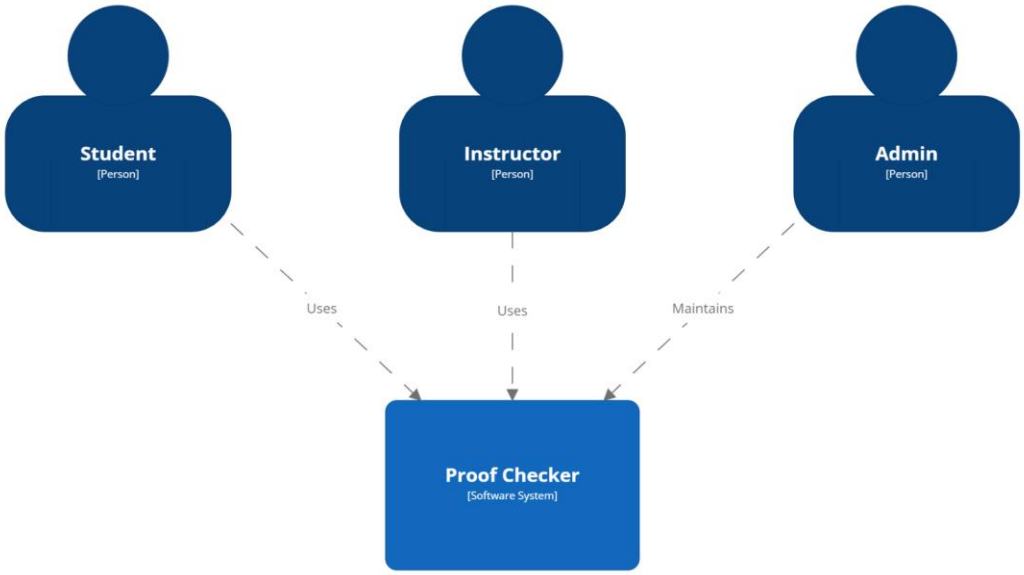
ProofChecker - SE691

Rakhfa Amin, Thomas Andrews, Kersley Jatto, Aiasha Sattar, 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/)

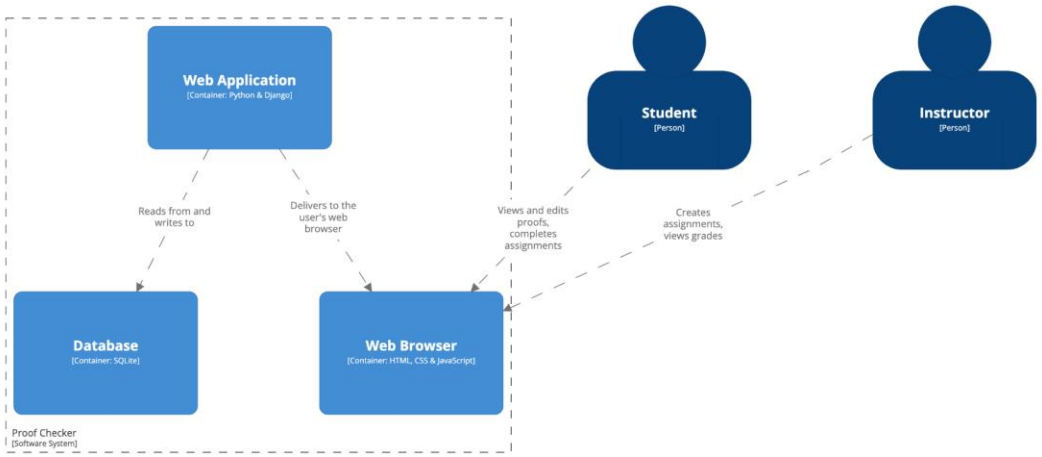
Context Diagram



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 “ProofChecker”.

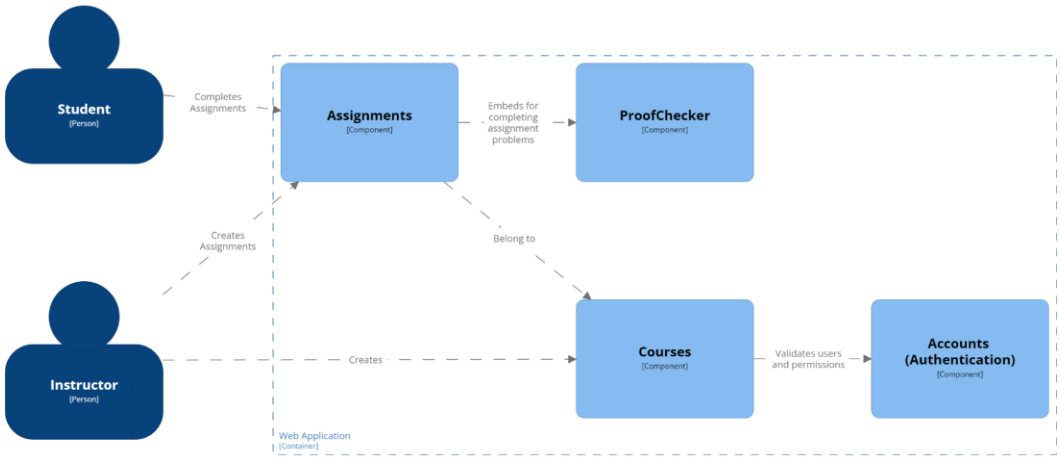
Container Diagram



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 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

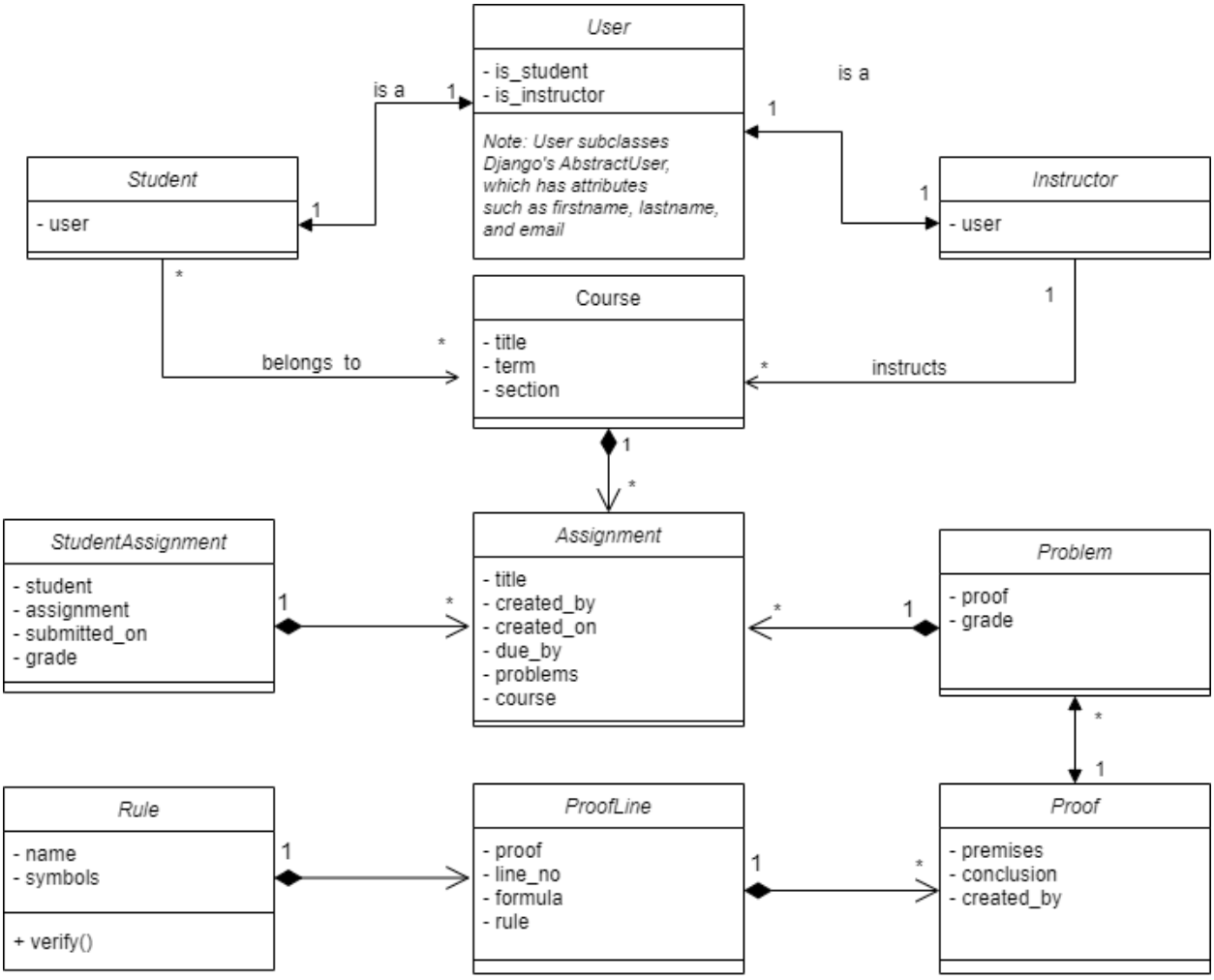


[Component] Proof Checker - Web Application
Thursday, March 10, 2022, 7:30 PM Eastern Standard Time

In the Component Diagram, we further decompose our web application container into various components, or internal elements (these are known as “apps” when using Django terminology). There are four components (apps) within our system: Accounts, Courses, Assignments, and ProofChecker. The Accounts component handles account creation (registering as a new user), user authentication (logging in and out), and handling permission to access resources and functionality. Users can register their account as either a Student or an Instructor. The Courses component allows Instructors to create courses and enroll students in their courses. Students can also self-enroll in available courses. The Assignments component allows Instructors to create assignments for their students to work on. Assignments can consist of one or more problems, which are essentially incomplete proofs. Instructors can distribute assignments to students within a course. The students can then access the Assignments component to begin working on these assignments. They can edit and save their solutions.

Lastly, we have the ProofChecker component. The ProofChecker component is essentially a web form that allows users to create, edit, save, and delete mathematical proofs. Users can add, delete, and reorder lines within a proof, and can even create subproofs within a proof. Users can reference the rules of Truth Functional Logic (TFL) and First Order Logic (FOL) within their proofs. The ProofChecker also provides feedback and validation of the proofs submitted to the tool. The ProofChecker is utilized to create and complete assignment problems.

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.