Design Specification Document for ProofBuddy

Team Name	Group 3
Team Members	Nicole Itchon, Raphael Perez, Viet Pham, Iftekhar Rahman
Stakeholders	Steve Earth, Jeremy Johnson
Date Submitted	March 17, 2022

Contents

1. Introduction

1.1. System Purpose

ProofBuddy is an educational tool for teaching computer science students proof techniques and logical reasoning. The system is web-based and designed for use by both instructors and students. The system will leverage its additional functionalities to compete with existing natural deduction tools that do not exist within these tools.

1.2. System Scope

ProofBuddy will have the following capabilities for instructors, create/edit/delete courses, assignments, problems, and proofs, and view proofs by students. The system will have the following capabilities for students, join courses, view/submit assignments, work through problems, and create/edit/delete proofs. The ProofBuddy system requires users to create an account to utilize its functionalities. The system allows all users to report bugs and provide feedback to the team through its interface.

1.3. System Design Goals

Our goal for the system includes stability, robustness, scalability, and extensibility. We want to ensure a smooth and consistent user experience. To achieve that, both frontend and backend designs of the system are throughout built and tested to achieve the core functionalities as well as rail-guard corner cases. We also build the system with an agile mindset. Each core function must be small and robust. By doing so, the system is easy to maintain and extend with new functionalities in the future.

1.4. Definitions

Natural deduction – technique where logical reasoning is expressed by inference rules closely related to the “natural” way of reasoning

2. System Overview

2.1. Architecture Diagram

The system architecture below is shown following the C4 model (reference provided in section 5 of report). The model uses 4 levels of abstraction: Context, Container, Component and Code (class). Each level has an associated diagram which is provided below. The diagrams were created with draw.io technology (referenced in section 5) and leveraged from the previous group’s documentation (when applicable). The Code model, while it is a part of the C4 model, is discussed in section 3.

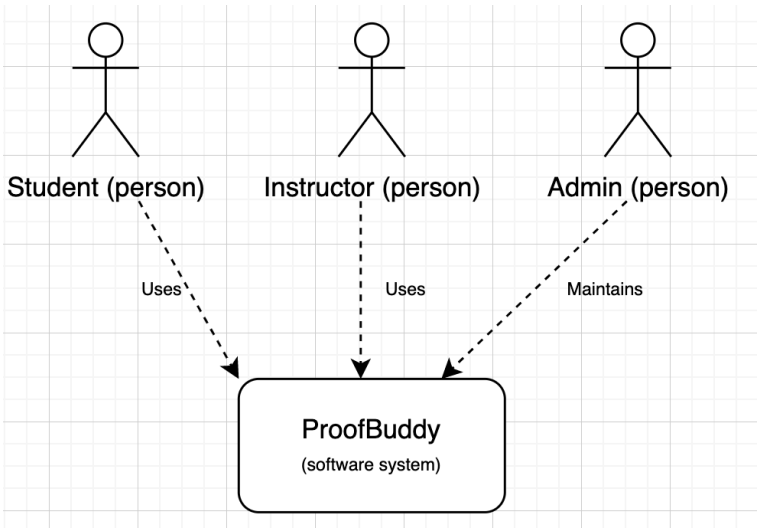

Figure 1: Context Diagram

The context diagram is the highest level of abstraction which provides the scope of the system and how relates to other systems and the world. Our context diagram involves our software system, ProofBuddy. It also includes its users, the students and instructors. Lastly, it includes admins to maintain the software the software.

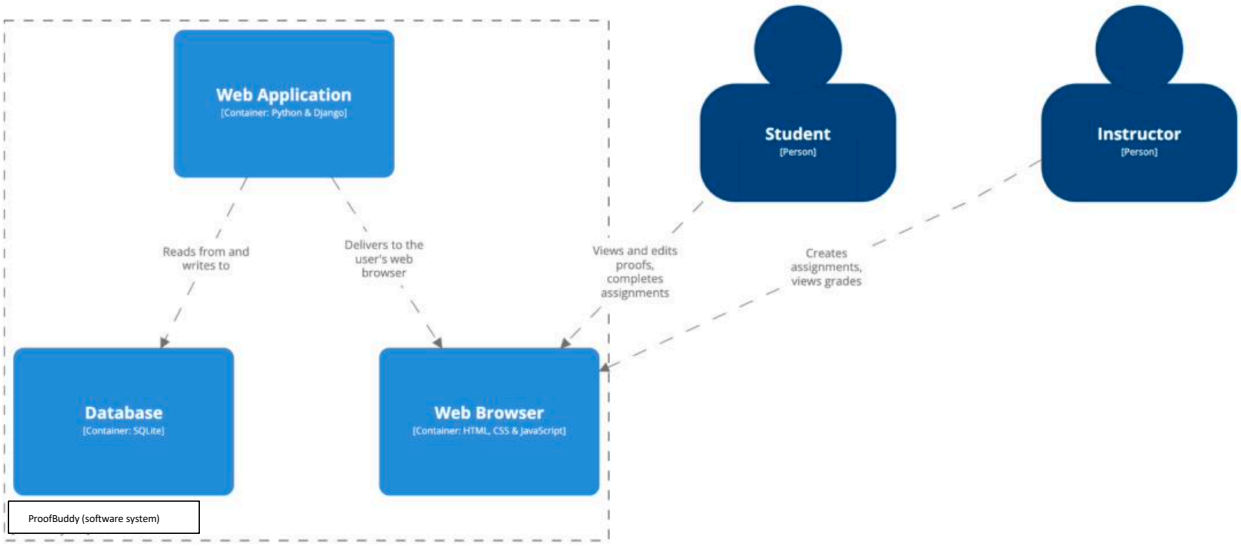

Figure 2: Container Diagram

The next model in the C4 model is the container diagram shows the 3 main containers. The Web Browser is the first, and it comprises of HTML, CSS and JS as technologies used. The web application is the second container, and it is developed using Python and Django. For storing and retrieving data, we have our final container, which is our database. We use an SQLite database due to it being lightweight but also scalable.

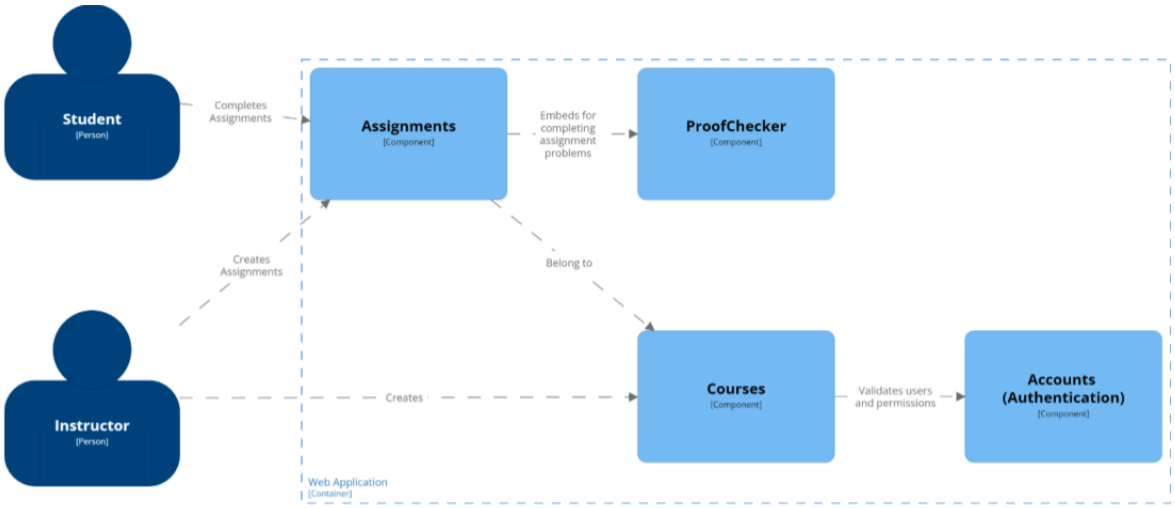

Figure 3: Component Diagram

In the component diagram, the system is further analyzed. Our project involves 4 main components (apps in Django terminology): Accounts, Courses, Assignments and ProofChecker.

The Accounts component is responsible for account creation, user authentication and permission handling. Users can be either students or instructors (admins will also be instructors) and sign up accordingly. The courses component allows Instructors to create new courses and enroll students onto those courses (Students can also self-enroll into courses). The assignment component allows instructors to create assignments in courses and assign incomplete proofs (problems) with premise and conclusion pre-set. Students can save their work and submit the assignment for grading. Finally, there is the ProofChecker component allows Student or Instructors to create, edit, save and submit proofs. Users can recreate subproofs or modify proofs by adding, deleting and reordering lines. Users can reference TFL or FOL rules in their proofs and get feedback on their proofs as they progress.

2.2. Supporting Technology

The Django framework is the core technology of the entire application. Django is a high-level Python web framework that is designed to fasten development of web applications while maintaining a high level of clean and extensible architecture. By using Django, we can extend the platform that the previous developers have left for us and continue to leave an expandable platform for future developers or anyone who wants to contribute to the project once the repo goes public.

Bootstrap is the UI cornerstone of Proof Buddy. Meshing Django with Bootstrap allows the application to present itself in a very readable and user-friendly format to the level that allows the application to serve as a learning tool. By using the Bootstrap framework, we can

rapidly add or remove features with minimal disruption or downtime so any efforts towards the application can be used to further enhance it.

The technology used to further enhance the UI experience was a framework called Django Widget Tweaks. Django does not allow users to change the form/table layout without using external extensions. With Django Widget Tweaks, we're able to clean up the proof table form UI to make full use of its screen space and make the line numbers do not appear editable because users are not able to anyway. We're also able to better communicate what the user can do while using the proof buddy engine without having to explicitly state it. The Django Widget Tweaks tool allowed us to guide the user in understanding the functions of the table the way any well designed UI should.

2.3. High Level Design and Implementation

The Proof Buddy application is equipped with extendable classes and objects that allows the code to be easily maintainable and extendable, as previously stated. The Proof Buddy engine itself is composed of separate Python files labeled with the different components of the proof buddy engine such as Syntax.py, which helps maintain all the different input possibilities of the proof buddy checker. Another example of this would be the valid input file that allows us to easily see what would constitute as a valid text input in the proof buddy engine and keep the user from entering any unknown input.

We stayed true to this system by making sure whatever extensions we have made; it would be easy for the next addition to those features would have the ability to be extended without breaking the original code. We have made sure all our extensions were modular so that it could also be used in other sections in future code if it were needed. As the bugs from the feedback were being fixed, they are also robust enough to withstand any further possible issues, so the users do not have to worry about the application breaking on them.

For the UI, the goal was to enhance the usability of the application, so the user doesn't feel intimidated by the complex nature of mathematical proofs. The application should serve as a segway to the concepts of proof as an encouraging learning tool and users should not have to fight the UI or the proof engine. With that in mind, we went out of our way to make sure the UI elements were easy to read and easy to collapse any element that could be collapse so the user can create a space that works for them, not against.

Finally, as we wrapped up this quarter, we were able to load the application to Tux since Heroku was no longer providing a free tier for the application. The rollout of the Tux implementation went very smoothly thanks to the modern Python and Django technologies. While we work towards having the ability to access Proof Buddy while off the Drexel VPN, at least for now there's still a viable way for student to access the application.

In addition to Tux, we do plan to shift the Proof Buddy into a public repository that can be used to extend Proof Buddy in anyway a person can see fit. The control of the pull request will go to Steve Earth and hopefully, we'll have an audience large enough to see the potential of Proof Buddy's growth to justify making the application source code publicly available.

3. Detailed System Architecture

3.1. Class Diagram

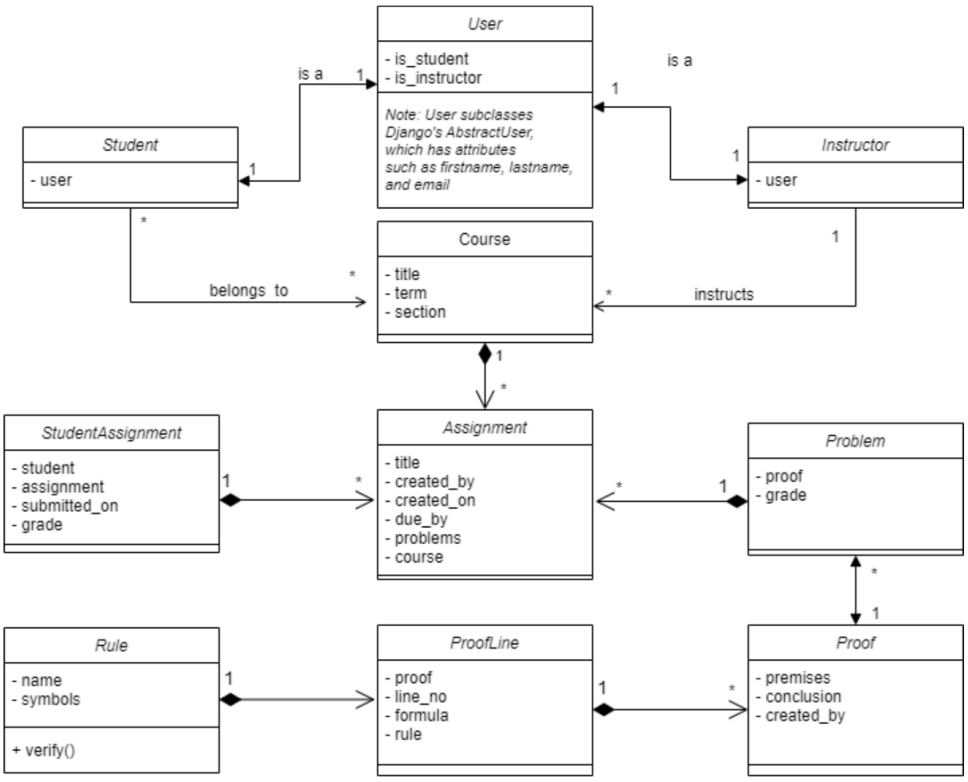

Figure 4: Class Diagram

The class diagram above, which we leveraged from the previous team's documentation, shows the different classes in the project. The diagram depicts the attributes and methods within each class, as well as the relationships between classes. This class diagram followed the Unified Modeling Language, which is a standardization to visualize the design of a system.

4. Database Overview

4.1. Database Schema

The database is designed to have the same schema as the above class diagram.
The relationship between each entity is as the following:

- Each user is either a student or an instructor
- Each user belongs to one or more courses
- Each course has one or more assignments
- Each assignment is either a student assignment or a problem
- Each problem has one or more proofs
- Each proof has one or more proof line
- Each proof line has one rule

5. References

Architecture and Design Specification from Team 1
https://c4model.com/
https://en.wikipedia.org/wiki/Natural_deduction