





Proof Checker

Rakhfa Amin, Thomas Andrews, Kersley Jatto and Colton Shoenberger



Project Description

Goal: Implementing a web-based platform for teaching students proofs

	Platform	Scalable and extendable system for professor and student use to introduce and test computer science topics.
	Admin	User authentication, course creation, adding students to courses, problem creation, assignment creation, grading, grade storing, grade exporting, viewing progress, and toggling settings on assignments.
	Individual Tools	Seamless creation of additional tools, hints, immediate feedback, topic resources and explanations, automatic grading, exporting problems.
	User Experience	Easy navigation, quick access to resources, clear instructions.

Team Member Roles

Rakhfa Amin

- ❑ Lead back-end developer
- ❑ Lead designer of user authentication
- ❑ Lead database engineer

Kersley Jatto

- ❑ Lead designer of client-side proof validation

Thomas Andrews

- ❑ Lead front-end developer
- ❑ Lead designer of User Interface (UI) and User Experience (UX)

Colton Shoenberger

- ❑ Team lead
- ❑ Lead of testing and quality assurance (QA)
- ❑ Lead designer of server-side proof validation

Requirements

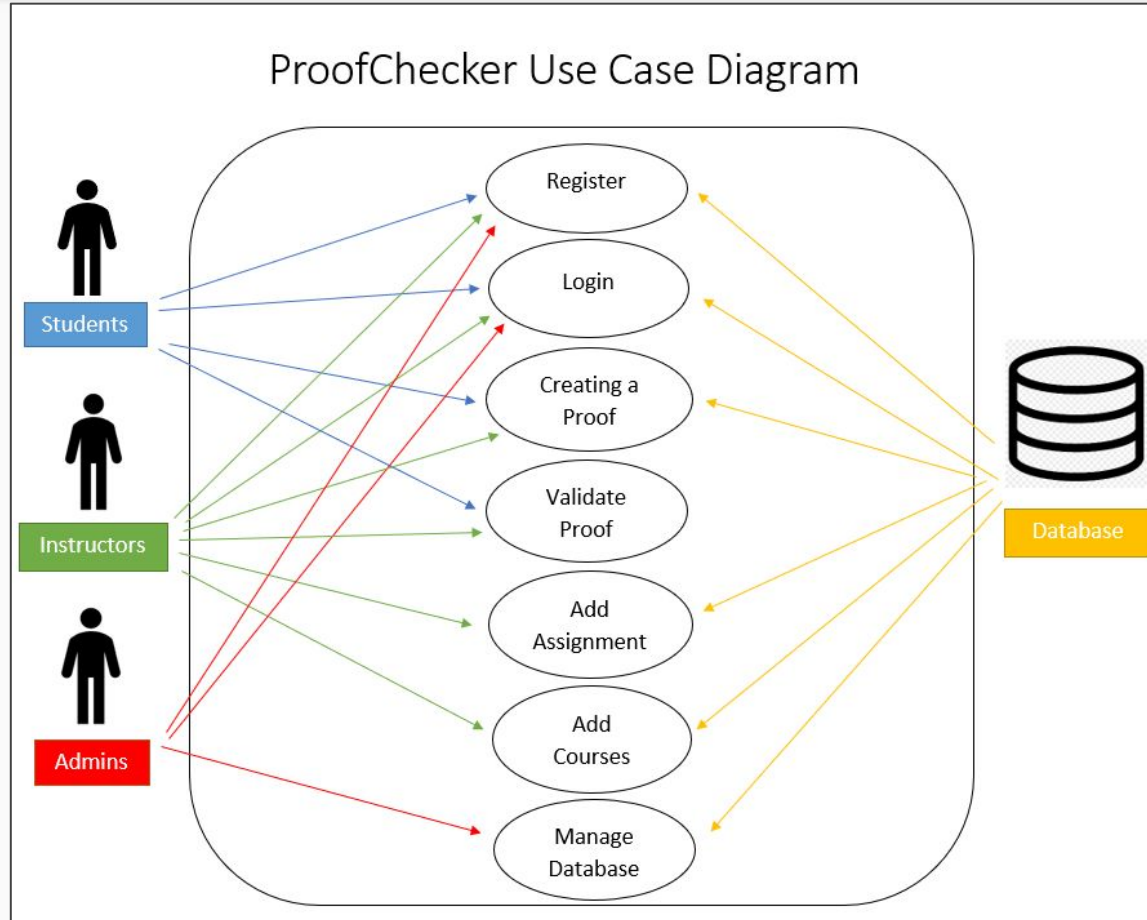
System Requirements

- The system will be operated as a web-based proof checking tool for users.
- The Natural deduction tool would allow users to create proof with reference to Truth-Functional logic.
- The tool shall be able to validate a proof for correctness and completeness.

User Requirements

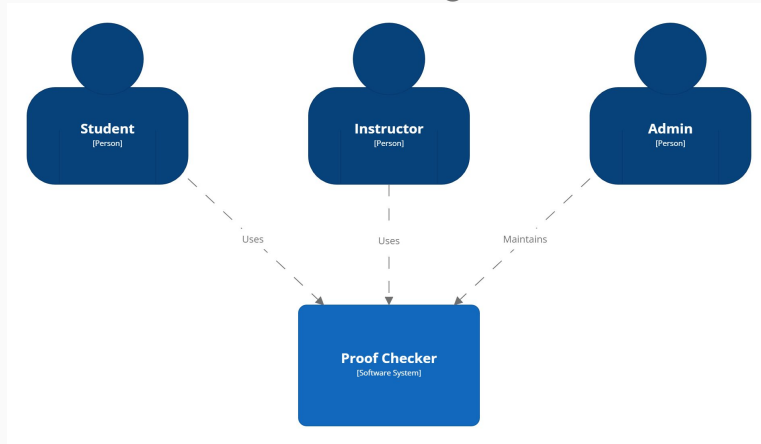
- Users will be able to sign up as either student or professor.
- Users would be able to create, save, edit and delete their work.
- Professors would be able to create/upload/export assignments consists of problems.
- Students should be able to view and work on the assignments for submission that the system automatically grades.

Use Cases

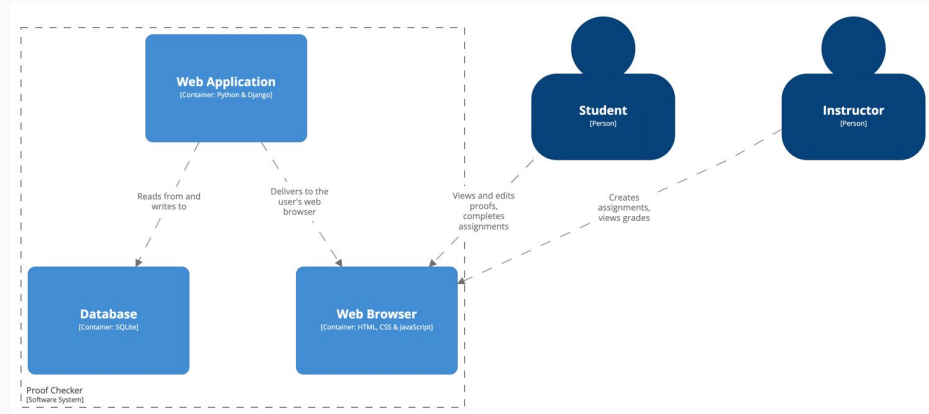


Architecture - High Level

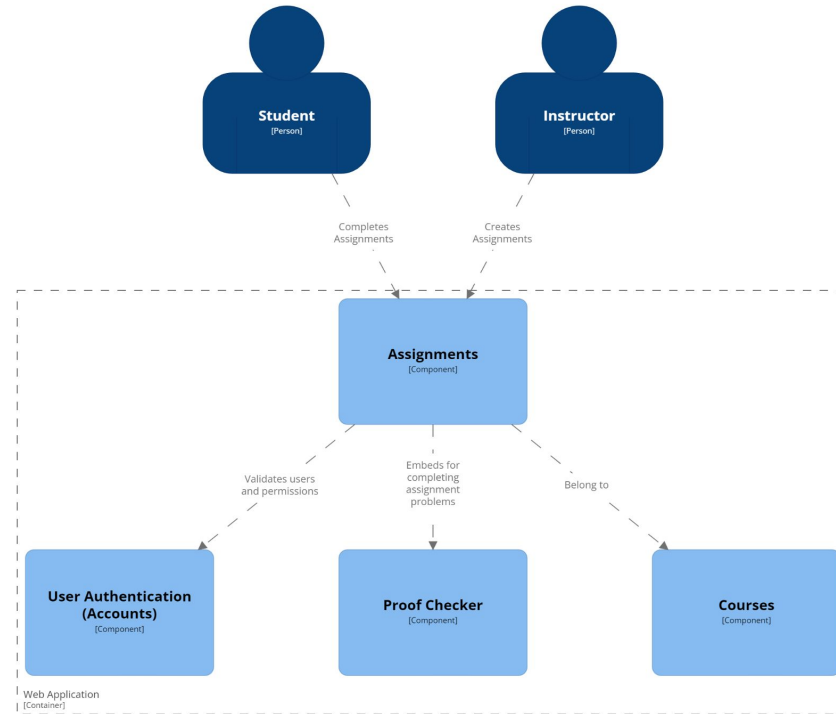
Context Diagram



Container Diagram



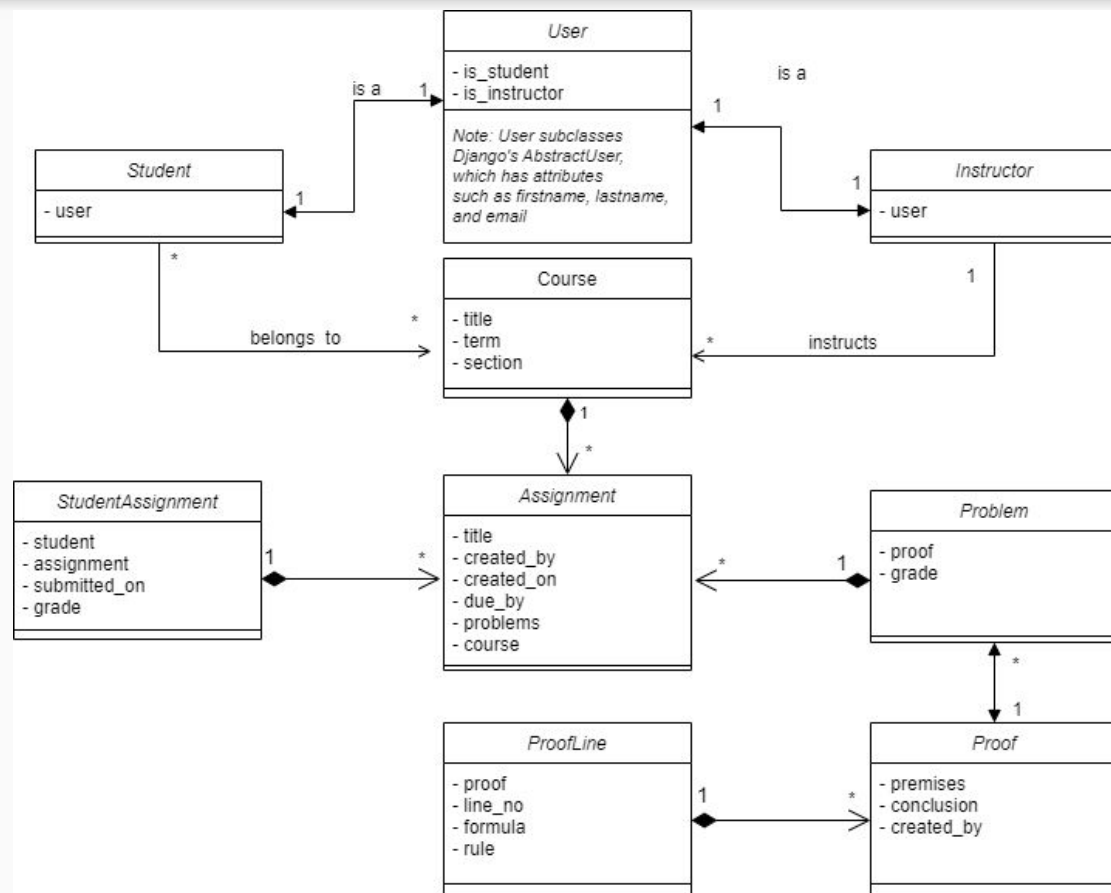
Architecture - Component Level



Component diagram for Proof Checker - Web Application

Tuesday, December 7, 2021, 10:03 AM Eastern Standard Time

Design and Implementation



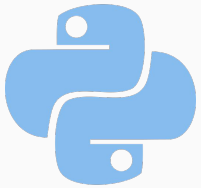
Test Plan

Name	Stmts	Miss	Cover
accounts\apps.py	4	0	100%
accounts\forms.py	29	0	100%
accounts\tests\test_forms.py	12	0	100%
accounts\tests\test_views.py	21	0	100%
accounts\views.py	28	0	100%
proofchecker\admin.py	14	0	100%
proofchecker\apps.py	4	0	100%
proofchecker\forms.py	30	0	100%
proofchecker\models.py	70	5	93%
proofchecker\proof.py	718	91	87%
proofchecker\urls.py	5	0	100%
proofchecker\utils\binarytree.py	104	8	92%
proofchecker\utils\constants.py	8	0	100%
proofchecker\utils\numlex.py	16	1	94%
proofchecker\utils\numparse.py	9	0	100%
proofchecker\utils\syntax.py	96	5	95%
proofchecker\utils\tfllex.py	23	1	96%
proofchecker\utils\tflparse.py	35	0	100%
proofchecker\views.py	133	4	97%
prooftool\settings.py	27	0	100%
prooftool\urls.py	5	0	100%
...			
TOTAL	2592	117	95%

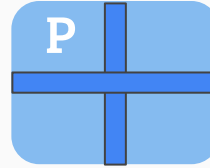
- Using Django's testing framework for unit tests and integration tests
- Using Coverage.py for measuring code coverage
- Currently at **95%** coverage for server-side code

Note: Not all project files included in the report on this slide

Supporting Technology



Python 3.10



pipenv
(Virtual environment)



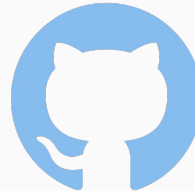
Django 3.2
(Server-Side Web Framework)



Coverage.py
(Code coverage tool)



Python Lex & Yacc
(Parsing tools for Python)

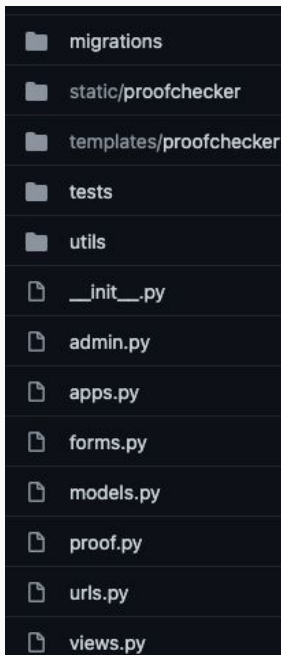
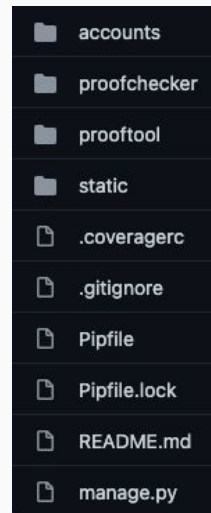


GitHub
(Version control)

Selection of Django

Main Directory

Application



Secure

- Secure management of user accounts and passwords
- Password hash storage
- Default vulnerability protection (SQL injection, cross-site scripting, etc.)



Scalable

- Component-based architecture so each part is independent
- Separation enables easy scaling of servers for increased traffic



Maintainable

- Written with design principles in mind to encourage maintainable and reusable code
- Groups related functionality into reusable applications (as seen on left)

Project Timeline

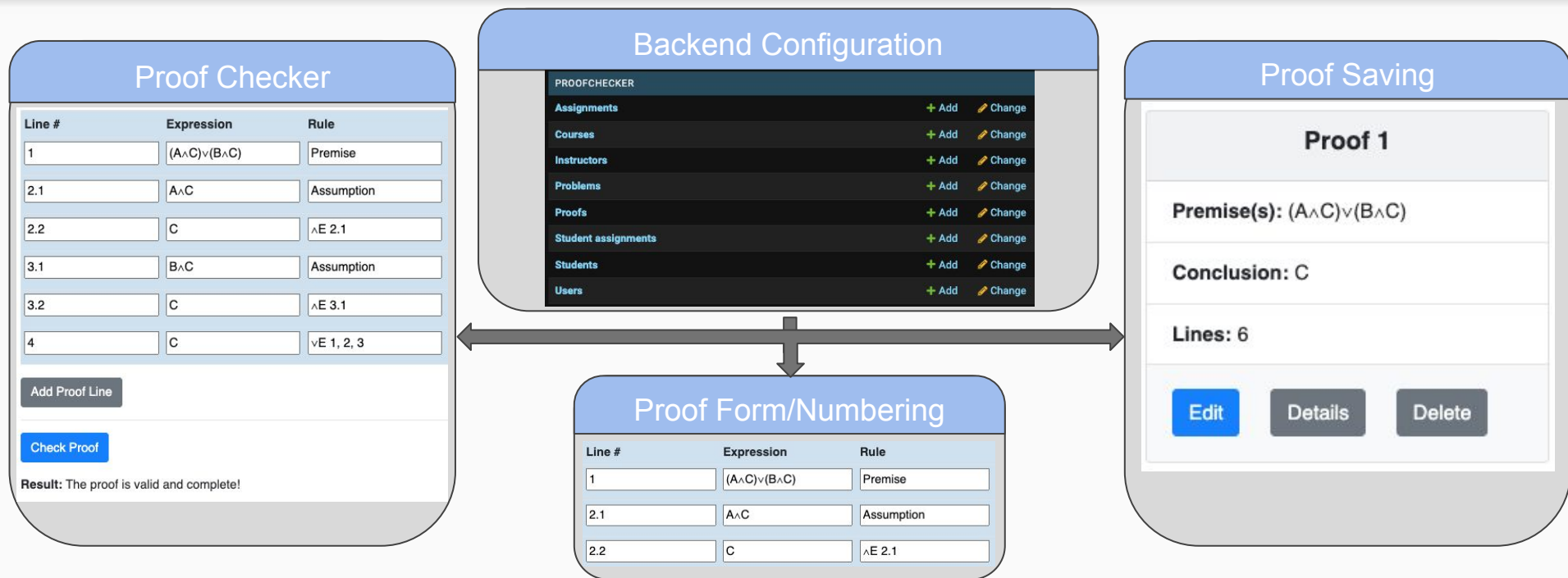
Current Term:

- **Week 3:** Define project requirements, system architecture, and initial design plans
- **Weeks 4-5:** Begin development of UI, user authentication, and syntax validation
- **Weeks 6-7:** Refactor syntax validation, enhance UI, and begin proof validation development
- **Weeks 8-10:** Conclude proof validation development, integrate proof validation with UI, begin development of client-side syntax validation

Next Term (Projections):

- **Weeks 1-2:** Refactor rule verification, begin development of assignments component, enhance client-side syntax validation
- **Weeks 3-4:** Conclude development of assignments component, begin development of client-side proof validation
- **Weeks 5-6:** Begin development of courses component
- **Weeks 7-8:** Begin development of automatic grading functionality
- **Weeks 9-10:** Finalize any remaining requirements

Prototype Description



Demo

<https://github.com/cmscho/Proof-Tool>