





ProofChecker

Rakhfa Amin, Thomas Andrews, Kersley Jatto, Aiasha Sattar and Colton Shoenberger

Project Description

Goal: Implementing a web-based platform for teaching students mathematical reasoning

	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, 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

Colton Shoenberger

Team lead

| Lead designer of proof validation

| Lead of server-side testing

Rakhfa Amin

Lead back-end developer

| Lead designer of user authentication

| Lead database engineer

Thomas Andrews

Lead front-end developer

| Lead designer of User Interface (UI) and User Experience (UX)

Kersley Jatto

Front-end developer

| Lead of client-side testing

Aiasha Sattar

User Experience (UX) developer

| Lead Tux Integration specialist

| Future team lead

Requirements

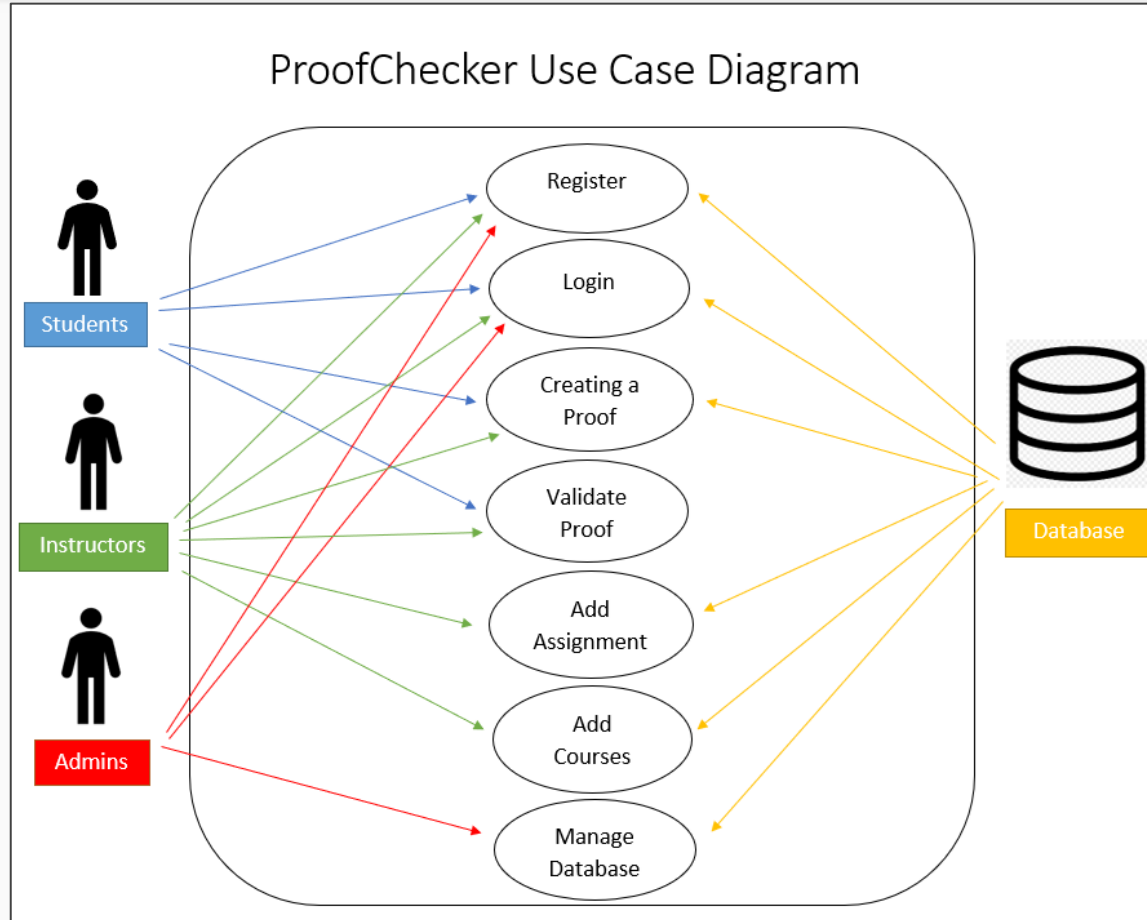
System Requirements

- 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 and First-Order logic.
- The tool shall be able to validate a proof for correctness and completeness.

User Requirements

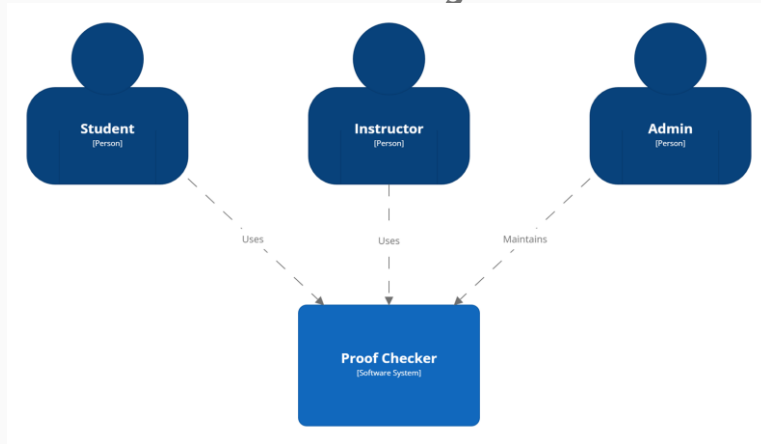
- Users will be able to sign up as either student or instructor.
- Users would be able to create, save, edit and delete their work.
- Professors would be able to create, upload, and export assignments consisting of problems
- Students should be able to view and work on the assignments for submission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

Use Cases

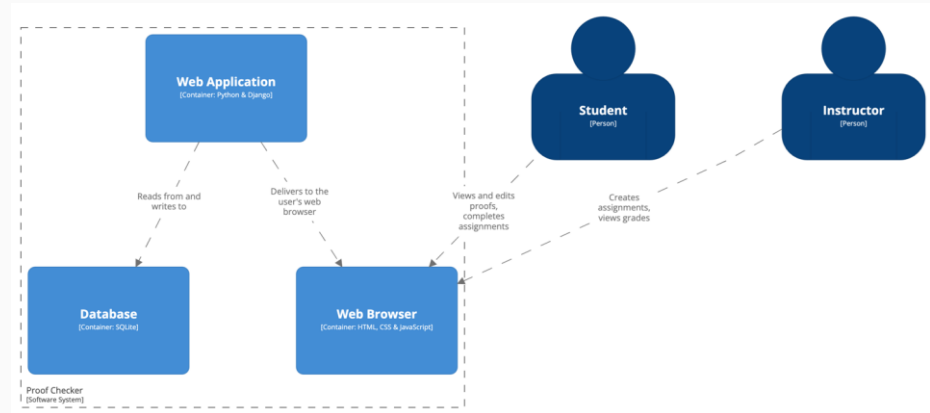


Architecture - High Level

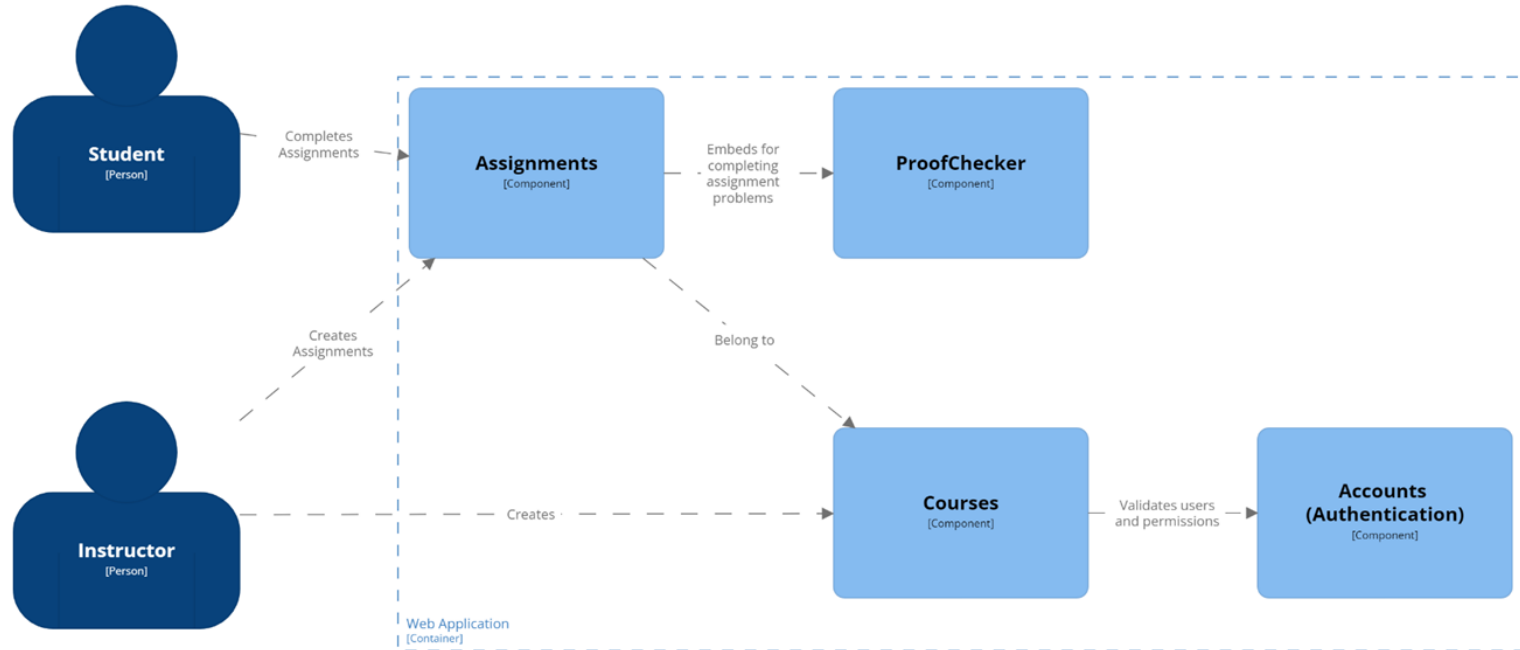
Context Diagram



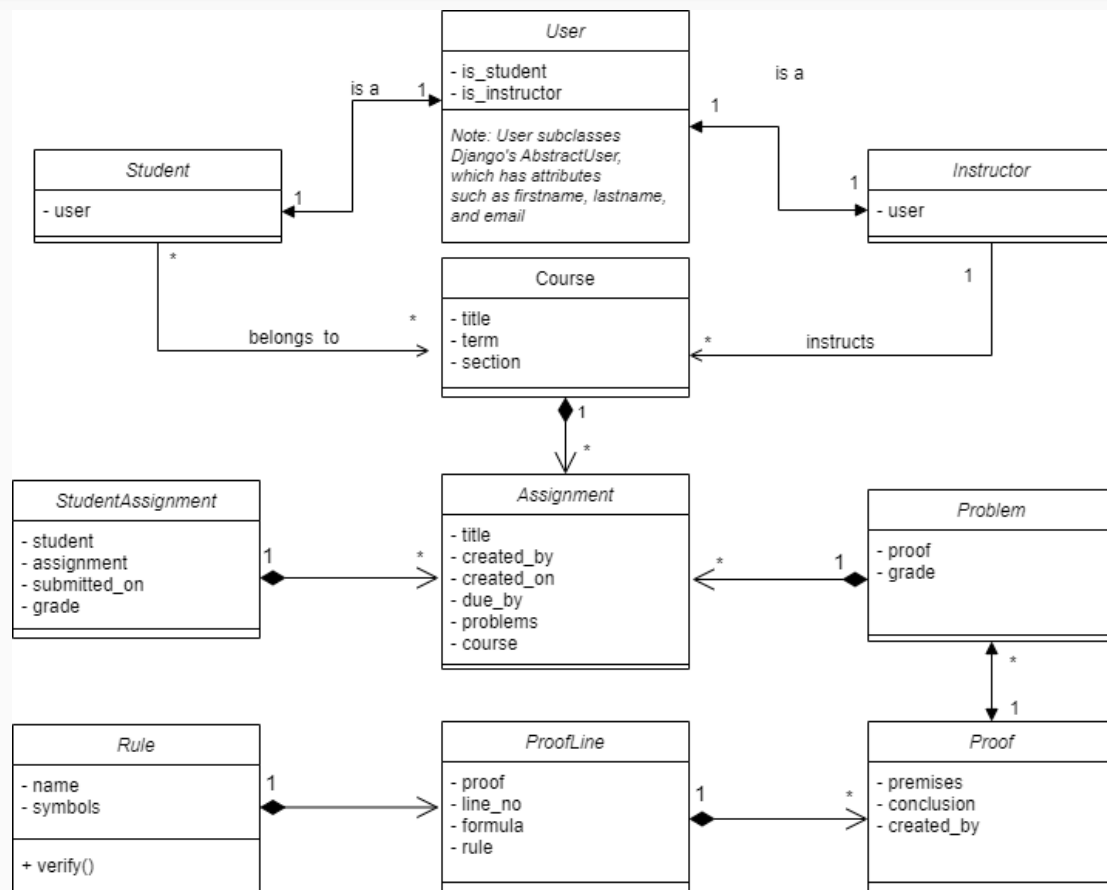
Container Diagram



Architecture - Component Level



Design and Implementation



Server-Side Testing

Name	Stmts	Miss	Cover

proofchecker\admin.py		26	3
88%			
proofchecker\forms.py	30	3	90%
proofchecker\models.py		116	14
88%			
proofchecker\rules\rulechecker.py	51	2	96%
proofchecker\proofs\proofchecker.py	54	0	100%
proofchecker\proofs\proofobjects.py	25	0	100%
proofchecker\proofs\proofutils.py	389	45	88%
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		201	36
82%			
prooftool\settings.py	27	0	100%
prooftool\urls.py	5	0	100%
...			

TOTAL	5681	569	90%

- Using Django's testing framework for unit tests and integration tests
- Using Coverage.py for measuring code coverage
- Currently at **90%** coverage for server-side code

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

Client-Side Testing

- Achieved through the use of the QUnit testing framework.
- Used to the JavaScript methods for that run on the client side of the application where feasible
- Utilizes BlanketJS to obtain code coverage

Client-Side Testing Showcase

Syntax.js QUnit Test Suite

☐ Hide passed tests ☐ Check for Globals ☐ No try-catch ☒ Enable coverage

Filter: Go Module:

QUnit 2.17.2; Mozilla/5.0 [Windows NT 10.0; Win64; x64] AppleWebKit/537.36 (KHTML, like Gecko) Chrome/99.0.4844.51 Safari/537.36

14 tests completed in 8 milliseconds, with 0 failed, 0 skipped, and 0 todo.
24 assertions of 24 passed, 0 failed.

1. Remove Justification with Just Symbol Test (1) Rerun	1 ms
2. Remove Justification without Just Symbol Test (1) Rerun	0 ms
3. Has Valid Symbols with Valid Symbols Test (1) Rerun	1 ms
4. Has Valid Symbols with Invalid Symbols Test (1) Rerun	0 ms
5. Has Valid Symbols with Multiple Commas Test (1) Rerun	0 ms
6. Has Balanced Parenthesis Test with Balanced Parenthesis (1) Rerun	1 ms
7. Has Balanced Parenthesis Test with Unbalanced Parenthesis (3) Rerun	0 ms
8. Set Depth Array Test (2) Rerun	0 ms
9. Finding the Main Operator without Parenthesis Test 1 (1) Rerun	0 ms
10. Finding the Main Operator without Parenthesis Test 2 (1) Rerun	1 ms
11. Atomic Sentence Being Valid TFL Test (2) Rerun	0 ms
12. Well Formed Formulas with One Operator Being Valid TFL Input Test (5) Rerun	1 ms
13. Well Formed Formulas with Multiple Operators Being Valid TFL Input Test (2) Rerun	1 ms
14. Is Valid TFL Against Invalid Input Test (2) Rerun	0 ms

Blanket.js results

1. <http://127.0.0.1:8000/static/js/syntax.js>

Totals

Covered/Total Smts. Coverage (%)

94/105 89.52 %

12400/14208 87.27 %

Supporting Technology



Python 3.10



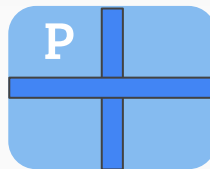
Django 3.2
(Server-Side Web Framework)

PL

Python Lex & Yacc
(Parsing tools for Python)



Heroku
(Temporary website platform)



pipenv
(Virtual environment)



Coverage.py
(Code coverage tool)



Qunit Testing
(Front end testing)



GitHub
(Version control)

Project Timeline

First Term:

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

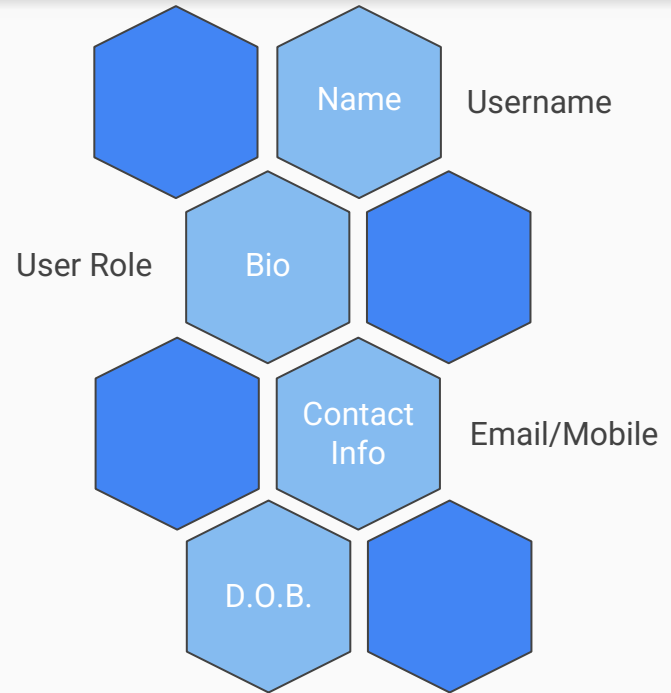
Second Term:

- **Weeks 1-2:** Begin development of FOL proof validation, begin working on UI improvements
- **Weeks 3-4:** Integrate FOL proof validation and UI improvements, begin testing the system with students in the classroom setting
- **Weeks 5-6:** Respond to bug discoveries and other student feedback, begin development of Course and Assignment components
- **Weeks 7-8:** Continue improving the system based on student feedback, continue work on Course and Assignment components
- **Weeks 9-10:** Integrate Course and Assignment components

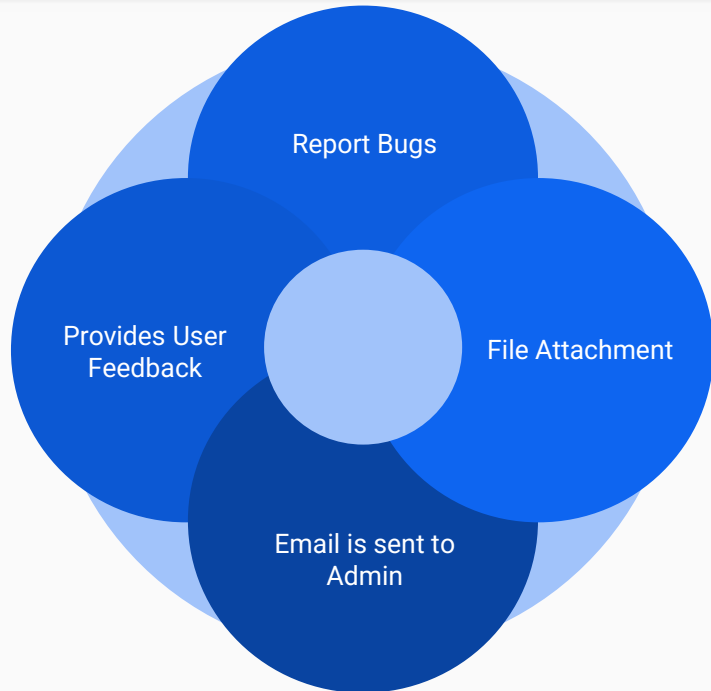
User Profile

Users customize their account by...

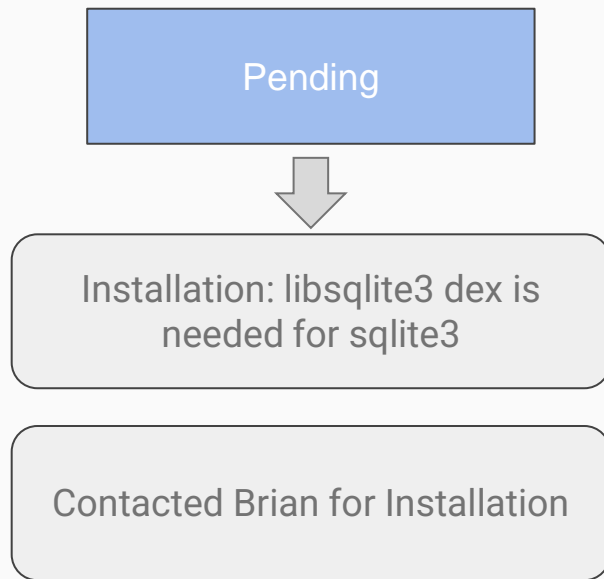
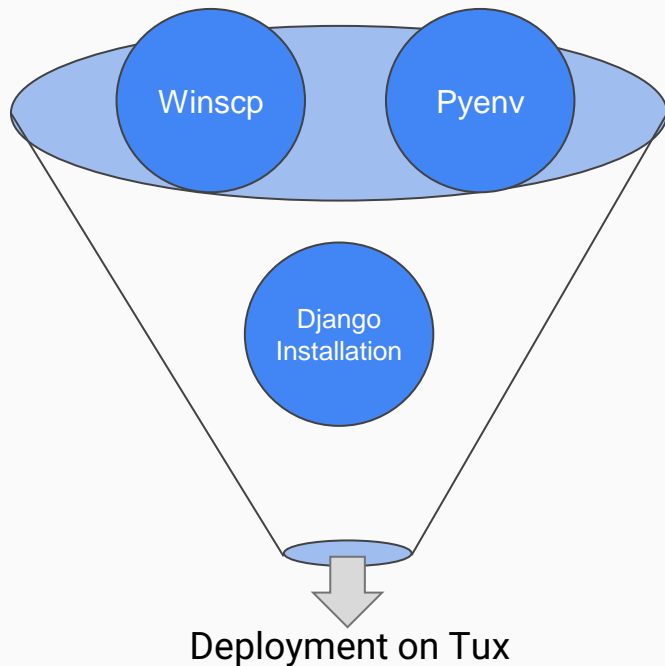
- Adding Name
- Adding a bio about themselves
- Inserting their contact information
- Adding date of birth for fun future birthday features



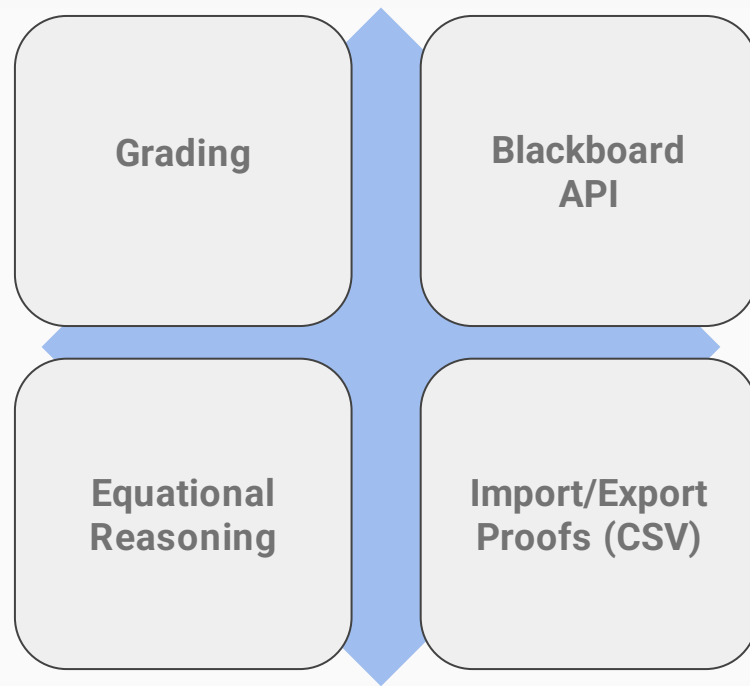
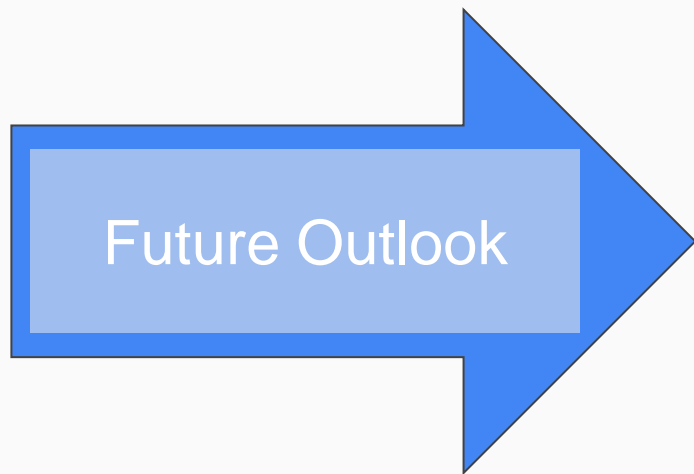
Report Bug and Feedback Feature



Transitioning to more sustainable service



Next Steps



Website demonstration