



Gradify

Final Report

Dakin Werneburg

Davy Marrero

Eric Hutchins

Kenneth Lovingood

Ryan Brady

CMSC 495 – Summer 2019

## Table of Contents

Revision History	5
Overview	6
Project Plan	6
Project Plan Overview	6
Project Description	6
Project Objectives and Scope	7
Other Requirements	8
Assumptions	8
Project Management	8
Team Roles	8
Role Definitions	8
Project Roles	8
Scrum Roles	9
Communication Plan	9
Project Toolset	9
Work Breakdown Structure (WBS)	10
Week 1   Defining the Project and Forming the Team	10
Week 2   Project Plan with Analysis	10
Week 3   Test Plan and User Guide & Peer Reviews	11
Week 4   Design	11
Week 5   Phase 1 Coding	12
Week 6   Phase 2 Coding	12
Week 7   Phase 3 Coding	12
Week 8   Final Submission	12
WBS Task Assignments	12
Assigned Tasks	12
Team Tasks	15
Risk management	16
Scenarios	17

Epics and User Stories	18
Scrum Backlog	20
System Design	<b>21</b>
High Level Architecture	21
Architecture Overview	21
Hardware & Software Requirements	22
Information Architecture	22
Entity Relationship Diagram (ERD)	22
Data Dictionary	25
Application Architecture	34
Solution UML	34
Actors	35
Sequence Diagrams	35
User Interface (UI)	42
Site Hierarchy	42
Security Controls	43
External Endpoints	44
Google Classroom API	45
User Guide	<b>52</b>
User Guide Overview	53
Account Management	53
Create a new account	53
Login with Google Account	54
Login to existing account	57
Gradify For Teachers	58
Managing Classes	58
Importing Classes from Google Classrooms	58
Creating a New Course in Gradify	60
Exporting Classroom Data	62
Managing Assignments	62

Creating Assignments	62
Assigning Grades	65
Managing Courses	66
Test Plan	<b>66</b>
Test Plan Overview	67
Objectives	67
Scope	67
Tactics	67
Testing Strategy	67
Unit Testing	68
Integration Testing	69
Regression Testing	70
Defect Management	70
Quality Assurance & User Acceptance Testing	70
Test Schedule	71
Control Procedures	71
Problem Reporting	71
Change Requests	71
Test Cases	71
Resource, Roles, and Responsibilities	85
Roles	85
Resources	86
Major Deliverables	86
Tools	86
Test Approvals	86
Development History	<b>87</b>
Sprints	87
Phase 1 Sprint	87
Phase 2 Sprint	88
Phase 3 Sprint	89

Phase 4 Sprint	90
GitLab Repository	90
Conclusions	<b>90</b>
Design Strengths	90
Limitations	91
Future Improvements	91
Lessons Learned	93
References	<b>94</b>

## Revision History

Date	Version	Description	Author
07/07/2019	.1	Draft Created	K. Lovingood
07/12/2019	.9	Team Review	Team
07/14/2019	1	Final	Team

## Overview

This final report provides details on what the Gradify project is, its purpose, and how the development project was managed & executed. This report will also describe the solution design of Gradify, how it will be used by end users, and how each feature & user story was tested and validated for accuracy.

## Project Plan

### Project Plan Overview

This Software Project Plan provides details on what this project is, how it will be managed, why the solution is needed, and when the expected outcomes are to be completed for this new software solution named Gradify.

### Project Description

There is no way to view all grades from a single page in Google Classroom. Teachers must click on each assignment and copy grades to another file to create a master list.

Google classroom has excellent tools for creating assignments and being able to assign grades for each student. However, there is no single pane of glass to view all grades. Figure 1 illustrates a teacher's Assignment list where each line is a specific graded assignment.

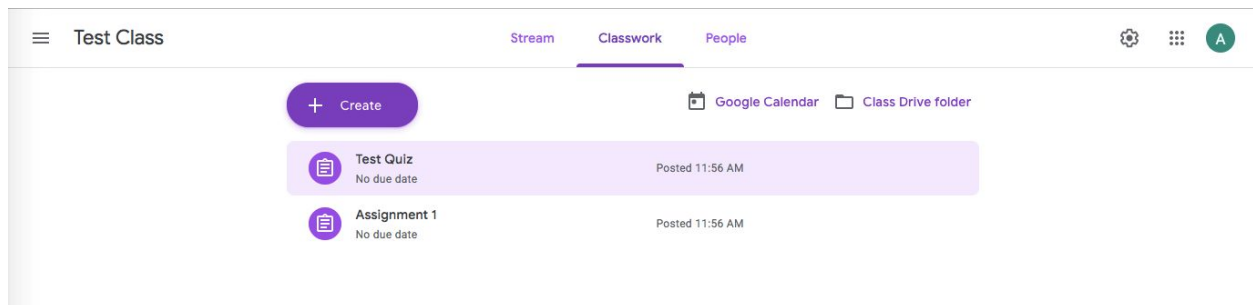


Figure 1: Teacher Assignment List

The progress and assigned grade for each student is listed within Assignment Line item. Figure 2 provides an example of the drilldown into an assignment to view the status for the entire classroom to include each student's progress and grade for that assignment.

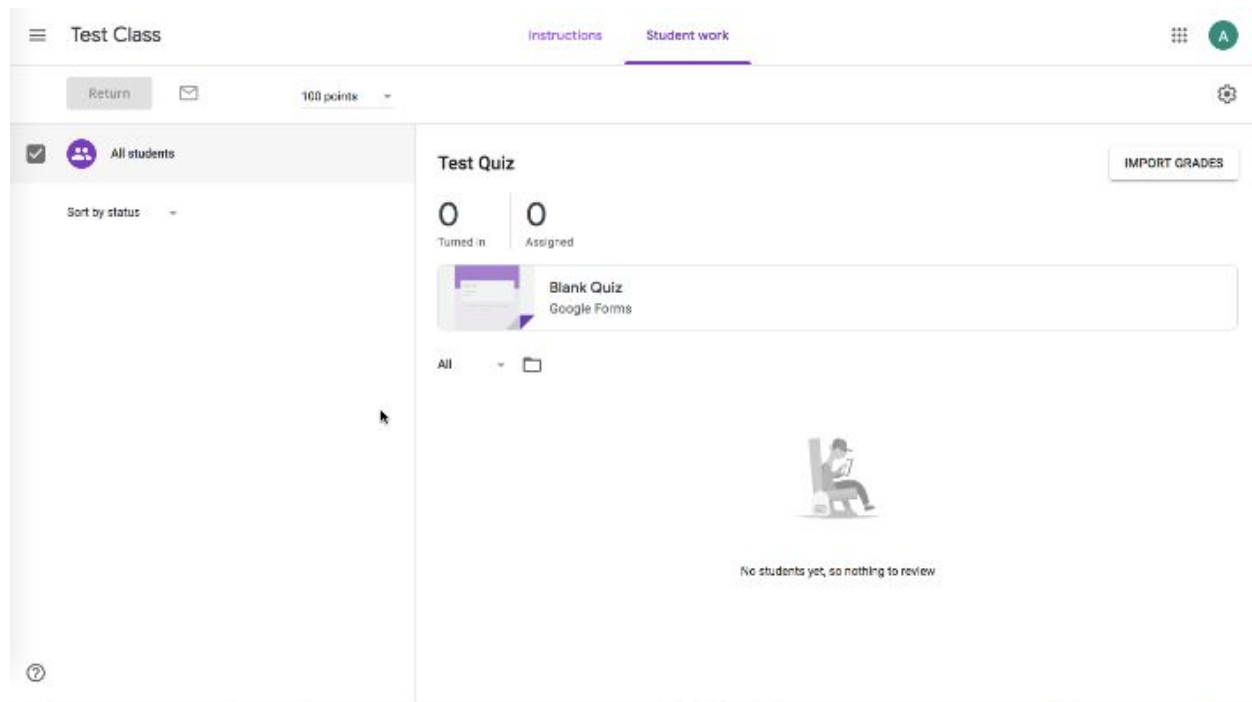


Figure 2: Assignment Drilldown

Currently, teachers can only use Google classroom to distribute videos and do weekly writing journals because it is not worth the effort to do an assignment, grade it, then copy all the grades to another file. There are other learning management platforms in use along with other systems that have maintain grading information, however, finding a centralized platform to aggregate and manage grades in a standard way is problematic.

Gradify shall provide an easy to use and secure solution for teachers to import Google Classroom grades, along with manually imported grade information, into a single dashboard for review and reporting. A simplified export feature should also be made available for use to export all information when needed.

## Project Objectives and Scope

The following objectives are in scope for this project:

- Provide support to maintain a list of classes where each class has students and assignments
- Maintain a list of assignments
- Maintain a secure list of students
- Provide an ability to import Google Classroom assignments, students, and grades
- Manually adjust and/or set student's assignment grades not imported from Google API
- Report on aggregated grades per class and per student
- Export all grades per class and student



## Other Requirements

The following other requirements have been identified:

- The final solution shall be secure and prevent unauthorized access to identity data and any other Personally Identifiable Information (PII) for all teachers, students, and users to include the following regulations:
  - FCC Children's Internet Protection Act (CIPA) (FCC, 2015)
- The final solution, including all source code shall utilize the Apache Open Source License version 2 (Apache License, Version 2.0. 2004).

## Assumptions

The following assumptions will be made during the scope of this project:

- Gradify needs to be completed within the eight-week project schedule. Missed due dates will not be accepted.
- Open Source tools and services shall be utilized where possible.

## Project Management

The Strange-Co team will use the Scrum Agile framework throughout this project. Scrum is a self-regulated process where team members collaboratively identify and assign project tasks that will be worked during a preset span of time known as "sprints" (Schwaber & Sutherland, 2017). This project shall use 1-week sprints where Sprint Planning will occur every Monday and Sprint Review every Friday. Sprint planning will review the tasks that need to be worked, assign new tasks, and review any open tasks that may have issues, require feedback, or remain uncompleted. Team members will review completed tasks and to identify potential tasks for next week during each Sprint Review. Weekly Scrum meetings shall be held daily for team members to align on current Sprint Goals and to assist other development team members.

## Team Roles

Name	Project Role	Scrum Role
<b>Dakin Werneburg</b>	DBA Development Lead	Team Member
<b>Davy Marrero</b>	Source Control & Docker Support Lead	Team Member
<b>Eric Hutchins</b>	Communications Liaison	User Advocate for Product Owner
<b>Kenneth Lovingood</b>	Document Manager	Team Member
<b>Ryan Brady</b>	Django Development Lead	Scrum Master

## Role Definitions

### Project Roles

- **DBA Development Lead:** Provides governance and support for data retrieval, storage, and security in the Relational Database Management System.

- **Communications Liaison:** Acts as the external voice of the project team to the product owners and professor.
- **Django Development Lead:** Technical lead over the application framework and provides technical assistance and mentorship to more novice team members.
- **Document Manager:** Organizes document templates, maintains Work Breakdown Structure, and maintains document versioning and control.
- **Source Control & Docker Support Lead:** Establishes and maintains technical leadership of code versioning support to include providing merge resolutions and to provide Docker master image creation and changes.

### Scrum Roles

- **Scrum Master:** Ensures that Scrum sessions stay on track and guides team members to reduce the product backlog in a timely and orderly manner.
- **Team Member:** Participates in Scrum meetings and works Sprint assignments as they are assigned.
- **User Advocate for Product Owner:** Interviews and gathers feedback from the main stakeholder of the project to make sure that sprint deliverables meet outcome expectations.

### Communication Plan

The project team shall meet regularly to provide status updates, review outcomes of assigned work, and request for assistance when needed. The Communications Liaison will also be responsible to meet with all project owners to include the classroom professor and feedback from the teacher who will use this system. Team shall meet regularly to make sure all team members are up to date on project status. The following meeting cadence has been established to facilitate project collaboration:

- **Daily Open Chat:** The Slack utility shall be used by team members to ask questions, inquire feedback, and provide task updates every day.
- **Scrum / Sprint Planning:** A weekly conference call that occurs every Monday, at the start of a sprint, to review:
  - What is being worked on
  - What will need to be worked on
  - Address any issues that may occur
- **Sprint Review:** A weekly conference call that occurs every Friday, at the end of a sprint, to discuss:
  - A review of task status and demo completed work
  - New tasks that need to be worked in the next sprint

### Project Toolset

The following tools will be used to provide support for and maintain this project.

- **GitLab:** Code Source Control, Scrum backlog, and defect tracking and management.
- **Google Docs:** Document versioning and online collaboration.
- **Slack:** Persistent Communication, meeting notes, daily scrum sessions, and general online collaboration.

- **Workast:** Project management service/Slack plugin for Risk management and Work Breakdown schedule.
- **Zoom:** Video conferencing platform.
- **Draw.io:** Architecture Modeling
- **Figma:** Prototyping and wireframing

## Work Breakdown Structure (WBS)

The following project WBS tasks have been identified for this project

### Week 1 | Defining the Project and Forming the Team

✓ Select Team Roles	May 26	TN4E
✓ Hardware and Software Tools Selection	May 26	TDKX
✓ Create Communication Plan	May 26	T74Z
✓ Select Project	May 26	TFHJ
✓ Introductions	May 26	T8K1
✓ Form Groups	May 26	T5CK

### Week 2 | Project Plan with Analysis

✓ Create Project Plan Template	May 28	TVFE
✓ Complete User Stories	May 30	TNA2
✓ Complete Storyboards	May 31	TJJE
✓ Define Actors	May 30	TZVB
✓ Draft High Level Architecture	May 30	TCM1
✓ Setup Source Code Control (GitLab) & Create Base Django Project	May 28	T8JE
✓ Review Project Plan Draft	Jun 2	TPCQ
✓ Post Project Plan	Jun 5	TBVX

## Week 3 | Test Plan and User Guide & Peer Reviews

✓ Create Draft Test Plan	Jun 6	T2WA
✓ Begin System Design	Jun 7	T3ZC
✓ Develop User Guide	Jun 8	TPEG
✓ Test Scope	Jun 8	T1GG
✓ Test Approach	Jun 8	TQWB
✓ Develop Inputs and Outputs	Jun 8	TB3V
✓ Develop Source Code Style Guidelines	Jun 6	T19J
✓ Develop Regression Tests	Jun 8	TG5W
✓ Create Bug/Defect Tracking Tool and Processes	Jun 8	TJ2Q
✓ Invite outside members for Peer Review	Jun 6	TFUT
✓ Peer Review #1 On Other Team's Test Plan	Jun 8	T8EK
✓ Post Test Plan	Jun 9	T359
✓ Post Peer Reviewed Test Plan	Jun 9	TBJ8

## Week 4 | Design

✓ Finalize System Design	Jun 12	TBUY
✓ Start Solution Development	Jun 16	TBW7
✓ Draft Project Design Document	Jun 15	TXVZ
✓ Create UML Class Diagrams	Jun 13	TM5U
✓ Create ERD	Jun 13	T6P7
✓ Create Sequence Diagrams	Jun 15	TF1K
✓ Document System Integrations / API definitions / JSON-REST descriptions	Jun 15	T9BZ
✓ Document Service Layers	Jun 15	TTJX
✓ Document File & Other Expected Outputs	Jun 15	TFZ8
✓ Document Web & Service Endpoints	Jun 15	TVUV
✓ Review Project Design Document	Jun 16	TQUN
✓ Post Project Design Document	Jun 16	TF4E

## Week 5 | Phase 1 Coding

✓ Continue Solution Development	Jun 23	TXCX
✓ Finalize Project Build Scripts	Jun 20	T382
✓ Create Any Needed Input Files & Data	Jun 21	TQL7
✓ Conduct Peer Review #2	Jun 22	T1M1
✓ Post Phase 1 Report	Jun 23	TN7B
✓ Post Peer Review #2	Jun 23	TP79

## Week 6 | Phase 2 Coding

✓ Continue Solution Development	Jun 30	T57A
✓ Post Phase 2 Report	Jun 30	T1PW

## Week 7 | Phase 3 Coding

✓ Complete Solution Development	Jul 7	T37P
✓ Execute Test Plan	Jul 6	TJKD
✓ Post Phase 3 Report	Jul 7	T3MJ

## Week 8 | Final Submission

✓ Overview - including summary of individual contributions	Jul 13	T722
✓ Finalize Project Plan	Jul 13	T18U
✓ Finalize Requirements Specification	Jul 13	T8NM
✓ Finalize System Specification	Jul 13	T6NU
✓ Finalize User's Guide	Jul 13	TTBK
✓ Finalize Design and Alternate Designs	Jul 13	TN9N
✓ Complete Conclusions, Lessons Learned, Design Strengths, Limitations, and Suggest Future Enhancements	Jul 13	T69V
✓ Conduct Peer Review #3	Jul 13	T5CA
✓ Post Peer Review #3	Jul 13	TF81
✓ Post Final Report	Jul 14	TT6A

## WBS Task Assignments

### Assigned Tasks

Each WBS task has been assigned more than one project resource. The following resources have been assigned the following WBS tasks in Workast. WBS task have been assigned to

individuals who will take a lead role in making sure that the task is completed. All tasks will be reviewed by the team.

#### ^ Dakin T Werneburg

✓	Complete User Stories	May 30		TNA2
✓	Review Project Plan Draft	0/1 Jun 2		TPCQ
✓	Begin System Design	Jun 7		T3ZC
✓	Develop Inputs and Outputs	Jun 8		TB3V
✓	Create UML Class Diagrams	Jun 13		TM5U
✓	Create ERD	Jun 13		T6P7
✓	Create Sequence Diagrams	Jun 15		TF1K
✓	Finalize System Specification	Jul 13		T6NU

#### ^ Davy L. Marrero

✓	Review Project Plan Draft	0/1 Jun 2		TPCQ
✓	Develop Source Code Style Guidelines	Jun 6		T19J
✓	Develop Regression Tests	Jun 8		TG5W
✓	Create Bug/Defect Tracking Tool and Processes	Jun 8		TJ2Q
✓	Finalize Project Build Scripts	Jun 20		T382
✓	Finalize Design and Alternate Designs	Jul 13		TN9N
✓	Hardware and Software Tools Selection	May 26		TDKX
✓	Setup Source Code Control (GitLab) & Create Base Django Project	May 28		T8JE

## ^ Eric Hutchins

✓ Complete User Stories	May 30	TNA2
✓ Review Project Plan Draft	0/1 Jun 2	TPCQ
✓ Develop User Guide	Jun 8	TPEG
✓ Invite outside members for Peer Review	Jun 6	TFUT
✓ Post Test Plan	Jun 9	T359
✓ Post Peer Reviewed Test Plan	Jun 9	TBJ8
✓ Post Project Design Document	Jun 16	TF4E
✓ Post Phase 1 Report	Jun 23	TN7B
✓ Post Peer Review #2	Jun 23	TP79
✓ Post Phase 2 Report	Jun 30	T1PW
✓ Post Phase 3 Report	Jul 7	T3MJ
✓ Finalize User's Guide	Jul 13	TTBK
✓ Post Peer Review #3	Jul 13	TF81
✓ Post Final Report	Jul 14	TT6A
✓ Post Project Plan	Jun 2	TBVX
✓ Post Plans and Specifications	Jun 2	T8TF
✓ Define Actors	May 30	TZVB

## ^ Ken Lovingood

✓ Review Project Plan Draft	0/1 Jun 2	TPCQ
✓ Create Draft Test Plan	Jun 6	T2WA
✓ Start Storyboards	Jun 9	T7CM
✓ Draft Project Design Document	Jun 15	TXVZ
✓ Document System Integrations / API definitions / JSON-REST descriptions	Jun 15	T9BZ
✓ Document Service Layers	Jun 15	TTJX
✓ Document File & Other Expected Outputs	Jun 15	TFZ8
✓ Document Web & Service Endpoints	Jun 15	TVUV
✓ Finalize Project Plan	Jul 13	T18U
✓ Finalize Requirements Specification	Jul 13	T8NM
✓ Create Communication Plan	May 26	T74Z
✓ Create Project Plan Template	May 28	TVFE

## ^ Ryan Brady

✓ Review Project Plan Draft	0/1 Jun 2	TPCQ
✓ Begin System Design	Jun 7	T3ZC
✓ Develop Inputs and Outputs	Jun 8	TB3V
✓ Develop Source Code Style Guidelines	Jun 6	T19J
✓ Finalize System Design	Jun 12	TBUY
✓ Finalize Design and Alternate Designs	Jul 13	TNYN
✓ Hardware and Software Tools Selection	May 26	TDKX
✓ Draft High Level Architecture	May 30	TCM1
✓ Setup Source Code Control (GitLab) & Create Base Django Project	May 28	T8JE

**Team Tasks**

The following Workast WBS tasks have no assignments. These tasks are worked collaboratively amongst the team with no central owner.



✓ Complete Storyboards	Jun 15	TJJE
✓ Test Scope	Jun 8	T1GG
✓ Test Approach	Jun 8	TQWB
✓ Peer Review #1	Jun 8	T8EK
✓ Review Project Design Document	Jun 16	TQUN
✓ Start Solution Development	Jun 16	TBW7
✓ Continue Solution Development	Jun 23	TXCX
✓ Create Any Needed Input Files & Data	Jun 21	TQL7
✓ Conduct Peer Review #2	Jun 22	T1M1
✓ Continue Solution Development	Jun 30	T57A
✓ Complete Solution Development	Jul 7	T37P
✓ Execute Test Plan	Jul 6	TJKD
✓ Overview - including summary of individual contributions	Jul 13	T722
✓ Complete Conclusions, Lessons Learned, Design Strengths, Limitations, and Sugest Future Enhancements	Jul 13	T69V
✓ Conduct Peer Review #3	Jul 13	T5CA
✓ Select Project	May 26	TFHJ
✓ Form Groups	May 26	T5CK
✓ Introductions	May 26	T8K1
✓ Select Team Roles	May 26	TN4E

## Risk management

This project will utilize a Risk, Action, Issue, and Decision (RAID) approach to risk management. RAID log items are not to be confused with Scrum Backlog items. These RAID items are purely for project level risk management and are broken down into the following types:

- **Risk:** An issue that is now a problem and needs to be remediated.
- **Action:** A project task that needs to be completed.
- **Issue:** Something that can be mitigated before it can become a risk.
- **Decision:** A team or product owner decision that impacts the performance, scope, or direction of the project.

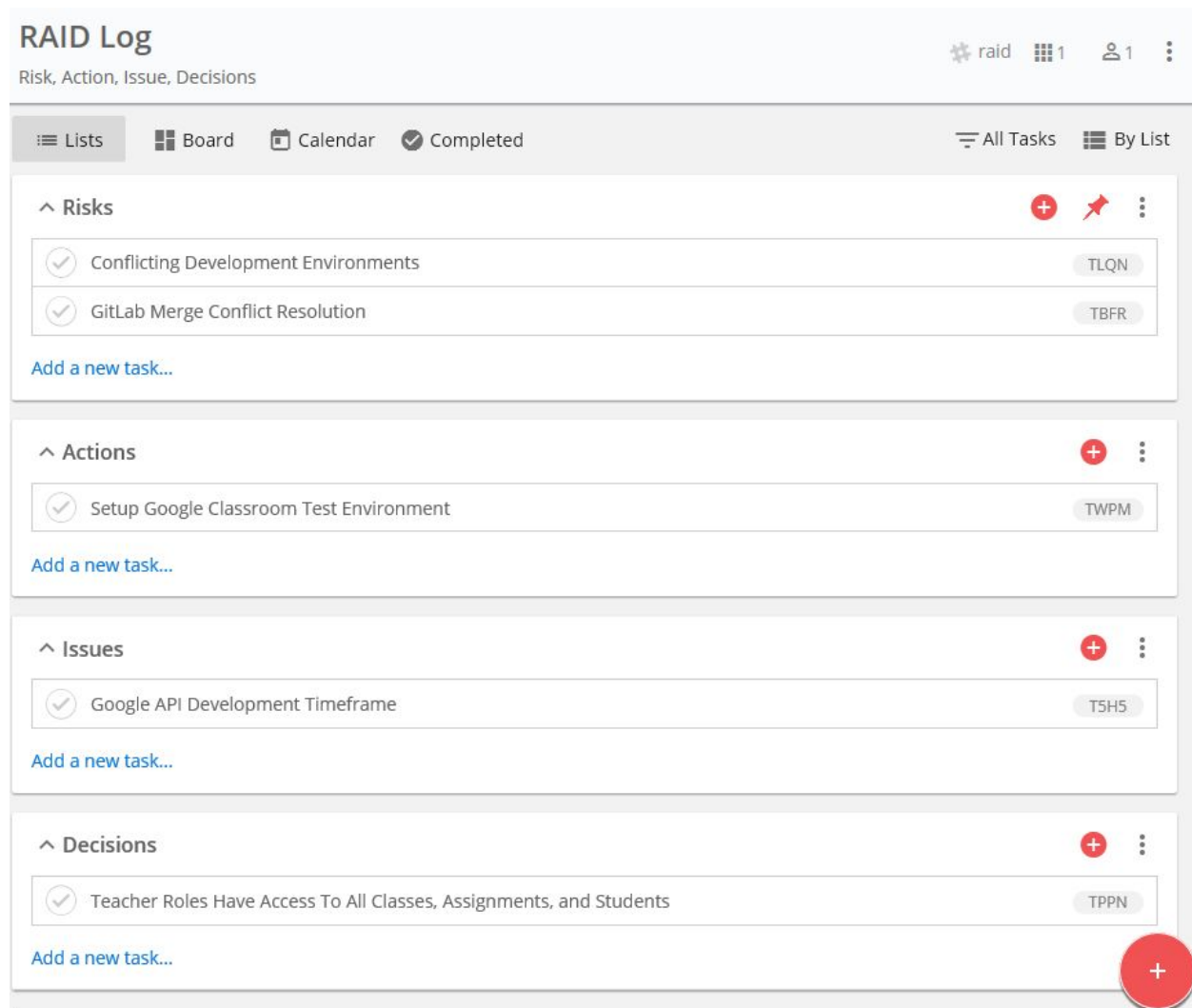


Figure 3: Initial RAID log in Workast

## Scenarios

The following Scenarios have been identified for this project

1. Jane is a teacher that wants to view her Google Classroom data on a single page format, so she visits the Gradify home page. She does not have an account, so she navigates to the account creation page. She creates an account and logs in. After logging in, she imports an existing Google Classroom to Gradify. All of her existing assignments and grades are now visible on the Gradify dashboard.
2. Jon is an existing Gradify teacher preparing to start a new school year. He visits the Gradify page and logs in. He creates a new course and adds students to the course from his class list. Jon does not have any assignments prepared so he logs off.
3. April is a teacher that primarily uses Google Classroom to manage assignments. She already has a Gradify account and has imported her Google Classroom data. During a

free period today, she created a worksheet that her students completed. She wants to track grades but does not want to take the time to put it into Google Classroom. She logs into Gradify and adds an assignment. Since she already has students assigned to her class, she enters all the students grades from the dashboard.

## Epics and User Stories

The following Epics and user stories define what the end user experience and outcomes shall be. An Epic is a collection of user stories or activities that are to describe end user needs and expected outcomes. Each expected outcome has one or more acceptance criteria that must satisfy each user story.

Epic	User Story	Acceptance Criteria
<b>Managing user Account Information</b>	As a user, I want to create an account, so I can save my classroom configurations	A form will appear asking for <ul style="list-style-type: none"> <li>• First Name</li> <li>• Last Name</li> <li>• Organization/School</li> <li>• Email (email is username)</li> <li>• password (must be ...)</li> <li>• Role (Choice: Teacher, Student, Guardian)</li> <li>• submit</li> </ul> Commits to database
	As a user, I want to login to my account, so I can access my courses	A Form that prompts the user for <ul style="list-style-type: none"> <li>• email</li> <li>• password</li> <li>• submit</li> </ul> goes to courses Dashboard
	As a user, I want to change my password, for added security.	A Form that prompts user for <ul style="list-style-type: none"> <li>• new password (must be..)</li> <li>• password verification (must match)</li> <li>• submit</li> </ul> updates password field in database
<b>Managing Courses Dashboard to view and update all my courses</b>	As a Teacher, I want to display courses that are associated with my account so I can view all my course quickly	A dashboard page that displays all courses associated with the logged in user.
	As a Teacher, I want to create a course, so I can add courses to my account	A form will appear asking for <ul style="list-style-type: none"> <li>• Course Name</li> <li>• Course Number</li> <li>• Grade Level</li> <li>• Term Date</li> </ul>
	As a Teacher, that is logged in, I want to	Form prompting for <ul style="list-style-type: none"> <li>• Google Classroom username</li> </ul>



<b>Managing Assignments Dashboard</b>	As a Teacher, I want to modify details an individual assignment	Form that retrieves from database the following editable fields <ul style="list-style-type: none"> <li>• Title</li> <li>• Due Date</li> <li>• max grade</li> <li>• Submit</li> </ul> Commits to Database
<b>Managing Student Assignment Dashboard</b>	As a Teacher, I want to grade an assignment	Form that displays <ul style="list-style-type: none"> <li>• Title</li> <li>• Student</li> <li>• Due Date</li> <li>• max grade</li> <li>• Grade. (editable)</li> <li>• comments (editable)</li> <li>• Submit</li> </ul> commits editable fields to database

*Table 1: Scrum Epics and User Stories*

## Scrum Backlog

The following list of items, as shown in Figure 4, have been identified to be completed. GitLab's issue tracking features are being used to categorize and track backlog items. As requirements are refined, new backlog items will be added.

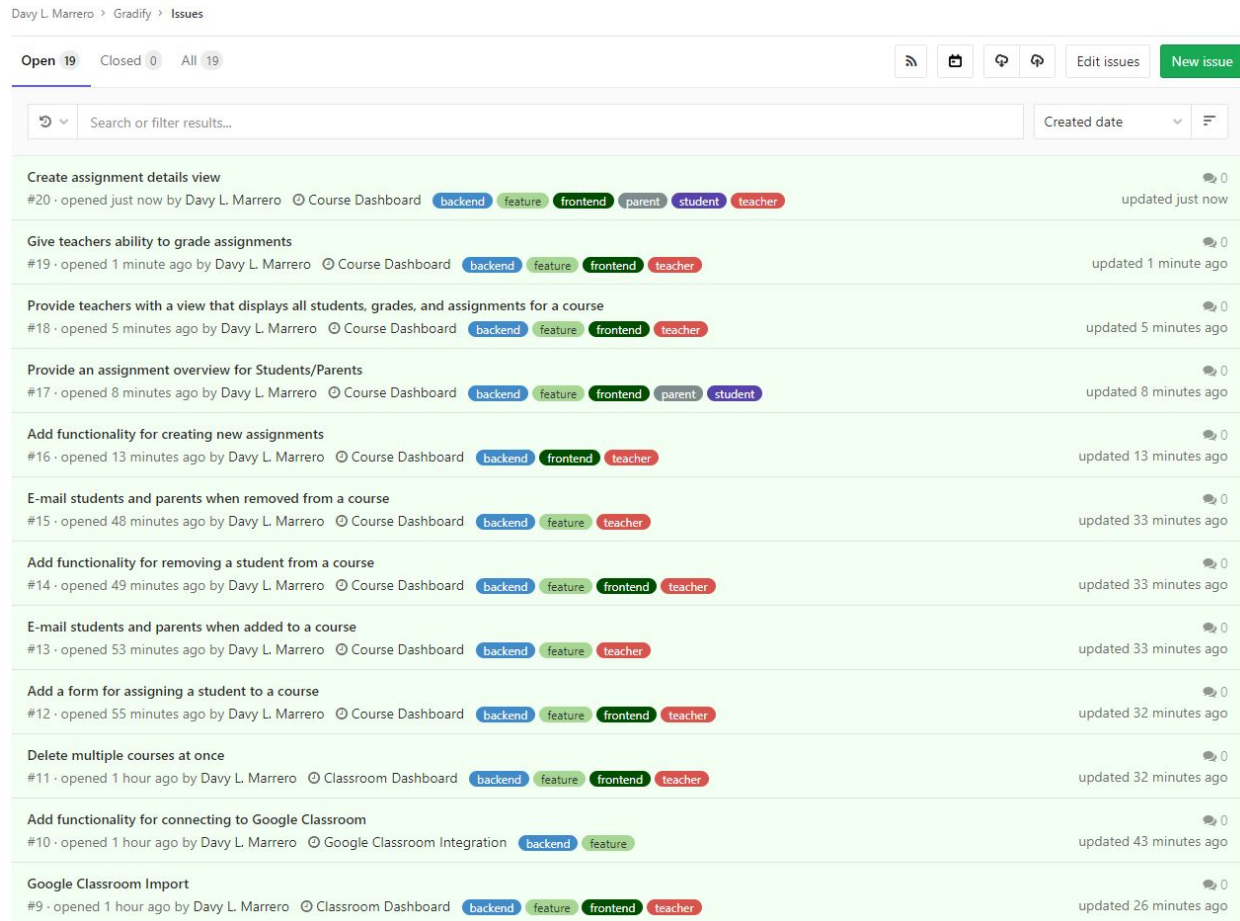


Figure 4: Scrum Backlog in GitLab

## System Design

### High Level Architecture

### Architecture Overview

Gradify shall include a web application that will connect to a relational database as depicted in Figure 5. End Users (teachers, students, and parents) shall use an internet browser to gain access to the web application. The web site shall connect to on an as needed basis. An interface to Google Classroom API shall be used to pull in information as needed.

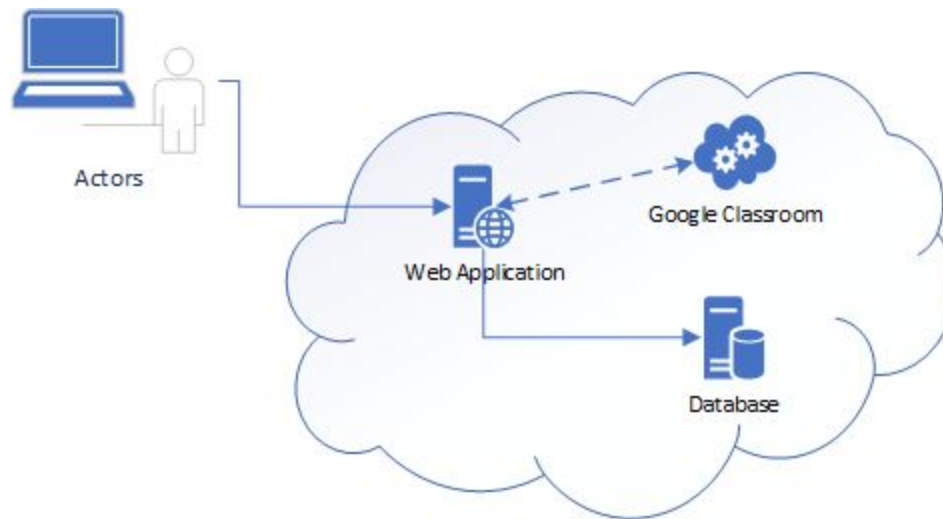


Figure 5: High Level Architecture

## Hardware & Software Requirements

The environment shall consist of any available desktop or laptop computing environment capable of running the Python3 runtime. This solution shall utilize the following technologies:

- Python3 Runtime Environment
- Django 2.2 Web Framework
- SQLite3 Database

## Information Architecture

Gradify is designed as a grade aggregator, rather than an assignment repository. Only the most basic information is stored for assignments, such as the assignment name and student's grade. All other assignment details are handled by Google Classroom, such as due dates and file submissions.

Additionally, Gradify will contain data on class students. There are strict rules to follow when handling personally identifiable data (PII), and Gradify takes care to safeguard the data of students and teachers carefully. All data that is stored locally (i.e. that does not reside in Google Classroom or other potential import locations) will be encrypted using AES-256 via the Django Encrypted Fields module (Defrex, 2017).

## Entity Relationship Diagram (ERD)

The Entity Relationship diagram (ERD) is a graphical representation of the Gradify database. The key entities in the design are the Users, Courses, and Coursework to track grades. Users can be either a Teacher, Student, or Guardian. The attributes used are similar to the Google

Classroom data structure and the data types will closely represent those provide by the Django framework.

The primary goal in designing this ERD was to allow easy expansion to add additional features that are provided by Google Classroom. With this goal in mind, the data was normalized to the third form to reduce redundancy.



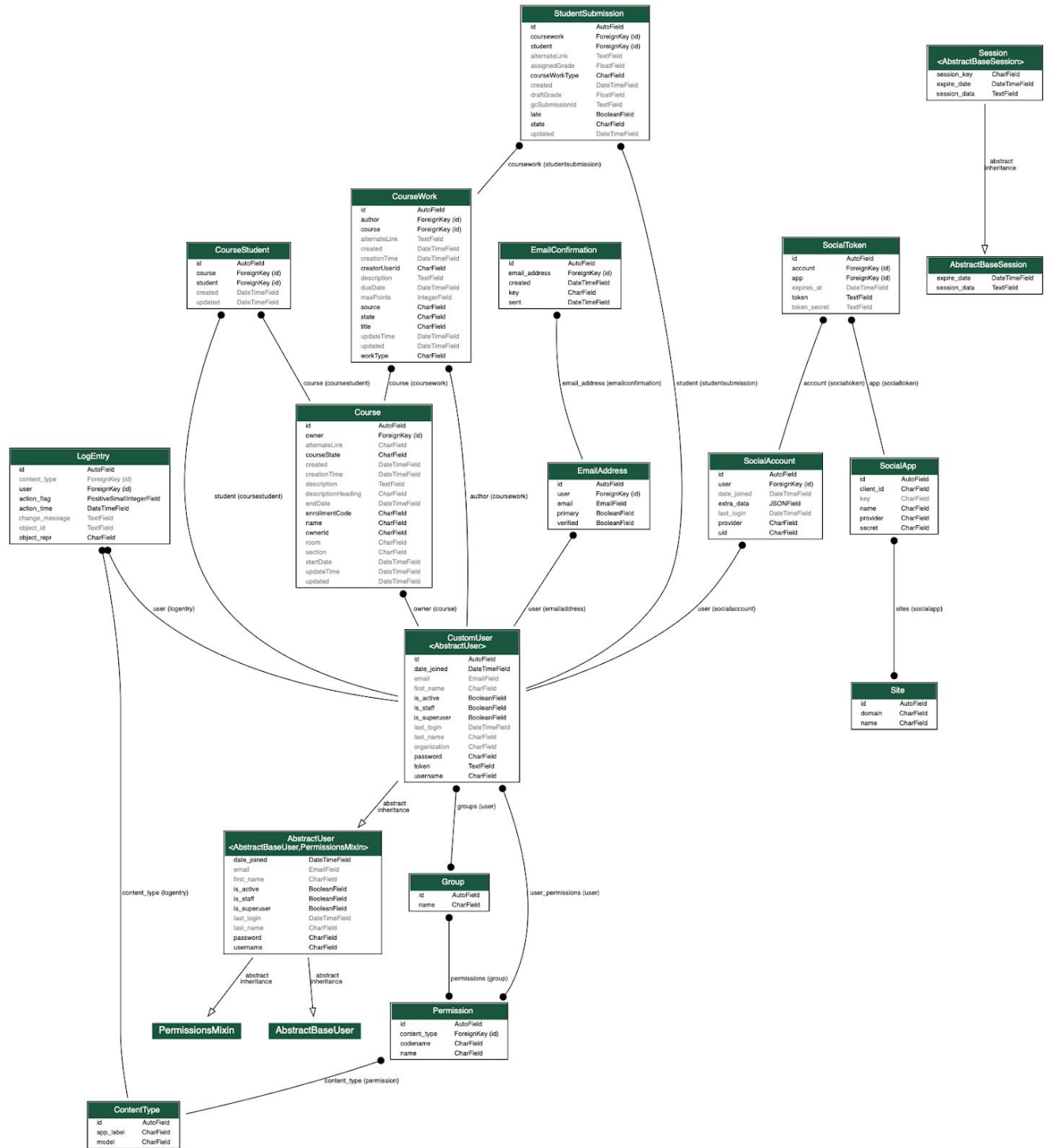


Figure 6: Gradify ERD

## Data Dictionary

Gradify uses the following informational entities and attributes in the system.

### Session

**Description:** Inherited from Django's AbstractBaseSession. Custom database-backed session engine built upon those included in Django

Attribute Name	Data Type	Notes
session_key	CharField	Primary key. The field itself may contain up to 40 characters. The current implementation generates a 32-character string (a random sequence of digits and lowercase ASCII letters).
expire_date	DateTimeField	A datetime designating when the session expires.
session_data	TextField	A string containing an encoded and serialized session dictionary.

### Social Token

**Description:** A SocialToken has foreignkey to SocialApp and SocialAccount, and it stores access\_token and its expiry\_date

Attribute Name	Data Type	Notes
id	AutoField	
account	FK	
app	FK	
expires_at	DateTimeField	
token	TextField	

token_secret	TextField	
--------------	-----------	--

## SocialAccount

**Description:** django-allauth framework to support social account authentication. When a User signs up using OAuth social logins, a SocialAccount object is created

Attribute Name	Data Type	Notes
id	AutoField	
user	FK	
date_joined	DateTimeField	
extra_data	JSONField	
last_login	DateTimeField	
provider	CharField	
uid	CharField	

## SocialApp

**Description:** Most OAuth providers require us to sign up for a so called API client or app, containing client ID and API secret. We need to add a Social App for storing those credentials.

Attribute Name	Data Type	Notes
id	AutoField	
client_id	CharField	
key	CharField	
name	CharField	
provider	CharField	
secret	CharField	

## Site

**Description:** The site that is used for authentication i.e facebook, google etc.

Attribute Name	Data Type	Notes
id	AutoField	
domain	CharField	
name	CharField	

## EmailConfirmation

**Description:** django-allauth framework to confirm email

Attribute Name	Data Type	Notes
id	AutoField	
email_address	FK	
created	DateTimeField	
key	CharField	
sent	DateTimeField	

## EmailAddress

**Description:** django-allauth framework to store email address

Attribute Name	Data Type	Notes
id	AutoField	
user	FK	
email	EmailField	
primary	BooleanField	
verified	BooleanField	

## Course

Description: A Course in Gradify		
Attribute Name	Data Type	Notes
id	AutoField	
owner	FK	The identifier of the owner of a course.
alternateLink	CharField	Link to source of courses
courseState	Choices	[Active, Suspended, etc.]
created	DateTimeField	Auto
createTime	DateTimeField	GoogleClassroom API
description	CharField	Optional description. For example, "We'll be learning about the structure of living creatures from a combination of textbooks, guest lectures, and lab work. Expect to be excited!"
descriptionHeading	CharField	Optional heading for the description. For example, "Welcome to 10th Grade

		Biology."
endDate	DateTimeField	When course ends
enrollmentCode	CharField	Auto generated code to enroll in course
name	CharField	Name of the course. For example, "10th Grade Biology". The name is required.
ownerId	CharField	Google Classroom API
room	CharField	Optional room location. For example, "301"
section	CharField	Section of the course. For example, "Period 2"
startDate	DateTime	When course began
updateTime	DateTimeField	Google Classroom API
updated	DateTimeField	Auto

## Course Students

**Description:** List the Students associated with a course

Attribute Name	Data Type	Notes
id	PK, Autofield	
course	FK(id)	Identifier of the course

student	FK(id)	Identifier of the student
created	DateTime	Auto
updated	DateTime	Auto

## Course Work

Description: Course work created by a teacher for students of the course..		
Attribute Name	Data Type	Notes
id	AutoField	
course	FK	Identifier of the course.
author	FK	Identifier the creator of the course
alternateLink	TextField	Link of source of grades
created	DateTimeField	
creationTime	DateTimeField	Google Classroom API
creatorUserId	CharField	Google Classroom creatorUserId
description	TextField	Optional description of this course work
dueDate	DateTimeField	
maxPoints	IntegerField	Maximum grade for this course work. If zero or unspecified, this assignment is considered ungraded. This must be a non-negative integer value
source	CharField	source of grades. [Manual, Google Classroom]

state	CharField	Status of this course work. [Published, Draft, Deleted]
title	CharFiled	Title of this course work
updateTime	DateTimeField	
updated	DateTimeField	
workType	Choice	[Quiz, Test, Final, etc]. Not associated with Google Classroom

## Group

**Description:** The role of the user, i.e Teacher, Student, Guardian.

Attribute Name	Data Type	Notes
id	AutoField	
name	CharField	Title of Role [Teacher, Student, Guardian]

## Permission

**Description:** Django builtin permission

Attribute Name	Data Type	Notes
id	AutoField	
content_type	FK	
codename	CharField	
name	CharField	



## Student Submissions

Description: Student submission for course work		
Attribute Name	Data Type	Notes
id	AutoField	
coursework	FK	Identifier of the course work
student	FK	Identifier of the student
alternateLink	TextField	Link to source of submissions
assignedGrade	FloatField	Optional grade. If unset, no grade was set.
courseWorkType	CharField	
created	DateTimeField	Auto
draftGrade	Float	Optional pending grade. If unset, no grade was set
gcSubmissionId	TextField	Google Classroom API submissionId
late	Boolean	Whether this submission is late.
state	Choices	[New, Created, Turned-In, Returned, Reclaim]
updated	DateTimeField	Auto

## CustomUser

Description: Inherited from Django's AbstractUser		
Attribute Name	Data Type	Notes
id	AutoField	

date_joined	DateTimeField	
email	EmailField	
first_name	CharField	
is_active	BooleanField	
is_staff	BooleanField	
is_superuser	BooleanField	
last_login	DateTimeField	
last_name	CharField	
organization	CharField	
password	CharField	
token	TextField	
username	CharField	

## LogEntry

**Description:** Show all LogEntry objects in the Django admin site.

Attribute Name	Data Type	Notes
id	AutoField	
content_type	FK	
user	FK	
action_flag	PositiveSmallIntegerField	
action_time	DateTimeField	
change_message	TextField	
object_id	TextField	
object_repr	CharField	

## ContentType

**Description:** Django represents and stores information about the models installed in a project, and new instances of ContentType are automatically created whenever new models are installed.

Attribute Name	Data Type	Notes
id	AutoField	
app_label	CharField	
model	CharField	

## Application Architecture

Gradify is a Python-based data-driven web solution using the Django application framework. This design utilizes, as shown in Figure 7, shows a Model-View-Template (MVT) data pattern where Models that interact with the database, the View where the application's code is maintained, and the Template that formulates the user interface for the user (The Django Book, 2019). This design philosophy separates the data layer from the application layer and user interface.

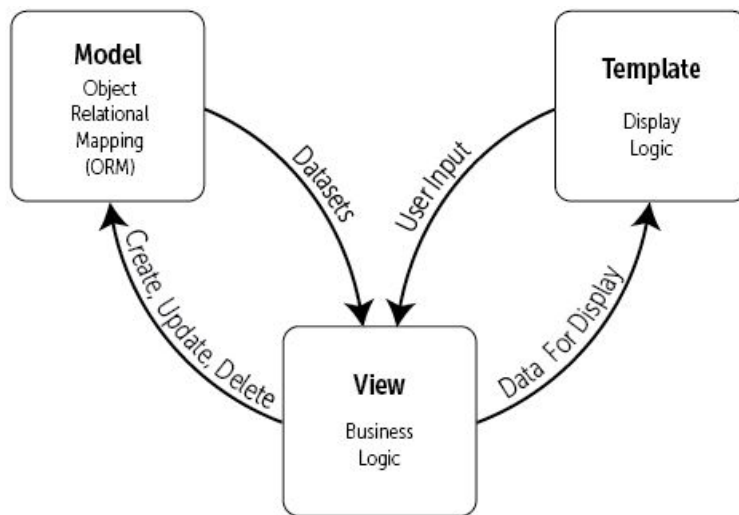


Figure 7: Django MVT Pattern (The Django Book, 2019)

## Solution UML

The high-level UML of the MVT structure of Django shall provide a structure as seen in Figure 8. The Model components, in green, formulate how information is stored and retrieved from the database. The dark blue View Logic and the light blue Application Logic components apply business logic to the flow of information including external calls to remote systems like Google

Classroom. The purple Template components render the UI for interactivity with end users.

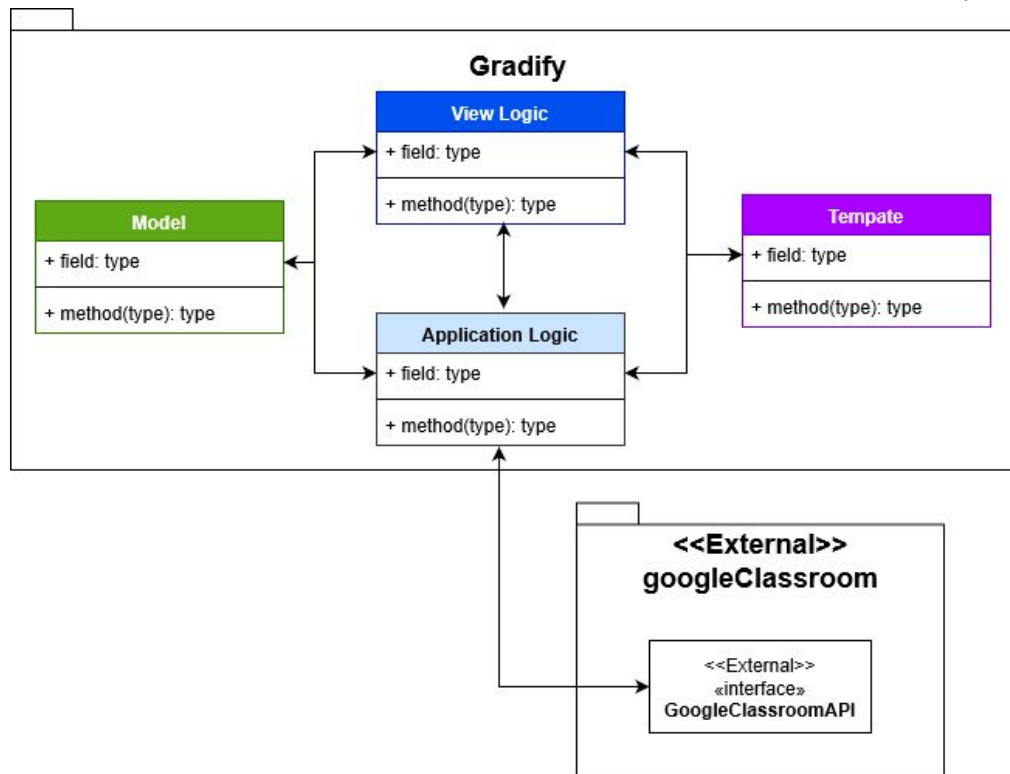


Figure 8: High Level Solution UML

## Complete Solution UML

Figure 4 illustrates the UML for the solution, maintaining the same color key as above where Models are green, View Logics are dark blue, Application Logics are light blue, and Templates are purple.

## Actors

The following Actors have been identified as users within this system:

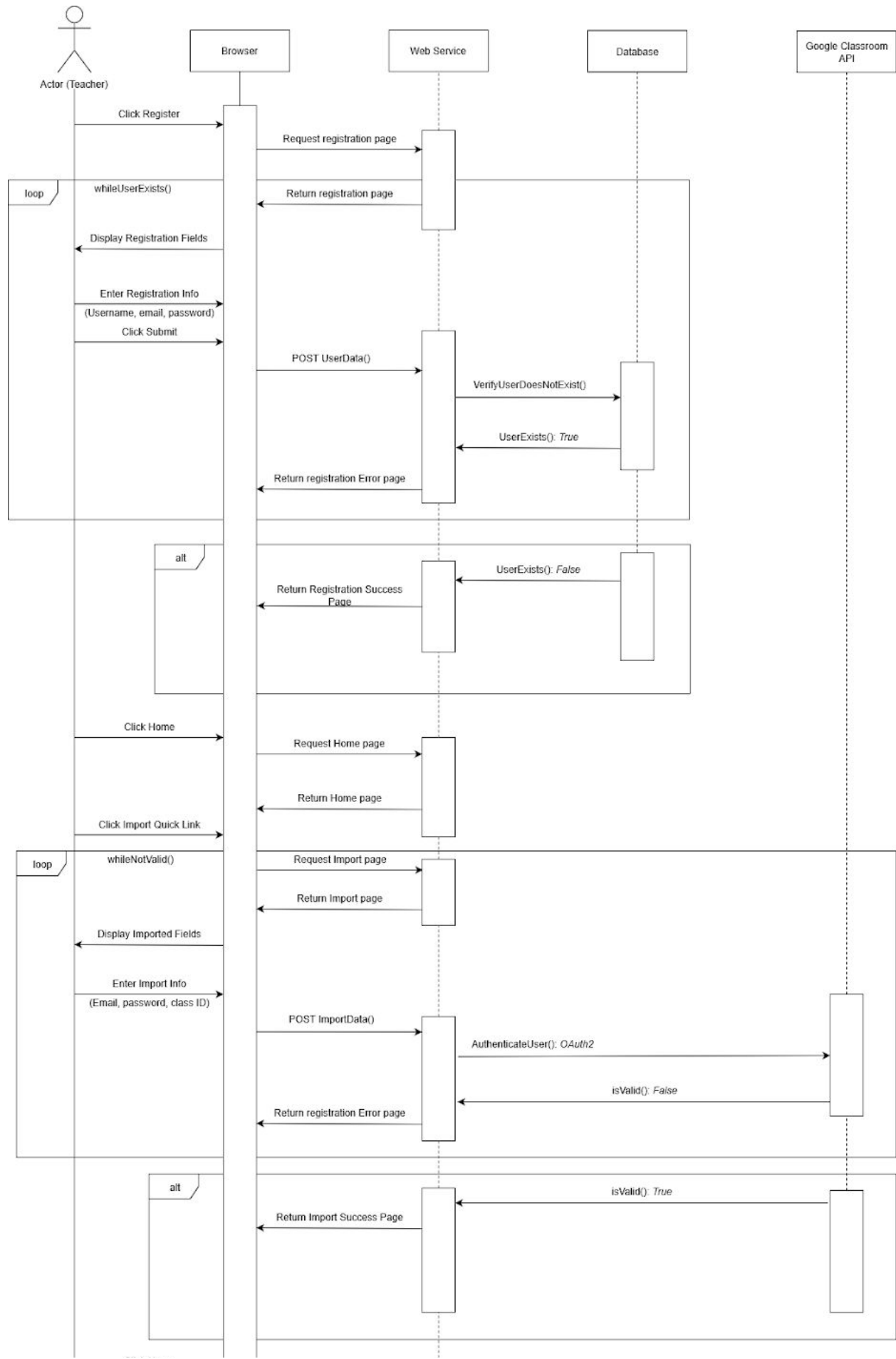
- **Teachers:** An administrative user who can create classes, assignments, and students. Teachers can set and modify grades
- **Students:** A user who has assignments and grades.
- **Guardians:** A user who can review one or more student assignment grades.

## Sequence Diagrams

### Scenario 1

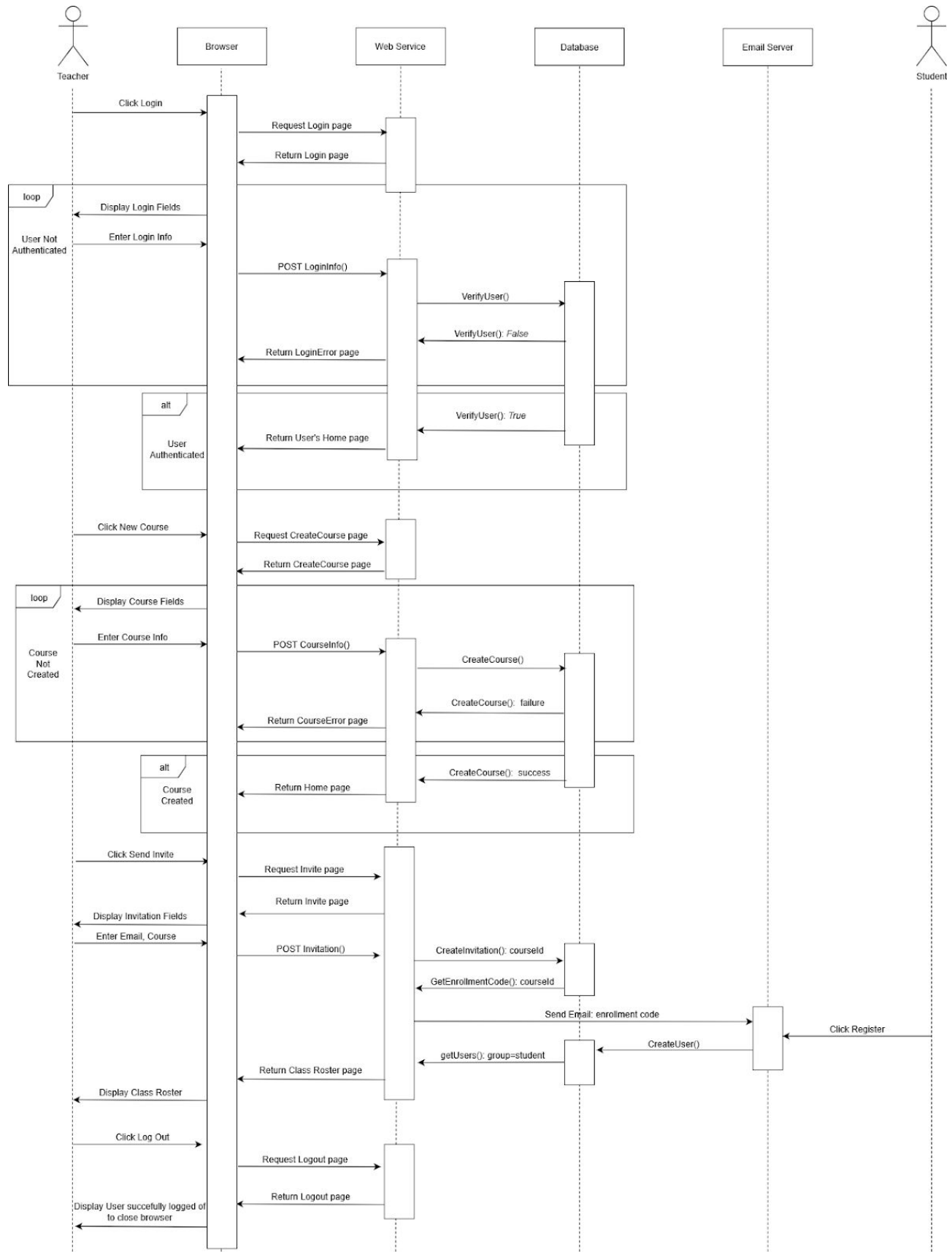
Jane is a teacher that wants to view her Google Classroom data on a single page format, so she visits the Gradify home page. She does not have an account, so she navigates to the account creation page. She creates an account and logs in. After logging in, she imports an existing Google

Classroom to Gradify. Each existing assignments and grades are now visible on the Gradify dashboard.



**Scenario 2**

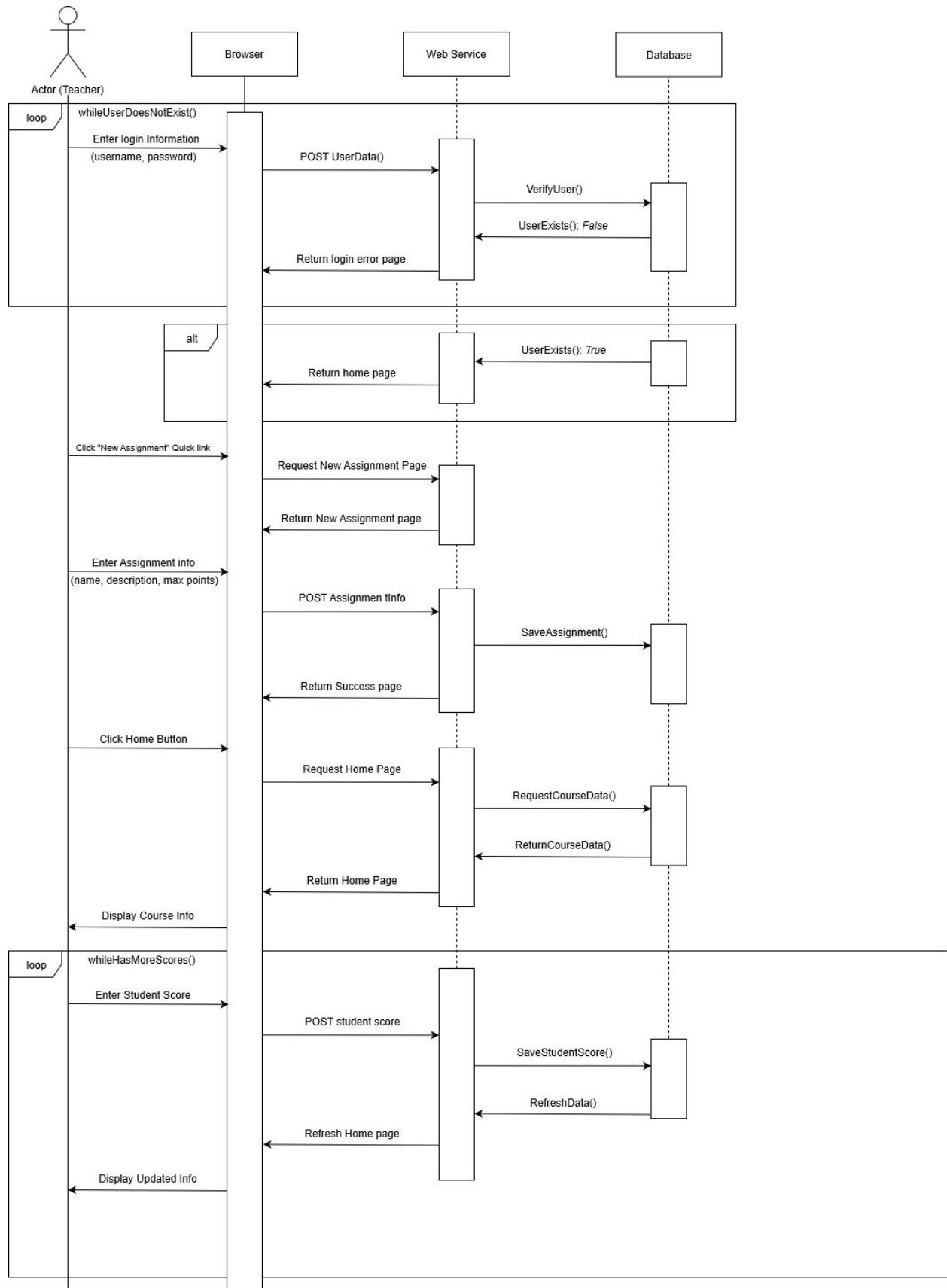
Jon is an existing Gradify teacher preparing to start a new school year. He visits the Gradify page and logs in. He creates a new course and adds students to the course from his class list. Jon does not have any assignments prepared so he logs off.





**Scenario 3**

April is a teacher that primarily uses Google Classroom to manage assignments. She already has a Gradify account and has imported her Google Classroom data. During a free period today, she created a worksheet that her students completed. She wants to track grades but does not want to take the time to put it into Google Classroom. She logs into Gradify and adds an assignment. Since she already has students assigned to her class, she enters all the students grades from the dashboard.



## User Interface (UI)

Gradify's User Interface (UI) is designed to make it easy for users to quickly access the information needed. Common tasks, such as inviting students to join a course, are provided as "Quick Links" for convenience. The design seeks to create a natural hierarchy about the information, guiding the user to the most relevant links and providing a fluid experience for the user.

### Site Hierarchy

Once a user has logged in, the Courses List page displays a list of all courses the user is associated with. A link is provided for each course, which takes the user to that course's details. From there, the gradebook, class roster, and assignments list can be accessed. As the user navigates around Gradify, "breadcrumbs" are displayed in the navigation bar for quick access to parent pages in the hierarchy.

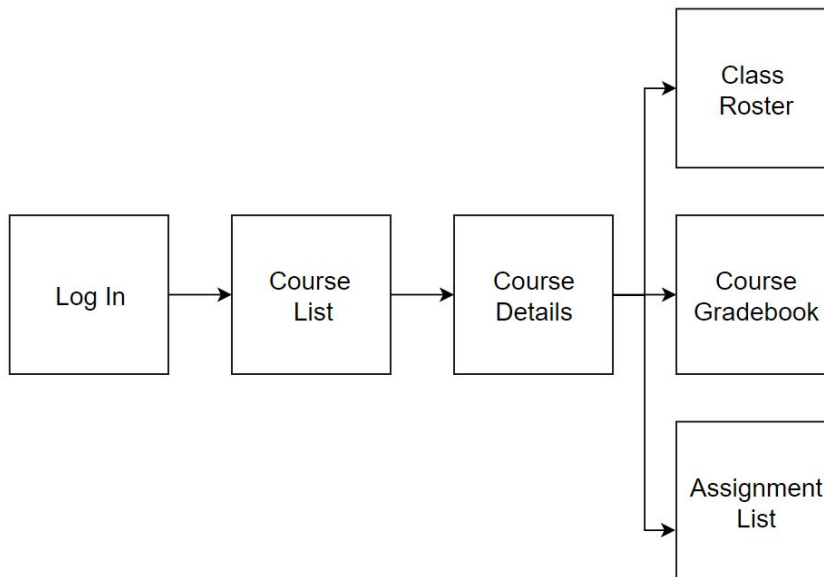


Figure 9: Gradify site hierarchy

### Login Page

The login page prompts the user to enter a username and password. After hitting submit, the user is taken to the Courses List view if the credentials are valid. Otherwise, an error message is displayed. If the user does not have an account, clicking the "Register" link will bring them to the register page.

### Registration Page

The Register page provides users with a form for creating a new Gradify account.

## Course List Page

The Courses List page displays a list of all courses with which a user is associated. For each course, the course number, title, and start/end dates are displayed. An accordion element can be expanded to show the next few assignments due for the class, which contain links to the assignment details. Quick Links are also available to allow convenient access to common tasks the user may wish to perform. Each Quick Link displays a modal dialog prompting the user for the information required to complete the task.

## Course Details Page

The Course Details page provides links to work with course grades, assignments, and the roster. Additionally, a list of upcoming assignments are displayed.

## Class Roster Page

The Class Roster page provides teachers with links to email a student, edit a student's account details (such as updating an email address), or remove a student from the course.

## Course Gradebook Page

The Gradebook page allows teachers to view and edit their students' grades in one convenient location. A table is presented with each row representing a student and columns representing assignments. The Name and Average columns are fixed on the page, while the assignments scroll horizontally. This allows the instructor to keep names always in view of the columns being worked with. To edit student grades, the user may click on a cell, causing it to become editable. When the user is done editing grades, clicking the Save button will save all changes. Clicking the Cancel button will discard all unsaved changes.

## Assignment List Page

The Assignment List Page provides the ability for teachers to review all assignments associated with that course. They have the ability to create, update, and delete assignments as needed.

## Security Controls

The Django application framework provides numerous security protections including, Host Header validation, Cross Site Scripting prevention, SQL Injection checks, and Clickjacking remediation to name a few (The Django Project, 2019). The framework also provides the following out of the box authentication and authorization systems:

- **User Store:** A repository is kept maintaining username, passwords, email, and other types of attributes attributed to the user.
- **Authentication:** Base Authenticate(), login() and logout() methods allow for easy and convenient methods to validate who the users claim to be.
- **Authorization:** Groups can be created to contain more than one permission where each permission is provides the user the ability to perform a specific action.

- **Encryption:** SSL will be used to encrypt information during transport (to and from the user) and the django-encrypted-fields module will store information using AES-256 at rest (SQLite database files on disk).

## External Endpoints

External endpoints are applications, services, and microservices that exist outside of Gradify. Each external resource leverages following technologies to exchange information and execute commands/actions as required.

### JSON format

JSON is a lightweight data exchange format is human readable and easily interpreted by software. JSON messages consist of name:value pairs (Crockford). Sample JSON message containing a string name value, an array of messages, and a numeric age:

```
{"name":"Student Name","messages":["msg 1","msg 2","msg 3"],"age":8}
```

### REST Web Services

Representational State Transfer (REST) web services provide the ability for applications and end user clients the ability to make requests in a lightweight, stateless, and consistent manner (Fielding, 2000). REST services provide the ability for clients make a request while the data format used in the exchange of information is usually JSON or XML. REST services provide the backbone for many Application Programming interface (API).

### OAuth 2.0

OAuth is a standard protocol used in the industry to authenticate users from different organizations using a single common user identity. One organization represents the Authentication Server or Identity Provider (IdP) that is used to authenticate a user. Other organizations, the Resource Services or Service Providers (SP), trusts that the user that the IdP has authenticated is who they say they are and allows them into the systems and platforms (Richer). Figure 10 steps through the OAuth authentication process.

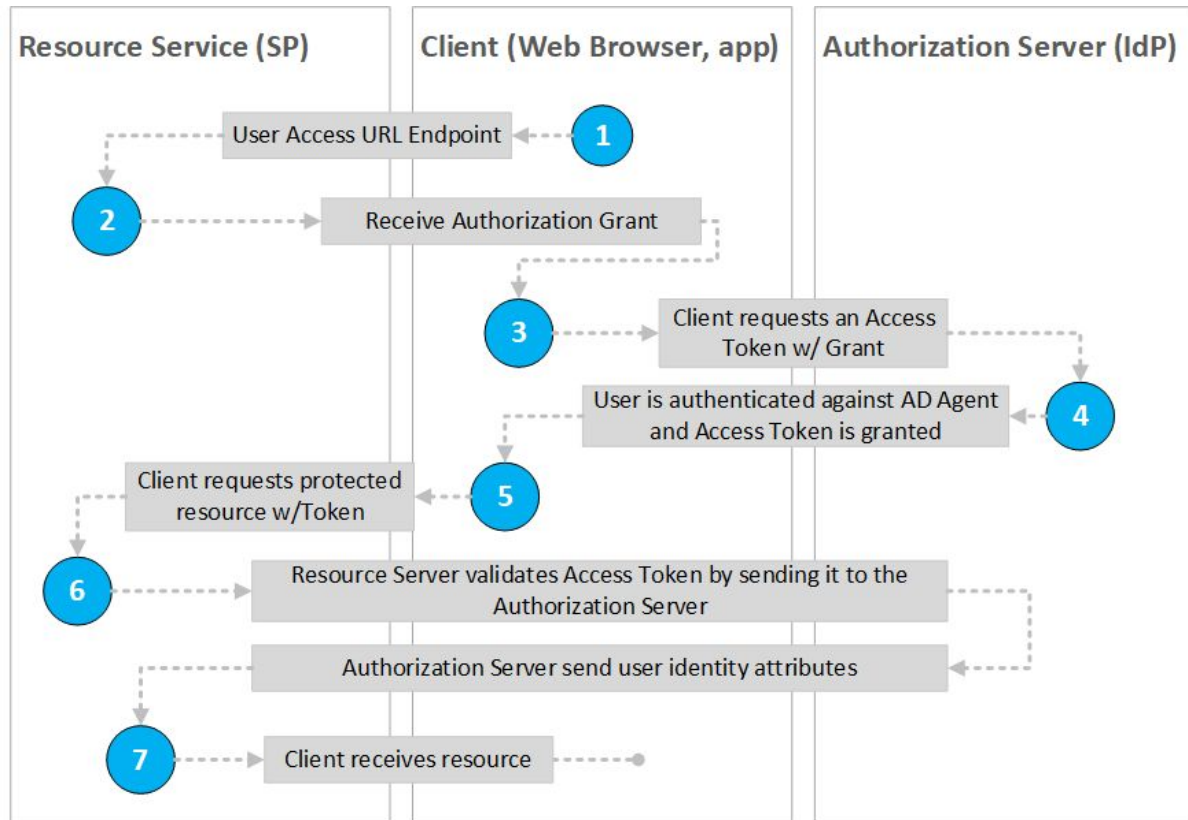


Figure 10: The OAuth Authentication Process

## Google Classroom API

A core feature of the Gradify application is to provide the ability for the application to import student grade. This is accomplished through the usage of the following Google Classroom REST API methods (Google Inc). The Google APIs leverage OAuth to authenticate users into their Google Classroom tenants, REST services to host their API methods, and JSON as the format to exchange data.

### List Course

Gets a full listing of courses found for the user specified in the OAuth session. A course is an instance of a specific class that is used to maintain student assignments and grades.

### HTTP REST Endpoint

GET <https://classroom.googleapis.com/v1/courses>

### OAuth Scope

<https://www.googleapis.com/auth/classroom.courses>

### Returns Course Array

```
{
```

```
"id": string,
"name": string,
"section": string,
"descriptionHeading": string,
"description": string,
"room": string,
"ownerId": string,
"creationTime": string,
"updateTime": string,
"enrollmentCode": string,
"courseState": enum(CourseState),
"alternateLink": string,
"teacherGroupEmail": string,
"courseGroupEmail": string,
"teacherFolder": {
    object(DriveFolder)
},
"courseMaterialSets": [
    {
        object(CourseMaterialSet)
    }
],
"guardiansEnabled": boolean,
"calendarId": string
}
```

## Get Courses By ID

Returns a specific course based on the unique identifier assigned by google.

## HTTP REST Endpoint

GET <https://classroom.googleapis.com/v1/courses/{id}>

## Input Parameters

id: string, the id of the classroom to get

## OAuth Scope

<https://www.googleapis.com/auth/classroom.courses>

## Returns Course Array

```
{
  "id": string,
  "name": string,
  "section": string,
  "descriptionHeading": string,
  "description": string,
  "room": string,
  "ownerId": string,
  "creationTime": string,
  "updateTime": string,
  "enrollmentCode": string,
  "courseState": enum(CourseState),
  "alternateLink": string,
  "teacherGroupEmail": string,
  "courseGroupEmail": string,
  "teacherFolder": {
    object(DriveFolder)
  },
  "courseMaterialSets": [
    {
      object(CourseMaterialSet)
    }
  ],
  "guardiansEnabled": boolean,
  "calendarId": string
```



```
}
```

## List courseWork

Returns all student assignments for a specified course.

### HTTP REST Endpoint

GET <https://classroom.googleapis.com/v1/courses/{courseId}/courseWork>

### Input Parameters

courseId: string, the course identifier

### OAuth Scope

<https://www.googleapis.com/auth/classroom.coursework.students>

### Returns CourseWork

```
{
  "courseId": string,
  "id": string,
  "title": string,
  "description": string,
  "materials": [
    {
      object(Material)
    }
  ],
  "state": enum(CourseWorkState),
  "alternateLink": string,
  "creationTime": string,
  "updateTime": string,
  "dueDate": {
    object(Date)
  },
  "dueTime": {
```

```

        object(TimeOfDay)
    },
    "scheduledTime": string,
    "maxPoints": number,
    "workType": enum(CourseWorkType),
    "associatedWithDeveloper": boolean,
    "assigneeMode": enum(AssigneeMode),
    "individualStudentsOptions": {
        object(IndividualStudentsOptions)
    },
    "submissionModificationMode": enum(SubmissionModificationMode),
    "creatorUserId": string,
    "topicId": string,
    // Union field details can be only one of the following:
    "assignment": {
        object(Assignment)
    },
    "multipleChoiceQuestion": {
        object(MultipleChoiceQuestion)
    }
    // End of list of possible types for union field details.
}

```

## Get courseWork

Returns a specific assignment associated with a course.

## HTTP REST Endpoint

GET <https://classroom.googleapis.com/v1/courses/{courseId}/courseWork/{id}>

## Input Parameters

courseId: string, the course identifier

id: string, the unique identifier of the course work to get

**OAuth Scope**

<https://www.googleapis.com/auth/classroom.coursework.students>

**Returns CourseWork**

```
{
  "courseId": string,
  "id": string,
  "title": string,
  "description": string,
  "materials": [
    {
      object(Material)
    }
  ],
  "state": enum(CourseWorkState),
  "alternateLink": string,
  "creationTime": string,
  "updateTime": string,
  "dueDate": {
    object(Date)
  },
  "dueTime": {
    object(TimeOfDay)
  },
  "scheduledTime": string,
  "maxPoints": number,
  "workType": enum(CourseWorkType),
  "associatedWithDeveloper": boolean,
  "assigneeMode": enum(AssigneeMode),
  "individualStudentsOptions": {
    object(IndividualStudentsOptions)
  }
}
```

```
},
"submissionModificationMode": enum(SubmissionModificationMode),
"creatorUserId": string,
"topicId": string,
// Union field details can be only one of the following:
"assignment": {
    object(Assignment)
},
"multipleChoiceQuestion": {
    object(MultipleChoiceQuestion)
}
// End of list of possible types for union field details.
}
```

## List Students

Returns all students that have been configured for a course.

### HTTP REST Endpoint

GET <https://classroom.googleapis.com/v1/courses/{courseId}/students>

### Input Parameters

courseId: string. the id of the course to retrieve

### OAuth Scope

<https://www.googleapis.com/auth/classroom.rosters>

### Returns Student

```
{
  "courseId": string,
  "userId": string,
  "profile": {
    object(UserProfile)
  },
}
```

```
"studentWorkFolder": {  
    object(DriveFolder)  
}  
}
```

### **Get UserProfile**

Looks up the base information of a specific user based on their unique identifier.

### **HTTP REST Endpoint**

GET <https://classroom.googleapis.com/v1/userProfiles/{userId}>

### **Input Parameters**

userId: string, the id of the profile to return

### **OAuth Scope**

<https://www.googleapis.com/auth/classroom.rosters>

### **Return UserProfile**

```
{  
    "id": string,  
    "name": {  
        object(Name)  
    },  
    "emailAddress": string,  
    "photoUrl": string,  
    "permissions": [  
        {  
            object(GlobalPermission)  
        }  
    ],  
    "verifiedTeacher": boolean
```

# User Guide

## User Guide Overview

Gradify allows seamless and secure access to grades for teachers, students, and guardians. After creating an account, users can log in at any time and see a current snapshot of the status of assignments that are important to them.

As a teacher:

1. Create an account
2. Login
3. Import Google Classroom data
4. Manually create courses
5. Manually create assignments and assign grades
6. View grades and assignments from the dashboard
7. Export grade data

## Account Management

### Create a new account

First time users will need to create a Gradify account to access the application. After navigating to Gradify homepage the user will find the Gradify login screen (Figure 11).

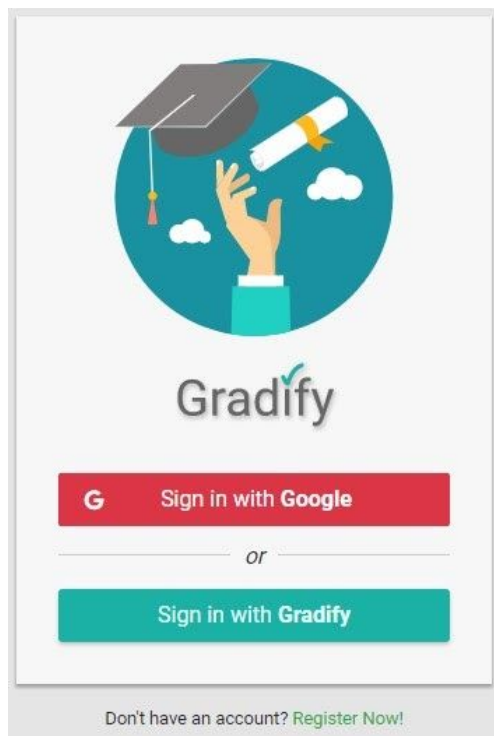
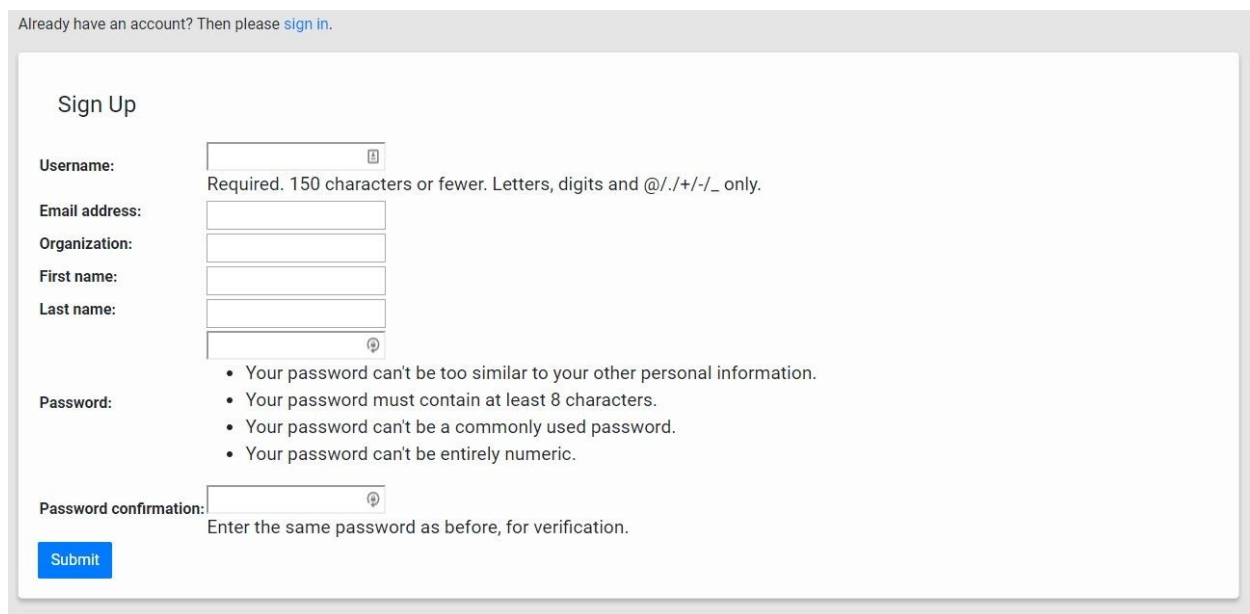


Figure 11: Gradify Login Page

New users should click the Register Now link at the bottom of the screen. The user will be required to fill out a form with their first and last name, organization, email address, and role. They will also need to create a password (Figure 12). Passwords must conform to the following rules:

- Minimum of 8 characters
- Contain letters and numbers
- Does not include user's name or email address
- Not a common password



Already have an account? Then please [sign in](#).

### Sign Up

**Username:**  ⓘ  
Required. 150 characters or fewer. Letters, digits and @/./+/-/\_ only.

**Email address:**

**Organization:**

**First name:**

**Last name:**  ⓘ

**Password:**   

- Your password can't be too similar to your other personal information.
- Your password must contain at least 8 characters.
- Your password can't be a commonly used password.
- Your password can't be entirely numeric.

**Password confirmation:**  ⓘ  
Enter the same password as before, for verification.

Figure 12: New User Registration

## Login with Google Account

Users with existing Google classrooms can login with the Gmail account associated with those classrooms. To do so, users should click the Sign in with Google button on the home page (Figure 11). This takes the user to the Sign in with Google page where they can choose the Google account they wish to use (Figure 13).

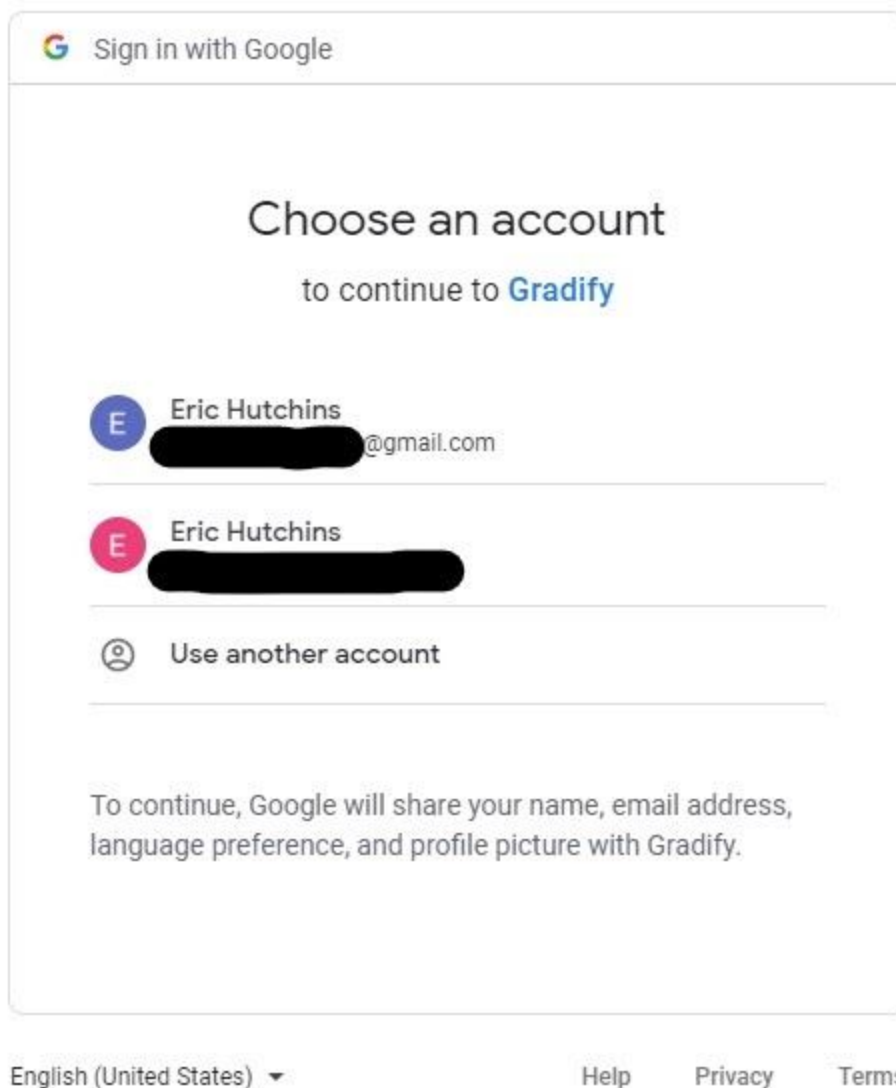


Figure 13: Choose Google account

The user will then be asked to give permission to the Gradify application to access certain Google Classroom information (Figure 14).



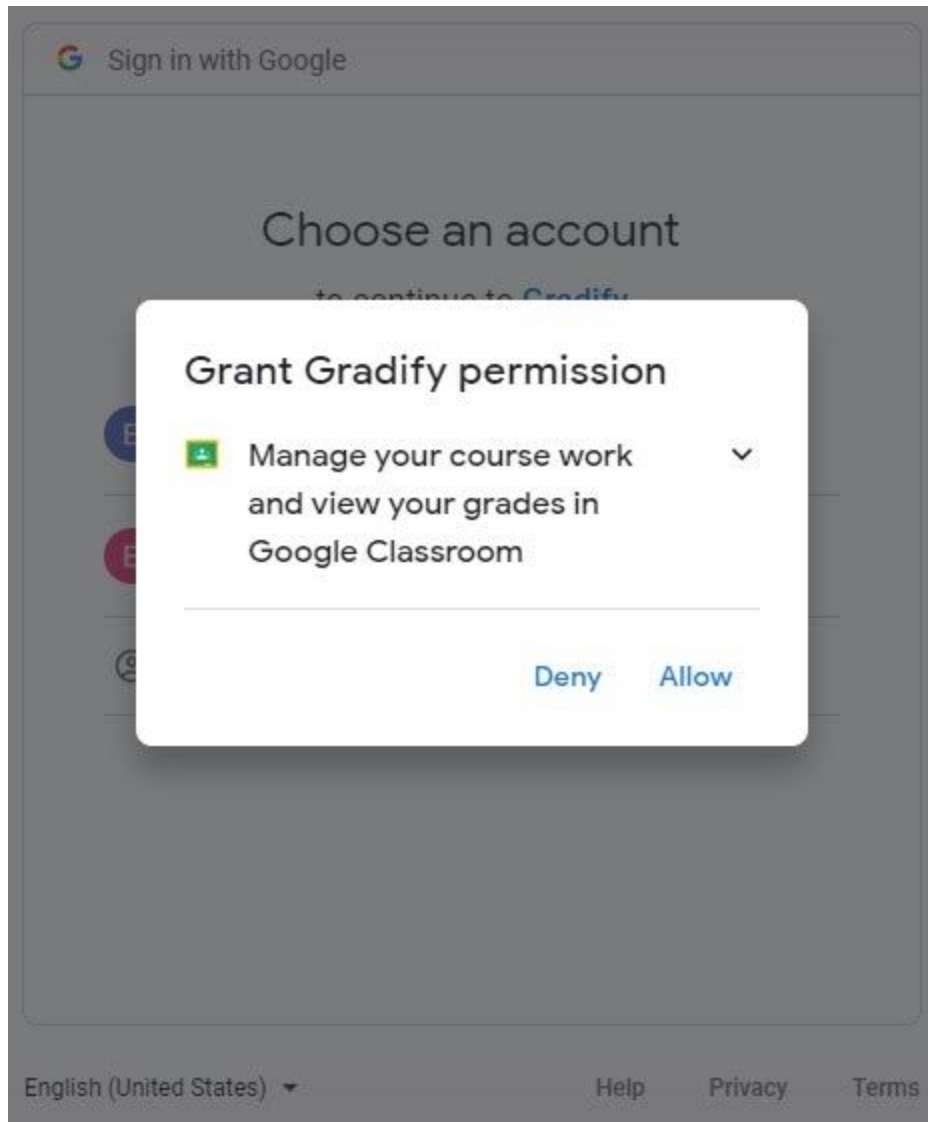


Figure 14: Gradify permissions

The next screen is a confirmation screen showing the permissions the user just agreed to (Figure 15).

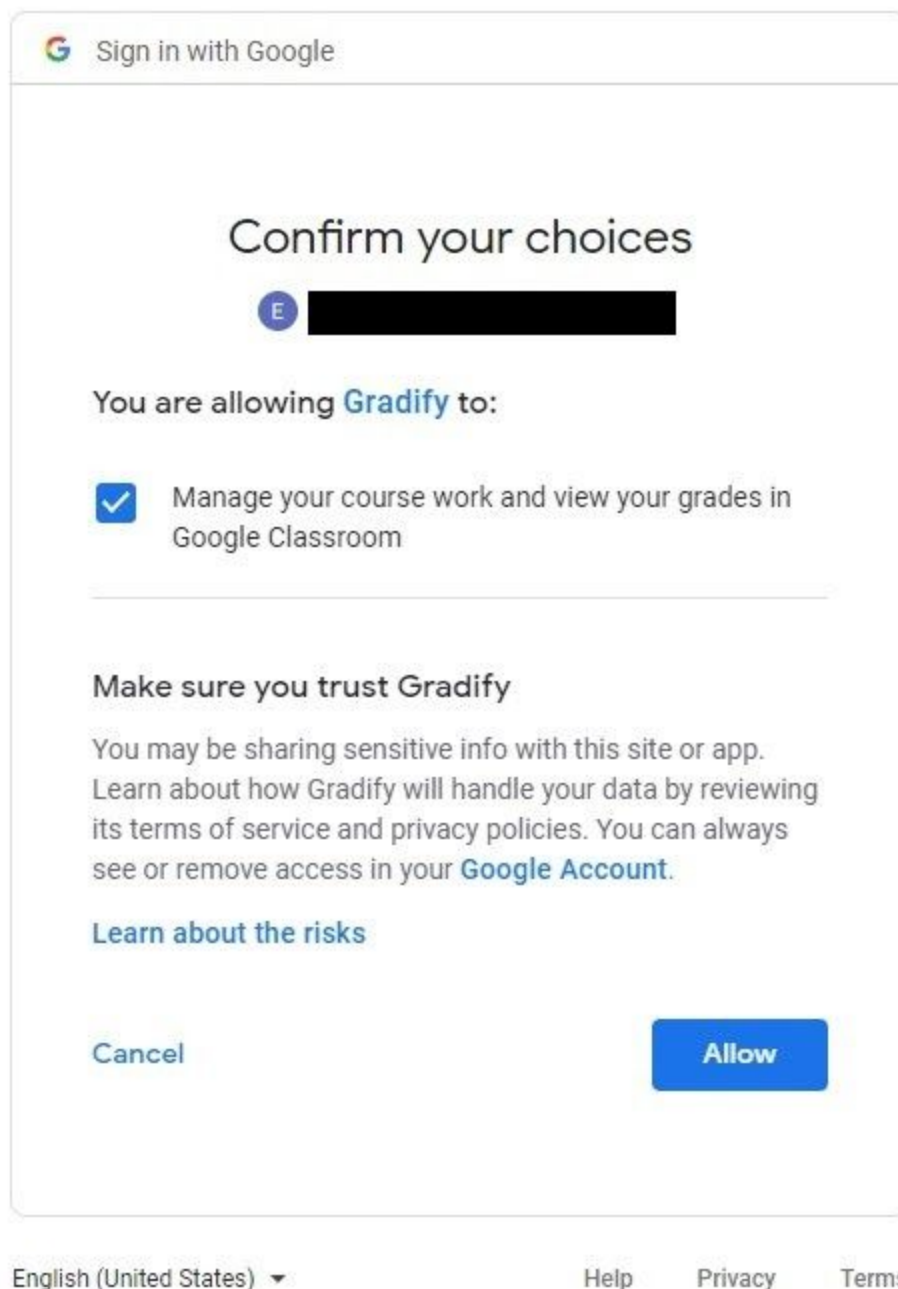


Figure 15: Gradify permissions confirmation

### Login to existing account

For users that already have a Gradify account, click the Sign in With Gradify button. (Figure 11). This takes the user to the Gradify login screen where the user should enter their email address and password and click submit (Figure 16).

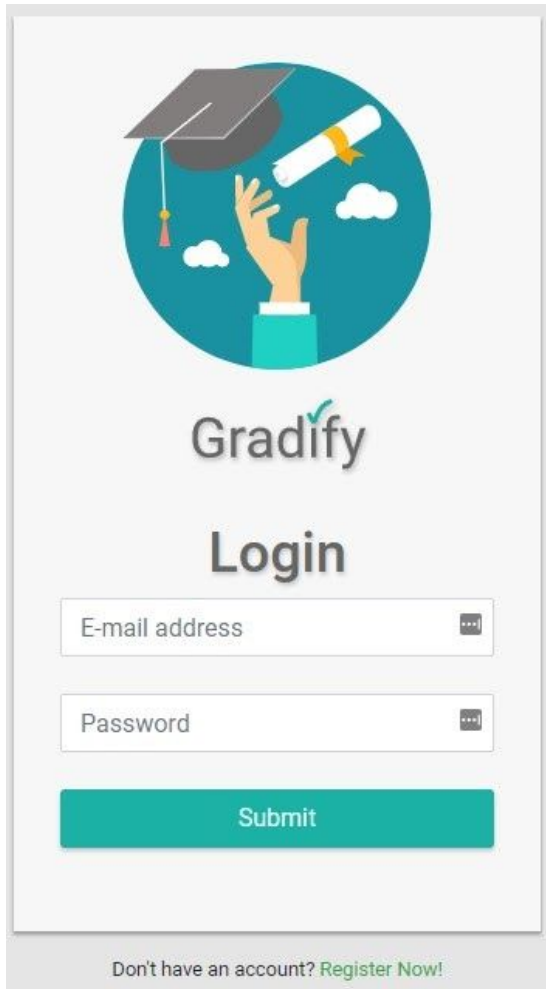
The image shows a login form for Gradify. At the top is a circular logo featuring a graduation cap, a rolled diploma, and a hand holding a pen. Below the logo is the word "Gradify" with a small green checkmark above the 'i'. Underneath is the word "Login" in a large, bold font. There are two input fields: "E-mail address" and "Password", both with placeholder text and a small "x" icon on the right. Below these fields is a teal "Submit" button. At the bottom, there is a link that says "Don't have an account? Register Now!" in green text.

Figure 16: Login with existing account

## Gradify For Teachers

### Managing Classes

After logging in, the user will be taken to the Courses Dashboard which provides an overview of all the user's existing classes (Figure 13). From this screen, teachers are able to manage their classes by importing data from Google Classroom or by creating a new course in Gradify.

#### *Importing Classes from Google Classrooms*

To import existing Google Classroom information, the user will need to be logged in through Google. Clicking Import Grades while signed in with a Gradify account will alert the user that they must sign in with Google (Figure 17).

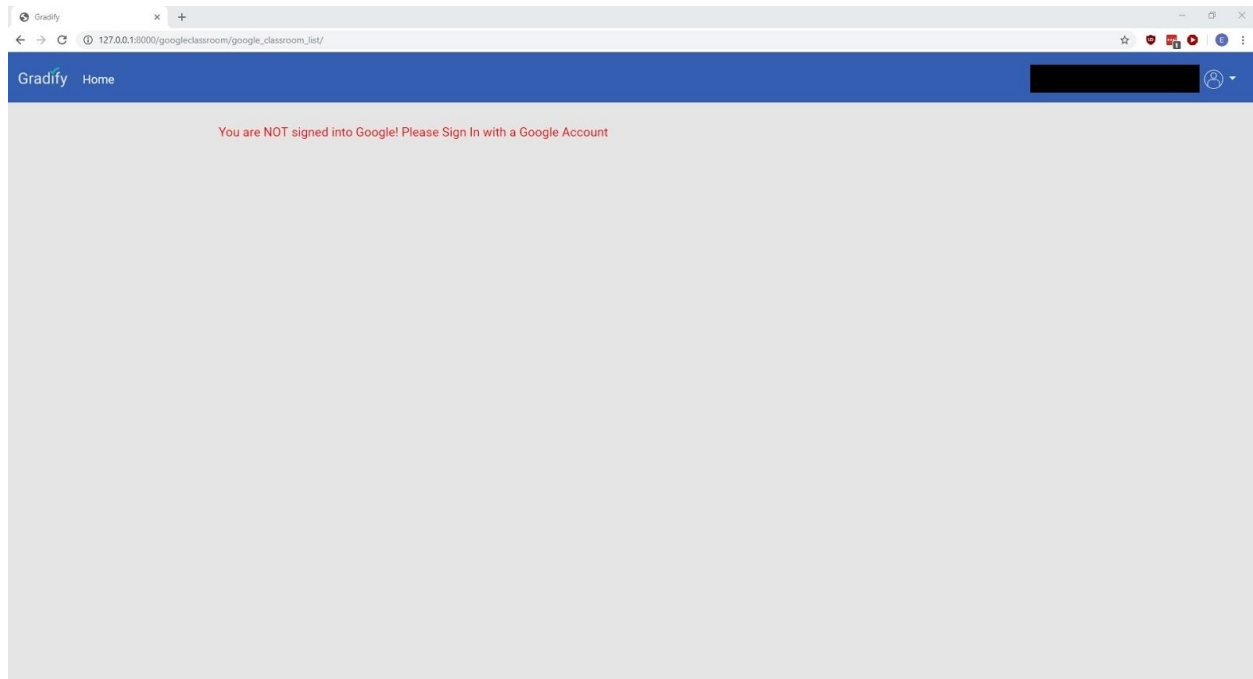


Figure 17: Google Import Error

When the user is signed in with Google and ready to import their classroom data, they must click on the Import Grades button on the Gradify Courses Dashboard page (Figure 18).

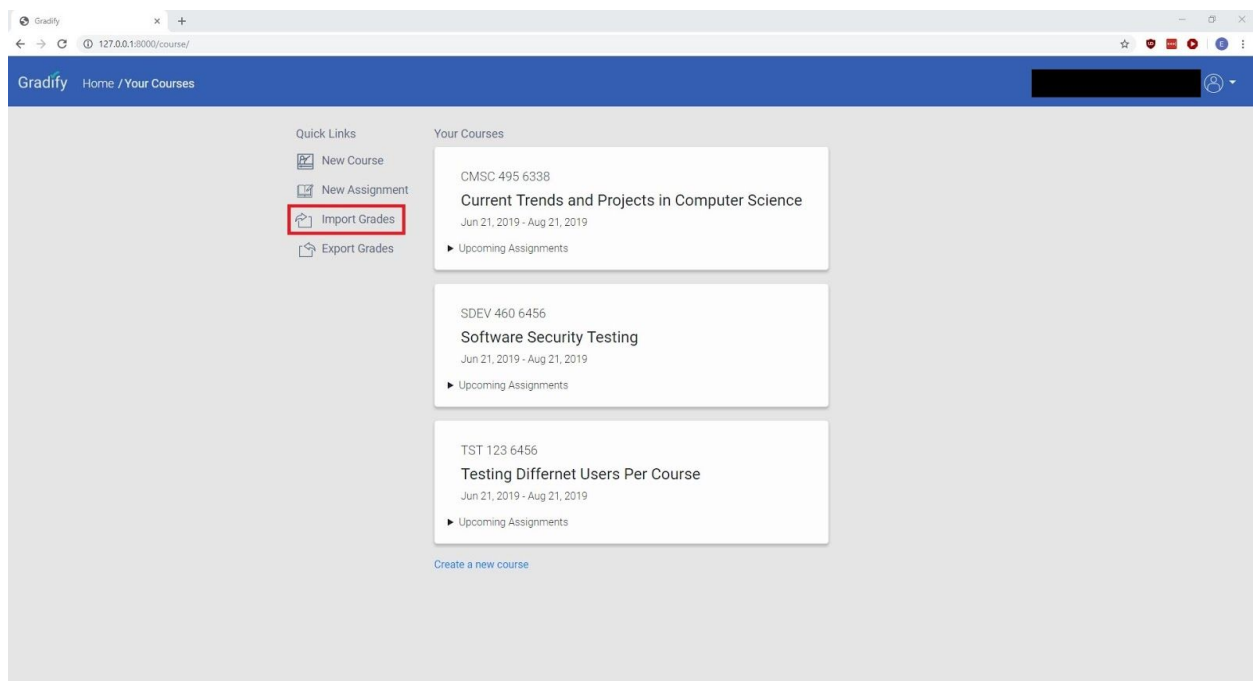


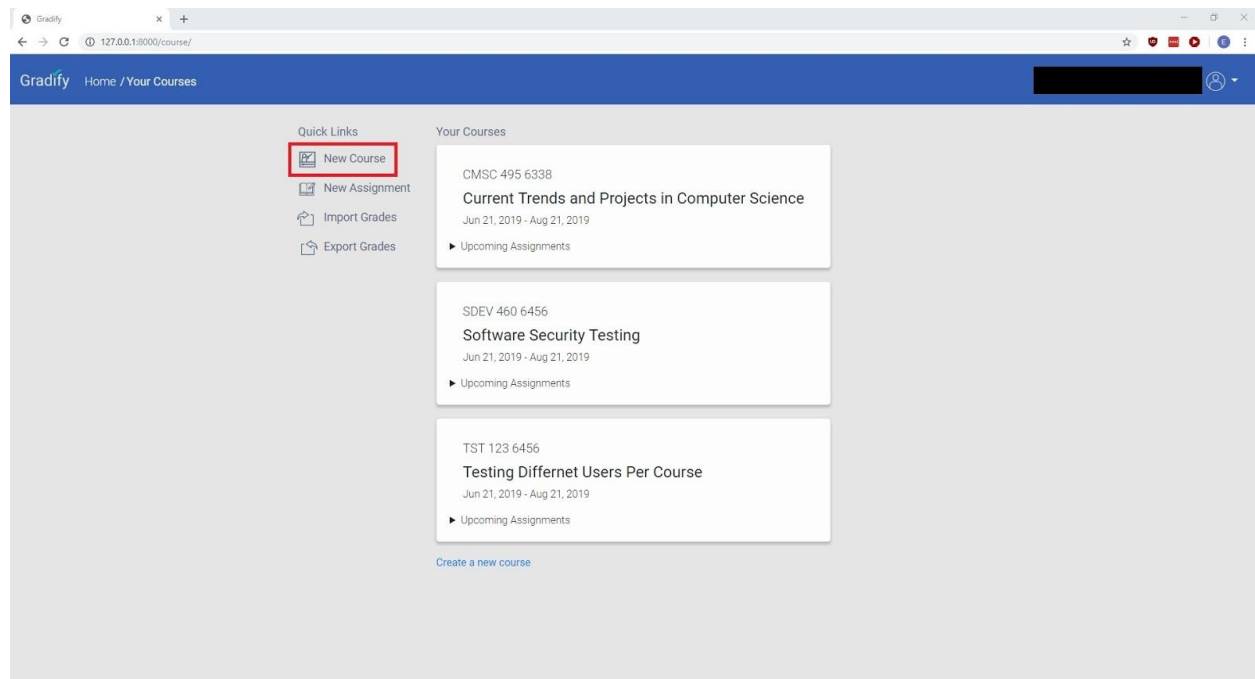
Figure 18: Import from Google Classroom

The import will run in the background and when new courses are found they will be displayed on the course list once finished.

Once the information is verified the data will be stored locally in the Gradify application. Any subsequent changes made in Google Classroom **WILL NOT** be reflected in Gradify. If changes are made in Classroom, the class will need to be imported into Gradify again to see the updates.

### *Creating a New Course in Gradify*

To create a new course in Gradify, the user can click the New Course button from the Courses Dashboard (Figure 19).



*Figure 19: Create a new course*

The user will be prompted to enter a class details (Figure 20) and the course will be created. Classes created and managed by Gradify are stored locally and any changes to assignments and grades will be updated automatically.

Create New Course

### Add Course

Name

Section

Heading

Description

Room

Link

Start Date

End Date

Submit

Figure 20: Create New Course Form

### Exporting Classroom Data

Exporting grades will create a CSV spreadsheet file that can then be imported into other programs such as a district grade book or any other program that accepts the format. To begin the export process, the user can click the Export Grades button on the class overview page (Figure 21). A file explorer will open asking them where they want to save the CSV file. The user can navigate to their desired location and hit Submit. The CSV will be created and saved to that location.

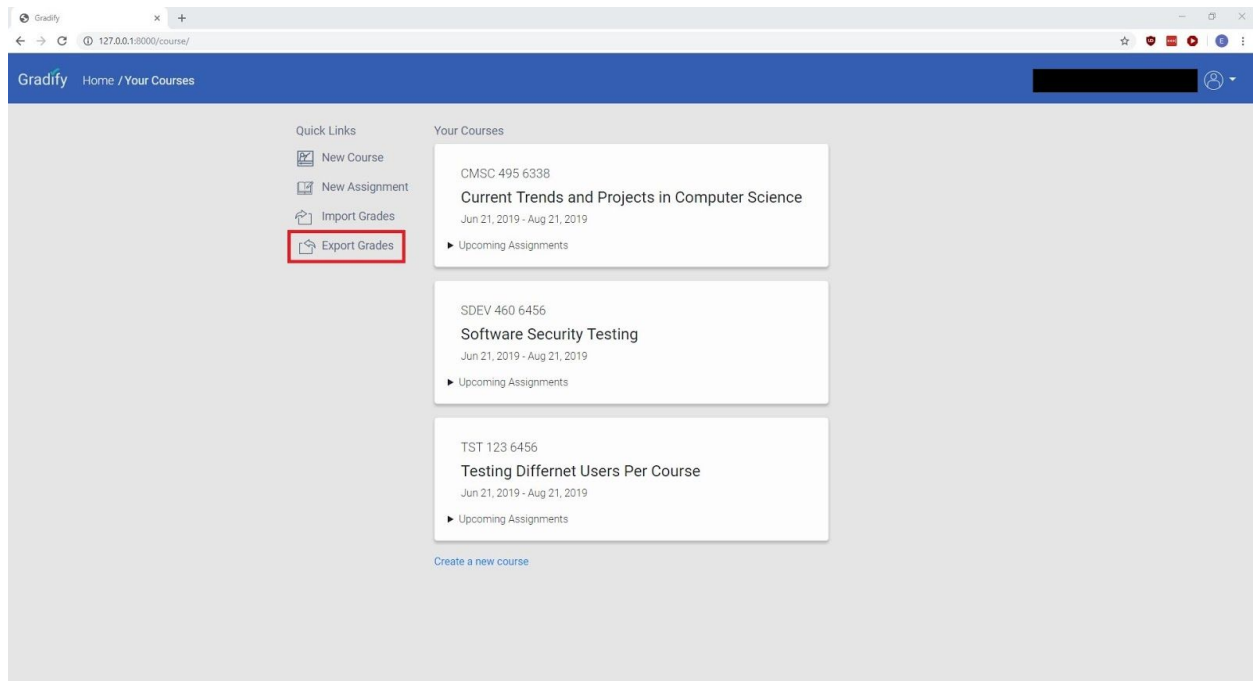


Figure 21: Export Grades

### Managing Assignments

If the user wants to keep track of assignments and grades in Google Classroom and only use Gradify to compile their data, then all activity should take place inside Classroom. In this case the user must import the class to Gradify after changes are made to see the new data. Gradify also allows users to manage assignments locally without using Google Classroom.

### Creating Assignments

To create an assignment in Gradify, the user can click the New Assignment button from the course dashboard (Figure 22).

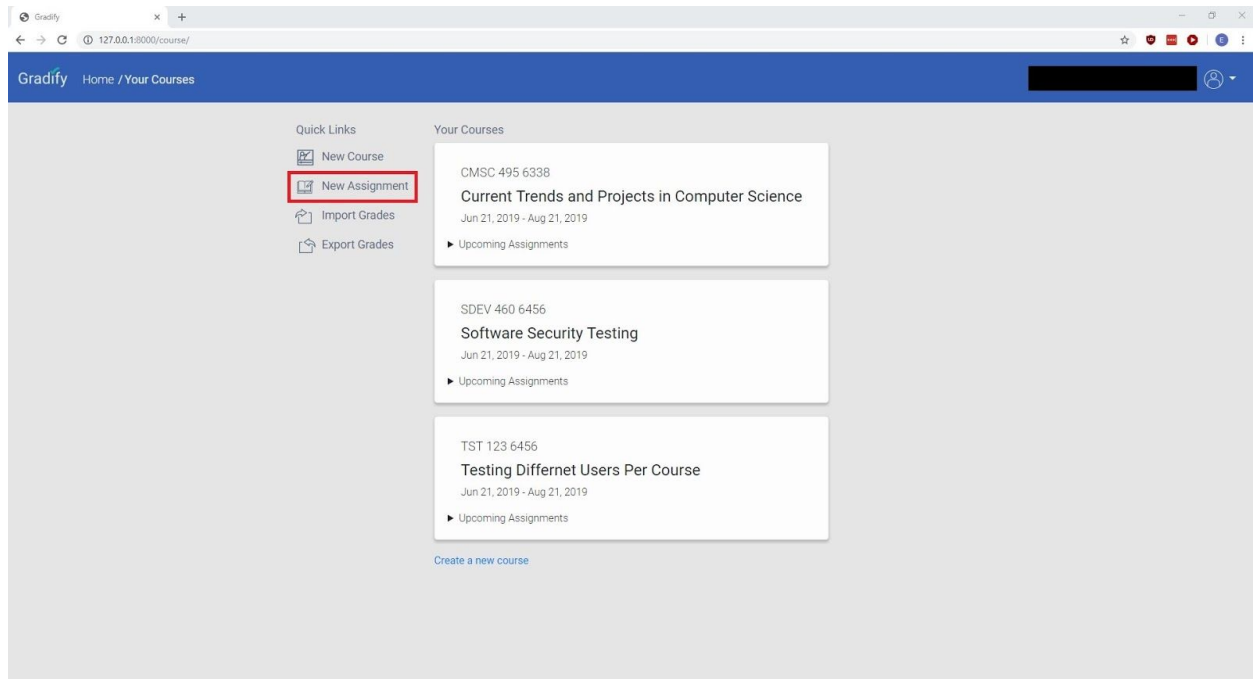


Figure 22: New Assignment

This brings up the New Assignment form where the user can enter the assignment details (Figure 23).



Create New Assignment

## Add Assignment

Course

----- ▼

Title

Description

Max Points

0

Due Date

mm/dd/yyyy

Type

Unspecified ▼

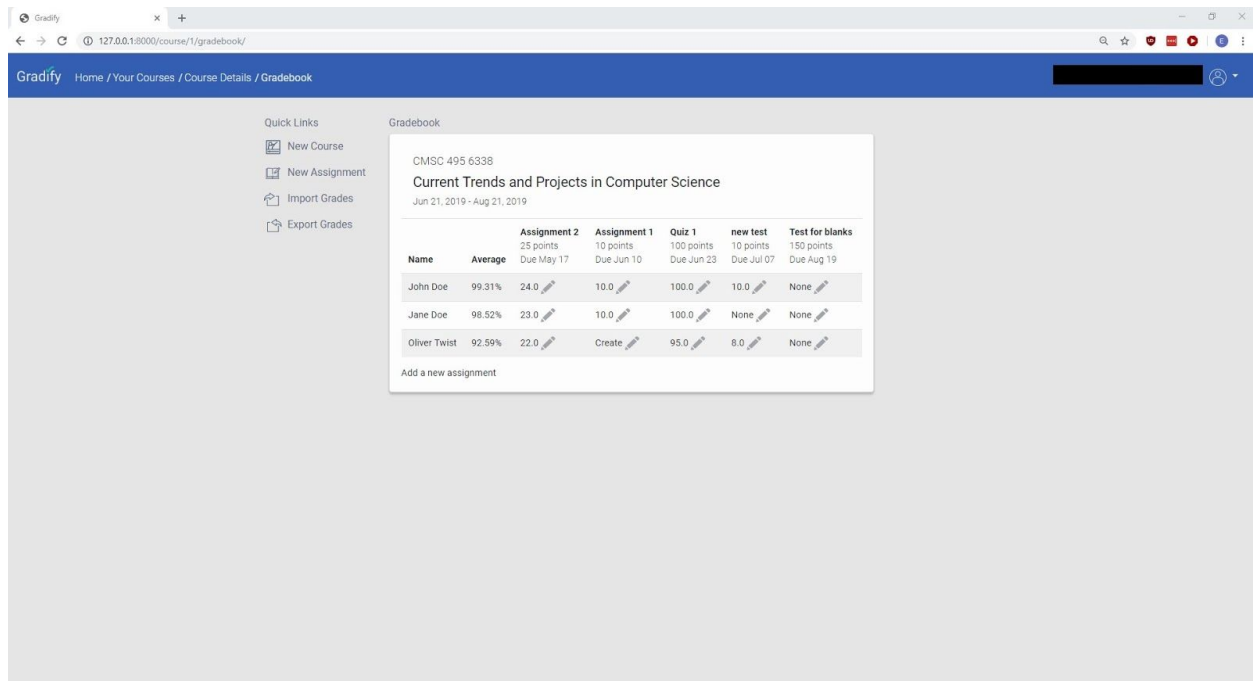
Submit

Figure 23: New Assignment Form

Once the assignment is created it will be visible in the course dashboard page and all students in the class are automatically assigned to it.

### Assigning Grades

Assigning grades in Gradify can be done from the course overview page. The overview includes a list of all assignments and the students in the class (Figure 24).



Quick Links

- New Course
- New Assignment
- Import Grades
- Export Grades

Gradebook

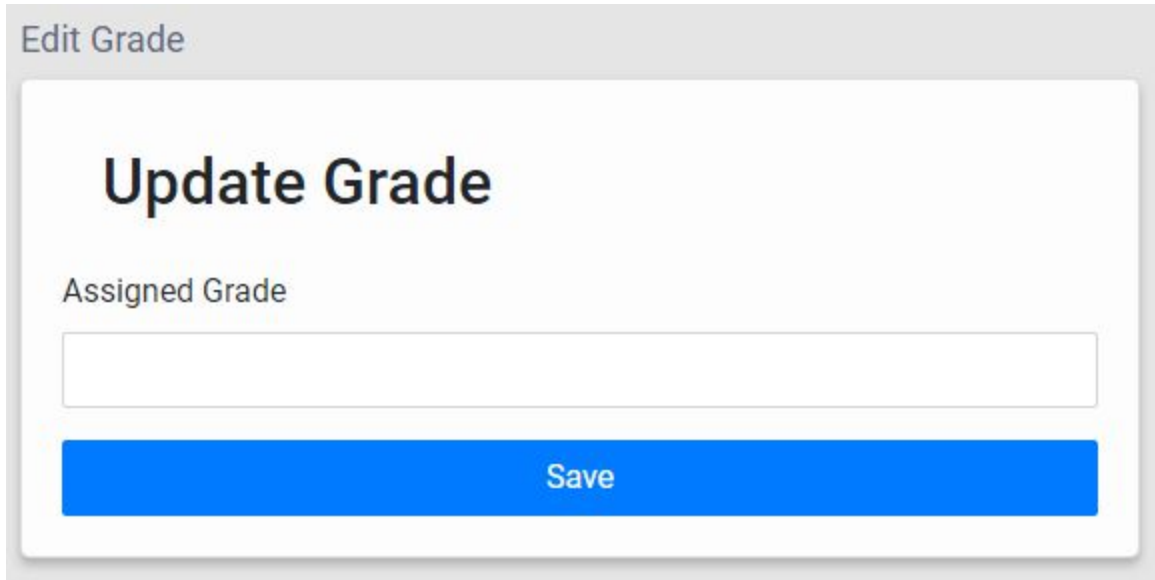
CMSC 495 6338  
Current Trends and Projects in Computer Science  
Jun 21, 2019 - Aug 21, 2019

Name	Average	Assignment 2 25 points Due May 17	Assignment 1 10 points Due Jun 10	Quiz 1 100 points Due Jun 23	new test 10 points Due Jul 07	Test for blanks 150 points Due Aug 19
John Doe	99.31%	24.0	10.0	100.0	10.0	None
Jane Doe	98.52%	23.0	10.0	100.0	None	None
Oliver Twist	92.59%	22.0	Create	95.0	8.0	None

Add a new assignment

Figure 24: Course Gradebook

To add/change a grade, the user selects the appropriate row and column on the overview and enters that student's grade in the update grade form (Figure 25).



The image shows a web form titled "Edit Grade". At the top, the text "Edit Grade" is displayed in a light blue font. Below this, the main heading "Update Grade" is centered in a large, bold, black font. Underneath the heading, the label "Assigned Grade" is followed by a large, empty white text input field with a thin grey border. At the bottom of the form, there is a prominent blue rectangular button with the word "Save" written in white, centered text.

Figure 25: Edit Grade Form

## Managing Courses

The Course Details page allows the user to manage class rosters, the class gradebook, and class assignments (Figure 26). The user can use these links to make changes to existing Gradify courses if certain details change.

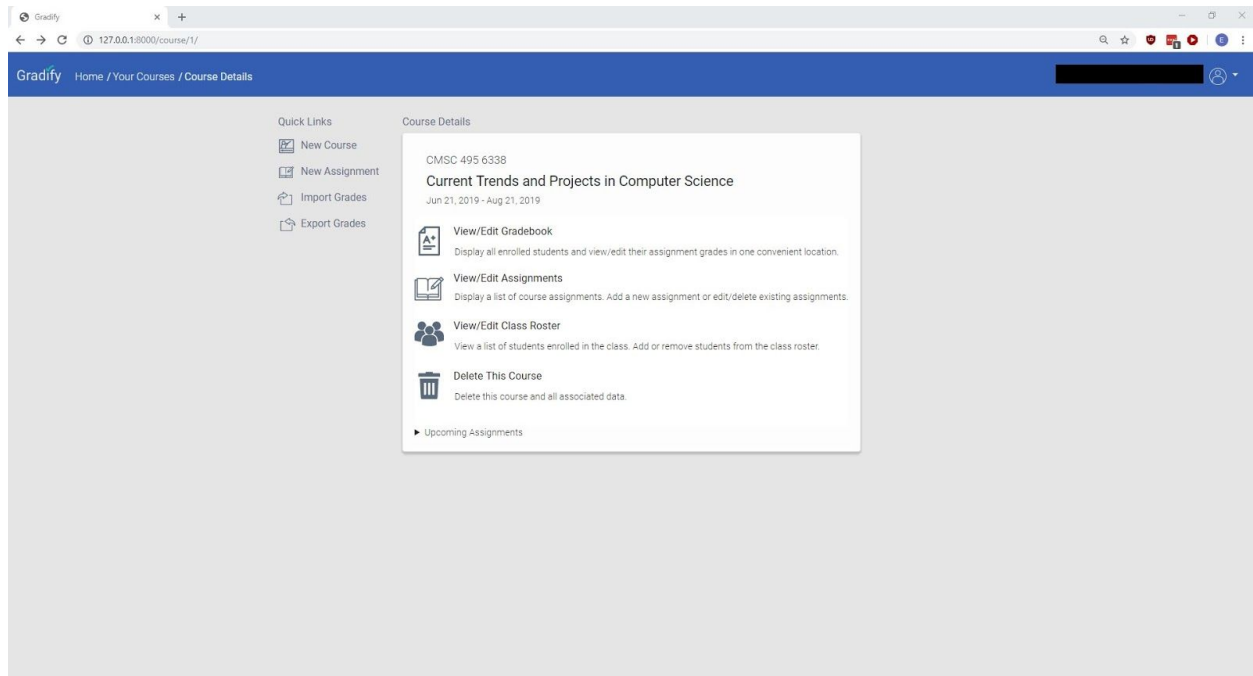


Figure 26: Edit Grade Form

# Test Plan

## Test Plan Overview

### Objectives

The primary objective of this test plan is to ensure all aspects of the application meet the customer's requirements by thoroughly evaluating the functionality, the graphical user interface, and data integrity of the system for any and all defects (bugs) associated with the application. The second objective is to ensure all modifications to the requirements are included in this plan and that all members of the quality assurance team effectively communicate with one another.

This master test plan will achieve the following objectives:

- Establish unit test requirements that shall be enforced during development.
- Create process and controls around software defect reporting and remediation during software development.
- Develop Test Cases that identify how Epic user stories are validated and tested for accuracy.

### Scope

Testing all possible situations that may exist for a piece of software is unfeasible in terms of time and cost. A test plan can only evaluate specific items that have been identified and are able to be controlled by the project. The project can control and test the software to make sure it adheres to a specific use case (in scope), for example, but it cannot control and test an end user's internet connection to the software (out of scope). Specific test cases identified in this document are the only In Scope items to be tested. Any test, condition, or situation that is not listed in this document shall be considered out of scope and will not be tested, evaluated, or assured to be accurate.

### Tactics

The Scrum Agile method shall be used to develop Gradify. Each backlog item shall undergo automated testing to make sure that the submission meets coding requirements such as coding style and merge conflicts. Each developer is also required to create and maintain their own automated unit test that can be reused for integration and regression testing. The identified Epics and User Stories shall be evaluated by both the development team and the project stakeholders on a weekly basis, as per the test schedule, to make sure that the software is adhering to the identified outputs and requirements of the project.

### Testing Strategy

The following test strategies will be performed throughout the life of this project. The test strategy begins with development where Unit Tests evaluates code before it is merged into the software solution. Integration and Regression tests then validate that changes have no impact

to other parts of the software solution. Quality Assurance (QA) and User Acceptance Testing (UAT) makes sure that the end product matches stakeholder expectations. If any test does not meet its expected outcome, then new development will occur until each test case passes, as shown in Figure 27.

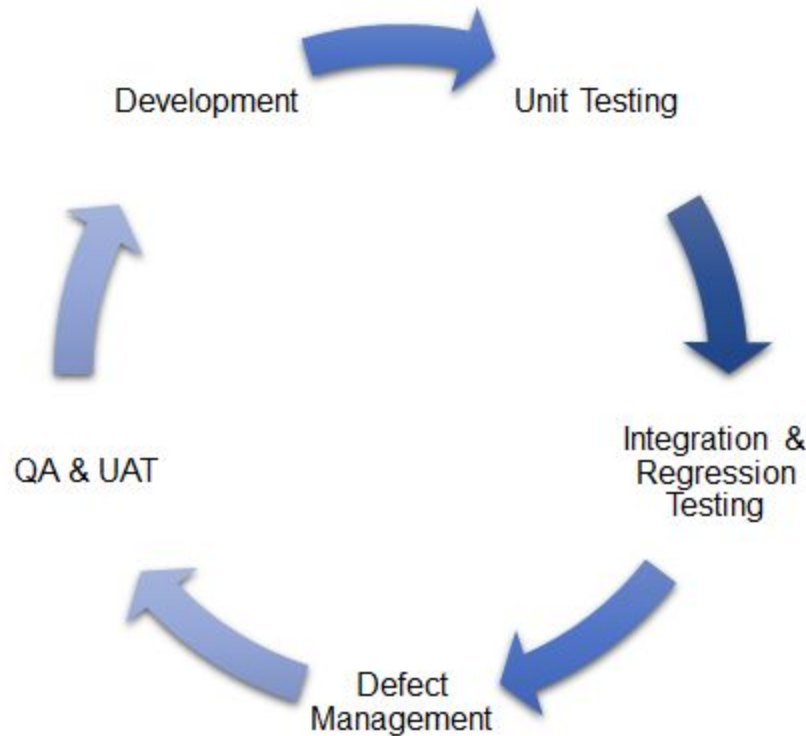


Figure 27: Testing Life Cycle Workflow

## Unit Testing

### Definition

Each piece of code can be thought of as a single unit. Unit testing is the evaluation of that unit of code to make sure it is developed correctly and within specifications. This type of testing occurs during the software development phase of the project.

### Participants

- Software Development Team

### Methodology

The Django web framework is used to support rapid agile software development. Test automation is used as part of this platform such that each unit of code has one or more corresponding test cases that validate that code for accuracy (Django, 2019). Each testcase asserts expected outputs for provided inputs. Expected outputs may not just include valid return values, but also may return exceptions for known error conditions. Each developer will be required to execute the associated test cases before submitting

their code into the code repository, GitLab. GitLab has also been configured to provide the following automated workflows through the usage of the following plugins and hooks (GitLab, 2019).

- Unit Tests: Invoke the Django unit tests
- Flake8: Ensures proper Python coding styles are enforced
- Pyint: Provides informational style hints and coding recommendations
- Bandit: Executes audits to validate security controls are enforced

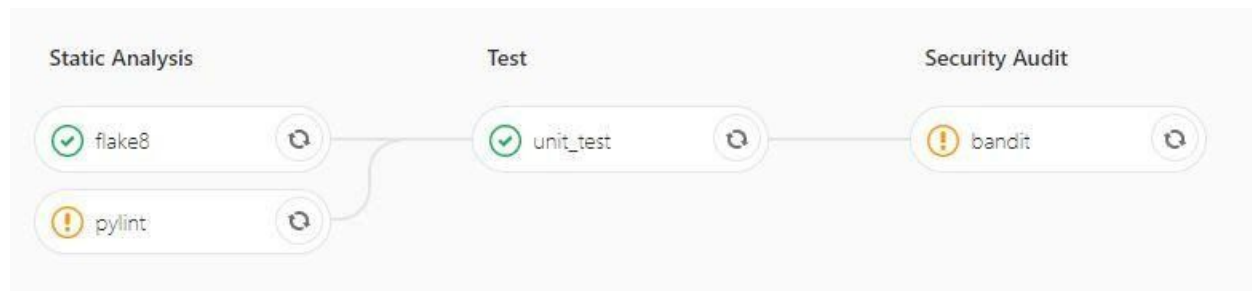


Figure 28: GitLab CI Pipeline

The GitLab CI pipeline provides instant feedback whether there are issues with any commits (Getting started with GitLab CI/CD. n.d.). Each developer will ensure their commits pass all voting tests prior to a merge request. Code reviewers can use the automated testing to ensure code quality quickly to avoid reviewing poor code.

## Integration Testing

### Definition

Integration Testing ensures that each software component and module is integrated correctly such that each part works in unity with each other. A component and module includes any class, software library, external API, or any other logical piece of software that is included in the final software solution.

### Participants

- Software Development Team

### Methodology

Each developer will need to make sure that a Django Unit Test is created anytime a call to an external service or included module is used. Acceptance criteria should include normal operating inputs & expected outputs, while other criteria such as external service unavailability will need to be included as part of the test. The call to Google Classroom API to pull student grades, for example, should include both a valid classroom Id as a success and an a invalid classroom id as an expected and handled failure (Google Inc.).

## Regression Testing

### Definition

Regression Testing assures that code that once worked is still working as the software is continually updated. New enhancement and bug fix revisions should ensure that each update does not cause a previously corrected bug from coming back or cause new issues from being created.

### Participants

- Software Development Team
- Primary Stakeholders

### Methodology

Each change to the software shall undergo Unit Testing. Each bugfix and enhancement shall include one or more Unit Tests to ensure that the new change is working correctly at the time the change is submitted and when future changes are made. The Primary Stakeholders may evaluate changes as needed to ensure the solution is continues to function as required.

## Defect Management

### Definition

Bugs or other issues will be identified during the development of this solution. These errors will arise no matter how well a developer may code their solution or how well a feature may behave. A process is needed to ensure that these bugs and issues are gathered and logged in a central place for reporting

### Participants

- Software Development Team
- Primary Stakeholders

### Methodology

Peer reviews will be conducted weekly so that another developer can review others' submitted code to ensure accuracy. Peer reviews can catch up to 60 percent of defects earlier in the development phase, saving time (Boehm and Basili, p. 136). GitLab shall be used to record, track, and manage software defects as they are found and corrected through the software development process. The *Control Procedures* section below defines the controls governing Defect Management to include how bugs are reported, tracked, and mitigated during the Software Development Life Cycle (SDLC).

## Quality Assurance & User Acceptance Testing

### Definition:

The purpose of Quality Assurance (QA) and User Acceptance Testing (UAT) is to ensure that the software provides the expected behavior as defined in the Epics and User Stories documented in the Project Plan. The test criteria for each user story is defined as a Use Case within the *Features to be Tested* section below. A Use Case describes how a user shall use the system to achieve an expected outcome (Larson and Larson, 2004). QA testing continually executes these Use Cases throughout software development to ensure each feature behaves as expected without creating unnecessary confusion or frustration to the end user. UAT shall be the final review and acceptance by the primary stakeholder to establish that all requested features and functions operate to specifications.

**Participants:**

- Software Development Team
- Project Stakeholders

**Methodology:**

Each test shall have one or more user(s) take the role of an Actor to interact with software solution and perform the steps identified in each Use Case as that Actor role. Every action performed shall be evaluated, where specified, to make sure the behavior and response of the software solution meets the acceptance criteria of that Use Case.

## Test Schedule

A Sprint Review will be conducted to review all work that has occurred over the past week. Peer code reviews are conducted as Merge Requests are created in Gitlab to validate the accuracy of completed tasks. Primary stakeholders will be requested to review features as they are completed to solicit feedback and check for solution accuracy.

## Control Procedures

### Problem Reporting

The team will be using GitLab's *Issues* (Issues. n.d.) feature to track problems encountered during testing. If the problem cannot be easily resolved, a bug report will be filed which describes at a minimum the component in which the problem occurred, the specific actions that led to the problem, and a description of the intended behavior. The GitLab Issue will be assessed by the team and assigned to a team member for remediation. If possible, this team member will also create an automated regression test to ensure that the team is alerted if the problem resurfaces during development.

### Change Requests

Potential modifications to the software must be proposed to the team during planning meetings. Any proposed modification should improve the quality of the software, support project



requirements, and be feasible to complete within the time allotted for the project. Team members suggesting a change should provide justification as to why the modification meets these criteria. The proposal will be assessed as a team. If it is decided that the requested change is justified, a GitLab Issue will be created and added to the product backlog.

## Test Cases

The following Test Cases have been created from the Project Plan's Epics and User Stories and includes other system tests that help maintain system integrity and security controls. Each Test Case and shall be used to evaluate the performance of this software solution.

### Test Case Template

The following test case template will be used through this test plan.

Test Case #[number]			
Test Case Name		Execute Date	
Priority (Low/Medium High)		Executed By	
Use Case			
Description			

Step #	Actions Taken	Expected Results	Actual Results	Pass/Fail

### Test Cases

The following test cases will be evaluate specific capabilities and features of the software solution. Each Test Case includes what is being tested, the specific steps to reproduce the action, the expected outcome, and the actual observed outcome. If the expected outcome matches the observed outcome then the Test Case will be marked as a *Pass*, otherwise the Test Case will be marked as a *Fail* and will need remediation through Defect management.

### Test Case #1

<b>Test Case Name</b>	User Signon	<b>Execute Date</b>	July 13, 2019
<b>Priority (Low/Medium High)</b>	High	<b>Executed By</b>	Eric Hutchins
<b>Use Case</b>	User Security (Valid Credential)		
<b>Description</b>	Verify that only a valid user can login in		

Step #	Actions Taken	Expected Results	Actual Results	Pass/Fail
1	User enters a <i>username</i> and <i>password</i>	Form accepts username and password	Parameters accepted	Pass
2	User clicks "Sign On"	System authenticates user and allows them entry into web site	User redirected to courses page	Pass

### Test Case #2

<b>Test Case Name</b>	Invalid User Signon	<b>Execute Date</b>	July 13, 2019
-----------------------	---------------------	---------------------	---------------

<b>Priority (Low/Medium High)</b>	High	<b>Executed By</b>	Ken Lovingood
<b>Use Case</b>	User Security (invalid credential)		
<b>Description</b>	Verify that invalid users cannot login		

Step #	Actions Taken	Expected Results	Actual Results	Pass/Fail
1	User enters a <i>username</i> and <i>password</i>	Form accepts username and password		Pass
2	User clicks "Sign On"	System presents the user with an "Username or password is incorrect" error message"	For displays "Please enter a valid email address" when attempting to login. Access is not granted	Pass

### Test Case #3

<b>Test Case Name</b>	User Account Create	<b>Execute Date</b>	July 11, 2019
<b>Priority (Low/Medium High)</b>	High	<b>Executed By</b>	Dakin Werneburg
<b>Use Case</b>	User Security (Account Create)		
<b>Description</b>	Verify that a new user can create an account		

--	--

Step #	Actions Taken	Expected Results	Actual Results	Pass/Fail
1	User navigates to Create new User screen	User is taken to user create screen	Registration page pops up and asks for information	Pass
2	Enter First Name, Last Name, organization, email, and their role where prompted and clicks <i>Submit</i>	Web page opens and user is taken to course dashboard listed user courses	Dashboard page loads with user courses listed. Since this was a new account no courses were listed.	Pass

## Test Case #4

<b>Test Case Name</b>	User Change Password	<b>Execute Date</b>	July 13, 2019
<b>Priority (Low/Medium High)</b>	High	<b>Executed By</b>	Ken Lovingood
<b>Use Case</b>	User Security (Change Password)		
<b>Description</b>	Verify that a user can change their password		

Step #	Actions Taken	Expected Results	Actual Results	Pass/Fail
--------	---------------	------------------	----------------	-----------

1	User clicks "Sign On"	User is taken to Sign On screen	Page displayed	Pass
2	User clicks "Change Password" link	User is taken to Change Password screen	User is prompted for old password and new password	Pass
3	User enters a new password twice	Complex passwords are accepted and the user is prompted to log in again.	New password is accepted and the user is taken to login page	Pass

### Test Case #5

<b>Test Case Name</b>	Invalid User Change Password	<b>Execute Date</b>	July 13, 2019
<b>Priority (Low/Medium High)</b>	High	<b>Executed By</b>	Ken Lovingood
<b>Use Case</b>	User Security (google accounts cannot change password)		
<b>Description</b>	Verify that only local accounts can change passwords		

Step #	Actions Taken	Expected Results	Actual Results	Pass/Fail
1	User clicks "Sign On"	User is taken to Sign On screen	User logs into application using google account	Pass
2	User clicks "Change Password" link	User is taken to Change Password screen	Link is displayed	Pass

3	User cannot change password message	User is notified that they cannot change their password	Message is displayed: "Google users cannot reset their password in Gradify Please set your password in Google"	Pass
---	-------------------------------------	---	--	------

## Test Case #6

<b>Test Case Name</b>	View Courses	<b>Execute Date</b>	July 13, 2019
<b>Priority (Low/Medium High)</b>	Medium	<b>Executed By</b>	Eric Hutchins
<b>Use Case</b>	Course Dashboard (Course Listing)		
<b>Description</b>	Verify that the user's course dashboard displays a list of all of their courses		

Step #	Actions Taken	Expected Results	Actual Results	Pass/Fail
1	Click "Your Courses" Link	User is taken to Course Dashboard	Courses page is displayed as the main landing page.	Pass

## Test Case #7

<b>Test Case Name</b>	Add Course	<b>Execute Date</b>	July 13, 2019
<b>Priority (Low/Medium High)</b>	Medium	<b>Executed By</b>	Ken Lovingood
<b>Use Case</b>	User's Course Dashboard (course overview)		

<b>Description</b>	Verify that a Teacher can add a course to their course list.
--------------------	--

Step #	Actions Taken	Expected Results	Actual Results	Pass/Fail
1	Click "Your Courses" Link	User is taken to Course Dashboard	Course list is displayed.	Pass
2	Click "New Course" under the Quick Link	New Course entry form displays	"Create a new course" link is displayed	Pass
3	Enter New Course information and click "Submit" button	Return to course overview and new course is displayed in course list	Form is completed and submitted. Form redirects back to courses page and new course is present.	Pass

## Test Case #8

<b>Test Case Name</b>	View Class	<b>Execute Date</b>	July 13, 2019
<b>Priority (Low/Medium High)</b>	Medium	<b>Executed By</b>	Ken Lovingood
<b>Use Case</b>	User's Course Dashboard (class overview)		
<b>Description</b>	Verify that user can access more information about the individual course		

Step #	Actions Taken	Expected Results	Actual Results	Pass/Fail
--------	---------------	------------------	----------------	-----------

1	Click "Your Courses" Link	User is taken to Course Dashboard	Courses is displayed	Pass
2	Click the name of the course on the Course Dashboard	User is taken to the Class Dashboard	User has options to view the gradebook, view assignments, and view the roster.	Pass

## Test Case #9

<b>Test Case Name</b>	Delete Course	<b>Execute Date</b>	July 13, 2019
<b>Priority (Low/Medium High)</b>	High	<b>Executed By</b>	Eric Hutchins
<b>Use Case</b>	User's Course Dashboard (delete course)		
<b>Description</b>	Verify that a Teacher can delete multiple courses at time		

Step #	Actions Taken	Expected Results	Actual Results	Pass/Fail
1	Click "Your Courses" Link	User is taken to Course Dashboard	Courses is displayed	Pass
2	Click on course	The Course Details page is displayed	The Course Details is displayed providing a link to "Delete the Course"	Pass
3	Click "Delete"	Course Dashboard is refreshed with updated list of courses	Clicking the "Delete This Course" link takes the user to a confirmation page. If the user clicks "delete" then the course is	Pass



			deleted and the user is taken back to the course details page where the old course is no longer listed.	
--	--	--	---	--

## Test Case #10

<b>Test Case Name</b>	Add assignment	<b>Execute Date</b>	July 13, 2019
<b>Priority (Low/Medium High)</b>	Medium	<b>Executed By</b>	Ken Lovingood
<b>Use Case</b>	Teacher Class Management		
<b>Description</b>	Verify that a Teacher can add an assignment to a class		

Step #	Actions Taken	Expected Results	Actual Results	Pass/Fail
1	Click "Your Courses" Link	User is taken to Course Dashboard	Courses are displayed.	Pass
2	Click "New Assignment" quick link	New assignment form displays	New Assignment Form is displayed.	Pass
3	Add assignment details (name, description, max. points) and click "Submit"	Assignment is accepted and user is returned to Class page where new assignment is visible	Web form is completed and new assignment is listed for the course.	Pass

## Test Case #11

<b>Test Case Name</b>	Import Google Classroom	<b>Execute Date</b>	July 13, 2019
<b>Priority (Low/Medium High)</b>	Medium	<b>Executed By</b>	Ken Lovingood
<b>Use Case</b>	Teacher Class Management		
<b>Description</b>	Verify that a Teacher can import classroom data from Google Classroom		

Step #	Actions Taken	Expected Results	Actual Results	Pass/Fail
1	Click "Your Courses" Link	User is taken to Course Dashboard	Course list is displayed.	Pass
2	Click "Import Grades" quick link	Import course page displays	Page cycles while import is in progress	Pass
3	Course List is returned showing imported courses	New course appears in the course list.	Once the import is complete, the course list is displayed and the new courses are viewable. Navigating into the course shows that the assignments and grades are present as well.	Pass

## Test Case #12

<b>Test Case Name</b>	Export Course Data	<b>Execute Date</b>	July 13, 2019
-----------------------	--------------------	---------------------	---------------

<b>Priority (Low/Medium High)</b>	Medium	<b>Executed By</b>	Ken Lovingood
<b>Use Case</b>	Teacher Class Management		
<b>Description</b>	Verify that a Teacher can export data to a CSV file		

Step #	Actions Taken	Expected Results	Actual Results	Pass/Fail
1	Click "Your Courses" Link	User is taken to Course Dashboard	Course list is displayed.	Pass
2	Click "Export Grades" quick link	File Explorer displays	Link is clicked and the user is presented a download prompt for the submissions.csv file.	Pass
3	Enter name of file and select export location and click "Submit"	CSV file is created at the selected location	File contents shows all courses and the rollup of grades for each student per teacher's course.	Pass

### Test Case #13

<b>Test Case Name</b>	Student Assignments	<b>Execute Date</b>	July 13, 2019
<b>Priority (Low/Medium High)</b>	Medium	<b>Executed By</b>	Ken Lovingood
<b>Use Case</b>	Managing Assignments Dashboard		

<b>Description</b>	Verify that a Teacher can view a list of students that displays their grades for each assignment on one page.
--------------------	---

Step #	Actions Taken	Expected Results	Actual Results	Pass/Fail
1	Click "Your Courses" Link	User is taken to Course Dashboard	Course list is displayed.	Pass
2	Course Details View is displayed	Course details is displayed prompting the user to view the gradebook.	View Gradebook is displayed and user can click on link	Pass
2	Click "Assignment Overview"	List appears with students assignments and grades	All assignments and student grades are listed.	Pass

## Test Case #14

<b>Test Case Name</b>	Modify Assignment	<b>Execute Date</b>	July 13, 2019
<b>Priority (Low/Medium High)</b>	Medium	<b>Executed By</b>	Ken Lovingood
<b>Use Case</b>	Teacher Class Management		
<b>Description</b>	Verify that a Teacher can modify the details of an individual assignment		

Step #	Actions Taken	Expected Results	Actual Results	Pass/Fail
1	Click "Your Courses" Link	User is taken to Course Dashboard	Course list is displayed.	Pass
2	Course Details View is displayed	Course details is displayed prompting the user to view the gradebook.	User clicks on the View/Edit Assignments link.	Pass
3	Assignment List	Assignment list is displayed prompting the user to select which assignment to view.	The assignment list is displayed with a pencil next to each assignment name.	Pass
3	Make modification to the available fields and click "Save"	Message states the data was successfully modified in the database and user taken to the Class Dashboard	Pencil is clicked and the details of the assignment is shown to the user. Modifications can be made and is correctly saved.	Pass

## Test Case #15

Test Case Name	Grade Assignment	Execute Date	July 13, 2019
Priority (Low/Medium High)		Executed By	Ken Lovingood
Use Case	Teacher Class Management		
Description	Verify that a Teacher can grade an assignment		

Step #	Actions Taken	Expected Results	Actual Results	Pass/Fail
1	Click “Your Courses” Link	User is taken to Course Dashboard	Course list is displayed.	Pass
2	Course Details View is displayed	Course details is displayed prompting the user to view the gradebook.	View Gradebook is displayed and user can click on link. Click on the pencil to modify a student’s grade.	Pass
3	Make modification to the available fields and click “Save”	Message states the data was successfully modified in the database and user taken to the Class Dashboard	Option is displayed to modify or add a user’s grade	Pass

## Resource, Roles, and Responsibilities

### Roles

Role	Description
<b>DBA</b>	Tests and validates proper storage and integrity of data throughout testing
<b>Documentation Manage</b>	Maintains document versioning and recording of test results
<b>GitLab Control Manager</b>	Maintains FitLab backlog, bug tracking, and other defect management process controls
<b>GitLab Automation Lead</b>	Maintains automation workflows that are used to execute unit tests, code style enforcement, and other types of test automation executed inside GitLab
<b>Project Stakeholder Advocate and Liaison</b>	Coordinates changes, change review, Quality Assurance, and Unit Testing with Project Stakeholders
<b>Scrum master</b>	Leads Scrum review sessions that include peer code reviews, Scrum backlog item review, and coordinates peer demonstrations

--	--

## Resources

Name	Test Roles
<b>Dakin Werneburg</b>	Software Development Team member, DBA
<b>Davy Marrero</b>	Software Development Team member, GitLab Control Manager
<b>Eric Hutchins</b>	Software Development Team member, Project Stakeholder Advocate and Liaison
<b>Kenneth Lovingood</b>	Software Development Team member, Documentation Manager
<b>Ryan Brady</b>	Software Development Team member, Scrum master, GitLab Automation lead

## Major Deliverables

The following deliverables will be made available during testing and once testing is complete:

- Test Plan
- Test Cases Results
- User Guide
- Weekly GitLab Bug Tracking Reports

## Tools

The following tools will be used to provide support testing and test automation.

- **GitLab**: Code Source Control, Scrum backlog, and defect tracking and management
- **Unit Tests**: Invoke the Django unit tests
- **Coverage**: Provides information on percentage of code tested
- **Flake8**: Ensures proper Python coding styles are enforced
- **Pyint**: Provides informational style hints and coding recommendations
- **Bandit**: Executes audits to validate security controls are enforced

## Test Approvals

Specify the names and titles of all persons who must approve this plan. Provide space for the signatures and dates.

Printed Name	Signature	Date
--------------	-----------	------

Printed Name	Signature	Date
--------------	-----------	------

Printed Name	Signature	Date
--------------	-----------	------

## Development History

### Sprints

The Gradify development project was split into 3 development sprints.

### Phase 1 Sprint

<b>Need to determine where to store Google API token</b>
#32 · opened 3 weeks ago by Davy L. Marrero · v0.1 · <span>Needs Investigation</span> <span>sprint1</span>
<b>Add Core App Structure</b>
#29 · opened 3 weeks ago by Ryan Brady · v0.1 · <span>backend</span> <span>sprint1</span>
<b>Create course details page</b>
#24 · opened 3 weeks ago by Davy L. Marrero · v0.1 · <span>frontend</span> <span>sprint1</span>
<b>Provide teachers with a view that displays all students, grades, and assignments for a course - User Story 6</b>
#18 · opened 1 month ago by Davy L. Marrero · v0.1 · <span>backend</span> <span>feature</span> <span>frontend</span> <span>sprint1</span> <span>teacher</span>
<b>Create Course List Page - User Story 4</b>
#7 · opened 1 month ago by Davy L. Marrero · v0.1 · <span>backend</span> <span>feature</span> <span>frontend</span> <span>sprint1</span>



## Phase 2 Sprint

### Override base.html & other allauth templates

#42 · opened 2 weeks ago by Ken Lovingood sprint2

### Update all links to the correct route

#41 · opened 2 weeks ago by Davy L. Marrero sprint2

### Use slugs to create URL routes

#40 · opened 2 weeks ago by Davy L. Marrero sprint2

### Implement grade averages for the gradebook view

#39 · opened 2 weeks ago by Eric Hutchins v0.1 frontend sprint2 teacher

### Refactor Tests Into Multiple Files

#38 · opened 2 weeks ago by Ryan Brady sprint2 tech-debt

### Refactor Template Directory

#37 · opened 2 weeks ago by Ryan Brady bug sprint2 tech-debt

### Create a script for loading fixtures

#36 · opened 2 weeks ago by Davy L. Marrero sprint2

### Implement navbar interactivity

#34 · opened 2 weeks ago by Davy L. Marrero v0.1 frontend sprint2 sprint3

### Create class roster page

#25 · opened 3 weeks ago by Davy L. Marrero v0.1 frontend sprint2

### Create assignment details view - User Story 14

#20 · opened 1 month ago by Davy L. Marrero v0.1 backend feature frontend parent sprint2 student teacher

### Add functionality for connecting to Google Classroom - User Story 7

#10 · opened 1 month ago by Davy L. Marrero v0.1 backend feature sprint2

### Google Classroom Import - User Story 7

#9 · opened 1 month ago by Davy L. Marrero v0.1 backend feature frontend sprint2 sprint3 teacher

### Create Registration Page - User Story 1

#3 · opened 1 month ago by Davy L. Marrero v0.1 backend feature frontend sprint2

### Create Login Page - User Story 2

#2 · opened 1 month ago by Davy L. Marrero v0.1 backend feature frontend sprint2 sprint3

## Phase 3 Sprint

### Export Grades

#54 · opened 5 days ago by Ken Lovingood sprint3

### Create/style home page view for logged in users

#51 · opened 1 week ago by Eric Hutchins v0.1 sprint3

### Prevent unauthenticated users from accessing protected views

#50 · opened 1 week ago by Davy L. Marrero sprint3

### Change Password not redirecting and should not be available for google users

#48 · opened 1 week ago by Ken Lovingood sprint3

### Sign in with e-mail as your username

#47 · opened 1 week ago by Davy L. Marrero sprint3

### Rename Google API to read Gradify instead of Quickstart

#46 · opened 1 week ago by Ken Lovingood sprint3

### Implement navbar interactivity

#34 · opened 2 weeks ago by Davy L. Marrero v0.1 frontend sprint2 sprint3

### Add functionality for deleting a course - User Story 8

#27 · opened 3 weeks ago by Davy L. Marrero backend frontend sprint3

### Add password reset functionality - User Story 3

#22 · opened 3 weeks ago by Davy L. Marrero backend frontend sprint3

### Add functionality for creating new assignments - User Story 11

#16 · opened 1 month ago by Davy L. Marrero backend frontend sprint3 teacher

### Delete multiple courses at once - User Story 8

#11 · opened 1 month ago by Davy L. Marrero backend feature frontend sprint3 teacher

### Google Classroom Import - User Story 7

#9 · opened 1 month ago by Davy L. Marrero v0.1 backend feature frontend sprint2 sprint3 teacher

### Add functionality for creating a course - User Story 5

#8 · opened 1 month ago by Davy L. Marrero backend feature frontend sprint3

### Create Login Page - User Story 2

#2 · opened 1 month ago by Davy L. Marrero v0.1 backend feature frontend sprint2 sprint3

## Phase 4 Sprint

**Checkmark is not lined up with the "i" in login page logo**

#60 · opened 1 day ago by Davy L. Marrero sprint4

**Configure deployment to use HTTPS**

#59 · opened 1 day ago by Davy L. Marrero sprint4

**Use Continuous Deployment to deploy master branch to cloud**

#56 · opened 3 days ago by Davy L. Marrero sprint4

**App crashes if you try to import after your Google token has expired**

#53 · opened 6 days ago by Davy L. Marrero Needs Investigation bug sprint4

**Check if start and end dates exist before rendering in templates**

#52 · opened 1 week ago by Davy L. Marrero bug sprint4

**Remove sensitive data before deploying**

#49 · opened 1 week ago by Davy L. Marrero sprint4

**Add functionality to assign/change grades as a teacher - User Story 15**

#28 · opened 3 weeks ago by Eric Hutchins backend frontend sprint4 teacher

## GitLab Repository

Gitlab was used as both issue tracking and code commits. The following is a final extract of all work committed by team members. The activity data for issue tracking, commits, comments, and merges can be found at our source code repository.

<https://gitlab.com/dlmarrero/gradify/activity>

## Conclusions

The following conclusions have been made at the end of this project.

### Design Strengths

The Gradify solution is strong in the following areas:

#### Portable

The solution is easily deployable on any platform that supports python 3.

#### Extendable

Django is easily customizable and can be extended to accommodate future needs easily.

## Secure

Leveraging Google's OAuth and Django's user management systems made developing a secure application easy

## Limitations

The Gradify solution is lacking in a few key areas:

### Not all Original Requirements Met

The original Gradify requirements were too ambitious for a 3-week development project.

- Missed role-based logins for students and guardians
- Encrypting Personal Identifiable Information and the SQLite DB was not accomplished
- Student and Guardian roles were not included in this release. Initial release of Gradify supported teacher roles only. Student-specific and Guardian-specific roles proved to be too challenging to include in this initial release.
- There is no way for a teacher to add or remove students from the application itself. Students will either need to be created from the import or manually added from the admin site.

### Multiple Accounts

Teachers can use more than one account to sign-on, but each account will be independent of their other accounts.

- A google account import cannot be automatically merged to a different google account sign in.
- If they had previously registered as a Gradify user, then they would either need to manually copy over their information from Gradify user account to their Google sign-in account.

### Google API Needs to be enabled

A teacher's google classroom will need to have the Google Classroom API enabled on their account in order for the import to work. Some organizations may limit this ability and prevent teachers from enabling that feature as well. An error will result informing the teacher that they need to have the Google Classroom API enabled in order to use that import feature in the event the API permission is not enabled.

## Future Improvements

The following issues have been identified for future improvements.

### Add Student and Teacher Roles

Gradify was intended to be used by all classroom members: students, teachers, and guardians. Due to time limitations and privacy concerns, we were not able to implement the student and guardian roles. These users should be able to create their own accounts to view their associated information from the application.

### **Redo URLs to Contain Slugs**

The Gradify URLs are not fully secure. They utilize primary keys in the addresses instead of a slug string as we intended. Future versions of the application should include slugs to contain course and assignment titles rather than integer values.

### **Add Information Page**

An “About Gradify” page would be useful to explain the application to potential customers. There should be sections dedicated to students, teachers, and guardians and screenshots of the main features, in particular the grade book view. Creating an online version of the User Guide would also be helpful.

### **Remaining GitLab Issues**

The following 13 GitLab issues were also not addressed during this development project. Further development addressing these items will make this application more complete and feature rich.

Delete Student #58 · opened 5 days ago by Ryan Brady enhancement feature teacher	0 updated just now
Create Students #57 · opened 5 days ago by Ryan Brady enhancement feature teacher	0 updated 1 minute ago
New Pending-Please Wait page displayed during import #55 · opened 1 week ago by Ken Lovingood frontend suggestion	0 updated just now
Use slugs to create URL routes #40 · opened 2 weeks ago by Davy L. Marrero Needs Investigation backend enhancement	0 updated 1 minute ago
Add functionality for inviting students and guardians - User Story 9 #26 · opened 3 weeks ago by Davy L. Marrero backend frontend	0 updated 3 weeks ago
Provide an assignment overview for Students/Parents - User Story 12 #17 · opened 1 month ago by Davy L. Marrero backend feature frontend parent student	0 updated 3 weeks ago
E-mail students and parents when removed from a course - User Story 10 #15 · opened 1 month ago by Davy L. Marrero backend feature teacher	0 updated 3 weeks ago
Add functionality for adding/removing a student from a course - User Story 10 #14 · opened 1 month ago by Davy L. Marrero backend feature frontend teacher	1 updated 5 days ago
E-mail students and parents when added to a course - User Story 9 #13 · opened 1 month ago by Davy L. Marrero backend feature teacher	0 updated 3 weeks ago
Add a form for assigning a student to a course - User Story 9 #12 · opened 1 month ago by Davy L. Marrero backend feature frontend teacher	0 updated 5 days ago
Delete multiple courses at once - User Story 8 #11 · opened 1 month ago by Davy L. Marrero backend feature frontend sprint3 teacher	0 updated 1 week ago
Grades Overview for Parents - User Story 6 #6 · opened 1 month ago by Davy L. Marrero feature frontend	1 updated 3 weeks ago
Grades Overview for Students - User Story 6 #5 · opened 1 month ago by Davy L. Marrero feature frontend	0 updated 3 weeks ago

## Lessons Learned

The following lessons have been learned during the scope of this project.

### Gitlab Issue & Merge Management

Gitlab proved to be an invaluable resource. Each unit of work was tracked as an issue where each developer would assign themselves and issue and perform the necessary development work in their own branch. Once development was completed, the issue would then be raised as a Merge Request. Merges would then get peer reviewed and approved to be included in the



master version. This made it very easy for multiple developers to work on a application at the same time.

### **Merge Conflicts and Resolution**

Sometimes a merge conflict would be raised where one development branch would try to update master to something that the master version had a later version of. This would often require the developer to review the conflict and manually adjust their branch prior to merging again.

### **Pair Programming**

Sometimes an issue is either too large from one developer to complete on their own or sometimes an issue can only be tested by a different user / developer. In those situations, both developers work together at the same time to resolve this issue. This type of team work can resolve issues more efficiently and cleanly.

### **Pipeline Management and Automated Test Cases**

Leveraging Gitlab's automated pipelines to leverage the execution of test cases grew to be a very important tool and asset for this development project. Each time a project is pushed to Gitlab, or the run\_pipeline.sh script that can be run locally. These tests are performed to make sure that the code is clean, does not break any existing functionality, and conforms to best practices.

### **Team Communication & Collaboration**

The success of this development project was in large part a result on how well the group communication. Agile development projects can only be successful when all team members can communicate with each other in an easy and simple manner. The team used Slack for daily communications and had it configured with each tool used (GitLab, Workast, etc.). If there was an issue or question by one team member, then the other team members were alerted and can respond and help. The team also used Zoom for 1+ hr. video conferencing twice each week; Mondays for sprint planning & issue ranking and Fridays for sprint review, demos, and any other major topics that needed to be discussed amongst the team. These tools provided the means for the teams to fully collaborate with each other successfully.

## References

Apache License, Version 2.0. (2004, January). Retrieved May 27, 2019, from <https://opensource.org/licenses/Apache-2.0>

Boehm, B. and Basili, V. (2001). Software defect reduction top 10 list. *Computer*, 34(1), pp.135-137. doi: 10.1109/2.962984

Crockford, D. (n.d.). Introducing JSON. Retrieved June 16, 2019, from <http://json.org/>

Defrex. (2017, January 26). Defrex/django-encrypted-fields. Retrieved from <https://github.com/defrex/django-encrypted-fields>

Django. (n.d.). Documentation. Retrieved June 9, 2019, from <https://docs.djangoproject.com/en/2.2/topics/testing/overview/>

Django. (n.d.). Retrieved May 27, 2019, from <https://www.djangoproject.com/>

The Django Book. (2019). Django's Structure – A Heretic's Eye View - Python Django Tutorials. Retrieved June 16, 2019, from <https://djangobook.com/mdj2-django-structure/>

The Django Project. (n.d.). Documentation. Retrieved June 16, 2019, from <https://docs.djangoproject.com/en/2.2/topics/security/>

FCC. (2015, November 3). Children's Internet Protection Act (CIPA). Retrieved June 1, 2019, from [https://www.fcc.gov/sites/default/files/childrens\\_internet\\_protection\\_act\\_cipa.pdf](https://www.fcc.gov/sites/default/files/childrens_internet_protection_act_cipa.pdf)

Fielding, R. T. (2000). CHAPTER 5. Retrieved from [https://www.ics.uci.edu/~fielding/pubs/dissertation/rest\\_arch\\_style.htm](https://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm)



Getting started with GitLab CI/CD. (n.d.). Retrieved June 7, 2019, from [https://docs.gitlab.com/ee/ci/quick\\_start/](https://docs.gitlab.com/ee/ci/quick_start/)

GitLab. (n.d.). GitLab Plugin system. Retrieved June 9, 2019, from <https://docs.gitlab.com/ee/administration/plugins.html>

Google Inc. (n.d.). Overview | Classroom API | Google Developers. Retrieved June 16, 2019, from <https://developers.google.com/classroom/guides/get-started>

Google Inc. (n.d.). Authorizing Requests | Classroom API | Google Developers. Retrieved June 9, 2019, from <https://developers.google.com/classroom/guides/auth>

Issues. (n.d.). Retrieved June 8, 2019, from <https://docs.gitlab.com/ee/user/project/issues/>

Larson, E. & Larson, R. (2004). Use cases: what every project manager should know. Paper presented at PMI® Global Congress 2004—North America, Anaheim, CA. Newtown Square, PA: Project Management Institute.

MySQL. (n.d.). Retrieved May 27, 2019, from <https://www.mysql.com/>

Richer, J. (n.d.). End User Authentication with OAuth 2.0. Retrieved June 16, 2019, from <https://oauth.net/articles/authentication/>

Schwaber, K., & Sutherland, J. (2017, November). *The Scrum Guide*. Retrieved from <https://www.scrumguides.org>:  
<https://www.scrumguides.org/docs/scrumguide/v2017/2017-Scrum-Guide-US.pdf>

SQLite. (n.d.). Retrieved May 27, 2019, from <https://sqlite.org/index.html>

Welcome to Python.org. (n.d.). Retrieved May 27, 2019, from <https://www.python.org/about/>