**A shield with a book and a sun

AI-generated content may be incorrect.**

Project Title: eLearning Application

Name: Jeremy Loh Wei Bin

Student ID: 230367120

Contents

[1. Executive Summary 6](#_Toc207311563)

[2. Introduction 6](#_Toc207311564)

[2.1 Problem Context & Goals 6](#_Toc207311565)

[2.2 Scope of the Application 6](#_Toc207311566)

[2.3 Success Criteria 7](#_Toc207311567)

[2.31 Usability 7](#_Toc207311568)

[2.32 Reliability 7](#_Toc207311569)

[2.33 Extensibility 7](#_Toc207311570)

[3. System Overview 7](#_Toc207311571)

[3.1 Architecture 7](#_Toc207311572)

[3.11 Web Tier 7](#_Toc207311573)

[3.12 Data Tier 7](#_Toc207311574)

[3.13 Async/ Realtime 8](#_Toc207311575)

[3.14 Static Media 8](#_Toc207311576)

[3.15 Request and Response Lifecycle 8](#_Toc207311577)

[3.16 Security 8](#_Toc207311578)

[3.2 App Modules & Responsibilities 8](#_Toc207311579)

[3.21 Accounts 8](#_Toc207311580)

[3.23 Courses 9](#_Toc207311581)

[3.33 Rtchat 9](#_Toc207311582)

[3.3 Data Model 9](#_Toc207311583)

[3.4 Key Cross-Cutting Patterns 9](#_Toc207311584)

[3.41 Mixins 9](#_Toc207311585)

[3.42 Signals for event flow 9](#_Toc207311586)

[3.43 URL routing 9](#_Toc207311587)

[3.44 Form classes for validation 9](#_Toc207311588)

[3.45 Persistence and Idempotency 10](#_Toc207311589)

[4. Requirements Traceability 10](#_Toc207311590)

[4.1 Authentication and Roles 10](#_Toc207311591)

[4.2 Course Materials 10](#_Toc207311592)

[4.3 Enrollment and Feedback 10](#_Toc207311593)

[4.4 Real Time Chat 10](#_Toc207311594)

[4.5 Notification 10](#_Toc207311595)

[5. Design Rationale 11](#_Toc207311596)

[5.1 Module Split 11](#_Toc207311597)

[5.2 Data Modeling Choices 11](#_Toc207311598)

[5.3 Authorization Strategy 11](#_Toc207311599)

[5.4 Realtime via channels 12](#_Toc207311600)

[5.5 Signals for Decoupling 12](#_Toc207311601)

[5.6 Templates and UX 12](#_Toc207311602)

[6. Implementation of Highlights and Important Decisions 13](#_Toc207311603)

[6.1 Core Flows 13](#_Toc207311604)

[6.11 Create Course 13](#_Toc207311605)

[6.12 Enroll (Student) 13](#_Toc207311606)

[6.13 Upload Materials 13](#_Toc207311607)

[6.14 Real Time Chat 13](#_Toc207311608)

[6.2 Robustness Consideration 14](#_Toc207311609)

[6.21 Duplicate enrolment protection 14](#_Toc207311610)

[6.22 Unique Feedback Constraint 14](#_Toc207311611)

[6.23 Validation of Materials 14](#_Toc207311612)

[6.24 404 or 403 Guard 14](#_Toc207311613)

[6.25 Transactional Integrity 14](#_Toc207311614)

[6.26 Idempotent Notification 14](#_Toc207311615)

[6.27 Realtime Resilience 14](#_Toc207311616)

[6.28 Test 14](#_Toc207311617)

[6.3 Known Gaps and Bug fixes 14](#_Toc207311618)

[6.31 Template path mismatch 14](#_Toc207311619)

[6.32 Chat UI Enrolment notification 15](#_Toc207311620)

[6.33 Channel layer for production 15](#_Toc207311621)

[6.4 Security and Privacy 15](#_Toc207311622)

[6.41 CSRF & Sessions 15](#_Toc207311623)

[6.42 Authorization 15](#_Toc207311624)

[6.43 File Handling 15](#_Toc207311625)

[7. Evaluation and Critical Reflection 15](#_Toc207311626)

[7.1 What Worked Well 15](#_Toc207311627)

[7.11 Separation of concerns 15](#_Toc207311628)

[7.12 Signal as a lightweight event bus 15](#_Toc207311629)

[7.13 Role Based UX 15](#_Toc207311630)

[7.14 Data Integrity Hardening 16](#_Toc207311631)

[7.2 What Could Be Better 16](#_Toc207311632)

[7.21 Production hardening 16](#_Toc207311633)

[7.22 Client Experience in Chat 16](#_Toc207311634)

[7.23 Observability and Operations 16](#_Toc207311635)

[7.24 Teacher Ergonomics 16](#_Toc207311636)

[7.25 Security Posture 16](#_Toc207311637)

[7.26 Performance 17](#_Toc207311638)

[7.26 Accessibility 17](#_Toc207311639)

[7.27 Testing Depth 17](#_Toc207311640)

[7.3 If Rebuilding 17](#_Toc207311641)

[7.31 API first then UI 17](#_Toc207311642)

[7.32 Signals to Domain Events 17](#_Toc207311643)

[7.33 Strong Production Defaults 17](#_Toc207311644)

[7.34 Search and Discovery 17](#_Toc207311645)

[7.35 Moderation Model 17](#_Toc207311646)

[8. How to Run the Application 18](#_Toc207311647)

[8.1 Unzip to a working folder 18](#_Toc207311648)

[8.12 Open a terminal in that folder where manage.py 18](#_Toc207311649)

[8.13 Create an active virtual environment 18](#_Toc207311650)

[8.14 Install Dependencies 18](#_Toc207311651)

[8.15 Database Setup 18](#_Toc207311652)

[8.16 Create an admin 18](#_Toc207311653)

[8.17 Run the server 18](#_Toc207311654)

[8.18 Open http://127.0.0.1:8000 to verify 18](#_Toc207311655)

[8.19 Realtime 18](#_Toc207311656)

[8.2 Media and Static Files 18](#_Toc207311657)

[8.21 Smoke Test 18](#_Toc207311658)

[9. Credentials 19](#_Toc207311659)

[10. Seeding Database 19](#_Toc207311660)

[11. Unit Testing 20](#_Toc207311661)

[12. Package List 20](#_Toc207311662)

[13. User Guide 20](#_Toc207311663)

[13.1 Teacher Workflow 20](#_Toc207311664)

[13.2 Student Workflow 22](#_Toc207311665)

[14. Usability Testing 23](#_Toc207311666)

[14.1 Objective: 23](#_Toc207311667)

[14.2 Participant: 23](#_Toc207311668)

[14.3 Task in order: 23](#_Toc207311669)

[14.4 Results: 23](#_Toc207311670)

[14.5 Observed Friction: 23](#_Toc207311671)

[14.6 Changes Implemented: 23](#_Toc207311672)

[14.7 Changes not implemented yet: 23](#_Toc207311673)

[14.8 Accessibility Check: 24](#_Toc207311674)

[14.9 Risk and limitations: 24](#_Toc207311675)

[15. Conclusion 25](#_Toc207311676)

# 1. Executive Summary

My Django application is an E-learning platform that enables teachers to run online classes and allow students to participate with minimal friction. It provides small cohorts with real time discussion and materials sharing. Core features include account creation and login, role-based access for teachers and students, teacher-led course creation and roster management, student enrollment, course feedback with ratings, upload of teaching materials visible on each course home page, real time chat with history and notifications to keep teachers and students informed.

# 2. Introduction

### 2.1 Problem Context & Goals

* + Higher education cohorts increasingly rely on digital platforms but still stitch ad-hoc tools as environmental forms, file sharing and chat. This fragmentation creates friction where teachers duplicate effort setting up courses across systems, students missing updates on materials and real time communication offer no integration with enrolment or moderation.
  + This application aims to address this pain points by unifying workflows needed to run a course. For teachers, the goal is to make course creation, material distribution, roaster management and moderation safe while automatically notifying students about important events. For students, the goal is self-discovering courses, access all materials from a single course in one page, contribute to course feedback as well as communicating in real time with peers and teachers.

### 2.2 Scope of the Application

* + User authentication (sign up, login/log out)
  + Role-Based access for teachers and students
  + Course management
  + Feedback
  + Teaching materials
  + Real time chat
  + Notifications to alert students when new materials are added
  + Status Post on user homepage

### 2.3 Success Criteria

### 2.31 Usability

* + - Students can discover and self-enroll in a course from the course list
    - Teachers can publish a new material to a course
    - Feedback Validates duplicates and surface current average rating

### 2.32 Reliability

* + - All chat messages are shown and reloaded with accurate timestamps
    - Enrolment creation has no duplicates
    - Broadcasts are fired exactly once per event
    - File uploads are stored under safe paths

### 2.33 Extensibility

* + - The codebase is split into focused Django Apps such as accounts, courses and rtchat
    - Mixins centralize authorization and reduce duplication
    - Signals decouple event producers

# 3. System Overview

## 3.1 Architecture

### 3.11 Web Tier

* + - Django handles HTTP request through URL routing
    - Django channels upgrades selected HTTP connections to web sockets via ASGI

### 3.12 Data Tier

* + - A relational database SQLite persists domain entities via Django ORM
    - Migrations capture schema evolution

### 3.13 Async/ Realtime

* + - In-memory channel layer are provided by channels

### 3.14 Static Media

* + - Django staticfiles servers CSS/JS during development
    - User uploaded materials are stored under MEDIA\_ROOT and addressed via MEDIA\_URL

### 3.15 Request and Response Lifecycle

* + - For HTTP interactions, the browser issues a request and Django resolves the URL to a view
    - The view queries or updates the model and a server rendered template is returned
    - For WebSocket, the browser opens a socket to the course room and the channels consumer authenticates the user and adds them to a group named for the course with inbound messages as well as broadcasts payloads to the group with connection rendered instantly

### 3.16 Security

* + - CSFR protection is enable for forms and authentication relies on Django session middleware
    - Authorization uses role checks in mixins and per object permission gates
    - File uploads are validated and stored with safe filenames under segregated paths
    - Spoofing are mitigated by Django auth and HTTS
    - CSFR token prevents tampering on all POST forms
    - Server-side validation in forms and modes also prevents tempering
    - Admin access limited to superusers with strong passwords and authorized personnel

## 3.2 App Modules & Responsibilities

### 3.21 Accounts

* + - User onboarding and profile features
    - User Login and logout
    - User home with status post
    - Views protected by login requirement

### 3.23 Courses

* + - Teachers create courses, manage rosters, upload materials and students self-enroll and leave feedback

### 3.33 Rtchat

* + - Real time collaboration with channels consumer per course room, routing configuration and chat templates

## 3.3 Data Model

* + User has many status posts and course memberships. Group membership drives authorization decisions
  + Course aggregates course materials, collect course feedback and relate to users via enrollment
  + CourseBlock records moderation decision
  + CourseFeedback enforces one review per user and supports average rating display
  + CourseMaterial stores metadata and file paths and files live under MEDIA\_ROOT
  + ChatMessage stores message text, sender, course room and timestamps
  + Indexing focuses on common lookups

## 3.4 Key Cross-Cutting Patterns

### 3.41 Mixins

* + - Encapsulate role checks so views remain declarative which prevents scattered permission logic

### 3.42 Signals for event flow

* + - Hooks such as post\_save and CourseMaterial publish events without coupling controllers to side effects

### 3.43 URL routing

* + - Each app owns its URLconf and template namespace

### 3.44 Form classes for validation

* + - Dedicated forms centralize validation rules

### 3.45 Persistence and Idempotency

* + - Transaction boundaries in enrolment ensure that retries do not create duplicate rows

# 4. Requirements Traceability

## 4.1 Authentication and Roles

* + Users self-register via a sign-up page backed by Django UserCreationForm and authenticate it through Django built in login
  + Logout uses the auth view and on sign up users are added to the student group by default and teacher accounts are provisioned by an admin
  + Status updates appear on each user’s homepage
  + Strong password validation via Django settings with duplicate username rejected
  + All protected views are decorated with login\_required

## 4.2 Course Materials

* + Teacher can create courses
  + Teacher can upload course material items
  + Materials render on the course homepage for teachers and students

## 4.3 Enrollment and Feedback

* + Students enroll themselves into one or more courses using a multi select form
  + The application prevents duplicate enrolments
  + Students can submit one course feedback per user per course
  + Course Feedback aggregate ratings render on the course page

## 4.4 Real Time Chat

* + A per course chat room using Django channels
  + Only teachers and students enrolled can chat
  + Messages are broadcasted as rows to all users
  + Authorization checks on WebSocket and graceful handling of disconnect

## 4.5 Notification

* + Event driven notification via Django signals
  + When an enrollment is created, a post save signal broadcast an enrollment event for the course
  + When a course material is created, a signal broadcast a material event
  + Signals are idempotent

# 5. Design Rationale

## 5.1 Module Split

* + The codebase is intentionally split into three focused Django apps to respect the single responsibility principle and reduce cognitive load
  + Authentication profiles and user centric pages is in accounts
  + Classroom domain logic is in courses
  + Realtime collaboration is in rtchat
  + The split clarifies ownership and makes it possible to iterate on one area without unintentionally destabilizing another
  + This separation also enables independent testing
  + Each app exposes a surface for unit testing

## 5.2 Data Modeling Choices

* + Enrollment as a through model instead of a bare many to many between user and course, this design introduces an explicit enrollment model
  + This model provides uniqueness and a natural anchor for signals
  + CourseBlock as a first-class model preserves an audit trail prevents confusing state transitions on enrolment rows and allows blocks to exist even when a student has never successfully enrolled.
  + CourseMaterial as file or URL and teachers may upload a file or link to external content
  + Modeling a single course material with optional file field and URL achieves flexibility with minimal schema complexity
  + Files are stored under MEDIA\_ROOT using safe names
  + CourseFeedback enforces one review per student per course through a uniqueness constraint enabling aggregation of ratings for display
  + Indexing with foreign keys such as on\_delete=CASCADE where cascade semantics are intuitive
  + Composite indexes speed up membership checks

## 5.3 Authorization Strategy

* + The application adopts a role-based access control implemented with Django
  + TeacherRequiredMixin and StudentRequiredMixin centralize group checks with LoginRequiredMixin and every protected view declares its intent and failure
  + This avoid duplicating checks across views and reduce risk of drifting permission over time
  + Per object checks are layered on top
  + URL design reinforces the policy of teacher actions live under routes bound to course IDs of their own
  + Templates hide off limit controlls so users will not hit a hard denial in normal use when a 403 forbidden error occurs
  + All mutating requests are protected by CSRF tokens

## 5.4 Realtime via channels

* + Real time chat notifications are over long polling or external chat services
  + Each client connects to a course named group
  + Consumer validates roles on connect and persist any inbound messages
  + Channels with an in-memory channel layer is sufficient for single process development

## 5.5 Signals for Decoupling

* + The application uses Django signals as a simple event bus
  + Creating an enrollment emits an enrollment event
  + Saving a CourseMaterial emits a material event
  + To avoid duplicates sends on multi save flows, handlers check created flags and strive for idempotency

## 5.6 Templates and UX

* + The UI is deliberately server rendered using Django templates
  + Templates are name spaced per app giving consistent layout and form styling
  + Role based navigation ensures users only see actions they can perform
  + Forms provide inline validation messages that align with Django form error structures
  + Course home operates as the single pane of glass and materials appear in a canonical list
  + A link opens real time chat
  + People search used by teachers are kept minimal and secure

# 6. Implementation of Highlights and Important Decisions

## 6.1 Core Flows

### 6.11 Create Course

* + - Teacher navigates to /courses/create/ and the CourseCreateView renders a model form bound to the course model
    - On post, server-side validation runs and the view sets form.instance.created\_by = request.user to establish ownership
    - After the course row is inserted, the user is redirected to the course home where empty material list and roster action are visible

### 6.12 Enroll (Student)

* + - Student opens /courses/enroll/ and selects one or more courses in a multi-select widget backed by MultiEnrollForm
    - Form clean computes three sets, already enrolled, blocked via CourseBlock and new enrolments to create
    - Inside transaction.atomic, the view performs a bulk create of enrollment rows
    - A post save signal fires for each created enrollment enqueuing a teacher notification

### 6.13 Upload Materials

* + - Owner visits /courses/materials/new and CourseMaterialCreateView enforces that the current user is the course owner
    - Form requires at least one file or URL
    - Files are stored and saved under MEDIA\_ROOT
    - A post save receiver then emits a material event to the course WebSocket and chat recipients receive a system message summarizing the upload

### 6.14 Real Time Chat

* + - User opens the chat page and the page initializes a WebSocket
    - The channel consumer connect() authenticates the session and adds the channel to a course named group
    - On receive\_json, the consumer persists a ChatMessage and broadcasts a compact payload to the group via group\_send
    - On reload the latest message are fetched to rebuild history

## 6.2 Robustness Consideration

### 6.21 Duplicate enrolment protection

* + - Enforced both in application logic and at the database layer via a uniqueness constraint
    - This dual guard ensures correctness under concurrent requests

### 6.22 Unique Feedback Constraint

* + - CourseFeedback uses a unique constraint so a student can leave exactly one review

### 6.23 Validation of Materials

* + - The form rejects submission where neither file nor URL is provided

### 6.24 404 or 403 Guard

* + - Object fetches use ge\_object\_or\_404
    - Role checks in mixins return 403 when a legitimate user lack rights
    - Templates hide off limits controls to reduce error frequency

### 6.25 Transactional Integrity

* + - Multi course enrolment writes occurs within transaction.atomic()
    - Signals may be deferred with transaction.on\_commit

### 6.26 Idempotent Notification

* + - Signal handlers branch on created is true to avoid duplicate sends

### 6.27 Realtime Resilience

* + - On reconnect the client fetch persisted messages so transient WebSocket drops do not lose context

### 6.28 Test

* + - Unit test for forms
    - An Integration test that assets signal broadcasts on enrolment or material creation

## 6.3 Known Gaps and Bug fixes

### 6.31 Template path mismatch

* + - The teacher people-search view rendered “accounts/people\_search.html” while it is at “templates/people\_search.html”
    - The correct render path has been rendered since

### 6.32 Chat UI Enrolment notification

* + - Signals broadcast an enrolled event to surface teacher notification in real time

### 6.33 Channel layer for production

* + - Uses in-memory layer

## 6.4 Security and Privacy

### 6.41 CSRF & Sessions

* + - All post forms include CSRF tokens
    - Authentication relies on Django sessions with secure cookie flags

### 6.42 Authorization

* + - Mutating routes are protected by role mixin and per object checks
    - Block and unblock ensures a blocked student cannot re-enrol or join chats

### 6.43 File Handling

* + - Uploads are stored under segregated MEDIA\_ROOT paths

# 7. Evaluation and Critical Reflection

## 7.1 What Worked Well

### 7.11 Separation of concerns

* + - Splitting the codebase into accounts, courses and rtchat delivered exactly what was expected, low coupling, clear ownership and faster iteration
    - Each app has its own model so bug hunting and feature stay localized

### 7.12 Signal as a lightweight event bus

* + - Using post\_save on enrollment and courseMaterial on notification proved effective
    - It kept view thin, made side-effects discoverable and add user visible outcome without entangling core controllers

### 7.13 Role Based UX

* + - Mixin based authorization strategy helped teachers see creation, roster and moderation controls
    - Students see enrollment, materials and feedback
    - Hiding inapplicable actions reduced 403s

### 7.14 Data Integrity Hardening

* + - Duplicate enrollment prevention via both application logic
    - Enforcing a single CourseFeedback per student prevented duplicates and simplified rating aggregation

## 7.2 What Could Be Better

### 7.21 Production hardening

* + - * Redis channel layer for multi process fan out and reliability
      * SMTP with retry and bounce handling
      * Object storage for media
      * Optionally signed URLs for access control

### 7.22 Client Experience in Chat

* + - Presence and typing indicators
    - Delivery states
    - Inline attachments
    - Better error surfaces for reconnect or timeouts
    - Offer user preferences for text size

### 7.23 Observability and Operations

* + - Structed logs (JSON)
    - Request WebSocket correlation IDs
    - Error capture across HTTP and channels

### 7.24 Teacher Ergonomics

* + - A small dashboard summarizing enrollment, feedback average, recent materials and open chat activity
    - Roster filters

### 7.25 Security Posture

* + - Rate Limits recommended on chat sends and enrollment attempts to prevent denial of service
    - File Scanning
    - Hardened cookies
    - 2FA for admin account
    - Configurable retention policy to remove chat more than a set amount of days
    - Backups of database and media can be done and encrypted at rest

### 7.26 Performance

* + - Introduce per course caching with invalidation on post\_save signals
    - Initial profiling shows database time dominating on course pages with large rosters and long material lists
    - Chat to persist then broadcast to avoid message loss
    - Background job to compact chat logs older than a retention window

### 7.26 Accessibility

* + - Templates should be checked for keyboard navigation
    - Ensure minimum 4.5:1 contrast ratio
    - Add focus outlines on interactive elements

### 7.27 Testing Depth

* + - Unit test cover forms and signals but lack comprehensive consumer tests
    - Add channel’s WebsocketCommunicator tests for flows, permission denials and message persistence

## 7.3 If Rebuilding

### 7.31 API first then UI

* + - * Django REST Framework would decouple domain logic from templates

### 7.32 Signals to Domain Events

* + - * Small domain events layer would make flows more testable
      * Creates a choke point for observability and retries

### 7.33 Strong Production Defaults

* + - * A prestart script and environment variable validation

### 7.34 Search and Discovery

* + - * As material and chat grows, course discovery benefits from full-text search

### 7.35 Moderation Model

* + - * Soft bans, shadow bans, reason codes and appeal logs

# 8. How to Run the Application

## 8.1 Unzip to a working folder

## 8.12 Open a terminal in that folder where manage.py

## 8.13 Create an active virtual environment

* + - python -m venv .venv

## 8.14 Install Dependencies

* + - python -m pip install --upgrade pip
    - pip install -r requirements.txt

## 8.15 Database Setup

* + - python manage.py migrate

## 8.16 Create an admin

* + - python manage.py createsuperuser

## 8.17 Run the server

* + - python manage.py runserver

## 8.18 Open <http://127.0.0.1:8000> to verify

* + - Sign up
    - Log in
    - Log out
    - Admin works as superuser
    - Static files render and upload go to MEDIA\_ROOT

## 8.19 Realtime

* + - No extra setup for in-memory channel layer for local Development

## 8.2 Media and Static Files

* + - Ensure MEDIA\_ROOT exists
    - Use object storage

## 8.21 Smoke Test

* + - Create a teacher to add a course and upload a material
    - Sign in as student, enroll, post feedback and open chat for that course

# 9. Credentials

|  |  |  |  |
| --- | --- | --- | --- |
| Role | Username | Password | Where to Log In |
| Django Admin | Jeremy | Elearning25 | http://127.0.0.1:8000/admin/ |
| Teacher | Taylor | Pass1234 | http://127.0.0.1:8000 |
| Teacher | Morgan | Pass1234 | http://127.0.0.1:8000 |
| Teacher | Riley | Pass1234 | http://127.0.0.1:8000 |
| Student | Alex | Pass1234 | http://127.0.0.1:8000 |
| Student | Sam | Pass1234 | http://127.0.0.1:8000 |
| Student | Jo | Pass1234 | http://127.0.0.1:8000 |
| Student | Kai | Pass1234 | http://127.0.0.1:8000 |
| Student | Nia | Pass1234 | http://127.0.0.1:8000 |
| Student | Zen | Pass1234 | http://127.0.0.1:8000 |

# 10. Seeding Database

* + python manage.py shell <<'PY'
  + from django.contrib.auth.models import User, Group
  + teacher\_g, \_ = Group.objects.get\_or\_create(name="Teacher")
  + student\_g, \_ = Group.objects.get\_or\_create(name="Student")
  + # Admin
  + if not User.objects.filter(username="admin").exists():
  + User.objects.create\_superuser("admin", "admin@example.com", "DemoAdmin123!")
  + # Teacher
  + t, \_ = User.objects.get\_or\_create(username="teacher1", defaults={"email":"t1@example.com"})
  + t.set\_password("Teacher123!"); t.is\_staff=True; t.save(); t.groups.add(teacher\_g)
  + # Student
  + s, \_ = User.objects.get\_or\_create(username="student1", defaults={"email":"s1@example.com"})
  + s.set\_password("Student123!"); s.save(); s.groups.add(student\_g)
  + print("Seeded: admin, teacher1, student1; groups: Teacher, Student")

# 11. Unit Testing

* + Django Test Runner:
    - python manage.py test -v 2
  + Ensure pytest.ini defines:
    - DJANGO\_SETTINGS\_MODULE = Elearning.settings

# 12. Package List

|  |  |  |
| --- | --- | --- |
| Package | Version | Purpose |
| Django | 5.2 | Web Framework |
| Daphne | 4.2.1 | ASGI Server for Channels |
| Channels | 4.0.0 | WebSocket Support |
| asgiref | 3.8.1 | ASGI Utilities |
| sqlite3 | Built in | Default Database |
| Pillow | 10.2.0 | Image Handling for uploads |

## 12.1 Environment Details

|  |  |
| --- | --- |
| Operating System | MacOS 14 |
| Python | 3.11.7 |
| Node | 18.16.0 |
| Npm | 9.5.1 |

# 13. User Guide

## 13.1 Teacher Workflow

1) Sign in as teacher using a teacher account

* Log in as Teacher

2) Create a course and fill in the necessary details

* Navigate to course and click on “add a course”
* Fill in all the form details and click save

3) Upload teaching materials

* Under the teaching section, click on the link to course homepage for the course
* Click on “Add Materials”
* Fill in the form details and click save
* Students now see the uploaded material and link in the course homepage

4) Share the course

* Copy the course home URL and share with students

5) Manage the roster

* Under teaching section, click on the link to the roster page for the course
* Remove or block students from the course

6) Open real-time chat

* Under teaching section, click on the chat link for the course
* Messages post instantly and chat history persist

7) Notification

* Teachers will receive a notification in the chat when a student enrolls
* Students will receive a notification when the teacher uploads a material

8) Post a status

* Add a brief status update that will be visible to others on the user homepage

9) Troubleshooting

* 403 Forbidden error indicates the user is not the course owner or not in teacher group
* Students who are unable to see materials have to confirm if they are enrolled into the course

## 13.2 Student Workflow

1) Sign up and log in

* Sign Up to create an account
* Log in with the same username and password

2) Enroll in a course

* Go to courses and “Enroll in Courses”
* Checkboxes with available courses will show up
* Select the course you would like to enroll in and click “Enroll”
* If you are blocked from the course, you will encounter a pop up and you will not be added into the course

3) Access Materials

* Visit the course hompage
* Added materials will be displayed under the “Materials” section
* Click the document or link to display the material

4) Leave Feedback

* On your homepage, click on the link “My courses”
* Click on “Leave / Edit feedback”
* A form displaying the feedback will appear
* Fill in the details in the form
* Click Add or update to successfully update or add feedback
* One review per course is enforced

5) Join the chat

* In the Enrolled section, click on the link to the chat
* Past Messages will be visible
* Notification will appear in chat when a new course material is added

6) Status Updates

* The user can post a short status visible to others

7) Troubleshooting

* If you are unable to enroll you might be blocked or have already enrolled from the course
* If you are unable to open chat, you must be enrolled into the course to be able to chat

# 14. Usability Testing

## 14.1 Objective:

* Validate that first time teachers and students can complete core flows without training and measure the time to complete as well as to surface UI issues before submission. This ensures that all usability and bugs issues are rectified before production.

## 14.2 Participant:

* A test with 10 participants (5 Teachers and 5 Students) validated core workflows. They tested by running the usual workflow by creating a course, uploading a file, enrolling into a course, post feedback and sending messages using Realtime chat. All participants are first time users of the application. Sessions were around 30 mins long and in person. All participants are using a 13-inch laptop.

## 14.3 Task in order:

1) Create a course (Teachers)

2) Upload a file to that course (Teachers)

3) Leave feedback (Students)

4) Send a chat message (Teachers and Students)

## 14.4 Results:

* 100% completion for all task with no moderator intervention

## 14.5 Observed Friction:

* Participant expected an “Enroll” button on course cards and course home when not enrolled
* Feedback entry location is not accessible

## 14.6 Changes Implemented:

* Added a more prominent “Enroll in courses” button in course list
* Added a “My courses” shortcut in the homepage to enter my course details row to leave feedback to enrolled course

## 14.7 Changes not implemented yet:

* Materials drag to re-order
* Teacher dashboard card to show details such as enrollment count, latest feedback, most recent rating uploaded)

## 14.8 Accessibility Check:

* Labels and error messages are programmatically associated with inputs
* Low contrast for muted links

## 14.9 Risk and limitations:

* Findings of sample size 10 are directional and not statistically generalizable
* All sessions are fast on local hardware and not tested on mobile devices

# 15. Conclusion

This E-Learning application is set out to deliver a compact and comprehensible e-learning platform that solves user pain points without the full weight of a learning management system. The application includes systems and services such as authentication and role separation, teacher led course and material management, student self-enrollment, course feedback, real time per course chat and event driven notifications.

In terms of architectural, Django’s included stack provides dependable primitives such as auth, ORM, admin and forms and channels layer in real time collaboration without fragmenting the codebase. The modules are split into accounts, courses and rtchat to isolate concerns, keep views thin and make testing targeted. Role based mixins were chosen for authorization and Django signals for notification reduced duplication and decoupling side effects from controllers. The use of Django channels also provides real time collaboration without introducing a separate service as messages had persisted first and then broadcasted giving both reliability and immediacy.

Beyond these core modules, the application is designed for simplicity and classroom fit use. The application is optimized for the shortest path from enrollment to accessing materials, useful notifications, form feedback to reduce cognitive load so first-time users can be productive without training.

In terms of security, CSRF protection, HTTP-only cookies, password hashing and rate limited endpoints mitigate common threats. PDPA/GDPR considerations such as explicit consent on sign up and retention policies for chat history and course artefact are documented.

In terms of scalability, pagination by default, cached course home views and a persist then broadcast pattern so WebSocket fan out happens only after a durable commit. Unit testing covers permissions, enrollment flows and message broadcast.

Limitations are known and form a clear roadmap. Deeper assessment tooling, institutional SSO, calendar integration, richer learning analytics and enhanced file handling.

In sum, the application meets the objectives and validates a lean, teacher centric alternative to an LMS. It establishes a stable and extensible foundation that can be evolved into a production ready platform with measurable impact on teaching efficiency and student engagement.