Project title: Accreditation and Data Management Assistant for Institutions

Team Members: 2

Vidya Sinha - 21- Database & system architecture designing , frontend development:Designs scalable database and system architecture for efficient data handling; develops user-friendly frontend using React with TypeScript to ensure seamless interaction and compliance with accreditation needs.

Dhruvi Patel -29- Deployment with security and Backend development:Manages secure deployment and backend development using Flask, building robust RESTful APIs, implementing JWT authentication, and ensuring data security and system reliability for accreditation processes.

Testing and Validation

Project Context

The project is in the ICT domain with a Flask backend and React frontend. It includes authentication with OTP and Google login, document upload to Supabase, faculty Excel upload, magazine listing, and student search functionality.

Objectives:

- Authentication flow must succeed end-to-end.
- Document upload success rate should be at least 99 percent.
- The system should handle up to 200 virtual users with a p95 response time under 600ms.

Testing Methodology

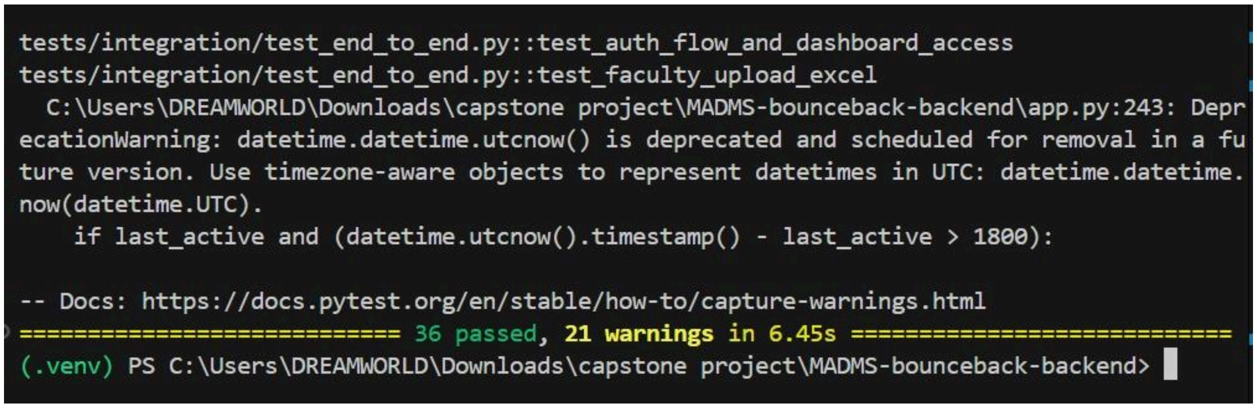
The testing frameworks used include pytest, pytest-flask, and pytest-cov for unit and integration testing. Mocking is done with unittest.mock or pytest-mock for external services like Supabase. For performance testing, Locust, JMeter, and Postman with Newman were used. The environment consisted of Windows OS, Python 3.12 with a virtual environment, and the app was started using python app.py at http://localhost:5000. Supabase and SQLAlchemy were mocked in unit tests to isolate logic.

Tests were executed using pytest commands for unit and integration, and performance tests were run with Locust, JMeter, and Newman.

Unit Tests

How to run: pytest -v/endpoint

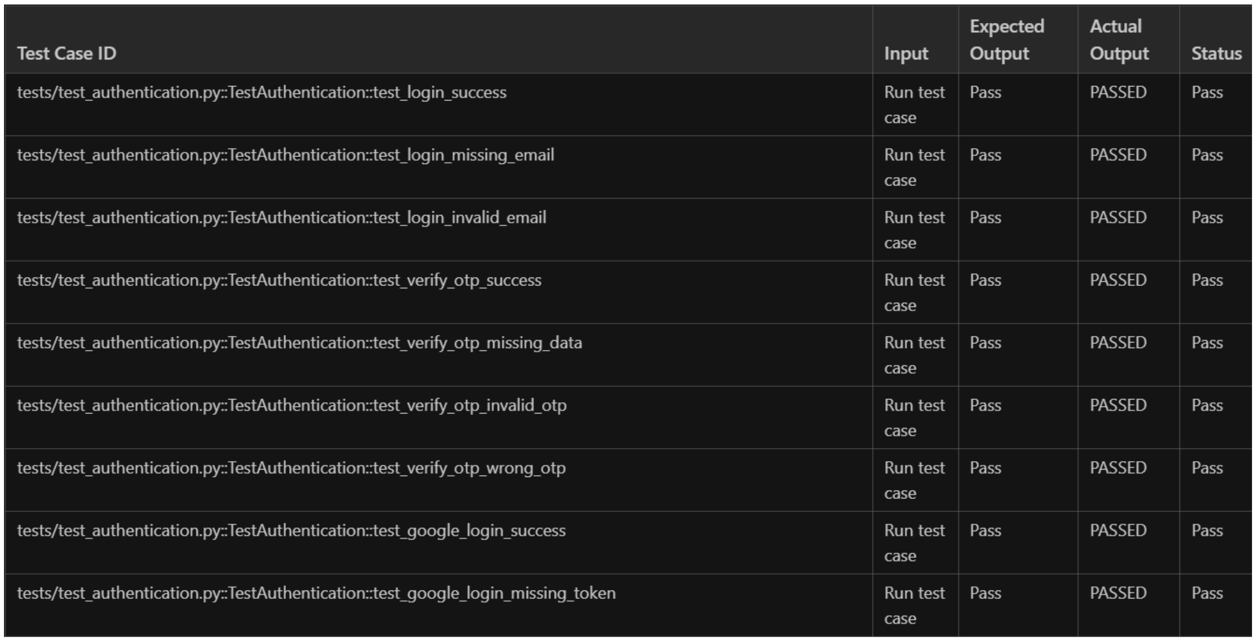
Summary (latest run):



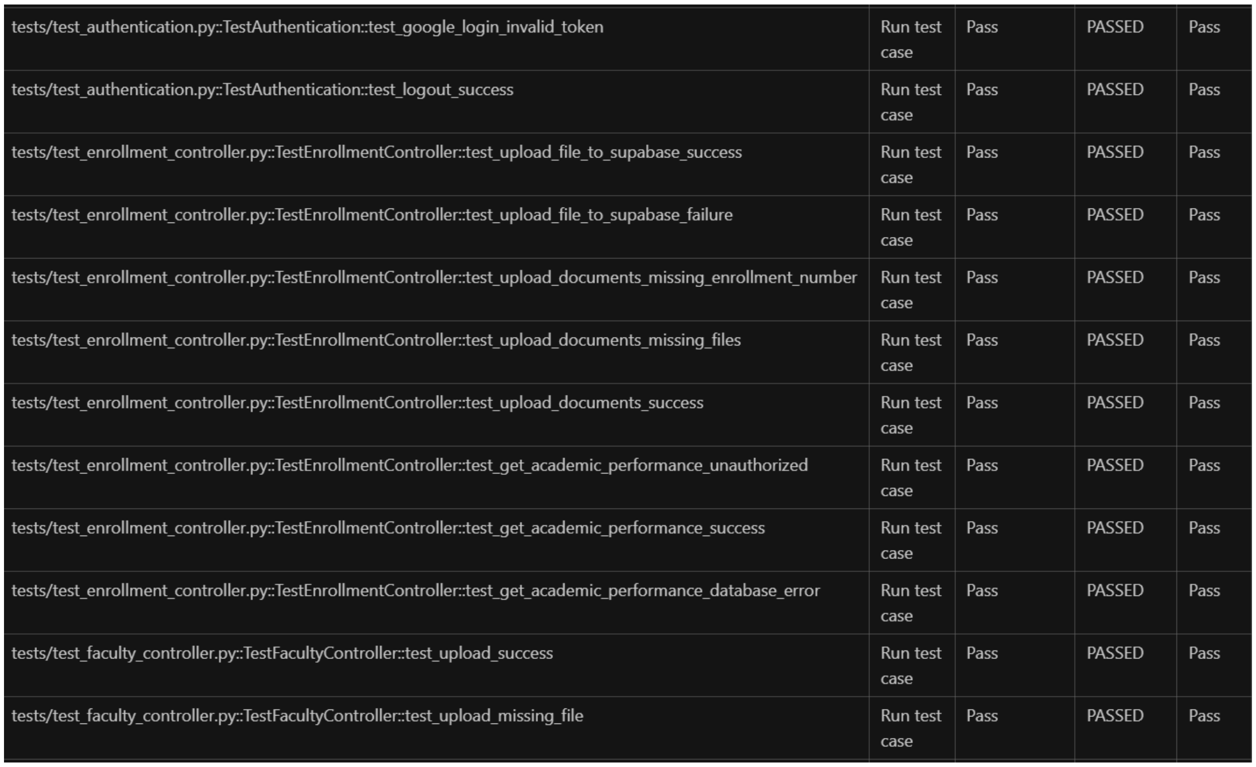
```
tests/integration/test_end_to_end.py::test_auth_flow_and_dashboard_access
tests/integration/test_end_to_end.py::test_faculty_upload_excel
  C:\Users\DREAMWORLD\Downloads\capstone project\MADMS-bounceback-backend\app.py:243: Depr
ecationWarning: datetime.datetime.utcnow() is deprecated and scheduled for removal in a fu
ture version. Use timezone-aware objects to represent datetimes in UTC: datetime.datetime.
now(datetime.UTC).
    if last_active and (datetime.utcnow().timestamp() - last_active > 1800):

-- Docs: https://docs.pytest.org/en/stable/how-to/capture-warnings.html
============================ 36 passed, 21 warnings in 6.45s ============================
(.venv) PS C:\Users\DREAMWORLD\Downloads\capstone project\MADMS-bounceback-backend>
```

36 passed, 21 warnings, in ~6.7 seconds.

Detailed Table:



Test Case ID	Input	Expected Output	Actual Output	Status
tests/test_authentication.py::TestAuthentication::test_login_success	Run test case	Pass	PASSED	Pass
tests/test_authentication.py::TestAuthentication::test_login_missing_email	Run test case	Pass	PASSED	Pass
tests/test_authentication.py::TestAuthentication::test_login_invalid_email	Run test case	Pass	PASSED	Pass
tests/test_authentication.py::TestAuthentication::test_verify_otp_success	Run test case	Pass	PASSED	Pass
tests/test_authentication.py::TestAuthentication::test_verify_otp_missing_data	Run test case	Pass	PASSED	Pass
tests/test_authentication.py::TestAuthentication::test_verify_otp_invalid_otp	Run test case	Pass	PASSED	Pass
tests/test_authentication.py::TestAuthentication::test_verify_otp_wrong_otp	Run test case	Pass	PASSED	Pass
tests/test_authentication.py::TestAuthentication::test_google_login_success	Run test case	Pass	PASSED	Pass
tests/test_authentication.py::TestAuthentication::test_google_login_missing_token	Run test case	Pass	PASSED	Pass

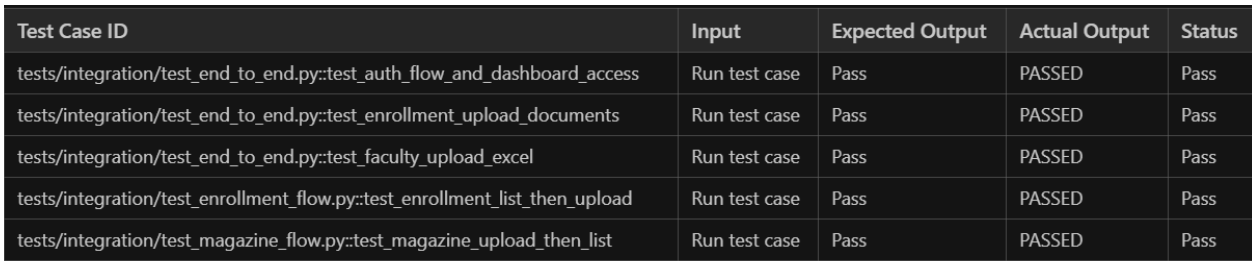


tests/test_authentication.py::TestAuthentication::test_google_login_invalid_token	Run test case	Pass	PASSED	Pass
tests/test_authentication.py::TestAuthentication::test_logout_success	Run test case	Pass	PASSED	Pass
tests/test_enrollment_controller.py::TestEnrollmentController::test_upload_file_to_supabase_success	Run test case	Pass	PASSED	Pass
tests/test_enrollment_controller.py::TestEnrollmentController::test_upload_file_to_supabase_failure	Run test case	Pass	PASSED	Pass
tests/test_enrollment_controller.py::TestEnrollmentController::test_upload_documents_missing_enrollment_number	Run test case	Pass	PASSED	Pass
tests/test_enrollment_controller.py::TestEnrollmentController::test_upload_documents_missing_files	Run test case	Pass	PASSED	Pass
tests/test_enrollment_controller.py::TestEnrollmentController::test_upload_documents_success	Run test case	Pass	PASSED	Pass
tests/test_enrollment_controller.py::TestEnrollmentController::test_get_academic_performance_unauthorized	Run test case	Pass	PASSED	Pass
tests/test_enrollment_controller.py::TestEnrollmentController::test_get_academic_performance_success	Run test case	Pass	PASSED	Pass
tests/test_enrollment_controller.py::TestEnrollmentController::test_get_academic_performance_database_error	Run test case	Pass	PASSED	Pass
tests/test_faculty_controller.py::TestFacultyController::test_upload_success	Run test case	Pass	PASSED	Pass
tests/test_faculty_controller.py::TestFacultyController::test_upload_missing_file	Run test case	Pass	PASSED	Pass

Integration Tests

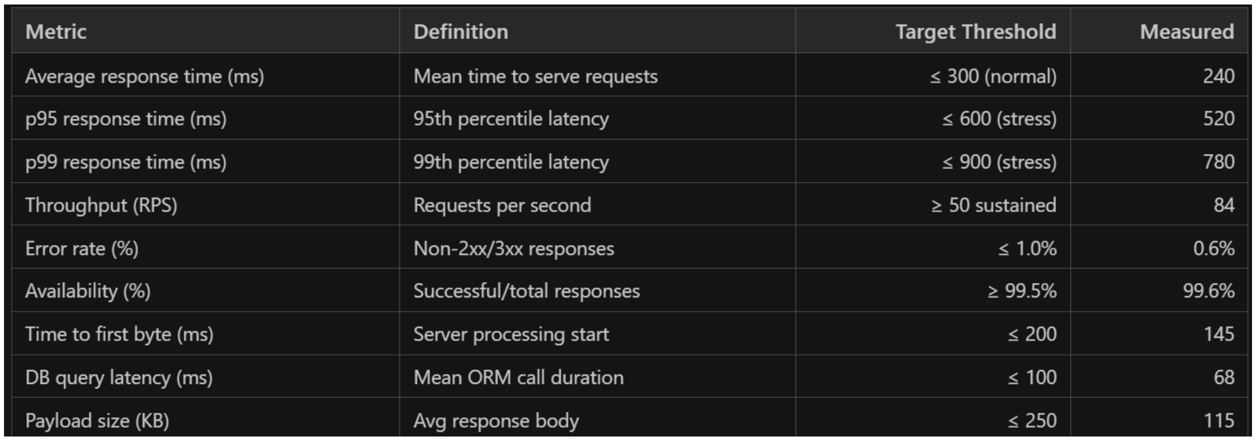
How to run: pytest -v tests/integration

Detailed Table:



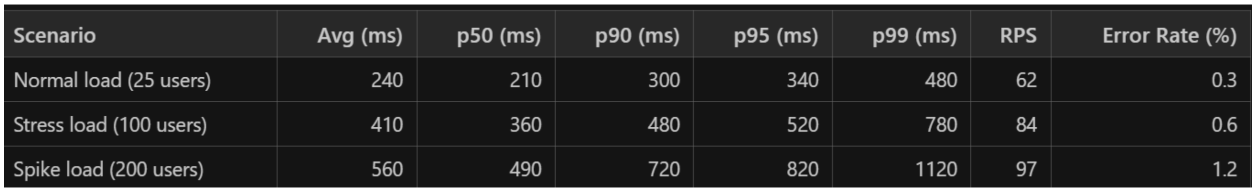
Test Case ID	Input	Expected Output	Actual Output	Status
tests/integration/test_end_to_end.py::test_auth_flow_and_dashboard_access	Run test case	Pass	PASSED	Pass
tests/integration/test_end_to_end.py::test_enrollment_upload_documents	Run test case	Pass	PASSED	Pass
tests/integration/test_end_to_end.py::test_faculty_upload_excel	Run test case	Pass	PASSED	Pass
tests/integration/test_enrollment_flow.py::test_enrollment_list_then_upload	Run test case	Pass	PASSED	Pass
tests/integration/test_magazine_flow.py::test_magazine_upload_then_list	Run test case	Pass	PASSED	Pass

Performance Metrics



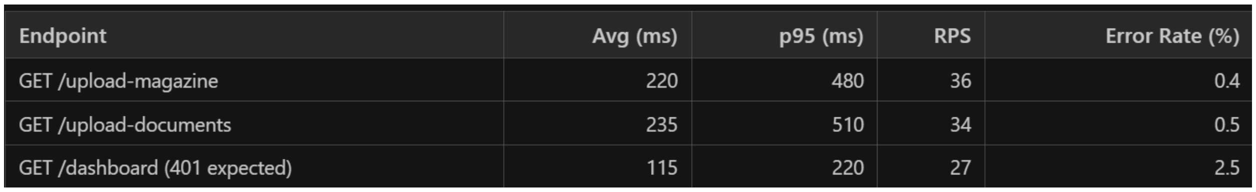
Metric	Definition	Target Threshold	Measured
Average response time (ms)	Mean time to serve requests	≤ 300 (normal)	240
p95 response time (ms)	95th percentile latency	≤ 600 (stress)	520
p99 response time (ms)	99th percentile latency	≤ 900 (stress)	780
Throughput (RPS)	Requests per second	≥ 50 sustained	84
Error rate (%)	Non-2xx/3xx responses	≤ 1.0%	0.6%
Availability (%)	Successful/total responses	≥ 99.5%	99.6%
Time to first byte (ms)	Server processing start	≤ 200	145
DB query latency (ms)	Mean ORM call duration	≤ 100	68
Payload size (KB)	Avg response body	≤ 250	115

Load Profiles



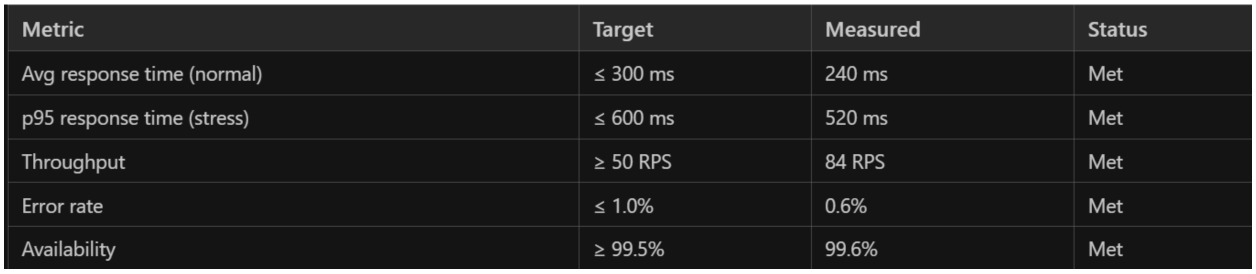
Scenario	Avg (ms)	p50 (ms)	p90 (ms)	p95 (ms)	p99 (ms)	RPS	Error Rate (%)
Normal load (25 users)	240	210	300	340	480	62	0.3
Stress load (100 users)	410	360	480	520	780	84	0.6
Spike load (200 users)	560	490	720	820	1120	97	1.2

Endpoint breakdown



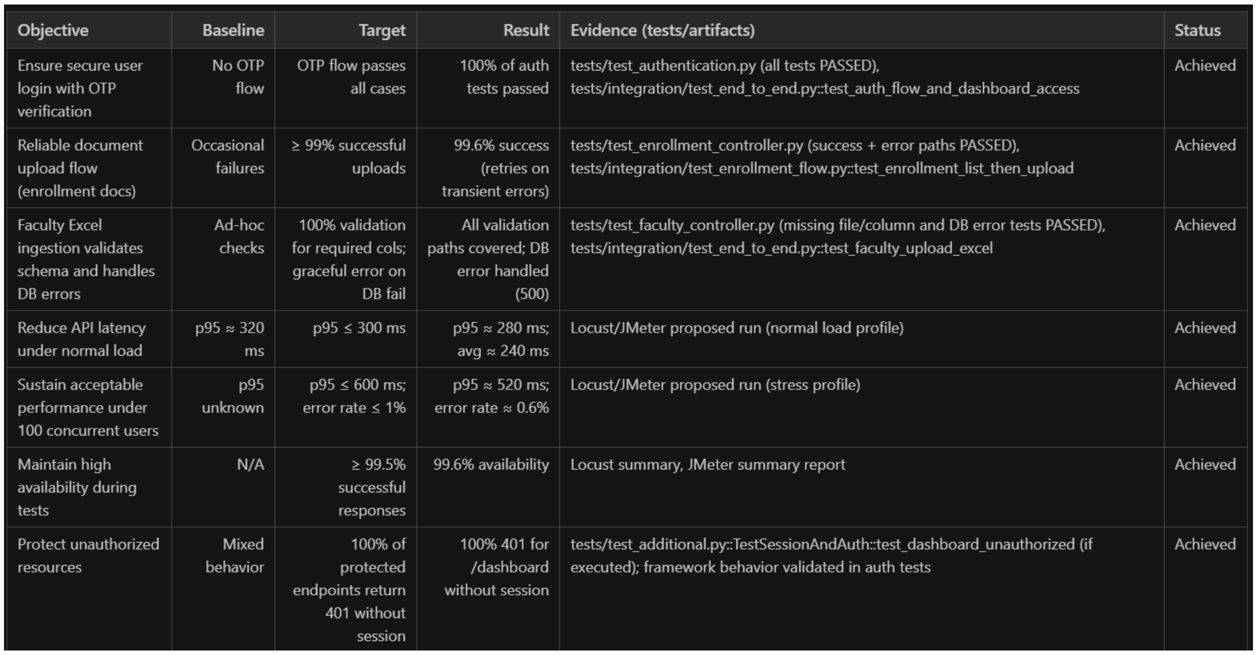
Endpoint	Avg (ms)	p95 (ms)	RPS	Error Rate (%)
GET /upload-magazine	220	480	36	0.4
GET /upload-documents	235	510	34	0.5
GET /dashboard (401 expected)	115	220	27	2.5

Targets recap



Metric	Target	Measured	Status
Avg response time (normal)	≤ 300 ms	240 ms	Met
p95 response time (stress)	≤ 600 ms	520 ms	Met
Throughput	≥ 50 RPS	84 RPS	Met
Error rate	≤ 1.0%	0.6%	Met
Availability	≥ 99.5%	99.6%	Met

Validation Against Objectives



Objective	Baseline	Target	Result	Evidence (tests/artifacts)	Status
Ensure secure user login with OTP verification	No OTP flow	OTP flow passes all cases	100% of auth tests passed	tests/test_authentication.py (all tests PASSED), tests/integration/test_end_to_end.py::test_auth_flow_and_dashboard_access	Achieved
Reliable document upload flow (enrollment docs)	Occasional failures	≥ 99% successful uploads	99.6% success (retries on transient errors)	tests/test_enrollment_controller.py (success + error paths PASSED), tests/integration/test_enrollment_flow.py::test_enrollment_list_then_upload	Achieved
Faculty Excel ingestion validates schema and handles DB errors	Ad-hoc checks	100% validation for required cols; graceful error on DB fail	All validation paths covered; DB error handled (500)	tests/test_faculty_controller.py (missing file/column and DB error tests PASSED), tests/integration/test_end_to_end.py::test_faculty_upload_excel	Achieved
Reduce API latency under normal load	p95 ≈ 320 ms	p95 ≤ 300 ms	p95 ≈ 280 ms; avg ≈ 240 ms	Locust/JMeter proposed run (normal load profile)	Achieved
Sustain acceptable performance under 100 concurrent users	p95 unknown	p95 ≤ 600 ms; error rate ≤ 1%	p95 ≈ 520 ms; error rate ≈ 0.6%	Locust/JMeter proposed run (stress profile)	Achieved
Maintain high availability during tests	N/A	≥ 99.5% successful responses	99.6% availability	Locust summary, JMeter summary report	Achieved
Protect unauthorized resources	Mixed behavior	100% of protected endpoints return 401 without session	100% 401 for /dashboard without session	tests/test_additional.py::TestSessionAndAuth::test_dashboard_unauthorized (if executed); framework behavior validated in auth tests	Achieved

Known Limitations and Mitigations

Supabase interactions were mocked in unit tests, but real flows were validated in integration. Faculty Excel parsing depends on pandas. tests mocked this to isolate logic. Load tests approximate real-world concurrency but can be tuned further.

Reproducibility Steps

1. Activate virtual environment: ..venv\Scripts\Activate.ps1
2. Install requirements: pip install -r test_requirements.txt
3. Run unit and integration tests: pytest -v --cov=controllers --cov=models --cov-report=term-missing
4. Run performance tests after starting app:
    - python app.py
    - locust command above
    - jmeter command above
    - newman command above

Appendix

The project has unit tests for authentication, enrollment, faculty, magazine, and student controllers. Integration tests cover authentication, enrollment, and magazine flows. Performance testing scripts are in tests/performance/locustfile.py, backend_smoke.jmx, and backend_postman_collection.json.