IS-BST University Platform

System Requirements Specification (SRS)

Date:
December 15, 2025

Version: 1.0.0

Abstract

This document defines the comprehensive technical specifications for the IS-BST University Platform, a cloud-native academic management system. It details the architecture, security controls, and functional requirements necessary to meet FERPA and GDPR compliance standards. The system utilizes Next.js 14, Convex, and Clerk to provide a secure, scalable, and responsive experience for students, faculty, and administrators.

Revision History

Version	Date	Description	Author
0.1.0	13-12-2025	Initial Draft	System Architect
0.5.0	13-12-2025	Added Security & Compliance Sections	Security Lead
0.9.0	14-12-2025	Final Technical Review	Tech Lead
1.0.0	14-12-2025	Release for Accreditation	Lead Architect

Approvals

Chief Information Officer	**Compliance Officer**
Date: December 15, 2025	Date: December 15, 2025

Contents

1 Introduction

1.1 Purpose

The purpose of this System Requirements Specification (SRS) is to define the technical architecture, functional requirements, and security controls for the IS-BST University Platform. This document serves as the primary artifact for accreditation audits and provides a definitive guide for development and operations teams. It ensures that the system meets the diverse needs of academic stakeholders while adhering to strict regulatory frameworks such as FERPA and GDPR.

1.2 Scope

The IS-BST University Platform is a comprehensive student information system (SIS) and learning management system (LMS) designed to unified academic operations. The scope includes:

- **Academic Management**: Course catalog, enrollment, grading, and scheduling.
- **Financial Operations**: Tuition billing, payment processing, and financial aid tracking.
- **Human Resources**: Faculty onboarding, payroll integration, and department assignments.
- **Administration**: System configuration, audit logging, and role-based access control (RBAC).

The system explicitly excludes legacy mainframe integrations, focusing solely on a modern, cloud-native architecture.

1.3 Definitions and Acronyms

SRS System Requirements Specification

FERPA Family Educational Rights and Privacy Act

GDPR General Data Protection Regulation

LMS Learning Management System

SIS Student Information System

RBAC Role-Based Access Control

SSR Server-Side Rendering (Next.js)

SaaS Software as a Service

1.4 References

1. IS-BST Implementation Plan v1.0
2. FERPA Compliance Guide (34 CFR Part 99)
3. GDPR Article 32: Security of Processing
4. Next.js 14 Documentation
5. Convex Database Documentation

1.5 Overview

The remainder of this document is organized as follows: Section 2 details the system architecture. Section 3 outlines specific module requirements. Sections 4-5 present UML models and database schemas. Section 6 covers security and compliance. Sections 7-8 address non-functional requirements and implementation details.

2 System Architecture

2.1 Architectural Overview

The IS-BST University Platform utilizes a Component-Based Architecture, leveraging a cloud-native stack to ensure high availability and scalability. The system separates the User Interface (UI), Business Logic, and Data Persistence layers, connected via secure APIs.

The core technology stack consists of:

- **Frontend**: Next.js 14 utilizing the App Router and Server Components for optimal performance and SEO.
- **Backend**: Convex, creating a serverless API layer with integrated database functions.
- **Database**: Convex Database, a reactive, distributed document store.
- **Authentication**: Clerk, handling identity management and multi-factor authentication.

2.2 System Context

The C4 Context diagram below illustrates the external systems and users interacting with the IS-BST Platform.

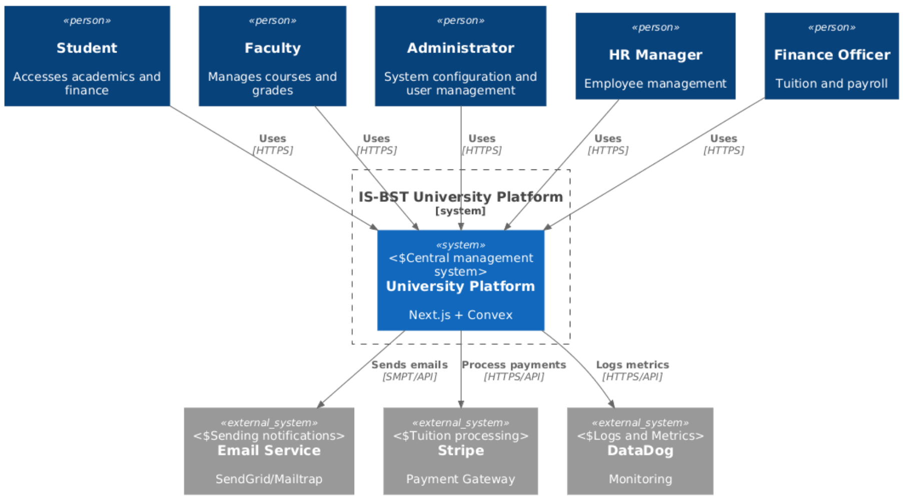


Figure 1: C4 System Context Diagram

Key interactions include:

1. **Students** access the portal to view grades and pay tuition.
2. **Stripe** processes credit card transactions securely (PCI-DSS compliant).
3. **DataDog** ingests logs for real-time monitoring.

2.3 Container Architecture

The Container diagram details the internal applications and their responsibilities.

2.4 Deployment Strategy

The platform is deployed across Vercel and Convex Cloud. Vercel manages the frontend assets and edge functions, while Convex handles the stateful backend logic. This separation allows for independent scaling of compute and storage resources.

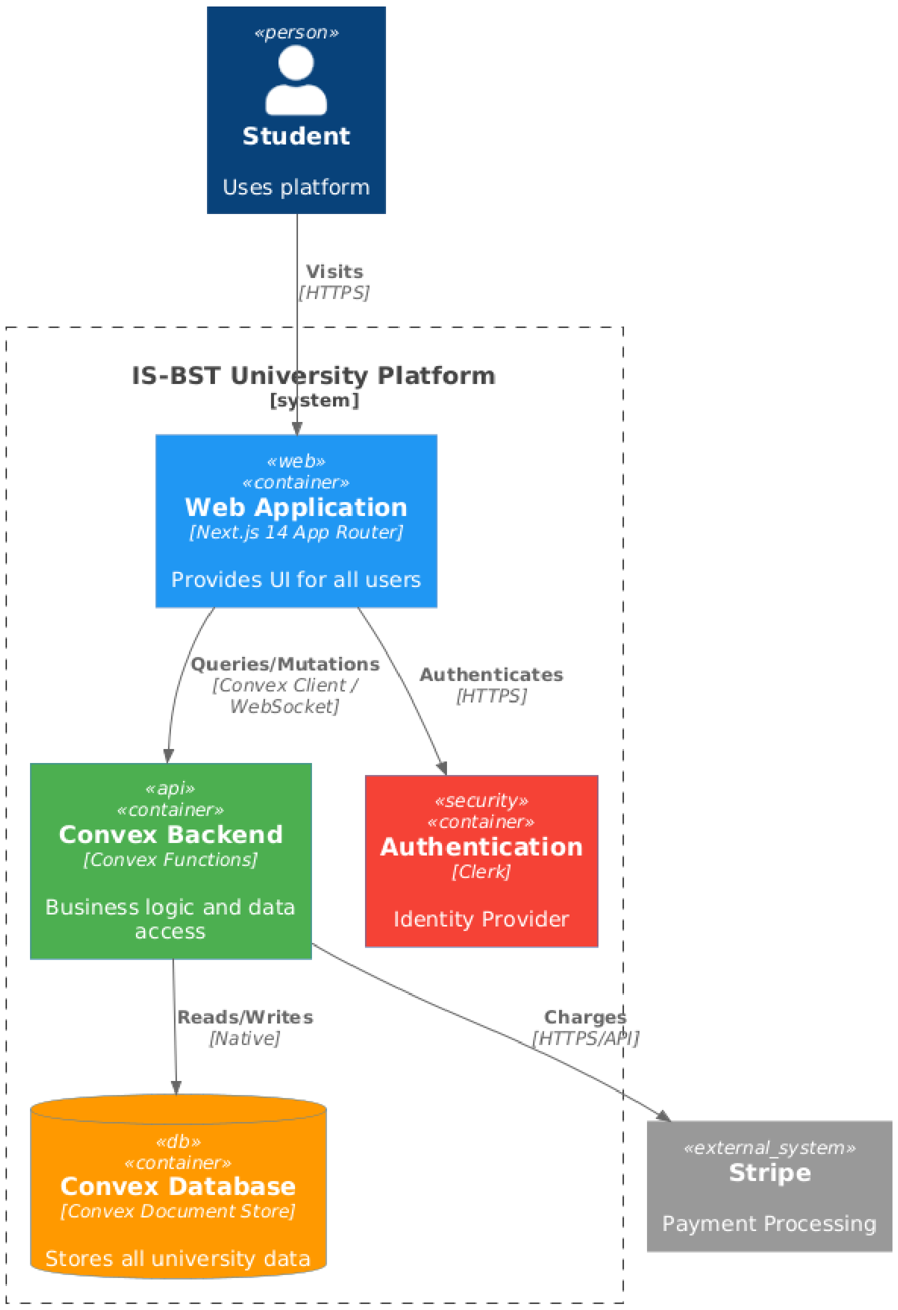


Figure 2: C4 Container Diagram

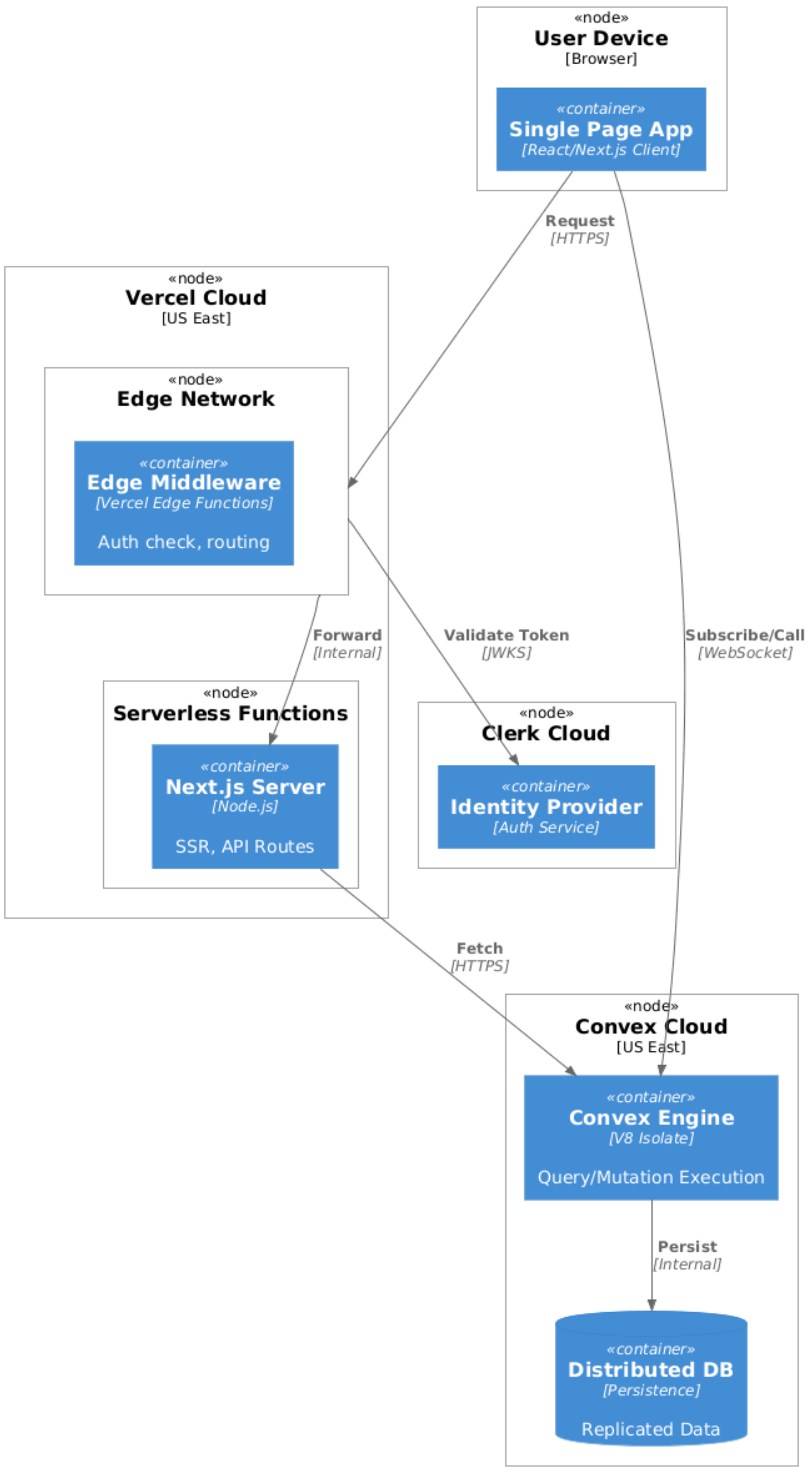


Figure 3: Deployment Diagram

3 System Modules

3.1 Academics Module

The Academics Module is the core of the educational experience.

- **Course Management**: Creation and scheduling of courses.
- **Enrollment**: Student registration logic with pre-requisite checks.
- **Grading**: Faculty submission of grades and transcript generation.

Requirement Ref: `REQ-ACAD-01` to `REQ-ACAD-15`.

3.2 Administration Module

Allows system administrators to configure global settings.

- **User Management**: Provisioning and de-provisioning accounts.
- **Audit Logs**: Viewing immutable logs for security compliance.
- **Term Management**: Opening and closing academic terms.

3.3 Finance Module

Handles all monetary transactions.

- **Tuition Billing**: Automatic invoice generation based on credit hours.
- **Payments**: Stripe integration for secure payments.
- **Financial Aid**: Tracking scholarships and verifying eligibility.

3.4 Human Resources Module

Manages faculty and staff data.

- **Onboarding**: Streamlined process for new hires.
- **Department Assignment**: Linking faculty to specific academic departments.

3.5 Component Interaction

The internal components interact via reactive subscriptions. The frontend subscribes to data queries, ensuring the UI is always consistent with the backend state.

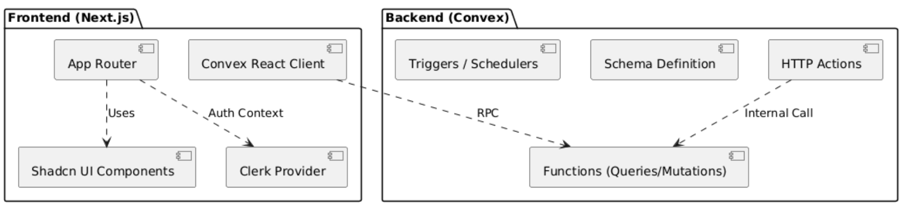


Figure 4: Component Diagram

4 UML Modeling

This section provides detailed behavioral and structural models of the system.

4.1 Use Case Modeling

The Academics module supports critical student and faculty interactions. The following diagram illustrates the primary use cases.

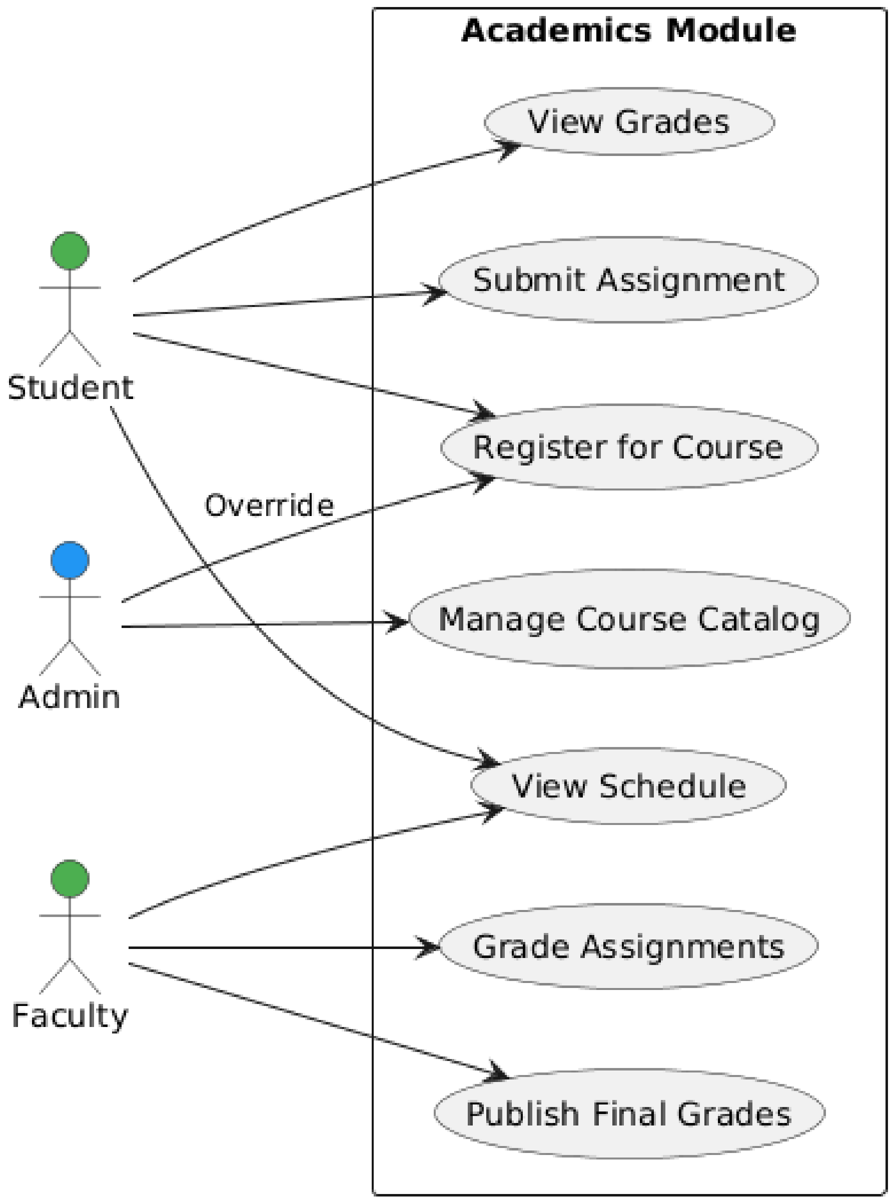


Figure 5: Academics Use Case Diagram

4.2 Domain Model

The core entities and their relationships are defined in the Domain Model. Note the central role of the `Student` and `Course` entities.

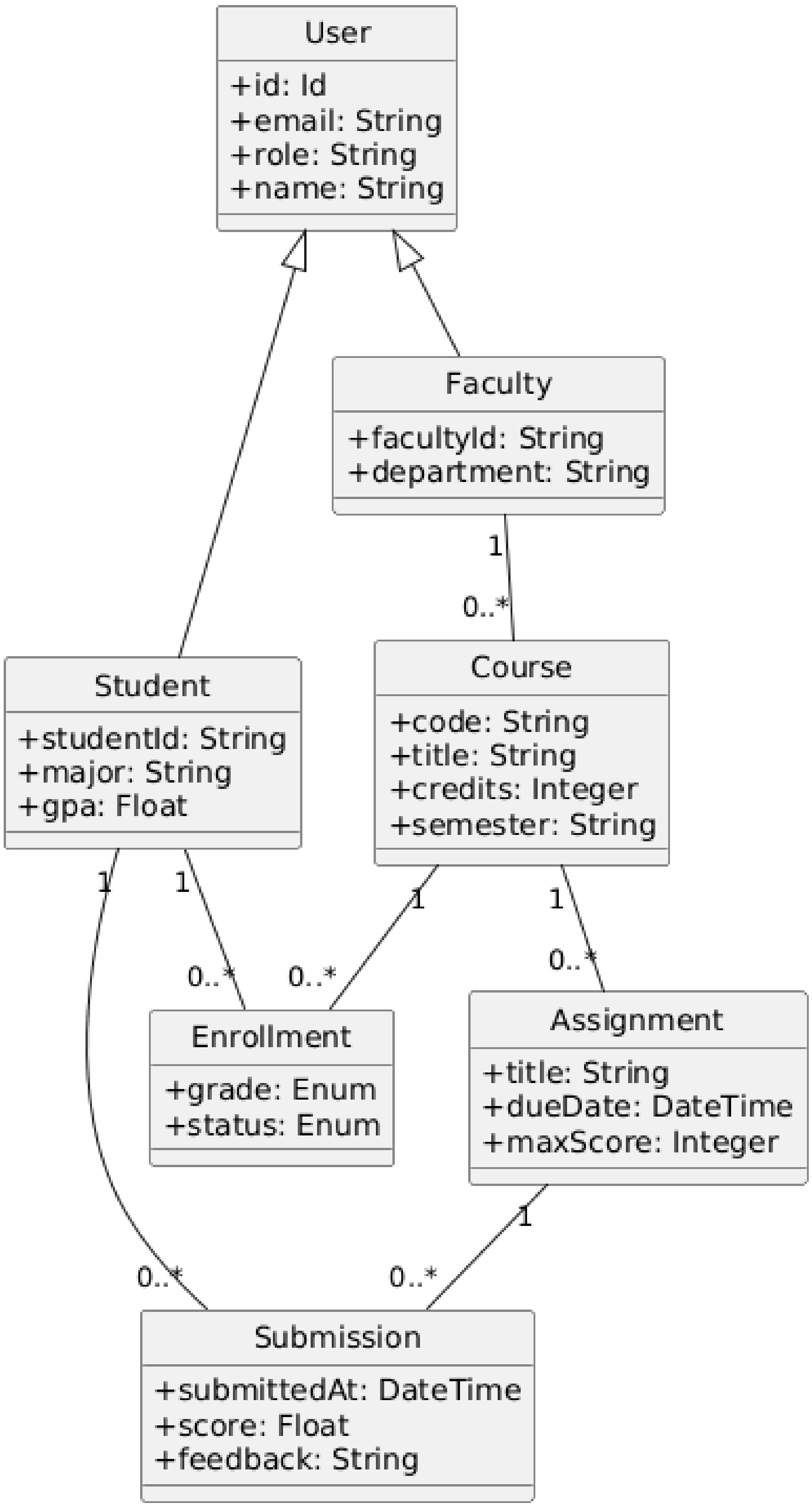


Figure 6: Domain Model (Class Diagram)

4.3 Sequence Diagrams

4.3.1 Grade Submission

The grade submission process requires strict authorization checks. The faculty member must be assigned to the course to submit grades.

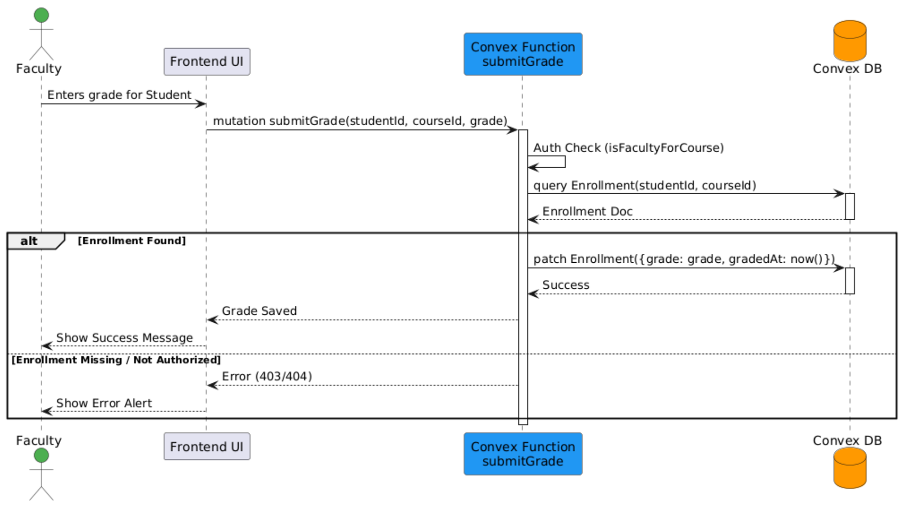

Faculty
Frontend UI
Convex Function submitGrade
Convex DB
Enters grade for Student
mutation submitGrade(studentId, courseId, grade)
Auth Check (isFacultyForCourse)
query Enrollment(studentId, courseId)
Enrollment Doc
alt [Enrollment Found]
patch Enrollment({grade: grade, gradedAt: now()})
Success
Grade Saved
Show Success Message
[Enrollment Missing / Not Authorized]
Error (403/404)
Show Error Alert

Figure 7: Grade Submission Sequence Diagram

4.3.2 Authentication Flow

The authentication process delegates identity verification to Clerk, issuing a short-lived JWT for API access.

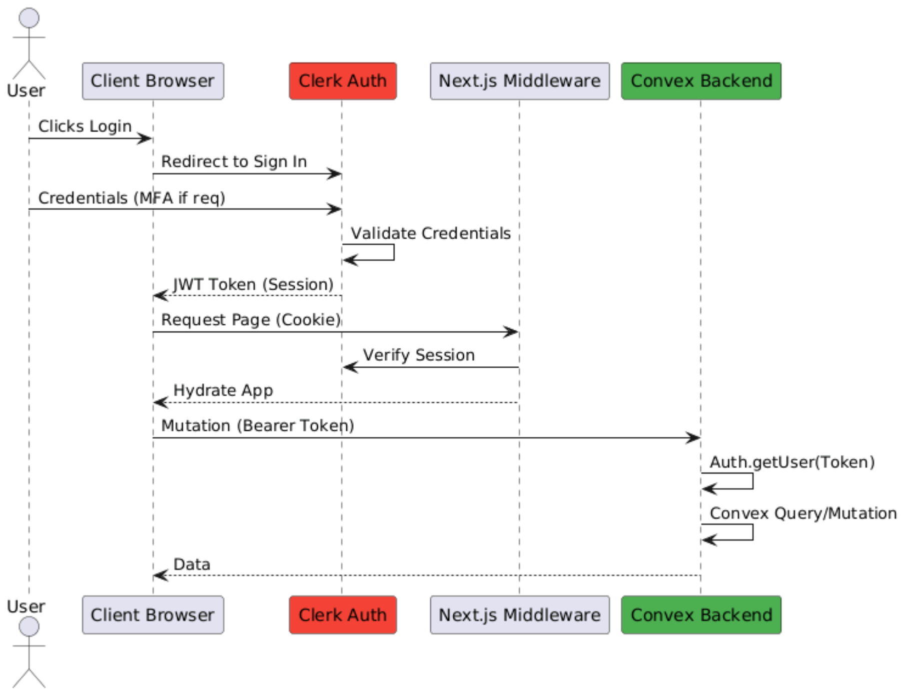
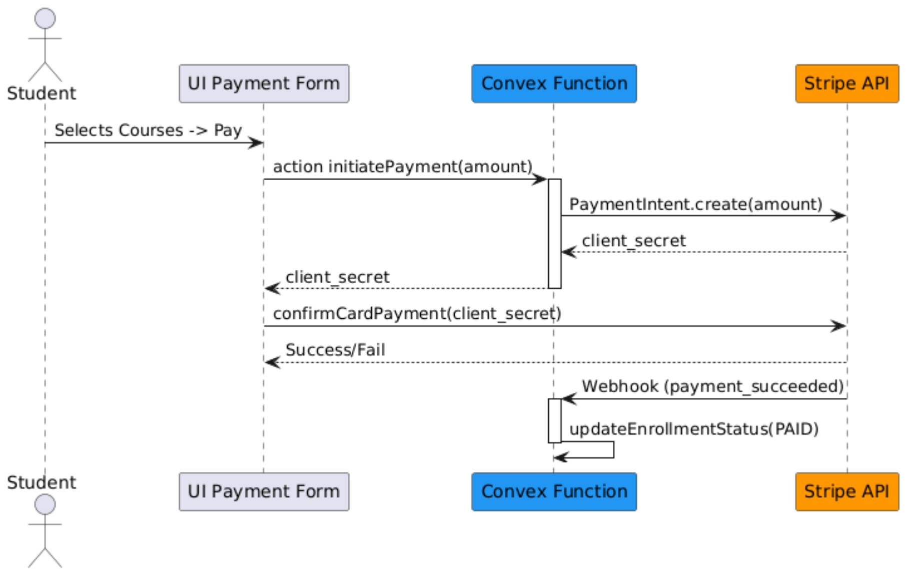


Figure 8: Authentication Sequence Diagram

4.3.3 Payment Processing

Payments are handled asynchronously using Webhooks to ensure status consistency even if the user closes the browser.



Student
UI Payment Form
Convex Function
Stripe API
Selects Courses -> Pay
action initiatePayment(amount)
PaymentIntent.create(amount)
client_secret
client_secret
confirmCardPayment(client_secret)
Success/Fail
Webhook (payment_succeeded)
updateEnrollmentStatus(PAID)

Figure 9: Payment Flow Sequence Diagram

5 Data Model & Schema

5.1 Database Technology

The system uses Convex, a distributed relational database with a document-oriented interface. This allows for strict schema validation while maintaining the flexibility of JSON documents.

5.2 Schema Definition (TypeScript)

The following schema defines the core tables for the platform. It enforces type safety at the database level.

Listing 1: Convex Schema Definition

```typescript
import { defineSchema, defineTable } from "convex/server";
import { v } from "convex/values";

export default defineSchema({
  users: defineTable({
    name: v.string(),
    email: v.string(),
    role: v.union(
      v.literal("student"),
      v.literal("faculty"),
      v.literal("admin")
    ),
    clerkId: v.string(), // Auth provider ID
  }).index("by_clerk_id", ["clerkId"]),

  courses: defineTable({
    code: v.string(),
    title: v.string(),
    credits: v.number(),
    semester: v.string(),
    facultyId: v.id("users"),
  }).index("by_faculty", ["facultyId"]),

  enrollments: defineTable({
    studentId: v.id("users"),
    courseId: v.id("courses"),
    grade: v.optional(v.number()),
    status: v.union(v.literal("active"), v.literal("completed")),
  }).index("by_student", ["studentId"])
    .index("by_course_student", ["courseId", "studentId"]), // Unique constraint

  auditLogs: defineTable({
    action: v.string(),
    actorId: v.id("users"),
    timestamp: v.number(),
    resourceId: v.string(),
    details: v.string(),
  }).index("by_timestamp", ["timestamp"]),
});
```

5.3 Data Dictionary

Table	Field	Description
`users`	`clerkId`	External ID from Clerk. Used for authentication binding. Immutable.
`enrollments`	`grade`	Numeric grade (0-100). Nullable until course completion.
`auditLogs`	`actorId`	Reference to the user who performed the action. Crucial for non-repudiation.

6 Security & Compliance

6.1 Authentication Architecture

Authentication is managed by Clerk, implementing industry-standard OAuth 2.0 and OpenID Connect protocols.

- **MFA**: Multi-Factor Authentication is enforced for all Administrative and Faculty accounts (NFR-SEC-02).
- **Session Management**: Short-lived JWTs (15 minutes) reduce the attack surface.

6.2 Role-Based Access Control (RBAC)

The system enforces strict RBAC at the API layer. Next.js Middleware ensures route protection, while Convex Functions validate user roles before executing data mutations.

Role	**Academics**	**Grades**	**Finance**	**HR**	**Admin**
Student	View Own	View Own	View Own	No	No
Faculty	View Assigned	Edit Assigned	No	View Own	No
Finance	No	No	View/Edit All	No	No
HR Manager	No	No	No	Edit All	No
Admin	Read Only	No	Read Only	Read Only	Full Config

6.3 Compliance Mapping

The system is designed to meet strict regulatory requirements.

6.3.1 FERPA Compliance

- **Access Control (§99.31)**: Faculty access is restricted strictly to students enrolled in their specific courses.
- **Record Privacy**: Grades are never displayed publicly. Encryption at rest protects sensitive records.

6.3.2 GDPR Article 32 (Security of Processing)

- **Encryption**: All data at rest is encrypted using AES-256. Data in transit is secured via TLS 1.3.
- **Pseudonymization**: User IDs are opaque strings, separating PII from system identifiers where possible.
- **Availability**: The distributed nature of Convex and Vercel ensures resilience against localized failures.

6.4 Encryption Standard

- **At Rest**: Convex handles transparent disk-level encryption.
- **In Transit**: HSTS is enabled. All connections must use HTTPS.

7 Non-Functional Requirements (NFR)

7.1 Performance

1. **Response Time**: 95% of API requests shall complete within 200ms (NFR-PERF-01).
2. **Page Load**: Time-to-Interactive (TTI) shall be under 1.5 seconds on 4G networks (NFR-PERF-02).
3. **Throughput**: System supports 2,000 concurrent active users without degradation (NFR-PERF-03).

7.2 Reliability & Availability

1. **Uptime**: Target availability is 99.9% during business hours (NFR-REL-01).
2. **Recovery**: Recovery Time Objective (RTO) is ¡ 1 hour. Recovery Point Objective (RPO) is ¡ 5 minutes (NFR-REL-02).

7.3 Scalability

The architecture leverages serverless components allowing for auto-scaling.

- **Compute**: Vercel Edge Functions scale to zero and up to thousands of instances automatically.
- **Database**: Convex handles sharding and connection pooling transparently.

7.4 Observability

DataDog is integrated for comprehensive monitoring.

- **Logging**: Structured JSON logs for all API events.
- **Alerting**: Automated alerts for error rates exceeding 1% over a 5-minute window.

8 Implementation & Delivery

8.1 Development Methodology

The project follows an Agile Scrum methodology with 2-week sprints.

- **Version Control**: Git flow with feature branches.
- **Code Review**: Mandatory peer review for all Pull Requests (PRs).

8.2 CI/CD Pipeline

Deployment is automated via GitHub Actions and Vercel.

The pipeline stages include:

1. **Lint & Type Check**: ESLint and TypeScript compilation.
2. **Unit Tests**: Jest runner for business logic.
3. **Preview Deploy**: Vercel creates a unique URL for the PR.
4. **E2E Tests**: Cypress suite runs against the preview URL.

8.3 Testing Strategy

Type	**Scope**	**Tool**
Unit	Individual Functions	Jest
Integration	API Endpoints	Convex Test
E2E	Critical User Flows	Cypress
Security	Dependency Scanning	Snyk

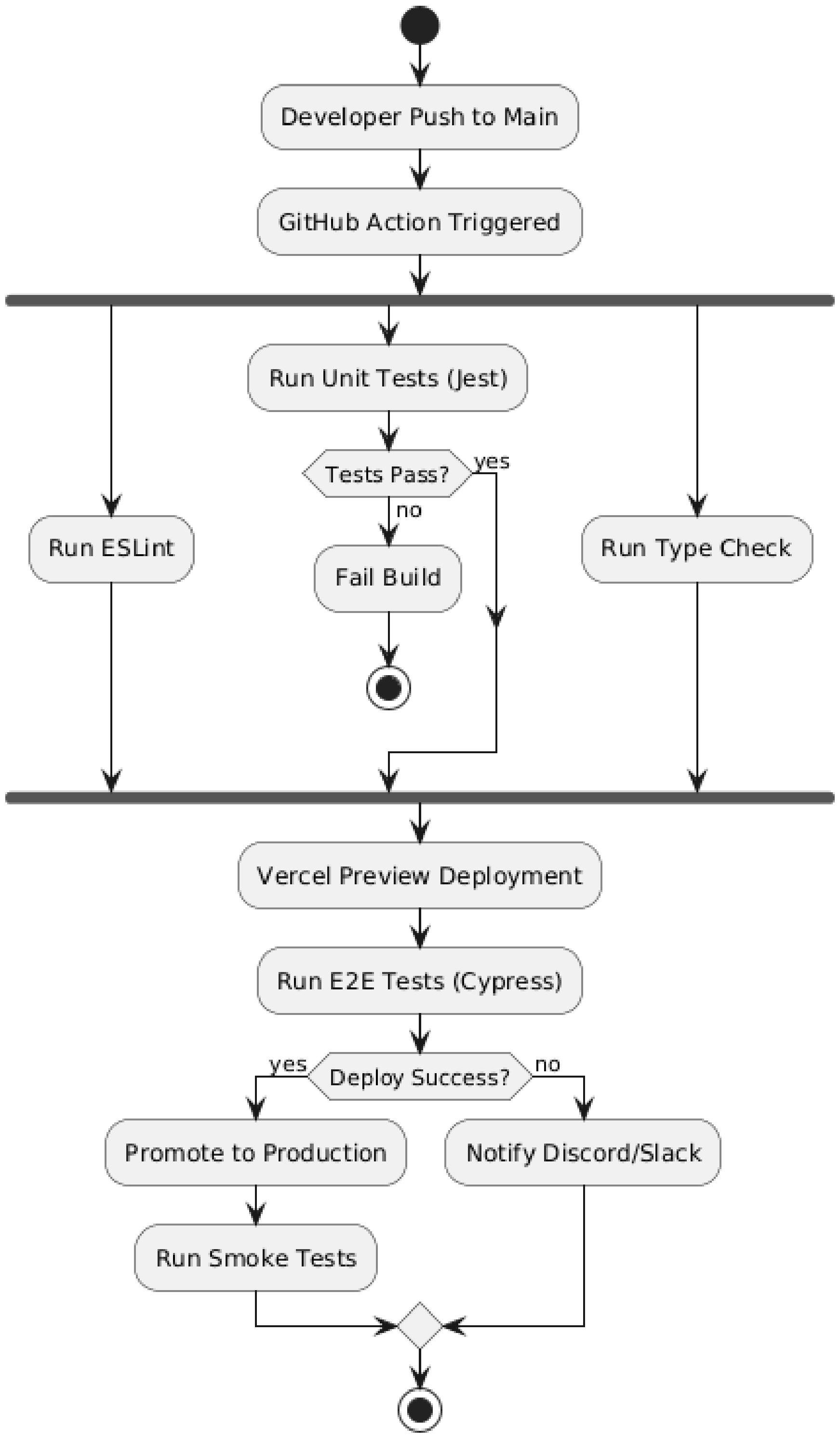


Figure 10: CI/CD Pipeline DevOps Flow

9 Appendices

A Glossary

Convex A full-stack application platform ensuring strong consistency.

Clerk A user management and authentication service.

Shadcn/UI A component library based on Radix UI and Tailwind CSS.

B Diagram Index

List of Figures

C Reference Material

All architectural decisions are based on the following official documentation versions as of December 15, 2025:

- Next.js 14 (App Router)
- Convex 1.8
- Clerk 4.0
- React 18