System Requirements Specification (SRS)

1. Introduction

1.1 Purpose

This SRS describes the requirements for the AI-powered Student Tutoring Platform — a web-based marketplace and learning environment that connects students with verified tutors, provides live interactive classrooms, AI-assisted learning, resource management, and secure payments.

1.2 Scope

The system will provide user registration, tutor onboarding and verification, course and session management, live video classrooms, AI tutoring and summarisation features, payments with escrow, admin dashboards, and reporting. The platform comprises a frontend web application, backend REST API, a persistent database, AI microservices, storage for media, and third-party integrations (WebRTC provider, payment gateway, speech/LLM APIs).

2. Overall Description

2.1 Stakeholders

Stakeholder	Description
Students	Primary end-users seeking tutoring and learning resources.
Tutors	Educators offering lessons and managing courses.
Administrators	Platform operators for verification, dispute resolution and governance.
Parents/Guardians	Optional monitors and payers for underage students.
Developers/Operators	Maintain the system and manage deployments.

2.2 System Context

The platform consists of: Frontend (Next.js web app), Backend API (Django REST Framework), Database (PostgreSQL), Media storage (S3-Compatible), AI microservices (LLM + STT/TTS), WebRTC provider for live video, Payment gateway (Stripe Connect), and Monitoring/CI infrastructure.

3. Functional Requirements

ID	Feature	Description
FR-1	User Registration & Authentication	Users can register and authenticate using email. JWT tokens or httpC
FR-2	Tutor Onboarding & Verification	Tutors submit credentials for admin verification; status tracked (pendi
FR-3	Course & Session Management	Tutors create courses; students enroll and book sessions; bookings h
FR-4	Live Classroom	Create live session rooms with WebRTC tokens, in-session chat, whit
FR-5	AI Assistant & Summaries	Session transcripts are summarised; LLM provides tutoring Q&A; moc
FR-6	Payments & Escrow	Payments captured via Stripe; funds held until session completion an
FR-7	Resource Library	Upload, tag and control access to resources linked to courses.
FR-8	Admin Dashboard	Admin actions for user management, verification, dispute handling, ar

4. Non-Functional Requirements

Area	Requirement
Performance	System must support 1000 concurrent users and 100 concurrent live sessions in initial de
Availability	99.9% uptime target; monitoring and automated recovery in place.
Security	HTTPS/TLS; JWT; password hashing; input validation; rate limiting; audit logs; PCI DSS
Privacy	GDPR compliance: data export and deletion endpoints; data minimisation and encryption
Maintainability	Modular services, clear API contracts, automated tests and CI/CD pipelines.
Scalability	Use managed DB, auto-scaling groups, and CDN for media distribution.

5. Data Model (High-Level)

Primary entities: User, Profile, Course, Session, Booking, Resource, Transcript, Payment, Review. See ER diagram in deliverables for relationships.

6. API Endpoints (Representative subset)

Endpoint	Purpose
POST /api/auth/register	Register user
POST /api/auth/login	Login and receive tokens
GET /api/tutors	Search and list tutors
POST /api/courses	Create a course (tutor)
POST /api/bookings	Create booking & payment intent
POST /api/sessions/{id}/start	Create live session & WebRTC token
GET /api/admin/tutors/pending	List tutors awaiting verification
POST /api/ai/summarize	Generate session summary from transcript

7. Security & Privacy Considerations

Authentication via JWT with short-lived access tokens and httpOnly refresh tokens. Role-based access controls. PCI compliance for payment processing via Stripe. Sensitive files stored encrypted in S3 with restricted access. Logging and monitoring for suspicious activity. User data export and deletion flows for GDPR.

8. Testing Strategy

Unit tests (pytest, jest), integration tests for API endpoints, end-to-end tests (Cypress), load testing (k6/Locust), security testing (OWASP mobile/web scans). UAT with pilot cohort of students and tutors.

9. Deployment & DevOps

Containerised services (Docker), CI/CD via GitHub Actions, Infra-as-Code with Terraform (optional), deploy frontend to Vercel or CDN-backed host, backend to AWS ECS/EKS or managed services. Monitoring via Sentry/Datadog and logs in ELK/CloudWatch.

10. Project Roadmap & Milestones (12-week plan)

Weeks 1-2: Foundations — project setup, auth, DB models, basic frontend Weeks 3-5: Core features — tutor onboarding, course CRUD, booking, basic front UI Weeks 6-7: Live sessions — WebRTC integration, whiteboard, chat, recording Weeks 8-9: AI features — transcript, summaries, LLM integration Week 10: Payments & escrow, admin dashboard Week 11: QA, load testing, security audit Week 12: Deployment, docs, pilot launch

11. Acceptance Criteria

Platform runs locally with docker-compose; auth & roles work; tutors can be verified by admin; booking workflow works end-to-end; live session can be started; basic AI summaries available on recorded sessions; payments simulated in test mode.