IHSAN DOGRAMACI BILKENT UNIVERSITY

CS319: Object-Oriented Software Engineering

Deliverable-1 Report

Group: Insight Coding

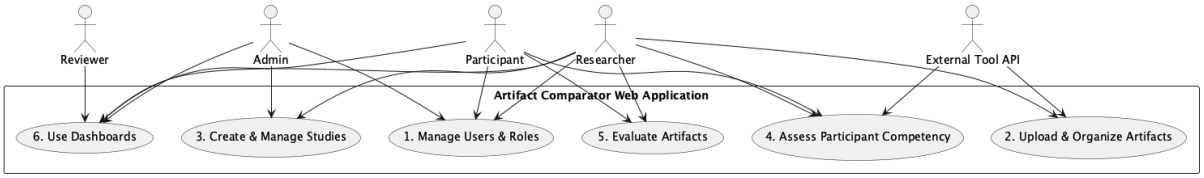
Group Members:	Mustafa Mert Gülhan	22201895
	Ece Bulut	22202662
	Yasemin Altun	22202739
	Altay İlker Yiğitel	22203024
	Efe Erdoğmuş	22203553

Instructor: Anıl Koyuncu

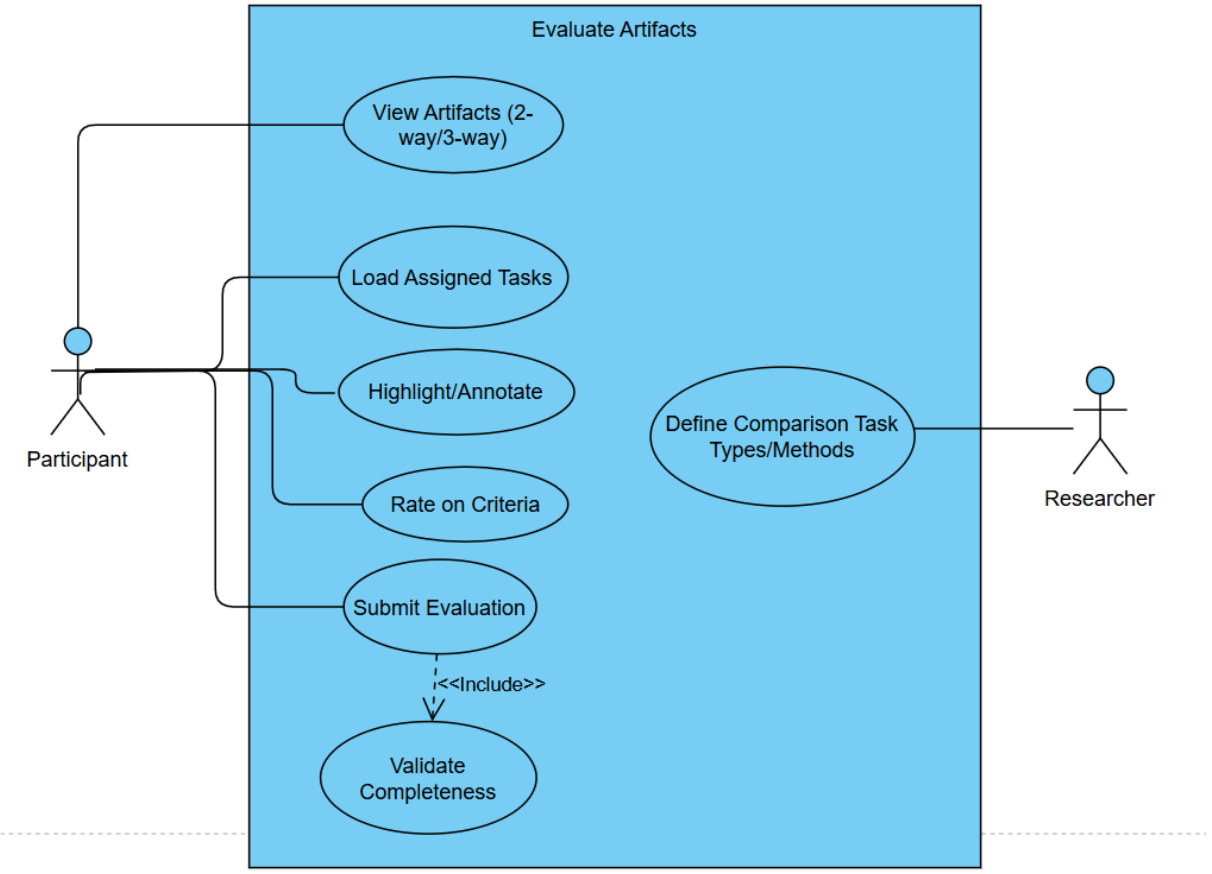
Submission Date: 12 October, 2025

1. Use Case Diagrams

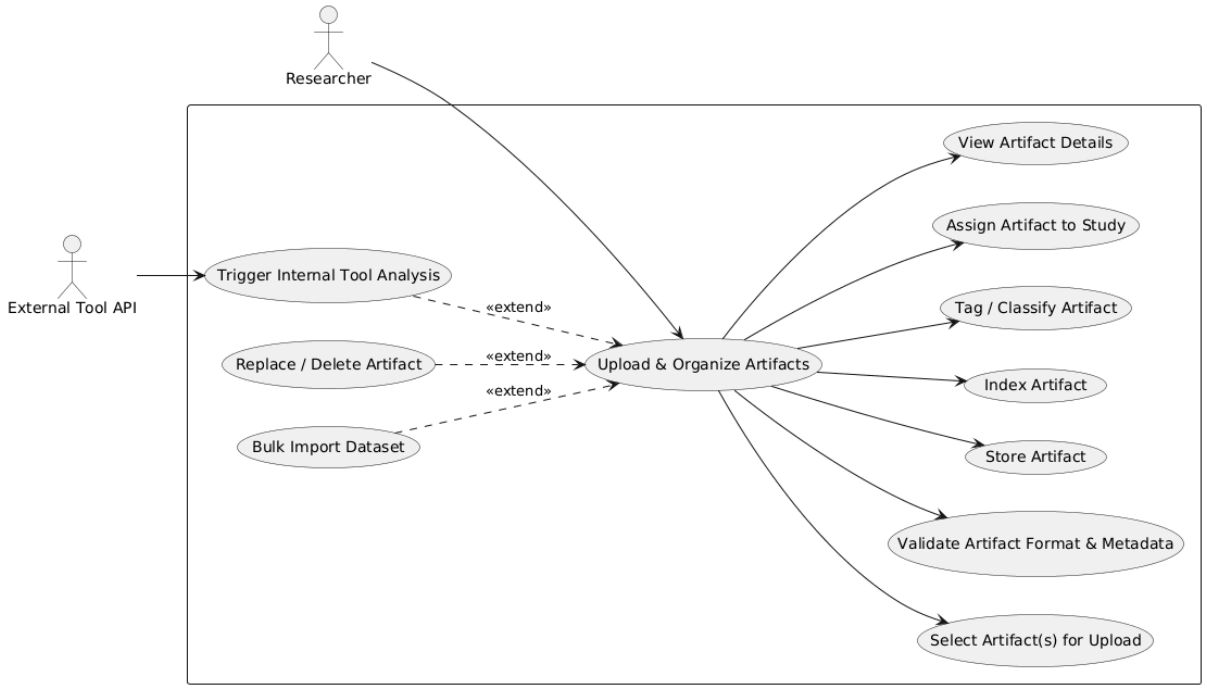
1.1. Summary Use Case Diagram



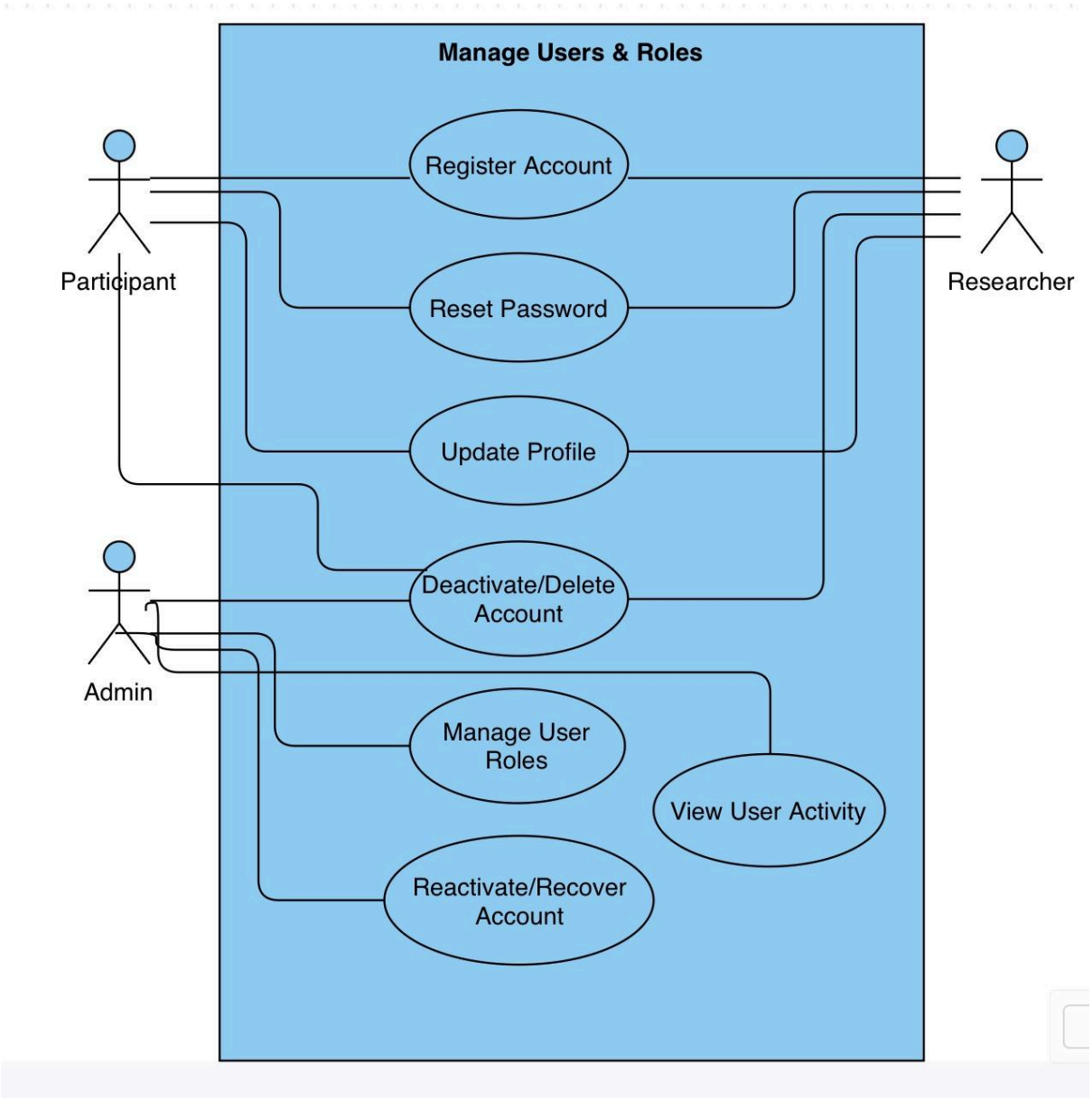
1.2. Evaluate Artifacts - Mustafa Mert Gülhan



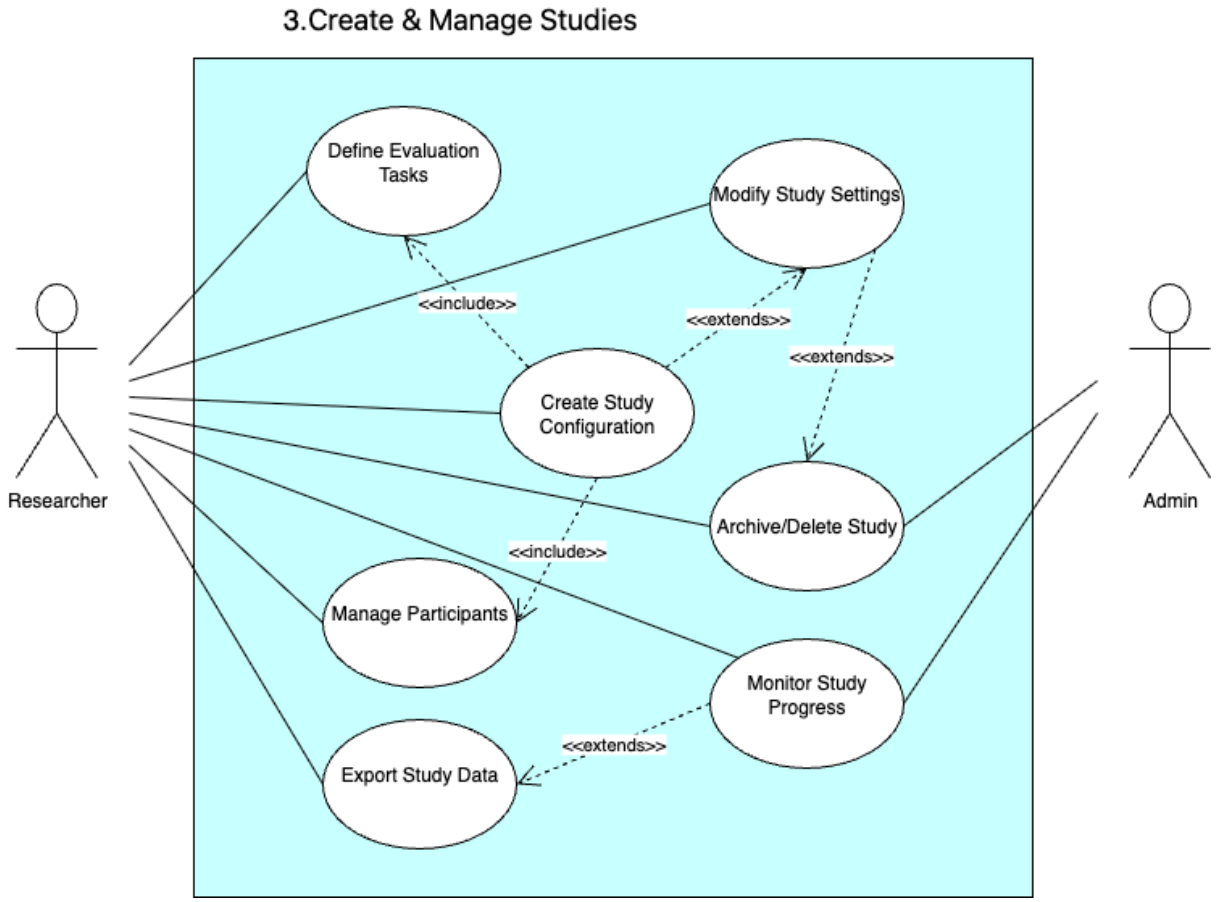
1.3. Upload & Organize Artifacts - Yasemin Altun



1.4. Manage Users & Roles - Group Work

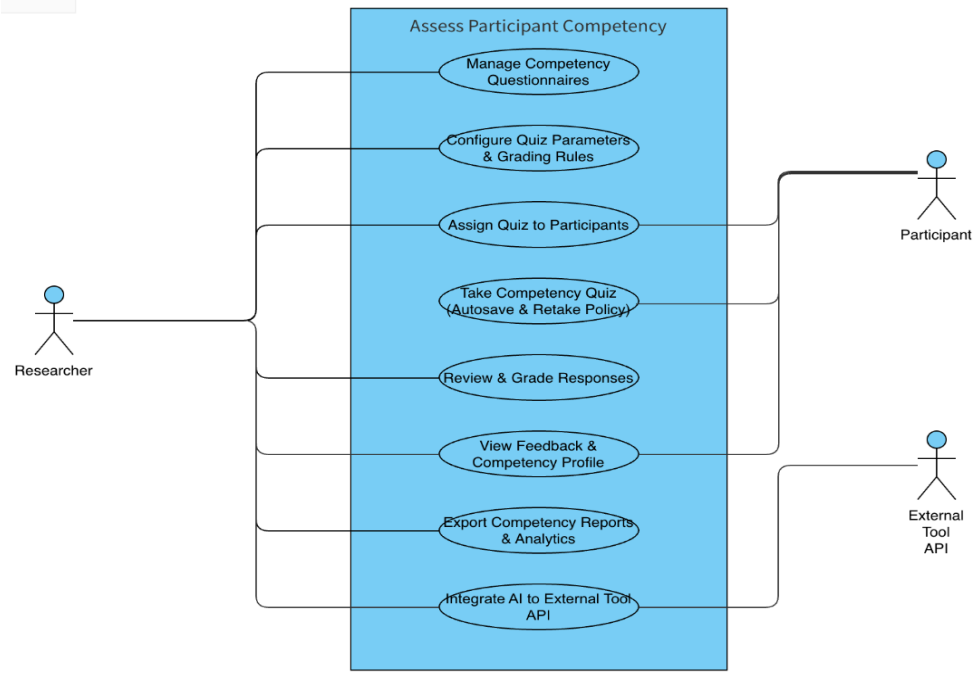


1.5. Create & Manage Studies - Efe Erdoğmuş

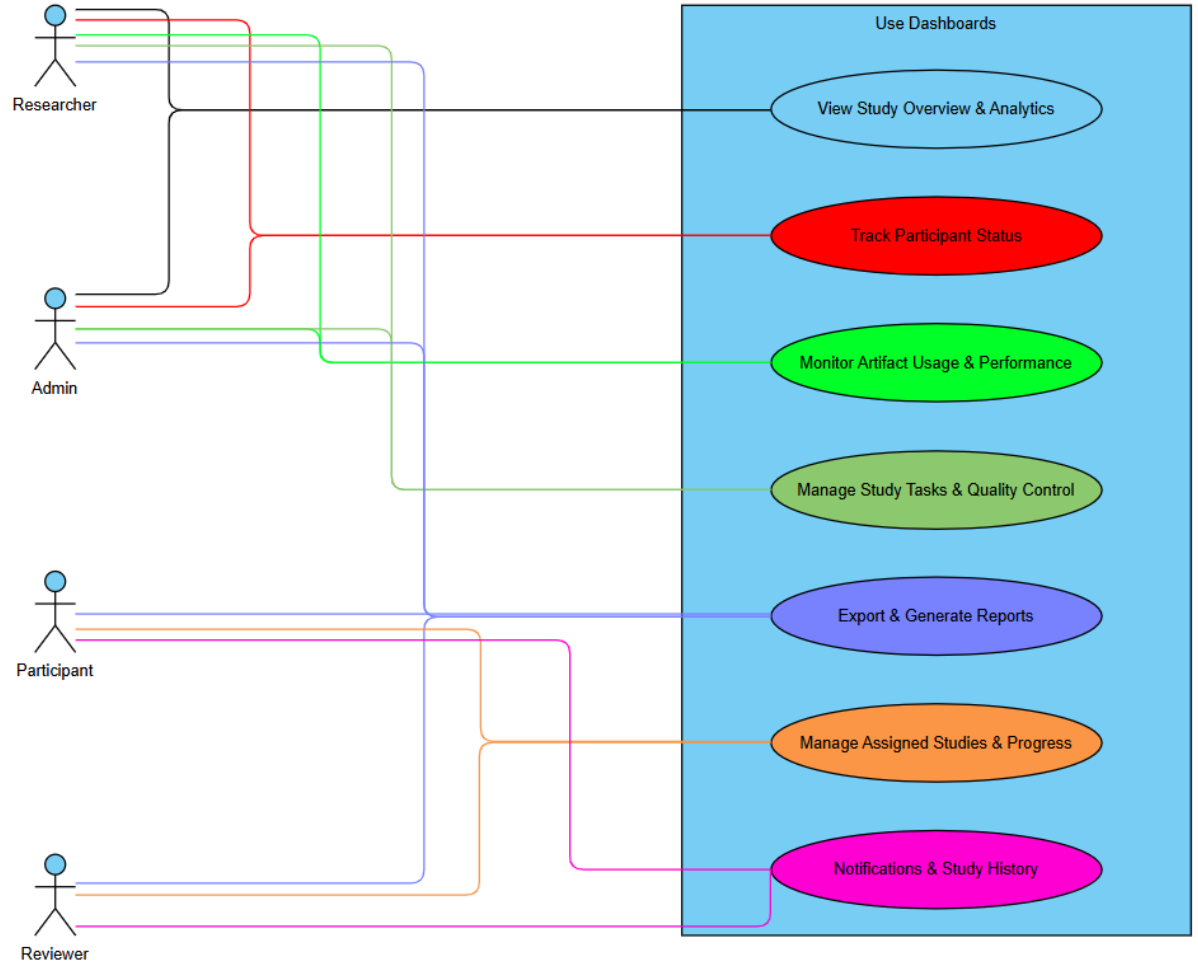


(https://drive.google.com/file/d/1UyxDCuZ8toumTg2PsDWE3X4hO7bl6He3/view?usp=sharing)

1.6. Assess Participant Competency - Altay İlker Yiğitel



1.7. Use Dashboards - Ece Bulut



2. Non-Functional Requirements

2.1. Security

All HTTP traffic has to be encrypted with TLS 1.3, and all data stored will have AES-256 encryption at rest.

Role-based Access Control(RBAC) will be imposed on every API request; unauthorized tries will be recorded and stored.

Under blind-review mode, origin information on artifacts will be fully masked, as confirmed with automated testing before release.

2.2. Usability&Accesability

2.2.1. Task completion

The essential workflows, comprising login, artifact upload, assignment, comparison, scoring, and export, are required to achieve a first-time completion success rate of at least 90% and an average completion duration not exceeding 5 minutes, as determined via user acceptance testing with a sample size of 10 or more participants.

2.2.2. Interface responsiveness

It has to be completely responsive to a series of display sizes, ranging from 360×640 (mobile portrait) to 1920×1080 (desktop).

All elements in layout are required to have appropriate alignment, spacing, and legibility of text such that nothing overlaps or gets clipped upon rendering at varying resolutions or zoomed or zoomed back in the browser.

2.2.3. Touch and Interaction Design

Interactive components (buttons, dropdowns, checkboxes) will have at least a surface that can be touched to be usable on phones and tablets.

Placement of the buttons must be systematic.

Hover and focus states will have to be clearly identified visually so that keyboard and touch users can clearly identify interaction targets.

2.2.4. Visual design consistency

The frontend will have a consistent color scheme and type system.

Contrast against background will meet WCAG 2.1 AA contrast ratio ≥ 4.5:1.

Error messages and validation feedback will be displayed within 3 seconds of invalid user entry and will clearly identify the issue in simple language.

2.2.5. Consistency across platforms

The interface also must have the same functionality and content across both a phone version and a desktop version, not something that is device-specific.

Critical workflows (login, comparison view, scoring) should be fully functional on touchscreen devices with standard mobile browsers (Chrome, Safari, Firefox).

2.3. Compatibility

The program will have to function correctly on the most recent two major versions of Firefox, Safari, Edge, and Chrome, confirmed by continuous integration testing.

Local deployment will be able to deploy Windows, macOS, and Linux environments; cloud deployment will have storage and processing queues configurable.

2.4. Performance

It will have a non-cluttered and interactive user interface such that the key pages (dashboard, comparison view, upload screen) will open in under 3 seconds under standard network conditions.

The backend (Spring Boot) will handle up to 100 concurrent users with no noticeable decrease in performance, with typical response times under 500 milliseconds for typical API requests.

Files uploading up to 50 MB will be compared and posted within 30 seconds after uploading. The frontend (React) will render pages effectively with minimal unnecessary re-renders and API calls to deliver seamless navigating between components.

The system will be architected to scale horizontally, and therefore more backend or frontend instances will be able to be provisioned whenever more concurrent users are wanted without architectural changes.

2.5. Scalability

The system must be able to support increasing numbers of users and artifacts without requiring major changes in architecture.

The backend (Spring Boot) needs to be horizontally scalable, thus multiple instances operate behind a load balancer in parallel in order to handle greater amounts of requests.

The cache and database layers must support a minimum of 10,000 stored artifacts with maximum response time for comparison and search queries.

The frontend (React) must be as efficient on big datasets through the use of pagination, lazy loading, and state management optimization to avoid slowdowns in performance.

2.6. Reliability

System uptime ≥99% during normal operations

Failed uploads retry automatically (max 3 attempts)

3. Selected Tech-Stack

3.1. Frontend: React + TypeScript + Vite

- React's component-based architecture is ideal for our complex multi-pane comparison views and annotation overlays
- TypeScript adds type safety for team coordination and reduces bugs
- Vite provides fast development experience for rapid iteration

3.2. Backend: Spring Boot

- Mature ecosystem with Spring Security for RBAC and authentication
- Spring Data JPA simplifies database operations for our relational model
- Strong OOP support aligns with course expectations for design patterns

3.3. Database: PostgreSQL

- ACID guarantees ensure data consistency for collaborative workflows
- JSONB support for flexible artifact metadata storage
- Proven scalability for handling thousands of artifacts

3.4. Deployment: Docker + Docker Compose

- Docker Compose enables one-command local setup across all platforms

- Production deployment via containerized services on cloud platforms

- Ensures dev/prod parity

3.5. Integrations

- Google Gemini API for optional AI-assisted features (quiz generation, summaries)

- Static analysis tools (SonarQube, ESLint) for code quality metrics

- S3 for scalable artifact storage