B.Tech. CSE (Hons.) (III YEAR – V SEM) (2025-2026)

DEPARTMENT OF COMPUTER ENGINEERING & APPLICATIONS

GLA University

17km Stone, NH-19, Mathura-Delhi Road, P.O. Chaumuhan, Mathura - 281406 (Uttar Pradesh) India

Project Title: Agentic Code Refactoring and Debt Auditor

Team Leader:	Ashish Kumar	**UR:**	2315800018
Team Member 1:	Akrati Gupta	**UR:**	2315800007

Mentor Name: Mr. Preshit Desai

Signature:

Project Synopsis: Agentic Code Refactoring and Debt Auditor

0. Cover

- **Project title:** Agentic Refactoring and Debt Auditor
- **Team name & ID:** Team T-25
- **Institute / Course:** GLA University, Mini-Project
- **Version:** v1.0
- **Date:** 28 Aug 2025

Revision history

Version	Date	Author	Change
v0.1	17 Aug 2025	Ashish Kumar	Initial draft
v1.0	26 Aug 2025	Akrati Gupta	Finalised Draft

1. Overview

- **Problem statement:** Agentic Code Refactoring and technical Debt Auditor.
- **Goal:** To build an intelligent agent that detects, analyses, and automatically refactors technical debt in codebases, making software cleaner, safer, and more sustainable.

- **Non-goals:** The system will not replace developers or act as a full IDE/debugger; it only assists with detecting and refactoring technical debt in supported languages.
- **Value proposition:** An AI assistant that improves code quality by detecting technical debt and suggesting intelligent refactoring.

---

2. Scope and Control

2.1 In-scope

Here are two solid Assumptions for your project:

- Developers will review and approve AI-suggested refactorings before applying them to production code.
- The project will initially support only major programming languages (Python, Java, JavaScript, C++) for analysis and refactoring.'

2.2 Out-of-scope

- Full legacy system migration, real-time debugging, and replacing developer decision-making.

2.3 Assumptions

- Developers will review and approve AI-suggested refactorings before applying them to production code.
- The project will initially support only major programming languages (Python, Java, JavaScript, C++) for analysis and refactoring.

2.4 Constraints

- The system's accuracy depends on the quality of training data and effectiveness of static analysis tools
- Limited to supported languages (Python, Java, JavaScript, C++) in the initial phase.
- Requires integration with existing developer workflows (CI/CD, GitHub/GitLab) for full effectiveness.

2.5 Dependencies

- Static Analysis Tools – SonarQube, pylint, flake8, radon, PMD for detecting code issues.
- AI/ML Models – GPT, CodeT5, StarCoder, CodeBERT for generating refactoring suggestions.
- Code Parsing Libraries – AST (Python), Spoon (Java), ESLint/Babel (JavaScript).
- Backend Frameworks – FastAPI/Flask (Python) or Node.js for orchestration.
- Frontend Frameworks – React + TailwindCSS for reporting dashboard.
- Visualization Libraries – Chart.js, D3.js, Plotly for debt scoring reports.
- CI/CD Tools – GitHub Actions, GitLab CI, Jenkins for continuous monitoring.
- Infrastructure – Shared lab server for hosting and testing.

2.6 Acceptance criteria and sign-off

- System scans code in Python, Java, JavaScript, C++.
- Detects technical debt and gives AI-powered refactoring suggestions.
- Generates reports with debt scores and explanations.
- Sign-Off: Final deliverables reviewed and approved by Mentor.

Sign-off table

Stakeholder	Role	Decision area	Signature/Approval	Date
Mr. Preshit Desai	Mentor	Scope, final acceptance	Approved	28 Aug 2025
Mr. Ashish Kumar	Product Lead	Release readiness	Approved	26 Aug 2025

3. Stakeholders and RACI

Activity	Responsible (R)	Accountable (A)	Consulted (C)	Informed (I)
Requirements	Akrati Gupta	Ashish Kumar	Mentor	T-25
Design	Akrati Gupta	Ashish Kumar	Mentor	T-25
Implementation	Akrati Gupta	Ashish Kumar	Mentor	T-25
Testing	Akrati Gupta	Ashish Kumar	Mentor	T-25
Release	Ashish Kumar	Ashish Kumar	Mentor	Dept

4. Team and Roles

Member	Role	Responsibilities	Key skills	Availability	Contact
Ashish Kumar	Product Lead	Scope, backend, reviews,docs	Product, APIs	8 hrs/wk	ashish.kumar_cs.h23@gla.ac.in
Akrati Gupta	Tech Lead & Backend &Frontend	Docs	Node, Express, SQL	10 hrs/wk	akrati.gupta_cs.h23@gla.ac.in

5. Week-wise Plan and Assignments

(Example schedule for Sep–Oct 2025; adjust to your calendar.)

Week	Milestones	Ashish Kumar	Akrati Gupta	Deliverables	Status

1	Requirements	Define scope and plan	Research Tools	Draft SRS	Planned
2	Architecture	Design workflow and DB	Diagram and docs	Arch Doc	Planned
3	Setup and Research	Backend env setup	AI/ML env setup	Dev Env Report	Planned
4	Analysis Module	AST and metrices engine	Repo testing support	Analysis v1	Planned
5	Refactoring Engine	LLM integration	Rule-based refactors	Refactor v1	Planned
6	Reporting System	Backend Reports	UI/Dashboard Design	Report v1	Planned
7	CI/CD & Testing	Pipeline Setup	Test cases execution	Stable Build	Planned
8	Finalization	Demo and Presentation	Final Report/Guide	Final Dileverables	Planned

---

6. Users and UX

6.1 Personas

- **Developer Devika: Wants quick insights into technical debt in her project; values clarity and explanations.**
- Team Lead Tarun: Wants to monitor overall code health and debt scores; values reporting, visibility, and prevention.

6.2 Top user journeys

- Home → Upload/Connect Repo → Scan → View detected issues → Apply suggested refactorings → Commit changes.

KPI: Debt report generated in ≤ 2 minutes, refactoring suggestions ≥ 80% relevant.

- **Team Lead:**

  Dashboard → View overall debt score → Check history & trends → Export report → Share with team.

  KPI: Dashboard loads ≤ 1s p95, debt score accuracy ≥ 90%.

6.3 User stories

- As a developer, I want to see why a piece of code is flagged so I can learn and trust the system's recommendations.
- GIVEN I scan my repo, WHEN issues are detected, THEN I see a clear explanation of what's wrong and why it matters.
- As a team lead, I want to track debt reduction over time so I can ensure the team is improving code quality.
- GIVEN I open the dashboard, WHEN I check metrics, THEN I see a timeline chart of technical debt trends.

6.4 Accessibility & localization

- Keyboard-only navigation for reports & dashboard.
- High-contrast charts (WCAG AA compliance).
- Language: English only in phase 1; multi-language support in roadmap.
- RTL (Right-to-Left) not required.

---

7. Market and Competitors

7.1 Competitor table

Competitor	Product / Tool	Target Users	Key Features	Pricing	Strengths	Weaknesses	Our Differentiator

SonarQube	Static Analysis & Debt Auditor	Enterprises, Dev teams	Code smells, bug detection, debt scoring, dashboards	Free + Paid	Strong analysis engine, widely adopted	No AI refactoring, limited automation	Combines debt audit + AI refactoring
Code Climate	Code Quality & Maintainability	Startups, Agile teams	Maintainability checks, test coverage, debt reports	Paid	Simple integration with CI/CD	No automatic refactoring, limited AI support	End-to-end automation + AI learning
Amazon CodeGuru	AI Reviewer	AWS-based developers	Performance profiling, bug detection, recommendations	Paid (per hr)	Backed by AWS, good for Java/Python	Limited language support, AWS lock-in	Multi-language & extensible beyond AWS
Snyk Code	Security & Bug Detection	Security-focused teams	AI-powered vulnerability detection, code scanning	Paid	Great for DevSecOps	Focused only on security, not maintainability	Covers both quality + security debt

JetBrains IDEs	IDE Refactoring Tools	Individual developers	Manual refactoring, code hints, inspections	Paid	Developer-friendly, strong IDE integration	Manual effort required, not autonomous	Autonomous agent, works outside IDE
Facebook SapFix	Auto Bug-Fixer	Internal (FB Engineers)	AI-driven bug patching & suggestions	Internal	Advanced automation at scale	Internal-only, not available to public	Public, open solution for all teams

7.2 Positioning

- **Unique angle:**An AI-driven coding assistant that combines technical debt auditing and autonomous refactoring in a single workflow.
- **Measurable delta:** Cuts average refactoring and debt analysis time from **hours/days → minutes**, with automated debt scoring and explainable fixes integrated directly into CI/CD.

---

8. Objectives and Success Metrics

- **O1 Onboarding:** Median signup + email verify < 60 s by 15 Oct 2025. KPI: median seconds.
- **O2 Search performance:** p95 search latency ≤ 1 s by 20 Oct 2025. KPI: p95 ms.
- **O3 Listing completion:** ≥ 80% of started listings are published by 20 Oct 2025. KPI: completion rate.
- **O4 Accessibility:** AA issues 0 on core flows by release. KPI: a11y violations.

9. Key Features

Feature	Description	Priority	Dependencies	Acceptance Criteria
Auth & Email Verify	Register/login with college email	Must	SMTP, Database	GIVEN email WHEN register THEN verification mail sent in ≤ 5s; login possible after verify
Listings CRUD	Create, edit, delete items with images	Must	Auth, Storage	GIVEN form WHEN submit THEN listing visible and searchable
Search & Filters	Keyword, category, price sort	Must	Listings	GIVEN query WHEN search THEN results returned in ≤ 1s (p95 latency)
Profiles	Public seller profile	Should	Auth	GIVEN profile URL WHEN open THEN show listings and contact details
Moderation	Report abuse, hide listing	Could	Auth, Admin Panel	GIVEN report WHEN submit THEN status recorded; listing hidden if approved

10. Architecture

10.1 High-level

- **Clients:**
    - React Single Page Application (SPA) for web users
    - Mobile app (optional future extension)

- **Services:**
    - Auth Service → Handles registration, login, email verification (with campus SSO optional)
    - Listing Service → CRUD operations for items, image management
    - Search API → Full-text + filtered search (category, price, keyword)

- Data Stores:
    - MySQL → Primary database for users, listings, and transactions
    - Object Storage → Stores item images (e.g., AWS S3, MinIO, or campus server storage)

- Integrations:
    - SMTP → Email verification and notifications
    - Campus SSO (Optional) → Seamless login using university credentials

10.2 API spec snapshot

Endpoint	Method	Auth	Purpose	Request Schema	Response Schema	Codes
/api/auth/register	POST	—	Create account	`{ "email":`	`201 { "id":`	`201 Created`,

				string, "password": string }	string }	400 Bad Request
/api/auth/login	POST	—	Authenticate user	{ "email": string, "password": string }	200 { "token": string }	200 OK, 400 Bad Request, 401 Unauthorized
/api/listings	POST	JWT	Create listing	{ "title": string, "price": number, "images": [string] }	201 { "listingId": string }	201 Created, 400 Bad Request, 401 Unauthorized

/api/listings/{id}	GET	—	Get listing by ID	—	`200 { "id": string, "title": string, "price": number, "images": [string], "sellerId": string }`	`200 OK`, `404 Not Found`
/api/listings/{id}	PUT	JWT	Update listing	`{ "title"?: string, "price"?: number, "images"?: [string] }`	`200 { "updated": true }`	`200 OK`, `400 Bad Request`, `401 Unauthorized`, `404 Not Found`

/api/listings/{id}	DELETE	JWT	Delete listing	—	`200 { "deleted": true }`	`200 OK`, `401 Unauthorized`, `404 Not Found`
/api/search	GET	—	Search listings	Query params: `q`, `category`, `page`	`200 { "items": [], "total": number }`	`200 OK`

10.3 Config and secrets

- `.env` for local development (Git-ignored).
- Rotate **SMTP credentials** each academic term.
- Restrict access to **CI/CD secrets** to core maintainers only.

---

11. Data Design

11.1 Data dictionary

Entity	Field	Type	Null?	Allowed values	Source	Notes

User	id	UUID	No	—	System	Primary Key (PK)
	email	String	No	RFC 5322	User	Unique, indexed
	password_hash	String	No	—	System	Hashed + salted
	created_at	Timestamp	No	—	System	Record creation time
	role	Enum	No	{student, admin}	System	Access control

11.2 Schemas and migrations

- **Schema definition**:
  - All entities (User, Listing, Report, etc.) follow the **data dictionary** (11.1).
  - Relationships:
    - `User (1) – (M) Listing`
    - `User (1) – (M) Report`
    - `Listing (1) – (M) Report`
  - **ERD** provided in *Appendix*.

- **Migrations**:
  - Versioned using a migration tool (e.g., Sequelize, Prisma, or Knex).
  - Naming convention: `YYYYMMDDHHMM_<change>.sql`
  - Each migration includes:
    - `up` (apply schema change)

            - `down` (rollback)

- **Testing**:
    - Rollback tested on **staging DB** before production deploy.
    - Data integrity checks run post-migration.

11.3 Privacy, retention, backup/DR

- **PII**
    - Collected: *name, email*.
    - Protected via hashed passwords and encrypted transport (TLS).
    - Access restricted to authorized services only.

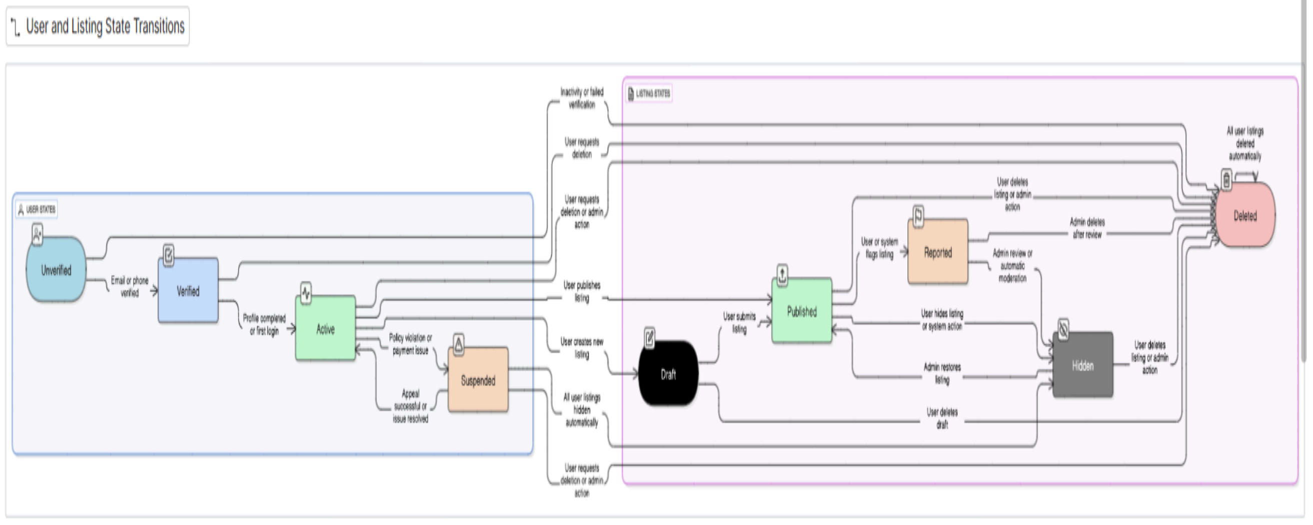
- **Retention**
    - User accounts deleted after **12 months of inactivity**.
    - Listings auto-purged after expiry or account deletion.
    - Audit logs retained for **90 days**.

- **Backups & Disaster Recovery**
    - **Nightly backups** of MySQL and object storage.
    - **RTO (Recovery Time Objective): 4 hours**.
    - **RPO (Recovery Point Objective): 24 hours**.
    - Backups stored in a separate availability zone.
    - Quarterly DR drills performed to validate recovery process.

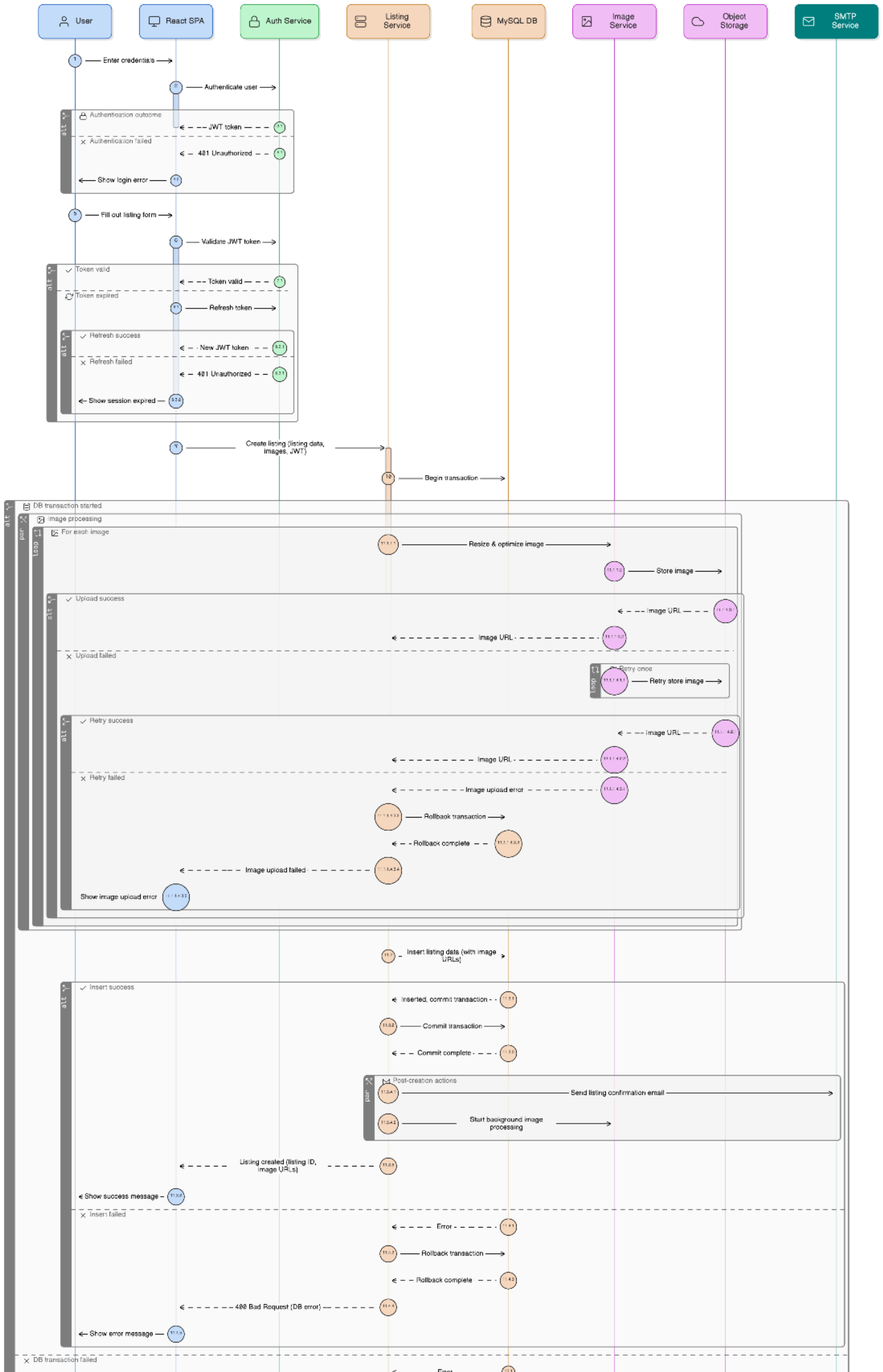
---

12. Technical Workflow Diagram

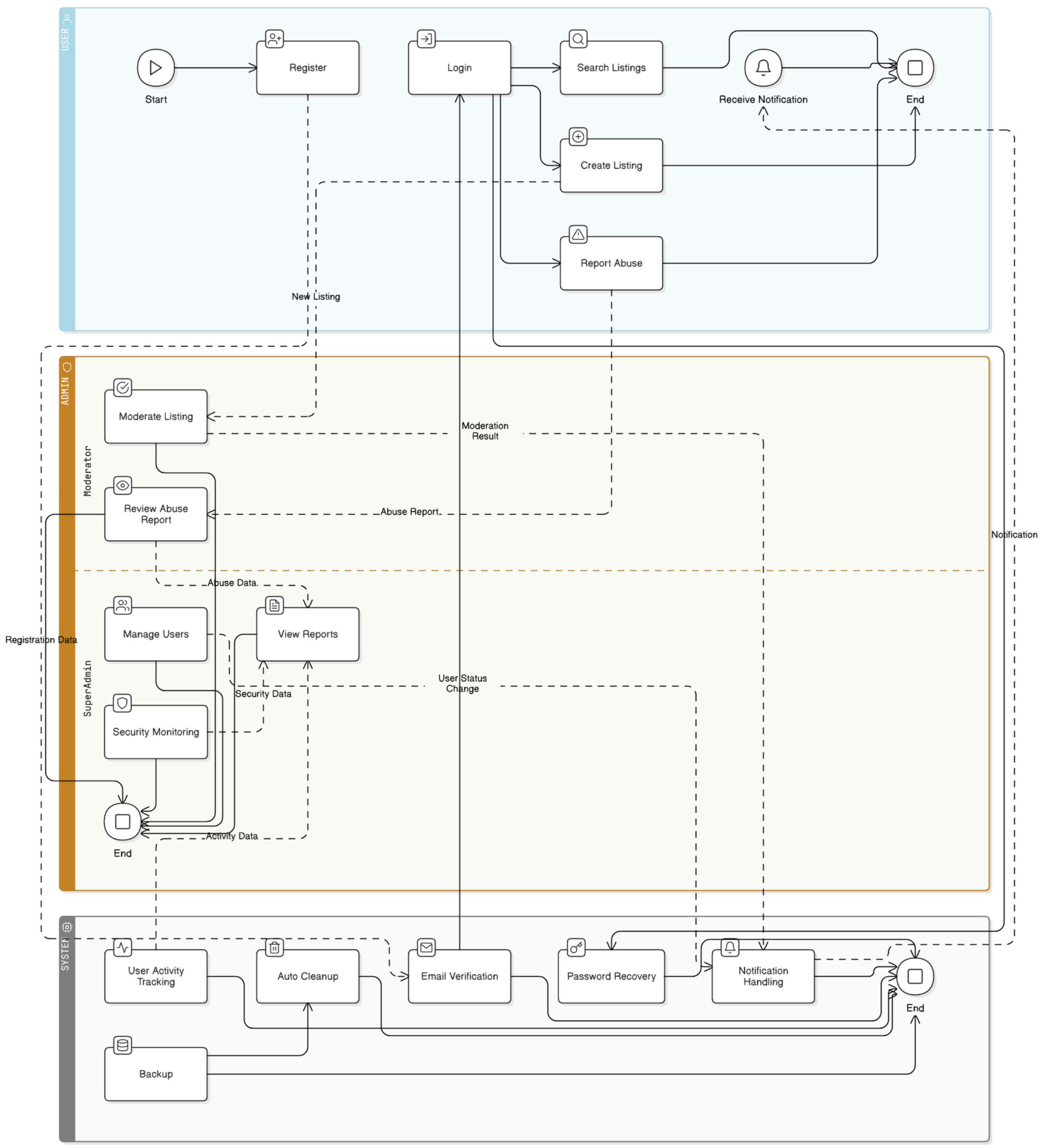
i. State Transition Diagram


User and Listing State Transitions
USER STATES
LISTING STATES
Unverified
Email or phone verified
Verified
Profile completed or first login
Active
Policy violation or payment issue
Suspended
Appeal successful or issue resolved
Inactivity or failed verification
User requests deletion
User requests deletion or admin action
User publishes listing
User creates new listing
All user listings hidden automatically
User requests deletion or admin action
Draft
User submits listing
Published
User or system flags listing
Reported
User deletes listing or admin action
Admin deletes after review
Admin review or automatic moderation
User hides listing or system action
Admin restores listing
Hidden
User deletes listing or admin action
User deletes draft
All user listings deleted automatically
Deleted

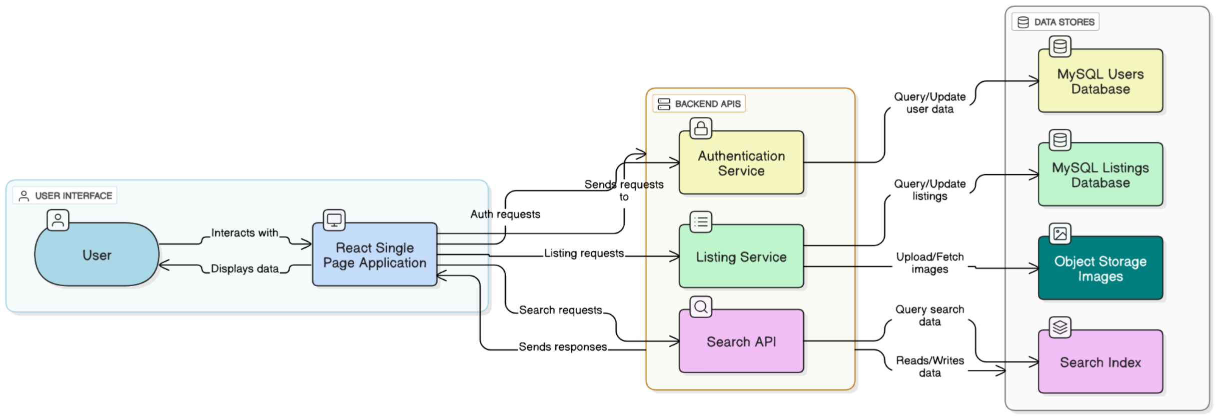
ii. Sequence Diagram


User
React SPA
Auth Service
Listing Service
MySQL DB
Image Service
Object Storage
SMTP Service
Enter credentials
Authenticate user
alt
Authentication outcome
JWT token
Authentication failed
401 Unauthorized
Show login error
Fill out listing form
Validate JWT token
alt
Token valid
Token valid
Token expired
Refresh token
alt
Refresh success
New JWT token
Refresh failed
401 Unauthorized
Show session expired
Create listing (listing data, images, JWT)
Begin transaction
alt
DB transaction started
par
Image processing
loop
For each image
Resize & optimize image
Store image
alt
Upload success
Image URL
Image URL
Upload failed
loop
Retry once
Retry store image
alt
Retry success
Image URL
Image URL
Retry failed
Image upload error
Rollback transaction
Rollback complete
Image upload failed
Show image upload error
Insert listing data (with image URLs)
alt
Insert success
Inserted, commit transaction
Commit transaction
Commit complete
par
Post-creation actions
Send listing confirmation email
Start background image processing
Listing created (listing ID, image URLs)
Show success message
Insert failed
Error
Rollback transaction
Rollback complete
400 Bad Request (DB error)
Show error message
DB transaction failed
Error

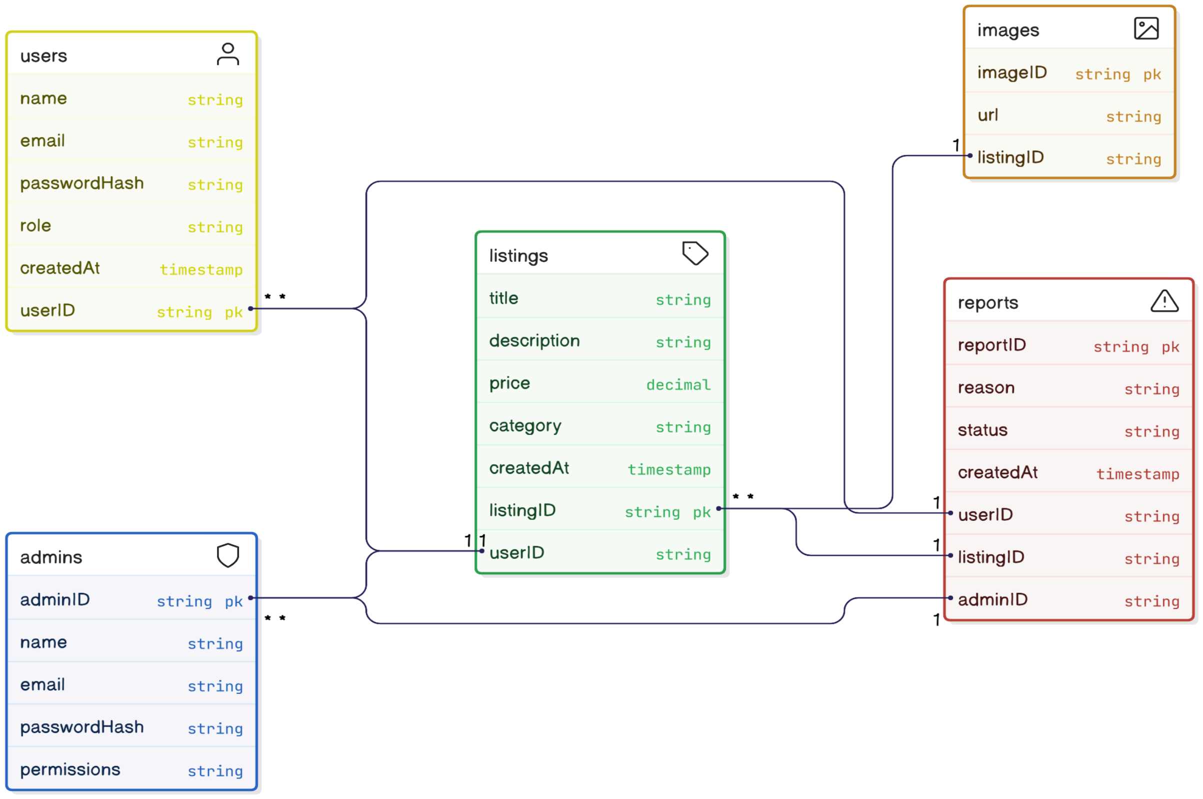
iii. Use Case Diagram



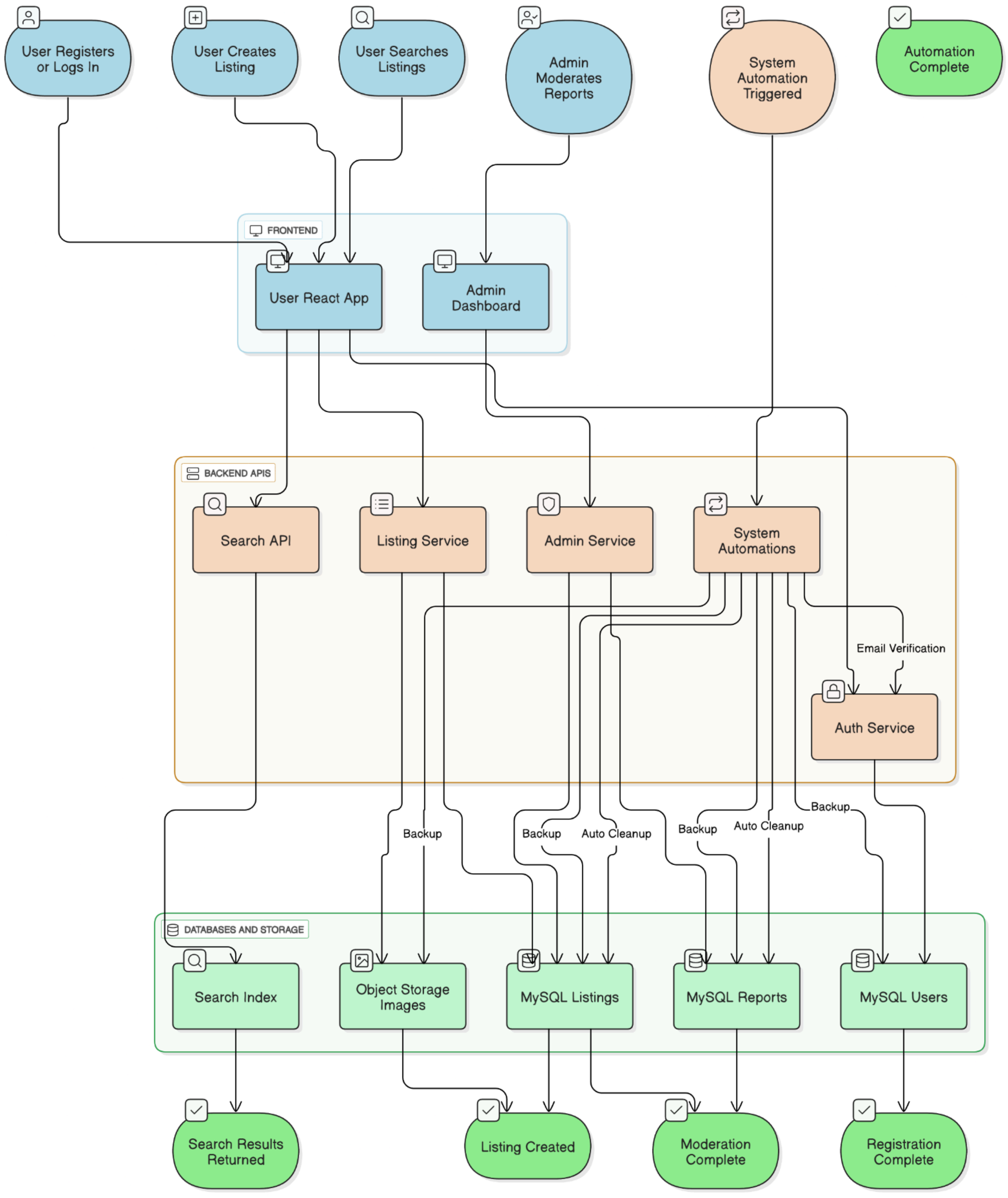
iv. Data Flow Diagram


USER INTERFACE
User
Interacts with
Displays data
React Single Page Application
Auth requests
Sends requests to
Listing requests
Search requests
Sends responses
BACKEND APIS
Authentication Service
Listing Service
Search API
Query/Update user data
Query/Update listings
Upload/Fetch images
Query search data
Reads/Writes data
DATA STORES
MySQL Users Database
MySQL Listings Database
Object Storage Images
Search Index

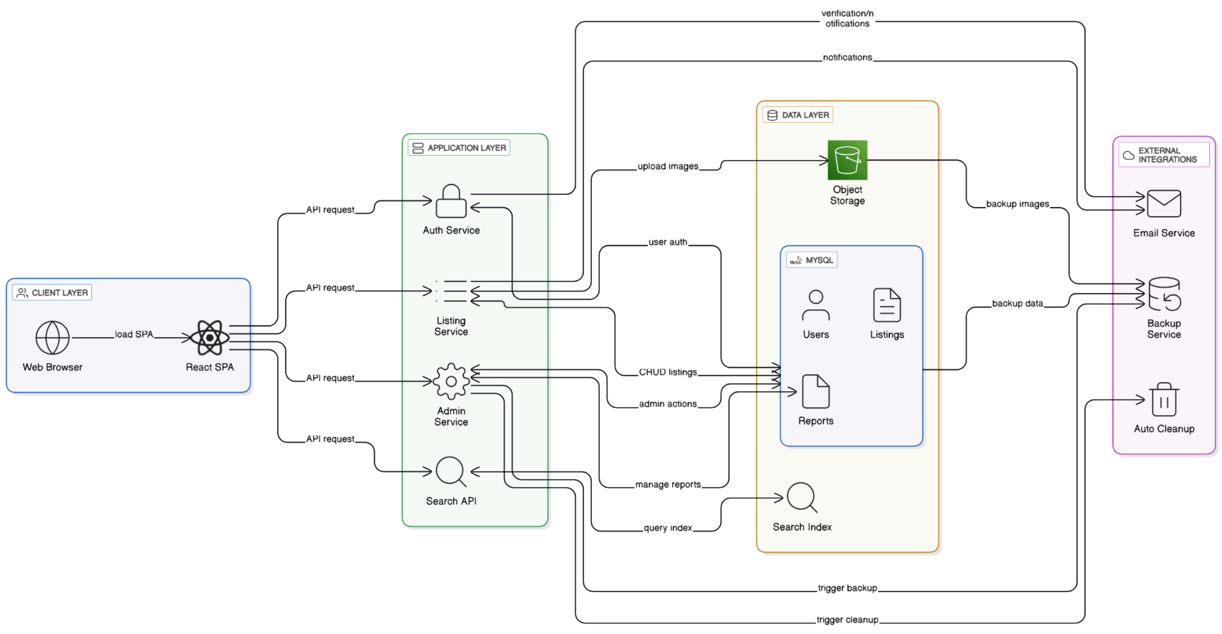
v. ER Diagram


users
name string
email string
passwordHash string
role string
createdAt timestamp
userID string pk
admins
adminID string pk
name string
email string
passwordHash string
permissions string
listings
title string
description string
price decimal
category string
createdAt timestamp
listingID string pk
userID string
images
imageID string pk
url string
listingID string
reports
reportID string pk
reason string
status string
createdAt timestamp
userID string
listingID string
adminID string
* *
* *
* *
1
1
1
1
1
1

vi. Technical Workflow Diagram



vii. Work Architecture Diagram



13. Quality: NFRs and Testing

13.1 Non-functional requirements

Metric	SLI (Service Level Indicator)	Target (SLO)	Measurement Method
Availability	Uptime %	≥ 99.0%	Continuous uptime monitoring (Pingdom / CloudWatch)
Latency	p95 response time	≤ 1000 ms	Application Performance Monitoring (APM: Datadog, New Relic)
Error Rate	5xx error percentage	≤ 1%	Centralized logging & alerts (ELK/CloudWatch Logs)

Security	Open critical CVEs in dependencies	0	Automated security scans (Snyk/OWASP Dependency Check)
Scalability	Concurrent users supported	500 baseline, scalable	Load tests (JMeter, k6)
Maintainability	Code quality score (SonarQube)	≥ B grade	Static code analysis
Usability	Task completion success rate	≥ 90% in usability tests	User testing & surveys
Backup/DR	RPO (Recovery Point Objective)	≤ 24 hours	Backup/restore test
	RTO (Recovery Time Objective)	≤ 4 hours	Disaster recovery drill

13.2 Test plan

Area	Type	Tools	Owner	Coverage Target	Exit Criteria
Backend	Unit Testing	Jest	Rohit	≥ 70% code coverage	No P1/P2 defects open

UI	End-to-End (E2E)	Playwright	Priya	≥ 60% user flows	Test pass rate ≥ 95%
API	Integration	Postman / Newman	Team	All scenarios covered	All critical test cases pass
Security	Vulnerability	OWASP ZAP, Snyk	Team	Top 10 risks checked	0 unresolved high/critical issues
Performance	Load/Stress	JMeter / k6	Team	Handle 500 users baseline	Meets latency & error SLOs
UAT	Acceptance	Manual scripts, Figma	Stakeholders	All core features validated	Stakeholder sign-off

13.3 Environments

- **Development (Dev)**
  - Purpose: Individual developer testing, rapid iteration.
  - Data: Seeded sample data.
  - CI runs unit + integration tests automatically.
- **Staging**
  - Purpose: Pre-production environment mirroring Prod.
  - Data: Anonymized copy of production dataset.
  - Runs full regression, E2E, load, and security tests.
  - Feature flags enabled for risky/experimental changes.
- **Production (Prod)**
  - Purpose: Live user-facing system.
  - Data: Real customer data with full monitoring.
  - Controlled rollouts with **blue/green** or **canary releases**.

    - Critical monitoring: uptime, latency, error rates.
    -
- **Change Management**
    - Flow: **Dev → Staging → Prod** with approvals and automated pipelines.
    - Rollback plan: Immediate revert via CI/CD if SLOs breached.

---

14. Security and Compliance

14.1 Threat model (STRIDE)

Asset	Threat	STRIDE Category	Impact	Likelihood	Mitigation	Owner
Auth tokens	Theft / replay attack	Spoofing	High	Medium	HTTPS/TLS, short TTL, refresh & rotation, secure storage (HttpOnly cookies)	Rohit
User data	SQL Injection	Tampering	High	Low	Parameterized queries, ORM usage, WAF rules, input validation	Rohit
Passwords	Brute-force / leaks	Information Disclosure	High	Medium	Strong hashing (bcrypt/argon2), rate-limiting, MFA	Priya

APIs	DoS / DDoS attacks	Denial of Service	High	Medium	Rate limiting, caching, auto-scaling, CDN, WAF	Team
Listings	Fake or malicious posts	Repudiation	Medium	Medium	Audit logs, moderation workflow, report/flag system	Priya
Backups	Unauthorized access	Elevation of Privilege	High	Low	Encrypted backups, role-based access control, periodic key rotation	Team

14.2 AuthN/AuthZ

14.2.1 Authentication (AuthN):

- Users register with email + password.
- Verification via email confirmation link before first login.
- Passwords stored using bcrypt with salted hashing.
- Session managed using JWT (short TTL, refresh token rotation).

14.2.2 Authorization (AuthZ):

- Roles:
    - Buyer → Browse, search, and purchase listings.
    - Seller → All buyer rights + create/edit/delete listings.
    - Admin → All rights + moderate listings, manage users, view audit logs.

- Role-based access control (RBAC) enforced at API layer (middleware).
- Feature flags for experimental or risky features tied to roles.

14.3 Audit and logging

- **Events logged:**
  - User signups and logins (success/failure).
  - Listing **CRUD actions** (create, edit, delete).
  - Abuse reports and moderation actions.
  - Security events (token expiry, failed auth attempts).
- **Retention:**
  - Logs stored for **90 days**, then rotated/purged.
- **Storage & Access:**
  - Centralized log store (e.g., ELK / CloudWatch).
  - Access restricted to **admins only**.
- **Compliance:**
  - Timestamped, immutable logs to support audits and investigations.

14.4 Compliance

- **Context:** Academic project, not for commercial deployment.
- **Policies:** Adhere to institute IT and data handling policies.
- **Data Sharing:** No third-party data sharing or external monetization.
- **PII Handling:** Store only minimal required data (email, name).
- **Retention:** Respect retention/deletion policies defined in Section 11.3.

---

15. Delivery and Operations

15.1 Release plan

- **Target Release:** v1.0 demo on **26 Oct 2025**.
- **Deployment Strategy:**
  - **Feature flags** to control rollout of new features (e.g., Search facets).
  - **Dark-launch search** to validate performance before full release.
- **Stages:**
  - **Dev → Staging → Prod** with CI/CD pipeline.
  - Canary testing for risky modules.
- **Post-release:** Monitor logs, uptime, and error rates for 48 hours; rollback option if critical defects

15.2 CI/CD and rollback

- **Continuous Integration (CI):**
  - Steps: **Lint → Unit/Integration Tests → Build → Dockerize**
  - Trigger: On every PR and merge to `main`
  - Tools: GitHub Actions / GitLab CI
- **Continuous Deployment (CD):**
  - Auto-deploy to **Staging** on merge to `main`
  - Manual approval for **Production** release
  - Feature flags for risky/incomplete features
- **Rollback Strategy:**
  - Keep **previous Docker image tags** in registry
  - Rollback by re-deploying last stable image version
  - Database migrations: apply with safe rollback plan (down scripts tested on staging)

15.3 Monitoring and alerting

Metric	Threshold	Alert To	Runbook

p95 latency	> 1200 ms	Tech Lead	“API Latency” runbook
Error rate (5xx)	> 2%	Tech Lead	“Error Spike” runbook

15.4 Runbooks

- **API Latency Runbook**
  - Check database indexes and query performance.
  - Scale application pods horizontally.
  - Revert recent deployment if latency persists.
- **Error Spike Runbook**
  - Inspect application and API logs.
  - Roll back to previous stable image if errors link to a change.
  - Record root cause and mitigation in incident notes.

15.5 Communication plan

- **Standups:** 15 min sync on **Mon/Wed/Fri** to review progress, blockers, and priorities.
- **Weekly Status Report:** Shared with mentor every **Friday** (progress, risks, next steps).
- **Bi-weekly Demo:** Walkthrough of new features and feedback collection.
- **Channels:**
  - **Slack/Teams** – daily communication & quick issue resolution.
  - **Email** – formal updates to mentor/faculty.
  - **GitHub Issues/Projects** – task tracking & documentation.

16. Risks and Mitigations

16.1 Risk heatmap

Risk	Probability	Impact	Score	Mitigation	Owner	Status
Schedule slip	Medium	High	12	Scope freeze, weekly demos	Aisha	Open
DB performance	Medium	Medium	9	Indexes, EXPLAIN, caching	Rohit	Open
Image storage costs	Low	Medium	6	Size limits, compression	Neeraj	Open

17. Research and Evaluation

- **Market review**: Studied OLX, Quikr, FB Marketplace for insights on onboarding flow, search performance, and trust-building mechanisms.
- **Evaluation plan**: Track KPIs defined in §8 on a weekly basis; conduct user survey post-release for qualitative feedback.
- **Limitations**: No integrated payment system, restricted to single-campus pilot, limited moderation capabilities.

18. Appendices

- **Glossary:** C2C, SSO, KPI, SLO, p95.

- **References:** Course handbook; React docs; Express docs; MySQL manual.