EduGame AI — Design Documents

1. **Contents:** 1) Executive summary 2) High-level architecture • 3) Detailed component design • 4) Data & infra design • 5) API & integration patterns • 6) Security & compliance • 7) UX principles & visual language • 8) Wireframes (text + annotated) • 9) Handoff artefacts & next steps

---

1. Executive summary

EduGame AI is an intelligent learning assistant for **Students** and **Teachers**. It provides: accurate academic answers, summarisation, quizzes, assignments, analytics, and two signature features — **"Explain My Mistake"** for students and **"AI Class Assistant — Auto Lesson Planner"** for teachers. The platform should be scalable, privacy-aware, and modular so AI modules can be improved independently.

Primary nonfunctional priorities: accuracy, responsiveness (<5–10s for most AI ops), reliability (SLA for classroom use), and data privacy.

---

2. High-level architecture

2.1 Architectural overview (3-tier + AI microservices)

- **Frontend**: Single Page App (React + TypeScript + Tailwind). Routes: `/student`, `/teacher`, `/auth`, `/admin`.
- **API Layer**: REST + WebSocket gateway (Node.js or FastAPI). Handles auth, routing, RBAC, and orchestration between services.
- **AI Microservices** (containerized):
  - Doubt-Solver Service (LLM orchestration + prompt manager)
  - Summarizer Service (PDF/text ingestion + summarization + flashcards)
  - Quiz Engine (question generation, difficulty scaling, auto-grading)
  - Lesson Planner (lesson generation, activities, predicted doubts)
  - Mistake Analysis (Explain My Mistake pipeline)
  - Evaluation Service (assignment auto-evaluation & rubric engine)
- **Data Layer**:
  - PostgreSQL (transactional)
  - pgvector (vector embeddings)
  - Object Store (S3-compatible) for uploaded PDFs, attachments
  - Redis (cache, session store, rate limiting)

- **Observability & Infra**:
  - Prometheus + Grafana (metrics)
  - ELK / Loki (logs)
  - CI/CD pipelines (GitHub Actions / GitLab CI)
  - Container orchestration: Kubernetes with GPU nodes or Docker Compose for small installs.

2.2 Deployment options

- **Cloud (managed)** — for general customers.
- **On-prem (school-hosted)** — for privacy-sensitive institutions.
- **Hybrid** — AI microservices in a private VPC/GPU cluster; frontend and API in a secure cloud.

---

3. Detailed component design

3.1 Frontend

- Tech: React + TypeScript + Tailwind CSS
- Pages:
  - Auth (Login/SSO)
  - Student Dashboard (`/student`)
  - Teacher Dashboard (`/teacher`)
  - Class view (per class)
  - AI Doubt Chat
  - Summaries / Uploads
  - Quiz Arena
  - Assignments
  - Lesson Planner Editor
- Real-time: WebSocket for live quizzes, leaderboards, and collaborative lesson edits.

3.2 API / Backend

- Gateway handles routing, RBAC (student/teacher/admin), rate limiting.
- Microservice communication via internal HTTP or gRPC (RAG heavy services may use gRPC).
- JWT / SSO session tokens; access tokens for service-to-service auth.
- Request flow example — **Explain My Mistake**:
  1. Frontend POST `/api/student/mistake-check` with student answer & problem context.
  2. API authenticates, writes job to queue, stores payload in object store.

3. Mistake Analysis service picks job, retrieves context, runs LLM with prompts + RAG (if needed).
4. Service returns JSON: {mistake_summary, correction_steps, corrected_solution, practice_questions[]}
5. API stores record in `mistake_checks` table and notifies frontend via WebSocket.

3.3 AI orchestration patterns

- **Prompt/versioning service**: central store of canonical prompts and versions for reproducibility.
- **RAG pipeline**: embed uploaded notes → store in pgvector → when generating answers include top-k relevant chunks.
- **Fallbacks**: If AI times out, return graceful "Try again / human-assisted" with logging for debugging.
- **Quality control**: confidence/scoring metadata for each AI output; flag low-confidence outputs for teacher review.

---

4. Data & infrastructure design

4.1 Database schema (summary)

- `users` (id, name, email, role)
- `organizations` (schools)
- `classes` (id, teacher_id, code, meta)
- `enrollments` (class_id, user_id)
- `quizzes`, `questions`, `options`, `quiz_attempts`, `question_attempts`
- `assignments`, `submissions`
- `lesson_plans` (auto-generated JSON + teacher edits)
- `mistake_checks` (student answer, analysis JSON)
- `summaries` (uploaded source, generated notes, flashcards)
- `embeddings` (pgvector vectors + metadata)
- `analytics` (aggregated snapshots)

(Using earlier DDL as reference — migrate to final schema in next steps)

4.2 Storage

- **Object Storage** for PDFs and media (S3-compatible)
- **Encrypted DB fields** via pgcrypto for sensitive text if required
- **Backups and retention**: daily backups for DB; retention policy configurable per Org.

4.3 Vector store

- Use `pgvector` (on Postgres) or external vector DB (Milvus, Pinecone) if scaling needed.
- Embedding dimension chosen per model (ex: 1536 or 768).

---

5. API & integration patterns

5.1 Representative endpoints

- `POST /api/auth/login` — login or SSO
- `GET /api/users/me` — profile
- `POST /api/classes` — create class
- `POST /api/classes/:id/join` — join by code
- `POST /api/student/doubt` — submit doubt (returns chat id)
- `POST /api/student/mistake-check` — Explain My Mistake
- `POST /api/summaries/upload` — upload PDF/notes
- `POST /api/teacher/lesson-plans/generate` — AI lesson generation
- `POST /api/teacher/quizzes` — generate quiz
- `GET /api/analytics/class/:id` — class analytics

5.2 Webhooks / Notifications

- Push notifications for assignment deadlines, new feedback, leaderboard changes via WebSocket or push service.

---

6. Security & compliance

6.1 Access control

- Role-based Access Control (RBAC): student / teacher / admin
- Principle of least privilege for DB and services.

6.2 Data protection

- TLS for transport
- Encryption at rest for sensitive fields (pgcrypto)
- Object store server-side encryption
- Audit logging for all AI outputs and accesses.

6.3 Privacy & policy

- Option for on-prem deployments for privacy-conscious institutions.
- Data retention policies configurable by admin (e.g., 12 months review).
- User data deletion and export endpoints to comply with local regulations.

---

7. UX principles & visual language

7.1 UX principles

- **Role-first UI**: Always detect user role and show relevant flows.
- **Clarity**: AI outputs should surface sources/citations and a confidence score.
- **Actionable**: Every AI output should suggest next steps (practice, review lesson, ask teacher).
- **Minimal friction**: Upload notes as drag-drop; single-click generate for lesson plans/quizzes.
- **Gamified micro-feedback**: XP, levels, streaks with clearly visible progression.

7.2 Visual language & components

- Clean, educational palette (soft blues, greens, high-contrast CTAs)
- Card-based layout for lessons, quizzes, and AI responses
- Components:
    - Chat bubble (Doubt Solver)
    - Document uploader (summaries)
    - Quiz card (start / resume / view results)
    - Lesson plan card (preview / edit / publish)
    - Leaderboard panel

---

8. Wireframes (textual + annotated)

Note: Use these as the base for designer to create high-fidelity mockups.

8.1 Student Dashboard — `/student` (Top-level)

```
+-----------------------------------------------------------+
| Header: EduGame AI  | Search | Notifications | Profile (S)  |
+-----------------------------------------------------------+
| Sidebar: Classes     | MAIN: Quick stats (XP, Level, Streak)|
| - Class A            | - Active Quizzes                     |
| - Class B            | - Recent AI Doubts                   |
```

```
| - Join class (code)  | - Upload notes (Summariser)         |
|                      | - Quick links: Explain My Mistake   |
+-----------------------------------------------------------+
| Bottom: Weekly leaderboard (horizontal cards)              |
+-----------------------------------------------------------+
```

Interactions: Click “Explain My Mistake” → modal: paste/upload answer + attach question → Submit → shows progress spinner → result card with sections: Mistake, Why wrong, Correct solution (steps), 1–3 practice Qs (Try button).

8.2 Doubt Solver (Chat)

```
[Chat header: Subject dropdown | Attach files]
-----------------------------------------------
| Chat history (infinite scroll)               |
| -> Student question (left)                   |
| -> AI answer with steps & small code blocks  |
|    [Show sources] [Add to flashcards]        |
-----------------------------------------------
[Input area] [Send] [Attach]
```

UX notes: Each AI answer shows small tags: confidence %, sources (clickable), “Explain step” toggles.

8.3 Lecture Summariser

```
[Upload area: Drag & Drop or paste notes]
-----------------------------------------
| After upload: Top section = Generated Summary |
| Key points (bullet list)                      |
| Flashcard preview (card carousel)             |
| Quick quiz: 3 sample MCQs                     |
| Buttons: Export PDF | Save to Class | Review  |
-----------------------------------------
```

8.4 Quiz Arena

```
[Filter: Subject | Topic | Difficulty | Mode (Timed / Practice)]
List of quizzes (card style)
Card: title | time | #qs | start button | leaderboard icon
Quiz UI: question panel | options | timer | submit
Results: score, explanation per question, XP gained
```

8.5 Teacher Dashboard — `/teacher`

```
+-------------------------------------------------------------+
| Header: EduGame AI (Teacher) | Notifications | Profile (T)   |
+-------------------------------------------------------------+
| Sidebar: Classes | Create Quiz | Lesson Planner | Analytics |
| Main: Class list + quick actions (Create Quiz, Assign HW)  |
```

```
+-------------------------------------------------------------+
| Lower: Analytics snapshot (avg score, weak topics, top students) |
+-------------------------------------------------------------+
```

8.6 Lesson Planner UI

```
[Top: Input: Topic | Duration | Grade Level | Key objectives]
[Generate] button
[Result area]
 - Day 1: Objectives | Activities (10-15 min) | Small quiz (2Qs)
 - Day 2: ...
Editable card for each day. Buttons: Save | Publish to class | Export PDF
```

UX note: Provide “Regenerate” for fine-tuning and an “Explain rationale” toggle showing prompt + context used.

---

9. Handoff artefacts & next steps

9.1 Artefacts to produce for engineering handoff

- OpenAPI specification for all endpoints (Swagger)
- Database migration scripts (SQL)
- Kubernetes deployment manifests (Helm charts)
- Prompt library (versioned)
- Sample dataset / seed data for QA
- High-fidelity mockups (Figma / Sketch)

9.2 Testing & QA

- Unit & integration tests for APIs
- Automated E2E tests for key flows (login, join class, submit mistake check, generate lesson plan)
- Manual UX testing with 3 teacher and 8 student participants (pilot)

9.3 Roadmap milestones (suggested)

- MVP (Core): Doubt Solver, Explain My Mistake, Summariser, Student & Teacher dashboards
- Phase 2: Lesson Planner, Quiz Arena, Assignment auto-evaluation, basic analytics
- Phase 3: Gamification refinements, advanced analytics, on-prem deployment option

---

10. Appendix — example prompt templates (brief)

- **Explain My Mistake (student math)**
  Prompt inputs: question statement, student answer, correct answer (if available). Instructions: identify incorrect steps, produce short bullet explanation for each mistake, provide corrected step-by-step solution, and generate 2 practice questions of similar difficulty with answers.
- **Lesson Planner**
  Prompt inputs: topic, grade level, duration. Instructions: produce day-wise objectives, activities with time allocations, quick quiz questions per day, expected student mistakes and suggestions to address them.