**Gamified Coding & AI Classroom System — Full Spec (Semester 1)**

**0) Scope**

* Designed for a **9th grade Coding & AI course**.
* Gamified layer + AI tutor + teacher console.
* Fully aligned to **Florida CPALMS standards**, with evidence tracking.
* Compliant with **FERPA-style privacy**, **WCAG 2.2 AA accessibility**, and robust enough for scale.

**1) Data Model (superset)**

**Core**

* **students**(id, first\_name, last\_name, display\_name, email, created\_at, accommodations\_jsonb)
* **classes**(id, name, term, timezone="America/New\_York", starts\_on, ends\_on)
* **enrollments**(id, class\_id FK, student\_id FK, role ENUM('student','teacher'), active BOOLEAN)
* **teams**(id, class\_id FK, name, created\_at)
* **team\_memberships**(id, team\_id FK, student\_id FK, joined\_at, left\_at NULLABLE)

**Curriculum & Assessment**

* **assignments**(id, class\_id, title, type ENUM('SOLO','TEAM','QUIZ','TEST','HOMEWORK','SHOWCASE'), base\_xp INT, due\_at, rubric\_json JSONB, standards\_json JSONB)
* **submissions**(id, assignment\_id FK, student\_id FK NULLABLE, team\_id FK NULLABLE, score\_pct NUMERIC(5,2), submitted\_at, metadata\_json JSONB)
* **peer\_reviews**(id, submission\_id FK, reviewer\_student\_id FK, rating INT 1–5, comment TEXT)
* **regrade\_requests**(id, submission\_id FK, student\_id FK, reason, status ENUM('OPEN','APPROVED','DENIED'), resolution\_notes, created\_at, resolved\_at)

**XP, Badges, Leaderboard**

* **xp\_events**(id, class\_id, student\_id, source ENUM('ASSIGNMENT','QUIZ','TEST','HOMEWORK','TEAM\_SHARE','PRACTICE','BONUS','ADJUSTMENT'), assignment\_id NULLABLE, points INT, meta JSONB, created\_at)
* **badges**(id, code UNIQUE, name, description, rule\_json JSONB)
* **student\_badges**(id, badge\_id FK, student\_id FK, awarded\_at, evidence JSONB)
* **weekly\_leaderboards**(id, class\_id, week\_start\_date, week\_end\_date, entries JSONB, created\_at)

**AI Tutor**

* **tutor\_policies**(id, class\_id, week\_no, mode ENUM('LEARN','ASSESS','OFF'), scope\_level ENUM('TIGHT','NORMAL','OPEN'), max\_tokens INT, rate\_per\_min INT, created\_by, updated\_at)
* **tutor\_allowed\_topics**(id, policy\_id FK, cpalms\_code, topic\_slug, description)
* **tutor\_blocklist**(id, policy\_id FK, pattern, reason, severity ENUM('HARD','SOFT'))
* **tutor\_tool\_whitelist**(id, policy\_id FK, tool\_code)
* **tutor\_queries**(id, class\_id, student\_id, policy\_id, ts, prompt\_text, resolved\_topics JSONB, decision ENUM('ALLOW','SOFT\_BLOCK','HARD\_BLOCK','ESCALATE'), reason, tokens\_in INT, tokens\_out INT)
* **tutor\_responses**(id, query\_id FK, content, assist\_level ENUM('HINT','SNIPPET','EXAMPLE','EXPLANATION'), redactions JSONB)
* **moderation\_queue**(id, class\_id, query\_id FK, status ENUM('OPEN','APPROVED','REJECTED'), reviewer\_id, notes, created\_at, resolved\_at)
* **lesson\_corpus**(id, class\_id, week\_no, title, content\_url, vector\_index\_id)

**Integrity, Accessibility, Privacy**

* **integrity\_flags**(id, submission\_id FK, type ENUM('PLAGIARISM','AI\_ABUSE','FRAUD','OTHER'), weight INT, details JSONB, detected\_at)
* **parent\_links**(id, student\_id, contact\_email, access\_token, scopes JSONB, expires\_at)
* **audit\_logs**(id, actor\_id, role, action, entity, before JSONB, after JSONB, ts)

**2) XP System**

**Base XP (with caps)**

* Solo starter projects: 50 XP ea
* Python projects: 50 XP ea
* Homework: 15 XP ea (cap 3/wk)
* Quizzes: 20 XP ea
* Mini-tests: 50 XP ea
* Team project (midterm): 100 XP
* Web projects: 40 XP ea
* Showcase participation: 20 XP
* Final team capstone: 200 XP
* Tutor practice micro-wins: 2–5 XP (cap 20/day, 60/wk)
* Peer feedback: 5 XP (cap 15/wk)
* Teacher bonus: up to +20 XP (audited)

**Formula (graded items)**

xp = base \* (score\_pct/100) \* timeliness\_mult \* early\_bonus \* creativity\_mult

* **Timeliness:** +10% on-time, –10%/day late, min 50%
* **Early bonus:** +5% if ≥24h early
* **Creativity bonus:** up to +10% (teacher input)

**Team split**

* 60% equal share
* 40% weighted by contribution (peer review + teacher logs)
* Fallback → equal share if insufficient data

**3) Badges**

* **First Builder**: first submission
* **Bug Slayer**: ≥10 bug fixes
* **Python Prodigy**: ≥90% on Python mini-test
* **AI Tinkerer**: completed AI integration project
* **Pixel Pioneer**: finished all Web projects
* **Team MVP**: top contributor on team project
* **Leaderboard Streak (3/6)**: Top-5 3 or 6 weeks straight
* **Final Boss**: ≥85% capstone + showcase

**4) Leaderboard**

* **Top-5 only** (no “last place” view)
* Shows: rank, XP, movement (UP/DOWN/SAME/NEW/RE-ENTRY), streak
* Snapshotted weekly (Sunday 23:59 ET)
* Tie-breakers: capstone XP > test XP > earlier first submission

**5) Draft Day**

* **Snake draft** with tier shuffling
* Balanced by XP tiering
* Constraints supported (avoid pairs, lock pairs, min/max size)
* Event log + rollback option
* UI: spin wheel or envelopes for suspense

**6) Rubrics & CPALMS**

* Each assignment has:
  + **Rubric JSON**: Functionality 40%, Code Quality 25%, Documentation 15%, Creativity 20%
  + **Standards JSON**: one or more CPALMS codes
* Evidence export: rubric scores + student artifact links + CPALMS alignment

**7) AI Tutor Oversight**

* **Modes:** OFF / LEARN / ASSESS
* **Scope levels:** TIGHT (week corpus only), NORMAL (semester corpus), OPEN (rare)
* **Allowlist/Blocklist/Whitelists:** config per week
* **Output filters:**
  + Snippets max 12 lines
  + Scaffold only, no full solutions for graded items
  + Concept-only in ASSESS mode
* **Moderation queue:** out-of-scope or flagged queries → teacher approval
* **Audit logs:** every query + decision stored
* **Alerts:** spike detection (too many tutor calls, repeated out-of-scope, plagiarism flags)

**8) Integrity Layer**

* **Plagiarism check:** code similarity %
* **AI abuse check:** repeated “full solution” requests → soft/hard blocks
* **Fraud flags:** XP anomalies (too many tutor events in burst)
* **Moderation log:** teacher can override or clear flags

**9) Accessibility & Inclusion**

* WCAG 2.2 AA compliance for UI
* IEP/504 accommodations → stored in accommodations\_jsonb
  + Extended deadlines
  + Relaxed rate caps
  + Simplified explanations
* ELL supports: bilingual glossaries, simplified text mode

**10) Privacy & Compliance**

* Encrypt PII at rest
* Tokenized parent/guardian portal: read-only badges, progress, showcase RSVP
* Data retention: purge/anonymize 12 months after term
* Teacher moderation logs non-editable (audit trail)

**11) Observability**

* Metrics: XP distribution (Gini coefficient), leaderboard churn, badge unlock rates, tutor query volume, late penalties
* Alerts: cron job failures, >5% tutor HARD\_BLOCK, suspicious XP bursts
* Dashboards: teacher view (class health), admin view (program health)

**12) API & Jobs**

* Endpoints for assignments, submissions, XP events, badges, leaderboards, tutor queries, policy edits, Draft Day, moderation
* Cron jobs: weekly leaderboard snapshot, badge awarder, plagiarism scan, standards coverage check

**13) Operational Resilience**

* All APIs idempotent
* Background jobs retry with exponential backoff
* Draft Day rollback supported
* Soft-delete on all entities

**14) Engagement Extras**

* **Class economy:** XP → redeem for perks (skip line, bonus hint packs)
* **Portfolio autogen:** deploy artifacts to GitHub Pages / student site
* **Community showcase:** external judges + community vote (fraud-resistant)
* **Shadow rank:** private progress rank for every student even if not Top-5

**✅ Result**

With these additions, you now have:

* **Core gamification (XP, badges, leaderboards, Draft Day).**
* **AI tutor with teacher oversight + scope enforcement.**
* **Integrity, fairness, privacy, accessibility, compliance safeguards.**
* **Operational + observability controls.**
* **Engagement extras (economy, showcase, portfolio).**

This is **classroom-ready, accreditation-friendly, parent-safe, and fun for kids**.

**AI Tutor Oversight & Scope Control — Tech Addendum**

**0) Goals**

* Tutor answers must stay **in-scope** (this week’s standards, lesson content, allowed tools).
* Teachers can **configure, pause, or narrow** the tutor at any time.
* Every interaction is **auditable, rate-limited, and policy-checked**.

**1) Data Model (additions)**

* **tutor\_policies**(id, class\_id, week\_no, mode ENUM('LEARN','ASSESS','OFF'), scope\_level ENUM('TIGHT','NORMAL','OPEN'), max\_tokens INT, rate\_per\_min INT, created\_by, updated\_at)
* **tutor\_allowed\_topics**(id, policy\_id FK, cpalms\_code, topic\_slug, description)  
  *e.g., “SC.912.CS-CP.1.2”, “loops”, “conditionals”*
* **tutor\_blocklist**(id, policy\_id FK, pattern, reason, severity ENUM('HARD','SOFT'))
* **tutor\_tool\_whitelist**(id, policy\_id FK, tool\_code)  
  *e.g., “PY\_SNIPPETS”, “HTML\_HELPER”, “NO\_FULL\_SOLUTION”*
* **tutor\_queries**(id, class\_id, student\_id, policy\_id, ts, prompt\_text, resolved\_topics JSONB, decision ENUM('ALLOW','SOFT\_BLOCK','HARD\_BLOCK','ESCALATE'), reason, tokens\_in INT, tokens\_out INT, latency\_ms INT)
* **tutor\_responses**(id, query\_id FK, content, assist\_level ENUM('HINT','SNIPPET','EXAMPLE','EXPLANATION'), redactions JSONB)
* **moderation\_queue**(id, class\_id, query\_id FK, status ENUM('OPEN','APPROVED','REJECTED'), reviewer\_id, notes, created\_at, resolved\_at)
* **lesson\_corpus**(id, class\_id, week\_no, title, content\_url, vector\_index\_id)  
  *(per-week RAG index for strict grounding)*

**2) Tutor Modes (teacher switch)**

* **OFF**: tutor unavailable; UI shows teacher note.
* **LEARN** (default): tutor gives **hints/snippets/examples**, never full graded solutions. Uses **TIGHT/NORMAL** scope.
* **ASSESS**: locked down; **no code generation**, no step-by-step for graded items; only definitions, rubric reminders, and references. Strict **TIGHT** scope.
* **Scope levels**
  + **TIGHT**: answers must cite (internally) a passage from **this week’s lesson\_corpus**; if not found → “OUT\_OF\_SCOPE”.
  + **NORMAL**: can use prior weeks’ corpus.
  + **OPEN**: (rare) allow broader CS basics but still enforce blocklist + tool whitelist.

Teacher toggles via console; changes apply immediately and are logged.

**3) Policy Layers (enforced in order)**

1. **Rate & size gate** → per-student rate\_per\_min, max\_tokens.
2. **Intent classifier** (fast text classifier or rules) → maps prompt to **topics** & **risk flags**.
3. **Scope check** → topics must intersect tutor\_allowed\_topics for current policy.
4. **RAG grounding** → retrieve from **lesson\_corpus** (week\_no). If **TIGHT**, require a match above threshold (e.g., cosine ≥ 0.35).
5. **Tool/Output controls** → allow **HINT/SNIPPET/EXAMPLE** only; block “full solution” patterns.
6. **Blocklist** → regex/glob or semantic patterns (e.g., “write the entire assignment X”).
7. **Assessment lock** → if mode='ASSESS' and assignment is active, refuse code, offer rubric-aligned guidance.
8. **Escalation** → if unsure, queue to **moderation\_queue**; show student: “Your question was sent to your teacher.”

All decisions logged to **tutor\_queries** with decision + reason.

**4) Prompt Contract (server-side, never editable by students)**

**System prompt (core excerpt):**

* “You are a classroom tutor constrained to **Week {{week\_no}}** for **Class {{class\_id}}**.  
  Only answer using **the provided Lesson Corpus** and **Allowed Topics**: {{topics}}.  
  Disallowed topics: {{blocklist}}.  
  Output policy:
  + Prefer **HINT** → **SNIPPET** → **EXAMPLE**, avoid full solutions.
  + If student asks for full code for graded tasks, reply with **scaffold + reasoning**, not the full final code.
  + In **ASSESS mode**, do not provide code; offer concepts, definitions, and rubric reminders only.
  + If out-of-scope, respond with: ‘That’s outside this week’s focus. Let’s connect it to {{allowed\_topics}}.’”

**Retriever prompt (RAG):** “Return top-k chunks strictly from lesson\_corpus for week {{week\_no}}; reject if confidence < threshold.”

**5) Out-of-Scope & Full-Solution Detection**

**Heuristics (fast path)**

* Regexes for “write my whole …”, “complete code”, “give entire solution”, “paste full program”, assignment titles, file names.
* Length limits: if user asks for > N lines of code, downgrade to SNIPPET.
* Known **assignment hash**: detect if prompt includes ≥ X% of the assignment text.

**Lightweight classifier (optional)**

* Distinguish **Conceptual**, **Hint Request**, **General Web**, **Full Solution Request**, **Off-topic**.
* If **Full Solution** or **Off-topic** → SOFT/HARD block.

**Responses**

* **SOFT\_BLOCK** (coach):  
  “I can’t provide the entire solution, but here’s how to structure it…” + scaffold + 10-line max snippet.
* **HARD\_BLOCK**:  
  “This goes beyond our current scope. Let’s stick to **{{allowed\_topics}}**.”

**6) Teacher Console (controls a real teacher needs)**

* **Mode switch**: OFF / LEARN / ASSESS, with scope slider (TIGHT/NORMAL/OPEN).
* **Topic allowlist**: add/remove CPALMS codes & tags per week.
* **Blocklist editor**: patterns + severity; import/export CSV.
* **Tool whitelist**: toggle “PY\_SNIPPETS”, “HTML\_HELPER”, “AI\_CONCEPTS\_ONLY”.
* **Rate limits**: per student (day/week); override by IEP/504 (from accommodations).
* **Live transcript & flags**: stream of queries; click “Approve/Reject/Reply” for queued items.
* **One-click “Lock Assignment”**: links an assignment id → switches to **ASSESS** for its duration.
* **Analytics**: out-of-scope attempts, blocks, top topics, struggling students (quiz <70%).

**7) API Endpoints (key ones)**

* PATCH /classes/:id/tutor-policy  
  Body: { mode, scope\_level, max\_tokens, rate\_per\_min }
* POST /classes/:id/tutor-policy/topics  
  Body: { week\_no, topics: [{cpalms\_code, topic\_slug}] }
* POST /classes/:id/tutor-policy/blocklist  
  Body: { patterns: [{pattern, reason, severity}] }
* POST /classes/:id/tutor-query *(gateway)*  
  Body: { studentId, text }  
  Returns: { decision, assist\_level, response, redactions }
* POST /classes/:id/moderation/:queryId/resolve  
  Body: { status, notes }
* POST /classes/:id/lesson-corpus/index  
  Body: { week\_no, files: [urls] } → builds/updates vector index

**8) Gateway Orchestration (pseudocode)**

async function handleTutorQuery(req) {

const { classId, studentId, text } = req.body;

enforceRateLimit(classId, studentId);

const policy = getActivePolicy(classId);

if (policy.mode === 'OFF') return deny("Tutor is currently unavailable.");

const intent = classifyIntent(text);

const topics = mapIntentToTopics(intent);

const allowed = topics.some(t => isAllowed(t, policy.allowedTopics));

if (!allowed) return softBlock("Out of scope for this week.", topicsHint(policy));

if (policy.mode === 'ASSESS') {

// No code; concept-only response

return explainConceptOnly(text, policy);

}

// Retrieve lesson-grounded context

const ctx = retrieveFromCorpus(classId, policy.week\_no, text);

if (policy.scope\_level === 'TIGHT' && ctx.conf < 0.35) {

return softBlock("Let’s stay within this week’s material.", topicsHint(policy));

}

// Blocklist & full-solution guards

if (matchesBlocklist(text, policy.blocklist)) return hardBlock("Blocked by class policy.");

if (isFullSolutionRequest(text)) return scaffoldOnly(text, ctx);

const response = await generateHintSnippet(text, ctx, policy.toolWhitelist);

return allow(response);

}

**9) Redaction & Output Filters**

* Enforce **line-count ceilings** for code in **LEARN** (e.g., max 12 lines/snippet).
* Strip/obfuscate **solution constants** (e.g., final answer keys) for graded prompts.
* Replace disallowed URLs, libraries, or APIs not on whitelist.

**10) Auditability & Alerts**

* Every decision → **tutor\_queries** with decision, reason, tokens, latency.
* **Alerts** (teacher email/UI): high out-of-scope rate, repeated HARD\_BLOCK for the same student, sudden spikes in tutor usage.
* **Playback**: per-student transcript with assist levels (HINT/SNIPPET/EXAMPLE) for grading context.

**11) Accessibility & Equity Hooks**

* Per-student accommodations.jsonb adjusts: rate limit, token max, reading level, bilingual glossaries.
* Tutor can auto-offer **simplified explanations** or **step-through mode** when quizzes <70%.

**12) CPALMS Linkage**

* tutor\_allowed\_topics entries **must include CPALMS codes**; the tutor’s answer template can print the code (“This aligns with SC.912.CS-CP.1.2: loops and conditionals”) to reinforce standards.

**13) Quick UI Copy (student-facing)**

* **ASSESS mode:** “Assessment mode is on: I can help with concepts and rubric expectations, not code.”
* **Out-of-scope:** “Great curiosity! For this week, let’s focus on {{allowed\_topics}}. Here’s how it connects…”
* **Full solution request:** “I can’t provide the entire solution, but here’s a scaffold + key steps to help you build it yourself.”

**14) Minimal Dev Checklist**

* Create tables and migrations (Section 1).
* Implement policy editor & toggles (Section 6).
* Build query gateway with policy pipeline (Section 8).
* Add RAG index per **lesson\_corpus** (Section 1) and enforce **TIGHT** mode.
* Output filters + code line caps (Section 9).
* Moderation queue & teacher resolve endpoints (Section 7).
* Logging, alerts, analytics (Sections 10, 6).