Study Tutor — User Roles & RBAC Snippets

This doc provides quick-drop code and config fragments to wire roles and permissions into the Study Tutor (AI + Web3) MVP. Copy–paste friendly for Node/Next + Postgres, but language-agnostic enough to port.

---

1) Roles

- **LEARNER** — studies, attempts quizzes, claims benefits; owns wallet address
- **TUTOR** — registers courses, sets checkpoints & rules, reviews analytics
- **BENEFITS_ADMIN** — manages trips/discounts, validates proofs, issues claim codes
- **SYSTEM** — internal service account used by the bot/orchestrator to call tools
- **PLATFORM_ADMIN** *(optional)* — manages users, role assignments, schema versions

> Principle: tutors drive pedagogy & rules; the bot is the engine that executes tool calls and persists state.

---

2) Database schema (Postgres/SQLite)

```
-- users table; wallet is primary identity
CREATE TABLE users (
  wallet TEXT PRIMARY KEY,
  display_name TEXT,
  email TEXT,
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- role assignments; composite PK for idempotency
CREATE TABLE user_roles (
  wallet TEXT NOT NULL REFERENCES users(wallet) ON DELETE CASCADE,
  role TEXT NOT NULL CHECK (role IN
('LEARNER','TUTOR','BENEFITS_ADMIN','SYSTEM','PLATFORM_ADMIN')),
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  PRIMARY KEY (wallet, role)
);

-- minimal course scaffolding
CREATE TABLE courses (
  course_id TEXT PRIMARY KEY,
  title TEXT NOT NULL,
  syllabus_url TEXT,
  version INTEGER DEFAULT 1,
  created_by TEXT REFERENCES users(wallet),
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

```sql
CREATE TABLE modules (
  course_id TEXT REFERENCES courses(course_id) ON DELETE CASCADE,
  module_id TEXT NOT NULL,
  passing_rule_json JSON NOT NULL,
  is_checkpoint BOOLEAN DEFAULT FALSE,
  PRIMARY KEY (course_id, module_id)
);

-- attempts + progress (as in main plan)
CREATE TABLE attempts (
  attempt_id TEXT PRIMARY KEY,
  wallet TEXT REFERENCES users(wallet),
  course_id TEXT NOT NULL,
  module_id TEXT NOT NULL,
  quiz_id TEXT,
  score_raw INTEGER NOT NULL,
  score_max INTEGER NOT NULL,
  duration_s INTEGER NOT NULL,
  passed BOOLEAN NOT NULL,
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  request_id TEXT UNIQUE
);

CREATE TABLE progress (
  wallet TEXT REFERENCES users(wallet),
  course_id TEXT,
  module_id TEXT,
  latest_attempt_id TEXT REFERENCES attempts(attempt_id),
  status TEXT CHECK (status IN
('NOT_STARTED','IN_PROGRESS','READY','BENEFIT_CLAIMED')),
  passed_at TIMESTAMP,
  version INTEGER DEFAULT 1,
  PRIMARY KEY (wallet, course_id, module_id)
);

-- benefit claims
CREATE TABLE benefit_claims (
  claim_code TEXT PRIMARY KEY,
  wallet TEXT REFERENCES users(wallet),
  benefit_id TEXT NOT NULL,
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

---

3) Role-aware JWT/SIWE session claim

```ts
// After SIWE verify, load roles, then mint a short-lived session token
interface SessionClaims {
  sub: string;            // wallet (0x...)
```

```
  roles: ('LEARNER'|'TUTOR'|'BENEFITS_ADMIN'|'SYSTEM'|'PLATFORM_ADMIN')[];
  iat: number; exp: number;
}
```

Header example

```
Authorization: Bearer <session-jwt>
X-Request-Id: 9bb9c3c8-a5a8-4d9d-9a0d-2f6a67c3b2e0
```

---

4) Permission matrix (tools)

Tool	LEARNER	TUTOR	BENEFITS_ADMIN	SYSTEM	PLATFORM_ADMIN
register_course	✖	✔	✖	✖	✔
generate_quiz	✖*	✔	✖	✔	✔
score_attempt	✔	✔	✖	✔	✔
update_progress	✔	✖	✖	✔	✔
issue_attestation	✖	✔	✖	✔	✔
mint_badge_sbt	✖	✔	✖	✔	✔
grant_benefit	✖	✖	✔	✔	✔

*Learner cannot directly call `generate_quiz`; the bot (SYSTEM) invokes it based on syllabus rules.

---

5) Middleware (RBAC + Idempotency)

```
// rbac.ts
export function requireRole(...allowed: string[]) {
  return (req, res, next) => {
    const roles: string[] = req.auth?.roles || [];
    if (!allowed.some(r => roles.includes(r))) return
res.status(403).json({error:'forbidden'});
    next();
  };
}

// idempotency.ts
import crypto from 'crypto';
export async function withIdempotency(handler) {
  return async (req, res) => {
    const key = req.headers['x-request-id'];
```

```
    if (!key) return res.status(400).json({error:'missing X-Request-Id'});
    const hit = await idempotencyStore.get(key);
    if (hit) return res.status(200).json(hit);
    const out = await handler(req, res, {requestId: key});
    await idempotencyStore.set(key, out);
    return res.status(200).json(out);
  };
}
```

Usage

```
app.post('/tools/register_course', requireRole('TUTOR','PLATFORM_ADMIN'),
withIdempotency(registerCourse));
app.post('/tools/generate_quiz',
requireRole('SYSTEM','TUTOR','PLATFORM_ADMIN'),
withIdempotency(generateQuiz));
app.post('/tools/score_attempt',
requireRole('LEARNER','SYSTEM','TUTOR','PLATFORM_ADMIN'),
withIdempotency(scoreAttempt));
app.post('/tools/
update_progress',requireRole('LEARNER','SYSTEM','PLATFORM_ADMIN'),
withIdempotency(updateProgress));
app.post('/tools/
issue_attestation',requireRole('SYSTEM','TUTOR','PLATFORM_ADMIN'),
withIdempotency(issueAttestation));
app.post('/tools/mint_badge_sbt',
requireRole('SYSTEM','TUTOR','PLATFORM_ADMIN'),
withIdempotency(mintBadgeSbt));
app.post('/tools/grant_benefit',
requireRole('BENEFITS_ADMIN','SYSTEM','PLATFORM_ADMIN'),
withIdempotency(grantBenefit));
```

---

6) Example payloads by role

TUTOR → register_course

```
{
  "request_id": "a5d7...",
  "course_id": "MATH101",
  "title": "Intro Math",
  "syllabus_url": "https://.../slides.pdf",
  "checkpoints": [
    { "module_id": "readiness", "passingRule": {"minScore": 8, "maxTime":
180}, "is_checkpoint": true },
    { "module_id": "module1", "passingRule": {"minScore": 7, "maxTime":
240}, "is_checkpoint": false }
```

```
  ]
}
```

SYSTEM → generate_quiz

```
{ "request_id": "b1c2...", "course_id": "MATH101", "module_id": "readiness",
"count": 10, "difficulty": "EASY" }
```

LEARNER/SYSTEM → score_attempt

```
{
  "request_id": "c9e3...",
  "quiz_id": "QZ-001",
  "answers": [{"id":"Q1","value":"YES"}, {"id":"Q2","value":"NO"}],
  "started_at": 1695632000,
  "submitted_at": 1695632150
}
```

SYSTEM/LEARNER → update_progress

```
{
  "request_id": "d0f4...",
  "learner_addr": "0xABC...",
  "course_id": "MATH101",
  "module_id": "readiness",
  "attempt_id": "ATT-001",
  "score_raw": 9,
  "score_max": 10,
  "duration_s": 150,
  "passed": true
}
```

SYSTEM/TUTOR → issue_attestation

```
{
  "request_id": "e1a5...",
  "learner_addr": "0xABC...",
  "course_id": "MATH101",
  "module_id": "readiness",
  "score_pct": 90,
  "passed_at": 1695632150
}
```

BENEFITS_ADMIN/SYSTEM → grant_benefit

```
{ "request_id": "f2b6...", "learner_addr": "0xABC...", "benefit_id":
"TRIP-2025-UG-001" }
```

---

7) UI visibility rules (client-side)

- **LEARNER**: show “Start Quiz,” “Resume,” “My Achievements,” “Claim Benefit”
- **TUTOR**: show “Create Course,” “Checkpoints,” “Question Banks,” “Analytics”
- **BENEFITS_ADMIN**: show “Validate Proof,” “Issue Claim Code,” “Redemptions”
- **PLATFORM_ADMIN**: show “User Roles,” “System Health,” “Schemas”

```
function can(roleSet: string[], action: string): boolean {
  const policy = {
    START_QUIZ: ['LEARNER'],
    REGISTER_COURSE: ['TUTOR','PLATFORM_ADMIN'],
    ISSUE_ATTESTATION: ['TUTOR','SYSTEM','PLATFORM_ADMIN'],
    GRANT_BENEFIT: ['BENEFITS_ADMIN','SYSTEM','PLATFORM_ADMIN']
  } as Record<string,string[]>;
  return (policy[action]||[]).some(r => roleSet.includes(r));
}
```

---

8) Seed script

```
INSERT INTO users(wallet, display_name) VALUES
  ('0xLEARNER', 'Learner One'),
  ('0xTUTOR', 'Tutor T'),
  ('0xBENEFIT', 'Benefits B'),
  ('0xSYSTEM', 'System Bot');

INSERT INTO user_roles(wallet, role) VALUES
  ('0xLEARNER','LEARNER'),
  ('0xTUTOR','TUTOR'),
  ('0xBENEFIT','BENEFITS_ADMIN'),
  ('0xSYSTEM','SYSTEM');
```

---

9) API error model

```
{ "error": "forbidden", "required_roles": ["TUTOR"], "request_id": "..." }
```

---

10) Audit & observability

- Log `wallet`, `roles`, `tool`, `request_id`, `latency_ms`, `status`
- Keep a separate `rbac_denies` counter by tool and role to surface UX/permissions friction

---

11) Test cases (quick hits)

1. **Learner tries register_course → 403**
2. **Tutor registers course → 200; module rows exist**
3. **System generates quiz → 200; learner can't**
4. **Learner scores attempt & updates progress → 200; state moves to READY when rule met**
5. **Benefits admin grants benefit with no READY → 403; with READY → 200 + claim_code**

---

12) Notes

- Prefer server-side checks; client gating is only UX.
- Keep `SYSTEM` scoped to internal calls (mTLS or signed service token).
- Idempotency is non-negotiable for all mutating tools.