Academce — End-to-End Project Guide (Presenter Notes)

This guide explains what the app does, how it works internally, and talking points for your presentation. It is written for quick reading and can be exported to PDF.

---

1) Product Overview
- **What it is:** A learning dashboard where students generate AI-powered courses, study lessons, take quizzes, track progress, earn points, and maintain learning streaks. A friendly chatbot helps with study questions.
- **Key features:**
  - AI course generation (Gemini) + quick template fallback
  - Lessons with points + completion tracking
  - Quizzes with scoring and points
  - Daily activity log with current/longest streaks
  - Personalized recommendations (AI)
  - JWT authentication; protected APIs

---

2) Architecture (High Level)
- **Frontend:** React + Vite; routes for Dashboard, Courses, Lessons, Quiz, Auth, Chatbot.
- **Backend:** Node + Express with modular routes; MongoDB via Mongoose.
- **Database:** MongoDB Atlas (cloud). Collections: `users`, `courses`, `lessons`, `quizzes`.
- **AI:** Google Gemini 2.5-flash via SDK — two models:
  - `getCourseGenerationModel()` for JSON-structured generation
  - `getChatbotModel()` for conversational replies

---

3) Data Model (Essential Fields)
- **User** (`models/User.js` — simplified):
  - `email`, `password(hash)`, `name`, `theme`
  - `totalPoints` (int)
  - Streak fields: `currentStreak`, `longestStreak`, `lastActivityDate`, `activityLog[{date, lessonsCompleted}]`
- **Course**:

  - `userId`, `title`, `topic`, `description`, `difficulty`, `estimatedDuration`, `learningObjectives[]`
  - `totalLessons`, `completedLessons`
- **Lesson**:
  - `courseId`, `title`, `content(HTML/Markdown)`, `order`, `points`, `duration`, `videoSearchTerm`, `completed`, `completedBy[userId[]]`
- **Quiz**:
  - `courseId`, `title`, `description`, `questions[{ques, options[], correctAnswer, explanation}]`, `score`, `attempts[{userId, score, answers[], completedAt}]`

---

4) Authentication & Authorization
- **Registration (`POST /api/users/register`)**
  - Hashes password with bcrypt, stores user, returns a JWT.
- **Login (`POST /api/users/login`)**
  - Validates password, returns JWT.
- **JWT Middleware (`middleware/auth.js`)**
  - Reads `Authorization: Bearer <token>`
  - Verifies with `JWT_SECRET`
  - Sets `req.user.userId` (normalized) and rejects invalid/expired tokens.
- **Protected routes** use `authenticateToken` — e.g., lessons, quizzes, course generation.

Talking point: “All sensitive endpoints require a valid JWT; the middleware normalizes the user id and blocks unauthorized access.”

---

5) AI Course Generation — Flow & Resilience
- **Endpoint:** `POST /api/generate-course` (protected)
- **Inputs:** `{ topic, difficulty, numberOfLessons }`
- **Prompting strategy:**
  - Asks for strictly JSON output only
  - Lesson content sized to 200–300 words for speed
  - 3–5 quiz questions
- **Retries & Parsing:**
  - Retries up to 3 times on overload/429 with exponential backoff

  - Cleans model response: strips markdown blocks, scans for a balanced JSON object by counting braces, normalizes quotes, removes trailing commas, converts stray line breaks
  - Attempts JSON.parse twice with fallbacks
- **Persistence:**
  - Creates `Course` → bulk creates `Lessons` → creates `Quiz`
  - Returns created resources
- **Quick Fallback:** `POST /api/generate-course/quick`
  - Bypasses AI by generating a professional template instantly using the requested topic

Talking point: “We hardened the AI parsing with a balanced-brace scanner and sanitizers, and added a quick non-AI fallback so the feature is demo-reliable.”

---

6) Points, Progress, and Streaks — Algorithms
Lesson Completion (`PATCH /api/lessons/:id/complete`)
- Input: `{ isCompleting: true|false }`
- Logic when marking complete:
  - Add user to `lesson.completedBy`; mark `completed = true` if needed
  - Increment `course.completedLessons`
  - Add `+5` to `user.totalPoints`
  - Update streaks via daily activity log:
    1. Normalize today to midnight
    2. If today already has activity: increment `lessonsCompleted` only (streak unchanged)
    3. If first activity today:
       - If `lastActivityDate` was yesterday → `currentStreak++`
       - Else → `currentStreak = 1`
       - Update `longestStreak = max(longestStreak, currentStreak)`
       - Set `lastActivityDate = today`
- Logic when unmarking complete:
  - Remove user from `completedBy`; adjust course `completedLessons`
  - Deduct `-5` points (not below 0)
  - Decrement today’s `lessonsCompleted` if applicable (doesn’t retroactively change streak)

Pseudocode:
```
```

```
if isCompleting and not alreadyCompleted:
  user.totalPoints += 5
  if no activity logged today:
    if lastActivityDate == yesterday: currentStreak += 1
    else: currentStreak = 1
    longestStreak = max(longestStreak, currentStreak)
    lastActivityDate = today
  else:
    // streak unchanged, only lessonsCompleted++
```

Quiz Submission (`POST /api/quizzes/:id/submit`)
- Scores answers; saves attempt with per-question correctness
- Awards **+20 points** on user's first completion of that quiz
- If this is the first activity today, logs the day and applies the same streak rules (quiz doesn't count as a lesson but does maintain activity)

Talking point: "Same-day multiple completions don't inflate streaks; only the first activity of a day moves the streak forward."

---

7) Chatbot & Recommendations
- **Chatbot endpoint:** `POST /api/chat`
  - Builds a context-rich prompt: user name, email, enrolled courses with difficulty, progress and durations, plus short conversation history
  - Uses `getChatbotModel()` to generate a concise, helpful reply
- **Recommendations endpoint:** `GET /api/recommendations/:userId`
  - Summarizes the user's courses (progress %) and asks Gemini for 3 next-step courses
  - Returns strict JSON; includes a safe fallback set if parsing fails

Talking point: "We embed real user/course context into the prompts so answers feel personalized."

---

8) Frontend Highlights (brief)
- Auth screens with detailed validation
- Dashboard shows points, streaks and recent activity

- Courses page lists AI/quick-generated courses
- Lesson view supports completion toggle and point awards
- Quiz view submits answers and displays score
- Chatbot floating assistant for quick help

---

9) Performance & Reliability Improvements
- Reduced lesson and quiz sizes for faster AI responses (5–10s typical)
- Hardened JSON parsing (balanced braces; sanitization; multi-attempt parsing)
- Added instant quick-template generation
- Logging to `debug.log` for traceability

---

10) Deployment & Data
- **Database:** MongoDB Atlas (cloud). You can also view the same data via Compass connected to your Atlas URI.
- **Viewing in Atlas:** Go to your cluster → click **Browse Collections** → database `acadence_db` (collections: users, courses, lessons, quizzes). If nothing appears, whitelist your current IP in Atlas Network Access and refresh.
- **Environment:** `.env` holds `MONGO_URI`, `GEMINI_API_KEY`, `JWT_SECRET` (values must be set; never commit real secrets publicly).

---

11) API Cheat-Sheet (for demo)
- Auth: `POST /api/users/register`, `POST /api/users/login`
- Profile: `GET /api/users/profile`, `PUT /api/users/profile`
- Generate course: `POST /api/generate-course` or `POST /api/generate-course/quick`
- Lessons: `GET /api/lessons/course/:courseId`, `PATCH /api/lessons/:id/complete`
- Quizzes: `GET /api/quizzes/course/:courseId`, `POST /api/quizzes/:id/submit`
- Chatbot: `POST /api/chat`
- Recommendations: `GET /api/recommendations/:userId`

---

12) Demo Script (3–5 minutes)

1. Log in (show JWT stored in app state)
2. Generate a course (use Quick first for speed; then show AI flow if time)
3. Open the course → complete a lesson → points +5; streak logic explained
4. Take the quiz → submit → points +20
5. Show Dashboard: current points and streak updated
6. Open Chatbot: ask something course-related → show personalized reply
7. In Atlas: click **Browse Collections** → show updated docs

---

13) Q&A Talking Points
- **Why JWT?** Simple, stateless, works well with SPA; protects APIs.
- **How are streaks accurate?** We normalize dates to midnight and only advance streak on the first activity of the day; additional completions don't inflate streak.
- **How do you handle flaky AI JSON?** Balanced-brace extraction + quote/trailing-comma fixes + multi-attempt parse; plus a no-AI quick fallback.
- **Data model choices?** Split Course/Lesson/Quiz enables scalable content and per-user progress.
- **Security?** Passwords hashed (bcrypt), JWT secret, auth middleware, quiz answers not leaked in the GET endpoint.

---

14) How to Export to PDF
- Option A: Open `docs/Acadence_Project_Guide.html` in a browser → press `Ctrl+P` → Destination: **Save as PDF** → Save.
- Option B: Open this Markdown in VS Code with a Markdown Preview to PDF extension and export.

Prepared for: Acadence Presentation