Module 7: Agile & Scrum Workshop + Module 8: Capstone Project Sprint (14 Hours)

> *Module 7 (2 hrs): Agile/Scrum theory and setup — shared across all tracks.*

> *Module 8 (12 hrs): Full-stack project built in real Agile sprints — each track uses its own stack.*

Module 7: Agile & Scrum Workshop (2 Hours)

Session AG1 (2 hrs) — Agile Fundamentals & Sprint Setup

Learning Objective: Understand Agile/Scrum well enough to run sprints during the capstone.

Topics:

- Why Agile? — waterfall vs agile, the manifesto, iterative delivery
- Scrum framework — sprints (1-2 week cycles), inspect & adapt
- Scrum roles:
  - **Product Owner** — defines what to build (trainer plays this role)
  - **Scrum Master** — removes blockers, facilitates ceremonies (rotating student role)
  - **Developers** — the team building the product
- Scrum ceremonies:
  - **Sprint Planning** — what will we build this sprint? how much can we commit to?
  - **Daily Standup** — 5 min: what I did, what I'll do, any blockers
  - **Sprint Review** — demo working software to stakeholders
  - **Sprint Retrospective** — what went well, what to improve, action items
- Scrum artifacts:
  - **Product Backlog** — ordered list of everything the product needs
  - **Sprint Backlog** — items committed to this sprint
  - **Increment** — the working software at end of sprint
- User Stories — format: "As a [role], I want [feature], so that [benefit]"
- Acceptance criteria — how do we know a story is done?
- Estimation — story points, planning poker (Fibonacci: 1, 2, 3, 5, 8, 13)
- Definition of Done — code complete, tested, reviewed, merged, documented
- Kanban boards — columns: Backlog → To Do → In Progress → In Review → Done

Hands-on:

- Write 10 user stories for their chosen capstone project
- Add acceptance criteria to each story
- Estimate stories using planning poker
- Set up GitHub Projects board with proper columns
- Create GitHub Issues from user stories
- Assign a rotating Scrum Master for the project sessions
- Define team's Definition of Done

Deliverable: GitHub Projects board populated with estimated, prioritized user stories

---

Module 8: Capstone Project Sprint (12 Hours)

Project Options (Teams Pick One)

Option A: HealthConnect — Clinic Management System

- **Roles:** Admin, Doctor, Patient
- **Features:** Appointment booking, patient records, doctor schedules, prescription tracking
- **Dashboard:** Today's appointments, doctor availability, patient stats
- **AI Feature:** AI-powered symptom checker or appointment summary generator

Option B: TaskForge — Project Management Tool

- **Roles:** Admin, Project Manager, Developer
- **Features:** Project creation, task boards (Kanban), assignment, status tracking, comments
- **Dashboard:** Tasks by status, team workload, project progress
- **AI Feature:** AI task description generator or smart task prioritization

Option C: EduTrack — Student Performance Tracker

- **Roles:** Admin, Instructor, Student
- **Features:** Course management, assignment submission, grading, attendance
- **Dashboard:** Grade distributions, attendance trends, at-risk students
- **AI Feature:** AI-powered study plan generator or performance prediction

---

Technical Requirements (All Projects)

Backend

- Minimum 3 database tables with relationships (PostgreSQL)
- Audit logs / activity feed stored in PostgreSQL (JSONB column or dedicated table)
- JWT authentication with role-based access
- Minimum 8 REST API endpoints (CRUD + at least 2 custom queries)
- Input validation on all endpoints
- Proper error handling with meaningful messages
- At least 3 unit tests

Frontend

- Minimum 5 pages/views with routing
- Login/register with JWT flow
- Protected routes (redirect to login if not authenticated)
- CRUD interface for the main resource
- Dashboard with at least 1 chart

- Responsive design (works on mobile)
- Loading states and error handling

AI Integration

- At least 1 AI-powered feature using OpenAI API
- Proper prompt engineering (system prompt + user context)
- Error handling for API failures

Git & Agile (Enforced)

- **Feature branches** — no committing directly to `main`
- **Pull requests** for every feature — at least 1 teammate must review before merge
- **GitHub Projects board** — all tasks tracked, moved across columns during sessions
- **Meaningful commits** — not "fix", "update", "final"
- Minimum 15 commits showing progressive development
- README.md with: project description, tech stack, setup instructions, screenshots

---

Sprint Structure

The 12-hour capstone runs as **2 sprints**:

- **Sprint 1 (Sessions P1-P3, 7 hrs):** Backend + Frontend MVP
- **Sprint 2 (Sessions P4-P5, 5 hrs):** AI feature, dashboard, polish, demo

Every session starts with a **5-minute standup** — each team member answers:

- What did I do since last session?
- What will I do this session?
- Is anything blocking me?

Session P1 (2 hrs) — Sprint 1 Planning & Project Kickoff

Standup: N/A (first session)

Sprint Planning (30 min):

- Review the product backlog (created in AG1)
- Select stories for Sprint 1 — focus on: auth, backend CRUD, basic frontend
- Break stories into tasks on GitHub Projects board
- Assign tasks to team members
- Agree on Sprint 1 goal: "Working backend API with auth + basic frontend connected"

Development (90 min):

- Set up project repositories (backend + frontend)
- Create database schema and run migrations
- Implement user registration and login with JWT
- Start main resource CRUD API (e.g., patients, tasks, students)
- Each developer works on a **feature branch**

- First PRs submitted — teammate reviews and merges

Deliverable: Project skeleton running, auth working, at least 2 PRs merged

---

Session P2 (3 hrs) — Sprint 1: Backend & Database

Standup (5 min): What I did, what I'll do, blockers

Development (2 hrs 55 min):

- Complete CRUD API for main resource
- Add secondary resource with relationship (e.g., appointments → doctor, tasks → project)
- At least 1 custom query endpoint (e.g., "appointments by doctor", "tasks by status")
- Implement audit logging — log every create/update/delete action to PostgreSQL
- Write 3 unit tests for service layer
- Test all endpoints in Postman/Swagger
- Continue feature-branch + PR workflow — **every feature is a PR**
- Code review: each team member reviews at least 1 PR from a teammate

Mid-session check-in: Trainer walks through each team — quick blocker resolution

Deliverable: Backend API complete with auth, CRUD, relationships, audit logging, all PRs merged

---

Session P3 (2 hrs) — Sprint 1: Frontend & Integration

Standup (5 min): What I did, what I'll do, blockers

Development (1 hr 45 min):

- Set up frontend project with routing and layout (header, sidebar, content area)
- Implement login/register pages with JWT auth flow
- Build list view for main resource (with search/filter)
- Build create/edit form with validation
- Connect all pages to backend API via HTTP client
- Add loading spinners and error messages
- Continue PR workflow

Sprint 1 Review (10 min):

- Quick demo to trainer — show working backend + frontend
- Check: auth works? CRUD works? Data flows end-to-end?
- Note gaps for Sprint 2

Deliverable: Working full-stack MVP — auth, CRUD, and basic UI connected to backend

---

Session P4 (3 hrs) — Sprint 2: AI Feature, Dashboard & Polish

Sprint 2 Planning (15 min):

- Review what's left in the backlog
- Plan Sprint 2: dashboard, AI feature, UI polish, README
- Assign remaining tasks

Standup (5 min): What I did, what I'll do, blockers

Development (2 hrs 40 min):

- Add dashboard page with at least 1 chart (Chart.js, Recharts, or ng2-charts)
- Implement AI feature:
  - Backend: new endpoint that calls OpenAI API with proper system prompt
  - Frontend: button/section that triggers AI and displays result
  - Examples: "AI Summary" button, smart search, recommendation engine
- Fix integration issues, edge cases, error handling
- Polish UI — consistent styling, responsive checks
- Update README with:
  - Project description and tech stack
  - Screenshots of key screens
  - Setup and run instructions
  - Team member names
- Clean up code — remove console logs, unused imports, dead code
- Final PRs reviewed and merged

Deliverable: Complete app with dashboard, AI feature, polished UI, clean README

Session P5 (2 hrs) — Sprint Review, Demo & Retrospective

Sprint 2 Review / Demo (70 min — ~10 min per team):

Each team presents to the class (trainer acts as "stakeholder"):

- **Problem statement** (1 min) — what does the app solve?
- **Tech stack overview** (1 min) — backend, frontend, database, AI
- **Live demo** (5 min) — walk through key features:
  - Login flow
  - CRUD operations
  - Dashboard with charts
  - AI feature in action
- **Show GitHub** (2 min):
  - Repository structure and README
  - Commit history and PR workflow
  - GitHub Projects board — stories moved to Done
- **Q&A** (1 min) — peers and trainer ask questions

Peer Feedback (10 min):

- Each team gives 1 positive comment and 1 suggestion to another team
- Trainer provides code quality and architecture feedback

Sprint Retrospective (30 min):
Each team discusses:

- **What went well?** — things that worked, good team dynamics, tools that helped
- **What didn't go well?** — blockers, miscommunication, time pressure
- **What would we do differently?** — process improvements, technical decisions
- Each team shares their top insight with the class

Final 10 min:

- Trainer wraps up — key lessons from the project experience
- Advice for interviews — how to talk about this project, the Agile process, the AI feature
- Reminder: keep the GitHub repo clean, it's part of their portfolio now

Deliverable: Completed demo, peer feedback received, retrospective insights documented

---

Evaluation Criteria

Criteria	Weight	What We Look For
Working Application	25%	Does it run? Auth works? CRUD works? AI feature works?
Code Quality	15%	Clean code, proper patterns, service layer, error handling
Database Design	10%	Proper schema, relationships, indexes, audit logging
AI Integration	15%	Meaningful AI feature, good prompt engineering
Agile Process	15%	Standups happened, board used, stories estimated, retro done
Git Workflow	10%	Feature branches, PRs with reviews, meaningful commits, clean repo
Presentation	10%	Clear demo, confidence, handles questions

What Students Walk Away With

After this module, every student has:

- A **deployed full-stack application** they built as a team
- A **GitHub repository** that looks professional — PRs, clean commits, good README
- A **professional GitHub profile** — bio, pinned repos, contribution graph, portfolio-ready
- An **ATS-qualified resume** — formatted for applicant tracking systems, with quantifiable project experience
- An **updated Naukri profile** — headline, skills, project descriptions optimized for recruiter searches
- Real **Agile/Scrum experience** — sprint planning, standups, reviews, retrospectives
- **Code review experience** — reviewing teammates' PRs, giving and receiving feedback
- A **working AI feature** they can discuss in interviews

- **Demo experience** — presenting technical work to stakeholders
- Understanding of **how real software teams work** — not just coding, but process
- Something concrete for their **resume** and **interviews**

Agile Cheat Sheet for Students

Daily Standup Template

> *"Yesterday I [completed X]. Today I will [work on Y]. [No blockers / I'm blocked by Z]."*

User Story Template

> *"As a [role], I want to [action], so that [benefit]."*

> *Acceptance Criteria: Given [context], when [action], then [expected result].*

Definition of Done

- [ ] Code complete and compiles
- [ ] Unit tests pass
- [ ] PR created with description
- [ ] At least 1 teammate reviewed
- [ ] PR merged to main
- [ ] GitHub Projects board updated
- [ ] No console logs or dead code