Development Plan
FitCoachAR

Seyedmohamad Mirhosseininejad

Table 1: Revision History

Date	**Developer(s)**	**Change**
Apr 20, 2026	Mirhoseininejad	Consolidated Development Plan documenting the process actually followed during the January–April 2026 CAS 741 project cycle for FitCoachAR.

This Development Plan covers the administrative, process and technology decisions for FitCoachAR, a scientific-computing project done for CAS 741 between January and April 2026. FitCoachAR is a solo project, so sections of the course template that are written for a team are adapted for a single developer. The sections below cover confidentiality and IP, copyright, meetings and communication, roles, git workflow, scheduling, the proof-of-concept plan, the technology stack, and the coding standard.

1 Confidential Information

FitCoachAR has no confidential information from industry. The source code, documentation and derived artefacts use only publicly available libraries (MediaPipe Pose, MoveNet, Django, NumPy, SciPy) and public biomechanics knowledge. There is no non-disclosure agreement in place.

2 IP to Protect

No intellectual property beyond the author's own work is involved. No third-party IP agreement applies, and the Intellectual Property Guide Acknowledgement is not required for this project.

3 Copyright License

FitCoachAR is released under the MIT License. The license text is in the `LICENSE` file at the root of the repository (https://github.com/mmdmirh/cas741). MIT was chosen so that the kinematics and state-machine modules can be reused later in personal or academic projects without friction.

4 Team Meeting Plan

This is a solo project, so there are no team meetings in the usual sense. The working pattern was:

- 4–6 focused working sessions per week during term, mostly during the CAS 741 Tuesday/Thursday slots and weekends.
- Interaction with the instructor, Dr. Spencer Smith, through (a) in-class questions, (b) the POC demonstration on 2026-03-05, (c) written review comments on GitHub issues, and (d) the final documentation review on issue #27.
- Peer review with Yibing Mei on each major deliverable (SRS, MG, MIS, VnV Plan, VnV Report) through GitHub issues. Every piece of feedback was either applied or explicitly noted as deferred.

5 Team Communication Plan

GitHub issues on `mmdmirh/cas741` were the main channel, used for review requests, peer-review feedback and instructor follow-ups. Every review was a separate issue so that the history stayed traceable. Review requests always linked to the PDF at a specific commit SHA. In-class discussion and email were used for anything that didn't fit in an issue thread.

6 Team Member Roles

All roles were handled by Seyedmohamad Mirhosseininejad. For traceability, the work can be split into:

- Requirements engineer: wrote the SRS, scoped R1–R5 and NFR1–NFR2, kept the traceability matrix updated.
- Architect: wrote the Module Guide and the Module Interface Specification, and defined the eight-module decomposition M1–M8.
- Implementer: wrote the Python/Django backend, the JavaScript frontend, and the WebSocket layer between them.
- Test engineer: wrote the VnV Plan and VnV Report, and the tests in `src/backend/tests/test_vnv.py` (11 tests) plus the interface-conformance harness.
- Reviewer: gave peer-review feedback on Yibing Mei's project and handled incoming feedback on this one.

7 Workflow Plan

All work is tracked in git at `github.com/mmdmirh/cas741`. The `main` branch holds the canonical state. Small changes go straight to `main`. Larger changes (for example, the initial refactor from the CAS 772 prototype) used a short-lived topic branch and were merged once the tests passed locally. Pull requests are not used as a review gate; reviewers read the published PDF rather than an open diff. Every review request and every review comment becomes a GitHub issue, with a title like "Peer Review of [artefact] by [reviewer] — [topic]". Issues are closed with a comment naming the commit SHA that addressed them. A GitHub Actions workflow in `.github/` rebuilds the documentation PDFs on push. The Python test suite is run locally via `make test` before significant pushes. There is no production deployment pipeline because this is a research-grade project.

8 Project Decomposition and Scheduling

GitHub Projects is not used as a separate board; issue labels and milestones on the repository itself are enough for a solo workload. The repository is at https://github.com/mmdmirh/cas741.

Scheduling follows the CAS 741 course outline. The big-picture schedule as actually delivered is shown in Table 2.

Phase	**Target (2026)**	**Artefact**
Problem Statement	Jan	ProblemStatement.pdf
SRS (draft + peer review)	Feb 11–15	SRS.pdf
VnV Plan	Feb 18 – Mar 12	VnVPlan.pdf
Proof-of-Concept demo	Mar 5	live demo
MG + MIS	Apr 3–5	MG.pdf, MIS.pdf
VnV Report	Apr 8–15	VnVReport.pdf
Reflection and Trace	Apr	ReflectAndTrace.pdf
Final documentation pass	Apr 15–20	All PDFs
EXPO presentation	Apr	expo.pdf

Table 2: Project Schedule, as Delivered

9 Proof of Concept Demonstration Plan

Two main risks were identified at the start of the term:

1. **Pose-estimation feasibility.** Can a normal webcam and a third-party pose-estimation library (MediaPipe or MoveNet) produce skeletal keypoints at a high enough frame rate and accuracy for real-time form feedback?

2. **Architectural feasibility.** Can the monolithic CAS 772 prototype be rewritten as an 8-module decomposition with abstract base classes, without losing real-time behaviour?

The POC on 2026-03-05 retired both risks by showing the end-to-end pipeline running live: the browser capturing frames, the Django WebSocket backend producing MediaPipe landmarks, the new `KinematicEngine` computing joint angles, and the `ExerciseStateMachine` counting valid reps with an AR overlay on the frontend. With the POC passed, the project moved on to the design documents (MG, MIS, VnV Report).

10 Expected Technology

Programming languages

- Backend: Python 3.9.6.
- Frontend: HTML, CSS and JavaScript (ES2020). React with Vite is used as a light build setup; no heavy transpilation is needed.

Frameworks and libraries

- Django and Django Channels for the WebSocket transport between the browser and the backend pipeline.
- MediaPipe Pose as the production pose-estimation backend (M4). MoveNet is used as a comparison backend in the offline benchmark.
- NumPy and SciPy for vector arithmetic in `KinematicEngine.compute_angle` and for the Kalman smoother in M8.

Pre-trained models

MediaPipe Pose and MoveNet are used as distributed upstream. No custom model training is in scope.

Components implemented despite library availability

The repetition state machine (M6) is hand-written rather than taken from a general-purpose FSM library. The validity criteria (`IM_SquatValid`, `IM_BicepCurlValid`, and the t_{min}/t_{max} timing guards) are closely tied to the SRS instance models, and keeping the code readable next to the requirement text was worth more than reusing a generic library.

Linter

`flake8` and `pylint` are run locally on the Python backend. They are part of the static verification workflow in the VnV Plan.

Unit testing framework

`pytest 8.4.2`. Tests are in `src/backend/tests/test_vnv.py`, with an interface-conformance harness in `src/backend/test_interfaces.py`. Both are run by `make test`.

Code coverage tool

`coverage.py` (via `pytest-cov`) was used for the Code Coverage Metrics section of the VnV Report.

Continuous integration

A GitHub Actions workflow rebuilds the documentation site on push. The Python test suite is run locally before significant pushes. No production CI pipeline is in scope, since this is a research-grade solo project.

Performance measurement

Average latency and frame-to-frame stability were measured by a custom offline benchmark script rather than by a general-purpose profiler like Valgrind. The metrics that matter here (per-backend latency and $\overline{|\Delta\theta|}$) are domain-specific.

Other tools

git, GitHub, GitHub Issues and GitHub Pages (for hosted PDF review links) complete the toolchain.

11 Coding Standard

- Python: PEP 8, enforced by `flake8`. Public module interfaces (M3, M4, M5, M6, M8) have type hints, backed by `abc.ABC` base classes so that the MIS contracts are visible at the language level.
- JavaScript: ES2020 with single-responsibility modules like `m2_display_output.js` and `m7_ui_rendering.js`.
- LaTeX: standard article class across all CAS 741 deliverables. Shared macros for \`progname` and \`authname` live in `docs/Common.text` and `docs/Comments.text`.