January 15, 2026

TermWise: OSU Course Planner

Abderrahmane Rhandouri

Eduardo Balzan

Quinn Carey

ABSTRACT

TermWise is a web app that helps Oregon State University students build a term-by-term plan without repeatedly hunting through the catalog, scheduler, and prerequisite pages. Students select a major (starting with Computer Science), enter completed/planned courses, and drag classes into Fall/Winter/Spring/Summer terms. As the plan is built, TermWise checks prerequisites and term offerings, warns about invalid sequences, and summarizes credit load and progress. The goal is to reduce planning mistakes (e.g., missing prerequisites or waiting a full year for a course), reduce time spent cross-referencing sources, and make it easier to iterate on "what-if" plans before registration. Our MVP focuses on a reliable course database, prerequisite validation, and a simple planner UI, with optional grade/GPA calculation to support academic decision-making.

Contents

1 Team Info

1.1 Team members and roles

- Abderrahmane Rhandouri — Product + UX Lead (requirements, wireframes, usability, UI implementation support)
- Eduardo Balzan — Full-Stack Lead (core architecture, planner features, integration, data pipeline)
- Quinn Carey — Backend + QA/DevOps Lead (API + database, testing strategy, CI, release workflow)

1.2 Project artifacts

- Git repository: https://github.com/CareyQT/Software-Engr-II
- Issue tracker/backlog: GitHub Issues (same repo)
- Project board (sprint planning): GitHub Projects (linked from repo)
- UI mockups (if used): Figma link: https://www.figma.com/design/I1udQyiFAr4RtYFuvXuunC/CS-362-Figma?t=tIq0DZHvS46gtnjY-0
- Living document: `/docs/living-doc.typ` (generated PDF committed for submissions)

1.3 Communication channels and rules

- Primary chat: Microsoft Teams (private team channel)
- Async decisions: GitHub Issues (each task has an issue; decisions documented in comments)
- Meetings: 1 weekly sync (30–45 minutes) + additional short check-ins as needed
- Response expectations:
  - Discord: respond within 24 hours on weekdays
  - GitHub review requests: respond within 48 hours on weekdays
- Rules:
  - No direct commits to `main` (PRs only)
  - Every PR must reference an issue and include a short test note ("how I verified")
  - At least 1 approval required before merge
  - If blocked >24 hours, post the blocker in Discord + the issue thread

2 Product Description

2.1 Goal

Help OSU students create a valid, term-by-term class plan for an academic year (and beyond) by automatically applying constraints like prerequisites, co-requisites (where supported), and which terms a course is offered. The system should reduce planning mistakes and time spent cross-checking multiple OSU sites.

2.2 Current practice

Today, students typically plan by manually consulting the OSU catalog for prerequisites, the Schedule of Classes for term offerings, and degree audit tools to estimate progress. This is slow, error-prone, and repetitive—especially when a student changes a plan and must re-check prerequisite chains and course availability across Fall/Winter/Spring/Summer. Existing tools often feel fragmented (catalog here, planner elsewhere, GPA tools elsewhere) and don't provide immediate "this plan is invalid because…" feedback while building the schedule.

2.3 Novelty

TermWise combines (1) a planner UI and (2) automated validation against prerequisite rules and term offerings in one place, with fast feedback as students build "what-if" schedules. Instead of searching course pages repeatedly, the app surfaces key constraints directly inside the planning workflow

(warnings, unmet prereqs, next eligible term). The project is not trying to replace official registration; it targets the planning step before registration, where students need clarity and iteration speed.

2.4 Effects

If successful, TermWise will help students:
- Avoid delaying graduation due to missed prerequisites or missed once-per-year offerings
- Build more balanced term credit loads and spot overload early
- Reduce advisor meeting time spent on basic prerequisite/availability lookups
- Make faster, more confident schedule decisions before registration windows open

2.5 Technical approach

We will build a web application with:
- Frontend: React/Next.js planner UI (drag-and-drop term columns, course search, plan summary)
- Backend: REST API for courses, offerings, and plan validation
- Database: relational storage (e.g., PostgreSQL) for course metadata, offering terms, and prerequisite structures
- Data ingestion: a small ETL script that populates course data from public OSU sources (catalog + schedule listings) into our database
- Validation engine: given a student's completed courses + planned terms, compute unmet prerequisites and mark courses as eligible/ineligible per term

(Exact tech choices may be adjusted based on team strengths and course expectations, but the architecture remains: UI + API + DB + ingestion + validator.)

2.6 Risks

The most serious risk is accurately interpreting and validating prerequisite rules at the project scale and within time. OSU prerequisites are sometimes written in complex natural language (OR/AND groups, grade requirements, concurrent enrollment, placement tests). Mitigation:
- Scope the first release to a well-defined subset (e.g., OSU Computer Science core + common electives) with test cases for each prereq pattern we support
- Store prerequisites in a structured internal format (not only raw text), and manually curate edge cases for the subset
- Add automated tests using real prerequisite strings from our supported course set to prevent regressions

2.7 Major features (MVP)

- **Feature 1**: Course explorer and search
  ‣ Search/filter by subject/number, credits, and offered terms (Fall/Winter/Spring/Summer where known)
- **Feature 2**: Term-by-term planner
  ‣ Drag/drop courses into term columns, see total credits per term, and reorder terms easily
- **Feature 3**: Prerequisite and eligibility validation
  ‣ Real-time warnings for unmet prerequisites; highlight the earliest eligible term for a selected course
- **Feature 4**: Saved plans (persistence)
  ‣ Store a user's plan and completed courses; reload and edit later (basic accounts or local persistence acceptable for MVP)
- **Feature 5**: Grade/GPA calculator (simple)
  ‣ Let users enter expected grades and compute term GPA and cumulative GPA estimates

2.8 Stretch goals

- **Stretch 1**: Auto-plan suggestions
  - ‣ Given constraints (max credits/term, target graduation term), generate a recommended sequence
- **Stretch 2**: Degree progress tracking (limited)
  - ‣ For one major (starting with Computer Science), show requirement groups and completion status from the user's plan
- **Stretch 3**: Share/export
  - ‣ Export plan to PDF and/or share a read-only link for advisors/peers