Software Project Design Course – Term Project Assignment Document

Project Description and Scope

Project:

Within the scope of this project, students are expected to develop a mobile application and a backend system that allow students from primary school to high school level to track their assignments. Teachers must be able to assign homework to their classes, students must be able to complete their assignments through the mobile application, and both parents and teachers must be able to monitor the assignment status.

The core functionalities of the project are defined below. However, not all design details are strictly specified. In line with the course objectives, architectural decisions, user experience design, and technical implementation choices—other than the mandatory requirements—will be determined by the students. The purpose of this approach is not only to implement a given system, but to provide students with the experience of designing a software product from start to finish and making engineering decisions throughout the process.

Target Platform and Team Structure

Target Platform:

The application must run on both iOS and Android platforms. Cross-platform technologies (such as Flutter or React Native) or native development approaches may be used.

Team Structure

Each team consists of 6 students. Every team is responsible for the product end-to-end, including design, development, testing, and delivery.

The following roles are defined within the project:

1. Requirements Analyst
2. Software Developer
3. UI/UX Designer (User Interface / User Experience Designer)

Each student will have a primary role. However, team members are expected to support one another when necessary, and the entire team shares responsibility for the overall quality of the product.

Separate roles for process-oriented responsibilities such as DevOps or Quality Assurance (QA) are not defined. These responsibilities will be carried out collaboratively by the team.

Roles are not strictly separated. The objective is to help students develop role awareness while also fostering shared responsibility and a strong culture of teamwork.

1–4. Week	Team Member A – Requirements Analyst Team Member B – Requirements Analyst Team Member C – Software Developer Team Member D – Software Developer Team Member E – UI/UX Designer Team Member F – UI/UX Designer
5–8. Week	Team Member A – UI/UX Designer Team Member B – UI/UX Designer Team Member C – Requirements Analyst Team Member D – Requirements Analyst Team Member E – Software Developer Team Member F – Software Developer
9–12. Week	Team Member A – Software Developer Team Member B – Software Developer Team Member C – UI/UX Designer Team Member D – UI/UX Designer Team Member E – Requirements Analyst Team Member F – Requirements Analyst

The tasks defined under each role are generally as follows.

1. Requirements Analyst

Core Responsibilities:

- Clarifying the project vision and problem definition
- Analyzing target users (student, teacher, parent)
- Writing user stories and acceptance criteria
- Preparing the SRS (Software Requirements Specification) document
- Defining functional and non-functional requirements
- Contributing to the data inventory and Data Protection Impact Assessment (DPIA) activities
- Keeping the product backlog up to date and contributing to Minimum Viable Product (MVP) decisions
- Ensuring that change requests are properly reflected in the documentation

Essence of the role:
Providing clear answers to the questions: “What are we building?” and “Why are we building it?”

2. Software Developer

Core Responsibilities:

- Developing the mobile application (iOS/Android) and/or backend (for example, implementing the mobile screen where students view their assignment list and developing the backend service that provides this data)
- Designing and implementing APIs (for example, enabling teachers to create assignments via POST /assignments and allowing students to list their own assignments via GET /assignments)
- Implementing role-based access and authentication mechanisms (for example, ensuring that students cannot access the teacher panel and that parents can only view data related to their own child through proper token and role validation)
- Designing the data model and performing database operations (for example, creating user, class, and assignment tables and correctly associating assignment submissions with the appropriate student and assignment)
- Writing unit and integration tests (for example, a unit test verifying that the score calculation function works correctly and an integration test confirming that the assignment status is updated in the backend after a student log in and submits homework)
- Maintaining a working CI (Continuous Integration) process (for example, automatically running build and tests on each code push and ensuring that the pipeline fails if any test does not pass)
- Contributing to release build generation and technical distribution processes (for example, producing signed Android APK/AAB or iOS IPA files and initiating TestFlight or Google Play internal testing)

Essence of the role:
Transforming defined requirements into working, secure, and testable code.

3. UI/UX Designer (User Interface / User Experience Designer)

Core Responsibilities:

- Preparing clickable prototypes (for example, modeling a fully clickable flow in Figma from the teacher’s assignment creation screen to the student’s assignment viewing and submission screen)

- Designing separate user flows for students, teachers, and parents; making age-appropriate design decisions for the first-grade module (for example, creating clear and distinct flow diagrams for actions such as “create assignment → assign to class → view submissions” for teachers; “log in → view assignment → complete → submit” for students; and “select child → view pending assignments → view weekly summary” for parents)
- Applying accessibility principles (such as appropriate font sizes, sufficient color contrast, and adequately sized touch areas for younger users)
- Contributing to usability evaluations (for example, conducting simple task-based tests on the prototype to identify where users struggle with finding and submitting assignments and improving the design accordingly)
- Assisting in the preparation of store visuals and application description assets

Essence of the role:
Ensuring not only that the system functions correctly, but that it delivers an appropriate and meaningful user experience.

Shared Responsibility

Although each role has a primary area of responsibility:

- Testing, quality, and security are the responsibility of the entire team.
- The product, end-to-end, is the responsibility of the whole team.

In other words, no one can say, “That’s not my job.”

Notes

<u>Unit Test – Short Example</u>

Scenario: Does the score calculation function work correctly?

Assume the system contains the following function:

```
def calculate_score(correct, total):
    return (correct / total) * 100
```

Unit test:

```
def test_calculate_score():
    result = calculate_score(8, 10)
    assert result == 80
```

This test verifies the following:

- Whether a single function works correctly in isolation
- Whether the business rule (score calculation logic) is implemented correctly

A unit test focuses on testing a small, independent part of the system to ensure it behaves as expected.

For additional examples and detailed explanations, you may refer to: https://www.geeksforgeeks.org/software-testing/unit-testing-software-testing/

Integration Test – Short Example

Scenario: A student logs in → submits an assignment → is the status updated correctly?

Pseudo example:

```
def test_student_submit_assignment(client):
    login_response = client.post("/login", data={"email": "student@test.com", "password": 
"1234"})
    token = login_response.json()["token"]

    submit_response = client.post(
        "/assignments/5/submit",
        headers={"Authorization": f"Bearer {token}"},
        json={"content": "Ödev cevabı"}
    )

    assert submit_response.status_code == 200
```

This test verifies the following:

- Does authentication (auth) work correctly?
- Does the endpoint function as expected?
- Is the status correctly updated in the backend?

Integration testing means verifying whether multiple components of the system work together correctly.

For additional examples, you may refer to: https://www.geeksforgeeks.org/software-testing/software-engineering-integration-testing/

Weekly Tasks

Week 1

1) Project Vision

- **Acceptance criterion:** The project must clearly explain, in a single paragraph, the problem it aims to solve, the target users (student, teacher, parent), and the value proposition provided to these users.
- **Acceptance criterion:** Two separate lists must be prepared for the MVP (Minimum Viable Product):
    1. **Included (Features that will be implemented this semester):**
       This list must contain the essential features required for the application to function at a basic level. Without these features, the system cannot be considered operational. (For example: user authentication, teacher assignment creation, student assignment viewing and submission, role-based access control.)
    2. **Excluded (Features that will not be implemented this semester):**
       This list must include features that may be valuable but will not be developed due to time, resource, or priority constraints. This section aims to consciously limit the project scope. (For example: advanced analytics dashboard, EBA integration, advanced notification infrastructure, offline support.)

2) Team Agreement

- **Acceptance criterion:** The distribution of roles and responsibilities within the team must be clearly defined. Each member's area of responsibility (e.g., analysis, design, development, testing/quality) must be explicitly stated.
- **Acceptance criterion:** The weekly meeting day and time, communication channels to be used (e.g., WhatsApp, Discord, Teams), and the method for resolving internal conflicts must be clearly defined.

3) Product Backlog v0

- **Acceptance criterion:** At least 15 user stories must be written, and each must include an initial acceptance criterion.

Notes:

<u>What is MVP (Minimum Viable Product)?</u>

An MVP is the smallest working version of a product that fulfills its core purpose. It is not a fully featured system, but the first functional version that solves the main problem.

In this project, the core problem is defined as follows: A teacher must be able to assign homework → a student must be able to view and complete the assignment → both the parent and the teacher must be able to monitor the assignment status.

If these three core functionalities work correctly, the product can be considered to have reached MVP level.

What is a User Story?

A user story is a simple requirement-writing format commonly used in Agile development.

It is usually written in the following structure:

"As a ___, I want to ___ so that ___."

For example:

- "As a teacher, I want to assign homework to my class so that I can track students' progress."
- "As a parent, I want to see my child's completed assignments so that I can monitor their academic performance."

What is an Initial Acceptance Criterion?

Acceptance criteria answer the questions: "Does this feature really work?" and "How can we verify it?"

They define measurable and testable conditions under which a feature is considered complete.

For example, if the user story is:

"As a teacher, I want to assign homework to my class."

Possible acceptance criteria may include:

- The teacher must be able to enter a title and description.
- A due date must be selectable.
- The assignment must be visible only to students in the selected class.
- When the "Save" button is clicked, the data must be stored in the backend.

These criteria define when the feature can be considered completed.

<u>What is a Product Backlog?</u>

The Product Backlog is a prioritized list of all tasks, features, and improvements planned for the project.

The term “v0” indicates that this is the initial draft (version 0) of the list.

For example, the backlog for this project may include:

- User authentication system (student / teacher / parent)
- Role-based authorization mechanism
- Teacher class creation functionality
- Student class enrollment via code

Week 2

1) SRS v1 (Software Requirements Specification)

- **Acceptance criterion:** Functional requirements (what the system must do) and non-functional requirements (quality attributes such as performance, security, usability, etc.) must be written clearly and in an organized way under separate headings.
- **Acceptance criterion:** The document must include at least 10 acceptance test definitions. These tests must be written in **Given / When / Then** format (or a similar structure that clearly expresses the same logic) and must clearly show how the related feature of the system will be tested.

2) Data Model Draft (ERD – Entity–Relationship Diagram)

- **Acceptance criterion:** The data model must clearly define at least the following core structures: **User** (including student/teacher/parent roles), **Class**, **Assignment**, and **Assignment Submission/Status**. The fields of these tables (e.g., id, role, title, due date, status, etc.) and the relationships between them (e.g., one teacher can have multiple classes, one class can include multiple students, etc.) must be shown on the diagram.
- **Acceptance criterion:** The method for verifying the **parent–student relationship** must be clearly explained. For example, if an invitation code, pairing code, teacher approval, or another verification method is used, this method must be represented in the data model and briefly explained. In addition, the relationship structure that ensures a parent can access **only their own child’s data** must be clearly defined.

3) Privacy Impact Note (Mini DPIA – Data Protection Impact Assessment)

- **Acceptance criterion:** All data fields that will be collected within the application must be clearly listed (e.g., first/last name, email, role information, class information, assignment status, submission date, etc.). For each data field, a short and clear justification must be provided answering: "Why is this data collected?"

  The document must also describe the data minimization approach:

  - Which data is intentionally not collected
  - How the design avoids collecting unnecessary data
  - For what purpose and for approximately how long the data will be stored (assumption-level explanation is acceptable)
- In addition, potential privacy risks (e.g., unauthorized access, data leakage, showing data to the wrong user, etc.) must be briefly identified. Basic technical mitigation measures (e.g., role-based access control, password security, use of test data, etc.) must be explained.

  For this section, students should refer to relevant sources including KVKK (Turkey's Personal Data Protection Law) and GDPR (General Data Protection Regulation – EU) where DPIA is defined.

Notes

An example acceptance test for "Teacher Creates an Assignment" may be written as follows:

- **Given:** The teacher is logged in.
- **When:** The teacher enters a title, description, and due date and clicks the "Save" button.
- **Then:** The assignment is saved in the database and becomes visible in the assignment list of students in the relevant class.

Week 3

1) Clickable Prototype

- **Acceptance criterion:** Separate main screen flows for student, teacher, and parent must be clickable (including post-login landing screen and core actions). At minimum, the prototype must simulate the following basic interactions: logging in, navigating to the home screen, accessing the assignment list, and reaching the role-specific primary action screen. Screen transitions (buttons, menus, navigation elements, etc.) must be logically connected and properly simulated. Tools such as Figma may be used.

2) User Interaction Flows

- **Acceptance criterion:** An end-to-end user flow diagram covering the process "Create assignment → assign to class → student views assignment → student submits → teacher/parent views result" must be clearly presented. The diagram must explicitly indicate which role (teacher, student, parent) performs each step. The flow should clearly and consistently reflect the overall system logic.

3) Accessibility and Child User Experience (UX) Notes

- **Acceptance criterion:** For the first-grade module, at least 5 justified design decisions must be documented regarding typography (font size and readability), color selection (contrast and attention level), and interaction design (touch area size, simplified interface, etc.). Each decision must briefly explain why it is appropriate for the relevant age group.

Week 4

1) Code Repository and Working Setup

- **Acceptance criterion:** A source code repository must be created for the project. It must include a README file, a meaningful folder structure, and a defined branch strategy (e.g., main / dev / feature). The README must clearly state which technologies are used, the required setup steps, and step-by-step instructions for running the project locally. In this context, main can be considered the stable branch containing release-ready code, dev the integration branch where development work is merged, and feature a temporary branch created for implementing a specific new feature.
- **Acceptance criterion:** An issue board (task tracking board) must be set up for the project. The board must clearly list tasks selected from the product backlog for at least one sprint, organized as "to do / in progress / done." Each task must include a short description, an assigned team member, and a status.

2) Continuous Integration (CI) Setup

- **Acceptance criterion:** On every code push, the project must automatically build (at least for one platform). The CI pipeline must include at least one quality check step (e.g., linting). Linting refers to automatically checking the source code for programming issues and style inconsistencies. Even if tests have not been written yet, a test step must still be planned and configured in the pipeline.

3) Architecture Decision Records (ADR) Set

- **Acceptance criterion:** The team must document important technical decisions made throughout the project in written and well-justified ADRs. The goal is to answer not only “what did we choose?” but also “why did we choose it?” At least 3 ADRs must be created (e.g., mobile technology choice, backend approach, database choice). Each ADR must clearly answer the following three questions: What decision was made? Why was this decision made? What alternative was considered and why was it not chosen?

Week 5

1) Authentication (Auth) v1

- **Acceptance criterion:** Registration and login must be functional for at least two different user roles (e.g., student and teacher). Ideally, all three roles (student, teacher, parent) should be supported. Each role must be able to access only the screens that match its permissions.
- **Acceptance criterion:** After login, session management must be handled using a session mechanism or a token. The approach must be documented clearly, including: how the token is generated, where it is stored in the application (e.g., local storage / secure storage), and approximately how long it remains valid.

2) Role-Based Access Control (RBAC) v1

- **Acceptance criterion:** Users must be able to access only the screens and data that belong to their role (student, teacher, parent). For example, a student must not be able to access teacher screens, and a parent must be able to view only data related to their own child.
- **Acceptance criterion:** At least 5 negative test scenarios must be written and executed. These scenarios must test unauthorized access attempts (for example, a student attempting to call a teacher-only API endpoint, or a parent trying to view another student’s data). The results of these tests must be clearly reported.

3) Basic Profile Screens

- **Acceptance criterion:** The profile screen must display the essential information required for the system, such as the user’s role (student/teacher/parent), name or nickname, and class information.
- **Acceptance criterion:** The data minimization principle must be applied. The profile screen should include only necessary information, and unnecessary or sensitive personal data (such as national ID number, address, or phone number) must not be collected or displayed in the system.

Week 6

1) Class Management (v1)

- **Acceptance criterion:** A teacher must be able to create a new class and list the classes they have created. Each class must have at least a name (e.g., “5-A Math”) and a record linked to the teacher. Class data must be stored persistently in the backend.

2) Student Class Enrollment (v1)

- **Acceptance criterion:** A working student enrollment flow must be implemented and documented (e.g., invitation code, QR code, or teacher approval). After joining, the student must be able to view only the classes they are enrolled in. The system must include a basic control mechanism so that a student is not accidentally linked to the wrong class (e.g., unintended automatic enrollment).

3) Parent–Student Linking (v1)

- **Acceptance criterion:** A mechanism to verify and establish the parent–student relationship must be defined and implemented (e.g., pairing code, teacher approval, etc.). A parent must be able to access only the data of the student they are linked to. Unauthorized access must be prevented at the backend level.

Week 7

1) Assignment Core (Create and Assign) (v1)

- **Acceptance criterion:** A teacher must be able to create an assignment by entering a title, description, and due date, and then assign it to a selected class.

2) Assignment Listing (v1)

- **Acceptance criterion:** A student must be able to view only the assignments assigned to their own class(es). The listing screen must include at least an “active” vs. “past” separation. The list must be ordered by due date or presented in another logical and consistent order.

3) Assignment Status Model (v1)

- **Acceptance criterion:** At minimum, the assignment must support the following status values: **Assigned → In Progress → Submitted**.
- **Acceptance criterion:** Status changes must be recorded in the backend, and a timestamp (date/time) must be stored for each status change.

Week 8

1) Second Assignment Type (v1) — Fully Working

- **Acceptance criterion:** Assignment Type A and Assignment Type B must be clearly different in both **UI (User Interface)** and **business rules**. The screen structure, user interaction, and submission process must not be the same for both types. For example, one assignment type may require text or drawing submission, while the other may be a multiple-choice quiz with automatic scoring. The evaluation method and the way results are produced must also be different.

2) API Contract Document (OpenAPI / Markdown)

- **Acceptance criterion:** At least 8 API endpoints must be documented (e.g., login, class listing, assignment creation, assignment listing, assignment submission, etc.).

For each endpoint, the following information must be clearly provided:

1. Request example: HTTP method (GET/POST, etc.), endpoint path, and an example request body (if applicable).
2. Response example: Example response data structure for a successful request, including the HTTP status code (e.g., 200, 201).
3. Error codes and meanings: Possible error cases and corresponding HTTP status codes (e.g., 401 Unauthorized, 404 Not Found, 400 Bad Request) with short explanations.

3) Assignment Submission Evidence Approach

- **Acceptance criterion:** Each student submission must be stored in the backend as a **submission artifact** (e.g., text, drawing, or quiz result). This artifact must be linked in the database with information about **which student submitted it, for which assignment, and at what date/time** (e.g., student ID, assignment ID, timestamp). The teacher and/or the relevant role (depending on the design, this could also include the parent) must be able to view this submission artifact through the system.

Week 9

1) First-Grade Interactive Module

- **Acceptance criterion:** A working touch/canvas-based screen must be implemented where students can perform drawing or letter-writing practice. The student must be able to draw, save the work, and send it to the teacher. The submitted work must be stored in the backend.

- **Acceptance criterion:** The student must receive a simple and clear feedback mechanism (e.g., messages such as “Completed” or “Try again”). Basic progress information related to the work (e.g., whether it was submitted and when it was completed) must be recorded in the backend.

2) Multiple-Choice Quiz Module

- **Acceptance criterion:** Quiz questions and options must be displayed correctly and clearly in the UI. The student must be able to select an option for each question, and the selected answers must be stored in the backend.
- **Acceptance criterion:** The system must perform automatic scoring when the quiz is completed. The result screen must clearly show the student’s score and the number of correct/incorrect answers. The teacher must be able to view the student’s quiz result (score, correct/incorrect count, and optionally answer details) through the system.

Week 10

1) Parent Panel (v1)

- **Acceptance criterion:** The parent must be able to view pending and completed assignments for their own child separately. At least one date-based listing or filtering option must be available (e.g., “this week” / “all”). The parent must be able to view only the data of the student they are linked to.

2) Notification Approach (v1)

- **Acceptance criterion:** When a new assignment is assigned, a notification must be shown to the student and/or parent. The notification may be designed as in-app or push notification. The message content must be clear and informative (e.g., “A new Math assignment has been assigned – Due: March 15”). The event that triggers the notification (e.g., assignment assignment action) must be clearly defined.
- **Acceptance criterion:** At least one basic notification preference setting must be provided so users can enable/disable notifications. This setting must be accessible from the profile or settings screen.

3) Simple Weekly Summary

- **Acceptance criterion:** The parent panel must display a weekly summary for the related student. For example: “This week, your student Ada Yazılım completed 3 assignments, and 2 assignments are still pending.” The summary must be based on backend data and calculated using real or up-to-date records (i.e., it must not be hard-coded).

Week 11

1) Test Package and CI (Continuous Integration)–Based Quality Checks

- **Acceptance criterion:** At least **5 unit tests** and at least **3 integration tests** must be written and must run successfully. Integration tests must verify that multiple components work correctly together. Tests must not only check “it runs,” but must validate the system’s core business rules (for example, preventing unauthorized role access and ensuring score calculation is correct).
- **Acceptance criterion:** In the CI environment, tests must run automatically on every code push. If any test fails, the build process must be marked as **failed**.

2) Security and Privacy Deliverables

- **Acceptance criterion:** A **data inventory** must be prepared (including collected data fields, the purpose of collection, and the expected retention period).
- **Acceptance criterion:** A draft **privacy policy** must be prepared, and an in-app **“Information Notice” (Disclosure)** screen must be designed.
- **Acceptance criterion:** Basic security controls must be implemented: **input validation**, **role-based access control**, and **error logging mechanisms**. These controls are not expected to be added only in Week 11—rather, they should already exist in earlier work; Week 11 is the stage for final review and completion.

3) Store Release Preparation and Legal/Data Declarations (Draft)

- **Acceptance criterion:** An application icon, at least a few screenshots, and an app description text must be prepared.
- **Acceptance criterion:** The information that must be officially declared by the developer during store submission must be identified and clearly documented. For example: what data is collected, what the privacy policy URL is, whether the app is directed at children (this is especially important in this project because the target audience includes children), and whether the app contains ads.

Week 12

1) Release Candidate (RC)

- **Acceptance criterion:** A working **release build** must be produced for both iOS and Android (signed and ready for distribution).
- **Acceptance criterion:** Sensitive data such as API keys, access tokens, and database connection information must not be stored as plain text inside the source code. These values must be managed using **environment variables** or secure configuration files. A

short technical note must document how secrets are stored and how they are loaded at runtime.

2) Release Preparation / Proof of Submission

- **Acceptance criterion (iOS):** TestFlight distribution must be initiated or the App Store submission process must be started. If real publication is not possible due to developer account limitations, alternative proof must be provided. In this case, at minimum the following must be submitted:
    1. A screenshot showing that the app has been created in App Store Connect,
    2. A signed IPA file (release build),
    3. A short technical document explaining how far the submission process progressed. Reference https://developer.apple.com/testflight/

- **Acceptance criterion (Android):** Internal Testing or Closed Testing must be initiated through Google Play Console, or the store submission process must be started. In addition, evidence must be provided showing that the **Google Play Data Safety** form has been completed. References: https://play.google.com/console/about/internal-testing/ https://play.google.com/console/about/closed-testing/

3) Final Report and Presentation

- **Acceptance criterion:** The final report must include key project deliverables such as the system architecture, data model, API design, test strategy, privacy approach, and major technical risks.
- **Acceptance criterion:** The report must include at least one **retrospective evaluation** section. In this section, technical and/or organizational challenges encountered during the project must be analyzed, including their causes and the preventive actions that would reduce the likelihood of similar issues happening again. Even if the project progressed smoothly, areas for improvement and what could be done differently in the future must be stated. If the project experienced a serious bug, outage, or failure, this must be additionally documented with a detailed **root cause analysis**.

Evaluation and Grading

To reduce the risk of unequal contribution in team projects, an individual contribution multiplier will be applied to the final team grade.

Project Submission Requirements

a) Technical Deliverables

1. Source Code Repository: An accessible repository containing all source code (branch structure must be preserved).
2. CI Output: Evidence of the latest successful build and test results from the CI pipeline (log output or screenshot).
3. Backend and Database: Backend source code, data model/schema, and a brief explanation of how the system can be executed.
4. Test Package: Unit and integration tests along with test execution results.
5. UML Set: At minimum, a class diagram and diagram(s) showing user flows.
6. SRS Document: The updated requirements document.
7. Security and Privacy Package: Data inventory, draft privacy policy, and related documentation.
8. Store Package: Application icon, screenshots, description text, and proof of store submission/testing.

b) Non-Technical Deliverables

1. Weekly progress notes (what was done / what will be done / encountered obstacles / proposed solutions)
2. Risk register (risk description, probability, impact, mitigation actions)
3. Final report (including all technical and process outputs)
4. Presentation slides and a short demo video (recommended)

Risk Management and Process Rules

Weekly Scrum / Stand-up Expectations

A short stand-up meeting will be held each week during class hours. The purpose of this meeting is to monitor team progress and identify potential risks early. During the stand-up, each team must briefly and clearly answer the following three questions:

- What did we accomplish this week?
- What will we do next week?
- What obstacles are we facing?

The project should be planned to progress in short sprint cycles. Core process steps such as sprint planning, task tracking, and progress monitoring should align with the fundamental principles of the Scrum approach.

Weekly review and feedback sessions are designed to help teams detect mistakes early and complete the project on time. This process is intended not only for grading, but primarily to support learning and continuous improvement.