## Giriş

Bu yalnızca bir kodlama egzersizi değil, yazılım mühendisliği odaklı bir projedir. Projenin sonunda çalışan bir yazılım sistemi sunulmalıdır.

Geliştirmeniz **planlama** ile başlayacaktır. Projede yapacağınız şeyleri **task**lar (görev) aracılığıyla tanımlamanız gerekiyor. Her task, açık bir tanım ve bu taskların gerçekleştiğini anlayabilmek için *acceptance criteria*lar içerecekler. Bunun için **Github Projects** üzerinde bir **Kanban** board kullanılacak.

Sistem aynı zamanda, net bir şekilde tanımlanmış veri modeline sahip olacak ve bunu **ER diyagramı** ile göstereceksiniz. Tüm geliştirme süreci **Git** (ve Github) ile yürüyecek. Tüm yapılan commitler, direkt olarak ilgili tasklarla ilgili olacak ve commit mesajında task id'ler yer alacak.

Geliştirme sürecinde, **CI/CD pipeline** kurmak için **Github Actions** kullanılacak. Yapılan her push, bu pipeline'ı trigger edecek ve otomatik testler koşacak (unit testing, linting, vb.).

Bunlar, projenin başından itibaren takip edilmesi gereken şeyler. Bunları sonradan yapamazsınız.

## Ara Sınav

Ara sınav haftasında bir sunum yapacaksınız. Bu aşamada, size verilen gereksinimlerin minimum olanları yapılmış olmalı. Eğer bu olmazsa, o zaman doğru bir uygulama yapmaya ya da rapor yazmaya vaktiniz kalmaz.

Aşağıda belirtilen gereksinimler, kesinlikle yapılması şart olan gereksinimlerdir:

## Teknik Kapsam

En az bir küçük özelliğin, kullanıcı arayüzünden (UI) başlayıp, backend'e ve oradan veri tabanına kadar uçtan uca eksiksiz çalışıyor olması gerekir.

### Altyapı ve Süreç Kurulumu

- **Github Actions:** Her *push* işleminde otomatik olarak çalışmalı, pipeline sorunsuz işlemelidir.
- **Veri Tabanı:** Hazırlanan ER diyagramı, mevcut canlı veri tabanı ile birebir eşleşmelidir.
- **Kanban Panosu:** "Done" ve "In Progress" sütunları mantıklı bir şekilde dolu olmalıdır.

- **Git Geçmişi:** Yapılan çalışmalar düzenli ve tutarlı *commit* mesajlarıyla takip edilebilmelidir.

### Temel Kimlik Doğrulama ve Çekirdek Mantık

Kullanıcı giriş yapma (login) ve kayıt olma (sign-up) süreçleri hatasız çalışıyor olmalıdır.

## Değerlendirme Tabloları

### Ara Sınav Değerlendirmesi

| Category | Weight | Criteria | Excellent (A) |
|---|---|---|---|
| **Integrated Vertical Slice** | 35% | UI → API → DB Flow | A core user story is fully functional from the frontend to the persistent database layer. |
| **Engineering Quality** | 25% | Database & DevOps | Veritabanı şeması ER ile uyumlu olmalı, CI pipeline aktif olmalı. |
| **Project Management** | 20% | Git & Kanban Traceability | Her commit bir Task ID'ye bağı olacak, Kanban realistic olacak ve tasklar doğru olacak. |
| **Technical Defense** | 20% | Mastery & Live Edit | Koda hakimiyet, anlık editleme becerisi. |

**Genel Değerlendirme Kriterleri**

| Category | Weight | Criteria | Excellent (A) |
|---|---|---|---|
| **Project Management** | **25%** | Kanban, Git, & Task Tracking | Kanban'ın doğru kullanımı; taskların net kabul şartları; commit'lerin task IDleri içermesi. |
| **Engineering Quality** | **25%** | Database & DevOps | Normalized ER diagram; CI pipeline (lint/test) passes on every push; clean Git history. |
| **Technical Execution** | **30%** | The Software System | Full-stack functionality achieved; secure and efficient data handling; robust UI/UX. |
| **The Report (LaTeX)** | **10%** | Documentation & Structure | Professional use of LaTeX; follows the "Software-IMRAD" mapping; clear technical diagrams. |
| **The Defense (Jury)** | **10%** | Presentation & Mastery | Ability to justify architectural choices; clear demo; stays within time limits. |

## Detaylı Değerlendirme Kriterleri

### Proje Yönetimi & Git (%25)

| Score | Kanban & Task Definition | Git Workflow & Commits |
|---|---|---|
| **90-100** | Tasks have clear "Acceptance Criteria." Kanban reflects real-time progress. | Every commit is linked to a task ID (e.g., `#12`). Meaningful commit messages. |
| **70-89** | Tasks are defined but acceptance criteria are vague. | Most commits are linked to tasks, but some "fix" or "test" commits are stray. |
| **50-69** | Kanban used sporadically; tasks are too large (e.g., "Build Backend"). | Commit messages are generic (e.g., "update," "changes"). Linkage to tasks is rare. |
| **0-49** | No Kanban board or it was created all at once at the end. | Single "initial commit" or massive, undocumented pushes. |

### Kod Kalitesi & DevOps (%25)

| Score | Database & Data Modeling | CI/CD & Automated Checks |
|---|---|---|
| **90-100** | ER Diagram is professional (Mermaid/Crow's Foot). DB is normalized; constraints used. | CI pipeline (Linter + Tests) runs on every push. No "red" builds left unaddressed. |

| | | |
|---|---|---|
| **70-89** | ER Diagram exists but has minor logic flaws. DB works but lacks some constraints. | CI pipeline exists but only runs a linter or simple "smoke" tests. |
| **50-69** | ER Diagram is a rough sketch. DB structure is flat or inefficient (Redundancy). | CI pipeline was set up late or frequently bypassed/failing. |
| **0-49** | No ER Diagram. Hardcoded data or chaotic DB structure. | No automated checks. Manual testing only. |

**Teknik Uygulama (%30)**

| Score | Backend & API | Frontend & Integration |
|---|---|---|
| **90-100** | Robust API design. Handles errors gracefully (4xx/5xx codes). Secure. | Responsive UI. Seamlessly consumes the API. State management is logical. |
| **70-89** | Functional API but lacks consistent error handling or security. | UI is functional but has "bugs" or poor user feedback during loading/errors. |
| **50-69** | Basic CRUD only. Business logic is messy or repetitive. | Minimalist UI; poor separation of concerns between logic and view. |
| **0-49** | System crashes frequently. API/Database connection is unstable. | UI is broken or hardcoded; not a "working system." |

**Sunum: Rapor & Jüri (%20)**

| Score | LaTeX Report (Software-IMRAD) | Jury Presentation & Defense |
|---|---|---|
| **90-100** | Perfect LaTeX formatting. Clear mapping of Engineering to IMRAD. Professional diagrams. | Demonstrates deep mastery. Can justify every tech choice (e.g., "Why SQL?"). |
| **70-89** | Good use of template, but some sections (Discussion/Results) are thin. | Good demo. Answers most technical questions but struggles with "*edge cases*." |
| **50-69** | Significant LaTeX errors or poor image quality. Text is purely descriptive, not analytical. | Presentation is "just a walkthrough" with no engineering justification. |
| **0-49** | Plagiarism, missing sections, or failed to use the provided template. | Unable to explain how the code works or why specific tools were used. |