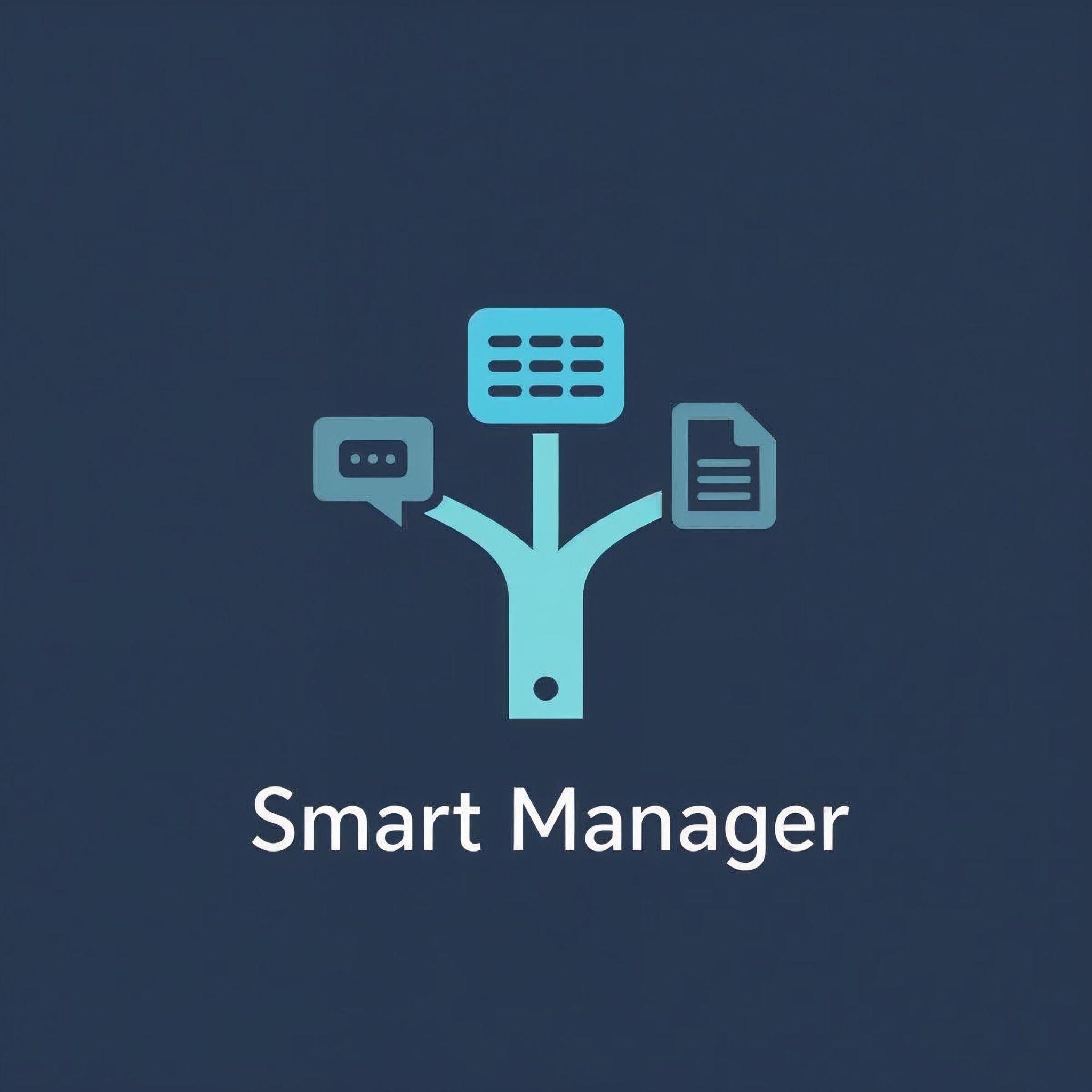
bevezető

**Smart Manager**

**Készítették:**

Magyar Márk

&

Nagy Huba

Tartalomjegyzék

**Tartalomjegyzék**

1. **Tartalomjegyzék**
2. **Bevezetés**
   1. Rövid projekt bemutatás
   2. Témaválasztási magyarázat
   3. Mi célt szolgál
   4. Funkciók és extrák
   5. Célközönség
3. **Fejlesztőkörnyezet és hardver - Fejlesztői Dokumentáció**
   1. A fejlesztéshez használt főbb eszközök
   2. Fejlesztésre használt számítógép (otthoni gép)
   3. UI/UX tervezés - Figma
   4. Technikai konzultáció
   5. Tárolt adatok
4. **Elvárások a feladattal kapcsolatban**

4.1 Operációs rendszer, környezet

4.2 Felhasználandó programozási nyelv

4.3 Megoldás formátuma

4.4 Szoftverfejlesztés

4.5 Modulok

1. **Szoftver specifikáció**

5.1 Megjelenés

5.2 Funkciók

1. **Dokumentáció**

6.1 Erőforrás-terv, munkaidő nyilvántartás

6.2 Technikai dokumentáció

6.3 Forráskód dokumentáció

6.4

1. **Adatlap**

2. Bevezetés

**2. Bevezetés**

**2.1 Rövid projekt bemutatás**

A Smart Manager egy oktatási és projektmenedzsment rendszert ötvöző komplex szoftvermegoldás, amely a tanárok és diákok közötti hatékony együttműködést segíti elő. A rendszer célja, hogy egyszerre biztosítson tanulói felületet (kurzusok, feladatok), valamint projektmenedzsment funkciókat (feladatok kiosztása, státuszkezelés, statisztikák, projektmunkák).

**2.2 Témaválasztási magyarázat**

Ez a munka elsősorban azért jött létre, mert sok tanár számolt be nehézségekről, illetve a diákok nagy száma és a különféle projektmunka-lehetőségek, valamint a közös munkát támogató platformok sokfélesége is kihívásokat jelent.

Hiszen vegyük alapul van 4 fő 2-2 fős csapatot alkotnak az egyik csapat csak githubot használ ahol követhető ki mit csinált és mikor, a másik csapat meg használt messengert, discordot és drive-ot ahol nem mindig lehet vissza követni, hogy ki mikor és mit változtatott.

**2.3 Mi célt szolgál**

A Smart Manager célja egy olyan integrált oktatási és projektmenedzsment rendszer létrehozása, amely egyszerre segíti a diákok tanulását és a tanárok adminisztrációs munkáját, átláthatóvá teszi a feladatokat, a statisztikákat, valamint támogatja a digitális tanulási környezetet.

**2.4 Funkciók és extrák**

Három fő funkciót egyesít: verziókezelés, feladatmenedzsment és valós idejű kommunikáció. A verziókezelő rész lehetővé teszi a projekthez tartozó fájlok rendszerezett tárolását és verziókövetését. A feladatmenedzsment egy Kanban tábla segítségével támogatja a csapatok munkaszervezését, a feladatok kiosztását és nyomon követését. A beépített chat modul biztosítja a gyors és egyszerű kommunikációt a csapattagok között, így minden fontos eszköz egyetlen felületen érhető el.

**2.5 Célközönség**

A platform elsődleges célközönsége a diákok és tanárok, akik egyszerű, átlátható eszközt keresnek közös projektek, beadandók és csapatfeladatok megszervezésére. Segíti a feladatok kiosztását, a közös dokumentum- és kódrendszerezést, valamint a gyors kommunikációt. Emellett a rendszer hasznos lehet kisebb fejlesztői csapatoknak, hobbi projektekhez vagy nonprofit szervezeteknek is, ahol fontos a könnyen kezelhető, integrált munkakörnyezet.

3. Fejlesztőkörnyezet és elvárások

**3. Fejlesztőkörnyezet és hardver - Fejlesztői Dokumentáció**

**3.1 A fejlesztéshez használt főbb eszközök**

IDE: Visual Studio 2022

Célplatform: Windows

Verziókezelés: Git (GitHub)

**3.2 Fejlesztésre használt számítógép (otthoni gép)**

RAM: 16GB DDR4

OS: Windows 10 64bit

GPU: AMD Radeon RX 570 8g

CPU: AMD Ryzen 5 2600

A fejlesztés főként otthon zajlott, az iskolai környezetet tervezésre használtam.

**3.3 UI/UX tervezés - Figma**

A játék felhasználói felületének és képernyőelrendezésének megtervezésére a Figma nevű online UI/UX eszközt használtuk. Ez lehetővé tette, hogy már a fejlesztés megkezdése előtt vizuálisan átgondolt terveket készítsünk a menürendszerről, szövegkörnyezetről, valamint az interaktív elemek elhelyezkedéséről. Figma miatt tudtuk elkerülni a lehetőségre okot adó hibákat hiszen mivel meg volt tervezve egy alap így tudtuk csökkenteni a felesleges újratervezések számát és a felesleges gondolkodást.

**3.4 Technikai konzultáció**

A közös munka során megbeszélésekre és egyeztetésekre a Discordot használtuk hiszen rendkívül segítőkész alkalmazás olyan téren, hogy meg tudjuk egymással osztani hívás közben a képernyőnket és élőben tudjuk követni mit csinál a másik miközben közösen agyaltunk a teendőkön.

**3.5 Tárolt adatok**

A játékhoz kapcsolódó összes változó adat, ide tartoznak többek között:

* Felhasználói adatok (felhasználónév, jelszó, email cím)
* Projektmunka (ha feltöltésre kerül, ha nem akkor nincs eltárolva semmi)
  + A feltöltött munka adatai. Pl: css, js, html, cs, xml, txt…stb.
* Valós idejű kommunikáció (Admin - User, User - User)
* Statisztika
* Aktivitás (kiírja az admin számára ki mikor volt utoljára elérhető)
* Admin
* User
* To-Do táblázat adatai (Határidő, feladat státusz: Befejezett, folyamatban, Késés, ha későn adja le a User akkor számolja mennyi ideje nem adta le)
* Jogosultságok és szerepkörök
  + pl: Admin, tanár, diák, projektvezető
  + jogosultság szint: be lehet állítani, hogy ki mit láthat és mit módosíthat
* létrejött csapat(ok) adatai
  + egy projektet több user közösen is vihet pl: 2-3 fős csapatban

4. Elvárások a feladattal kapcsolatban

**4. Elvárások a feladattal kapcsolatban**

**4.1 Operációs rendszer, környezet**

A rendszer többplatformos működésre lett tervezve.

Backend: egyedileg készített API, SQL adatbázis.

Frontend: Reszponzív web alkalmazás, amely modern böngészőkben (Chrome, Edge, Safari) és mobil eszközökön is működik.

Asztali kliens: Windows környezetre készített WinForms alkalmazás, amely közvetlenül a backend API-jához csatlakozik.

**4.2 Felhasználandó programozási nyelv**

Backend: C#

Frontend: JavaScript / HTML / CSS

Asztali kliens: C# (Windows Forms)

Adatbázis: SQL (MySQL)

Valós idejű kommunikáció: Websocket (Socket.IO)

**4.3 Megoldás formátuma**

A végső megoldás három komponensből áll:

1.Webes frontend – diákok számára (kurzusok, projektek kezelése).

2.Backend REST API – a központi logika és adattárolás.

3.Asztali kliens – tanári felület (kurzusok, statisztikák kezelése).

A teljes forráskód GitHub repóban kerül tárolásra, verzió követéssel és dokumentációval együtt.

**4.4 Szoftverfejlesztés**

A fejlesztés során Visual Studio-t és Figma-t használunk, a kommunikáció Discordon zajlik. A tiszta kód elveit követjük, és folyamatosan teszteljük a rendszer stabilitását.

**4.5 Modulok**

1. Felhasználókezelés (bejelentkezés, regisztráció, jogosultságok).
2. Kurzuskezelés tanárok számára (kurzusok létrehozása, megtekintése, szerkesztése).
3. Kurzuskezelés diákok számára (megtekintése, kitöltés és visszaküldés).
4. Projektfeladat-kezelés (Kanban board, státuszok).
5. Statisztikák és riportok (átlagpontszám, legjobb diák, export).
6. Adminisztrációs felület (tanári desktop kliens).

5. Szoftver specifikáció

**5. Szoftver specifikáció**

**5.1 Megjelenés**

**5.2 Funkciók**

6. Dokumentáció

**6. Dokumentáció**

### **6.1 Erőforrás-terv, munkaidő nyilvántartás**

Csapatlétszám: 2 fő fejlesztő.

Munkamegosztás: egyik fő a backend + desktop fejlesztésért felel, a másik fő a frontend + UI megvalósításáért.

Munkaidő: heti 8–10 óra fejenként, összesen kb. 150–160 munkaóra.

Ütemezés: 8 hét, heti mérföldkövekkel (lásd Gantt-diagram).

### **6.2 Technikai dokumentáció**

Adatbázismodell: ERD diagram, SQL dump.

Rendszerarchitektúra: backend, frontend és desktop komponensek kapcsolatának leírása.

API dokumentáció: Swagger/OpenAPI specifikáció a végpontokról.

### **6.3 Forráskód dokumentáció**

Kommentek: tiszta kód elvek, metódusonkénti rövid leírás.

README fájl: telepítési és futtatási útmutató.

Kódfelépítés: modulok és mappastruktúra bemutatása.

Tesztdokumentáció: futtatott tesztek és eredmények.

7. Adatlap

**7. Projekt adatlap**

Projekt neve: Smart Manager

Feladat rövid ismertetése: Projekt menedzser ,mely kiszolgál egy teljes projekt folyamat lebonyolításához szükséges minden igényt.

Programozási nyelvek: C#..

Specifikációt összeállította: Magyar Márk József, Nagy Huba Kende

+

minta

## **3. Elvárások a feladattal kapcsolatban**

### **3.1 Operációs rendszer, környezet**

A rendszer többplatformos működésre lett tervezve.

* Backend: Linux és Windows szerveren egyaránt futtatható (ASP.NET Core + SQL alapú adatbázis).
* Frontend: Reszponzív webalkalmazás, amely modern böngészőkben (Chrome, Edge, Firefox) és mobil eszközökön is működik.
* Asztali kliens: Windows környezetre készített WinForms/WPF alkalmazás, amely közvetlenül a backend API-jához csatlakozik.

### **3.2 Felhasználandó programozási nyelv**

* Backend: C# (ASP.NET Core Web API)
* Frontend: JavaScript (React keretrendszer)
* Asztali kliens: C# (Windows Forms vagy WPF)
* Adatbázis: SQL (MySQL vagy PostgreSQL)

### **3.3 Megoldás formátuma**

A végső megoldás három komponensből áll:

1. Webes frontend – diákok számára (kurzusok, projektek kezelése).
2. Backend REST API – a központi logika és adattárolás.
3. Asztali kliens – tanári felület (kurzusok, statisztikák kezelése).

A teljes forráskód GitHub repóban kerül tárolásra, verziókövetéssel és dokumentációval együtt.

### **3.4 Szoftverfejlesztés**

* Fejlesztési modell: agilis módszertan (Scrum/Kanban alapokon).
* Eszközök: GitHub/GitLab, Trello/Jira a feladatkövetéshez.
* Tesztelés: Unit tesztek (xUnit, Jest), manuális funkcionális tesztek.
* CI/CD: automatikus build és teszt futtatás GitHub Actions segítségével.

### **3.5 Modulok**

1. Felhasználókezelés (bejelentkezés, regisztráció, jogosultságok).
2. Kurzuskezelés (kurzusok létrehozása, megtekintése, szerkesztése).
3. Projektfeladat-kezelés (Kanban board, státuszok).
4. Statisztikák és riportok (átlagpontszám, legjobb diák, export).
5. Adminisztrációs felület (tanári desktop kliens).

## **4. Szoftver specifikáció**

### **4.1 Megjelenés**

* Frontend: modern, reszponzív felhasználói felület (React + CSS/Bootstrap).
  + Diák dashboard: kurzuslista, projektboard.
  + Reszponzív design: működik asztali gépen és mobilon is.
* Asztali kliens: Windows Forms/WPF alapú, egyszerű, gyors kezelőfelület a tanároknak.
* Egységes arculat: kék-fehér színvilág, átlátható ikonok, intuitív menük.

### **4.2 Funkciók**

* Diákoknak:
  + Regisztráció, bejelentkezés.
  + Kurzusok megtekintése, kitöltése.
  + Projektfeladatok kezelése Kanban board nézetben.
  + Saját statisztikák, eredmények megtekintése.
* Tanároknak:
  + Új kurzusok, feladatok létrehozása.
  + Diákok teljesítményének nyomon követése.
  + Statisztikák és riportok megjelenítése, exportálása.
* Rendszerszinten:
  + REST API az adatokhoz.
  + Jogosultságkezelés (tanár/diák szerepkörök).
  + Adatbázis-alapú adattárolás.

## 

## **5. Dokumentáció**

### **5.1 Erőforrás-terv, munkaidő nyilvántartás**

* Csapatlétszám: 2 fő fejlesztő.
* Munkamegosztás: egyik fő a backend + desktop fejlesztésért felel, a másik fő a frontend + UI megvalósításáért.
* Munkaidő: heti 8–10 óra fejenként, összesen kb. 150–160 munkaóra.
* Ütemezés: 8 hét, heti mérföldkövekkel (lásd Gantt-diagram).

### **5.2 Technikai dokumentáció**

* Adatbázismodell: ERD diagram, SQL dump.
* Rendszerarchitektúra: backend, frontend és desktop komponensek kapcsolatának leírása.
* API dokumentáció: Swagger/OpenAPI specifikáció a végpontokról.

### **5.3 Forráskód dokumentáció**

* Kommentek: tiszta kód elvek, metódusonkénti rövid leírás.
* README fájl: telepítési és futtatási útmutató.
* Kódfelépítés: modulok és mappastruktúra bemutatása.
* Tesztdokumentáció: futtatott tesztek és eredmények.