WallSpace Project – Målpinde status

1. [1] ? Kan ikke besvares før alle andre målepinde er opfyldt.
2. [3] Hele projektet
3. [3] C# OOP Backend > Models
4. [3] C# OOP Backend > Controller > AuthController > Linje 11-16 (private field og instans)
5. [3] C# OOP Backend > Models > FriendList > Linje 5. Her bruger vi List<>. Der findes også flere, men ikke noget vi bruger.
6. [3] Hele projektet
7. [2] SignalR DB Backend > chatplatform-wallspace > wallspace > Chathub.cs > Linje 197-212
8. [2] SignalR DB Backend > chatplatform-wallspace > wallspace > Chathub.cs > Linje 16 = Arv. Linje 18 = Indkapsling. AppDbContext.cs > Linje 4 = Polymorfi.
9. [3] C# OOP Backend > Models > Hvilken som helst model
10. [2] En abstrakt klasse kan ikke instantieres, men fungerer som en skabelon for andre klasser. Den kan have abstrakte metoder, som kræver, at subklasser implementerer deres egen version, og konkrete metoder, som allerede er implementeret. (Vi forstår hvad det er, men bruger det ikke i vores projekt)
11. [2] Override bruges til at ændre adfærden af en metode i en baseklasse, mens overload bruges til at definere flere metoder med samme navn, men forskellige parameterlister.
12. [3] C# OOP Backend > Models > FriendList > Linje 5. Her bruger vi getter/setter, det er vores primære metode til at kontrollere access. I det her tilfælde er det en public get og en private set, da det ikke er alle som skal kunne ændre ens venneliste.
13. [2] Et interface definerer metoder, som en klasse skal implementere, men indeholder ikke nogen en implementation. En klasse kan implementere flere interfaces og er påkrævet at bruge de metoder, som interfacet indeholder. (Vi forstår hvad det er, men bruger det ikke i vores projekt)
14. [2] Vi vælger at bruge promises i stedet for callbacks/pointers, fordi promises er den moderne og anbefalede metode til asynkrone operationer i JS. Det giver bedre struktur, enklere fejlhåndtering og undgår callback hell.

15. [2] Det er ikke noget vi umiddelbart bruger, men generics gør koden genanvendelig. Fx List<T> kan bruges til både int, string osv.

16. [2] Har lavet noget, skal tjekkes med Kim om det gælder som “UML”.

17. [2] Vi har designet en simpel domænemodel, hvor vi deler ansvaret i klasser såsom User, Chatroom og Message osv. Hver klasse har nogle egenskaber og relationer, så koden er struktureret og let at vedligeholde. Koden er baseret på OOP-konceptet (det er trods alt faget).

18. [3] C# OOP Backend > Controllers > AuthController > Linje 15. Her skal der bruges en injection for at gøre det mere fleksibelt, så det bliver løs kobling. På linje 19 bliver der også brugt et interface.

19. [3] Versionsstyringsværktøjer bruges til at holde styr på kodeændringer og muliggør samarbejde mellem udviklere. Det giver mulighed for at se, hvem der har lavet hvad, og at rulle tilbage til tidligere versioner, hvis der opstår problemer.

20. [3] Dokumentation sikrer, at både udviklere og brugere forstår systemets funktionalitet og struktur. Det gør koden nemmere at vedligeholde, videreudvikle og fejlfinde.

21. [3] Frontend > Obj. Ana. - Gruppe Chat > joachim-kode > CSS > style.css > Kommentarer hist og pist. Kommentarerne er simple arbejds-kommentarer, der hjælper udviklere med hurtigt at forstå, hvad de forskellige dele af koden gør.

22. [3] https://github.com/Spr1zze/WallSpace-Frontend > Der er blevet brugt GitHub.

23. [1] X

24. [1] X

25. [1] X

26. [1] X

27. [2] Kan bruges i ValidateLogin-metoden i AuthService backend.

28. [2] Anonyme metoder og lambda-metoder gør det muligt at skrive små funktioner direkte i koden uden at give dem et navn, hvilket gør koden kortere og nemmere at læse. (Vi forstår hvad det er, men bruger det ikke i vores projekt)