📑 Sommario per il progetto finale (App Padel)

1. Requisiti generali del corso

- Architettura **client/server** con pattern **MVC**.
- Uso di **almeno 2 design pattern documentati** → noi useremo **Observer, Strategy, Singleton**.
- **Programmazione a oggetti** (Java consigliato, classi, ereditarietà, polimorfismo).
- **Test unitari** con coverage ≥ 80% (JUnit + Jacoco).
- **Documentazione**: requisiti, UML, piano test, coverage.
- **Limite pagine**: max 40 (1 persona).
- Repository GitHub/GitLab opzionale (ma utile).

---

2. Requisiti funzionali (cosa deve fare l'app)

Utenti

- Registrazione/login.
- Scelta livello dichiarato: principiante / intermedio / avanzato / professionista.
- Visualizzare il proprio profilo: livello dichiarato, livello percepito (media feedback), partite giocate.

Partite

- **Partite fisse**: già programmate, con livello richiesto.
- **Partite proposte**: create dagli utenti, con data/ora/luogo/livello richiesto.

- Iscrizione e cancellazione a una partita.
- Conferma automatica quando si raggiungono 4 giocatori.
- Cambio stato partita a **FINISHED** dopo la data.

Feedback

- Dopo una partita conclusa:
    - Ogni utente può lasciare feedback su altri giocatori.
    - Feedback = livello suggerito + eventuale commento.
    - Il sistema aggiorna il livello percepito del giocatore.

Filtri & Ricerca

- Visualizzare lista partite:
    - ordinata per data, popolarità (# iscritti), o livello richiesto (**Strategy**).
    - filtrare per livello richiesto.

Notifiche

- Ricevere notifica quando:
    - una partita viene confermata,
    - una partita finisce (ricorda di lasciare feedback).
- Implementate con **Observer** + gestite centralmente dal **Singleton NotificationService**.

---

3. Requisiti non funzionali

- Usabilità: interfaccia semplice (liste, pulsanti join/leave).
- Prestazioni: gestione partite in tempo quasi reale.

- Sicurezza: password cifrate, accesso solo autenticato.
- Manutenibilità: separazione MVC + uso pattern.
- Testabilità: unit test su service/controller.

---

4. Componenti principali (MVC)

Model

- `User` (con livello dichiarato, livello percepito, matchesPlayed).
- `Match` (tipo fissa/proposta, livello richiesto, status, data/ora, luogo).
- `Registration` (legame user–match, stato join/cancel).
- `Feedback` (livello suggerito, commento, autore).

Controller

- `AuthController` (login, registrazione).
- `MatchController` (creazione proposta, lista, join, leave, gestione stato).
- `FeedbackController` (inserimento feedback, calcolo media).
- `UserController` (profilo).

View

- Login/registrazione.
- Lista partite con filtri.
- Dettaglio partita con iscritti e bottone join/leave.
- Pagina profilo con statistiche e feedback.

---

5. Pattern di progettazione (obbligatori)

- **Observer**: per le notifiche automatiche (partita confermata, partita finita).
- **Strategy**: per il criterio di ordinamento partite (data, popolarità, livello).
- **Singleton**: per il `NotificationService`, unico punto di gestione notifiche.

---

6. Testing

- **Unit test (JUnit)**:
    - `MatchService`: creazione, conferma automatica, join/leave con limite 4.
    - `FeedbackService`: aggiunta feedback, calcolo media livelli.
    - `UserService`: incremento partite giocate.
    - `Observer`: verifica che i listener ricevano l'evento.
- **Controller test (MockMvc)**: almeno 1–2 per endpoint principali.
- **Coverage**: generare report Jacoco ≥ 80% e inserirlo nella documentazione.

---

7. UML richiesto

- **Use Case Diagram**: attori = utente registrato, utente guest (registrazione), admin opzionale.
- **Class Diagram**: modelli principali + pattern.
- **Sequence Diagram**: almeno 2 →
    1. Conferma automatica partita → notifica iscritti.
    2. Partita finita → feedback utenti → aggiornamento livello percepito.

---

8. Documentazione finale (indice consigliato)

1. Introduzione e contesto.
2. Requisiti (funzionali + non funzionali).
3. Architettura MVC.
4. Pattern usati (Observer, Strategy, Singleton).
5. UML (Use Case, Class, 2 Sequence).
6. Implementazione (tecnologie, endpoints principali).
7. Piano test e risultati (con coverage).
8. Conclusioni ed estensioni possibili.

👉 Con questo sommario hai **tutto ciò che serve per coprire i requisiti del prof**.
Il prossimo passo sarà decidere se vuoi che ti prepari:

- la **lista dettagliata dei requisiti funzionali in formato ufficiale**, oppure
- iniziare direttamente dai **diagrammi UML** (Use Case come primo).

Vuoi che partiamo dai **requisiti scritti in forma ufficiale** (tipo documento di progetto), così hai subito un testo pronto da mettere nella tesi?