Финальный проект — MERN 2

1) Обязательные условия

- **Технологический стек строго фиксирован:** MongoDB, Express.js, React/**Next.js**, Node.js (MERN).
- **Язык и инструменты: TypeScript** везде (frontend + backend), **GraphQL** (Queries/Mutations/Subscriptions), **Jest**, **Docker**/**Docker Compose**, на фронте — **Next.js + TailwindCSS + Zustand + Apollo Client**.
- **Совместимость с демо:** Проект должен **запускаться из** `docker-compose up` (одно действие).
- **Антиплагиат:** Любое списывание → **0 баллов**.
- **Демонстрация:** Если нельзя запустить/показать рабочую версию → **0 баллов**.

---

2) Функциональные требования

2.1 Модели данных

- Минимум **4 доменные модели**, у каждой **минимум 5 полей** (не считая `_id`, `createdAt`, `updatedAt`).
- Поля должны иметь **строгие типы** и **валидации** (на уровне Mongoose схем и/или zod/yup на входе).
- Минимум **две связи** между моделями (1-ко-многим или многие-ко-многим).
- Рекомендуется: **soft delete** (`isDeleted`) или архивирование там, где логично.

2.2 Бэкенд (GraphQL API)

- **Схема**: минимум 6 Query, 6 Mutation, **1+ Subscription**.
- **Аутентификация/авторизация**: **JWT** (access-token), приватные резолверы закрыты.
- **Ошибки**: единый формат ошибок (GraphQL error extensions + коды).
- **Валидация входных данных**: до резолвера (middleware/pipe) и/или внутри резолверов.
- **Тесты (Jest)**:
  - Unit на резолверы/сервисы, минимум **10 тестов**;
  - Минимум **1 интеграционный** (in-memory MongoDB или test DB).
- **Конфигурация**: `NODE_ENV`, `MONGO_URI`, `JWT_SECRET` и др. через `.env`; есть `env.example`.
- **Сеединг**: скрипт `yarn seed` или аналог — для быстрой локальной проверки.

2.3 Фронтенд (Next.js App Router)

- Структура **App Router** (папка `app/`).
- **Apollo Client** с типами и кэшем; работа с **GraphQL Subscriptions**.
- **Zustand**: централизованное состояние (auth/session/UI или доменное).
- UI на **TailwindCSS**, адаптивная верстка, базовая **доступность (a11y)**.
- Формы с валидацией (react-hook-form/zod), предзагрузка/оптимистичные сценарии там, где уместно.
- Минимум 5 ключевых экранов (список, карточка, создание/редактирование, профиль/логин и т.п.).

---

3) Архитектура и качество кода

3.1 Кодстайл и инструменты

- **TSConfig** строгий (`"strict": true`), алиасы путей.
- **ESLint + Prettier**, Husky + lint-staged (по желанию).
- Коммиты в стиле Conventional Commits (желательно).

---

4) Развертывание (DevOps)

4.1 Docker/Compose (обязательно)

- Отдельные `Dockerfile` для **client** и **server**.
- `docker-compose.yml` поднимает: **api**, **client**, **mongo**, (опционально **mongo-express**).
- Healthchecks для сервисов. Один вход: `docker-compose up`.

4.2 Environments

- `.env.example` (оба пакета) — полный список переменных.
- Секреты не коммитятся.

4.3 Продакшн-демо (сильно рекомендуется)

- **Ссылки для проверки** в README:
    - `Frontend URL (prod)`
    - `GraphQL endpoint (prod)`
    - `WS endpoint for subscriptions`

---

5) Реалтайм-функциональность (обязательно ≥1)

- **Сквозная Subscription**: событие создаётся/обновляется на сервере → **моментально** отражается на фронте.
- Транспорт: **graphql-ws** или **Socket.IO** (через ApolloLink).
- README содержит **шаги для проверки** (как воспроизвести событие).

Примеры: онлайн-статусы, лайвы/комменты, чат, уведомления, прогресс задач/заказов, табло.

---

6) Документация

6.1 README (обязательно)

- **Описание проекта** (цели, домен, роли пользователей).
- **Схема данных** (кратко: модели, связи).
- **Как запустить локально** (1-2 шага с Docker).
- **Как проверить реалтайм** (пошагово).
- **Демо-ссылки** (prod/stage), тестовый пользователь.
- **Скрипты** (`dev` , `test` , `seed` , `lint`).
- **Команда/Роли**, кто что делал.

6.2 Презентация (для защиты)

- 6–10 слайдов: проблема → решение → архитектура → демо → реалтайм → тесты → выводы.

---

7) Команда и процесс

- **1 проект = 2 студента**.
- Оба **разбираются во всём** и могут объяснить любую часть.
- История коммитов отражает вклад (не один автор на всё).
- В README раздел «Вклад» (по пунктам/фичам на каждого).

---

8) Критерии оценивания (100 баллов)

8.1 Бэкенд — 45

- Node.js — **10** (структура, слои, конфиги, логирование)
- Express.js — **5** (middleware, ошибки, CORS, безопасность)
- MongoDB — **5** (схема БД, индексы, связи, запросы)
- TypeScript — **5** (строгие типы, типобезопасность в резолверах)
- Mongoose — **5** (валидации, хуки/плагины, lean(), агрегации где нужно)
- GraphQL — **5** (схема, N+1 контроль, DataLoader при необходимости)
- Jest (тесты) — **5** (покрытие критичных путей)
- Subscriptions — **5** (рабочий e2e поток)

> Итого по бэкенду: **45**

8.2 Фронтенд — 25

- Next.js — **5** (App Router)
- TailwindCSS — **5** (чистая, адаптивная верстка)
- Zustand — **5** (правильное разделение стора, селекторы)
- Дизайн/UI — **5** (структура, UX, аккуратность)
- Docker — **5** (корректный билд и запуск client)

> Итого по фронтенду: **25**

8.3 Общее — 70 (сумма блоков 8.1 + 8.2)

8.4 Индивидуальные баллы — 30 (каждому)

- Ясность объяснений, понимание кода, ответы на уточняющие вопросы.
- Возможность завершения собственного кода после удаления строки
- Защита кода

Всего: 100

---

9) Процедура защиты (15–20 минут)

1. **Коротко о продукте** (30–60 сек: кто пользователь/ценность).
2. **Архитектура** (1–2 мин: диаграмма модулей, потоки данных).
3. **Демо** (5–7 мин: ключевые сценарии + реалтайм с явной провокацией события).
4. **Тесты** (1–2 мин: что покрыто, как запускать, пример отчёта).
5. **Q&A** (оставшееся время).

Экзаменатор вправе попросить: открыть любой файл, объяснить резолвер, показать запрос в сети, изменить запись и увидеть событие в подписке, запустить тест.

---

10) Чек-лист перед сдачей

- ☐ `docker-compose up` поднимает **client**, **server**, **mongo**, **ws**.
- ☐ В README есть **URL продакшна** + тестовые креды.
- ☐ GraphQL schema: ≥6 Query, ≥6 Mutation, ≥1 Subscription.
- ☐ 4+ модели, 5+ полей каждая, связи и индексы на месте.
- ☐ JWT-auth работает, приватные резолверы закрыты.
- ☐ 10+ Jest тестов + ≥1 интеграционный.
- ☐ Реалтайм подписка проверяется пошагово (описание в README).
- ☐ Next.js App Router, Apollo, Zustand, Tailwind — корректно используются.
- ☐ `.env.example` присутствует, секреты не закоммичены.
- ☐ История коммитов отражает вклад **обоих** участников.

---

11) Формат сдачи

- **Ссылка на GitHub** (monorepo `client/` + `server/` или 2 репо).
- **README с демо-ссылками** и шагами запуска.
- **Презентация (PDF/PPTX)** — приложить ссылку в README.

Будьте готовы к финальному проекту и удачи!