## **Техническое задание**

### **Проект «SkillSwap» — платформа обмена навыками «Я научу / Хочу научиться»**

### **1. Название и краткое описание**

**SkillSwap** — одностраничное (SPA) приложение, в котором пользователи публикуют навыки двух типов:

* **“Учу”** — навыки, которыми пользователь готов делиться;
* **“Учусь”** — навыки, которым пользователь хочет научиться.

Сервис позволяет находить взаимно подходящие пары, отправлять заявки на обмен и вести список текущих/завершённых сессий. Все данные берутся из JSON-моков; постоянные состояния (избранное, заявки, настройки) — в localStorage.

### **2. Пользовательский сценарий (User Flow)**

| **№** | **Действие** | **Результат / Условие завершения** |
| --- | --- | --- |
| 1 | Гость открывает главную страницу | Отображается каталог навыков, кнопки «Войти» и «Регистрация» |
| 2 | Поиск/фильтрация | Список обновляется без перезагрузки |
| 3 | Открывает карточку навыка | Переходит на URL /skill/:id; отображается описание, автор, кнопка «Предложить обмен» |
| 4 | Нажимает «Предложить обмен» | Если не авторизован → редирект на /login с возвратом после успешного входа |
| 5 | Автор навыка принимает/отклоняет заявку | Заявке присваивается статус «В процессе» или «Отклонена»; оба участника получают toast-уведомление |
| 6 | Участники завершают сессию | Любой может нажать «Завершить» → статус «Завершена», дата окончания фиксируется |
| 7 | Пользователь создает новый навык | Заполняет форму /create; валидация полей (см. п. 3.8); карточка появляется в каталоге без перезагрузки |
| 8 | Пользователь добавляет навык в избранное | Кнопка-тоггл на карточке, хранение в localStorage; на странице /favorites виден список избранного |

### **3. Обязательные задачи (MVP) — расчёт на 3 недели**

| **ID** | **Фича / модуль** | **Подзадачи (issue-level)** | **Готово, когда…** |
| --- | --- | --- | --- |
| 3.1 | **CI + репозиторий** | Init Vite + React/TypeScript; ESLint (+ Airbnb), Prettier, Stylelint | npm run build и npm run lint проходят без ошибок |
| 3.2 | **Каталог навыков** | a) Скелет UI;  b) загрузка /db/skills.json; c) бесконечный скролл ИЛИ пагинация | При прокрутке догружается ≥ 20 карточек за раз |
| 3.3 | **Поиск и фильтры** | a) Инпут поиска;  b) select категории;  c) radio-группа «Учу / Учусь / Все» | Фильтры комбинируются логикой **AND** |
| 3.4 | **Страница навыка** | a) Рендер данных;  b) кнопка «Предложить обмен»;  c) related-блок до 4 похожих | URL /skill/:id отрабатывает прямой заход |
| 3.5 | **Аутентификация** | a) Форма login/registration;  b) mock хранится в localStorage | После логина происходит редирект на предыдущий URL |
| 3.6 | **Заявки на обмен** | a) Создание/отклонение/принятие;  b) хранение в localStorage | В профиле отображаются заявки с корректными статусами |
| 3.7 | **Профиль пользователя** | a) Таблица «Мои навыки»;  b) «Мои заявки» | Переключение вкладок без перезагрузки |
| 3.8 | **Форма создания навыка** | Поля: title (3-50 симв.), description (≤ 500), type (radio), category (select), image (file ≤ 2 MB, jpeg/png), tagging (≤ 5 тегов) | Инвалидные поля подсвечены; submit заблокирован, пока есть ошибки |
| 3.9 | **Избранное** | a) Toggl-кнопка на карточке;  b) список /favorites | Пустое состояние «Вернуться в каталог» |
| 3.10 | **Роутинг + 404** | React Router ≥ v6, lazy-routes | Несуществующий URL → страница 404; кнопка «Домой» |

### 

### **4. Дополнительные задачи (~1 неделя)**

| **Приоритет** | **Функция** | **Краткие критерии приёмки** |
| --- | --- | --- |
| Высокий | Тёмная тема | Тоггл, CSS-variables, автосохранение выбора |
| Средний | Экспорт сессии в ICS | Кнопка «Добавить в календарь» → скачивается .ics с датой и именами участников |
| Средний | Бейджи/достижения | Enum бейджей; появление toast-уведомления при выдаче |
| Низкий | Внутренний чат-заглушка | WebSocket-мок; хранение истории в localStorage |
| Низкий | PWA-офлайн | Service Worker; offline-fallback страница |

### **5. Архитектура и технологии**

| src/  ├── api/ # методы работы с мок-JSON (axios/fetch)  ├── app/ # инициализация, провайдеры, глобальные стили  ├── entities/ # модели домена (Skill, User, Request)  ├── features/  │ ├── auth/  │ ├── skills/  │ ├── favorites/  │ └── requests/  ├── widgets/ # готовые фич-блоки (SkillCard, FiltersBar)  ├── pages/ # главная, профайл, skill, favorites  ├── shared/  │ ├── ui/ # атомы/молекулы  │ ├── hooks/ # useDebounce, useLocalStorage ...  │ └── lib/ # helpers, constants  └── index.tsx public/  db/  ├── skills.json  └── users.json |
| --- |

**State-manager**: Context API + useReducer *или* Redux Toolkit (на усмотрение команды).

**Тесты**: Jest + React Testing Library; цель — ≥ 70 % statements покрытия ядра.  
**Branching**: Trunk Based Flow (main → feature/\* → PR → merge); линт + юниты в CI.

**Коммит-мессаджи**: Conventional Commits.

### 

### **6. План спринтов (рекомендация)**

| **Неделя** | **Цели** | **Контрольные точки (demo)** |
| --- | --- | --- |
| **1** | Инфраструктура, каталог, поиск | Главная рендерит карточки из JSON; фильтры обновляют список |
| **2** | Аутентификация, профиль, заявки | Логин/регистрация, создание/принятие заявки |
| **3** | Избранное, создание навыка, адаптив | Форма создания навыка; избранное; mobile-view |
| **4** | Доп. фичи, тесты, баг-фиксы | Тёмная тема + любые extra-функции; unit-coverage ≥ 70 % |

### **7. Распределение ролей (примерная матрица)**

| **Команда** | **Основные зоны ответственности** |
| --- | --- |
| TL (1) | Архитектура, code-review, CI |
| PM (1) | Канбан, daily, дедлайны |
| Frontend-A (2) | Каталог, фильтры, карточка |
| Frontend-B (2) | Аутентификация, профиль |
| Frontend-C (2) | Заявки, избранное |
| Frontend-D (2) | Форма создания, валидация |
| QA/Docs (1) | Тесты, Storybook, README |

Ротация по окончании каждого спринта приветствуется, но на усмотрение команды)

### **8. Критерии приёмки проекта в целом**

* Все задачи MVP отмечены «Done» в борде, PR смержены в main, деплой доступен по публичной ссылке.
* npm-скрипты build, lint, test завершаются без ошибок.
* При мануальном тесте заявленный User Flow выполняется полностью.
* Lighthouse performance ≥ 80 на desktop, accessibility ≥ 90.
* Покрытие юнит-тестами ядра ≥ 70 % (statements).

## **Этап 0. Организация и знакомство (D-0 → D-1)**

| **Задача** | **Ответственные** | **Что считается «готово»** |
| --- | --- | --- |
| Онбординг 1 ч | Наставник | Команда понимает цель, роли, макеты в Figma открыты |
| Настройка репо | TL | GitHub + мейн ветка, README с «как поднять» |
| Check-out окружения | все | npm i && npm run dev запускает пустую страницу |

## **Этап 1. Инфраструктура и «живой» каталог (D-2 → D-5)**

**Цель недели:** открыть сайт — увидеть список тестовых навыков, отфильтровать их, и ничего не упадёт.

| **День** | **Что делаем** | **Почему это важно** |
| --- | --- | --- |
| D-2 | подключаем ESLint/Prettier/Stylelint | сразу учим «красная сборка ≠ push» |
| D-2 | кладём skills.json в public/db | база для мок-API |
| D-3 | пишем первую компоненту **SkillCard** (заглушка) | команда видит первую визуальную победу |
| D-3 | делаем **Grid/List** на главной | учимся работать с макетом, CSS Grid |
| D-4 | добавляем бесконечный скролл **или** пагинацию | первая совместная логика + загрузка |
| D-5 | внедряем поиск (debounce) + селектор категории | сразу обкатываем state-management |

**Демо пятницы:** каталожная страница фильтруется без перезагрузки.

## **Этап 2. Аутентификация и личный кабинет (D-6 → D-10)**

| **Блок** | **Майлстоуны** | **Подсказка** |
| --- | --- | --- |
| Auth | форма Login / Sign-Up  mock-JWT — просто строка «token123» | используйте react-hook-form + yup — меньше кода, больше валидации |
| Guard Routes | PrivateRoute | любое открытие /profile без токена → redirect |
| Profile v0 | вкладка «Мои навыки» (пустая) | пока выводим «+ Добавить навык» кнопку |

**Демо понедельника (D-10):** можешь залогиниться фейковым email, перейти на профиль и обратно.

## **Этап 3. Заявки и страница навыка (D-11 → D-14)**

1. **SkillPage**
   * загрузка по id, кнопка «Предложить обмен»
   * related 4 карточки (фильтр по категории)
2. **Requests API (localStorage)**
   * модель { id, skillId, fromUserId, toUserId, status }
   * статусы enum: pending / accepted / rejected / inProgress / done
3. **Inbox/Outbox вкладки в профиле**
   * карточка заявки с action-кнопками

Быстрый лайфхак хранения

| // hooks/useLocalList.ts export const useLocalList = <T>(key: string, initial: T[]) => {  const [list, set] = useState<T[]>(() => {  const raw = localStorage.getItem(key);  return raw ? JSON.parse(raw) : initial;  });  useEffect(() => localStorage.setItem(key, JSON.stringify(list)), [list]);  return [list, set] as const; }; |
| --- |

**Демо пятницы (D-14):** два аккаунта создают заявку, автор её принимает.

## **Этап 4. Создание навыка + избранное + адаптив (D-15 → D-18)**

| **Под-этап** | **Ключевые acceptance-критерии** |
| --- | --- |
| Форма /create | валидация inline, превью загруженной картинки |
| Избранное | toggler ❤️ на карточке, страница /favorites |
| Адаптив | mobile 320px: единичная колонка, бургер-меню |

**Демо вторника (D-18):** на телефоне можно добавить навык в избранное, открыть детальную, вернутьcя назад.

## **Этап 5. Полировка и доп-фичи (D-19 → D-22)**

* Тёмная тема → CSS vars + [data-theme]
* Lighthouse pass — поджать картинки, lazy loading
* Unit-cover 70 % — тестируем **hooks** и **reducers**
* PWA (если успеваем) — vite-plugin-pwa за 20 минут

## **Этап 6. Freeze, баг-фиксы, релиз (D-23 → D-24)**

| **День** | **Действие** |
| --- | --- |
| D-23 | QA-чек-лист: весь флоу от гостя до «завершить» |
| D-23 | Code Freeze 18:00 — только крит-фиксы |
| D-24 | Финальный Lighthouse, gh-pages redeploy |
| D-24 | Демо наставнику + ретроспектива |

## **Как не дать команде «утонуть»**

1. **Правило 1 страницы PR.** Любой ревью ≤ 200 строк.
2. **Каждый день — видимую фичу.** Даже простейший toast-уведомитель — уже победа.
3. **Ментор-“коуч” вместо «старшего».** Помогает задавать вопросы, не пишет за студентов.
4. **Daily demo, не daily stand-up.** Покажи кнопку, а не расскажи, что «пилишь кнопку».
5. **Сложные задачи дробятся до «можно сделать за полдня».** Если нет — TL делит ещё.