# Requirement Decomposition Documentation

## Systematically break down and document the project requirements for clarity and manageability.

### Эпик 1: Управление учетной записью и доступом пользователя

Этот эпик охватывает все, что связано с регистрацией, авторизацией и управлением учетной записью.

#### Юзер-стори 1.1: Создание новых пользователей

Как админ пользователь, я хочу иметь возможность создавать аккаунты, имеющие доступ к системе.

Таски:

1. Создание интерфейса регистрации:
   * Спроектировать и создать форму для создания аккаунтов.
   * Добавить поля для ввода данных (имя пользователя, пароль).
2. Валидация и обработка данных:
   * Добавить валидацию для обязательных полей (проверка длины пароля).
   * Настроить проверку на уникальность имени пользователя.
3. Хранение данных пользователя:
   * Настроить базу данных для хранения данных пользователей.
   * Реализовать безопасное хэширование паролей.
4. Создание админа:
   * Создать скрипт с миграцией базы данных с добавлением админ пользователя в систему

#### Юзер-стори 1.2: Авторизация пользователя

Как зарегистрированный пользователь, я хочу иметь возможность войти в систему, чтобы получить доступ к личным функциям, с аутентификацией через JWT.

Таски:

1. Создание интерфейса входа в систему:
   * Спроектировать и создать форму авторизации.
   * Добавить поля для логина и пароля.
2. Обработка данных входа:
   * Проверить ввод логина и пароля (с учетом регистра).
   * Настроить выдачу сообщений об ошибке (неверный пароль, несуществующий аккаунт).
3. Аутентификация с JWT:
   * Сгенерировать JWT-токен при успешной авторизации пользователя.
   * Настроить выдачу JWT-токена и обновляемого токена (рефреш-токена) для длительных сеансов.
   * Настроить хранение и использование токена на клиенте (в localStorage).
4. Логирование и защита:
   * Логировать попытки входа и выхода для безопасности.

#### 

#### Юзер-стори 1.3: Выход из учетной записи

Как авторизованный пользователь, я хочу иметь возможность выйти из учетной записи, чтобы обезопасить свои данные.

Таски:

1. Реализация функции выхода:
   * Спроектировать и создать кнопку выхода.
   * При выходе удалить хранимые токены JWT с клиента (localStorage).
2. Завершение токенов:
   * Обновить статус токена на сервере, добавив его в список отозванных токенов или используя их короткий срок действия (например, 15 минут для access-токена).
   * Реализовать механизм проверки, чтобы отозванные или просроченные токены отклонялись при обращении к API.
3. Уведомление о выходе:
   * Добавить уведомление, подтверждающее успешный выход пользователя и завершение доступа к системе.

### Эпик 2: Управление вакансиями

Этот эпик охватывает все, что связано с вакансиями: создание, закрытие, удаление, просмотр и управление кандидатами на воронке внутри вакансии.

#### Юзер-стори 2.1: Просмотр доступных вакансий

Как авторизованный пользователь, я хочу просматривать вакансии, чтобы ознакомиться с доступными рабочими местами.

Таски:

1. Создать интерфейс списка вакансий, отображающий основные данные.
2. Настроить фильтры по “открытой”/“закрытой” вакансии и сортировку вакансий по дате создания.
3. Реализовать постраничный вывод вакансий.

#### Юзер-стори 2.2: Создание новой вакансии

Как авторизованный пользователь, я хочу добавить вакансию, чтобы найти сотрудников для компании.

Таски:

1. Создать форму для добавления вакансии.
2. Кнопка для создания вакансии с переходом на форму
3. Настроить валидацию обязательных полей.
4. Сохранить вакансию в базе данных.

#### Юзер-стори 2.4: Закрытие вакансии

Как авторизованный пользователь, я хочу закрыть вакансию, чтобы обозначить, что поиск кандидатов завершен.

Таски:

1. Добавить кнопку "Закрыть вакансию".
2. Настроить обновление статуса вакансии в базе данных.

#### Юзер-стори 2.5: Удаление вакансии

Как авторизованный пользователь, я хочу удалить вакансию, чтобы скрыть неактуальную информацию.

Таски:

1. Добавить кнопку для удаления вакансии.
2. Реализовать подтверждение перед удалением.
3. Удалить вакансию из базы данных для хард удаления или установить статус удалена в записи в базе данных для реализации софт удаления.

#### Юзер-стори 2.6: Добавление кандидатов в вакансию

Как авторизованный пользователь, я хочу добавлять кандидатов к вакансии для отслеживания их прогресса.

Таски:

1. Создать интерфейс для добавления кандидатов.
2. Настроить поля с информацией о кандидате.
3. Сохранить кандидатов в базе данных.

#### Юзер-стори 2.7: Перемещение кандидатов по этапам воронки

Как авторизованный пользователь, я хочу перемещать кандидатов по этапам воронки, чтобы отслеживать их статус.

Таски:

1. Создать интерфейс с этапами воронки.
2. Настроить возможность перетаскивания кандидатов между этапами.
3. Обновить статус кандидата в базе данных после перемещения.

#### Юзер-стори 2.8: Удаление кандидатов из вакансии или с воронки

Как авторизованный пользователь, я хочу удалять кандидатов с этапа воронки.

Таски:

1. Добавить возможность удаления кандидата из этапа воронки.
2. Добавить форму для подтверждения удаления с обязательным вводом причины в текстовое поле с лимитом на длину текста.
3. Обновить данные в базе, исключив удаленного кандидата.

#### Юзер-стори 2.9: Установка этапов для воронки по вакансии

Как авторизованный пользователь, я хочу устанавливать этапы при создании вакансии, чтобы управлять кандидатами используя этапы воронки.

Таски:

1. Добавить на страницу создания вакансии форму для установки этапов воронки
2. Сохранить изменения этапов в базе данных.

#### Юзер-стори 2.10: Просмотр статистики по вакансии

Как авторизованный пользователь, я хочу видеть статистику по выбранной вакансии, чтобы оценить эффективность процесса найма.

Таски:

1. Разработать интерфейс для отображения статистики вакансии:
   * Создать отдельную страницу для отображения статистики.
   * Добавить на страницу вакансии кнопку для перехода на страницу статистики по ней
2. Рассчитать и отобразить данные:
   * Общее количество кандидатов.
   * Количество кандидатов на каждом этапе воронки.
   * Среднее время на каждом этапе.
   * Конверсия между этапами.
   * Среднее время закрытия вакансий.

# **Git workflow definition**

Branches:

* *main* – a main branch, contains a stable version of the project.
* *dev* – a develop branch, is created from *main*, contains the latest version that is ready for testing.
* *featute/{feature-name}* – a feature branch, is created from *dev*, contains development of new feature; when *feat* is complete it is merged into *dev*.
* *release/{version}* – a release branch, is created from *dev*, contains new release version development; when *release* is done it is merged into dev and *main*.
* *hotfix/{bug-name}* – a bug fix branch, is created from main, when there are some bugs; when *hotfix* is done it is merged into the dev and main branch.

Commit messages:

* init: … – project’s start
* feat: … – add new feature
* fix: … – fix the bug
* docs: … add documentation

# **Technology stack specification**

Node.js, Express.js, PostgreSQL, Javascript, React, Docker