Individual Final Project: Marvel & DC: Ultimate Guide

1. **Project Objective.**
2. **Context** and the needs it addresses.
3. **General Description** of the application and its main features.
4. **Functional and technical requirements.**
5. **Technologies to be used.**
6. **Planning phase** (prototyping, data modeling, user stories, backlog).
7. **Development phase** (project setup, CRUD, testing).
8. **Programming best practices** (SOLID, DRY, KISS principles).
9. **Final documentation and presentation**, including a sample work plan.

1. General Project Objective

The goal of this application is to provide a complete, accessible, and well-organized platform where users can find detailed information about Marvel and DC characters. This digital encyclopedia is designed to solve the difficulty of finding consolidated and accurate information about these universes' characters in one place. The app also aims to offer an optimal user experience with advanced search features, an attractive design, and accessibility for users with disabilities.

💡 Brainstorming and Organization Phase

2. Context: What need does it address?

- **Centralized Information**: Marvel and DC fans often search through multiple sources to find detailed information about their favorite characters. This encyclopedia will provide a single reliable source where users can find all the relevant information.
- **Accessibility**: Many existing platforms do not meet accessibility standards, making it difficult for people with disabilities to access the information. Our application will ensure all information is accessible.
- **Organization and Navigation**: Character information can be disorganized and hard to navigate. The encyclopedia will present data in a clear and structured way, making it easy to search and explore characters.

---

3. Application Description

Marvel & DC: Ultimate Guide will be a comprehensive platform for exploring and managing detailed information about Marvel and DC characters. It will feature advanced search capabilities, full character management (CRUD), community interaction, and a focus on accessibility and modern design.

Examples of features:

- Exploration and Search
- Character Management (CRUD)
- Community Interaction
- Accessibility and Usability
- Attractive and Modern Design
- Reusable Components

💻 Requirements

4. Functional Requirements

- **API CRUD**: Full entity management (create, read, update, delete) via the API.
- **Frontend CRUD**: Full API management from the user interface.
- **100% Test Coverage**: Unit and integration tests ensuring 100% coverage.
- **Accessible App**: Compliance with accessibility standards.
- **Aesthetic Design**: Attractive and consistent visual design.

5. Technologies to be Used

- **Back-end**: Python, API with Django
- **Front-end**: React with Tailwind.

🖌️ Deliverables

6. Planning Phase

Prototype

- Figma to design the main views (PC and mobile).
- **Atomic Design Methodology**: with reusable components.

Data Modeling

- Main entities (e.g., Teams, Members) and their relationships.
- DrawSQL or DrawIO to visualize the data model.

User Stories (Format)

- **As a** [user type] **I want to** [what the user wants to do] **so that** [purpose].
- **Acceptance Criteria**: Define when a story is considered complete.

Example:

- As an admin, I want to create a new team to manage its members.

---

7. Development Phase

🛠️ Project Setup

1. **Backend (Django)**:
   - Create a Django project.
   - Configure the PostgreSQL database.
   - Define models based on the ER diagram.
2. **Frontend (React)**:
   - Create the app with Create React App.
   - Configure routing with React Router.
   - Create basic components.

CRUD in Backend

- Models, Serializers, Views, and URLs to implement CRUD functionality.

🧪 Testing

- **Backend**: Implement unit tests with pytest/unittest.
- **Frontend**: Use Jest and React Testing Library.

---

8. Programming Best Practices

SOLID Principles

- **Single Responsibility Principle (SRP)**: Each class/module should have a single responsibility.
- **Open/Closed Principle (OCP)**: Entities should be open for extension but closed for modification.

- **DRY (Don't Repeat Yourself)**: Avoid code duplication through reusable functions.

9. Documentation

- **README.md**:
  - Brief project description.
  - Technical dependencies used.
  - Installation and local execution instructions.

🎉 Final Presentation

Resources and Tools

- **Figma**: Interface design.
- **DrawSQL/DrawIO**: Data modeling.
- **Jira**: Project management.
- **Django REST Framework**: API creation.
- **React**: Front-end development.
- **Postman**: CRUD endpoint testing.

🔧 Sample Work Plan (Week and a Half):

Week 1:

- **Monday:**
  - Brainstorming, project definition, and prototype design.
  - Data modeling and ER diagram creation.
- **Tuesday:**
  - User stories and product backlog.
  - Project setup (React for frontend).
- **Wednesday:**
  - Implement CRUD in frontend (basic components and API calls).
- **Thursday:**
  - Project setup (Django for backend).
  - Implement CRUD in backend (models, serializers, views, URLs).
- **Friday:**
  - Initial frontend testing (Jest/React Testing Library).
  - Initial backend testing (pytest/unittest).

Week 2 (Half-Week):

- **Monday:**

    - Accessibility and responsive design (WCAG).
    - Final adjustments in frontend (usability and UX improvements).
- **Tuesday:**
    - Frontend-backend integration testing (using Axios/Fetch and Postman).
    - Complete test coverage.
- **Wednesday:**
    - Project documentation (README.md, installation, dependencies).
    - Final presentation preparation.
- **Thursday (optional):**
    - Presentation rehearsal and final reviews.