### **Backend Design for Portfolio Website**

#### **1. Overview**

The backend architecture for the portfolio website is built using Django, ensuring a robust and maintainable structure. The system handles bilingual content dynamically (English and Italian), supports dynamic filtering and relationships, and provides efficient data handling with SQLite as the database for development.

#### **2. Models**

The data layer includes the following models:

* **Project**:
  + Represents portfolio projects with bilingual titles, descriptions, dates, and technology stacks.
  + Relationships: Includes folders (Folder) and reviews (Review).
  + Flags: is\_public determines visibility.
* **Folder**:
  + Groups files (File) related to a project.
  + Bilingual fields for names and descriptions.
* **File**:
  + Represents individual files within folders.
  + Includes privacy controls with is\_private.
* **ContactInfo**:
  + Stores site-wide contact information. Ensures a single instance using custom validation.
* **Skill**:
  + Represents skills, linked to certificates, education, and job experiences.
* **Certificate**:
  + Bilingual descriptions for certifications, linked to skills (Skill).
* **Education**:
  + Represents academic achievements, linked to skills.
  + Tracks ongoing and completed education entries.
* **Review**:
  + Stores user-submitted reviews for projects, with statuses for admin moderation.
* **JobExperience**:
  + Represents professional experiences with bilingual titles and descriptions.
* **Section**:
  + Defines personal description sections with bilingual support.
* **Category** and **BlogPost**:
  + Categories group blog posts.
  + Blog posts feature bilingual content, excerpts, and categories.

#### **3. Views**

The logic layer includes views to manage and present content dynamically:

* **General Views**:
  + home: Renders the home page with dynamic bilingual sections.
  + education: Displays skills, certificates, education, and job experiences.
  + projects: Lists public projects and their associated folders and files.
  + contact: Handles contact form submissions and emails.
* **Detail Views**:
  + Handle detailed data presentation for projects, folders, education, job experiences, and certificates, with dynamic bilingual fields.
* **List Views**:
  + List job experiences, education entries, and certificates. Separate ongoing and completed entries dynamically.
* **Blog Views**:
  + blog\_list: Lists blog posts with optional search and category filtering.
  + blog\_detail: Displays single blog post details and related posts.
* **Utility Views**:
  + Redirects: redirect\_to\_linkedin and redirect\_to\_github handle external profile redirection.
  + open\_cv: Allows users to download CV files in the selected language.

#### **4. Admin Configuration**

Django Admin is customized to manage models effectively:

* **ProjectAdmin**:
  + Displays bilingual titles, dates, and public visibility.
  + Supports search and filtering.
* **Skill Relationships**:
  + Inline relationships for certificates, education, and job experiences.
* **Review Moderation**:
  + Admin actions to approve or reject reviews directly from the admin panel.
* **Prepopulated Fields**:
  + Auto-generates slugs for categories and blog posts.

#### **5. Testing Strategy**

The backend is rigorously tested using Django’s test framework. Test cases cover:

* **Model Testing**:
  + Validate field integrity and relationships for models like Project, BlogPost, and Review.
* **View Testing**:
  + Ensure views return correct templates and context data.
  + Test dynamic filtering (e.g., skills in education view).
* **Edge Cases**:
  + Handle missing or invalid data (e.g., nonexistent blog posts).

#### **6. CI/CD Pipeline**

The deployment is handled using **Render** with Continuous Deployment (CD). For CI, GitHub Actions is recommended:

* **CI Setup**:
  + Run tests automatically on code commits.
  + Linting for code quality.
* **CD Setup**:
  + Auto-deploy changes from the repository to the Render environment.
  + Environment configuration for staging and production.

#### **7. Future Plans**

* **Analytics Dashboard**:
  + Develop a React-based analytics page that fetches data from APIs.
* **AI Tools**:
  + Integrate AI features for enhanced user experience (e.g., project recommendations, content summarization).