### **Agile Project Plan: Portfolio Website Development**

#### **1. Scope and Goals**

**Scope**:

* Develop a functional and responsive portfolio website within three weeks.
* The website will showcase projects, skills, and professional information.
* Ensure cross-browser compatibility, mobile responsiveness, and aesthetic design.
* Deploy the website on a Linux server.

**Goals**:

* Deliver a fully functional portfolio website.
* Maintain high quality through iterative testing and reviews.
* Meet the three-week deadline using Agile methodology.

#### **2. Approach and Methodology**

**Agile Approach**:

* Follow a single sprint plan spanning three weeks.
* Daily stand-ups to assess progress and align on objectives.
* Weekly retrospectives to evaluate sprint progress and identify improvement areas.
* Utilize a Kanban board on Taiga to track and prioritize tasks.

**Key Agile Practices**:

* Iterative development: Deliver small, incremental improvements to the portfolio website.
* Continuous feedback: Review and adjust the work based on input from daily stand-ups and retrospectives.
* Focus on the definition of "Done": A functional feature ready for deployment.

#### **3. Sprint Schedule**

**Sprint Duration**: 3 weeks (15 working days)

**Schedule**:

* **Morning Stand-up**: Each day, outline progress, impediments, and daily objectives.
* **End-of-Week Retrospective**: Evaluate the week's work, identify blockers, and adjust priorities.

#### **4. Tools and Technology Stack**

**Technology Stack**:

* **Backend**: Django
* **Frontend**: HTML, CSS, JavaScript
* **Server Environment**: Linux

**Tools**:

* **Taiga Kanban Board**: Task tracking and progress visualization.
* **ChatGPT**: For stand-up support and retrospective facilitation.
* **Code Editor**: VSCode or equivalent.
* **Version Control**: Git with GitHub or GitLab.

### 

### **Work Breakdown Structure (WBS)**

This WBS includes the main tasks across the sprints and potential future work.

#### **1. Project Setup and Planning**

* Set up project repository on GitHub.
* Configure Django and database environment.
* Install dependencies and create virtual environment.

#### **2. Backend Development**

* **Models and Database Structure**
  + Set up models for Projects, Blog Posts, and Resume.
  + Implement relationships (e.g., tags for categorizing projects).
* **Views and CRUD Operations**
  + Create views and logic for Projects, Blog Posts, and Resume sections.
  + Implement filing system for project categorization.
* **Contact Form**
  + Build and validate a contact form.
  + Configure email settings or database storage for contact requests.

#### **3. Frontend Development**

* **Base Templates and Layout**
  + Set up consistent base layout (header, footer, main content).
* **Portfolio Page and Project Display**
  + Design and develop project list and detail pages.
  + Add sorting and filtering options for projects.
* **Blog Page**
  + Create blog list and detail templates.
* **Contact Page**
  + Style and validate the contact form.
* **Static Files Management**
  + Set up CSS, JavaScript, and image management.
* **Responsive Design**
  + Ensure the site is responsive and mobile-friendly.

#### **4. DevOps and CI/CD**

* Set up a CI/CD pipeline in GitHub for code testing and deployment.
* Explore potential hosting options.
* Set up automated testing for core features (optional).

#### **5. Testing and Documentation**

* **Unit and Integration Testing**
  + Write tests for key functionalities.
* **Documentation**
  + Document key sections of code and structure on GitHub.
* **Bug Fixes and Optimizations**
  + Identify and fix any initial bugs or performance issues.

#### **6. Future Development**

* **React Integration for Interactivity**
  + Plan for future integration of React for enhanced interactivity.
* **Advanced Project Management**
  + Add advanced features like project comparisons, client testimonials.
* **Branding and Design Updates**
  + Incorporate branding, logo, and custom styles as they are developed.

### **SPRINT 1\_CREATE WEBSITE**

### **Week 1: Initial Setup and Basic Structure**

#### **Monday**

* **Project Initialization**
  + Set up a new GitHub repository for the project.
  + Initialize a Django project using django-admin startproject.
  + Create a virtual environment and install required packages:
    - Django
    - psycopg2 (for PostgreSQL)
  + Verify the project runs locally.

#### **Tuesday**

* **Database Setup**
  + Install and configure PostgreSQL.
  + Create a new PostgreSQL database for your project.
  + Update Django settings.py to connect to the PostgreSQL database.
  + Test the database connection and run initial migrations.

#### **Wednesday**

* **Basic Django Apps Creation**
  + Create Django apps:
    - projects
    - blog
    - resume
    - contact
  + Update settings.py to include the new apps.
  + Set up basic URL routing in the main urls.py file.

#### **Thursday**

* **Template and Static Files Configuration**
  + Set up directories for templates and static files.
    - Create a templates directory and a base HTML template (base.html).
    - Configure STATIC\_URL and STATICFILES\_DIRS in settings.py.
  + Add placeholders for common layout elements (header, footer, navigation).

#### **Friday**

* **Models Definition**
  + **Projects App:**
    - Define the Project model with fields like title, description, technologies, image, github\_link, live\_demo\_link, etc.
  + **Blog App:**
    - Define the BlogPost model with fields like title, content, published\_date, tags.
  + **Resume App:**
    - Define models as needed for resume sections (education, experience, skills).
  + Run migrations to create the tables in the database.

### **Week 2: Core Features Development**

#### **Monday**

* **Admin Site Configuration**
  + Register the models in admin.py for each app.
  + Create a superuser account.
  + Use the Django admin site to add sample data for testing.
* **Project Views and URLs**
  + **Views:**
    - Create views for the projects app: project\_list, project\_detail.
  + **URLs:**
    - Set up URLs for the project list and detail views.
  + **Templates:**
    - Create project\_list.html and project\_detail.html templates extending base.html.

#### **Tuesday**

* **Project Filing System**
  + Implement a tagging or categorization system for projects.
    - Update the Project model to include tags/categories.
  + Modify the project\_list view to allow filtering by tag/category.
  + Update templates to display and filter tags/categories.

#### **Wednesday**

* **Contact Form Implementation**
  + **Forms:**
    - Create a ContactForm using Django forms with fields: name, email, subject, message.
  + **Views and URLs:**
    - Create a view to handle form display and submission.
    - Set up the URL for the contact form page.
  + **Templates:**
    - Create contact.html template with form rendering and validation messages.

#### **Thursday**

* **Email Backend Configuration**
  + Configure email settings in settings.py (using SMTP or console backend for testing).
  + Implement form submission logic to send an email upon successful validation.
  + Test the contact form thoroughly.

#### **Friday**

* **Blog Views and Templates**
  + **Views:**
    - Create views for BlogPost: blog\_list, blog\_detail.
  + **URLs:**
    - Set up URLs for blog list and detail views.
  + **Templates:**
    - Create blog\_list.html and blog\_detail.html templates.
  + Add sample blog posts via the admin site for testing.

### **Week 3: Frontend Refinement and Testing**

#### **Monday**

* **Frontend Styling Begins**
  + Start applying CSS to the base template and shared components (header, footer).
  + Choose a CSS framework if desired and integrate it.
  + Ensure static files are correctly linked and loading.
* **Styling the Portfolio Pages**
  + Style project\_list.html and project\_detail.html for visual appeal.
  + Ensure that the project filtering system is user-friendly.
  + Add images and media to enhance the portfolio display.

#### **Tuesday**

* **Styling the Blog and Contact Pages**
  + Style blog\_list.html and blog\_detail.html.
  + Add pagination to the blog list if necessary.
  + Enhance the contact.html page with better form styling and user feedback messages.

#### **Wednesday**

* **Responsive Design Implementation**
  + Use CSS media queries or responsive frameworks to ensure the site looks good on various devices.
  + Test the website on different screen sizes and browsers.
  + Optimize images and assets for performance.

#### **Thursday**

* **Testing and Bug Fixing**
  + Conduct thorough testing of all features:
    - Blog functionalities.
    - Contact form submissions.
  + Check for broken links, errors, and exceptions.
  + Review console logs for frontend errors.

#### **Friday**

* **CI/CD Pipeline Setup**
  + Set up GitHub Actions for continuous integration:
    - Configure workflows to run tests on each push or pull request.
    - Ensure the codebase passes all tests before merging.
  + Explore options for continuous deployment (optional at this stage).
* **Documentation**
  + Write README.md with project overview and setup instructions.
  + Document how to run the project locally.
  + Add comments and docstrings in your code where necessary.