Coaching Business Website - Wagtail Project Specification

Project Overview

A professional coaching business website built with Wagtail CMS, styled with inspiration from alyssanobriga.com. The site will feature a warm, elegant, and feminine design aesthetic with HTMX for dynamic interactions and Alpine.js for lightweight client-side functionality.

Tech Stack

Layer	Technology
CMS	Wagtail 6.x
Backend	Django 5.x / Python 3.12+
Database	PostgreSQL
Frontend	HTMX, Alpine.js, TailwindCSS
Search	Wagtail Search (PostgreSQL backend)
Media Storage	S3-compatible (or local for dev)
Deployment	Docker + your preferred host

Design System

Color Palette (Inspired by Reference)

```
:root {
  /* Primary - Warm, earthy tones */
  --color-primary: #C4A484;        /* Warm tan/camel */
  --color-primary-dark: #8B7355;   /* Darker earth tone */
  --color-primary-light: #E8DCD0;  /* Light cream */

  /* Secondary - Soft feminine accents */
  --color-secondary: #D4A5A5;      /* Dusty rose */
  --color-secondary-light: #F5E6E6; /* Blush pink */

  /* Neutrals */
  --color-white: #FFFFFF;
  --color-off-white: #FAF8F5;      /* Warm white background */
  --color-text: #3D3D3D;           /* Soft black for text */
  --color-text-light: #6B6B6B;     /* Secondary text */
  --color-border: #E5E0DB;         /* Subtle borders */

  /* Accents */
  --color-gold: #B8860B;           /* Gold accents */
  --color-success: #7CB69D;        /* Sage green */
}
```

Typography

```
/* Headings - Elegant serif */
--font-heading: 'Cormorant Garamond', 'Playfair Display', Georgia, serif;

/* Body - Clean sans-serif */
--font-body: 'Lato', 'Open Sans', system-ui, sans-serif;

/* Accent/Script - For special callouts */
--font-accent: 'Dancing Script', 'Great Vibes', cursive;
```

Design Principles

- **Generous whitespace -** Let content breathe
- **Large, high-quality imagery -** Professional photography throughout
- **Soft shadows and rounded corners -** Approachable feel
- **Subtle animations** - Gentle fade-ins and hover states
- **Mobile-first responsive design**
- **Warm, inviting color temperature**

Site Architecture

Page Types (Wagtail Models)

```
HomePage
|---- AboutPage
|---- ServicesIndexPage
|     `---- ServicePage (1:1 Coaching, Group Programs, etc.)
|---- CoursesIndexPage
|     `---- CoursePage
|---- BlogIndexPage
|     `---- BlogPage
|---- TestimonialsPage
|---- ResourcesPage (Lead Magnets/Freebies)
|---- ContactPage
|---- BookingPage
|---- FAQPage
`---- GenericPage (Privacy, Terms, etc.)
```

Wagtail Models Specification

Core App: `home`

```python
# home/models.py

from django.db import models
from wagtail.models import Page
from wagtail.fields import RichTextField, StreamField
from wagtail.admin.panels import FieldPanel, MultiFieldPanel, InlinePanel
from wagtail.images.blocks import ImageChooserBlock
from wagtail import blocks


class HomePage(Page):
    """Landing page with modular sections"""

    # Hero Section
    hero_title = models.CharField(max_length=255)
    hero_subtitle = models.CharField(max_length=500, blank=True)
    hero_cta_text = models.CharField(max_length=100, default="Work With Me")
    hero_cta_link = models.ForeignKey(
        'wagtailcore.Page',
        null=True, blank=True,
        on_delete=models.SET_NULL,
        related_name='+'
    )
    hero_image = models.ForeignKey(
        'wagtailimages.Image',
        null=True, blank=True,
        on_delete=models.SET_NULL,
        related_name='+'
    )

    # About Preview Section
    about_heading = models.CharField(max_length=255, blank=True)
    about_text = RichTextField(blank=True)
    about_image = models.ForeignKey(
        'wagtailimages.Image',
        null=True, blank=True,
        on_delete=models.SET_NULL,
        related_name='+'
    )

    # Services Preview
    services_heading = models.CharField(max_length=255, default="How I Can Help")
    services_subheading = models.TextField(blank=True)

    # Testimonials Section
    testimonials_heading = models.CharField(max_length=255, default="What Clients Say")
    featured_testimonials = StreamField([
        ('testimonial', blocks.StructBlock([
            ('quote', blocks.TextBlock()),
            ('author_name', blocks.CharBlock()),
            ('author_title', blocks.CharBlock(required=False)),
            ('author_image', ImageChooserBlock(required=False)),
        ]))
    ], blank=True, use_json_field=True)

    # Newsletter Section
    newsletter_heading = models.CharField(max_length=255, default="Join My Community")
    newsletter_text = models.TextField(blank=True)
    newsletter_freebie_title = models.CharField(max_length=255, blank=True)

    # Featured Blog Posts (auto-populated or manual selection)
    show_blog_section = models.BooleanField(default=True)
    blog_section_heading = models.CharField(max_length=255, default="From the Blog")

    content_panels = Page.content_panels + [
        MultiFieldPanel([
            FieldPanel('hero_title'),
            FieldPanel('hero_subtitle'),
            FieldPanel('hero_cta_text'),
            FieldPanel('hero_cta_link'),
            FieldPanel('hero_image'),
        ], heading="Hero Section"),
        MultiFieldPanel([
            FieldPanel('about_heading'),
            FieldPanel('about_text'),
            FieldPanel('about_image'),
        ], heading="About Preview"),
        MultiFieldPanel([
            FieldPanel('services_heading'),
            FieldPanel('services_subheading'),
        ], heading="Services Section"),
        MultiFieldPanel([
            FieldPanel('testimonials_heading'),
            FieldPanel('featured_testimonials'),
        ], heading="Testimonials"),
        MultiFieldPanel([
            FieldPanel('newsletter_heading'),
            FieldPanel('newsletter_text'),
            FieldPanel('newsletter_freebie_title'),
        ], heading="Newsletter/Freebie"),
        MultiFieldPanel([
            FieldPanel('show_blog_section'),
            FieldPanel('blog_section_heading'),
        ], heading="Blog Section"),
    ]

    class Meta:
        verbose_name = "Home Page"
```

Services App: `services`

```
# services/models.py

from django.db import models
from wagtail.models import Page
from wagtail.fields import RichTextField, StreamField
from wagtail.admin.panels import FieldPanel, MultiFieldPanel
from wagtail import blocks
from wagtail.images.blocks import ImageChooserBlock


class ServicesIndexPage(Page):
    """Services landing page"""

    intro_heading = models.CharField(max_length=255, default="Work With Me")
    intro_text = RichTextField(blank=True)
    hero_image = models.ForeignKey(
        'wagtailimages.Image',
        null=True, blank=True,
        on_delete=models.SET_NULL,
        related_name='+'
    )

    subpage_types = ['services.ServicePage']

    content_panels = Page.content_panels + [
        FieldPanel('intro_heading'),
        FieldPanel('intro_text'),
        FieldPanel('hero_image'),
    ]

    def get_services(self):
        return ServicePage.objects.live().child_of(self).order_by('service_order')

class ServicePage(Page):
```

```
"""Individual service offering"""

SERVICE_TYPES = [
    ('1on1', '1:1 Coaching'),
    ('group', 'Group Program'),
    ('course', 'Online Course'),
    ('vip', 'VIP Day'),
    ('retreat', 'Retreat'),
    ('corporate', 'Corporate/Speaking'),
]

service_type = models.CharField(max_length=20, choices=SERVICE_TYPES, default='1on1')
tagline = models.CharField(max_length=255, blank=True)
short_description = models.TextField(help_text="Brief description for cards/previews")

# Hero
hero_image = models.ForeignKey(
    'wagtailimages.Image',
    null=True, blank=True,
    on_delete=models.SET_NULL,
    related_name='+'
)

# Main Content
body = StreamField([
    ('heading', blocks.CharBlock(form_classname="title")),
    ('paragraph', blocks.RichTextBlock()),
    ('image', ImageChooserBlock()),
    ('quote', blocks.BlockQuoteBlock()),
    ('benefits_list', blocks.ListBlock(
        blocks.StructBlock([
            ('icon', blocks.CharBlock(required=False, help_text="Icon name or emoji")),
            ('title', blocks.CharBlock()),
            ('description', blocks.TextBlock(required=False)),
        ])
    )),
    ('pricing_card', blocks.StructBlock([
        ('title', blocks.CharBlock()),
        ('price', blocks.CharBlock()),
        ('frequency', blocks.CharBlock(required=False, help_text="e.g., 'per month', 'one-time'")),
        ('features', blocks.ListBlock(blocks.CharBlock())),
        ('cta_text', blocks.CharBlock(default="Apply Now")),
        ('cta_link', blocks.URLBlock(required=False)),
    ])),
    ('faq_section', blocks.ListBlock(
        blocks.StructBlock([
            ('question', blocks.CharBlock()),
            ('answer', blocks.RichTextBlock()),
        ])
    )),
    ('cta_banner', blocks.StructBlock([
        ('heading', blocks.CharBlock()),
        ('text', blocks.TextBlock(required=False)),
        ('button_text', blocks.CharBlock()),
        ('button_link', blocks.URLBlock()),
    ])),
], use_json_field=True)

# CTA
cta_text = models.CharField(max_length=100, default="Apply Now")
cta_link = models.URLField(blank=True, help_text="Link to application form or booking page")

# Ordering
service_order = models.PositiveIntegerField(default=0)

# Is this a featured/flagship offering?
is_featured = models.BooleanField(default=False)

parent_page_types = ['services.ServicesIndexPage']

content_panels = Page.content_panels + [
    MultiFieldPanel([
        FieldPanel('service_type'),
        FieldPanel('tagline'),
        FieldPanel('short_description'),
        FieldPanel('is_featured'),
        FieldPanel('service_order'),
    ], heading="Service Info"),
    FieldPanel('hero_image'),
    FieldPanel('body'),
    MultiFieldPanel([
        FieldPanel('cta_text'),
        FieldPanel('cta_link'),
    ], heading="Call to Action"),
]

class Meta:
    ordering = ['service_order']
```

Blog App: `blog`

```
# blog/models.py

from django.db import models
from wagtail.models import Page
from wagtail.fields import StreamField
from wagtail.admin.panels import FieldPanel, MultiFieldPanel
from wagtail.search import index
from wagtail import blocks
from wagtail.images.blocks import ImageChooserBlock
from modelcluster.fields import ParentalManyToManyField
from taggit.managers import TaggableManager
from modelcluster.contrib.taggit import ClusterTaggableManager


class BlogIndexPage(Page):
    """Blog listing page"""
    intro_heading = models.CharField(max_length=255, default="Blog")
    intro_text = models.TextField(blank=True)
    posts_per_page = models.PositiveIntegerField(default=9)

    subpage_types = ['blog.BlogPage']

    content_panels = Page.content_panels + [
        FieldPanel('intro_heading'),
        FieldPanel('intro_text'),
        FieldPanel('posts_per_page'),
    ]

    def get_context(self, request):
        context = super().get_context(request)
        posts = BlogPage.objects.live().child_of(self).order_by('-first_published_at')

        # Filter by category if provided
        category = request.GET.get('category')
        if category:
            posts = posts.filter(categories__slug=category)

        # Filter by tag if provided
        tag = request.GET.get('tag')
        if tag:
            posts = posts.filter(tags__name=tag)

        context['posts'] = posts
        context['categories'] = BlogCategory.objects.all()
        return context


class BlogCategory(models.Model):
    """Blog categories"""
    name = models.CharField(max_length=100)
    slug = models.SlugField(unique=True)
    description = models.TextField(blank=True)

    class Meta:
        verbose_name_plural = "Blog Categories"

    def __str__(self):
        return self.name


class BlogPage(Page):
    """Individual blog post"""

    # Meta
    author = models.CharField(max_length=255, blank=True)
    date = models.DateField("Post date")

    # Featured Image
    featured_image = models.ForeignKey(
        'wagtailimages.Image',
        null=True, blank=True,
        on_delete=models.SET_NULL,
        related_name='+'
    )

    # Excerpt for previews
    excerpt = models.TextField(
        max_length=500,
        help_text="Brief summary for blog cards and SEO"
    )

    # Main Content
    body = StreamField([
        ('heading', blocks.CharBlock(form_classname="title")),
        ('paragraph', blocks.RichTextBlock()),
        ('image', ImageChooserBlock()),
        ('quote', blocks.BlockQuoteBlock()),
        ('embed', blocks.RawHTMLBlock(help_text="For videos, podcasts, etc.")),
        ('code', blocks.StructBlock([
            ('language', blocks.CharBlock(default='python')),
            ('code', blocks.TextBlock()),
        ])),
        ('callout', blocks.StructBlock([
            ('type', blocks.ChoiceBlock(choices=[
                ('info', 'Info'),
                ('tip', 'Tip'),
                ('warning', 'Warning'),
            ])),
            ('content', blocks.RichTextBlock()),
        ])),
    ], use_json_field=True)

    # Categorization
    categories = ParentalManyToManyField('blog.BlogCategory', blank=True)
    tags = ClusterTaggableManager(through='blog.BlogPageTag', blank=True)

    # Reading time (auto-calculated or manual)
    reading_time_minutes = models.PositiveIntegerField(default=5)

    # Related content
    show_related_posts = models.BooleanField(default=True)

    parent_page_types = ['blog.BlogIndexPage']

    search_fields = Page.search_fields + [
        index.SearchField('body'),
        index.SearchField('excerpt'),
        index.FilterField('date'),
    ]

    content_panels = Page.content_panels + [
        MultiFieldPanel([
            FieldPanel('author'),
            FieldPanel('date'),
            FieldPanel('reading_time_minutes'),
        ], heading="Post Meta"),
        FieldPanel('featured_image'),
        FieldPanel('excerpt'),
        FieldPanel('body'),
        MultiFieldPanel([
            FieldPanel('categories', widget=forms.CheckboxSelectMultiple),
            FieldPanel('tags'),
        ], heading="Categorization"),
        FieldPanel('show_related_posts'),
    ]


class BlogPageTag(TaggedItemBase):
    content_object = ParentalKey(
        'BlogPage',
        related_name='tagged_items',
        on_delete=models.CASCADE
    )
```

About Page: `pages`

```
# pages/models.py

from django.db import models
from wagtail.models import Page
from wagtail.fields import RichTextField, StreamField
from wagtail.admin.panels import FieldPanel, MultiFieldPanel
from wagtail import blocks
from wagtail.images.blocks import ImageChooserBlock


class AboutPage(Page):
    """About/Bio page"""

    # Hero Section
    hero_heading = models.CharField(max_length=255, default="Hi, I'm [Name]")
    hero_subheading = models.TextField(blank=True)
    hero_image = models.ForeignKey(
        'wagtailimages.Image',
        null=True, blank=True,
        on_delete=models.SET_NULL,
        related_name='+'
    )

    # Story Section
    story = StreamField([
        ('heading', blocks.CharBlock(form_classname="title")),
        ('paragraph', blocks.RichTextBlock()),
        ('image', ImageChooserBlock()),
        ('image_text_block', blocks.StructBlock([
            ('image', ImageChooserBlock()),
            ('image_position', blocks.ChoiceBlock(choices=[
                ('left', 'Image Left'),
                ('right', 'Image Right'),
            ])),
            ('text', blocks.RichTextBlock()),
        ])),
        ('quote', blocks.BlockQuoteBlock()),
    ], use_json_field=True)

    # Credentials/Certifications
    credentials_heading = models.CharField(max_length=255, default="Credentials & Training")
    credentials = StreamField([
        ('credential', blocks.StructBlock([
            ('title', blocks.CharBlock()),
            ('organization', blocks.CharBlock(required=False)),
            ('year', blocks.CharBlock(required=False)),
            ('logo', ImageChooserBlock(required=False)),
        ]))
    ], blank=True, use_json_field=True)

    # Values/Philosophy
    philosophy_heading = models.CharField(max_length=255, default="My Approach")
    philosophy_text = RichTextField(blank=True)

    # Fun Facts / Personal Section
    personal_heading = models.CharField(max_length=255, default="Beyond the Bio")
    personal_facts = StreamField([
        ('fact', blocks.StructBlock([
            ('emoji', blocks.CharBlock(max_length=10, required=False)),
            ('text', blocks.CharBlock()),
        ]))
    ], blank=True, use_json_field=True)

    # CTA
    cta_heading = models.CharField(max_length=255, default="Ready to Start Your Journey?")
    cta_text = models.TextField(blank=True)
    cta_button_text = models.CharField(max_length=100, default="Let's Connect")
    cta_button_link = models.ForeignKey(
        'wagtailcore.Page',
        null=True, blank=True,
        on_delete=models.SET_NULL,
        related_name='+'
    )

    content_panels = Page.content_panels + [
        MultiFieldPanel([
            FieldPanel('hero_heading'),
            FieldPanel('hero_subheading'),
            FieldPanel('hero_image'),
        ], heading="Hero Section"),
        FieldPanel('story'),
        MultiFieldPanel([
            FieldPanel('credentials_heading'),
            FieldPanel('credentials'),
        ], heading="Credentials"),
        MultiFieldPanel([
            FieldPanel('philosophy_heading'),
            FieldPanel('philosophy_text'),
        ], heading="Philosophy"),
        MultiFieldPanel([
            FieldPanel('personal_heading'),
            FieldPanel('personal_facts'),
        ], heading="Personal Touch"),
        MultiFieldPanel([
            FieldPanel('cta_heading'),
            FieldPanel('cta_text'),
            FieldPanel('cta_button_text'),
            FieldPanel('cta_button_link'),
        ], heading="Call to Action"),
    ]


class ContactPage(Page):
    """Contact page with form"""

    intro_heading = models.CharField(max_length=255, default="Let's Connect")
    intro_text = RichTextField(blank=True)

    # Contact Info
    email = models.EmailField(blank=True)
    phone = models.CharField(max_length=50, blank=True)
    location = models.CharField(max_length=255, blank=True, help_text="City, State or 'Virtual'")

    # Social Links (can also be site-wide in settings)
    instagram_url = models.URLField(blank=True)
    facebook_url = models.URLField(blank=True)
    linkedin_url = models.URLField(blank=True)
    youtube_url = models.URLField(blank=True)

    # Form settings
    form_heading = models.CharField(max_length=255, default="Send a Message")
    thank_you_text = RichTextField(default="Thank you for reaching out! I'll be in touch soon.")

    # Booking Integration
    show_booking_widget = models.BooleanField(default=False)
    booking_heading = models.CharField(max_length=255, default="Schedule a Call")
    calendly_url = models.URLField(blank=True, help_text="Your Calendly link")

    content_panels = Page.content_panels + [
        MultiFieldPanel([
            FieldPanel('intro_heading'),
            FieldPanel('intro_text'),
        ], heading="Intro"),
        MultiFieldPanel([
            FieldPanel('email'),
            FieldPanel('phone'),
            FieldPanel('location'),
        ], heading="Contact Info"),
        MultiFieldPanel([
            FieldPanel('instagram_url'),
            FieldPanel('facebook_url'),
            FieldPanel('linkedin_url'),
            FieldPanel('youtube_url'),
        ], heading="Social Media"),
        MultiFieldPanel([
            FieldPanel('form_heading'),
            FieldPanel('thank_you_text'),
        ], heading="Contact Form"),
        MultiFieldPanel([
            FieldPanel('show_booking_widget'),
            FieldPanel('booking_heading'),
            FieldPanel('calendly_url'),
        ], heading="Booking Integration"),
    ]


class TestimonialsPage(Page):
    """Testimonials showcase page"""

    intro_heading = models.CharField(max_length=255, default="Client Love")
    intro_text = RichTextField(blank=True)

    testimonials = StreamField([
        ('testimonial', blocks.StructBlock([
            ('quote', blocks.TextBlock()),
            ('author_name', blocks.CharBlock()),
            ('author_title', blocks.CharBlock(required=False)),
            ('author_location', blocks.CharBlock(required=False)),
            ('author_image', ImageChooserBlock(required=False)),
            ('service_type', blocks.CharBlock(required=False, help_text="e.g., '1:1 Coaching Client'")),
            ('featured', blocks.BooleanBlock(required=False)),
        ])),
        ('video_testimonial', blocks.StructBlock([
            ('video_embed', blocks.RawHTMLBlock()),
            ('author_name', blocks.CharBlock()),
            ('author_title', blocks.CharBlock(required=False)),
        ])),
    ], use_json_field=True)

    # CTA
    cta_heading = models.CharField(max_length=255, default="Ready to Write Your Success Story?")
    cta_button_text = models.CharField(max_length=100, default="Start Your Journey")
    cta_button_link = models.ForeignKey(
        'wagtailcore.Page',
        null=True, blank=True,
        on_delete=models.SET_NULL,
        related_name='+'
    )

    content_panels = Page.content_panels + [
        FieldPanel('intro_heading'),
        FieldPanel('intro_text'),
        FieldPanel('testimonials'),
        MultiFieldPanel([
            FieldPanel('cta_heading'),
            FieldPanel('cta_button_text'),
            FieldPanel('cta_button_link'),
        ], heading="Call to Action"),
    ]


class ResourcesPage(Page):
    """Free resources / Lead magnets page"""

    intro_heading = models.CharField(max_length=255, default="Free Resources")
    intro_text = RichTextField(blank=True)

    resources = StreamField([
        ('resource', blocks.StructBlock([
            ('title', blocks.CharBlock()),
            ('description', blocks.TextBlock()),
            ('image', ImageChooserBlock(required=False)),
            ('resource_type', blocks.ChoiceBlock(choices=[
                ('pdf', 'PDF Download'),
                ('video', 'Video Series'),
                ('audio', 'Audio/Meditation'),
                ('course', 'Mini Course'),
                ('quiz', 'Quiz/Assessment'),
                ('webinar', 'Webinar Replay'),
            ])),
            ('delivery_method', blocks.ChoiceBlock(choices=[
                ('instant', 'Instant Download'),
                ('email', 'Email Delivery'),
                ('redirect', 'Redirect to Page'),
            ])),
            ('cta_text', blocks.CharBlock(default="Get Free Access")),
            ('form_or_link', blocks.URLBlock(help_text="Link to form, file, or landing page")),
        ]))
    ], use_json_field=True)

    content_panels = Page.content_panels + [
        FieldPanel('intro_heading'),
        FieldPanel('intro_text'),
        FieldPanel('resources'),
    ]


class FAQPage(Page):
    """Frequently Asked Questions"""

    intro_heading = models.CharField(max_length=255, default="Frequently Asked Questions")
    intro_text = RichTextField(blank=True)

    faq_sections = StreamField([
        ('faq_section', blocks.StructBlock([
            ('section_title', blocks.CharBlock(required=False)),
            ('questions', blocks.ListBlock(
                blocks.StructBlock([
                    ('question', blocks.CharBlock()),
                    ('answer', blocks.RichTextBlock()),
                ])
            )),
        ]))
    ], use_json_field=True)

    # CTA
    still_questions_heading = models.CharField(max_length=255, default="Still Have Questions?")
    still_questions_text = models.TextField(blank=True)
    contact_button_text = models.CharField(max_length=100, default="Get in Touch")

    content_panels = Page.content_panels + [
        FieldPanel('intro_heading'),
        FieldPanel('intro_text'),
        FieldPanel('faq_sections'),
        MultiFieldPanel([
            FieldPanel('still_questions_heading'),
            FieldPanel('still_questions_text'),
            FieldPanel('contact_button_text'),
        ], heading="Bottom CTA"),
    ]


class GenericPage(Page):
    """Flexible page for Privacy Policy, Terms, etc."""

    body = StreamField([
        ('heading', blocks.CharBlock(form_classname="title")),
        ('paragraph', blocks.RichTextBlock()),
        ('raw_html', blocks.RawHTMLBlock()),
    ], use_json_field=True)

    last_updated = models.DateField(blank=True, null=True)

    content_panels = Page.content_panels + [
        FieldPanel('last_updated'),
        FieldPanel('body'),
    ]
```

Site Settings (Global Configuration)

```
# settings/models.py

from django.db import models
from wagtail.contrib.settings.models import BaseSiteSetting, register_setting
from wagtail.admin.panels import FieldPanel, MultiFieldPanel


@register_setting
class SiteSettings(BaseSiteSetting):
    """Global site settings"""

    # Branding
    site_name = models.CharField(max_length=255, default="Coaching Business")
    tagline = models.CharField(max_length=255, blank=True)
    logo = models.ForeignKey(
        'wagtailimages.Image',
        null=True, blank=True,
        on_delete=models.SET_NULL,
        related_name='+'
    )
    favicon = models.ForeignKey(
        'wagtailimages.Image',
        null=True, blank=True,
        on_delete=models.SET_NULL,
        related_name='+'
    )

    # Contact Info
    contact_email = models.EmailField(blank=True)
    contact_phone = models.CharField(max_length=50, blank=True)

    # Social Media
    instagram_url = models.URLField(blank=True)
    facebook_url = models.URLField(blank=True)
    linkedin_url = models.URLField(blank=True)
    youtube_url = models.URLField(blank=True)
    tiktok_url = models.URLField(blank=True)
    pinterest_url = models.URLField(blank=True)
    podcast_url = models.URLField(blank=True)

    # Newsletter Integration
    email_provider = models.CharField(
        max_length=50,
        choices=[
            ('mailchimp', 'Mailchimp'),
            ('convertkit', 'ConvertKit'),
            ('flodesk', 'Flodesk'),
            ('mailerlite', 'MailerLite'),
            ('custom', 'Custom Form'),
        ],
        default='mailchimp',
        blank=True
    )
    newsletter_form_action = models.URLField(blank=True, help_text="Form action URL from your provider")

    # Booking Integration
    calendly_url = models.URLField(blank=True)

    # Analytics
    google_analytics_id = models.CharField(max_length=50, blank=True)
    facebook_pixel_id = models.CharField(max_length=50, blank=True)

    # Footer
    footer_text = models.TextField(blank=True)
    copyright_name = models.CharField(max_length=255, blank=True)

    panels = [
        MultiFieldPanel([
            FieldPanel('site_name'),
            FieldPanel('tagline'),
            FieldPanel('logo'),
            FieldPanel('favicon'),
        ], heading="Branding"),
        MultiFieldPanel([
            FieldPanel('contact_email'),
            FieldPanel('contact_phone'),
        ], heading="Contact"),
        MultiFieldPanel([
            FieldPanel('instagram_url'),
            FieldPanel('facebook_url'),
            FieldPanel('linkedin_url'),
            FieldPanel('youtube_url'),
            FieldPanel('tiktok_url'),
            FieldPanel('pinterest_url'),
            FieldPanel('podcast_url'),
        ], heading="Social Media"),
        MultiFieldPanel([
            FieldPanel('email_provider'),
            FieldPanel('newsletter_form_action'),
        ], heading="Newsletter"),
        FieldPanel('calendly_url'),
        MultiFieldPanel([
            FieldPanel('google_analytics_id'),
            FieldPanel('facebook_pixel_id'),
        ], heading="Analytics"),
        MultiFieldPanel([
            FieldPanel('footer_text'),
            FieldPanel('copyright_name'),
        ], heading="Footer"),
    ]
```

Key Features Implementation

1. Newsletter Signup with HTMX

```
<!-- templates/components/newsletter_form.html -->
<div id="newsletter-form" class="newsletter-section">
    <form
        hx-post="{% url 'newsletter_signup' %}"
        hx-target="#newsletter-form"
        hx-swap="outerHTML"
        class="newsletter-form"
    >
        {% csrf_token %}
        <input
            type="email"
            name="email"
            placeholder="Enter your email"
            required
            class="newsletter-input"
        >
        <button type="submit" class="btn btn-primary">
            <span class="btn-text">Get Free Access</span>
            <span class="htmx-indicator">
                <!-- Loading spinner -->
            </span>
        </button>
    </form>
</div>
```

2. Blog Filtering with HTMX

```
<!-- templates/blog/blog_index_page.html -->
<div class="blog-filters" hx-boost="true">
    <a href="{% pageurl page %}"
       class="filter-btn {% if not request.GET.category %}active{% endif %}">
        All
    </a>
    {% for category in categories %}
    <a href="{% pageurl page %}?category={{ category.slug }}"
       hx-get="{% pageurl page %}?category={{ category.slug }}"
       hx-target="#blog-posts"
       hx-swap="innerHTML"
       hx-push-url="true"
       class="filter-btn {% if request.GET.category == category.slug %}active{% endif %}">
        {{ category.name }}
    </a>
    {% endfor %}
</div>

<div id="blog-posts" class="blog-grid">
    {% include "blog/includes/post_cards.html" with posts=posts %}
</div>
```

3. FAQ Accordion with Alpine.js

```
<!-- templates/components/faq_accordion.html -->
<div class="faq-section" x-data="{ openItem: null }">
    {% for item in section.questions %}
    <div class="faq-item">
        <button
            @click="openItem = openItem === {{ forloop.counter }} ? null : {{ forloop.counter }}"
            class="faq-question"
            :class="{ 'active': openItem === {{ forloop.counter }} }"
        >
            <span>{{ item.question }}</span>
            <svg
                class="faq-icon"
                :class="{ 'rotate-180': openItem === {{ forloop.counter }} }"
                xmlns="http://www.w3.org/2000/svg"
                viewBox="0 0 24 24"
                fill="none"
                stroke="currentColor"
            >
                <path d="M6 9l6 6 6-6"/>
            </svg>
        </button>
        <div
            x-show="openItem === {{ forloop.counter }}"
            x-collapse
            class="faq-answer"
        >
            {{ item.answer|richtext }}
        </div>
    </div>
    {% endfor %}
</div>
```

4. Mobile Navigation with Alpine.js

```
<!-- templates/includes/header.html -->
<header x-data="{ mobileMenuOpen: false }" class="site-header">
    <div class="header-container">
        <a href="/" class="logo">
            {% if settings.settings.SiteSettings.logo %}
                {% image settings.settings.SiteSettings.logo width-200 as logo_img %}
                <img src="{{ logo_img.url }}" alt="{{ settings.settings.SiteSettings.site_name }}">
            {% else %}
                {{ settings.settings.SiteSettings.site_name }}
            {% endif %}
        </a>

        <nav class="main-nav" :class="{ 'open': mobileMenuOpen }">
            {% for item in navigation.items %}
            <a href="{{ item.link }}" class="nav-link">{{ item.title }}</a>
            {% endfor %}
            <a href="{% url 'contact' %}" class="btn btn-primary nav-cta">Work With Me</a>
        </nav>

        <button
            @click="mobileMenuOpen = !mobileMenuOpen"
            class="mobile-menu-toggle"
            :class="{ 'active': mobileMenuOpen }"
            aria-label="Toggle menu"
        >
            <span></span>
            <span></span>
            <span></span>
        </button>
    </div>
```

```
</header>
```

5. Testimonial Carousel with Alpine.js

```
<!-- templates/components/testimonial_slider.html -->
<div
    x-data="{
        currentSlide: 0,
        testimonials: {{ testimonials|length }},
        autoplay: null,
        init() {
            this.autoplay = setInterval(() => this.next(), 5000);
        },
        next() {
            this.currentSlide = (this.currentSlide + 1) % this.testimonials;
        },
        prev() {
            this.currentSlide = this.currentSlide === 0 ? this.testimonials - 1 : this.currentSlide - 1;
        }
    }"
    @mouseenter="clearInterval(autoplay)"
    @mouseleave="autoplay = setInterval(() => next(), 5000)"
    class="testimonial-slider"
>
    <div class="testimonial-track">
        {% for testimonial in testimonials %}
        <div
            class="testimonial-slide"
            x-show="currentSlide === {{ forloop.counter0 }}"
            x-transition:enter="transition ease-out duration-300"
            x-transition:enter-start="opacity-0 transform translate-x-4"
            x-transition:enter-end="opacity-100 transform translate-x-0"
        >
            <blockquote class="testimonial-quote">
                "{{ testimonial.value.quote }}"
            </blockquote>
            <cite class="testimonial-author">
                {% if testimonial.value.author_image %}
                    {% image testimonial.value.author_image fill-80x80 as author_img %}
                    <img src="{{ author_img.url }}" alt="{{ testimonial.value.author_name }}" class="author-
                {% endif %}
                <span class="author-name">{{ testimonial.value.author_name }}</span>
                {% if testimonial.value.author_title %}
                <span class="author-title">{{ testimonial.value.author_title }}</span>
                {% endif %}
            </cite>
        </div>
        {% endfor %}
    </div>

    <div class="slider-controls">
        <button @click="prev()" class="slider-btn prev" aria-label="Previous">
```

```
            <svg><!-- Arrow left --></svg>
        </button>
        <div class="slider-dots">
            {% for testimonial in testimonials %}
            <button
                @click="currentSlide = {{ forloop.counter0 }}"
                :class="{ 'active': currentSlide === {{ forloop.counter0 }} }"
                class="slider-dot"
                aria-label="Go to slide {{ forloop.counter }}"
            ></button>
            {% endfor %}
        </div>
        <button @click="next()" class="slider-btn next" aria-label="Next">
            <svg><!-- Arrow right --></svg>
        </button>
    </div>
</div>
```

Project Structure

```
coaching_site/
|---- coaching_site/              # Main Django project
|     |---- settings/
|     |     |---- base.py
|     |     |---- dev.py
|     |     `---- production.py
|     |---- urls.py
|     `---- wsgi.py
|---- home/                       # Home page app
|     |---- models.py
|     |---- templates/
|     |     `---- home/
|     |         `---- home_page.html
|     `---- ...
|---- pages/                      # About, Contact, etc.
|     |---- models.py
|     `---- templates/
|         `---- pages/
|             |---- about_page.html
|             |---- contact_page.html
|             |---- testimonials_page.html
|             |---- resources_page.html
|             |---- faq_page.html
|             `---- generic_page.html
|---- services/                   # Services/Offerings
|     |---- models.py
|     `---- templates/
|         `---- services/
|             |---- services_index_page.html
|             `---- service_page.html
|---- blog/                       # Blog
|     |---- models.py
|     `---- templates/
|         `---- blog/
|             |---- blog_index_page.html
|             |---- blog_page.html
|             `---- includes/
|                 `---- post_card.html
|---- core/                       # Shared components
|     |---- templatetags/
|     |     `---- core_tags.py
|     |---- views.py               # Newsletter signup, contact form, etc.
|     `---- forms.py
|---- settings_app/              # Site settings
|     `---- models.py
|---- templates/
|     |---- base.html
|     |---- includes/
|     |     |---- header.html
|     |     |---- footer.html
|     |     |---- newsletter_section.html
|     |     `---- social_links.html
|     `---- components/
|         |---- cta_banner.html
|         |---- testimonial_slider.html
|         |---- faq_accordion.html
|         `---- ...
|---- static/
|     |---- css/
|     |     |---- main.css          # Compiled Tailwind
|     |     `---- components/
|     |---- js/
|     |     |---- htmx.min.js
|     |     |---- alpine.min.js
|     |     `---- main.js
|     `---- images/
|---- media/                      # User uploads
|---- tailwind.config.js
|---- requirements.txt
|---- Dockerfile
|---- docker-compose.yml
`---- manage.py
```

Requirements.txt

```
# Core
Django>=5.0,<6.0
wagtail>=6.0,<7.0
```

```
psycopg2-binary>=2.9
gunicorn>=21.0

# Wagtail extensions
wagtail-modeladmin>=2.0
django-taggit>=4.0
wagtail-color-panel>=1.4

# Storage
boto3>=1.28
django-storages>=1.14

# Utils
python-dotenv>=1.0
Pillow>=10.0
django-compressor>=4.4
whitenoise>=6.6

# Forms
django-crispy-forms>=2.1
crispy-tailwind>=0.5

# Development
django-debug-toolbar>=4.2
```

Development Commands

```
# Create project
django-admin startproject coaching_site
cd coaching_site

# Create apps
python manage.py startapp home
python manage.py startapp pages
python manage.py startapp services
python manage.py startapp blog
python manage.py startapp core
python manage.py startapp settings_app

# Initial setup
python manage.py migrate
python manage.py createsuperuser

# Run development server
python manage.py runserver

# Tailwind (if using django-tailwind)
python manage.py tailwind start

# Collect static
```

```
python manage.py collectstatic
```

Integration Considerations

Email Marketing

- **ConvertKit** or **Flodesk** are popular with coaches
- Webhook integration for form submissions
- Tag/segment subscribers based on lead magnet

Booking/Scheduling

- **Calendly** - Easy embed integration
- **Acuity Scheduling** - More features
- Use iframe or popup embeds

Payment Processing (if selling courses)

- **Stripe** for payments
- Consider **Teachable**, **Thinkific**, or **Kajabi** for course hosting
- Or build custom with django-payments

SEO

- `wagtail-seo` package
- Open Graph meta tags
- Structured data for coaching business (LocalBusiness schema)

Deployment Checklist

- [ ] Set DEBUG=False
- [ ] Configure allowed hosts
- [ ] Set up PostgreSQL database
- [ ] Configure S3 for media storage

- [ ] Set up SSL certificate
- [ ] Configure email sending (SendGrid, AWS SES)
- [ ] Set up monitoring (Sentry)
- [ ] Configure backups
- [ ] Set up CI/CD pipeline
- [ ] Performance optimization (caching, CDN)

Next Steps

- **Gather Requirements**: Interview your friend about her specific services, brand colors, and content
- **Design Mockups**: Create wireframes/mockups based on the reference site
- **Set Up Development Environment**: Docker + PostgreSQL
- **Build Core Models**: Start with HomePage and basic pages
- **Create Templates**: Build out the design system in HTML/CSS
- **Add Interactivity**: Implement HTMX and Alpine.js features
- **Content Migration**: Help populate initial content
- **Testing**: Cross-browser and mobile testing
- **Deploy**: Set up hosting and domain

Questions to Ask Your Friend

- What is her coaching niche/specialty?
- What services does she currently offer? (1:1, groups, courses?)
- Does she have existing brand colors/fonts?
- Does she have professional photos?
- What email marketing platform does she use?
- Does she need course/membership functionality?
- What booking system does she prefer?
- Does she need e-commerce functionality?
- Who is her ideal client?
- What's her budget and timeline?