Django Full Stack Website Workflow

1. Overall Website Workflow

User → Frontend Pages (HTML/CSS/JS)
→ Django URLs
→ Views (Business Logic)
→ Models (Database)
→ Views
→ Templates (Response to User)

2. Pages to Create

A. Public Pages (No Login Required)
These are accessible to all users.

1. **Home Page**
   - Website introduction
   - Featured services/products
   - CTA buttons (Login / Register / Contact)
   - URL: /
2. **About Us Page**
   - Company/platform information
   - Vision, mission
   - URL: /about/
3. **Services / Features Page**
   - What your platform provides
   - Cards / sections for services
   - URL: /services/
4. **Contact Page**
   - Contact form (Name, Email, Message)
   - Store messages in database
   - URL: /contact/

B. Authentication Pages

Handled via Django authentication (custom or default).

5. **User Registration**
   - Create new account
   - Fields: Username, Email, Password
   - URL: /register/
6. **Login Page**
   - Authenticate user
   - URL: /login/
7. **Logout**
   - End user session
   - URL: /logout/

C. User Dashboard (After Login)

8. **User Dashboard**
   - Personalized welcome
   - User data summary
   - URL: /dashboard/
9. **Profile Page**
   - View & update user details
   - URL: /profile/

D. Core Functional Pages (Based on Project Logic)

(These depend on your project type – example: service / product platform)

10. **Create / Add Page**
    - User adds content (product, post, record, etc.)
    - URL: /add/

11. **List / View Page**
    - View all records
    - URL: /list/
12. **Detail Page**
    - Single record view
    - URL: /detail/<id>/
13. **Edit Page**
    - Update existing record
    - URL: /edit/<id>/
14. **Delete Functionality**
    - Remove record
    - URL: /delete/<id>/

E. Admin Pages (Superuser)

15. **Admin Dashboard**
    - Manage users
    - Manage content
    - View reports
    - URL: /admin/

3. Django Apps Structure

```
AutoBell/
 │
 ├─── accounts/    → Login, Register, Profile
 ├─── core/        → Home, About, Contact
 ├─── dashboard/   → User dashboard logic
 ├─── services/    → Main business logic
 ├─── templates/
 ├─── static/
 └─── manage.py
```

4. Files Inside Each App

```
services/
 ├─── migrations/
 ├─── models.py     → Database tables
 ├─── views.py      → Logic
 ├─── urls.py       → URL mapping
 ├─── forms.py      → Django forms
 ├─── admin.py      → Admin panel config
 └─── templates/services/
      ├─── add.html
      ├─── list.html
      ├─── detail.html
      └─── edit.html
```

5. Database Model Example

```
class Service(models.Model):
   title = models.CharField(max_length=200)
   description = models.TextField()
   created_by = models.ForeignKey(User, on_delete=models.CASCADE)
   created_at = models.DateTimeField(auto_now_add=True)
```

6. Request–Response Workflow

Example: User Adds a Record

1. User opens **Add Page**
2. Fills form → clicks submit
3. Request sent to views.py
4. Django validates form
5. Data saved in **database**
6. User redirected to **List Page**

7. Authentication Workflow

```
Register → Login → Session Created
             → Dashboard Access
             → Logout → Session Destroyed
```

8. Frontend Structure

```
templates/
├──── base.html       → Navbar, footer
├──── home.html
├──── about.html
├──── login.html
├──── register.html
├──── dashboard.html
```

Use:

- Bootstrap / Tailwind
- Django template inheritance

9. Security & Best Practices

- CSRF protection
- Login required decorators
- Role-based access (Admin / User)
- Form validation
- Password hashing (default Django)

10. Final Development Flow

1. Requirement analysis
2. Database design (ER diagram)
3. Create Django project
4. Create apps
5. Define models
6. Create views & URLs
7. Design templates
8. Add authentication
9. Test CRUD operations
10. Deploy (optional)