Moneyworth Web App Project Report

Group Members: Name1, Name2, Name3, Name4

May 13, 2025

1 Introduction

Moneyworth is a web application designed to help users manage and browse sales listings. The main features include user authentication, a dashboard, browsing and creating sales, and profile management. The application provides a modern, responsive interface with a navigation bar, hero section on the Home page, and a footer on all pages. Key pages include Home, Dashboard, Browse Sales, Create Sale, About, and Contact.

2 Part a: HTML and Django Template Code Implementation

This section presents the HTML and Django template code used for the project layout.

2.1 Base Template Explanation

The base template, `base.html`, defines the overall structure of the site. It includes a navigation bar, a hero (jumbotron) section (on the Home page), and a footer. Content blocks are defined using Django's template block tags, allowing child templates to override specific sections.

```
<!-- templates/core/base.html -->
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>{% block title %}Django Sales App{% endblock %}</title>
    <link rel="stylesheet" href="{% static 'css/bootstrap.min.css' %}">
    <link rel="stylesheet" href="{% static 'css/bootstrap-icons.min.css' %}">
    {% block extra_head %}{% endblock %}
</head>
<body>
    <nav>...navbar code...</nav>
    {% block content %}{% endblock %}
```

```
    <footer>...footer code...</footer>
    <script src="{% static 'js/bootstrap.bundle.min.js' %}"></script>
</body>
</html>
```

2.2 Child Template Example

A child template, such as `home.html`, extends the base template and fills in the content block:

```
<!-- templates/core/home.html -->
{% extends 'core/base.html' %}
{% block title %}Home - Django Sales App{% endblock %}
{% block content %}
<div class="jumbotron">
    <h1>Welcome to the Django Sales App!</h1>
    <p>Browse and manage sales listings easily.</p>
</div>
{% endblock %}
```

2.3 Use of Template Tags

Key Django template tags used include:

- `{% extends %}`: Allows a template to inherit from a base template.
- `{% block %}`: Defines sections that child templates can override.
- `{% static %}`: Resolves the URL for static files (CSS, JS, images).

These tags help organize templates, promote code reuse, and simplify static file management.

2.4 Bonus: URL and View Structure

The `urls.py` and `views.py` files connect URLs to views and templates. For example:

```
# core/urls.py
from django.urls import path
from . import views

urlpatterns = [
    path('', views.home, name='home'),
    path('dashboard/', views.dashboard, name='dashboard'),
    path('sales/', views.browse_sales, name='browse_sales'),
    path('sales/create/', views.create_sale, name='create_sale'),
    path('about/', views.about, name='about'),
    path('contact/', views.contact, name='contact'),
```

```
]

# core/views.py
from django.shortcuts import render

def home(request):
    return render(request, 'core/home.html')
# ...other view functions...
```

3 Part b: Benefits of Django for Responsive Web Apps

Django is ideal for building responsive web applications due to its template system, which separates logic from presentation. It integrates easily with CSS frameworks like Bootstrap, enabling mobile-friendly designs. Django also provides built-in security, scalability, and rapid development features, making it suitable for modern web apps.

4 Part c: Linking Static CSS and JavaScript in Django

To include static files in Django:

1. Set `STATIC_URL` in `settings.py` (e.g., `STATIC_URL = '/static/'`).
2. Add `{% load static %}` at the top of each template.
3. Reference static files using `{% static 'css/bootstrap.min.css' %}` or similar.

Example:

```
{% load static %}
<link rel="stylesheet" href="{% static 'css/bootstrap.min.css' %}">
```

5 Part d: Components of MTV Architecture in Django

Django follows the Model-Template-View (MTV) architecture:

- **Model:** Defines data structure. E.g., the Sales model in `core/models.py` manages product data.
- **View:** Handles requests and returns responses. E.g., functions in `core/views.py` render templates for each page.
- **Template:** Renders HTML. E.g., files in `templates/core/` display content to users.