Question 1 (20 marks)

(Each part is worth 1 mark except where indicated.)

a. Briefly describe three main properties that could reasonably determine the success or failure of an implemented Web application. (3 marks)

b. Briefly describe three important criteria that could reasonably influence the choice of framework to use for Web application development. (3 marks)

c. Name the three fundamental standards that define the World Wide Web.

d. Name two reasons for using cascading style sheets to define HTML document presentation in Web applications.

e. Name the two required elements of every valid HTML 4.01 strict document.

f. When submitting data from an HTML form, when should you use method GET and when should you use method POST?

g. Name the three tiers of the standard Web application architecture.

h. When should input data be validated on the client and when should it be validated on the server?

i. Briefly define a cookie.

j. Name three different methods of maintaining state in Web applications.

k. Name the three most common forms of possible security attacks against Web sites.

l. Briefly describe how Django protects against any one of these attacks.

m. What does the term ”Ajax” stand for and why are Ajax applications currently so popular?

n. Name two main JavaScript features that make Ajax applications possible.

o. What is the main design pattern used in Django applications? (Don’t just give the abbreviation, use words.)

p. What does the term ”XML” stand for? Give one example of the use of XML.

Question 2 (20 marks)

a. Name three important subdirectories that should be present in every Django project directory. (2 marks)

b. Name three important files that should be present in every Django app(lication) directory. (2 marks)

c. Name two methods that should be present in every Django model definition and briefly describe why they are so important. (3 marks)
d. Briefly describe two important methods of improving performance in a Web application to enable greater scalability. (3 marks)

e. Give two reasons why template inheritance is important and give an example of the main tag used to implement template inheritance. (2 marks)

f. Suppose class `Item` defines an item-for-sale model and class `ItemForm` defines a model form derived from class `Item`.

Define a Django view `process_form(request)` that (1) displays an instance of an `ItemForm` using template `item_form.html` if `request` has method `GET`, and (2) receives and validates the form data, including image data, to create and save a new instance of class `Item` in the database if `request` uses method `POST`. In case (2), the method should redirect to the URL pattern with name `index` after saving the new instance and it should redisplay the completed form if the form is invalid. (You may assume all required methods have been imported.) (8 marks)

Question 3 (20 marks)

a. Suppose you are designing a Web application that manages information about student enrolment in university courses. Consider the following database design for such an application.

```python
from django.db import models

class Enrolment(models.Model):
    student_name = models.CharField(max_length=40)
    student_number = models.IntegerField()
    student_email = models.EmailField()
    course_code = models.CharField(max_length=8)
    course_name = models.CharField(max_length=80)
    course_convenor = models.CharField(max_length=40)
    year = models.IntegerField()
    grade = models.CharField(max_length=2)
```

Clearly describe two different, significant reasons why this is a bad database design. Don’t just say what the correction should be; describe what’s wrong with this design. (Ignore any minor syntax errors, missing fields or constraints, and so on.) (4 marks)

Give a better database design for this application. (6 marks)

b. Consider the following database design that allows a store to keep track of which customers have placed which orders for which items for sale.

```python
class Customer(models.Model):
    name = models.CharField(max_length=40)
    address = models.CharField(max_length=80)
    email = models.EmailField()
```

...
class Item(models.Model):
    name = models.CharField(max_length=80, unique=True)
    code = models.IntegerField(unique=True)
    price = models.IntegerField()

class Order(models.Model):
    customer = models.ForeignKey(Customer)
    item = models.ForeignKey(Item)
    quantity = models.IntegerField()
    date = models.DateField(auto_now_add=True)

The query “Which customers live in Nathan?” can now be expressed using the database API by the following Django statement.

```python
customers = Customer.objects.filter(address__icontains="Nathan")
```

Write Django statements using the database API that express the following queries.

(i) What is the name of the item whose code is 1234?
(ii) What are the names of the items whose price is greater than 100?
(iii) What are the names of the customers who have ordered items with “camera” in their names?

(6 marks)

c. What is an atomic database transaction? Briefly describe one situation in which something can go wrong if transactions are not executed atomically. Briefly describe one way to ensure transactions are executed atomically in Django.

(4 marks)

---

Question 4 (26 marks)

Describe how you would design a Web application in Django to provide a new online auction system such as eBay. In brief, the system should allow users to register, offer items for sale, browse items offered for sale, and bid for items offered for sale.

In more detail, the system should operate as follows.

Users may register to become members, and members may login and logout. Each member has a unique username, first and last names, an email address, a phone number, and other information (which may be ignored for this question).

Items advertised for sale have a category, a name, a description, a starting price, a current highest price, and a closing datetime (the time when the auction for the item finishes). Categories are strings such as “books”, “phones”, “cameras”, etc.
Vendors (i.e., sellers) may login and advertise items for sale. To advertise an item for sale, a vendor must provide the item’s category, name, description, starting price and the item’s closing datetime.

Buyers may login and bid for an advertised item until the auction’s closing date. Each bid by a buyer for an item has an offer price and a datetime. Each bid for an item must be strictly higher than the previous highest bid for the item. Vendors cannot bid for their own item, and buyers cannot bid after the item’s closing datetime.

Users may browse the items currently advertised for sale, i.e., the items whose auctions have not yet closed, by category. Users may also search for items by words in their names or descriptions. Users may view all details on a current item, including the history of bids made for the item. The results of browsing and searching are displayed using the list-detail pattern.

Your description should include the following aspects of the design:

a. A database design (models.py): Describe all database models required, indicating the purpose of each model and field by carefully chosen names or by comments, and including any constraints that must hold. (8 marks)

b. A set of URL patterns (urls.py): Describe the set of URL patterns required and the (Django) view each pattern is associated with. (Views may be either user-defined views, generic views or (Django) library views.) (6 marks)

c. A set of Django views (views.py): Describe the most important view functions required. For each view, describe carefully, in English, exactly what it does, noting whether it returns an HTML page or redirects to a URL. (6 marks)

d. A transition diagram: Describe the state transitions the application makes during execution, where each state (or node) of the diagram is associated with a URL pattern whose view either generates an HTML page or updates the database. Label each node with its corresponding URL pattern. For simplicity, ignore all states and transitions associated with user registration, login and logout. (6 marks)

Question 5 (14 marks)

a. What is the difference between a news feed and a news aggregator? Briefly describe (a) Atom and (b) AtomPub. (6 marks)

b. Briefly describe the purpose of the tools Play!, Struts 2, Ant and IntelliJ in Web development using Java EE, relating these tools to Django equivalents where possible. (4 marks)

c. Briefly describe two significant differences between Web development in ASP.NET and Django. (4 marks)