Question	Points	Score
Question 1	5	
Question 2	10	
Question 3	10	
Question 4	10	
Question 5	5	
Question 6	5	
Question 7	5	
Total	50	

Answer all questions in the space provided here.

Part A. Unit Testing

Question 1 .... **5 marks**

In Lecture 3 we discussed Boundary Value Testing (BVT) and the (minimum) number of test cases for the "normal" (non-robust) version of BVT for n variables $(4n + 1)$. In this question, for **robust BVT**, do the following:

(a) Work out a formula for the minimum number of test cases.
(b) Briefly explain why.

Question 2 . . . . **10 marks**

In Lecture 3 we showed a triangle example to demonstrate test case generation for BVT (slide 11). In the example each of the three variables a, b and c is the length of a side from the range $[1, 200]$. In this question, use Equivalence Class Testing and Decision Table techniques to design test cases for the triangle problem.

(a) (5 marks) Come up with test cases for weak normal equivalence class testing that cover the same expected outputs (isosceles, equilateral, scalene, not a triangle).

(b) (5 marks) Based on your equivalence classes, develop a decision table for the triangle problem.

Part B. Integration Testing & System Testing

During the semester we've been working on the *Fly me to Mars* project. In this project, the goal is to create a simple Web application for mission management, dealing with `Persons`, `Missions` and `Invitations`. In the code base of the last part of the project, individual components have been integrated together to create a fully functional Web application.

Question 3 .... **10 marks**

With some simplification, the loading of the `Mission` page is shown below in Figure 1. The component `Router` is responsible for routing HTTP requests to the appropriate *resource* that handles them. As the name suggests, `MissionResource` is responsible for creating pages to present mission details. Once authentication and authorisation are determined to be required, the resource consults the `LoginManager` to grant/refuse access based on user input. The `LoginManager` in turn invokes `UserDAO` to load user details given the user name. If access is granted, the resource then invokes `MissionDAO` to obtain mission details. If access control is not required, or that the user is denied access, it constructs a page correspondingly. Finally, `EntityResource` returns the page back to the `Router`.

Note that the graphs within each component represent greatly simplified program graphs, with only some important control structures preserved. Nodes in these graphs don't represent individual statements, but rather blocks of statements that can be grouped together logically. The *italic label* beside each node explains the main functionality of that node. Edges still represent control flow. Thick edges represent transfer of control (message passing and return) between components.

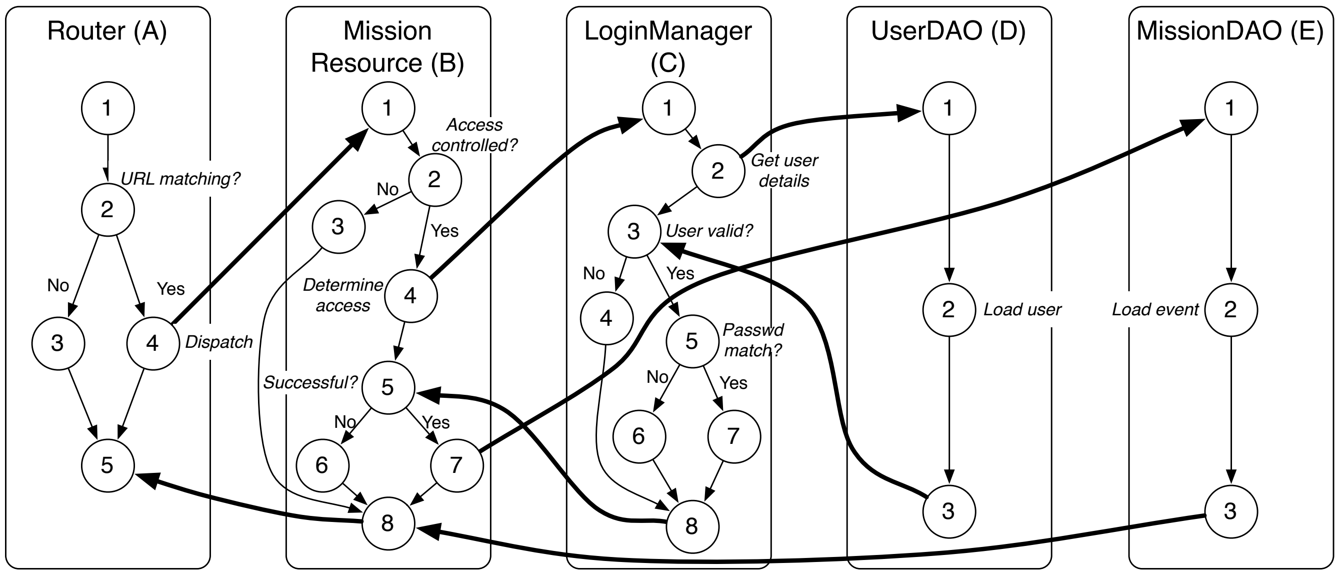

Figure 1: The interactions of some components of the *Fly me to Mars* system that handle mission presentation.

Draw a sequence diagram depicting the interactions between the above objects. Note that for simplicity reasons, do not include DAO objects as `PersonDAO` and model objects such as `Person`, `Mission` and `Invitation`.

Question 4 . . . . **10 marks**

Finite state machines can be very useful in system testing by capturing system functionality in terms of states, events, transitions, actions and guards. Suppose when we are creating an invitation, we need three inputs: a **name**, a **location** and a **receipient**. Draw a finite state machine depicting this scenario.

Hint: Think about what states represent.

Part C. Software Metrics

Question 5 .... **5 marks**

At the end of last lecture we introduced Weyuker's 9 properties to evaluate software metrics. Some of the properties (for example, properties 1, 3, 4 and 8) are quite simple and intuitive. However, some other properties are a bit more involving and needs further analysis. Property 5 states that the complexity of a program segment should be less than or equal to that of the whole program, i.e., $\forall P, Q @ M(P) \leq M(P+Q) \wedge M(Q) \leq M(P+Q)$.

For the above property 5 and the software science metric Program Volume metric: $V = N \times log_2 \mu$, do the following:

(a) State whether the property holds or not.
(b) Prove your claim (informally).

Part D. Object-oriented Testing

Question 6 .... **5 marks**

In the third part of the course project, we have implemented the user access control functionality. When a user access a *resource* (represented by a URL), he is prompted to authenticate himself (if he hasn't). After successful authentication, authorisation is performed to determine whether access should be granted to the authenticated user. In this question, draw a **sequence diagram** depicting the interactions among internal classes. You may include any external classes (Spring Security classes, for example) as required.

Part E. Other Verification Techniques

Question 7 . . . . 5 marks

Formal verification techniques, including model checking and theorem proving, work on mathematical abstractions of software systems. Systems and properties to be verified are expressed in certain mathematical logics. Inference is then performed to formally verify the truthfulness of the property. Discuss the pros and cons of these formal verification techniques and compare them against testing.