Romerico David

11/21/2024

Professor Sai, Assignment 3

Question 1a:

Architectural Style: Layered Architecture

Assumptions:

1. The system has distinct responsibilities like handling RFID sensor data, calculating parking demand, processing payments, and managing user interactions.
2. The system benefits from separating into layers to allow independent development and maintenance of different functionalities.
3. Each layer can interact with adjacent layers to provide modularity and flexibility.

Justification:

1. **Separation of Layers:**
    - **Presentation Layer:** Handles user interactions through kiosks or mobile apps like purchasing parking passes and checking availability.
    - **Application Logic Layer:** Implements business logic like calculating demand-based prices and validating parking durations.
    - **Data Access Layer:** Communicates with the database for storing and retrieving parking usage data and historical demand.
    - **Infrastructure Layer:** Interacts with physical RFID sensors and payment gateways.
2. **Modularity:** Each layer has a well-defined responsibility making the respective layer easier to develop, test, and maintain.
3. **Scalability:** Layers can be scaled without affecting other layers.
4. **Easy Testing:** Since layers are independent, issues can be traced and resolved within specific layers without impacting the entire system.

Question 1b: Usability Metric and Evaluation Plan

Chosen Usability Metric: Effectiveness

1. **Scope:**

- Evaluate the users' success rate in completing parking-related tasks like checking parking availability, calculating parking fees, and completing payments using the kiosk system.

2. **Purpose:**
   - Ensure the system allows users to accurately complete their parking goals and identify areas where users can encounter errors or fail to complete tasks.
3. **Sessions:**
   - Each session will involve users performing three tasks:
     1. Check parking availability for a nearby garage.
     2. Purchase a parking pass for 2 hours using a credit card.
     3. Pay for a parking pass using a TU OneCard.
   - Each session will last approximately 10–15 minutes.
4. **Equipment:**
   - A fully functional parking kiosk or a software simulation.
   - A logging system to record user interactions and completion rates.
5. **Participants:**
   - 10 participants, including a mix of students, faculty, and visitors and they should have different levels of familiarity with parking systems.
6. **Scenarios:**
   - **Scenario 1:** A user successfully checks parking availability and reserves a spot in a specific garage.
   - **Scenario 2:** A user completes the payment process using their TU OneCard without errors.
   - **Scenario 3:** A user encounters a failed credit card transaction and retries successfully.
7. **Metrics:**
   - **Task Success Rate:** Percentage of users who successfully complete each task without errors.
   - **Error Rate:** Number of user errors during each task
   - **Completion Rate:** Percentage of tasks completed versus attempted.
8. **Quantitative Metrics:**
   - **Task Success Rate Target:** 78% of users should successfully complete all tasks.
   - **Error Rate Target:** Less than 5% of total interactions should result in errors.
   - **Completion Rate Target:** 100% of tasks attempted should be completed.

Question 2a:

User	Task 1 Time	Task 2 Time	Task 3 Time
User 1	3 (success)	4 (success)	3 (success)
User 2	5 (success)	6 (failed)	4 (success)
User 3	5 (success)	6 (failed)	7 (failed)
User 4	6 (success)	9 (success)	3 (success)

- $N = 3$
- $R = 4$

Time-based Efficiency: $\frac{\sum_{j=1}^{R}\sum_{i=1}^{N}\frac{n_{ij}}{t_{ij}}}{NR} = \frac{(\frac{1}{3}+\frac{1}{4}+\frac{1}{3}+\frac{1}{5}+\frac{0}{6}+\frac{1}{4}+\frac{1}{5}+\frac{0}{6}+\frac{0}{7}+\frac{1}{6}+\frac{1}{9}+\frac{1}{3})}{\mathbf{12}} = 0.18 goals/sec$

Overall Relative Efficiency: $\frac{\sum_{j=1}^{R}\sum_{i=1}^{N} n_{ij}t_{ij}}{\sum_{j=1}^{R}\sum_{i=1}^{N} t_{ij}} \times 100 = \frac{(3+4+3+5+4+5+6+9+3)}{(3+4+3+5+6+4+5+6+7+6+9+3)} \times 100 = 68.85\%$

Question 2b:

i. Alice obtained the password to Bob's bank account, withdrew some amount from his account, and changed the password.

- **Violations:**
  - **Confidentiality:** Alice obtained Bob's password, which is private information that should not be disclosed to unauthorized users.
  - **Integrity:** Alice changed the password and withdrew money altering Bob's account data without authorization.
- **Justification:** Confidentiality is violated because sensitive information was accessed. Integrity is violated because the account was altered by withdrawing money and changing the password.

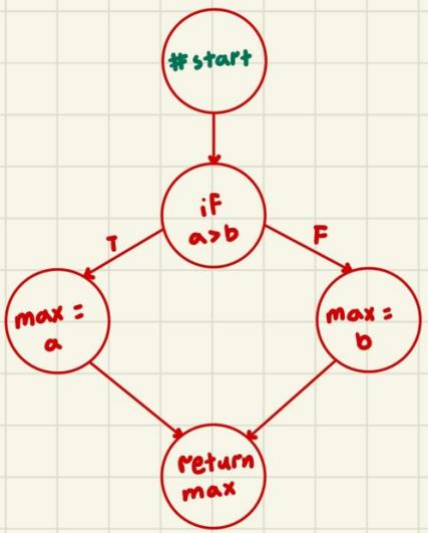
ii. A hacker gained access to a hospital's patient record database and locked it.

- **Violations:**
  - **Availability:** The hacker locked the database preventing legitimate users (hospital staff) from accessing patient records.
  - **Confidentiality:** The hacker gained unauthorized access to sensitive patient data.
- **Justification:** Availability is violated because the database is no longer accessible to authorized users. Confidentiality is violated due to unauthorized access to private patient information.

iii. A scammer impersonates the police, obtains your personal information over the phone, and uses it to open a new credit line.

- **Violations:**
  - **Confidentiality:** The scammer obtained sensitive personal information.
  - **Integrity:** The scammer used the stolen information to create a fraudulent credit line altering your financial records.
- **Justification:** Confidentiality is violated when personal information is disclosed to an unauthorized individual (scammer). Integrity is violated when the stolen data is used to manipulate financial records.

Question 3:



Test Case	Input	Expected	Actual
1	a = 5, b = 6	max = 6	max = 6
2	a = 100, b = 120	max = 120	max = 120

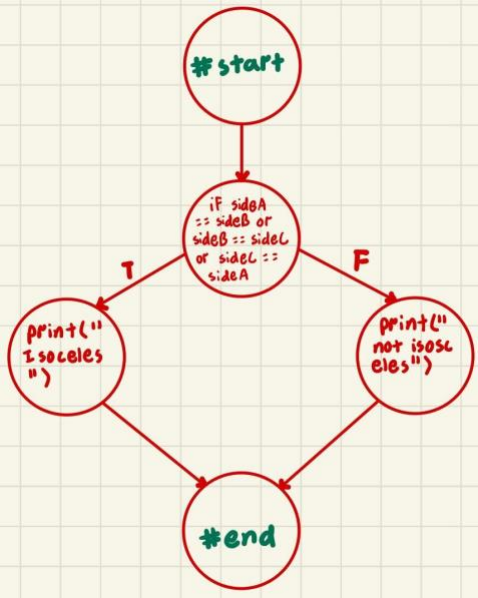
- Statements covered: 1, 3, 4
- Total statements: 4

Statement Coverage: $\frac{3}{4} \times 100 = 75\%$

The statement coverage is incomplete because the true branch (max = a) is never executed because both test cases only evaluate test cases where a<b. To get full statement coverage, we can add a new test case where a>b. For example, create a third test case with a=8 and b=4.

I disagree with the statement "If your statement coverage is good, there is no need to perform branch coverage." Statement Coverage only ensures that all statements in the code are executed. It does not ensure that all decision outcomes (True/False) are tested. Branch Coverage ensures that all outcomes of each decision are tested which subsumes Statement Coverage.

Question 4:



Test Case	Input	SideA == SideB	SideB == SideC	SideA == SideC	Decision
1	a = 5, b = 5, c = 6	True	False	False	True
2	a = 5, b = 6, c = 7	False	False	False	False
3	a = 5, b = 5, c = 5	True	True	True	True

Condition Coverage = 100% (All conditions are tested for both True and False values)

Decision Coverage = 100% (The decisions evaluate to both True and False)

Overall Condition/Decision Coverage = 100%

We can use a **Modified Condition/Decision Coverage** to ensure each condition in a decision independently affects the outcome of the decision which minimizes redundant test cases and reduces the number of test cases required.

Application to Problem:

Test Case	SideA == SideB	SideB == SideC	SideA == SideC	Decision
1	True	True	True	True
2	False	True	True	True
3	True	False	True	True
4	True	True	False	True
5	False	False	True	True
6	False	True	False	True
7	True	False	False	True
8	False	False	False	False

MC/DC Coverage:

Test Case	SideA == SideB	SideB == SideC	SideA == SideC	Decision
1	True	True	True	True
8	False	False	False	False

There were 8 possible outcomes but after using MC/DC, the number of outcomes eliminated were 6.

Question 5:

I find it frustrating that I have to grant network access to my teammates (without using 0.0.0.0/0 to not promote bad security) for them to connect to my MongoDB cluster. Initially, I planned to learn and set up Docker for our application but given the current state of the project—two inactive members and two teammates with no experience in full-stack development—it's not realistic. Asking those two teammates to learn backend development on top of React for the frontend within two weeks is unrealistic. However, with more time, I would implement Docker to containerize the backend which would simplify the local deployment of our backend API on our machines.