19

Draw a use case diagram for a ticket distributor for a movie theatre. The system includes two actors: a movie watcher who purchases different movie tickets, and a central computer system that maintains a reference database for the ticket price and cinema type. Use cases should include BuyGeneralTicket, BuyiMAXTicket, BuyLUXTicket, UpdateTicketPrice, UpdateMovieSession and SelectMovieSession. Also include the following exceptional cases: TimeOut (i.e., movie watcher took longer than 10 minutes to complete the process), TransactionAborted (i.e., movie watcher selected the cancel button without completing the process), CreditCardTransactionFailure, SessionFull and SystemOutOfPaper.

Question 4: Draw a class diagram representing a banking system defined by the following statement: 3 marks Each bank has location, unique code and a manager. A bank has many customers and each customer can open maximum of two bank accounts: saving account or current account. A customer has name, address, mobile number, card number and pin code for his account(s). Each bank account (either saving or current) has unique account number and belongs to a particular customer. Customers can deposit or withdraw cash from any of his or her bank accounts. You must add an abstract class and an inheritance relationship to factor out common attributes into the abstract class (if required).

Question 7:

5 marks

Write the name of the most relevant software design pattern in front of each of the natural language heuristics for the software product.

• The product must be platform and manufacturer independent

• The product must be able to support future protocols

• The product must comply with existing software and must reuse existing legacy component

• All transactions should be logged in the system and be undoable

• The product must allow different algorithms to be interchanged for a task

18

Question 7: 2 marks Briefly explain Requirements Elicitation. Mention four different techniques of requirements elicitation.

Write the name of the software design pattern in front of each of the following situations? You

need to mention the most optimal software design pattern which should be for each of the

following situations.

• Allowing for alternate implementation

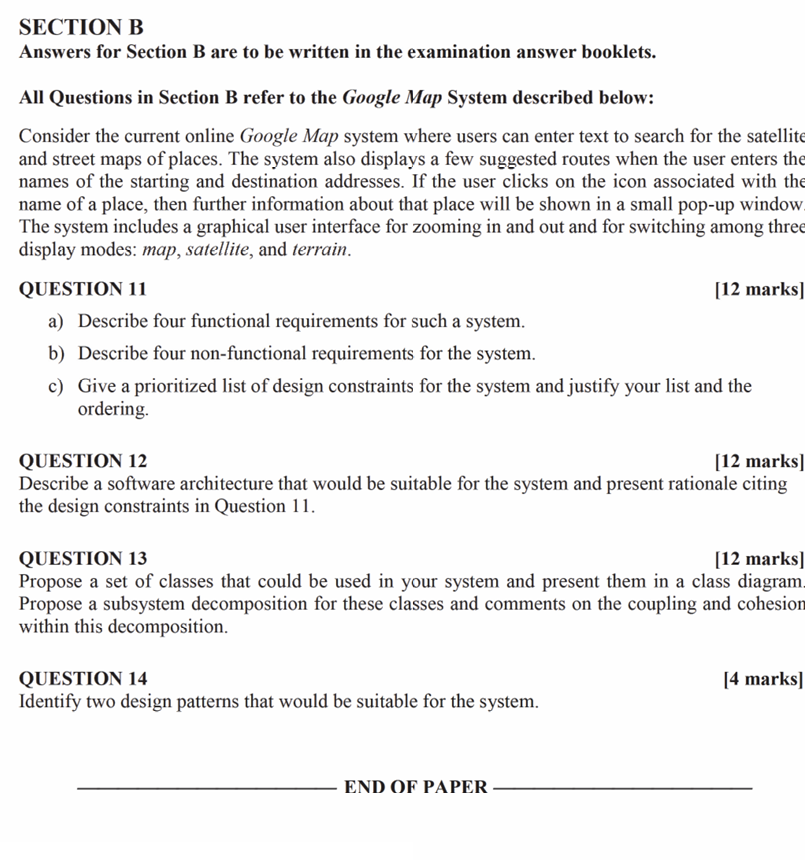
• Encapsulating subsystems

• Encapsulating algorithms

• Encapsulation control flow

• Wrapping around legacy code

17



**Question 11**

**a) Describe four functional requirements for such a system (4 marks)**

1. **Search for Locations:**
   * **Description:** The system must allow users to enter text to search for the satellite and street maps of places.
   * **Reason:** To provide users with the ability to find specific locations.
   * **Example:** A user enters "Eiffel Tower" and the system displays the location on the map.
2. **Route Suggestions:**
   * **Description:** The system must display a few suggested routes when the user enters the names of the starting and destination addresses.
   * **Reason:** To assist users in finding the best routes for their journeys.
   * **Example:** A user enters a starting point and destination, and the system suggests the shortest and fastest routes.
3. **Display Place Information:**
   * **Description:** When a user clicks on an icon associated with the name of a place, the system must show further information about that place in a small pop-up window.
   * **Reason:** To provide detailed information about specific places.
   * **Example:** A user clicks on a restaurant icon and sees details such as address, opening hours, and reviews.
4. **Interactive Map Display:**
   * **Description:** The system must include a graphical user interface for zooming in and out and for switching among three display modes: map, satellite, and terrain.
   * **Reason:** To offer users multiple views and interactive control over the map display.
   * **Example:** A user can zoom in to see street details or switch to satellite view for a more realistic representation.

**b) Describe four non-functional requirements for the system (4 marks)**

1. **Performance:**
   * **Description:** The system should load maps and route information quickly, within 2 seconds of user input.
   * **Reason:** To ensure a smooth and responsive user experience.
   * **Example:** When a user searches for a location, the results should appear almost instantly.
2. **Scalability:**
   * **Description:** The system must handle a large number of simultaneous users without degradation in performance.
   * **Reason:** To support a growing user base and high traffic periods.
   * **Example:** The system should maintain performance levels even during peak usage times.
3. **Usability:**
   * **Description:** The system should have an intuitive and user-friendly interface.
   * **Reason:** To ensure that users can easily navigate and use the system’s features.
   * **Example:** Clear icons and simple navigation for searching locations and switching views.
4. **Security:**
   * **Description:** The system must ensure secure handling of user data and protect against unauthorized access.
   * **Reason:** To protect user privacy and sensitive information.
   * **Example:** Implementing HTTPS for secure data transmission.

**c) Give a prioritized list of design constraints for the system and justify your list and the ordering (4 marks)**

1. **Performance:**
   * **Reason:** Fast response times are crucial for maintaining a good user experience. Delays in loading maps or route information can frustrate users and lead to abandonment of the system.
   * **Example:** Ensuring that maps load within 2 seconds of user input.
2. **Scalability:**
   * **Reason:** The system must be able to handle a large number of users simultaneously to ensure availability and reliability during peak times.
   * **Example:** The system should be designed to scale horizontally by adding more servers as the user base grows.
3. **Usability:**
   * **Reason:** An intuitive and user-friendly interface is essential for user satisfaction and ease of use. Users should be able to quickly learn and navigate the system.
   * **Example:** Designing clear and simple navigation for searching locations and switching views.
4. **Security:**
   * **Reason:** Protecting user data and ensuring secure transactions are vital to maintaining user trust and compliance with data protection regulations.
   * **Example:** Implementing HTTPS and secure authentication methods to protect user data.

**Question 12**

**Describe a software architecture that would be suitable for the system and present rationale citing the design constraints in Question 11 (12 marks)**

**Recommended Architecture: Microservices Architecture**

**Description:**

* The system is divided into small, independent services that communicate over a network. Each service is responsible for a specific piece of functionality, such as search, route suggestions, place information, and map display.

**Components:**

1. **Search Service:**
   * Handles user queries for locations and returns relevant map data.
2. **Routing Service:**
   * Provides route suggestions based on starting and destination addresses.
3. **Place Information Service:**
   * Displays detailed information about places when users click on icons.
4. **Map Display Service:**
   * Manages the graphical user interface for zooming and switching display modes.

**Rationale:**

1. **Performance:**
   * **Justification:** Each microservice can be optimized for performance independently, ensuring fast response times. For example, the Search Service can be optimized for quick query processing.
   * **Example:** The Search Service quickly processes user queries and returns results within 2 seconds.
2. **Scalability:**
   * **Justification:** Microservices architecture allows for horizontal scaling. Services can be scaled independently based on demand.
   * **Example:** During peak times, more instances of the Routing Service can be deployed to handle increased traffic.
3. **Usability:**
   * **Justification:** The front-end can be developed independently of the back-end services, ensuring a smooth and user-friendly interface. Each service provides a clear API, simplifying the development of the user interface.
   * **Example:** The Map Display Service ensures an intuitive interface for zooming and switching views.
4. **Security:**
   * **Justification:** Each service can implement its own security measures, ensuring secure handling of user data. Microservices can use secure communication protocols for inter-service communication.
   * **Example:** All services communicate over HTTPS to protect user data.

**Question 13**

**Propose a set of classes that could be used in your system and present them in a class diagram. Propose a subsystem decomposition for these classes and comments on the coupling and cohesion within this decomposition (12 marks)**

**Classes:**

1. **User:**
   * Attributes: userID, name, email
   * Methods: login(), logout()
2. **SearchQuery:**
   * Attributes: queryText, locationResults
   * Methods: executeQuery()
3. **Route:**
   * Attributes: startLocation, endLocation, routeDetails
   * Methods: calculateRoute()
4. **Place:**
   * Attributes: placeID, name, details
   * Methods: getPlaceInfo()
5. **Map:**
   * Attributes: mapType, zoomLevel
   * Methods: displayMap(), switchView()

**Class Diagram in PlantUML:**

plantuml

Copy code

@startuml

class User {

+ String userID

+ String name

+ String email

+ void login()

+ void logout()

}

class SearchQuery {

+ String queryText

+ List locationResults

+ void executeQuery()

}

class Route {

+ String startLocation

+ String endLocation

+ List routeDetails

+ void calculateRoute()

}

class Place {

+ String placeID

+ String name

+ String details

+ void getPlaceInfo()

}

class Map {

+ String mapType

+ int zoomLevel

+ void displayMap()

+ void switchView()

}

User -> SearchQuery : performs

User -> Route : requests

User -> Place : views

User -> Map : interacts with

SearchQuery --> Place : retrieves

Route --> Map : uses

@enduml

**Subsystem Decomposition:**

1. **User Interaction Subsystem:**
   * **Classes:** User, Map
   * **Responsibilities:** Handles all user interactions, including logging in, logging out, and interacting with the map interface.
   * **Cohesion:** High cohesion as all classes are focused on user interactions.
   * **Coupling:** Low coupling with other subsystems, communicates through well-defined interfaces.
2. **Search Subsystem:**
   * **Classes:** SearchQuery, Place
   * **Responsibilities:** Manages search queries and retrieves place information.
   * **Cohesion:** High cohesion as classes are focused on search functionality.
   * **Coupling:** Loose coupling with the User Interaction Subsystem, interacts with the Place class to retrieve information.
3. **Routing Subsystem:**
   * **Classes:** Route
   * **Responsibilities:** Calculates routes based on user input.
   * **Cohesion:** High cohesion as it is focused on route calculations.
   * **Coupling:** Interacts with the Map class to display routes, maintains loose coupling by using well-defined interfaces.

**Comment on Coupling and Cohesion:**

* **Cohesion:** Each subsystem is designed to handle specific responsibilities, ensuring high cohesion within subsystems.
* **Coupling:** Subsystems interact through well-defined interfaces, ensuring loose coupling. This design allows for independent development, testing, and maintenance of each subsystem.

**Question 14**

**Identify two design patterns that would be suitable for the system. Briefly justify your selection (4 marks)**

1. **Facade Pattern:**
   * **Justification:** Provides a simplified interface to a complex subsystem, making it easier for the user interface to interact with the system. The Facade pattern hides the complexities of the underlying services and provides a unified interface for common tasks such as searching for locations, calculating routes, and displaying maps.
   * **Example:** A Facade class could provide methods like , , and , encapssearchLocation()getRoute()displayMap()