LNMIIT
The LNM Institute of Information Technology

The LNM Institute of Information Technology

Department: Computer Science and Engineering

Software Engineering (CSE 0326)

Mid Term Examination

Time: 90 mins **Date:** 27/09/2019 **Max. Marks:** 90

Instruction: ***1. All the questions are compulsory.***
2. In Engineering, neatness is important. Thus, neatness carries 5 marks.
3. Make and State Appropriate Assumptions if and when required.

Case Study: M/S North India Electricity Board (NIEB) has engaged Y17 IT Services Inc (Y17ITS) to analyze and design a suitable software module that meets the following functional and non-functional requirements.

Functional Requirements: This software module should offer:

[A]. Processing User's Requests: Users are of two categories: Corporate Users and individual Citizens. Users should be able to sign-in with name, email-id, location address, ID-proof, and contact phone number. Users can request for a new electricity meter and connection for a new building / apartment. NIEB staff verify (i) the ID-proof and (ii) location address using two separate APIs provided by Govt of India. NIEB Regional Manager approves request for a new connection if ID-proof and location address are correct else rejects it. Once a user's request for a new connection is approved, the user should pay a safety-deposit through an online payment gateway given by this sub-module. The safety-deposit amount for a Corporate user is rupees one lakh whereas for a Citizen user, it is rupees five thousand. NGOs are a special case of Corporate users whose safety-deposit is only rupees fifty thousand. Existing users can make a request to change only email-id or contact phone number. NIEB staff use appropriate APIs to verify these changes and NIEB Regional Manager approves the change if correct else rejects the same. Existing users can also make a request for disconnection of electricity meter and connection. NIEB staff verify that all dues are paid and NIEB Regional Manager approves the request if dues are paid else rejects it. NIEB provides Digital Electricity Meters, which have an API to get the latest meter reading. Each user will get a meter with a unique number. This software module should provide an API to social media networks wherein users can offer comments on NIEB and these comments are stored in the user's database.

[B]. Processing Monthly Bills and Payments: On the 1st date of every month, between 00:01 to 03:00 hours, this sub-module uses the API of each meter, gets the latest meter reading and calculates the number of electricity units consumed by each user since the last reading. The Corporate Users are charged rupees one per unit and the Citizens are charged rupees 2 per unit consumed. NGOs are a special case of Corporate users who are charged only 0.5 rupee per unit. The generated bill is emailed to each user on the same day. Each user should pay the complete amount before the 10th of each month using online payment gateway, failing which, on the 11th, the amount is deducted from the safety-

deposit of that user, and a warning email is sent to that user to pay the full amount with a penalty by the 20th, so that the safety-deposit is restored. The penalty amount is rupees thousand for Corporate users, rupees five hundred for NGOs and rupees one hundred for Citizen users. If a user does not pay even by the 20th then the user's electricity is disconnected by NIEB.

Non-functional Requirements: The software module should have, (1) High Availability of 99.99% (only one day down-time in 10,000 days) and (2) High Modifiability of 10 changes to software module in 10 person days.

Questions on Case-Study:

Q.1. Analyze the functional requirements and (a) draw the DFD Level-0 and Level-1 diagrams [Marks 5 + 10 = 15] and (b) draw the Use-Case-Diagram for any one object that has life in the software module [Marks 6] and (c) for any one activity of the above UCD, draw the Use-Case-Activity-Diagram indicating the Units-of-Behavior [Marks 15].

Q.2. Design architecture diagrams for (a) High Availability and (b) High Modifiability [Marks 5 + 5 = 10].

Q.3. Design two Classes indicating at least 6 attributes and 6 methods in each class; show sub-classes if any and show relationship between the two Classes if any [Marks 10 + 10 = 20].

Q.4. Design a Component Diagram showing the APIs to be offered and APIs required [Marks 10].

Q.5. Show, using diagrams, 3 types of inconsistencies that should be avoided in DFDs [Marks 9].

[Total marks = Neatness 5 + Questions 85 = 90].