Software Engineering Case-Based Learning Exercise
LIC Market-Driven System

1) Stakeholders and users of the system

Stockholders & Users:

- ➔ LIC Insurance Company
- ➔ LIC Agents
- ➔ Insurer
- ➔ Retinoids Software company
- ➔ Banks
- ➔ Employees

User Stories:

1. As an Insurance Providing company, we want to develop consolidated insurance packages so that the customer is aware of the financial policies of the company.
2. As an Insurance Providing company, we want that insurance packages should compete with the packages provided by other insurance companies so that we can sustain in the market.
3. As an insurance customer, I want a feature to create my own package by combining different available insurance so that I can select the best option suitable.
4. As a user, I want to be able to see all available policies so that I am able to select the best from all.
5. As a user, I should be able to log in so that I can see all the available policies.
6. As a user, I like to get reminders for my policy installments so that I don't miss and end up paying an extra amount.
7. As a LIC Agent, I should be able to review customer requirements and offer them competing prices.
8. As a LIC Agent, I must be able to cross-check the authenticity of the insurer's documents beforehand.

Prioritization:

We can prioritize user stories using the MoSCoW technique.

High Priority (must have): User Stories 1, 4, 5, 8
Medium Priority (should have): User Stories 2, 7, 3
Low Priority (could have): User Story 6

2) Market - facing technologies for proper deployment of the product

1. Software Applications: Mobile App/Website,
2. Digital kiosks,
3. Rugged tablets,
4. Mobile Point-of-Sale (mPOS),

3) Requirement Engineering Framework:

For this case study, as we don't have any existing systems that can be analyzed to gather relevant information/requirements, so:

- We can conduct **Interviews with higher authority employees** of LIC to understand the project requirements and end deliverables more properly.
- We can build **prototypes** as we proceed further with requirement engineering. And take inputs from stakeholders.
- We can conduct **questionnaires** to gather domain-specific requirements.
- We can do **brainstorming** to settle conflicts and to refine, analyze, decompose requirements.

4) Features that are not feasible to consider :

1) Customer can create his/her own package and send a request for the review - This feature won't be feasible because the number of customers can be very high.
2) Customer providing suggestions and system providing competing price for the package - for the system to analyze every suggestion and providing a competing price on its own is not feasible.

5) Similarity between Default Package & User-defined Package

A possible reason behind such a defect is, we are not validating the user entered package with our default packages, and directly sending it for review.

To avoid this defect, We can check the user-entered package with default packages, and if we find similarity, we directly suggest a similar default package to the user and won't send his/her request for review.

6) Conflict Between Requirements:

1) If the Customer chooses his package the system will process and provide feedback if needed then the system will give you the cost of that package but sometimes that cost is

high compared to other companies and if the customer wants a lower price this will happen then this dispute will not be resolved.

This conflict can be resolved by discussion with LIC Agent and possibly suggesting another package which can be suitable with user requirement.

2) There is also a possibility that the default package has more price compared to user-generated features. So in-case LIC might lose some profit.

   This can be tackled by enhancing the custom package pricing module.

3) Users can add features that may cause hardship for the company and it is kind of a trap.

   This can be taken care of by having some conditions on the features that one policy can hold.

7) Non-Functional requirements:

- **Availability** - The system should be available for the customers 24/7
- **Durability** - All the transactions should stay intact, there should not be any loss to the customer due to system failure.
- **Scalability** - The system should be able to respond to multiple requests and also scalable in the future as the user-base increases.
- **Privacy** - The documents and private information provided by the user should be kept confidential and only visible to that user.
- **Reliability** - The system should live for most of the time.
- **Profitability** - The system should give estimated costs in case of custom packages such that the deal would be a profit maker for the LIC.

8) Open Issues :

Life insurance claim or accident claim - Both the systems need proofs but documents can be forged easily and the system can't physically check whether it's a true claim or not because the system has not been put in place to do so.