**INFO 6350**                                                       **Fall 2024**

**Assignment 7**

In this assignment, you will enhance the previous project by implementing the user interface using a combination of **Interface Builder (IB)** and **Storyboard**. You will also include additional entities like **Payments** and **Claims**.

The goal is to build an Insurance Policy System application that allows users to manage **Customers**, **Insurance Policies**, **Claims**, and **Payments**, ensuring smooth transitions between views with **segues** where applicable.

**Key Task:**

You are responsible for creating an application that performs basic **CRUD** (Create, Read, Update, Delete) operations for all entities, providing a seamless user experience. The UI will be designed using both Interface Builder and Storyboard, with **segues** managing transitions between the different screens.

**Core Requirements:**

**Entities and Fields:**

**Customer**

* id: Integer
* name: String
* age: Integer
* email: String

**Insurance Policy Plan**

* id: Integer
* customer\_id: Integer
* policy\_type: String (e.g., Health Insurance, Life Insurance, Travel Insurance)
* premium\_amount: Double
* start\_date: String
* end\_date: String

**Claims**

* id: Integer
* policy\_id: Integer
* claim\_amount: Double
* date\_of\_claim: String
* status: String

**Payments**

* id: Integer
* policy\_id: Integer
* payment\_amount: Double
* payment\_date: String
* payment\_method: String (e.g., Credit Card, Bank Transfer)
* status: String (e.g., Pending, Processed, Failed)

**Features:**

**Customer Management:**

* **Add Customer:** Create a new customer by providing all required details.
* **Update Customer:** Modify the customer's name and age.
* **Delete Customer:** Customers with existing Insurance Policies cannot be deleted.
* **View All Customers:** Display a list of all customers.

**Insurance Policy Plan Management:**

* **Add Insurance Policy:** Add a new insurance plan.
* **Update Insurance Policy:** Update the policy type, premium amount, and end date.
* **Delete Insurance Policy:** Active policies cannot be deleted.
* **View All Insurance Policies:** Display a list of all policies.

**Claims Management:**

* **Add Claim:** Create a new claim for a policy.
* **Update Claim:** Update claim amount and status (e.g., from 'Pending' to 'Approved' or 'Rejected').
* **Delete Claim:** Approved claims cannot be deleted.
* **View All Claims:** Users can view a list of all claims associated with specific insurance policies in the system.

**Payments Management:**

* **Add Payment:** Record a payment for a policy.
* **Update Payment:** You can only update the payment amount, status, payment method (e.g., changing from 'Credit Card' to 'Bank Transfer').
* **Delete Payment:** You cannot delete a payment that has been confirmed (i.e., if the payment status is marked as 'Processed').
* **View All Payments:** Users can view a list of all payments associated with specific insurance policies in the system.

**UI/UX Requirements:**

* **Use IB and Storyboard:** Build the UI using Interface Builder and Storyboard but ensure there is presence of some **segues**.
* **UITableViewController:** Use UITableViewController to display lists such as customer, policy, claim, or payment lists.
* **Add/Update Views:** Use separate views for adding and updating items. **Do not use UIAlert for this**.
* **Add Button:** Include an "Add" button in the navigation bar to add new items (customers, policies, claims, payments).
* **Date Picker:** Use a UIDatePicker to modify dates, such as policy start and end dates or claim dates.
* **Keyboard Management:** Ensure appropriate keyboards are used (e.g., numeric for IDs, dates, etc.).
* **Data Integrity:** Handle update/delete restrictions and ensure data validity (e.g., can’t delete a customer with active policies).
* **Preloaded Data:** Ensure the app contains valid data for testing during the demo.
* You may reuse relevant code from the previous assignment if applicable.

**Constraints:**

1. **Segues:** Ensure at least some transitions between scenes use segues.
2. **No Crashes:** The app should not crash during use.
3. **Scene Delegate (Optional):** If you're working with iOS 13 or later, remove any unnecessary Scene Delegate configurations as described:
   * Remove the “Application Scene Manifest” entry from Info.plist.
   * Delete any scene delegate classes and methods if present.
   * Add the var window: UIWindow? property to your app delegate if missing.

**Notes:**

* Focus on functionality and proper UI management using **IB** and **Storyboard**.
* Ensure that validation and error handling are implemented throughout the app, with appropriate alerts for missing or invalid fields