Overview of the Project

**INSURANCE MANAGEMENT SYSTEM**

For this project, we developed a system that uses the Repository design pattern to store and manage client and insurance policy data in JSON files. The repository pattern's capacity to establish a distinct separation between the business logic and data access layers improves maintainability and scalability. This method ensures data permanence by using structured files and makes it simple for brokers to handle policy and client data.

For the customer management module, we developed servlets that let brokers do CRUD operations. Brokers have access to detailed customer profiles, the ability to add and remove clients, modify client data, and add new clients. All data processing is done by the repository layer, which ensures that client data is safely saved and accessible from JSON files. This architecture allows for consistent and reliable data handling.

In the policy management module, we enhanced this functionality to manage insurance policies. Brokers have the ability to create new policies, edit existing ones, and assign them to clients. Policies that are out-of-date or no longer in force may also be removed. By using the repository design, which efficiently manages all policy data, brokers may handle customer policies more effectively and with data integrity.

Code Architecture

Model Layer: The Insurance Broker Management System's domain entities are represented by classes like Customer and Policy.

Controller Layer: This responds to client HTTP requests and communicates with the repository layer to retrieve or modify data.

Repository layer is in charge of granting access to the data storage. In this instance, reading and writing customer or policy data involves dealing with JSON files.

# Data Management System Using (JSON) Format

The data in this architecture is stored in JSON format. The JSON files (customers.json and policies.json and customerPolicies.json ) contain serialized data about customers and policies.

Jakson and JSON external jar files are used in class path to manage communication with files from java code.

# Repository Design Pattern

Applications utilize the repository pattern to abstract and encapsulate the functionality needed to access data sources. It offers a means of dividing the business logic for loose coupling from the data access logic.   
Because the Repository Design Pattern offers a transparent abstraction for data access, the system is more adaptable and manageable. In the future, this design might be expanded to incorporate databases or other storage methods without requiring alterations to the business logic.

# Class Diagram

A diagram of a company

Description automatically generated

# Sequence Diagram

A diagram of a company

Description automatically generated

OUTPUT SCREENSHOTS:

LANDING PAGE:

REGISTRATION:

A screenshot of a computer screen

Description automatically generatedA screenshot of a computer screen

Description automatically generatedA screenshot of a computer screen

Description automatically generated

AFTER SUCCESS FULL REGISTRATION:

LOGIN AS BROKER:

A screenshot of a login page

Description automatically generated

AFTER SUCCESSFUL LANDING PAGE :

A screenshot of a computer screen

Description automatically generatedA screen shot of a screen

Description automatically generated

A screen shot of a screen

Description automatically generated

IF LOGIN NOT SUCCESSFUL: INDICATING LOGIN ERROR

A screenshot of a login page

Description automatically generated

AFTER LOGIN:

A screen shot of a screen

Description automatically generated

A screenshot of a computer

Description automatically generatedA screenshot of a website

Description automatically generatedA screenshot of a website

Description automatically generated

CUSTOMER PAGE:

A screen shot of a computer

Description automatically generated

OPTIONS AS HEADERS IN CUSTOMER PAGE:

ADD CUSTOMER:

A screenshot of a computer

Description automatically generated

VIEW CUSTOMER :

A screenshot of a computer

Description automatically generated

UPDATE CUSTOMER:

A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated

DELETE CUSTOMER:

A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated

NOW NAVIGATING TO POLOICIES PAGE:

ADD POLICY:

A screenshot of a video

Description automatically generatedA screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated

VIEW POLICY:

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Update policy:

A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

DELETE POLICY:

A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated

ASSIGN POLICY:

A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated

**Collaboration Details**

In terms of task management and version control with the GITHUB, we as a team worked together. The Customer management module is handled by sai chandu, and reagrding policy management siva kumar has taken care, and mani and vamsi worked on design part of the site known as front-end development in technical, and thread safe management and Sarath chandran worked project setup and uml diagrams and file’s management, login and registration pages as well . With GITHUB each team member has worked on their allocated features on their own as part of achieving goal working parallel work made easier for us to achieve goal in desired timeline.

We have implemented all of functionalities that has been mentioned as part of assignment1 and all the code has been pushed to github.

Thank you.