## CHAPTER 1

**INTRODUCTION**

Insurance Policy data management system is a web based project which is developed for tracking the details of the insurance policy, customer details and company details. This series of web pages is an online insurance analysis and information management system that provides easy access of information regarding the people and resources of insurance. User can view their own personal details when login into the Policy Holder module. This project is useful for any kind of insurance company to manage the insurance details, to sanction the insurance for customer, process the insurance policy details and all kind of insurance process through online. The Insurance management system is a complete solution for organizations, which need to manage insurance for their vehicles, equipment, buildings, and other resources. This insurance management system can efficiently manage the company, records, provides instant access and one that improves the productivity. It will show details about insurance and its types, also it will show the details about different duration schemes to the corresponding insurance type or insurance policy. The main objective of the developed system is to allow admin users to register insured persons with their name, date of birth, residence address, medical history and also policy details.

## ABOUT SQL

* + - Structured Query Language (SQL) is comprehensive database language. Hence it has both DDL and DML.
    - Data Definition Language (DDL): We can use CREATE, INSERT, DELETE MODIFY statements. We cannot manipulate the data in the table.
    - Data Manipulation Language (DML): We can manipulate the data in the record using UPDATE and ALTER statements.
    - SQL has several different techniques for writing programs in various Programming languages that include SQL statements to access one or more database.
    - SQL has transaction control commands. These are used to specify units of database processing for concurrency control and recovery purpose.

MySQL is an [relational database management system](https://en.wikipedia.org/wiki/Relational_database_management_system)(RDBMS).Its name is a combination of "My", the name of co-founder Wideness’s daughter and "[SQL](https://en.wikipedia.org/wiki/SQL)", the abbreviation

for [Structured Query Language.](https://en.wikipedia.org/wiki/Structured_Query_Language) The MySQL development project has made its [sourcecode](https://en.wikipedia.org/wiki/Source_code) available under the terms of the [GNU General Public License,](https://en.wikipedia.org/wiki/GNU_General_Public_License) as well as under a variety of [proprietary](https://en.wikipedia.org/wiki/Proprietary_software) agreements. MySQL was owned and sponsored by a single [for-profit](https://en.wikipedia.org/wiki/Business) firm, the [Swedish](https://en.wikipedia.org/wiki/Sweden) company [MySQL AB](https://en.wikipedia.org/wiki/MySQL_AB), now owned by [Oracle Corporation.](https://en.wikipedia.org/wiki/Oracle_Corporation) For proprietary use, several paid editions are available, and offer additional functionality.

MySQL was created by a Swedish company, [MySQL AB,](https://en.wikipedia.org/wiki/MySQL_AB) founded by [David Axmark](https://en.wikipedia.org/wiki/David_Axmark), Allan Larsson and [Michael "Monty" Widenius](https://en.wikipedia.org/wiki/Michael_(Monty)_Widenius). Original development of MySQL by Widenius and Axmark began in 1994. The first version of MySQL appeared on 23 May 1995. It was initially created for personal usage from [mSQL](https://en.wikipedia.org/wiki/MSQL) based on the low-level language [ISAM](https://en.wikipedia.org/wiki/ISAM), which the creators considered too slow and inflexible. They created a new [SQL](https://en.wikipedia.org/wiki/Structured_Query_Language) interface, while keeping the same [API](https://en.wikipedia.org/wiki/Application_programming_interface) as mSQL

MySQL is written in [C](https://en.wikipedia.org/wiki/C_(programming_language)) and [C++.](https://en.wikipedia.org/wiki/C%2B%2B) Its SQL parser is written in [yacc,](https://en.wikipedia.org/wiki/Yacc) but it uses a home- brewed [lexical analyzer.](https://en.wikipedia.org/wiki/Lexical_analysis)

Mysql dump is a logical backup tool included with both community and enterprise editions of MySQL. It supports backing up from all storage engines. MySQL Enterprise Backup is a hot backup utility included as part of the MySQL Enterprise subscription from Oracle, offering native InnoDB hot backup, as well as backup for other storage engines.

Xtra Backup is an open-source MySQL hot backup software program. Features include hot, non-locking backups for InnoDB storage, incremental backups, streaming, parallel- compressed backups, throttling based on the number of I/O operations per second, etc.

MySQL Fabric is an integrated system for managing a collection of MySQL servers, and a [framework](https://en.wikipedia.org/wiki/Software_framework) on top of which high availability and database sharding is built. MySQL Fabric is open-source, and supports procedure execution in the presence of failure, providing an execution model usually called resilient execution*.*

[phpMyAdmin](https://en.wikipedia.org/wiki/PhpMyAdmin) is a free and open source tool written in PHP intended to handle the administration of MySQL with the use of a web browser. It can perform various tasks such as creating, modifying or deleting databases, tables, fields or rows; executing SQL statements; or managing users and permissions. The software, which is available in 78 languages, is maintained by ThephpMyAdmin Project.

## SCOPE AND APPLICATIONS OF MINI PROJECT

This Database plays an important role in Information Management of Students, Teachers, Faculties, IA marks of the Students that can greatly improve the accuracy, security and efficiency of the Information Management of our college. Mini projects can be developed using JAVA, PYTHON, PHP, HTML, CSS etc., but mainly we are discussing here about php.

The application of our project ‘Insurance Policy Data Management’ is like any other conventional management system i.e. we can store the details for the employees working in the company, clients of the company and also check the details of the policies registered. The user can also view a detailed policy data view. Our project can be implemented in daily life since mail is commonly used.

## CHAPTER 2

**LITERATURE SURVE**

Before this application all activities done manually, then all the activities take more time and also take more man power. Commission interests, dues calculate manually, based on this manual problem sometimes calculate the wrong All related information passed on this another branch through courier. Sometime miss the important documents. So these problem problems overcome when we will develop the system. This system is very helpful automation of entire Insurance system and also reduces the time and manpower.

## RELATED WORK

* **Lack of immediate retrievals**: In the conventional system, information is distributed across several files. This might also lead to data redundancy with repetition of the same information in various files. In the event of a complex or nested query, the search has to scan several files, thus making procurement of requested query results very cumbersome.
* **Maintenance of Accuracy and Reliability issues:** With redundancy comes consistency issues as the update of information in a single record should be echoed in all records containing the same information. Also, atomicity issues ie, completion of a transaction in totality or nothing at all; has to be maintained. This is difficult in a multi-file system.
* **Lack of prompt update:** Updates associated with a record in a file is to be reflected in all records wherein the particular record is present. This concurrent update poses the problem of time lag. Errors in commit operation to some particular files cause the grave issue of data inconsistency.
* **Error prone manual calculation:** Manual calculations are error prone and relatively immensely time consuming, in spite of which they may result in generation of incorrect information.
* Verification is another overhead, which can be saved through efficient design and implementation.
  + **Improved manual System**: One of the alternative solutions is the improvement of the manual system. Anything, which can be done by using automated methods, can be done manually. But the key question is how to perform a task manually in a sound and optimal manner. Following are some suggestions, which can be useful in manual system.

A more sophisticated register maintenance dedicated to each subsystem can be maintained with centralized control and evolution. Adequate dedicated staff may be maintained so that updates are made at very moment at the same time. Proper provision for proper work should be put into place.

This would require considerable extra work force.

**Batch system:** An alternative solution can be used of computer-based batch system for maintaining the information regarding personal details. A batch system refers to a system in which data is processed in a periodical basis. The batch system is able to achieve most of the goals and sub goals. But a batch system data is processed in sequential basis. Therefore, batch system is not suggested.

**Online system:** This system provides online storage, update and retrieval facility. This system promises very less or no paper work and also provides help to administrate and company personal. In the system everything is stored electronically so constants of paper work are eliminated and multiple files usage is not required. Information can be retrieved very easily without scanning multiple registers this system is been discussed here.

## PROBLEMS DESCRIPTION:

This conventional approach of recruiting people increases the work of maintaining candidates on paper work which requires a lot of manpower. Other than this, it increases the complexity and also contains less efficiency. The automated system needs to be found which replaces the current conventional system or at least decreases this whole process' complexity. The problem tackled in the project is to handle the policy data using database management system. This project would focus on both front-end as well as backend for systematic working.

Data input would be given from the front-end by users. The front-end would be a HTML form.

* + - Relation between client and his policies is a one to many relationship, but policy type to clients is a many to many relationship.
    - Data would be handled at the back-end using different tables and relations using MySQL.
    - A policy taken by a client has attributes like premium, sum assured, date of commencement, etc.
    - A client has attributes including personal details as well as details about the policy he/she has taken.
    - A policy type contains attributes describing the type of policies like premium based on the mode, risk cover.
    - There would be many other tables where records of policies taken by different clients would be present depending on its status like active, lapsed, etc. The developed system should allow admin users to register insured persons with their name, date of birth, residence address, medical history and also policy details. After registering all the insured persons, website should provide management facilities like delete unwanted persons’ data. And also should provide awareness to the visitors about micro insurance through articles.

## ADVANTAGES AND DISADVANTAGES OF PROPOSED SYSTEM

**ADVANTAGES:**

1. Very simple and easy to implement
2. Security of data
3. Ensure data accuracy
4. Reduces the damages of machine
5. Minimizes manual data entry
6. Greater efficiency
7. User friendly and interactive
8. Less time consuming

## DISADVANTAGES:

* + - Client cannot modify their data
    - Negotiations over the plan rates is not possible

## CHAPTER 3

**HARDWARE , SOFTWARE REQUIREMENTS AND FUNCTIONAL REQUIREMENTS**

**HARDWARE REQUIREMENTS:**

Processor : Intel Pentium V

Clock speed : 500 MHZ

System bus : 64bits

RAM : 4 GB of RAM

HDD : 40 GB or higher

Monitor : SVGA COLOR

Keyboard : 108 keys

Mouse : 2 button mouse

## SOFTWARE REQUIREMENTS:

Front End : HTML,PHP

Back End : My SQL

Tool Used : Xampp Operation System : Windows family

## FUNCTIONAL REQUIREMENTS

* An agent can login on to the website.
* An agent can add/delete/update the clients.
* An agent can add/delete/update policy of the company.
* An agent can add/delete/update the payment for the policy.
* An agent can add/delete/update the nominee details of the client.
* Client can view his policy, nominee, payments information.

Dept. Of CSE 8 2021-22

## CHAPTER 4

**ANALYSIS AND DESIGN**

System Design is the most creative and challenging phase in the system life cycle. Design is the first step into the development phase for any engineered product or system. Design is a creative process. A good design is the key to effective system. System design is a solution how to approach the creation of a new system. System design transforms a logic representation of what is required to do into the physical specification. The specification is converted into physical reality during development.

## ARCHITECTURE OF PROPOSED SYSTEMS

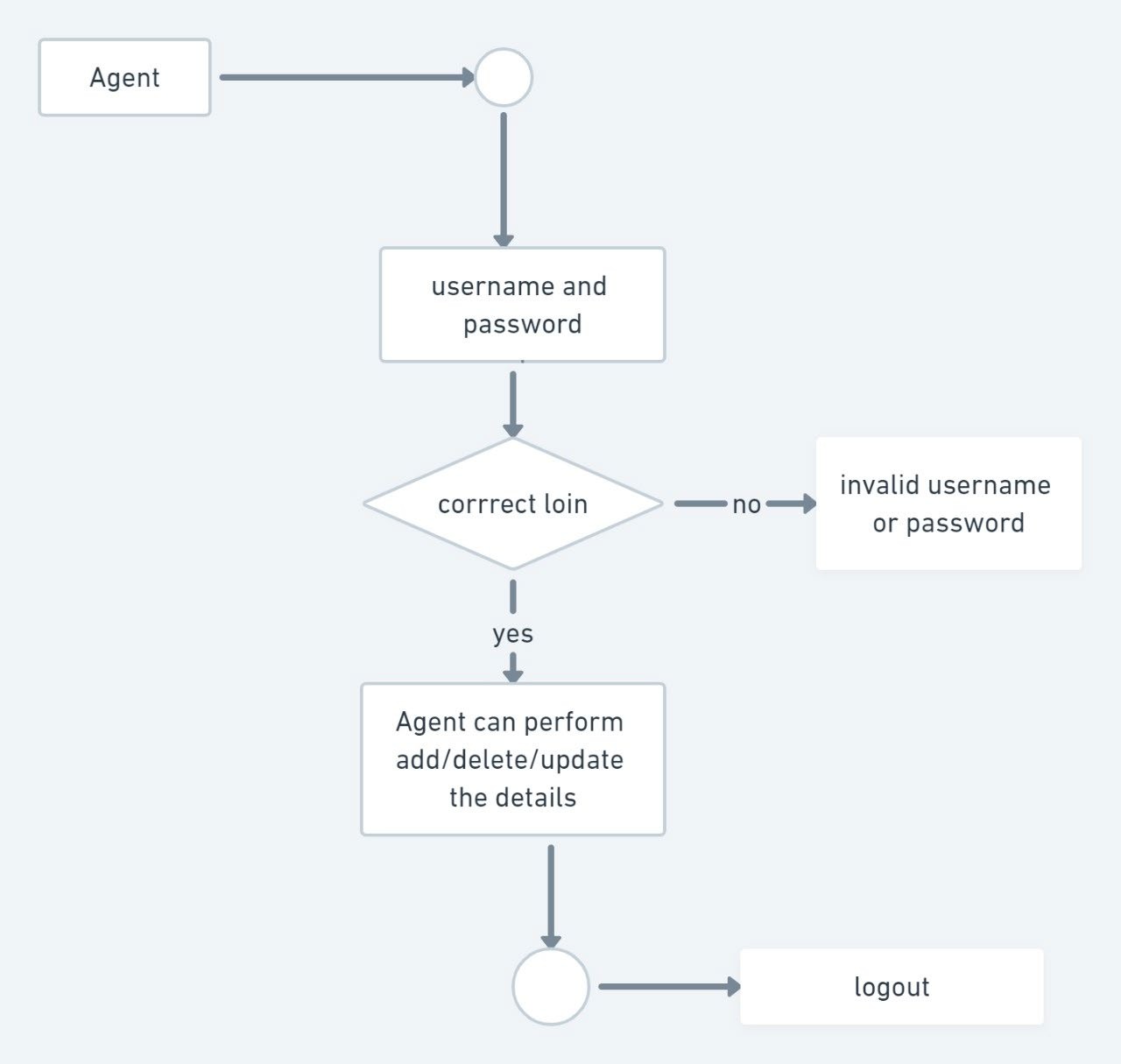


Figure 4.1 : Architecture of Proposed System

## LOGICAL DESIGN

The logical flow of a system and define the boundaries of a system. It includes the following steps:

* + - Reviews the current physical system – its data flows, file content, volume’s, frequencies etc.
    - Prepares output specifications – that is, determines the format, content and Frequency of reports.
    - Prepares input specifications – format, content and most of the input functions.
    - Prepares edit, security and control specifications.
    - Specifies the implementation plan.
    - Prepares a logical design walk through of the information flow, output, input, controls and implementation plan.
    - Reviews benefits, costs, target dates and system constraints.

## PHYSICAL DESIGN

Physical system produces the working systems by define the design specifications that tell the programmers exactly what the candidate system must do. It includes the following steps.

* + - Design the physical system.
    - Specify input and output media.
    - Design the database and specify backup procedures.
    - Design physical information flow through the system and a physical design walk through.
    - Plan system implementation.
    - Prepare a conversion schedule and target date.
    - Determine training procedures, courses and timetable.
    - Devise a test and implementation plan and specify any new hardware/software.

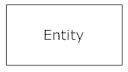
## ENTITY-RELATIONSHIP DIAGRAM AND SCHEMA DIAGRAM

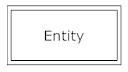
An entity relationship diagram (ERD) shows the relationships of entity sets stored in a database. An entity in this context is an object, a component of data. An entity set is a collection of similar entities. These entities can have attributes that define its properties.

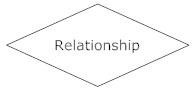
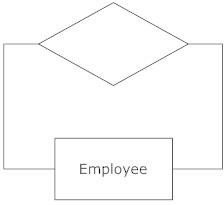
There are two reasons to create a database diagram. You're either designing a new schema or you need to document your existing structure.

If you have an existing database you need to document, you create a database diagram using data directly from your database. You can export your database structure as a CSV file ([there](https://www.smartdraw.com/developers/extensions/erd-scripts.htm) [are some scripts on how to this here](https://www.smartdraw.com/developers/extensions/erd-scripts.htm)), then have a program generate the ERD automatically.

An ER diagram is a means of visualizing how the information a system produces is related. There are five main components of an ERD:

* **Entities**, which are represented by rectangles. An entity is an object or concept about which you want to store information.
* A **weak entity** is an entity that must defined by a foreign key relationship with another entity as it cannot be uniquely identified by its own attributes alone.



* **Actions**, which are represented by diamond shapes, show how two entities share information in the database.
* In some cases, entities can be self-linked. For example, employees can supervise other employees.
* **Attributes**, which are represented by ovals. A key attribute is the unique, distinguishing characteristic of the entity. For example, an employee's social security number might be the employee's key attribute.
* A multivalued attribute can have more than one value. For example, an employee entity can have multiple skill values.



* A derived attribute is based on another attribute. For example, an employee's monthly salary is based on the employee's annual salary.
* **Connecting lines**, solid lines that connect attributes to show the relationships ofentities in the diagram.
* **Cardinality** specifies how many instances of an entity relate to one instance of another entity. Ordinality is also closely linked to cardinality.

## ENTITY RELATIONSHIP DIAGRAM

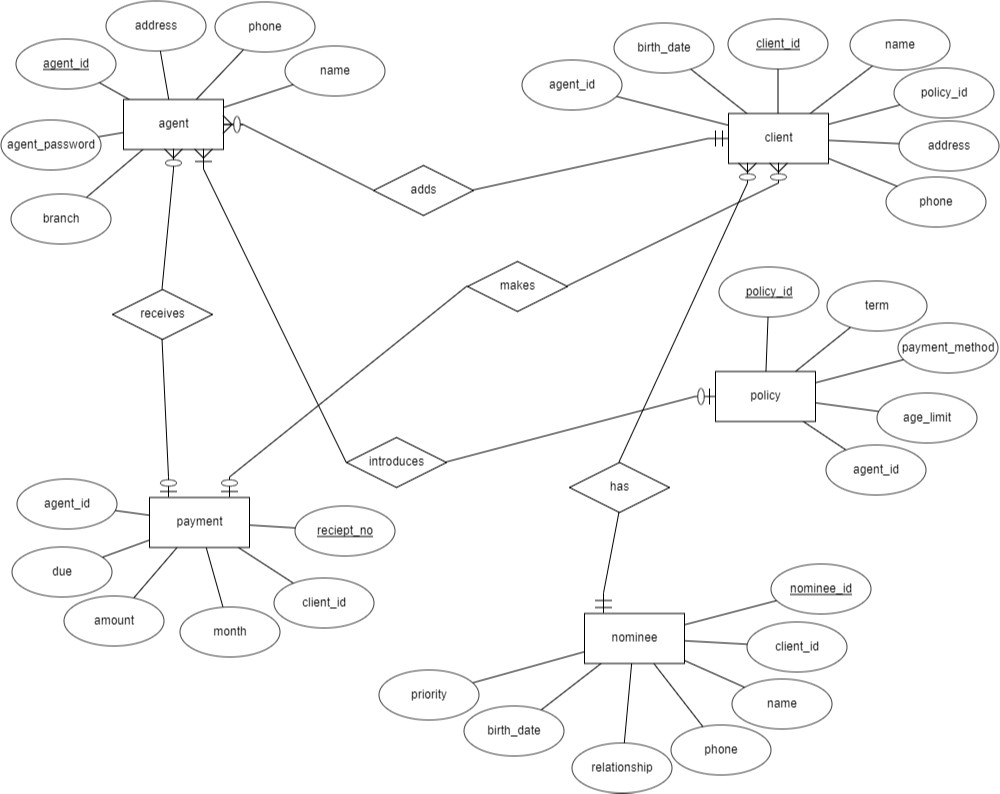


Figure 2.4: E-R Diagram of Online Insurance Management System

## SCHEMA DIAGRAM

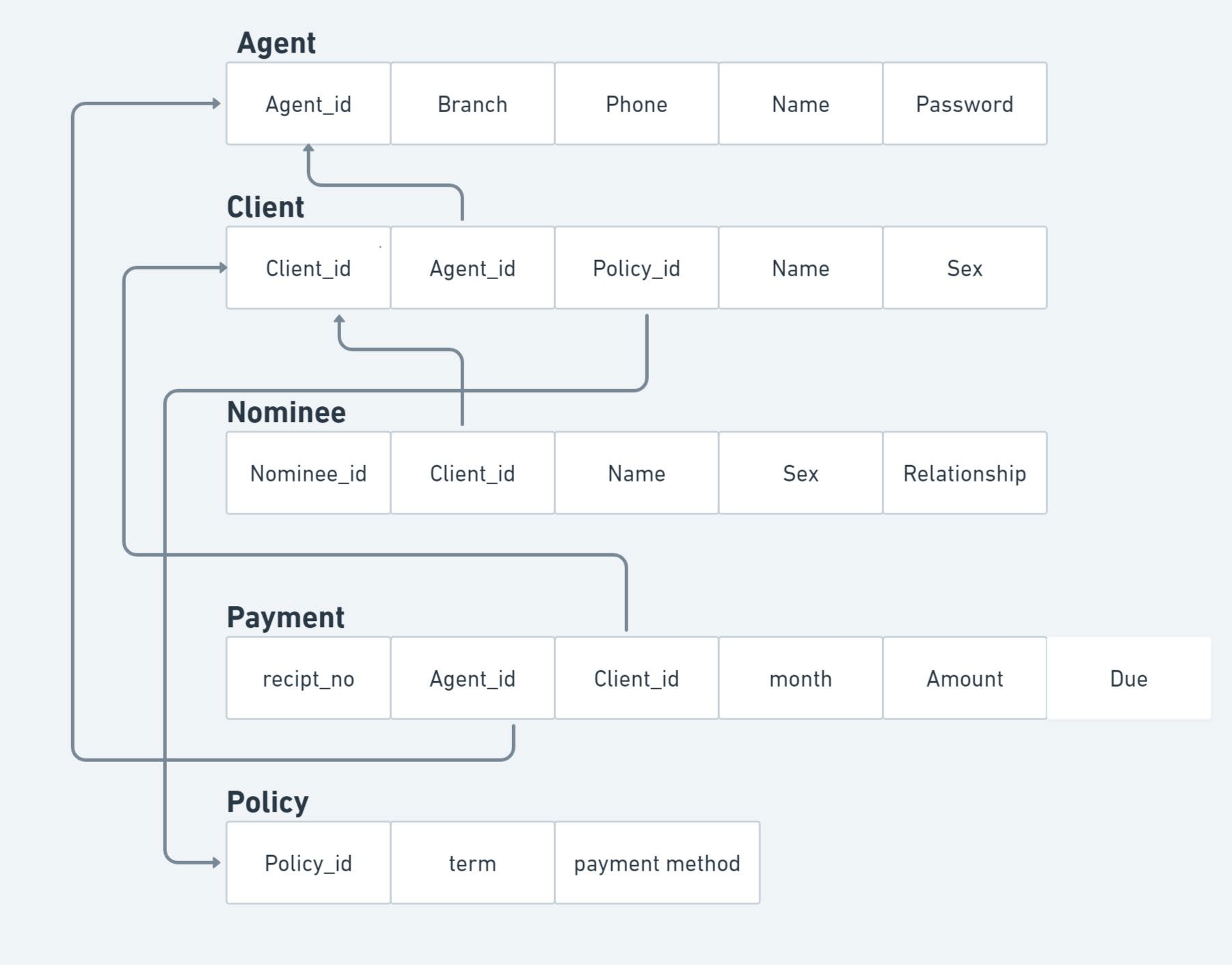


Figure 4.3: Schema diagram of Online Insurance Management System

## CHAPTER 5

**IMPLEMENTATION**

This phase is initiated after the system has been tested and accepted by the user. In this phase, the system is installed to support the intended business functions. System performance is compared to performance objectives established during the planning phase. Implementation includes user notification, user training, installation of hardware, installation of software onto production computers, and integration of the system into daily work processes.

This phase continues until the system is operating in production in accordance with the defined user requirements.

## MODULES DIVISION

This project is modularized as the following:

1. Agent entity.
2. Client entity.
3. Nominee entity.
4. Payment entity.
5. policy entity.

#### Agent

* + - The person who provide the information, create and update the clients data.
    - He also introduces new policies.
    - After login, admin can view/add/update/delete policy categories
    - Can add/update/delete policy. Can view/add/update/delete policy.

#### Client

* + - Client can view his/her policy details, payment details, nominee information, amount due date by entering the client id

#### Nominee

* + - Nomination is the process of determining one person or persons who will receive the benefits from the insurance policy in case of any casualties.
    - The nominee in the insurance can be anyone, per the policyholder’s choice.
    - In general, the nominee in the insurance is a close relative.

#### Payment

* + - Once earned, the premium is income for the insurance company.
    - It also represents a liability, as the insurer must provide coverage for claims being made against the policy.

#### Policy

* + - The insurance policy is a standard form contract between the agent and client, which determines claims which the agent is legally required to pay.
    - In exchange for an initial payment known as premium, the agent promises to pay for the loss caused by perils covered under the policy language.

## SQL CODE:

-- phpMyAdmin SQL Dump

-- version 4.2.11

-- [http://www.phpmyadmin.net](http://www.phpmyadmin.net/)

--

-- Host: 127.0.0.1

-- Generation Time: Nov 25, 2020 at 02:17 PM

-- Server version: 5.6.21

-- PHP Version: 5.5.19

SET SQL\_MODE = "NO\_AUTO\_VALUE\_ON\_ZERO";

SET time\_zone = "+00:00";

/\*!40101 SET @OLD\_CHARACTER\_SET\_CLIENT=@@CHARACTER\_SET\_CLIENT \*/;

/\*!40101 SET @OLD\_CHARACTER\_SET\_RESULTS=@@CHARACTER\_SET\_RESULTS \*/;

/\*!40101 SET @OLD\_COLLATION\_CONNECTION=@@COLLATION\_CONNECTION \*/;

/\*!40101 SET NAMES utf8 \*/;

--

-- Database: `lims`

--

--

-- Table structure for table `agent`

--

CREATE TABLE IF NOT EXISTS `agent` (

`agent\_id` varchar(50) NOT NULL,

`agent\_password` varchar(30) NOT NULL,

`name` varchar(50) NOT NULL,

`branch` varchar(50) NOT NULL,

`phone` varchar(50) NOT NULL) ENGINE=InnoDB DEFAULT CHARSET=latin1;

--

-- Table structure for table `nominee`

--

CREATE TABLE IF NOT EXISTS `nominee` (

`nominee\_id` varchar(200) NOT NULL,

`client\_id` varchar(100) NOT NULL,

`name` varchar(50) NOT NULL,

`sex` varchar(20) NOT NULL,

`birth\_date` varchar(50) NOT NULL,

`nid` varchar(50) NOT NULL,

`relationship` varchar(20) NOT NULL,

`priority` varchar(20) NOT NULL,

`phone` varchar(50) NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=latin1;

--

-- Table structure for table `payment`

--

CREATE TABLE IF NOT EXISTS `payment` (

`recipt\_no` varchar(20) NOT NULL,

`client\_id` varchar(50) NOT NULL,

`month` varchar(50) NOT NULL,

`amount` varchar(50) NOT NULL,

`due` varchar(50) NOT NULL,

`fine` varchar(50) NOT NULL,

`agent\_id` varchar(30) NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=latin1;

--

-- Table structure for table `policy`

--

CREATE TABLE IF NOT EXISTS `policy` (

`policy\_id` varchar(50) NOT NULL,

`term` varchar(50) NOT NULL,

`health\_status` varchar(50) NOT NULL,

`system` varchar(50) NOT NULL,

`payment\_method` varchar(50) NOT NULL,

`coverage` varchar(50) NOT NULL,

`age\_limit` varchar(20) NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=latin1;

--

-- Indexes for table `agent`

--

ALTER TABLE `agent`

ADD PRIMARY KEY (`agent\_id`), ADD UNIQUE KEY `agent\_id` (`agent\_id`);

--

-- Indexes for table `client`

--

ALTER TABLE `client`

ADD PRIMARY KEY (`client\_id`), ADD UNIQUE KEY `client\_id` (`client\_id`);

--

-- Indexes for table `nominee`

--

ALTER TABLE `nominee`

ADD PRIMARY KEY (`nominee\_id`);

--

-- Indexes for table `payment`

--

ALTER TABLE `payment`

ADD PRIMARY KEY (`recipt\_no`), ADD UNIQUE KEY `recipt\_no` (`recipt\_no`);

--

-- Indexes for table `policy`

--

ALTER TABLE `policy`

ADD PRIMARY KEY (`policy\_id`), ADD UNIQUE KEY `policy\_id` (`policy\_id`);

/\*!40101 SET CHARACTER\_SET\_CLIENT=@OLD\_CHARACTER\_SET\_CLIENT \*/;

/\*!40101 SET CHARACTER\_SET\_RESULTS=@OLD\_CHARACTER\_SET\_RESULTS \*/;

/\*!40101 SET COLLATION\_CONNECTION=@OLD\_COLLATION\_CONNECTION \*/;

## CHAPTER 6

#### BACKEND:

**SNAPSHOTS**

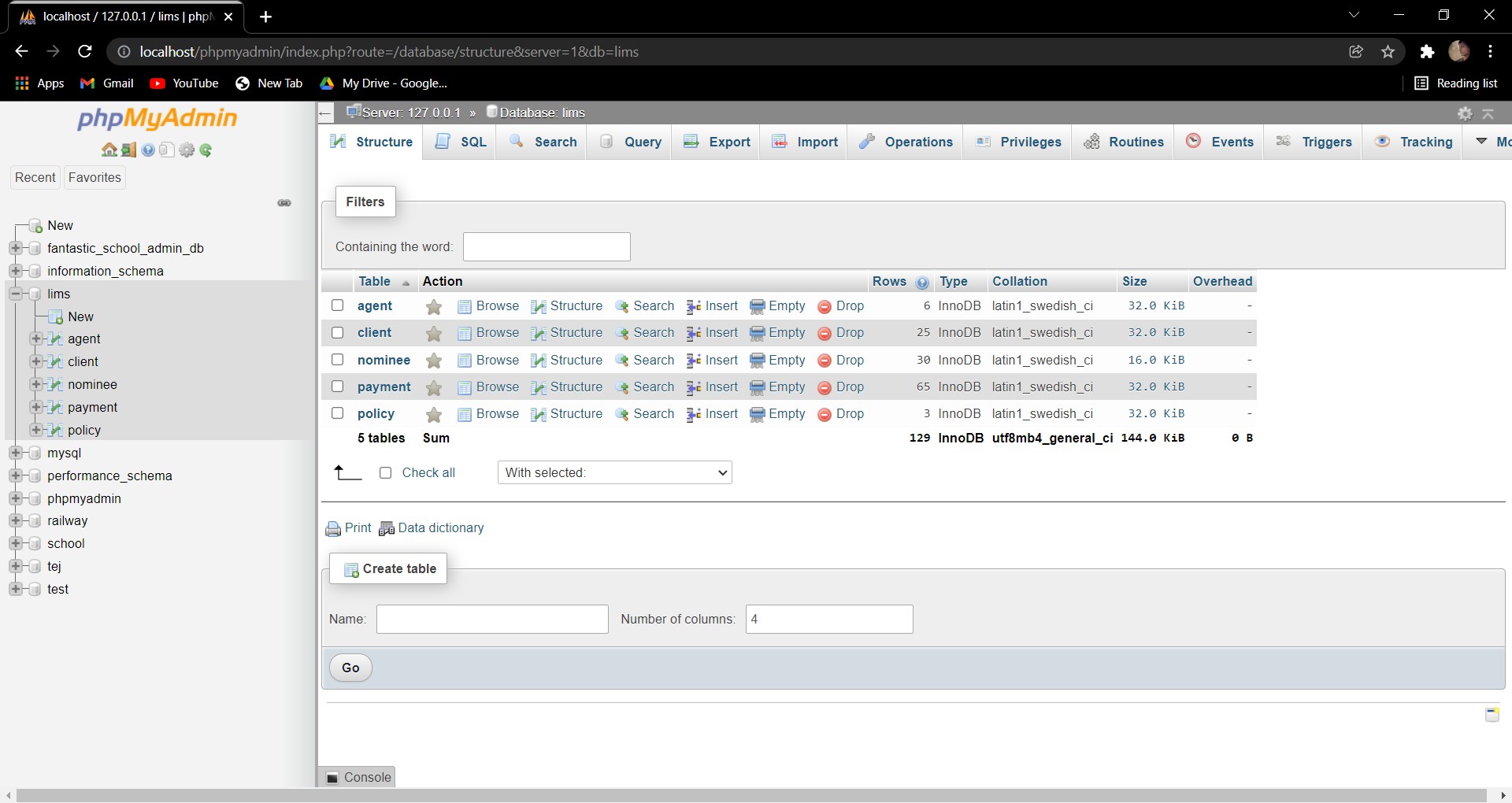


Figure 6.1: Backend

#### LANDING PAGE :



FIGURE 6.2: Landing page

**HOME PAGE :** This page consists of links reaching all the other pages for registration and showing the data stored of agents, customers, their policies nominee and payments.

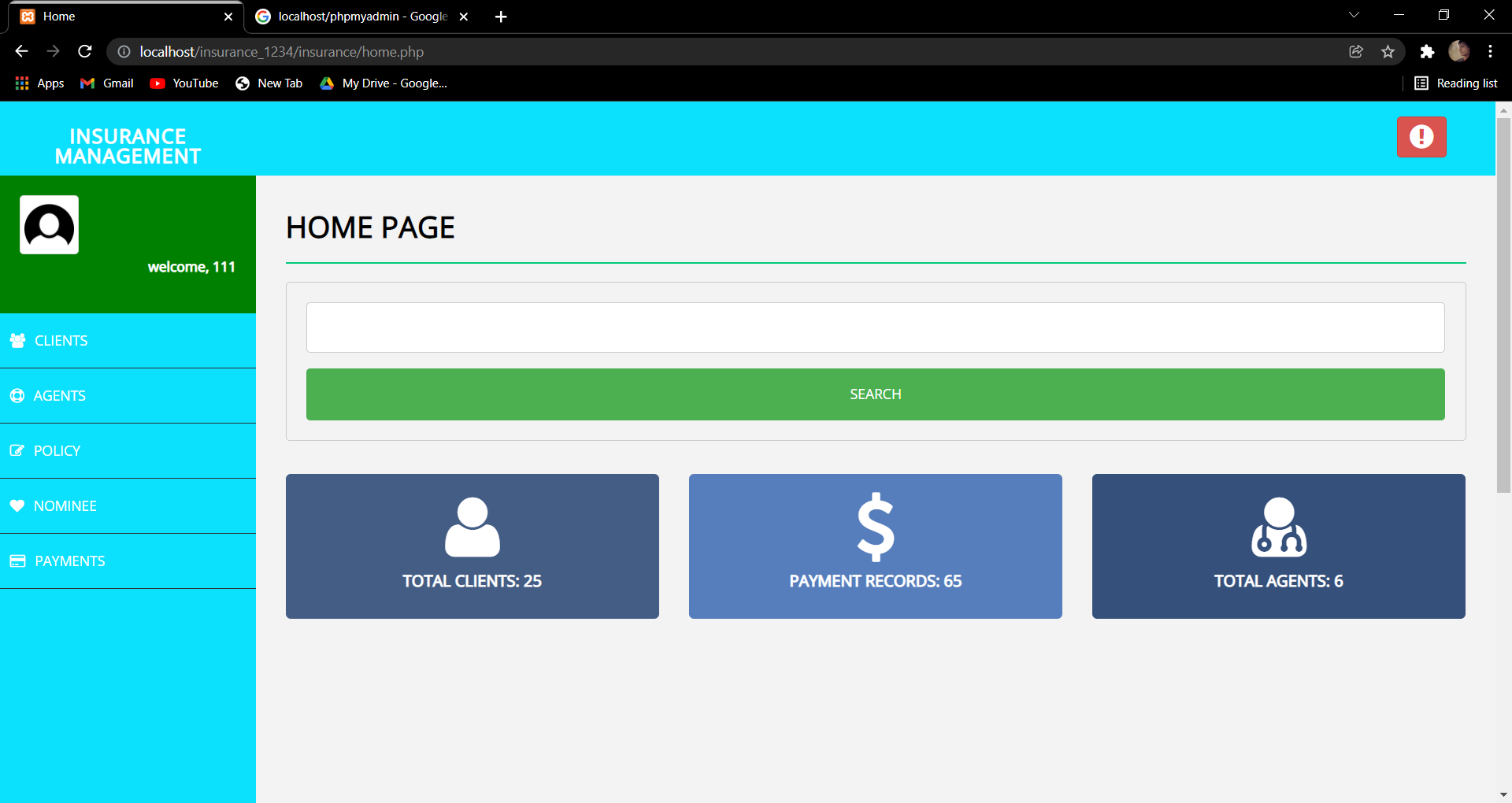


Figure 6.3: Home Page

**LOGIN PAGE:** This is the login page through which Administrator(Agent) is going to access the database.

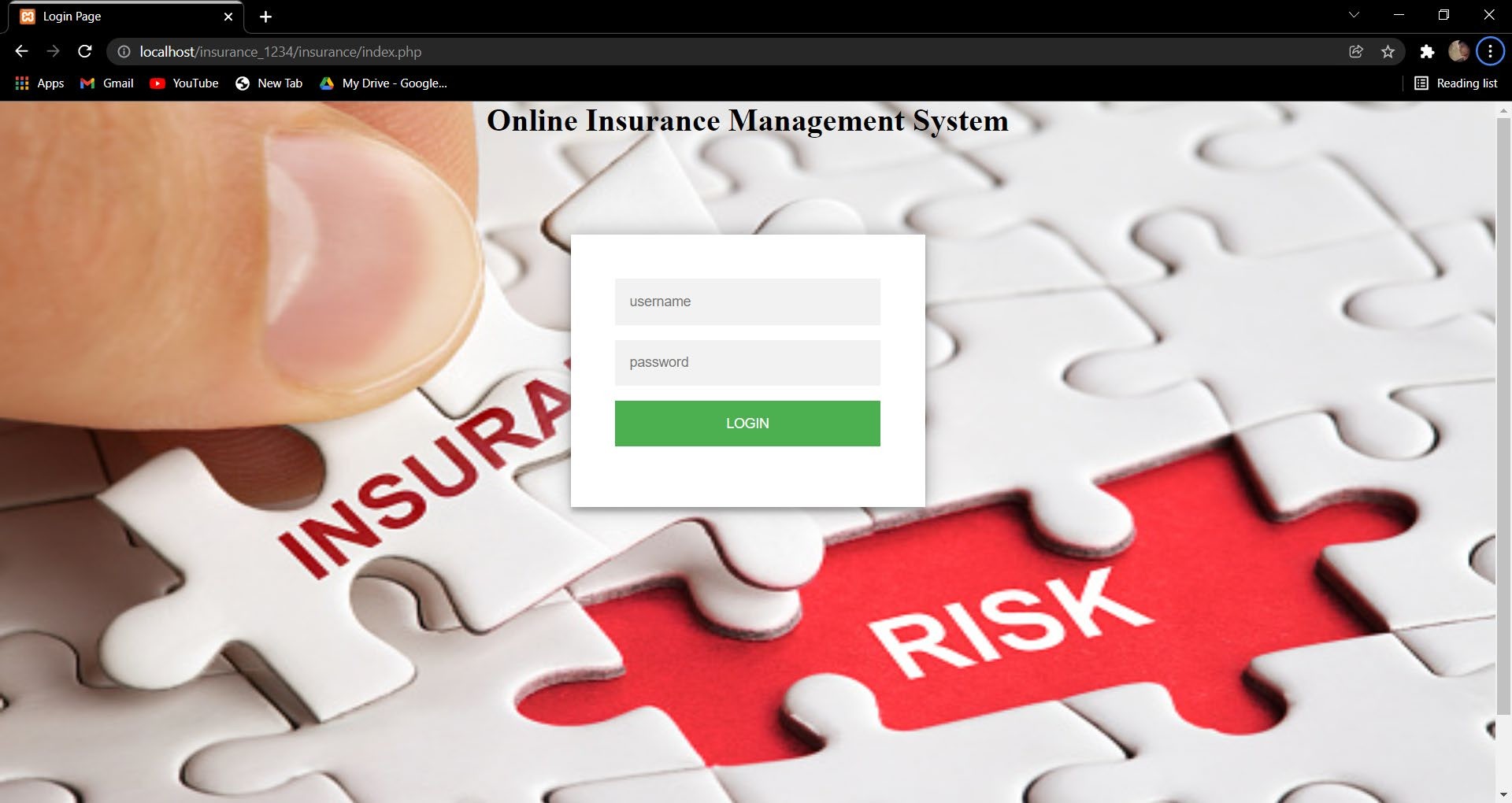


Figure 6.4 : Login page

**ADD CLIENT :** In this page agent can add client details.

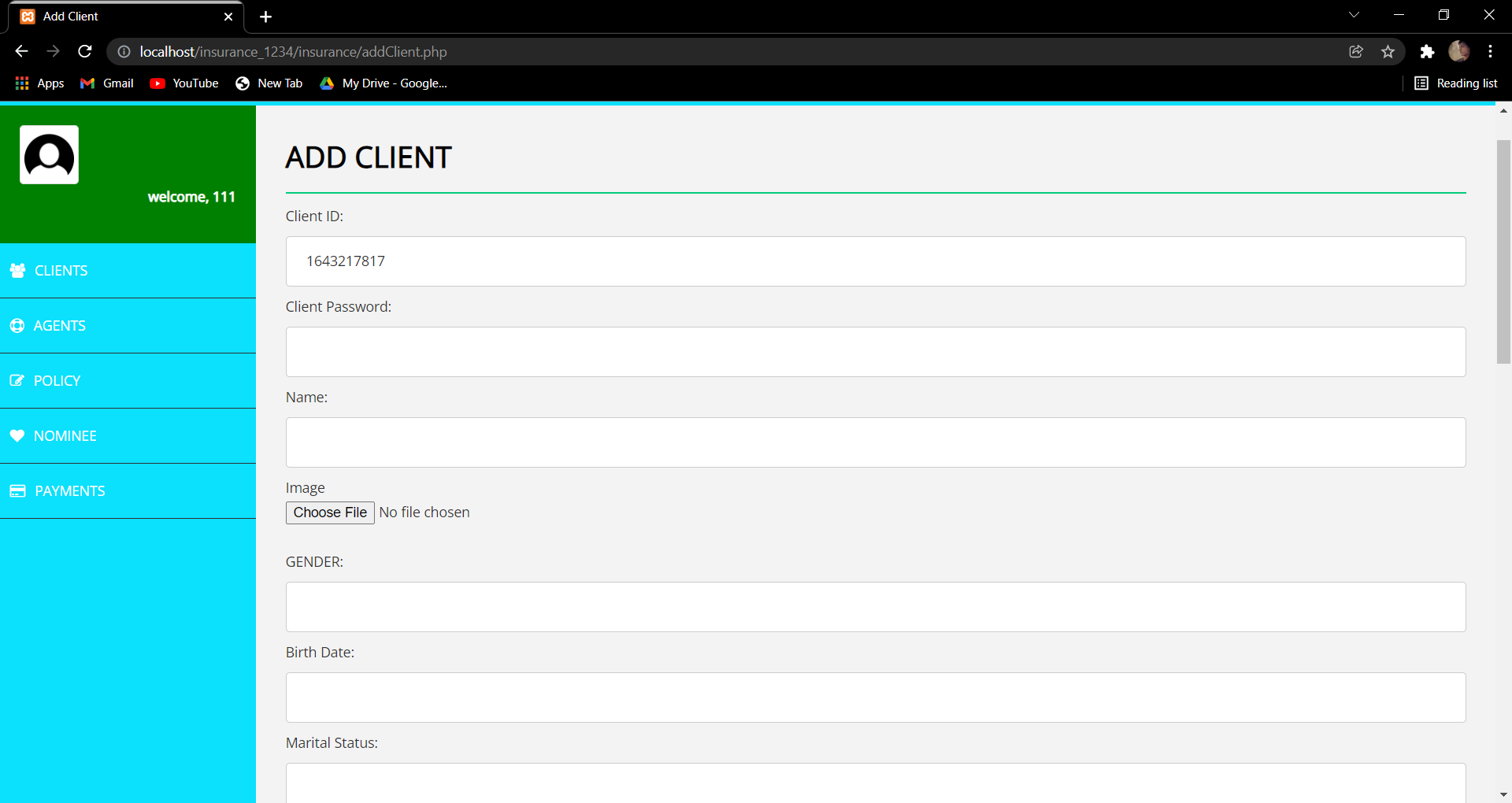


Figure 6.5 : Add client

**AGENTS :** Provide the information about the available agents.

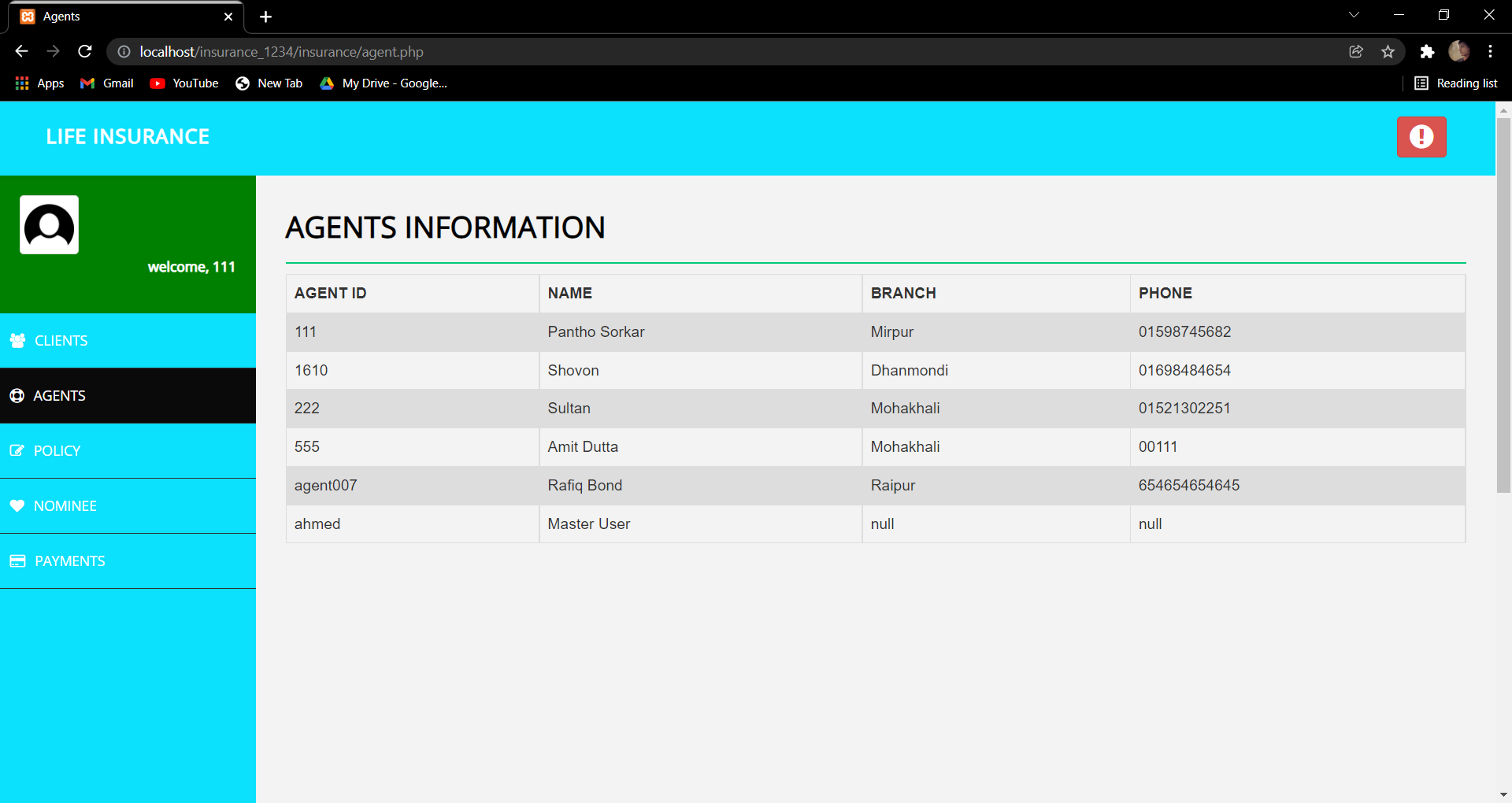


Figure 6.6: Agent information

**POLICY:** This page provides the information about the available Policies.

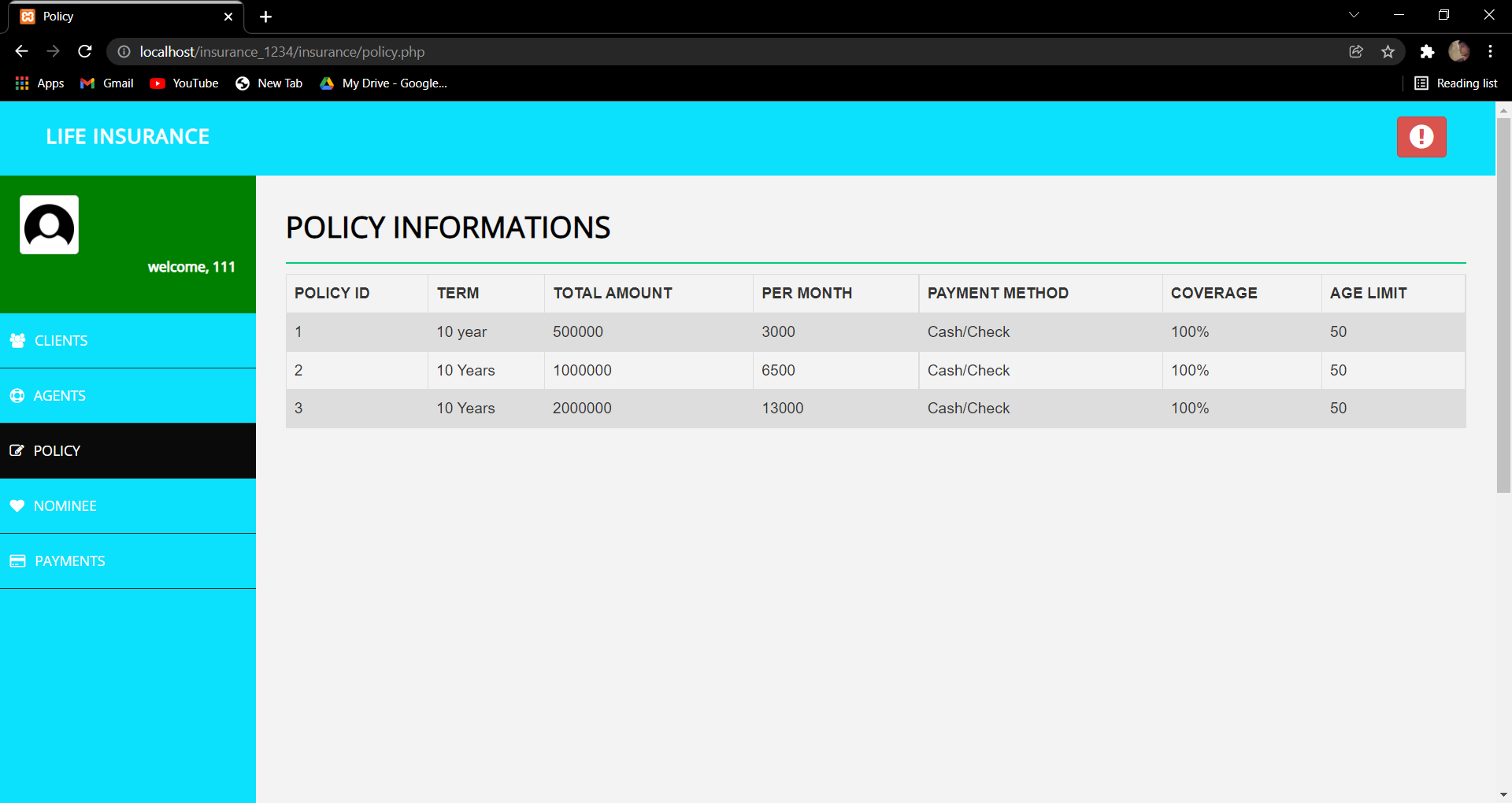


Figure 6.7: Policy information

**NOMINEE:** This page provides the information about the available Nominee.

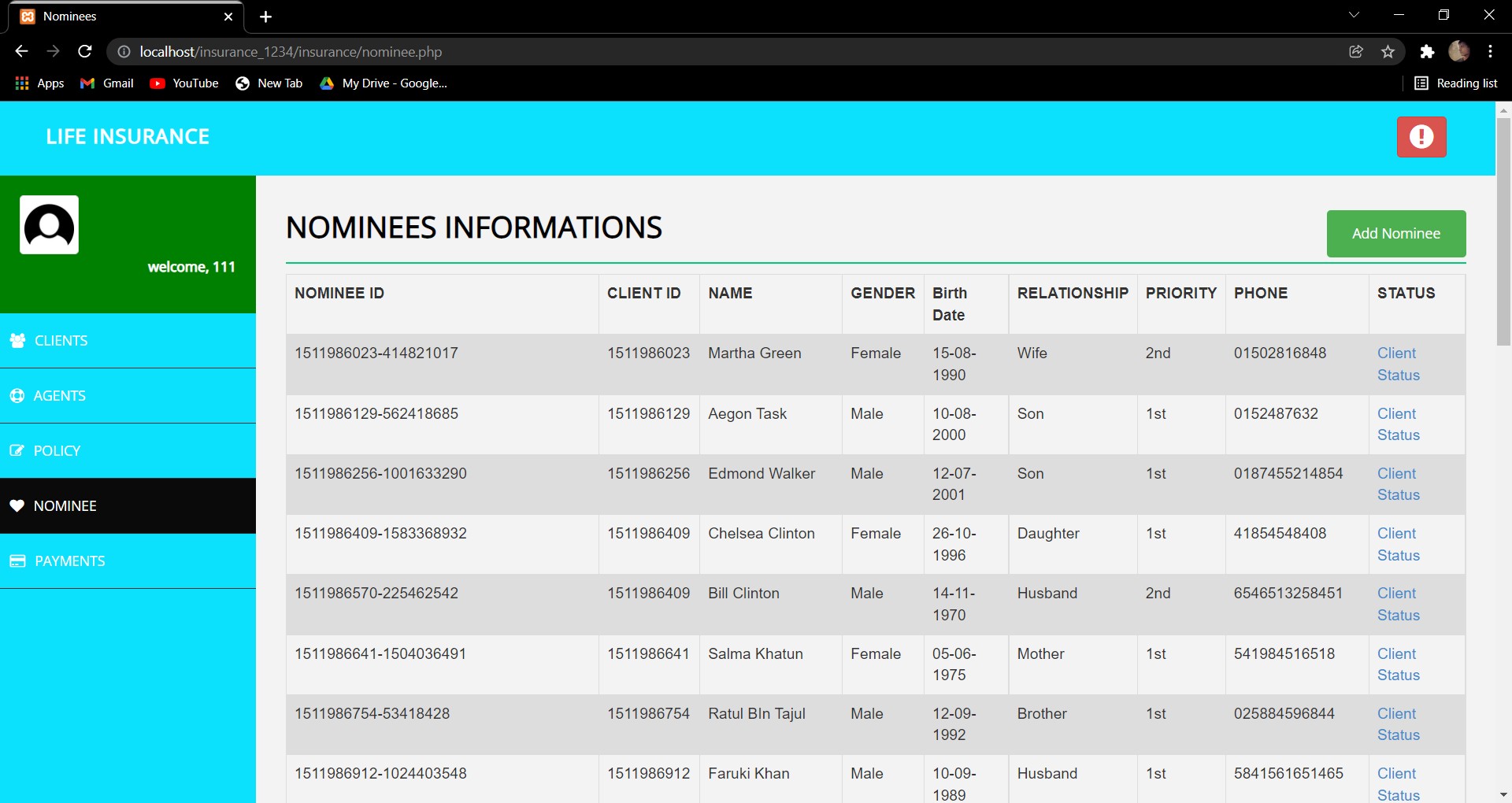


Figure 6.8 : Nominee information

**ADD NOMINEE:** This page allows to add nominee.

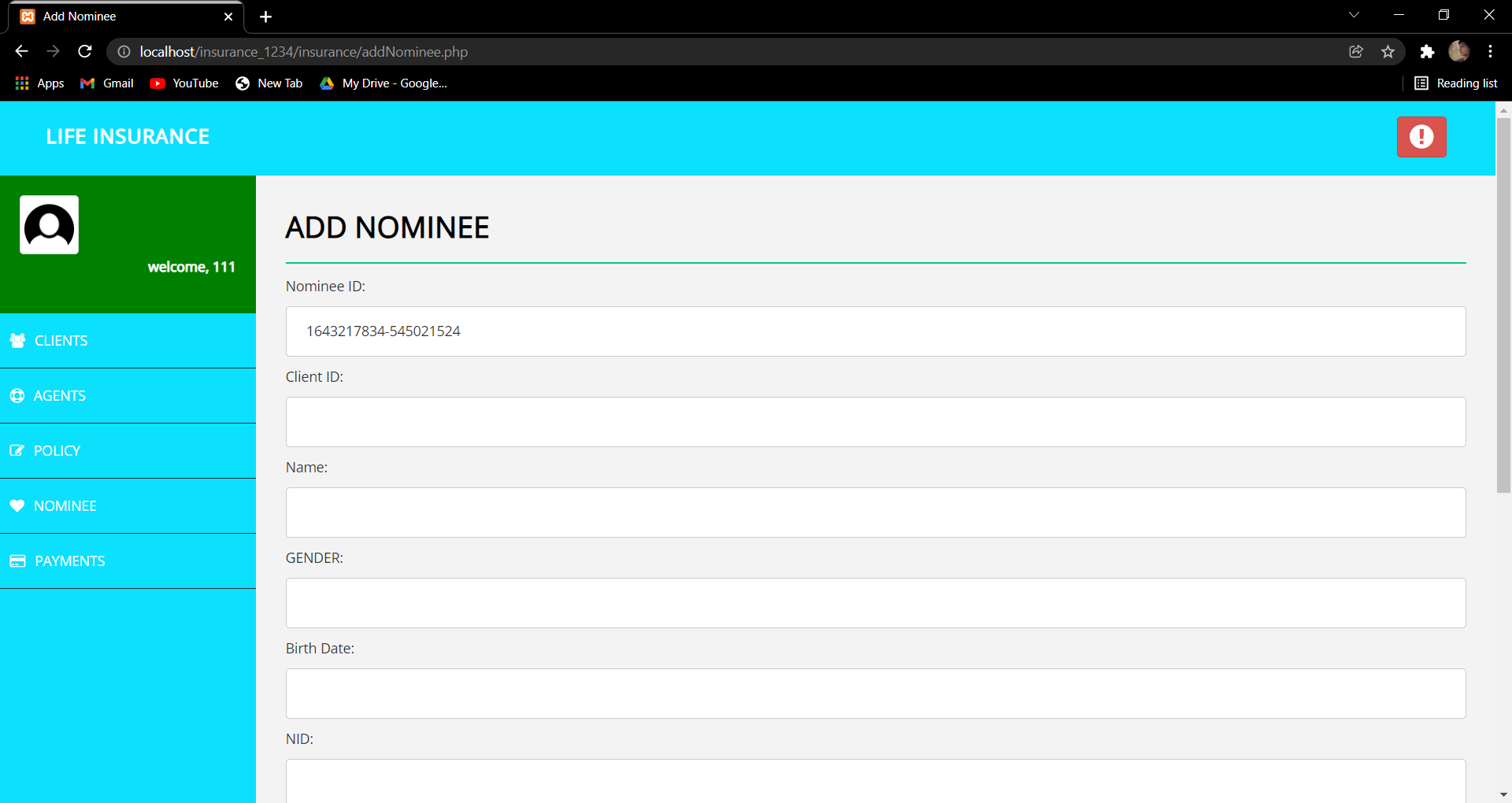


Figure 6.9 : Add nominee

**PAYMENT :** This page provide information about the payment and transaction of the client.

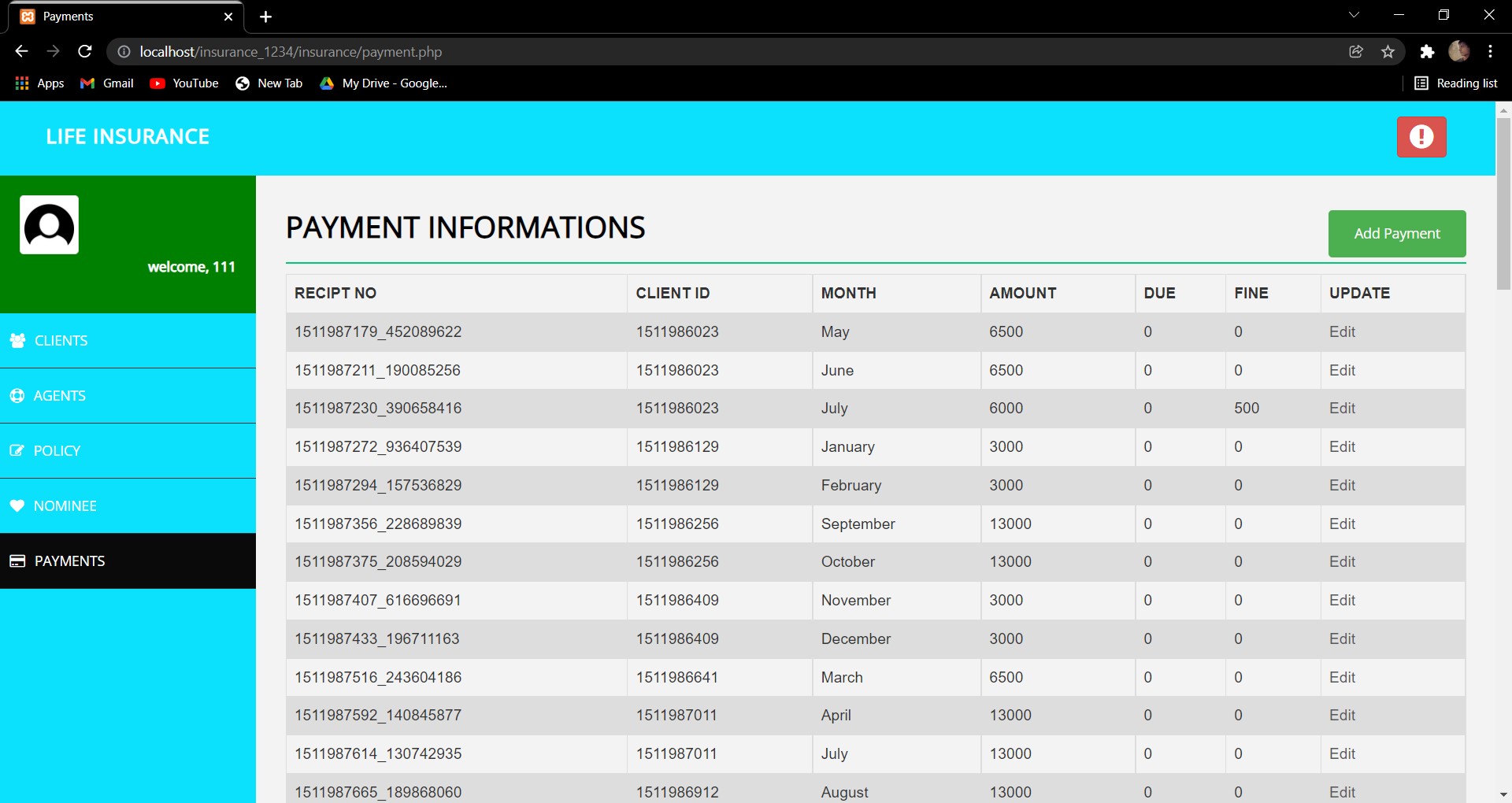


Figure 6.10 : Payment information

**ADD PAYMENT :** This page allows to add the payment details.

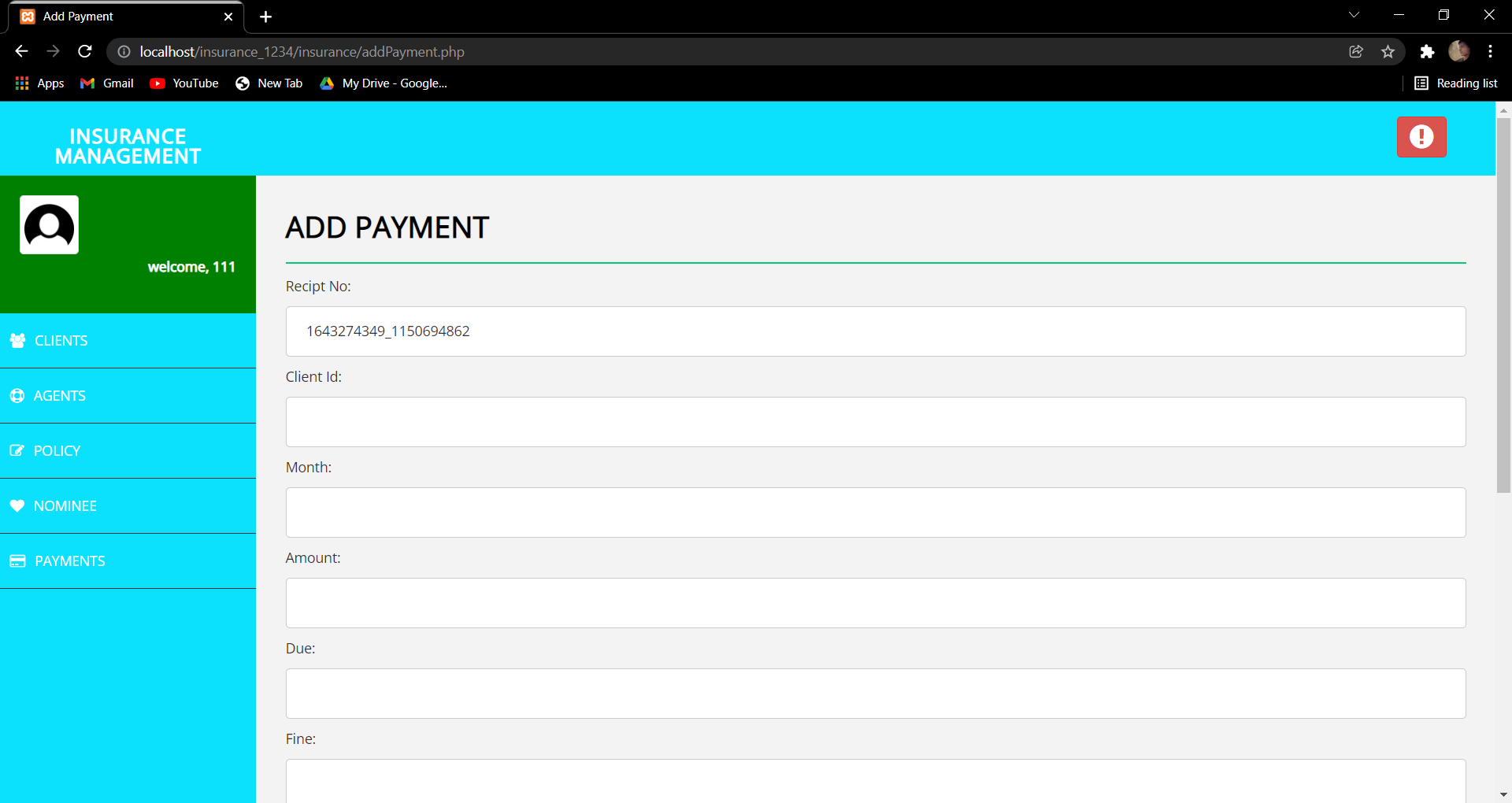


Figure 6.11 : Add Payment

**SEARCH CLIENT:** This page allows to search the client details.

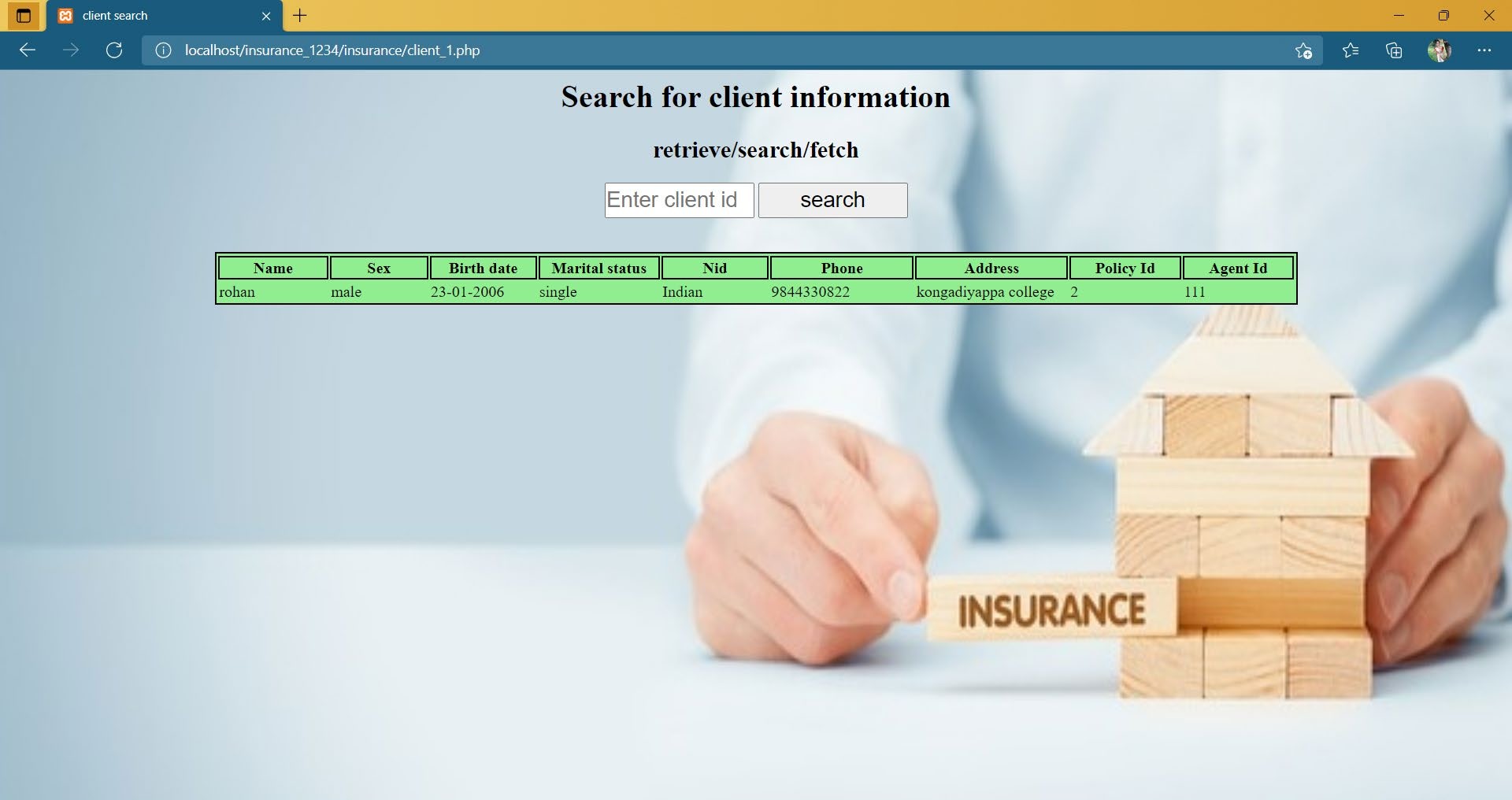


Figure 6.12 : Search Client

## CHAPTER 7

**CONCLUSION**

Insurance is the backbone of a country’s risk management system. Risk is an inherent part of our lives. The insurance providers offer a variety of products to businesses and individuals in order to provide protection from risk and to ensure financial security. In this project, we have to enhance the way the data is stored and the way we fetch the data from the database. The time required to access data has been reduced. In the existing system, unpaid and paid premiums are stored in one table, which in proposed system are in separate tables. So, whenever the admin needs to fetch the data for the paid and unpaid premiums the time required to sort and fetching data is saved.

For future of this project, we can the same thing for separating policies which are running and which are lapsed. The login for admin and customer can be created to protect the data

Dept. Of CSE 26 2021-22

# BIBLIOGRAPHY

## REFERENCES:

**Websites:**

* + 1. [https://www.youtube.com/watch?v=rn8T1bWBXcE](http://www.youtube.com/watch?v=rn8T1bWBXcE)
    2. [https://www.youtube.com/watch?v=kBy-La7DXpg](http://www.youtube.com/watch?v=kBy-La7DXpg)
    3. https://services.lovelycoding.org/insurance-managementsystemproject
    4. https://dl.acm.org/citation.cfm?id=1083734
    5. http://ilpubs.stanford.edu:8090/404/

## Books:

### Software Engineering – A Practitioner’s Approach

* + - 1. Database management System, Ramakrishna and Gehrke, 3rd Edition, 2014,McGraw Hill
      2. Matt Rutledge, 2004, PHP Game Programming, 25 Thomson P
      3. Julie C Meloni, 2002, Teach yourself PHP, MySQL and Apache in 24 Hours Los Altos, California

Dept. Of CSE 27 2021-22