JIMMA UNIVERSITY

JIMMA INSTITUTE OF TECHNOLOGY

FACULTY OF COMPUTING AND INFORMATICS

DEPARTMENT OF SOFTWARE ENGINEERING

DOCUMENT-1 : SOFTWARE ARCHTECTURE AND DESIGN

TITLE: ROAD TRAFFIC ACCIDENT RECORDING SYSTEM

GROUP NAME.....ID_NO

1. HAILU GUDATA .....RU 5717/11
2. MIRETU JALETA ..... RU 1414/12
3. SAMUEL MOKONNON..... RU 1417/12
4. KIBRU G/MEDIN ..... RU 1913/12
5. GEZAGN BEKELE.....RU 1342/12
6. FEYISA GIZAW.....RU 5816/11

Submission Date: 21/01/2023

Submitted to: Ms.Tesfanesh Y

Table of Contents

Introduction

A road traffic accident (motor vehicle collision, motor vehicle accident, car accident, or car/road crash) is when a road vehicle collides with another vehicle, pedestrian, animal, or geographical or architectural obstacle which can result in injury, property damage, and death. Road Traffic Accident (RTA) is major public health problem and a leading cause of death and injury around the world. Poorly maintained road network and vehicle fleets, weak regulatory environment, and undefined roles and responsibilities of various stakeholders towards road transport safety are favorable circumstances for increasing trend in road accidents and fatalities in the developing countries. Each year nearly 1.3 million people die and millions more are injured or disabled as a result of road crashes, mostly in low and middle income countries. As well as creating enormous social costs for individuals, families and communities, road traffic injuries place a heavy burden on health services and economies. The cost to countries, many of which already struggle with economic development, may be as much as 1–2% of their gross national product

2. Background

The World report on road traffic injury prevention, launched jointly in 2004 by the World Health Organization and the World Bank, identified improvements in road safety management and specific actions that have led to dramatic decreases in road traffic deaths and injuries in industrialized countries active in road safety. The use of seat-belts, helmets and child restraints, the report showed, has saved thousands of lives. The introduction of speed limits, the creation of safer infrastructure, the enforcement of limits on blood alcohol concentration while driving, and improvements in vehicle safety are all interventions that have been tested and repeatedly shown to be effective. The World report on road traffic injury prevention also identified the importance of collecting accurate, reliable data on the magnitude of the road traffic injury problem: it highlighted the need for data systems to be put in place to collect the information needed to allow countries to develop evidence-driven road safety policies. The international community must now take the lead to encourage good practice in road safety. To this effect, the United Nations General Assembly adopted a resolution on 14 April 2004 urging that greater attention and resources be directed towards the global road safety crisis. Resolution 58/289 on “Improving Global Road Safety” stressed the importance of international collaboration in the field of road safety. Two further resolutions (A/58/L.60 and A/62/244), adopted in 2005 and 2008 respectively, reaffirmed the United Nations‟ commitment to this issue, encouraging member countries to implement the recommendations of the World report on road traffic injury prevention. In November 2009, ministers and heads of delegations to the First Global Ministerial Conference on Road Safety echoed these calls with the adoption of the Moscow Declaration, resolving to take a number of actions to improve road safety, including improvements to national data collection systems and international comparability of data.

3. Functional Requirement

Functional requirements capture the intended behavior of the system. This behavior may be expressed as services, tasks or functions the system is required to perform. The functional requirements of the RTARS are the following:

- Register road traffic accident
- Generate various reports
- Allow searching of road accident record using different parameters
- Allow administrator, traffic controller to login into the system

Behavioral requirements describing all the cases where the system uses the functional requirements are captured in the use cases presented by the use case diagram illustrated in Figure

4. Quality Requirement

4.1 Usability

The system must be simple and easy to be used by all its potential users.

4.2 Availability

The System is available for 24 hours to the users per week.

4.3 Maintainability

Modifiability

The system should be easily extensible to the need of the government accident data formats availability and to add new functionalities to the system. The system is built from several more or less independent classes which can be used as a standalone application or replaced by other classes. This makes the system easy to change the existing functionality or add new ones when the need arises.

Portability

The system should be easily portable to different platforms. As the ASP.NET languages achieved platform independence through the Common Language Runtime (CLR), the end user can use the system using any browser such as Firefox and Internet Explorer.

Need for Maintenance –

Software Maintenance must be performed in order to:

- Correct faults.
- Improve the design.
- Implement enhancements.
- Interface with other systems.
- Accommodate programs so that different hardware, software, system features, and telecommunications facilities can be used.
- Migrate legacy software.
- Retire software.
- Requirement of user changes.
- Run the code fast

4.4 Testability

This Project (RTARS) testable at its individual stages starting from initial stage of document preparation and while it is in the stage of development. In addition to the testability of this system at each level of development it will also be tested at the end of or at the completion of the implementation of the overall projects by different parts which this project intended for. The System must be Complete, Correct, Feasible, Necessary, Prioritized, Unambiguous, Consistent, Traceable, Concise and Verifiable in order to satisfy the expectance of the users and all system's stakeholders.

5. Architecture Overview

5.1. System Context

The system architecture diagram is an abstract depiction of the system's component architecture. It provides a succinct description of the system's component architecture in order to assist in component-component relationships and system functioning. The system architecture diagram is a visual representation of the system architecture. It shows the connections between the various components of the system and indicates what functions each component performs. The general system representation shows the major functions of the system and the relationships between the various system components.

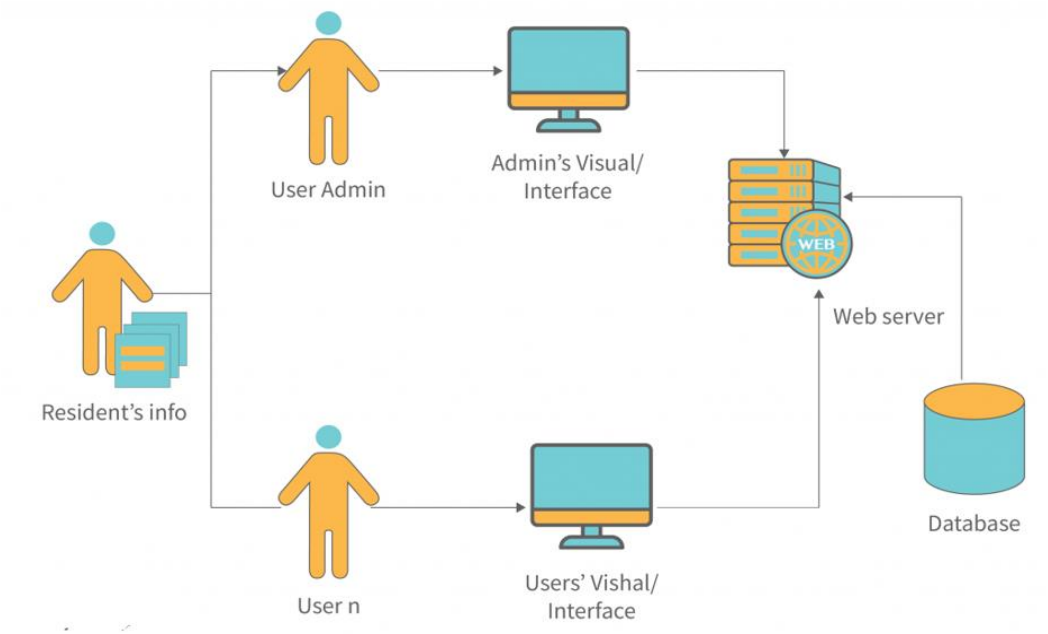

Fig 1: Subsystems, components, and modules Diagram of RTARS

The accident recording system process then consists of the following activities: collection of data, analysis of data, Recording, evaluation of Recording, and closing of the Recording process. The presented system architecture diagram is intended for the Road Traffic Accident Recording System. The system architecture is a representation of the business or customer requirements. The diagram illustrates all of the components that make the system function. As outlined in the

overview, the architecture illustrates how the system functions in general. The system, as a hypothetical user, first registers himself via the presented system architecture diagram is intended for Road Traffic Accident Recording System Architecture. The system architecture is a representation of the business or customer requirements. The diagram illustrates all of the components that make the system function. As outlined in the overview, the architecture illustrates how the system functions in general. The system, as a hypothetical user, first registers himself via the client application and this information is saved in the database. The resident may now gain access to the accident Recording on website.

5.2 User Interaction

The user interface designed for the system starts with a main interface window (homepage) that contains a button link to the main tasks of the system namely View Reports, Search Accident Record and Log In. Activating the “View Reports” link button opens a window which allows users to choose and see different varieties of accident reports and the accident location map. The “Search Accident Record” button link allows users to open a search page (window) that accepts search parameters from users, searches the records and displays the result. The “Log In” button allows an authorized user to log in as an administrator or as a traffic police officer to the system. For an administrator a window that allows him/her to create, rename or delete an account will be opened. For the traffic police officer a window that allows him/her to register accident records will be displayed. The user interface of the system depicted in figure 6.5 shows the interfaces organization hierarchically.

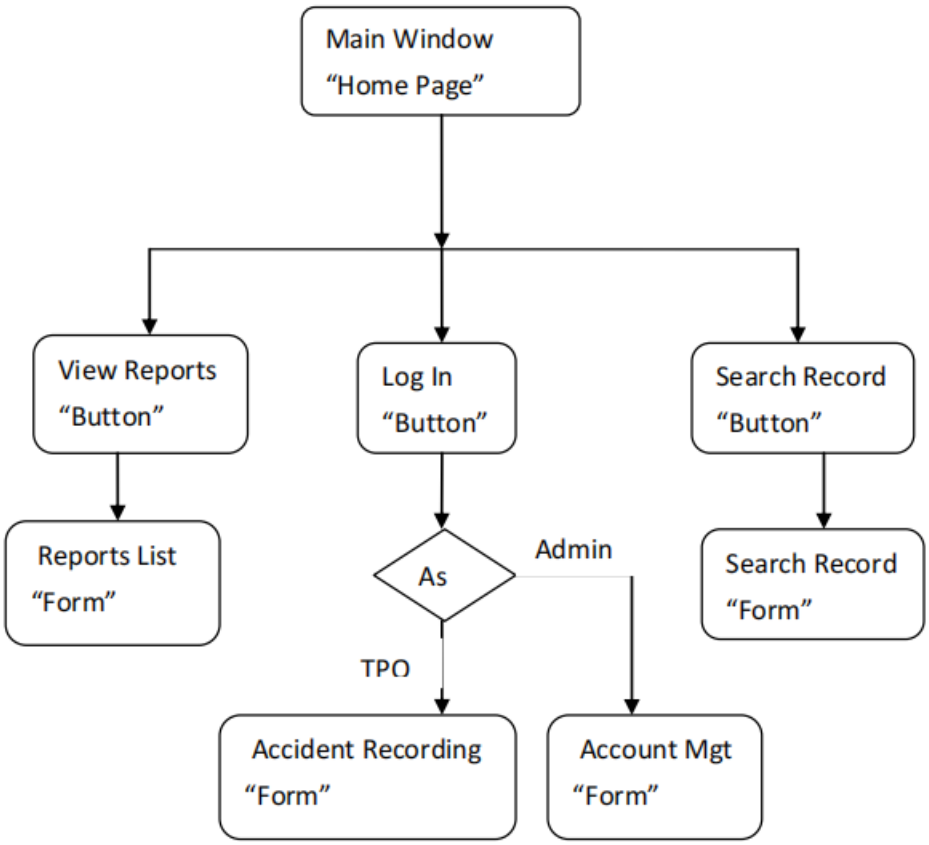

Fig 2: User Interaction

5.3 Data Flow

The System data flows within this system follows according to the below data flow diagram.

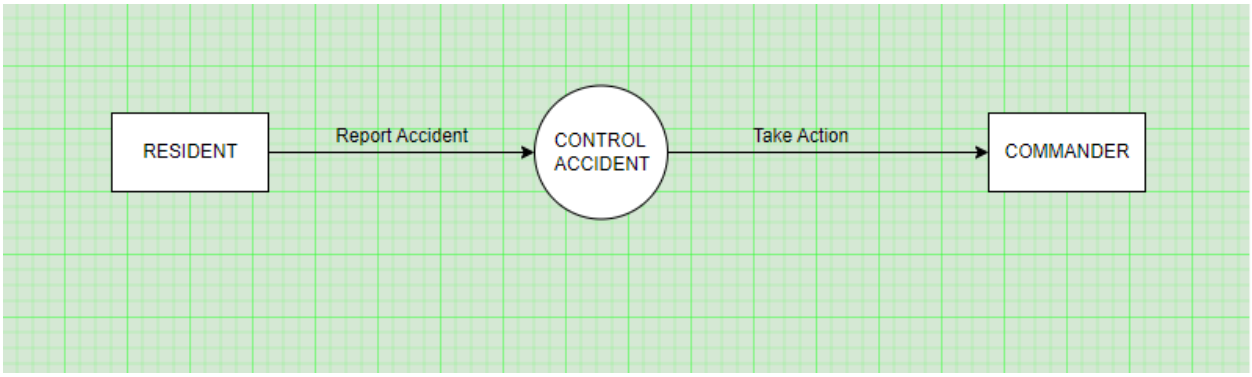

Fig 3: Context Diagram of RTARS.

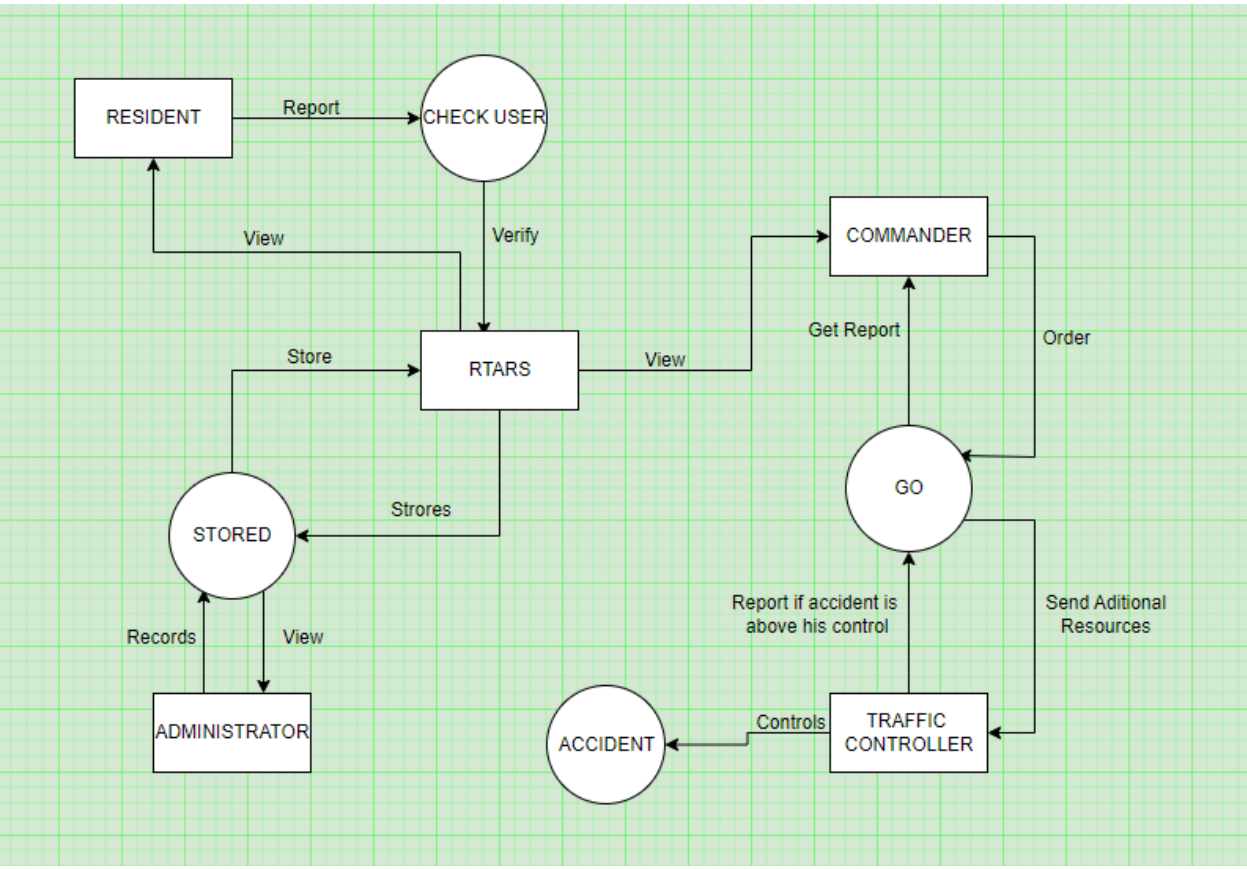

Fig 4: Overall Data Flow Diagram within our System.

6. View

6.1. Logical View

We use UML diagrams to represent the logical view, for example it includes class diagrams.So in our system if there are any accident regarding road accident any person having accessibility to an internet can report an accident. Here the person that may report the accident may include residents, traffic controller. The commander commands the traffic controller. When residents or any user of the system reports the accident the admin only can view and record it. But here since there may be security issue regarding the the account of each user there is authentication and authorization. But all the users of the system need not be registered to use the system, for example Residents, because if having account is necessary, residents that didn't have any account can't able to report. When we describe the above statements by using the class diagram, the logical view of the system looks like the following.

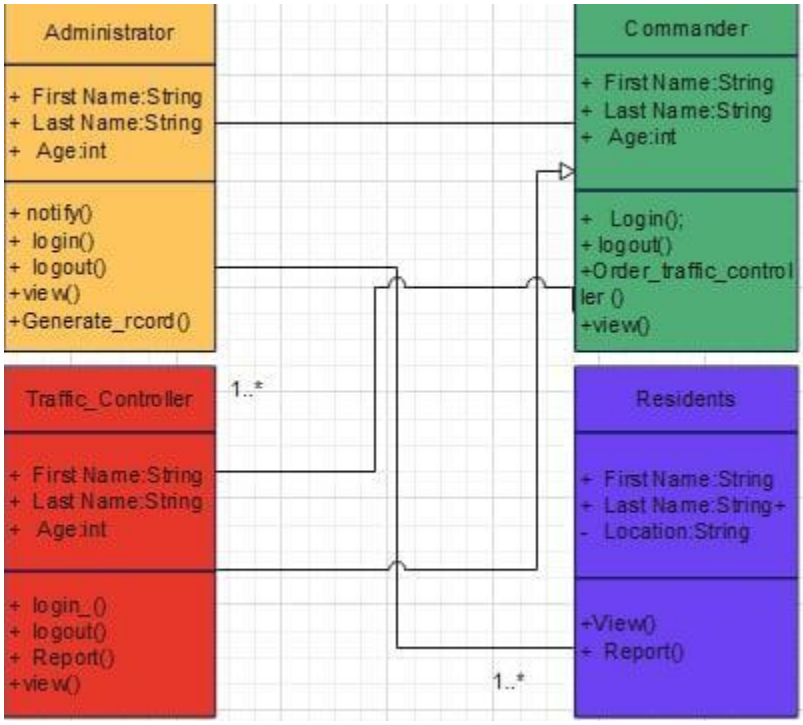

Fig 5: The layered view of the system is can be described as the following diagram.

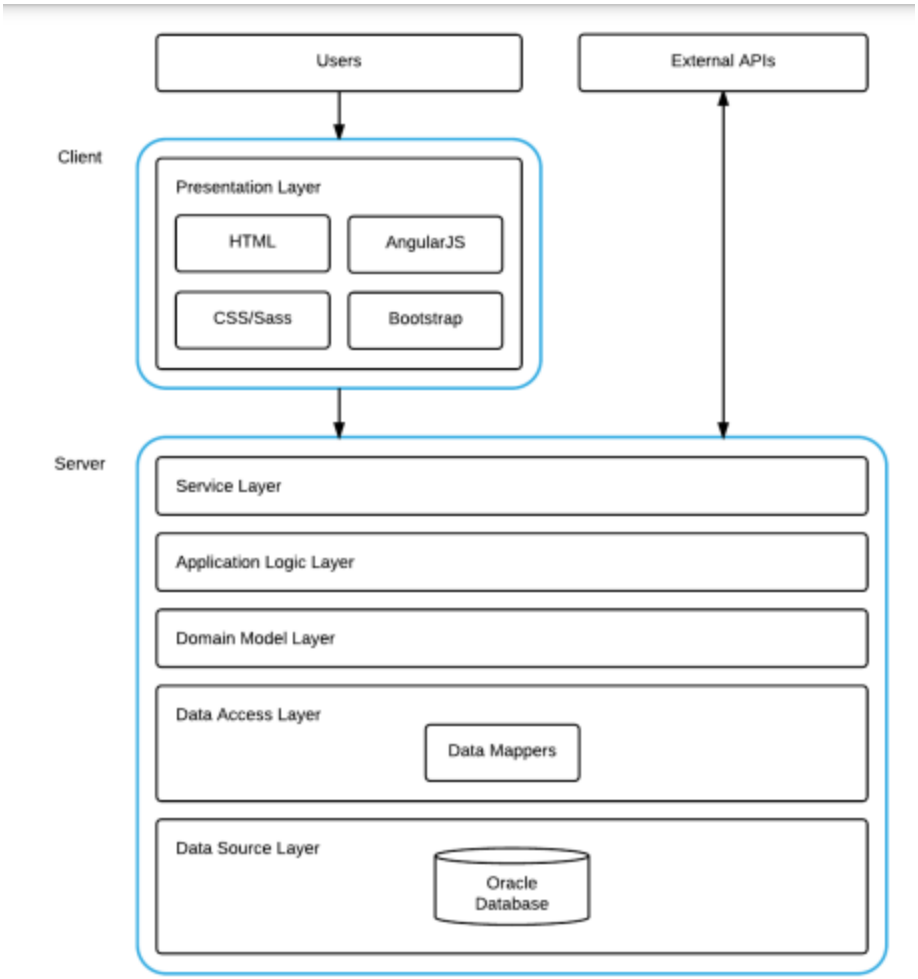

Fig 6: Layered Structure of the system

Presentation Layer

It will be composed of HTML, CSS, JavaScript, Angular and other related files that will run in the user's web browser of choice.

Application Logic Layer

The role of the application logic layer is to encapsulate the system controllers, which implement much of the core business logic. It will also serve as the connection between the user interface and the domain model, thus maintaining the separation of concerns.

Service Layer

The service layer is in charge of modularizing different features (services) of the application. Certain features, such as reporting, require access to external APIs; the service layer will be in charge of interacting with those external APIs. An individual service can also use the application logic layer to perform core business logic. Each service will act as a facade, serving as a general interface for a feature that can be accessed by the presentation layer. These facades (interfaces) will provide the ability to manipulate or fix the code underneath each service without affecting the way in which that service is called by other layers. Modularizing the services within the system will also make it easier to pinpoint potential defects, allowing for easy maintainability. Each service can be tested individually, which allows for good system wide testability.

Domain Model Layer

The domain model contains all of our system's object representations of data in our system. This also includes associated methods for any objects that contain their own functionality.

Data Access Layer (DAL)

The data access layer contains all of the mappers to the data in the system. The mappers are in charge of the coordination of all communication between the objects in the domain layer and their corresponding tables in the database. This ensures that domain layer objects have no knowledge of the database, its schema, or any SQL interface.

Data Source Layer

All persistent information and any external API integration (e.g. SIS, Simplicity) make up the data source layer. This includes the Oracle database that will contain all of the data for the system. At this time there is no expectation for integration with external systems, but the system should be architected to accommodate such integrations in the future, as there are a few options on the table.

Relations

Presentation Layer to Application Logic Layer

The presentation layer represents the view of the system, and the application logic layer contains the controllers, which house the logic for the different roles. In following with the MVC architectural pattern, the controllers take information from the view and use it to modify or request related data in the model. This prevents the view from directly modifying the model, and instead has the view display changes to the data.

Application Logic Layer to Domain Layer

The application layer modifies the data encapsulated in the domain layer according to the business logic rules before it reports changes back to the presentation layer to be displayed to the user.

Domain Layer to DAL to Data Source Layer

The domain layer is the active representation of the information stored in the data layer. When it is time to store data according to the procedures defined in the upcoming detailed design stage, the changes to the domain layer will be pushed to the data source layer to create a persistent copy of the data to be stored across sessions through the DAL. All interactions between the DAL and Data Source Layer will indirectly be SQL queries in accordance with the ACID properties.

6.2. Process View

The proposed client-server architecture is discussed in the context of a prototype software package that was developed to display real-time traffic information for Jimma. Unlike other Internet-based applications where the user downloads a static congestion map, this application is designed to run on the client (user) computer and access the Internet to retrieve the real-time traffic data from the road authority's data server and report any kind of information's and the problems like accident they faced. Using a graphical interface and a client-based map of the freeway, this Web-based application provides the user with information on congestion levels,

travel speed limit and real-time estimates of travel times between specific locations on the facility. This process is much faster than downloading an updated congestion map and allows the traffic data server to process many more clients concurrently and more efficiently. The proposed client-server architecture includes a number of enhanced features such as personalized travel information, traffic prediction, incident information and dynamic route guidance and navigation instructions. The prototype developed in this study clearly demonstrates the feasibility of implementing the enhanced client-server architecture as a basis for an Internet-based user information system for Jimma people's.

At the high level, the architecture of an application defines how different parts of the system are organized and logically separated yet ensuring that they work together. The architecture used for the system is three tiers Client-Server architecture: client tier, middle/web tier and the data tier as illustrated in Figure 6.1. Such architecture is one of the most commonly used type of architecture for web-based applications as it provides greater application scalability, high flexibility, high efficiency, lower maintenance, and better re-usability of components.

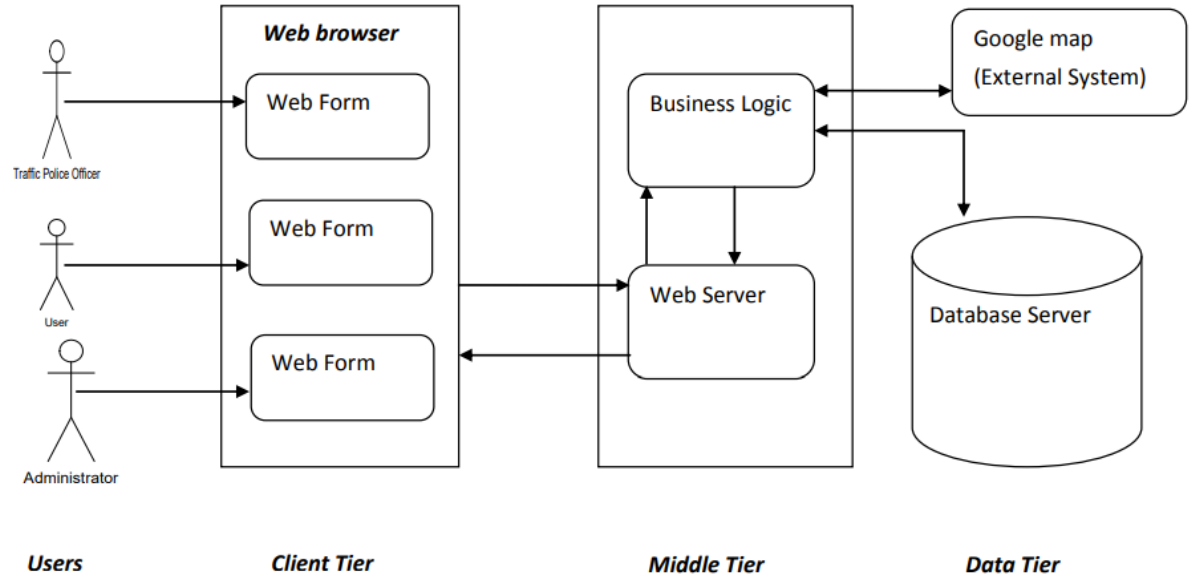

Fig 7: Client-Server Structure of the RTARS

The client tier is the application's user interface containing data entry forms, report access links and client side application that are running on the web browser of the user machine.

The traffic police officer, the administrator and other users of the system interact directly with the application through this user interface. It interacts with the web application server to make requests and to retrieve data from the database. It then displays to the user the data retrieved from the server.

The middle tier contains two parts of the RTARS application, i.e., the web server (application server) and the business logic. The web server (IIS) handles all HTTP requests coming from the client machines. The requests could be a request for adding new records, displaying existing records, or a request for report generation and others. It is also the web server which manages the responses that is forwarded to the client machines.

The business logic tier is responsible for handling all the core functionalities of the system such as input validation, performing calculations, report generation, access and retrieval of any data required by the client. When the data is submitted from the client machines, first it will be handled by the functions of the web server and then transferred to the business logic for processing. Again, the business logic processes the data and sends it either to the database or back to the web server, this is determined by the type of service required. It also interacts with an external system called Google Earth to display accident locations on a map.

The data tier layer is concerned with the data storage and persistence issues. It is implemented using an SQL database. The database can either be stored on the web server itself or on a different machine; however it needs to be easily accessible by the web server.

7. Architecture Style

At a high level, the architecture of an application defines how different parts of the system are organized and logically separated yet ensuring that they work together. The architecture used for the system is three tiers Client-Server architecture: client tier, middle/web tier and the data tier. Such architecture is one of the most commonly used type of architecture for web-based

applications as it provides greater application scalability, high flexibility, high efficiency, lower maintenance, and better re-usability of components.

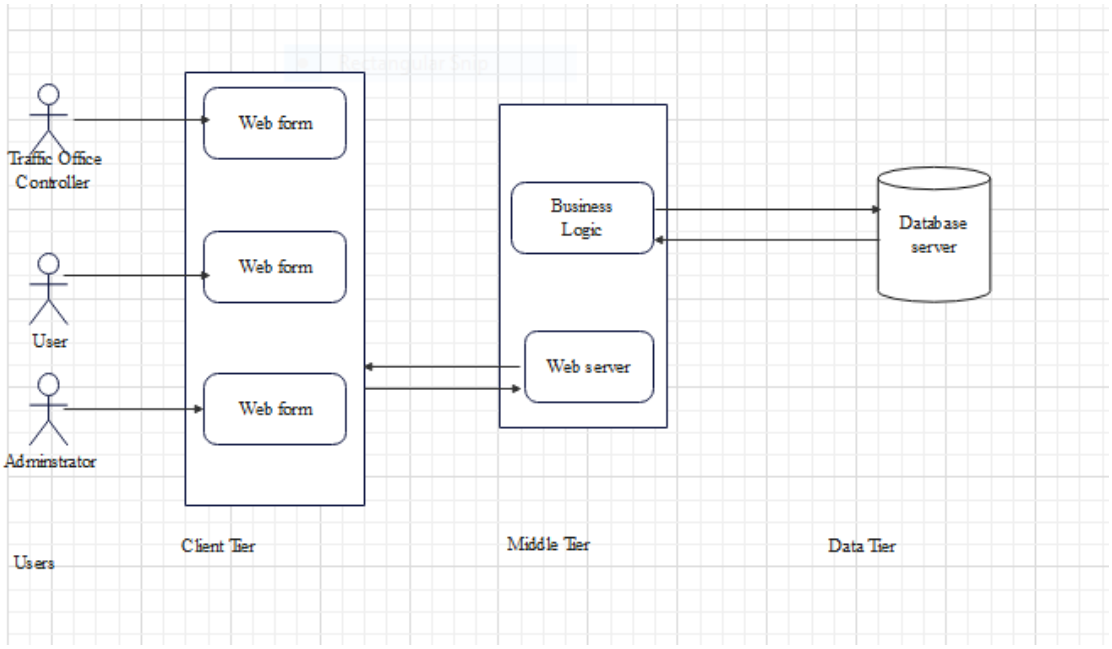

Fig 8: Architectural Styles of RTARS

The client tier is the application's user interface containing data entry forms, report access links and client side application that are running on the web browser of the user machine. The traffic police officer, the administrator and other users of the system interact directly with the application through this user interface. It interacts with the web/application server to make requests and to retrieve data from the database. It then displays to the user the data retrieved from the server.

8. Technologies Stack

In our projects we use different different technologies for front-end we use angular due to its ability to create multiple modules for a single web application. These modules can then act independently, and we can compile them to create a single running, dynamic application for end-users, for back-end we use Django; because it is a high-level ,open-source, MVC python web framework for secure and maintainable websites, database we use is SQLite.

API style we use composite API s. Because Composite API s generally combine two or more API s to craft a sequence of related or interdependent operations. Composite API s can be beneficial to address complex or tightly related API behaviors and can sometimes improve speed and performance of individual API s.

We use cloud due to its flexibility, reliability and security, cloud removes the hassle of maintaining and updating systems, allowing us to invest our time, money and resources into fulfilling our core business strategies.

9. References

1. https://www.prolifics-testing.com/news/ten-attributes-of-a-testable-requirement.

2. SRC: https://en.wikipedia.org/wiki/4%2B1_architectural_view_model.

3. https://ieeexplore.ieee.org/document/917550.