|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | **https://lh4.googleusercontent.com/proxy/YA9Xoqs7jhpeuwrEjwhdi_EVSCDwUdpr72V-2YHZ2lz2y1FaqityK8c8RlZRTvUDEw3Y2TekyGNi07wcREil5Ez3ii80dA-DE8G6HAQjEmJVz8W32Wy2uaDAWwuZs6uPZtJp2zrUJ_Qps2T1CUmSpuPR8dk2XA=w128-h144-k-no** | | | **Document Ref.:** | | **SRS003** | |
| **Version No.:** | | **V01** | |
| **Date :** | | **14.10.2019** | |
| **Project Name:** | **Automatic Testing Software for rating and ranking automobiles** | | | | | |
| **Project Code:** | **003** | | | | | |
| **Status:** | **Draft** | | | | | |
| **Document Type:** | **Controlled** | | | | | |
| **High Level Document for Automatic Testing System (ATS) for rating and ranking automobiles based on testing parameters.** | | | | | | |
| **This document is aimed at eliciting the High-Level Document for an Automatic Testing System (ATS) which is used to rank and rate automobiles based on various testing parameters.** | | | | | | |
| **Prepared By: Team 003** | | | | **Reviewed By:** | | |
| **Name** | | **Date** | | **Name** | | **Date** |
| **Abdul Qadir** | | **14.10.2019** | | **Dr Anant Koppar** | | **14.10.2019** |
| **Abhijit Venugopal** | | **Approved By:** | | |
| **Akash Shetty** | | **Name** | | **Date** |
| **Alan Jacob** | | **Dr Anant Koppar** | | **14.10.2019** |
| **Aman Bhat T** | |
| **Akilesh S** | |
| **Harshit Jitendra Dave** | |
| **Distribution List** | | | | | | |
| **Project Representative(s)** | | | | **Guide Representative(s)** | | |
| 1. **Abdul Qadir** 2. **Abhijit Venugopal** 3. **Akash Shetty** 4. **Alan Jacob** 5. **Aman Bhat T** 6. **Akilesh S** 7. **Harshit Jitendra Dave** | | | | **Dr Anant Koppar** | | |

TABLE OF CONTENTS

[**Definitions, Acronyms and Abbreviations 2**](#_heading=h.3as4poj)

[**References 2**](#_heading=h.1pxezwc)

[**Change History 2**](#_heading=h.49x2ik5)

[**1.0**](#_heading=h.2p2csry) **Introduction 2**

[**1.1**](#_heading=h.147n2zr) **Overview 2**

[**1.2**](#_heading=h.3o7alnk) **Scope 2**

[**2.0**](#_heading=h.23ckvvd) **Design Constraints, Assumptions and Dependencies 2**

[**3.0**](#_heading=h.ihv636) **Design Description 2**

[**3.1**](#_heading=h.32hioqz) **Master Class Diagram 2**

[**3.2**](#_heading=h.1hmsyys) **Module 1 2**

[*3.2.1*](#_heading=h.41mghml) *Description 2*

[*3.2.2*](#_heading=h.2grqrue) *Use Case Diagram 2*

[*3.2.3*](#_heading=h.vx1227) *Class Diagram 2*

[**3.2.3.1**](#_heading=h.44sinio) **Class Description 1 2**

[**3.2.3.2**](#_heading=h.2jxsxqh) **Sequence Diagram 2**

[**4.0**](#_heading=h.3fwokq0) **ER Diagrams 2**

[**5.0**](#_heading=h.1v1yuxt) **User Interface Diagrams 2**

[**6.0**](#_heading=h.4f1mdlm) **Report Layouts 2**

[**7.0**](#_heading=h.2u6wntf) **External Interfaces 2**

[**8.0**](#_heading=h.19c6y18) **Packaging and Deployment Diagrams 2**

[**9.0**](#_heading=h.3tbugp1) **Help 2**

[**10.0**](#_heading=h.28h4qwu) **Alternate Design Approach 2**

[**11.0**](#_heading=h.nmf14n) **Reusability Considerations 2**

[**12.0**](#_heading=h.37m2jsg) **Traceability Matrix 2**

**Definitions, Acronyms and Abbreviations**

ATS – Automatic Testing System – This is the software that is to be implemented in this project.

HTTP - HyperText Transfer Protocol

**Change History**

This section describes the details of changes that have resulted in the current High-Level Design document.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **#** | **Date** | **Document Version No.** | **Change Description** | **Reason for Change** |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

# Introduction

* 1. ***Overview***

The purpose of this High Level Design (HLD) Document is to add the necessary details to the current project description to represent a suitable model for coding. This document is also intended to help detect contradictions and design errors prior to coding, and can be used as a reference for how the modules interact at a high level. It also aids in traceability, wherein the different design and coding choices used in future can be traced to the original high level design choices.

* 1. ***Scope***

The HLD documentation presents the structure of the system, such as the database architecture, application architecture (layers), application flow (Navigation), and technology architecture. The HLD uses non-technical to mildly-technical terms which helps the administrators and stakeholders get an overall idea of the system and how it will be implemented in the future stages. Some available tools and packages will be used, whose design and architecture will not be explained intensively. Similarly, underlying network architecture will not be explained in detail as it uses standard web based architecture.

# Design Constraints, Assumptions and Dependencies

**General Constraints:**

The Automatic Testing System must be efficient, user friendly and as automated as possible. The interface provided to the different types of users must be simple and easy to use, while at the same time, making the process of ranking and displaying results efficient. Different types of users must be redirected to their interfaces, where they will only need to enter the relevant details, in order to enable ATS to do it’s functionality. Authentication is necessary, without which ATS will not admit entry to it’s interfaces. Different web pages must be present for different inputs and outputs. Response time at each interface must be kept minimum.

**Assumptions:**

The main assumption is that the different modules of ATS are capable of working together to give the final results without external or internal interferences. Another assumption is that the different levels of users of ATS will be able to input their contributions in the correct manner, enabling ATS to function optimally. Server-Client connection is assumed to be sufficiently fast and lossless, so that data and transactions are not lost. Another assumption is that the different technology used in ATS interfaces will be available on the client side without versions being a problem.

**Dependencies:**

* Proper functioning of Internet Protocols, for efficient and lossless communication.
* Browser functionalities and features for interfaces.
* Data provided is in intended format and is error-free.
* Database dependencies
* Header and library dependencies

# Design Description

This section clearly defines the interfaces that exist between the predictor tool module, the security/login module and the database module.

The predictor tool module is concerned with calculating the rating of the automobiles based on several testing parameters and ranking them accordingly.

The Security/login module is concerned with ensuring authorised access to the software components by the users.

The database module stores vehicle data that the software uses either directly or indirectly. It is concerned with storing auto mobile details of each car company wise.

* 1. ***Master Class Diagram***

A class diagram of the entire system will be given at a high level and then broken down into sub levels in each of the classes below.

* 1. ***Security Module***

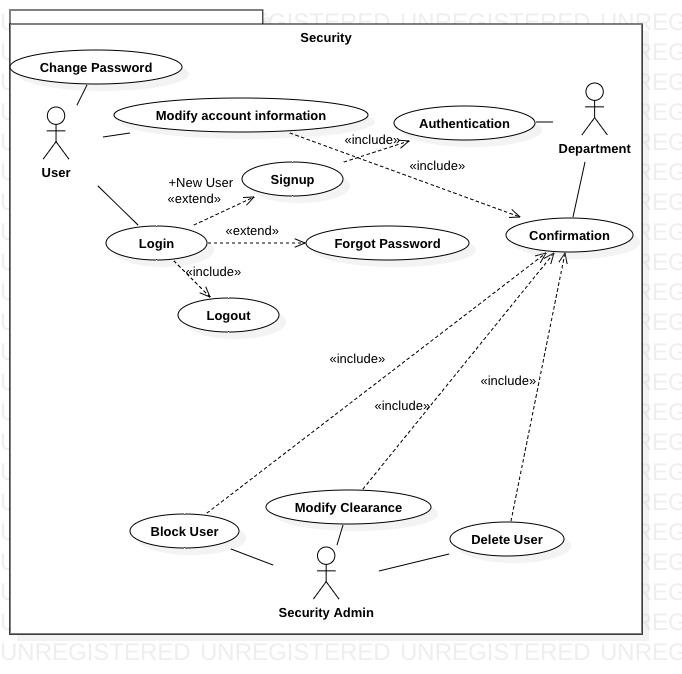
**…**

### Description

Security module assigns different levels of clearance to each user of the software. Users are issued a unique 8 digit UID which is used to determine their clearance level and redirect them to their dashboards which are customized based on their clearance level. New users can sign up by creating a new account and providing the relevant details like name, department, contact details etc.They are issued the UID after verifying with the specific departments of the company.

### Use Case Diagram

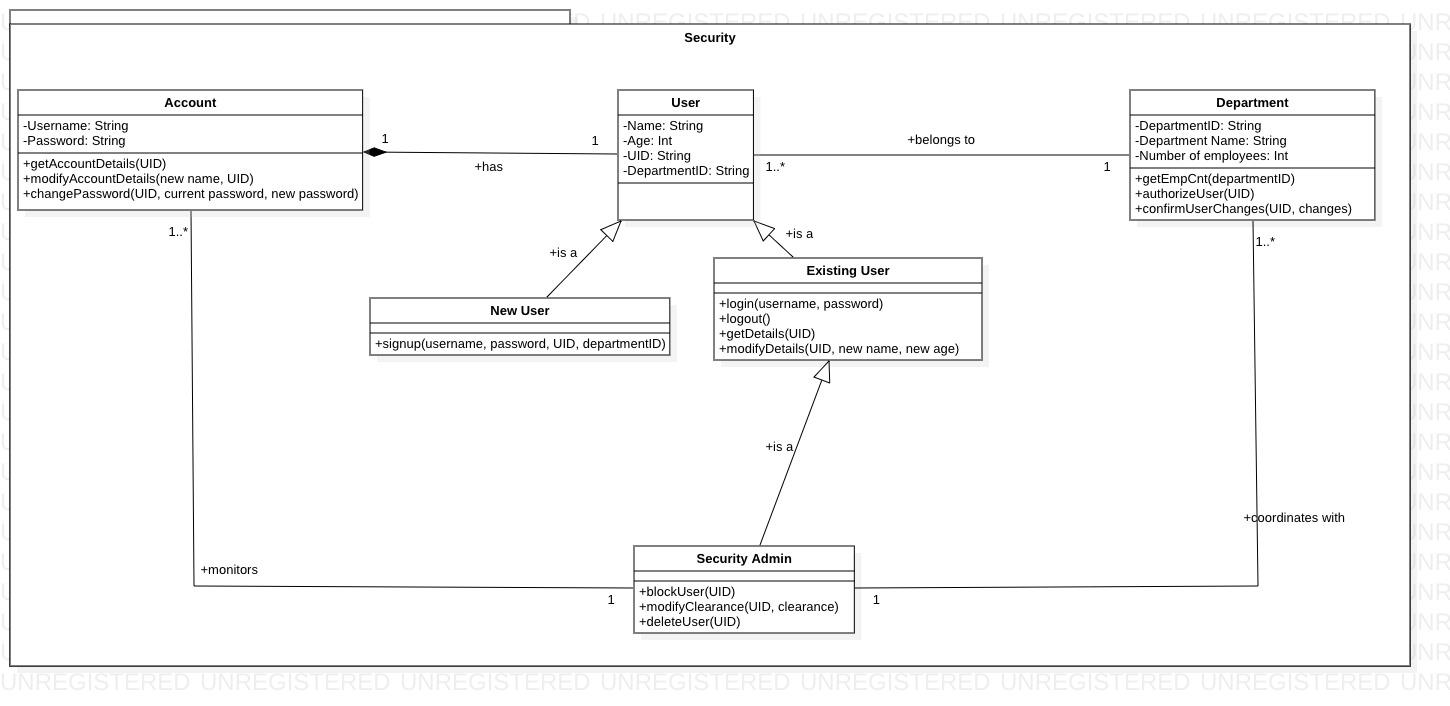
This section depicts the use-case diagram of the security module with descriptions of each use case item present in it.



|  |  |
| --- | --- |
| **Use Case Item** | **Description** |
| User | This item refers to the users of the software. |
| Security Admin | This is the administrator of the module who maintains the functionality of this module. |
| Department | This refers to the group to which each user belongs, which is used to determine his privileges. |
| Login | This refers to the users logging in to their accounts. |
| Logout | This refers to the users logging out of their accounts. |
| Change password | This is an option given to users to change their account’s password. |
| Modify  account information | This is an option given to users to change their account details. |
| Signup | This is an option for new users to create an account |
| Forgot password | This option is given to users in case they forget their passwords and need an alternate way to login. |
| Authentication | This refers to the department authenticating the validity of new users. |
| Confirmation | The department has to confirm any changes to its users’ accounts. |
| Block user | The security admin can block a user if he enters incorrect details 5 times. |
| Modify clearance | The security admin can modify the clearance of the use. |
| Delete user | The security admin can delete the account of a user. |

### Class Diagram

This section depicts the class diagrams of the security module with a detailed description.



#### Class Description

#### User

This class is used to represent each user in the organisation. It consists of attributes i.e. Name, Age, UID and DepartmentID. The methods in this class are:

1. getDetails(): Returns the details of the user.

2. modifyDetails(): Used to modify the user’s details.

* + - * 1. ***New User***

This is a specialization of the user class. This class has an additional method signup which is used to create a new account.

* + - * 1. ***Existing User***

This is a specialization of the user class. This class has the methods login and logout. Login is used to log in to the account and logout is used to log out of the account.

* + - * 1. ***Security Admin***

This is a specialization of the existing user class. This class has the following methods:

1. blockUser(): This method is used to block a particular user.

2. modifyClearance(): This method is used to modify the clearance of a particular user.

3. deleteUser(): This method is used to delete a user’s account

* + - * 1. ***Department***

This class is used to represent each department in the organization. It consists of attributes i.e. DepartmentID, Department Name and Number of employees. The methods in this class:

1. getEmpCnt(): This method is used to get the number of employees in the department.

2. confirmUserChanges(): This method is used by the department to authorize any change to the user’s account such as deletion, blocking and modification of clearance.

3. authorizeUser(): This method is used to authorize the department of a new user of the department.

* + - * 1. ***Account***

This class is used to implement the users’ accounts. It consists of attributes i.e. username and password. The methods that belong to this class are:

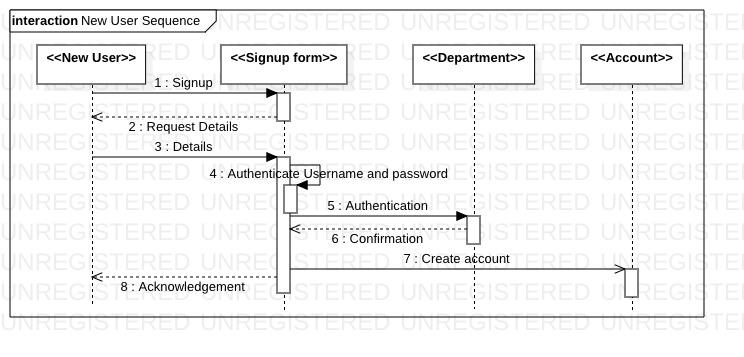
1. getAccountDetails(): Returns the account details of the user.

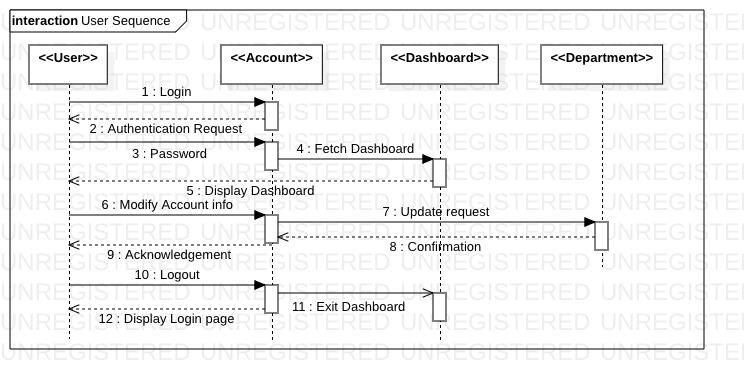
2. modifyAccountDetails(): This method is used to modify the account details.

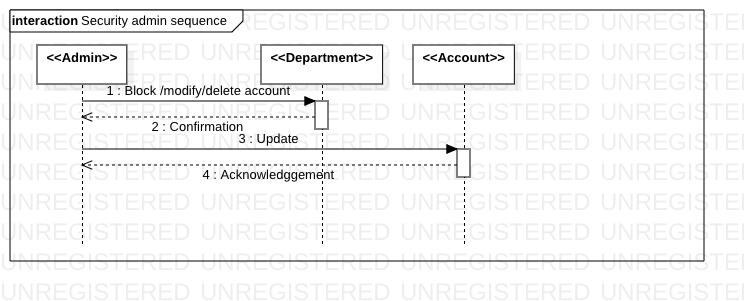
3. changePassword() : This method is used to change the password of the user.

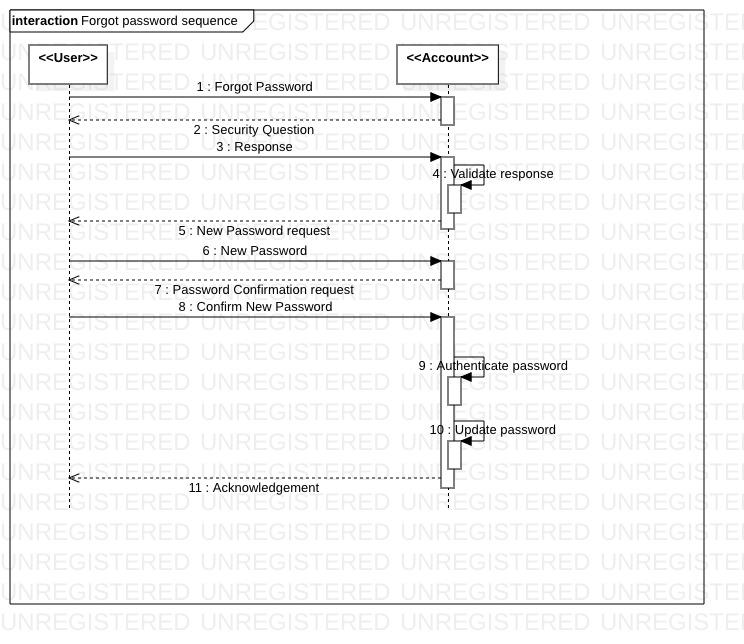
#### Sequence Diagram

The Sequence diagram for security module is presented here.







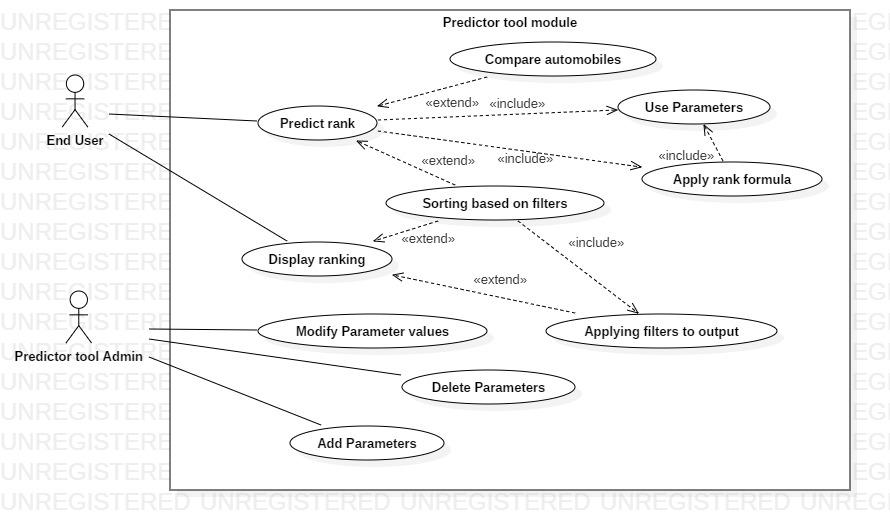


**3.3 Predictor Tool Module**

**3.3.1 Description:**

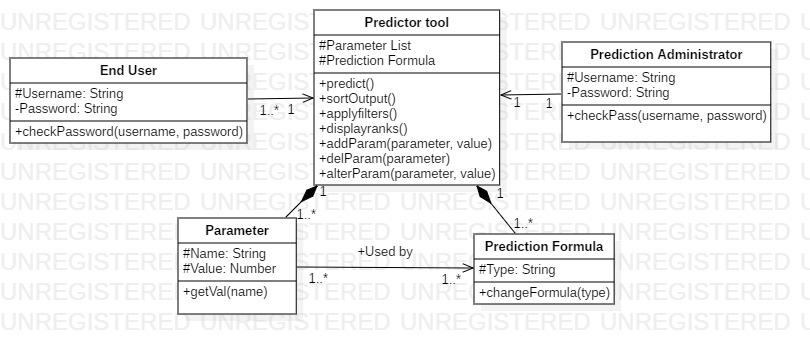
The Predictor tool module is the core of the ATS and is responsible for performing computations and evaluating the ranks of the automobiles. This module has interactions with the GUI and the Database backend. There is a score evaluation formula, which is a weighted formula involving the different parameters from the database, which is used to predict the score of the automobile. The end user can use this module to predict and display automobile ranks and apply filters on the results. The predictor tool administrator can add, alter and delete different parameters involved in rank prediction according to the type of automobile or other constraints.

**3.3.2 Use Case Diagram:**



|  |  |
| --- | --- |
| **Use Case Item** | **Description** |
| End User | This item refers to the end users of the software. |
| Predictor tool Admin | This item refers to the Predictor tool administrator who is responsible for altering the parameters used in rank prediction. |
| Compare automobiles | This item is a use case which is used to compare different automobiles based on predicted ranks. |
| Predict rank | This item is a use case which is used to predict the rank of the automobile. |
| Use Parameters | This item is a use case which uses the parameters stored in the database. |
| Apply rank formula | This item is a use case which applies a preset formula for score evaluation. |
| Display ranking | This item is a use case which displays the ranking of the automobile. |
| Sorting based on filters | This item is a use case which is used to sort the automobiles based on some filter criteria. |
| Applying filters to output | This item is a use case which is used to apply different filter criteria to the outputs displayed. |
| Add Parameters | This item is a use case used to add parameters to the parameter data table. |
| Delete Parameters | This item is a use case used to delete parameters from the parameter set. |
| Alter Parameters values | This item is a use case used to change the values of existing parameters. |

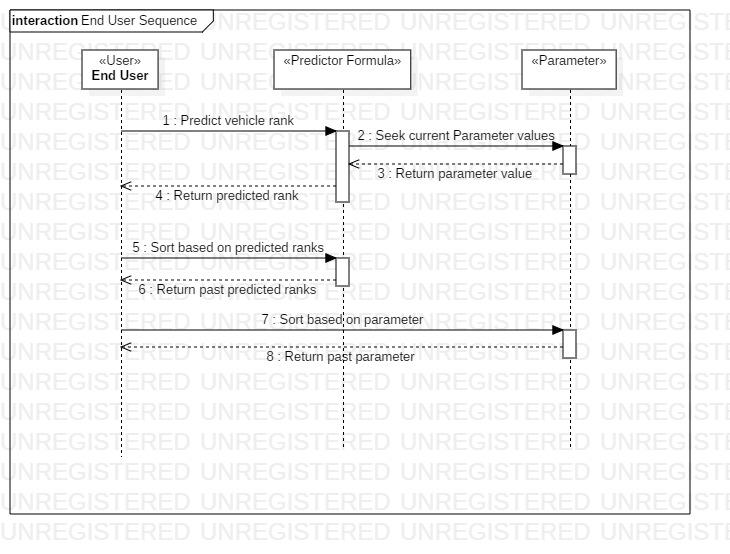
**3.3.3 Class Diagram:**

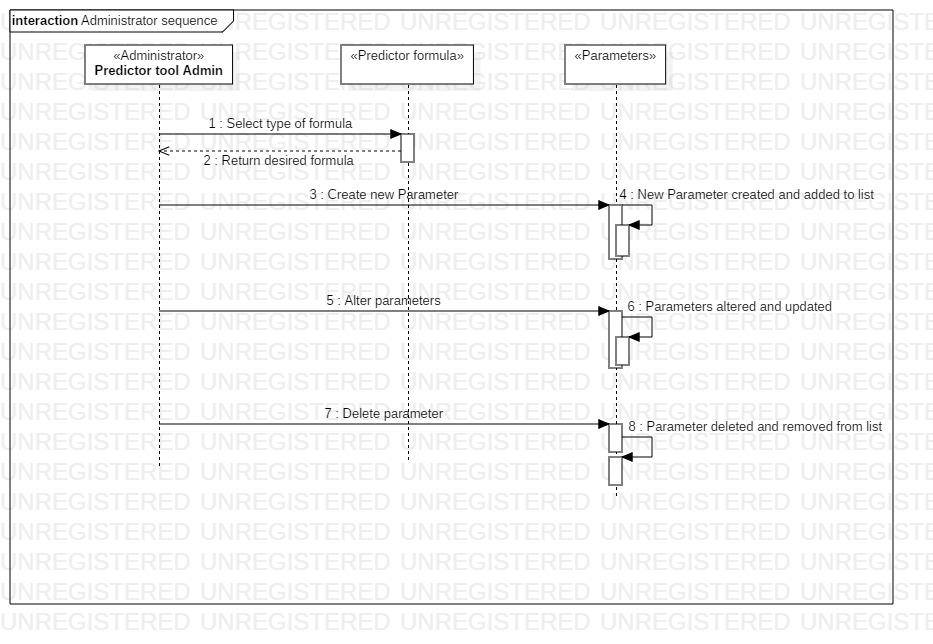
****

**3.3.3.1 Class Description:**

* End User class: This is used to represent the end user who uses the predictor module to predict automobile ranks. The user has attributes username and password and a method checkPassword to check if the entered password matches with the username. It has an association with the predictor tool class.
* Predictor tool class: This is used to represent the predictor tool which consists of the Parameter list and Formula used for rank prediction of automobiles. It consists of several operations such as predict(), displayranks() and so on, which are used by the end user to predict, display ranks, sort outputs based on filters and so on. There are also operations provided to add, alter and delete parameter values.
* Parameter class: This is used to represent the class of parameters used in rank prediction. Each parameter has a name and a value associated with it. This value can be retrieved using getval() operation. The Parameter class exists as a part of the Predictor tool class and cannot exist independently, hence the composition.
* Prediction Formula class: This is used to represent the weighted formula used for automobile rank prediction. Different types of formula (such as linear, exponential or priority based) are present which can be used according to the type of vehicle or type of rank prediction required. The operation changeFormula() can be used to switch between the formulae. This class also exists as a composition of Predictor tool class
* Prediction Administrator class: This is used to represent the predictor tool administrator who can alter the type of formula or the parameters in the parameter list. This type of user has access to functions of the predictor tool class such as addParam()- to add a parameter, alterParam()- to alter the value of an existing parameter and delParam()- to delete an existing parameter. The prediction administrator can also change the type of formula based on necessity using changeFormula() method.

**3.3.3.2 Sequence Diagrams:**

****

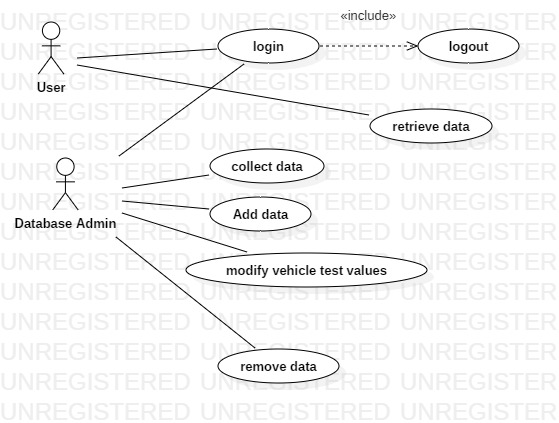
****

**3.4 Database Module**

**3.4.1 Description:**

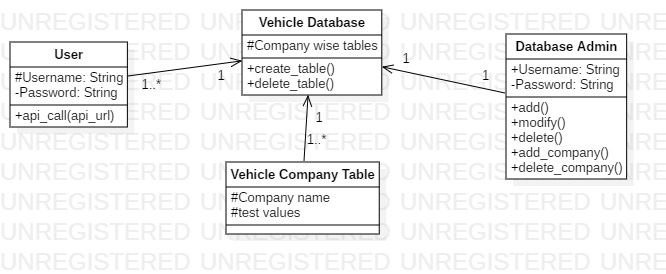
The Database Module is responsible for storing the performance test values of all vehicles. Only the database admin is capable of adding, deleting or modifying entries in the database. The database admin is responsible for collecting the data for each car and the admin has to feed the data into the database. Later on, if the performance test values of vehicles change then the admin has to make necessary modification in the database. The Database has several tables each table stores the data for a particular car manufacturing company. The predictor admin uses the data from this database to perform calculations and rank cars accordingly.

**3.4.2 Use Case Diagram:**



|  |  |
| --- | --- |
| **Use Case Item** | **Description** |
| Database Admin | This item refers to the database admin. |
| login | The database admin has to login. |
| logout | The database admin has to logout after modifying the database |
| Collect data | The database admin has to visit several car manufacturing companies and collect relevant data. |
| Add data | add new data to database. |
| Modify test values | When performance of vehicles changes, the database admin has to update the database |
| Remove data | If a car is not in production the database admin has to remove that entry from the database. |
| User | This is the user who wants to generate ranking. |
| Retrieve Data | retrieve automobile data from database |

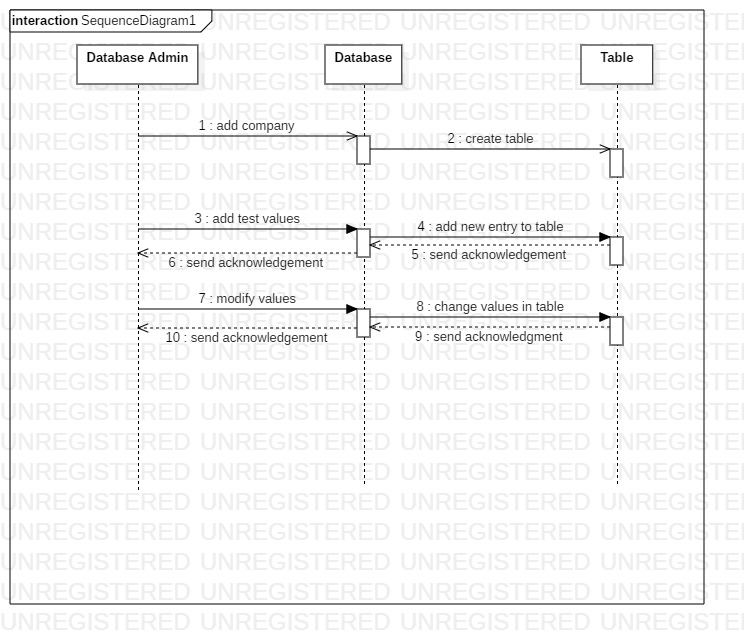
**3.4.3 Class Diagram:**

****

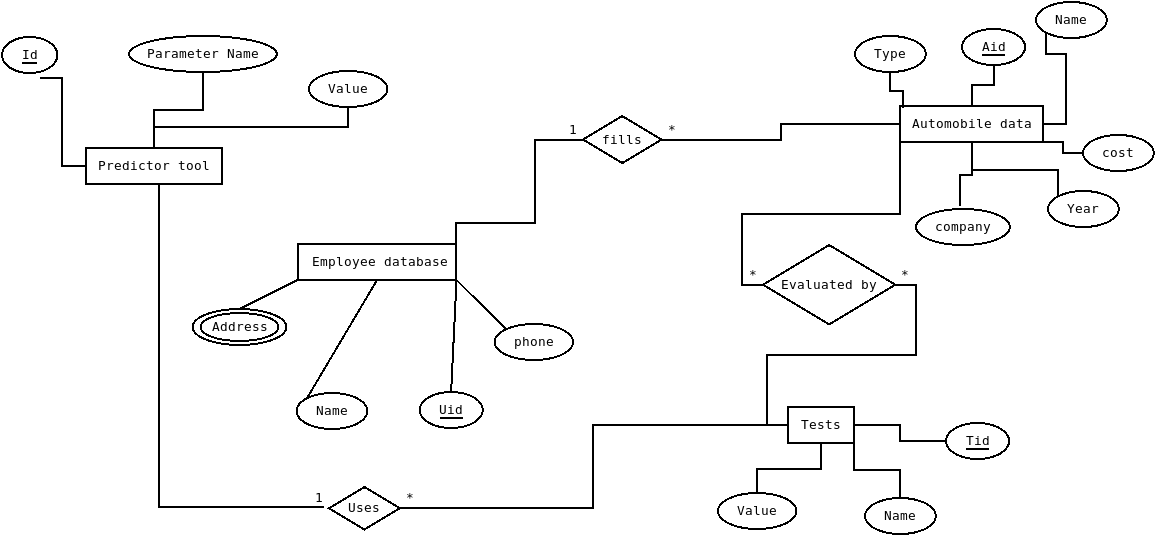
**3.4.3.1 Class Description:**

* User: This is the user who send api calls to the vehicle database to get information about vehicles.
* Vehicle Database: This database stores the tables of information about all cars. Each table in this database represents the car of a particular company.
* Vehicle company table: The vehicle company table stores information of cars of a particular company.
* Database Admin: This represent the database admin who is responsible for collecting adding, deleting or modifying test values of cars.

**3.4.3.2 Sequence Diagrams:**



# ER Diagrams



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **#** | **Entity** | **Name** | **Definition** | **Type** |
| **ENTITIES** | | | | |
|  | Employee | Employee database | The database admin is responsible for collecting the data for each car and the admin has to feed the data into the database. | Table |
|  | Test | Tests | Test Values stored in this table | Table |
| 3. | Predictor Tool | Predictor tool | Weights are stored in  this table | Table |
| 4. | vehicle | Automobile data | Stores the Vehicle data | Table |
| **#** | **Attribute** | **Name** | **Definition** | **Type (size)** |
| **DATA ELEMENTS** | | | | |
|  | Key Attribute | Uid | User id | int |
|  | Multivariate | Address | Address of the user | string |
| 3. | Attribute | Name | Name of user | string |
| 4. | Attribute | Phone | Phone number of user | string |
| 5. | Key Attribute | Tid | Test id | int |
| 6. | Attribute | Name | Parameter Name | string |
| 7. | Attribute | Value | test value | int |
| 8. | Key Attribute | Aid | Automobile id | int |
| 9. | Attribute | Name | Automobile Name | string |
| 10. | Attribute | Type | Automobile Type | string |
| 11. | Attribute | Cost | cost of the vehicle | string |
| 12. | Attribute | Year | Year of production | date |
| 13. | Attribute | Company | Vehicle company name | string |
| 14. | Key Attribute | Id | Parameter id | int |
| 15. | Attribute | Name | Parameter name | string |
| 16. | Attribute | Value | Weights used  in prediction function | int |

# User Interface Diagrams

The user interface is a very simple plain layout. It will display information very clearly for the user and will primarily output information to the user through HTML pages. Administrative screens are used mainly for input through text fields in HTML pages.

* The GUI is in the form of a webpage which will adapt to the user’s display

resolution automatically using responsive design in bootstrap.

* The user interface has a form to input different testing parameter values and

generate rank and a rating based on the information provided.

* The user or the employee has a form to enter values to be stored in the

respective database.

* The predictor tool administrator has an interface to modify the parameters

used for evaluating the rank and rating.

### 5.1 Admin Screens

· Start-up

o Login screen with username and password

o Signup form with company name, employee id

· Dashboard

o To update test parameters values

o To grant and revoke permissions of the users.

o View history of test results of the previous automobiles.

o Creating groups of different departments and phases in testing.

o To delete existing test parameters.

o To add new test parameters.

o Apply filters to output.

o Decide rank formula.

o Modify account details

o Forgot password with security questions.

o Provide feedback to tool developer.

· Shut-down

o Logout confirmation screen.

o Delete Account

### 5.2 User Screens

* Start up

o Login screen with username and password

o Signup form with company name, employee id, designation.

* Dashboard

o Input fields to enter the test parameter values.

o View history of test results of the previous automobiles.

o Delete existing test results.

o Compare test results of the predictor tool with other automobiles within products of the same company.

o Request for changes to the administrator.

o Modify account details

o Forgot password with security questions.

o Provide feedback to tool developer.

* Shut-down

o Logout confirmation screen.

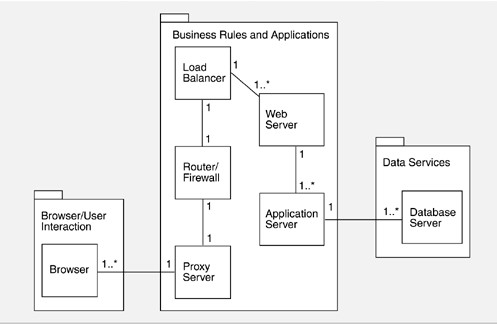
o Delete Account

# Report Layouts

The report consists of sorted ranks of vehicles based on the rating calculated using the predictor tool. Grouping and sorting is done based on several filters such as cost, time, type of automobile etc.The database module is used to generate the report.

# External Interfaces

* From browser to proxy server HTTP protocol is used for communication.
* The proxy server to router/firewall, from router/firewall to load balancer, from load balancer to web server, and from web server to application server messaging and queuing is used to communicate between systems and services.
* Communication from application server to database server uses port 3306 is the default port for the **MySQL Protocol**, which is used by the **mysql** client, **MySQL** Connectors, and utilities such as mysqldump and mysqlpump.



# Packaging and Deployment Diagrams



# Help

* By providing contact information, FAQ, User-manual for interface and Technical-manual for functionality of the product, for the customer to give details on the different test parameters that our tools considers and how to enter them.
* Technical-manual includes instructions for operating the predictor module such as entering test result values, updating weights for each parameter, generating graphical reports to compare different vehicles and their scores.
* The FAQ-Frequently Asked Questions and Common Questions are mentioned in this section.
* The contact information section contains phone number, email-id and address of the headquarters to share feedback and register complaints.

# Alternate Design Approach

Our 3 tier architecture is similar to the layered architecture. Where the database layer is at the core and application server layer and the client layer work on top of it in that order. 3-tier architecture is also similar to the client server architecture, or it is a specialization of client server architecture.

We have not preferred the pipe-filter architecture as the top level architecture since, in the broader scope, our requirements do not hint at only the data processing/manipulating aspects of ATS. There are many quality requirements which are crucial for the success of the project. We may have pipe-filter like architecture for some of the embedded modules in the application server layer.

Peer to Peer design would be preferred when there are multiple branches at different locations, want to work simultaneously and communicate only between there company. In this updating the content is difficult, and storing history, and maintainability is more cost driven.

A monolithic architecture is comfortable for small teams to work with. Сomponents of monolithic software are interconnected and interdependent, which helps the software be self-contained. Codebase gets cumbersome over time, Difficult to adopt new technologies, and Limited agility.

SOA-service-oriented architecture has two main roles: a service provider and a service consumer. Complex management, Extra overload and High investment costs.

Microservice is a type of service-oriented software architecture that focuses on building a series of autonomous components that make up an app. The microservices approach focuses mainly on business priorities and capabilities. Complexity, Security concerns, and Different programming languages.

# Reusability Considerations

Use the User class component for both New user and existing user class.

The login Module can be reused for different users which will direct them to their respective page.

# Traceability Matrix

|  |  |
| --- | --- |
| **CRS Reference Section No.**  **and Name.** | **DESIGN / HLD Reference Section No.**  **and Name.** |
| #RMFR1 | section: predict(),sortOutput() |
| #RMFR2 | section: changeFormula() |
| #LMFR1, LMFR2, LMFR3 | section: login() |
| #LMFR4 | section: authorizeUser() |
| #LMFR5 | section: signup() |
| #LMFR6 | section: checkPassword() |
| #LMFR7 | section: blockUser(), authorizeUser(), confirmUserChanges() |
| #DMFR1 | section: |
| #DMFR2 | section: |
| #DMFR3 | section: |
| #DMFR4 | section: |