DVT DRIVESTATS

Comprehensive Documentation

Axel Ind: 12063178

Nick Robinson: 12026442

Zander Boshoff: 12035671

https://github.com/AxelInd/COS301_DriveStats/

AUGUST 15, 2021

DVT

Version 1.3

Department of Computer Science. University of Pretoria

Contents

Vision and Scope 2
Vision 2
Scope 2
Access and Integration Requirements 4
Access Channels 4
Human Access Channels 4
System Access Channels 4
Integration Channels 4
Architectural Responsibilities 4
Quality Requirements 5
Scalability 5
Performance Requirements 5
Response Time 5
Workload 5
Platform 5
Maintainability 6
Reliability and Availability 6
Security 6
Monitor-ability and Auditability 6
Testability 7
Usability 7
Integratablity 7
Architectural Constraints 7
Build Details 7
High Level Class Diagrams 8
Client Side Android Functionality 8
Server Side Functionality 9
Database Diagram 10
Use Cases 11
Critical 11
userRegistration 11
userLogin 13
TripMonitorState 15
Important 17
DisplayTripInformation 17
Nice-To-Have 19
viewComparedResults 19

Vision and Scope

The following are extracts provided directly by the customer for the Drivestats application (DVT). These are neither modified nor abstracted and are presented directly as specified (DVT, 2015).

Vision

"Many people believe that they are outstanding drivers. Studies have shown that the majority of people believe that their level of safety while driving is above average - a statistical impossibility. [Svenson, 1981] To overcome this bias, an objective measure of driving safety for company vehicles is needed.

DVT would like to procure a mobile app that has the capability of objectively measuring the level of safety of a trip taken by a driver. The application will be used by fleet managers, car rental companies and by insurance companies to ensure that the vehicles belonging to or managed by the company are not being driven recklessly or irresponsibly.

The app should the sensors embedded in the mobile device, such as the GPS and accelerometer to determine different measures of driving safety, such as:

- *Speed of the vehicle relative to the speed limit*
- *Cornering speed*
- *Braking and acceleration forces*
- *Overall smoothness of the drive, measuring the number of speedbumps and/or potholes encountered*

The app should be able to calculate an overall safety rating ("score") for each trip measured. The rating, a score out of 10 with one decimal place, should incorporate all of the above factors, weighted according to a formula which will determine an objective metric of safety of the driving session."

Scope

"The project will consist of 3 main components:

- *Mobile Application*
- *REST service API*
- *Web administrative interface*

The requirement is to create a system that will, through the use of sensors within smartphone devices, be able to measure and report on the safety level of the driving activity in a vehicle.

Attention should be dedicated to designing an algorithm which will incorporate as input the data from all of the sensors identified. The output of the algorithm will be the overall driving score for an individual trip.

A driver-specific score for an individual user of the app should be calculated by taking the average of all the trip scores for that driver."

Architectural Diagram

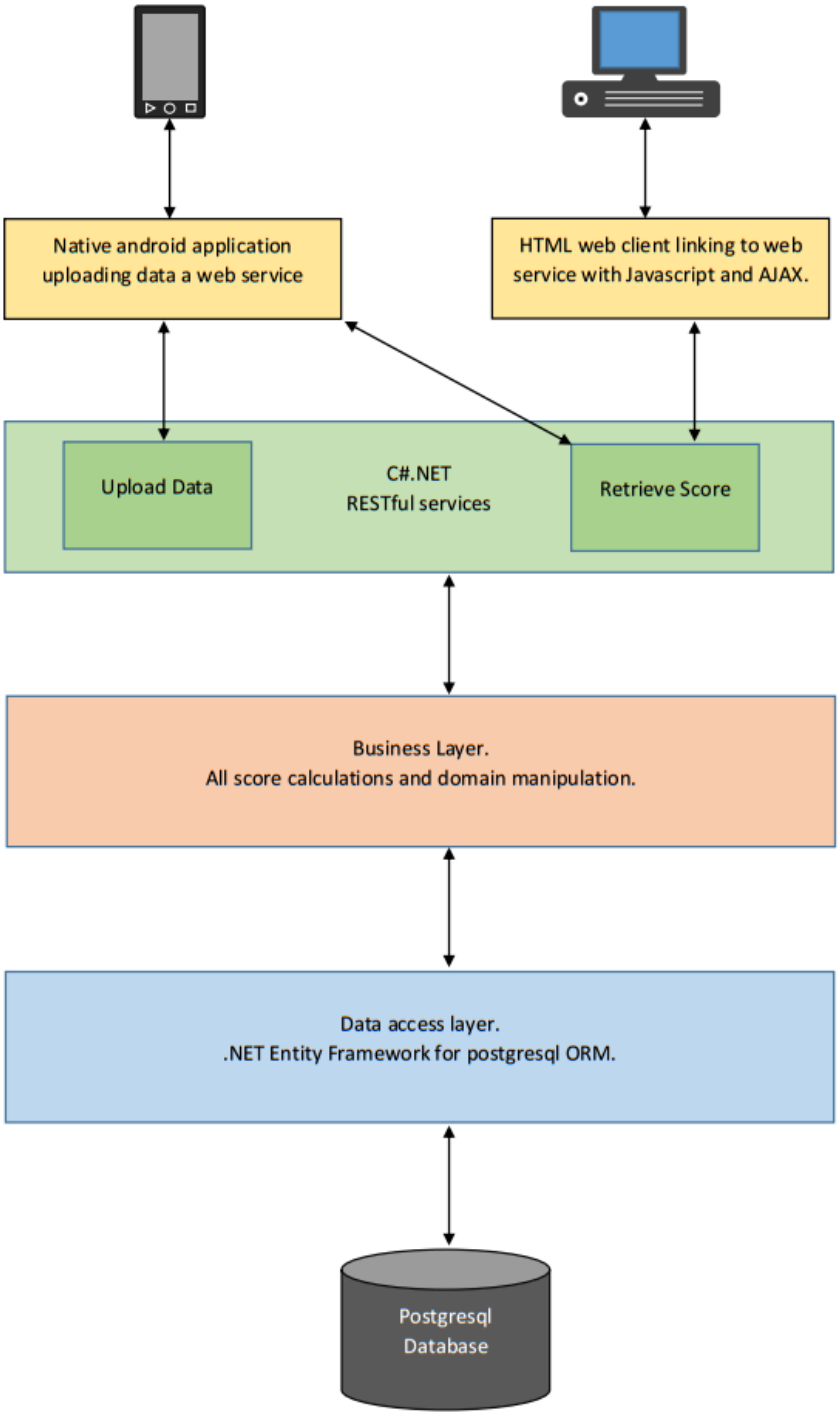

Figure 1 - Diagrammatic representation of the architectural layout of the DriveStats application.

Access and Integration Requirements

Access Channels

Human Access Channels

The Drivestats system user human access component must be an app for Android devices and must make use of Android sensing capabilities including but not limited to GPS and accelerometer. The Drivestats app must be made available for user download via the Android app store.

The Drivestats administrator functions must be made available via the Azure server to authorised users via a computer running Linux or Windows.

System Access Channels

Server-side operations will be cloud based and make use of Microsoft Azure. Client side operations will be done either within the app or the public API's relating to the accessing of sensors.

Integration Channels

The key integration requirement for Drivestats is the ability to efficiently and effectively make use of Microsoft Azure as the cloud component of the service.

Architectural Responsibilities

@ToBeAdded

Quality Requirements

Scalability

The Drivestats system should be able support 300 concurrent users initially. The system has the potential to be used by over 10000 clients simultaneously and the potential for even growth both locally and internationally. While initial testing will almost certainly be done using far fewer clients, scalability is an important requirement.

It is required that, as the customer base grows, the system will need to serve more users simultaneously. This will cause an increase in networking requirements, will require extensive analytics and storage capabilities. It is envisioned that, with increasing demand, additional services from the Drivestats system will be required to accommodate an evolving business model.

It is a requirement that the Drivestats product will make use of Azure Server, Microsoft's cloud computing platform (Microsoft, 2015), to support scalability requirements. Azure Server is currently in use by over half of Fortune 500 companies which ensures Microsoft's continued investment in its scalability aspects.

Scalability of Azure server represents one of its main selling points. Use of this product supports scalability in terms of services, analytics, storage, networking and growth in number of subscribers. In addition, Azure is the only major cloud provider ranked as a cloud storage industry leader by Gartner (Trent, 2014): as strong indicator that all Drivestats scalability requirements will be adequately met now and in the future.

Performance Requirements

Response Time

All operations within the application should respond within 1 second for client-side operations. In terms of client-server interactions, data request-response operations should take fewer than 2 seconds to complete. It is understood that the speed of operations across different mobile data-transfer communication standards can differ significantly. The delays inherent in using slower communication standards cannot be circumvented by the designers in any significant manner, but must be considered in result examination.

Workload

The system must be capable of handling the workload generated by 300 concurrent users without significant drop in response time. The use of multi-threading and a number of optimising algorithms for calculations must be supported. The calculations which require workload tolerance consideration include but are not limited to statistics generation and GPS co-ordinate logging.

Platform

The Drivestats application will run on Android 4.2 or newer technology. Drivestats requires the use of several sensors available only in newer Android enable devices such as GPS and accelerometer features. It is required that Drivestats server side should be a cloud based application.

Maintainability

Maintainability requirements for Drivestats include:

Defect isolation and correction supported by extensive server side logging of errors, maintainable code, the ability to provide downloadable updates to the Android devices once corrections have been made. Loose coupling of calculation methods and attributes is required to allow for dynamic, repairable and extensible code. Version control via GitHub will facilitate easy rollback and software releases as and when they become necessary. Continuous product improvement through feedback log analysis is required.

Reliability and Availability

Dynamic error correction of identified errors should be possible, within 4 hours, in all cases where server side faults occur.

Android app errors must notify the user in a timely and informative manner of the nature of uncorrectable errors.

It is requirement that all cases of un-correctable client side non-user errors should be appropriately logged and transmitted to the server for analysis and correction. It is an ethical consideration that permission must be given by the user for automated action logging to occur.

Correcting of app errors must be by managed update-release via the Android play-store.

Security

It is required that the app be password-protected on the user device. All users must be registered to access the application's functionality. Login must occur through the app and be confirmed by the server.

Data deemed sensitive such as password and user information may only be transferred from the client to the server via HTTPS or other secured protocol. The server side application will be a cloud application and thus make use of available Azure secure services. As such the assumption is that Microsoft will provide the necessary security (Kaufman & Venkatapathy, 2010).

Monitor-ability and Auditability

Server-side performance must be monitored via key performance metrics for cloud services in the Azure Management Portal (Boucher, 2014). Metrics to be monitored include CPU usage, network activity, storage usage rates, user registration metrics and predictive measures. It is required that analysis of diagnostics that occur during application operations be monitored and logged.

The Android app must utilise logging and transmission of errors as previously described. App download rates will be monitored by the Google play-store and are available for developer examination as and when required for application improvement.

Testability

It is required that application testing follow a test-plan and that testing results are recorded and monitored. All test results should be of a quantitive nature except in cases where direct usability testing of interface related components is done.

Unit testing must commence during development.

It must be possible to provide simulated testing of sensor related inputs such as GPS and accelerometer features.

Simulation of over 300 simultaneous login and calculation activities is required to ensure that the system meets its robustness and scalability requirements.

Usability

It is required that over 90% of users should be able to achieve any task outlined in the use-cases of this application within 5 minutes of first encountering the application.

It is required that the colour and style dynamics of the application must appeal to a broad audience and must not deter more than 10% of potential customers.

Integratablity

Integratablity of the app with standard Android functionality is required. In particular the functionality to access sensors and GPS related information while the app is not directly open is essential to the correct functioning of the application as a statistics gathering tool.

The Azure server provides integration capability with both Linux and Microsoft windows.

Logging and feedback mechanisms approach must align with the Azure monitoring services.

Architectural Constraints

- The mobile application must run on Android devices (of version 4.2 or higher).
- The use of MVC architecture makes continued access to a reliable mobile internet connection essential.
- Integration of sensors must be accomplished in such a way to adequately function on all devices for which this application was designed.
- Microsoft Azure is expected to provide a stable and reliable basis for the cloud management of server-side operations and database management.
- Device error logging is dependent on the user's permission to upload anonymous data on detected errors. (This is due both to limitations enforced by Google on Android applications and due to ethical considerations). Because the logging is dependent on user-cooperation, data gathered may not represent a representative sample of the user base and the errors it encounters.

Build Details

- Visual Studio 2012 used for C# server-side development
- Entity framework used as ORM
- Github used for version control
- Unit tests created with Test Explorer (built in to Visual Studio 2012)

- Normalisation and algorithm development was created in tandem with Wolfram Alpha online (testing specific code exists in initial distributions)

High Level Class Diagrams

Client Side Android Functionality

This section refers to the mobile application aspect of the DVT DriveStats program. This details the use of activities, external application requests, sensor data aggregation, and their respective management.

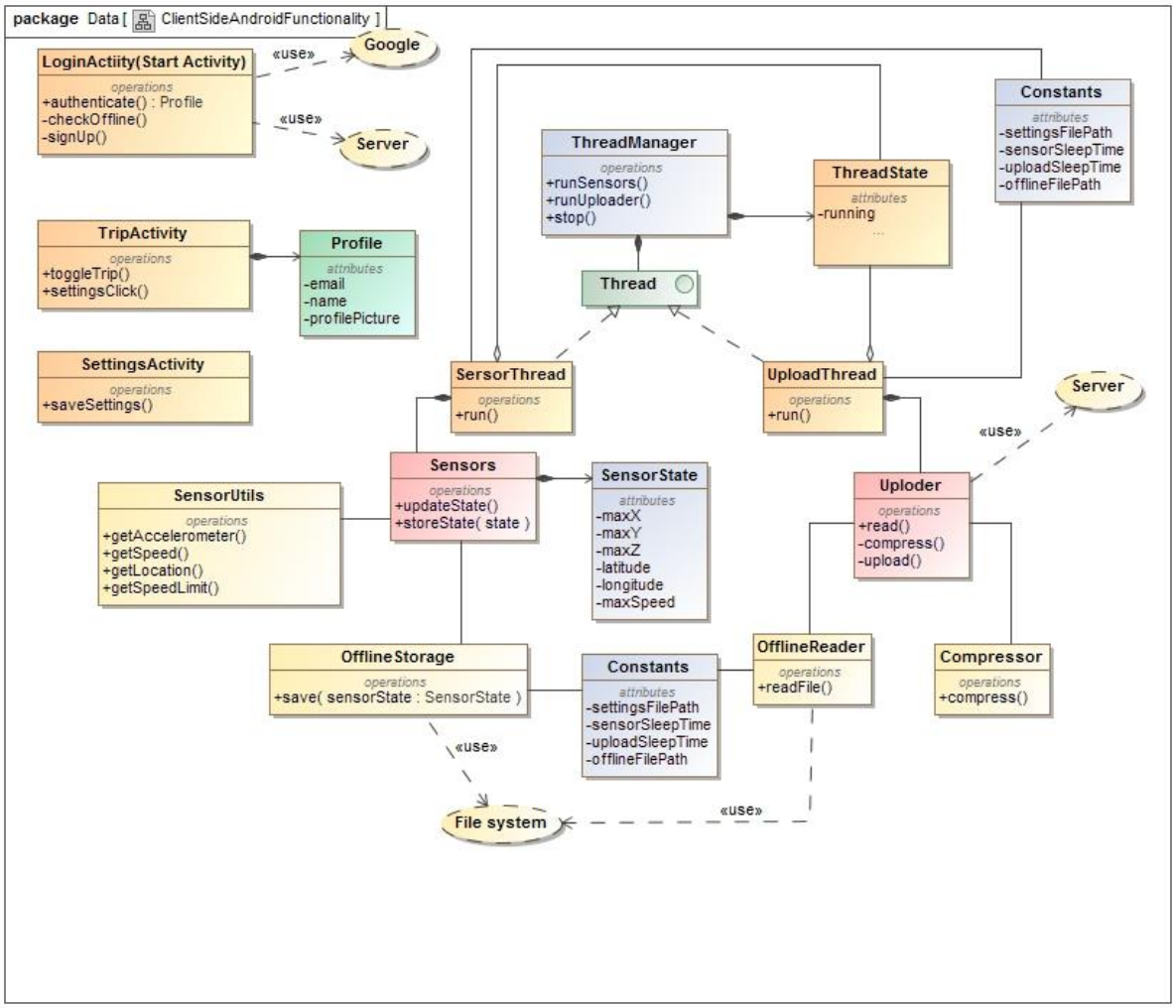

Figure 2- Client-Side Class Diagram

Server Side Functionality

This section refers to the server-side implementation of the database manager. The server is responsible for several tasks, including but not limited to, Object Relations Mapping, Equation Coefficient storage, login-authentication, user registering, and statistical and meta-statistical calculations

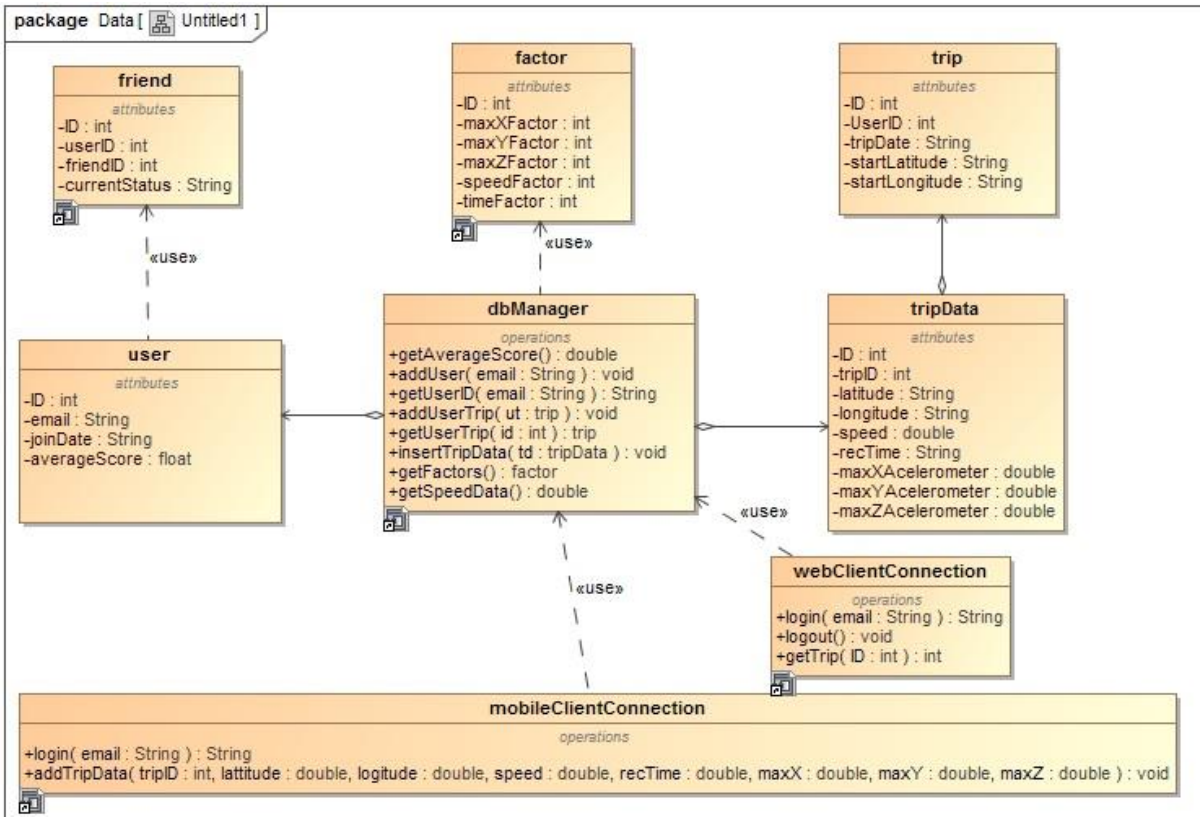

Figure 3- Server-side Class Diagram

Database Diagram

Tables relate to the storage of data, description of metadata, and algorithmic modifiers

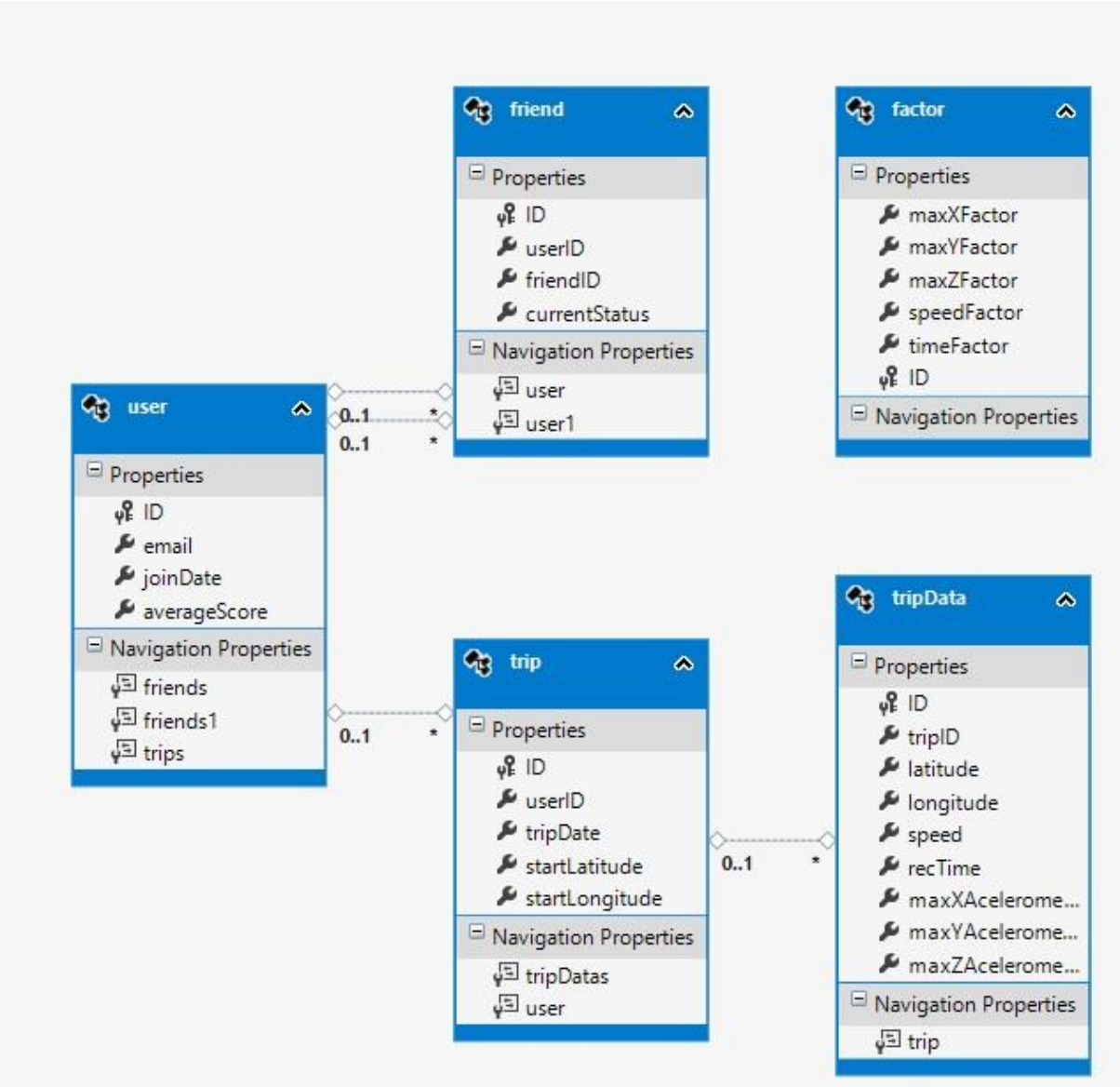

Figure 4- Database Organisation Description

Use Cases

All use cases described here-in are in direct compliance with the initial specification as released by the company, DVT (DVT, 2015). As expounded upon the first meeting with the client.

Critical

userRegistration

Description

This use case will be used by the android client and the web interface to allow new users to save their information in the database.

Use Case

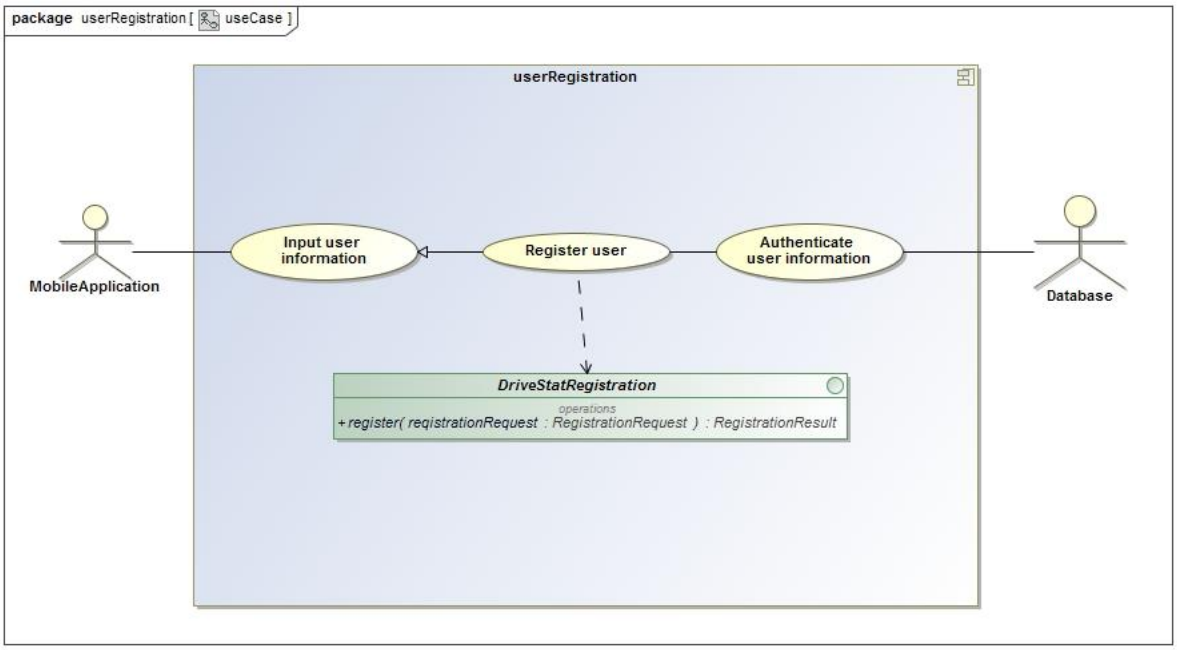

Figure 5- User Registration use case diagram

Service Contract

The Service contract for the userRegistration service is shown in Figure x. This is a simple database element creation service.

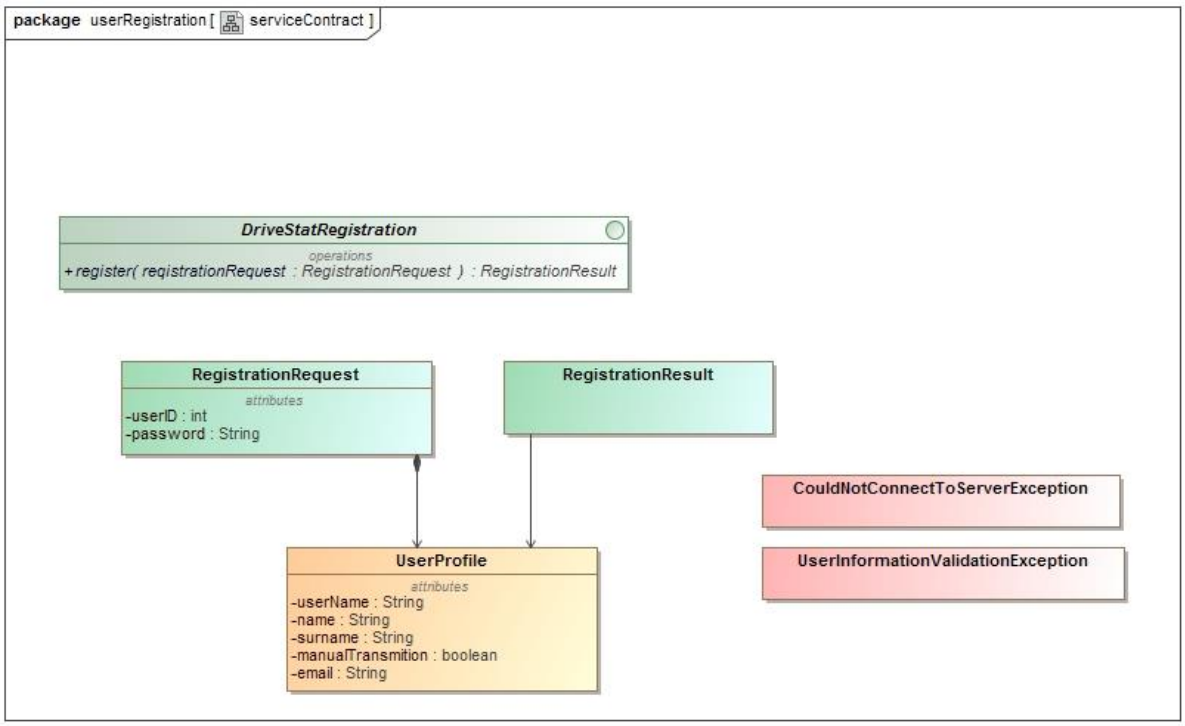

Figure 6- User Registration service contract

Process Specification

The process specification contract for the userRegistration service is shown in Figure x. This is a simple database element creation specification.

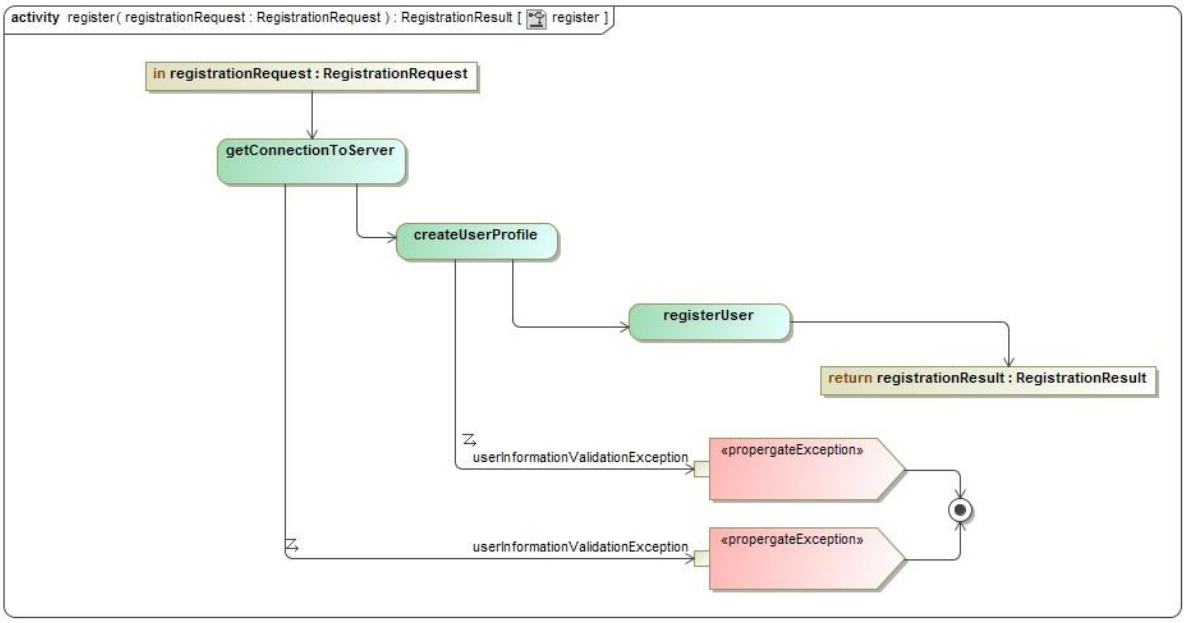

Figure 7 User Registration process specification

userLogin

Description

This use case will be used by the android client to initiate login, via the server, for use on the client-side Android application. This use case extends to direct login of a system admin for server manipulation.

Use case

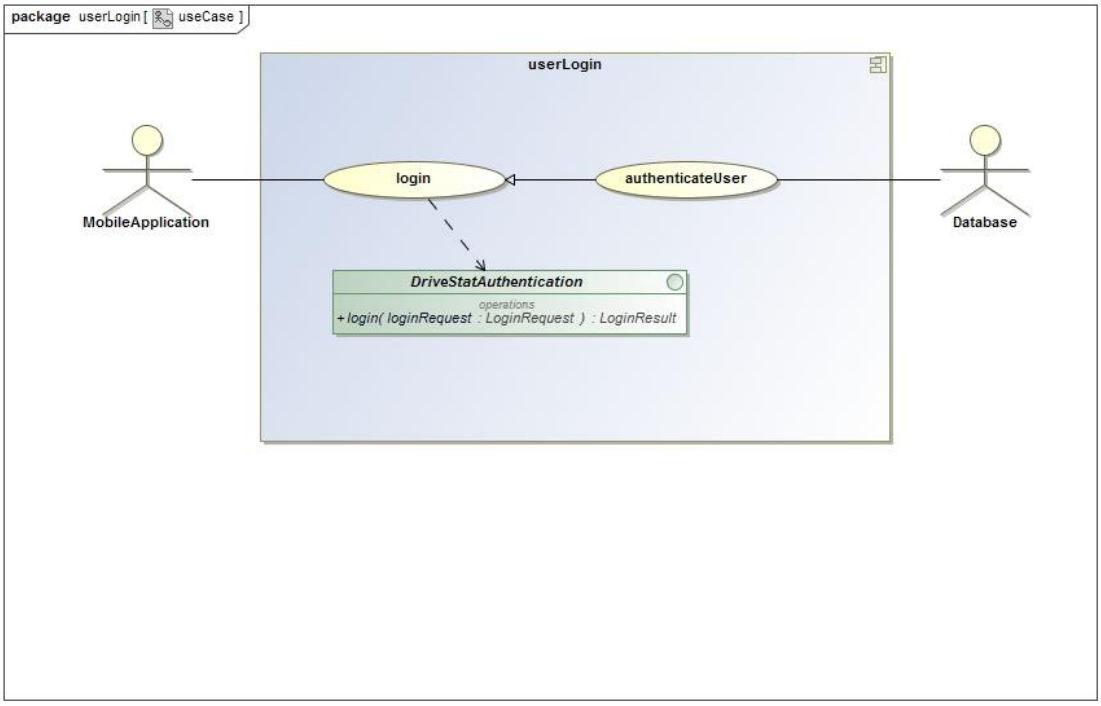

Figure 8 - User Login use case

Service Contract

This service contract outline the process used by the android client to initiate login, via the server, for use on the client-side Android application. This contract extends to description of the direct login of a system admin for server manipulation.

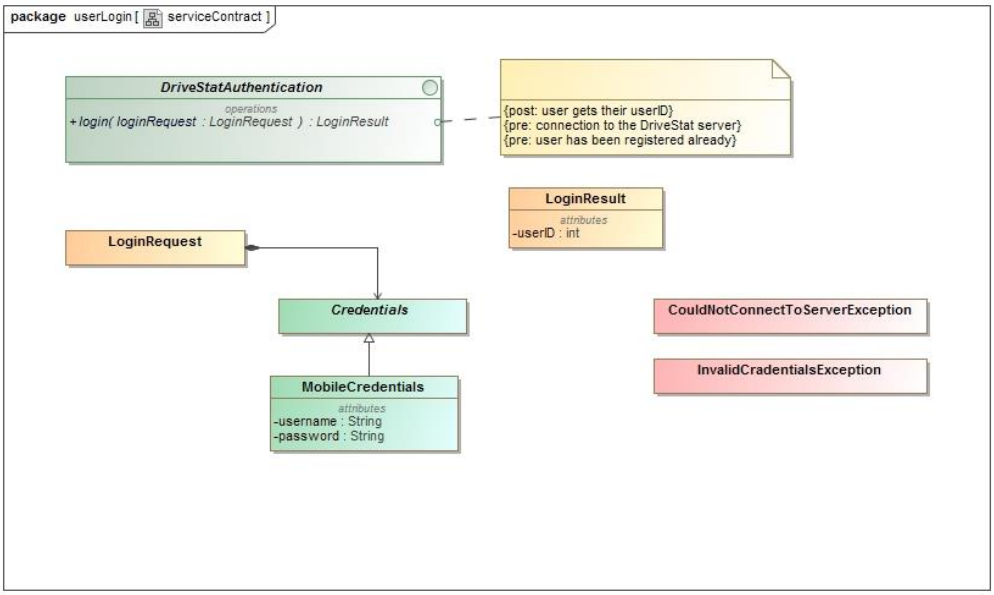

Figure 9- User Login service contract

Process Specification

This process specification outline the process used by the Android client to initiate login, via the server, for use on the client-side Android application. This specification extends to process description of the direct login of a system admin for server manipulation.

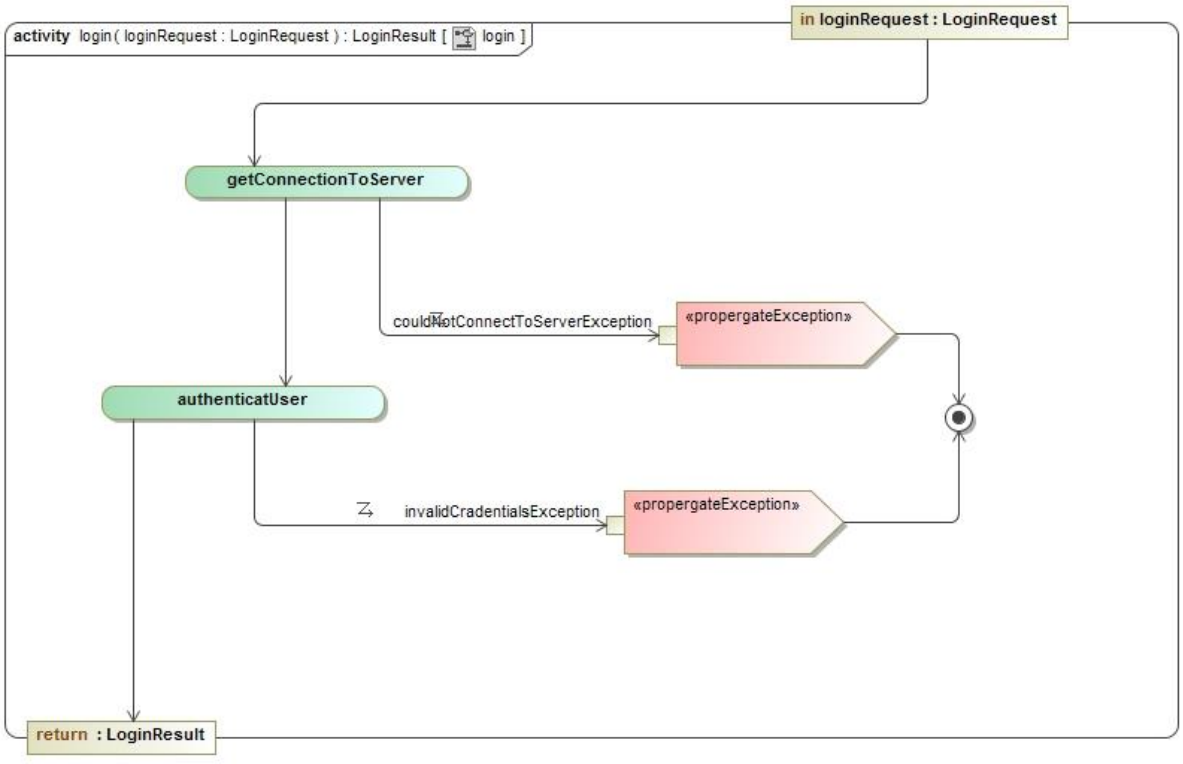

Figure 10 - User Login process specification

TripMonitorState

Description

This use case will be used by the user to activate and deactivate the monitoring of the phones sensors.

Use cases

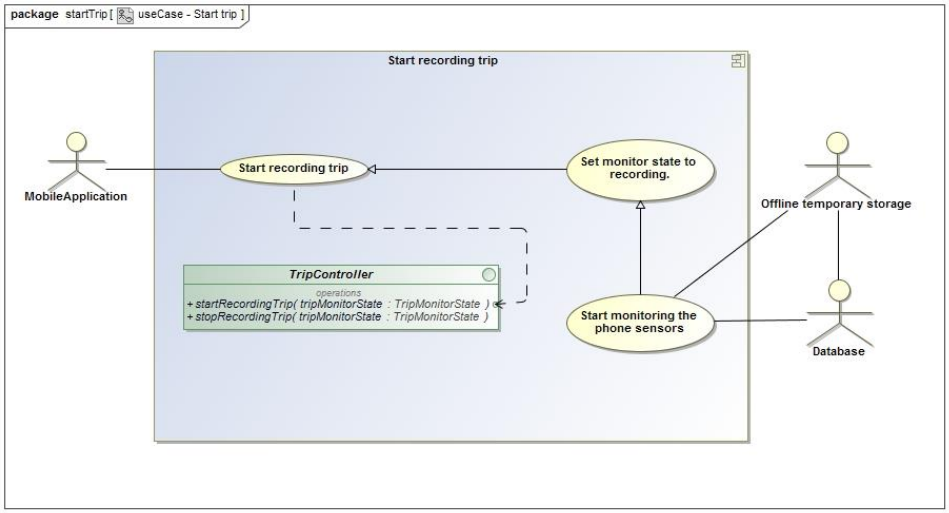

Figure 11 - Start Recording Trip use case

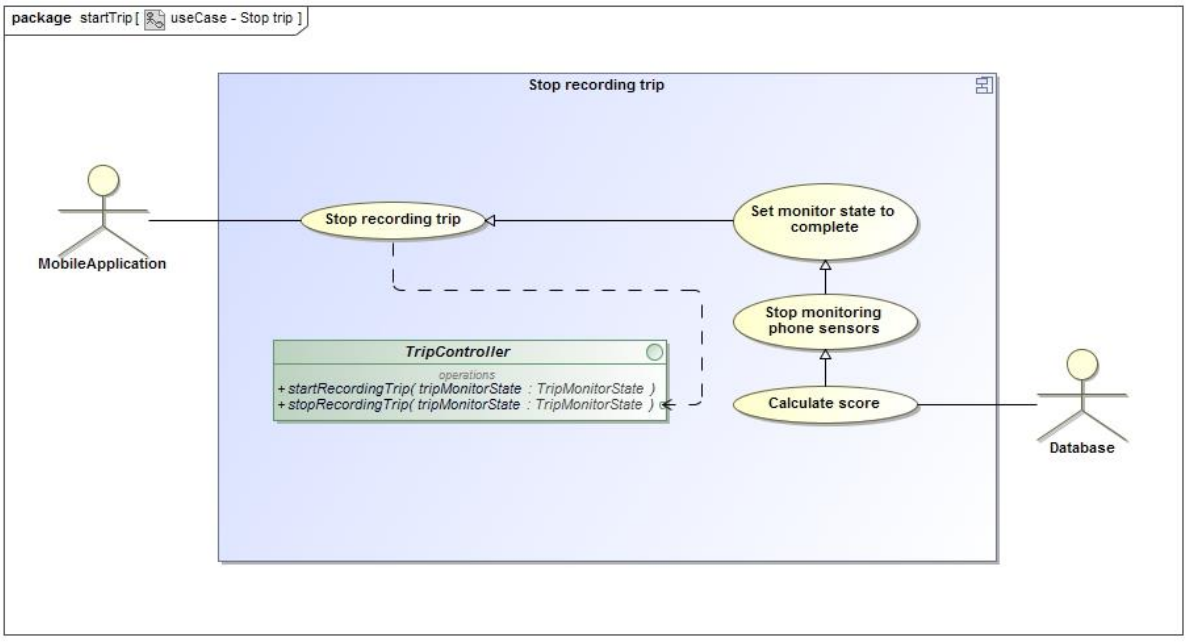

Figure 12 - Stop Recording Trip use case

Service Contract

The Service contract for the Trip Recording service is shown in Figure x. This is a dual functioned service providing sensor monitoring and feedback to the database.

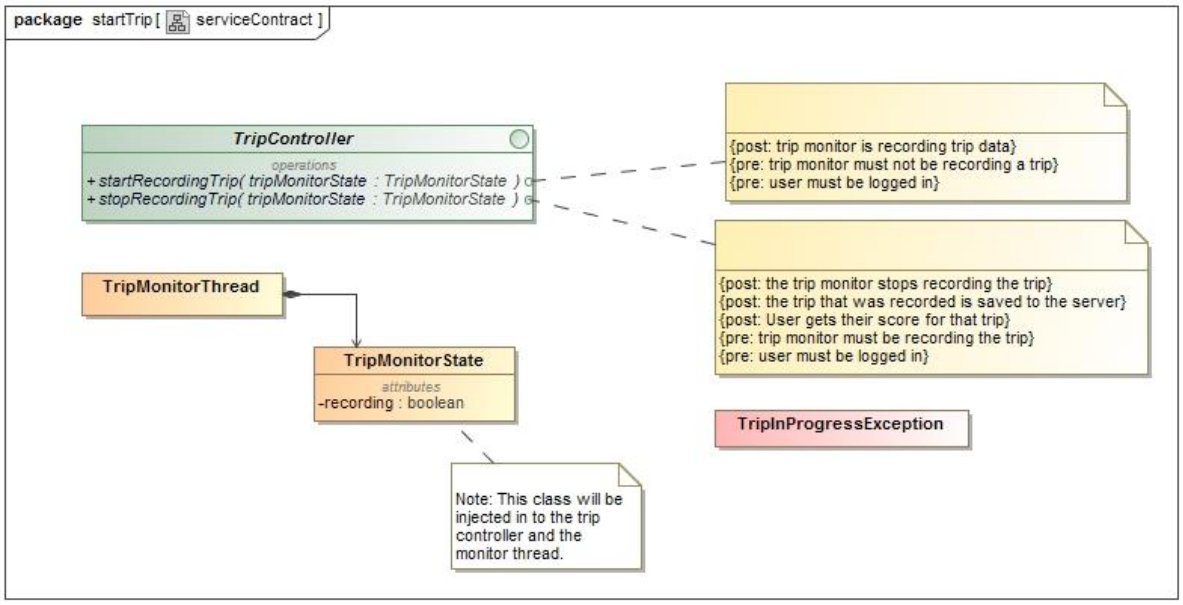

Figure 13 - Start Trip service contract

Process specification

The process specification for the Trip Recording service is shown in Figure x. This is dual-functioned specification outlining sensor monitoring and feedback to the database.

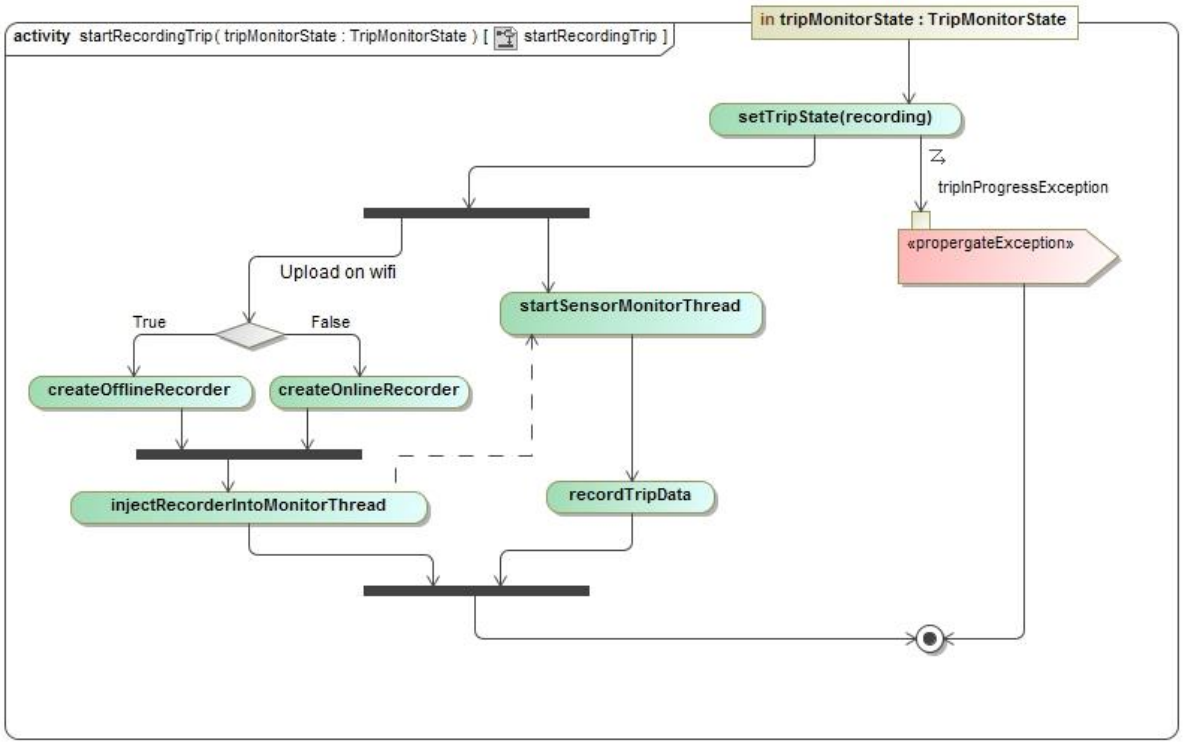

Figure 14 - Start Recording Trip service contract

Important

DisplayTripInformation

Description

This use case will be used by the user to receive a graphical display of the use information from their current trip.

Use case

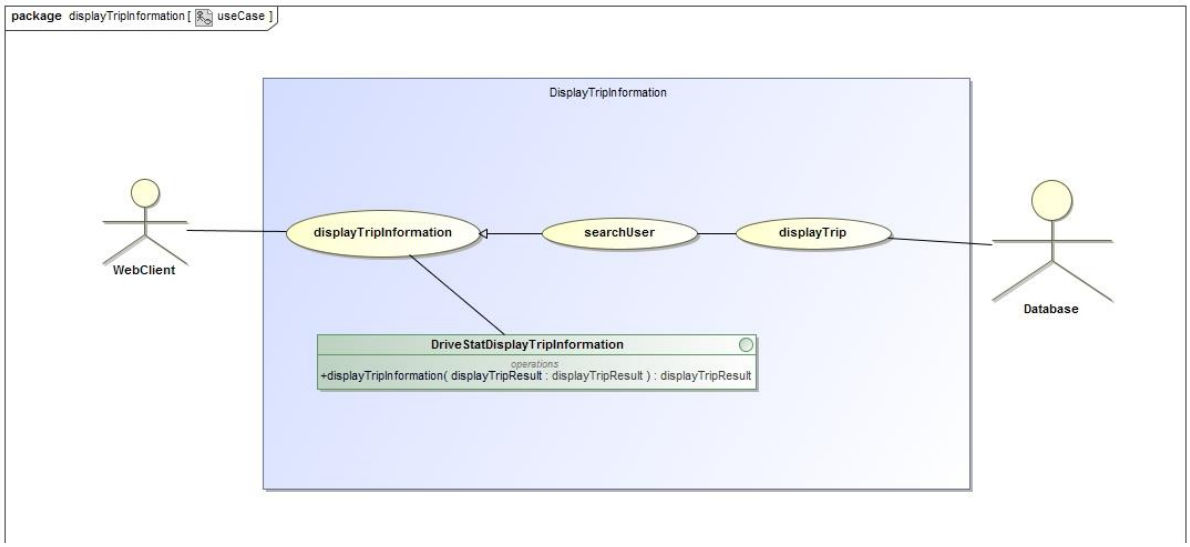

Figure 15 - Display Trip Information use case

Service Contract

This service contract describes the mechanism by which the user will receive a graphical display of the use information from their current trip.

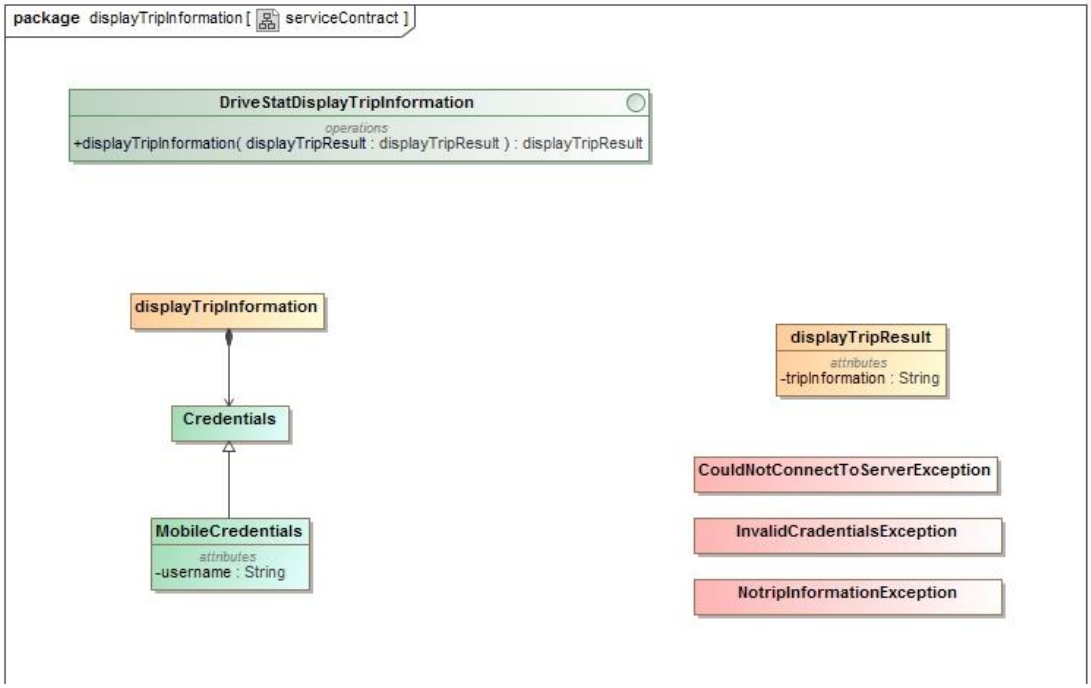

Figure 16 - Display Trip Information service contract

Process Specification

This process specification describes the mechanism by which the user will receive a graphical display of the use information from their current trip.

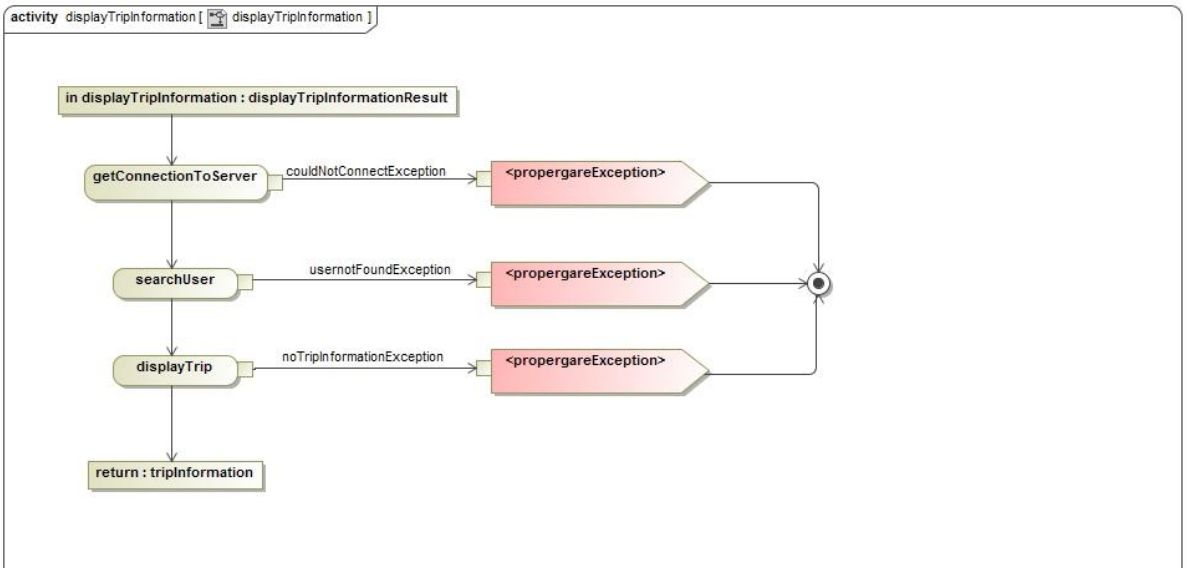

Figure 17 - Display Trip Information process specification

Nice-To-Have

viewComparedResults

Description

This use case describes the mechanism by which the user will be able to compare their trip information against that of their “Friends”.

Use case

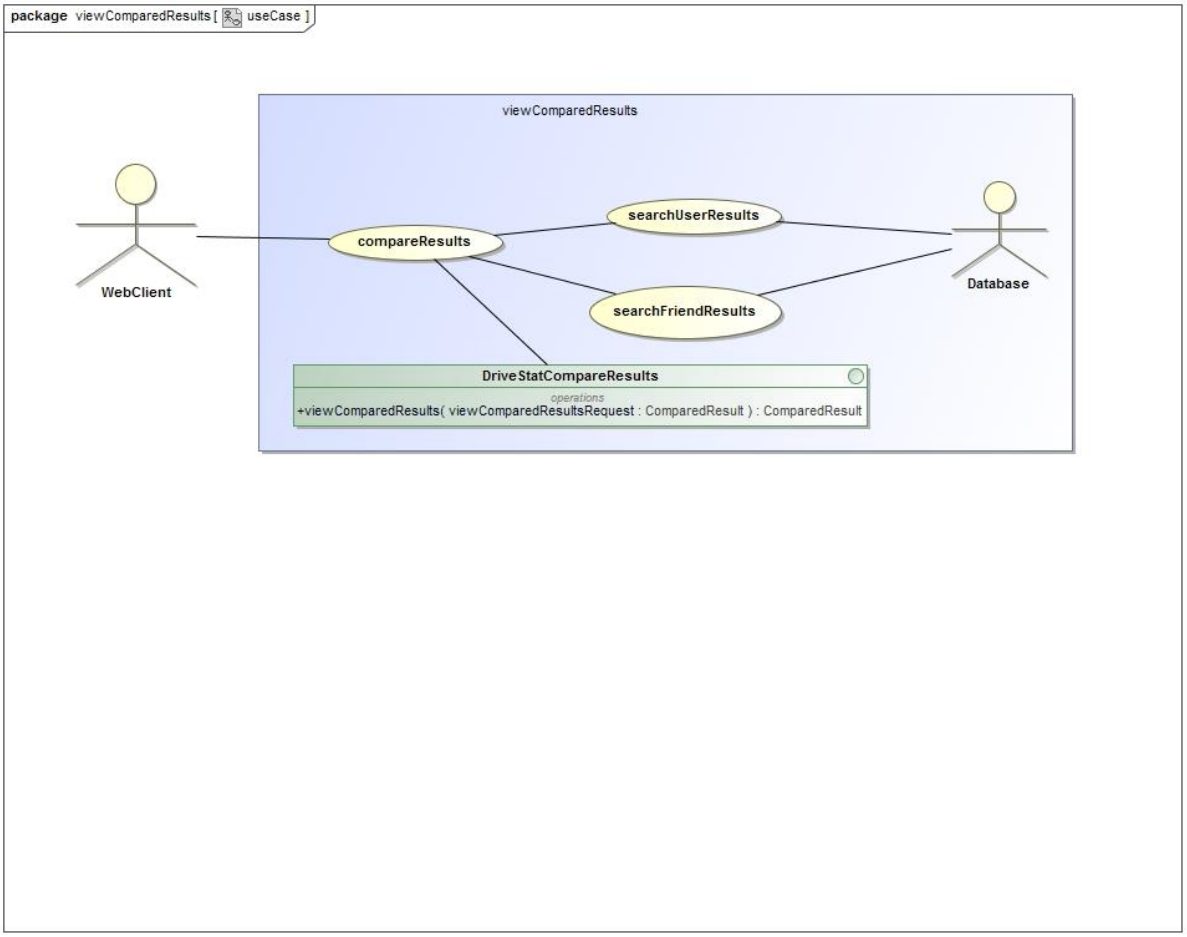

Figure 18 - View Compare Results use case

Service contract

This service contract describes the mechanism by which the user will compare their trip information against that of their “Friends”.

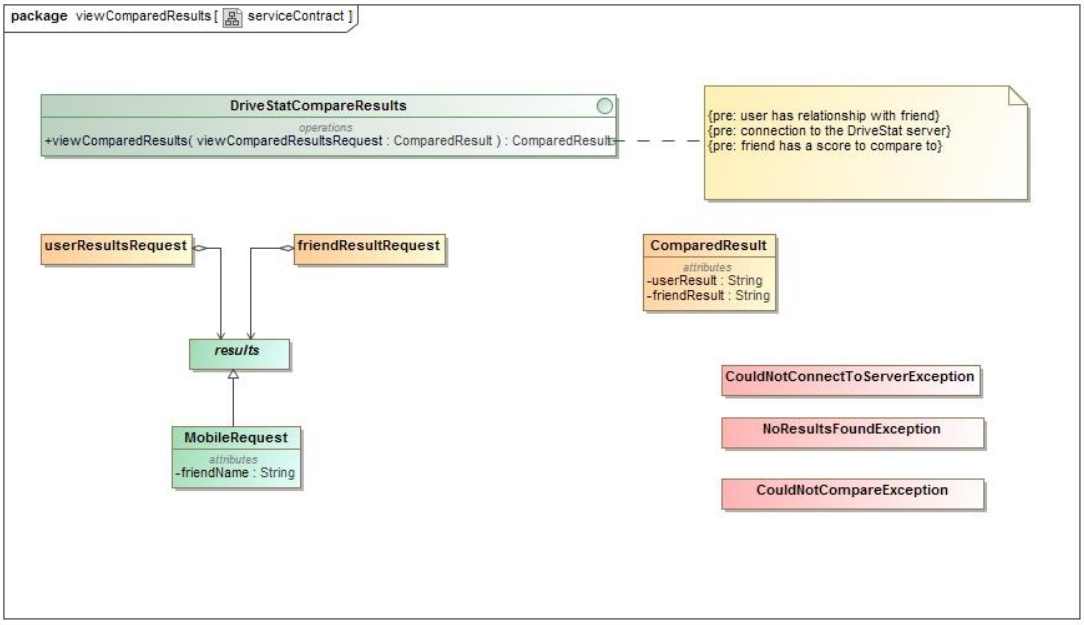

Figure 19 - View Compare Results Service Contract

Process specification

This process specification describes the mechanism by which the user will be able to compare their trip information against that of their “Friends”.

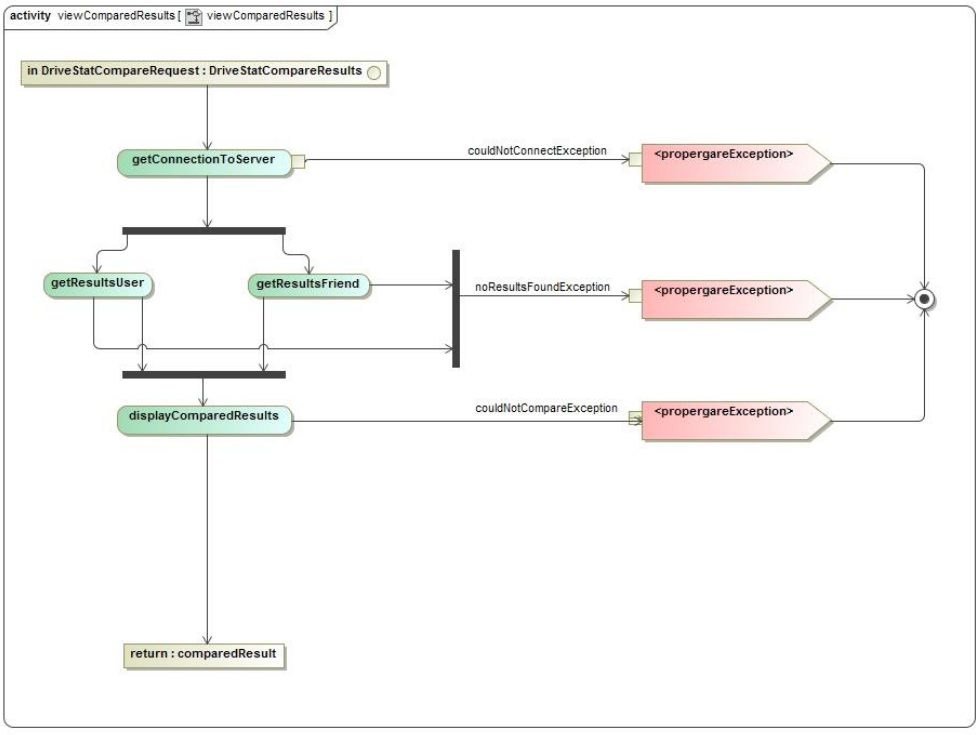

Figure 20 - View Compare Results Process specification