



UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

MAY 10, 2015



TEAM ANTZ



DriveStats

DVT DRIVESTATS ANDROID APPLICATION

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

Axel Ind: 12063178

Nick Robinson: 12026442

Zander Boshoff: 12035671

CLIENT: DVT

Version 2.1

Department of Computer Science. University of Pretoria

Table of Contents

1 Vision and Scope	3
1.1 Project Background	3
1.2 Project Vision	3
1.3 Project Scope	4
1.4 Architectural Diagram	5
2 Use Case Prioritisation.....	Error! Bookmark not defined.
3 Use Cases	7
3.1 Critical	7
3.1.1 userRegistration	7
3.1.2 userLogin	9
3.1.3 TripMonitorState	11
3.2 Important.....	13
3.2.1 DisplayTripInformation	13
3.3 Nice-To-Have.....	15
3.3.1 viewComparedResults	15
4 Domain Model.....	17
4.1 Client Side Android Functionality	17
4.2 Server Side Functionality.....	18
4.3 Database Diagram.....	19
5 Access Integration Channels.....	20
5.1. Access Channels	20
5.1.1 Human Access Channels	20
5.1.2 System Access Channels	20
5.2 Integration Channels.....	20
6 Quality Requirements.....	21
6.1 Scalability	21
6.2 Performance Requirements	21
6.2.1 Response Time	21
6.2.2 Workload	21
6.2.3 Platform.....	21
6.3 Maintainability	22
6.4 Reliability and Availability	22
6.5 Security	22
6.6 Monitor-ability and Auditability	22
6.7 Testability.....	23
6.8 Usability	23
6.8 Integratability	23
7 Architectural Responsibilities	24
8 Architecture Constraints	25
9 Technologies	26
9.1 Build Technologies	26
9.2 Implementation Technologies	26

Table of Figures

Figure 1 - Diagrammatic representation of the architectural layout of the DriveStats application.....	5
Table 2 - Use Case Prioritisation of the DVT DriveStats Application	6
Figure 3 - User Registration use case diagram.....	7
Figure 4 - User Registration service contract.....	8
Figure 5 - User Registration process specification.....	8
Figure 6 - User Login use case.....	9
Figure 7 - User Login service contract.....	9
Figure 8 - User Login process specification.....	10
Figure 9 - Start Recording Trip use case	11
Figure 10 - Stop Recording Trip use case	11
Figure 11 - Start Trip service contract.....	12
Figure 12 - Start Recording Trip service contract	12
Figure 13 - Display Trip Information use case	13
Figure 14 - Display Trip Information service contract	13
Figure 15 - Display Trip Information process specification	14
Figure 16 - View Compare Results use case	15
Figure 17 - View Compare Results Service Contract.....	16
Figure 18 - View Compare Results Process specification	16
Figure 19 - Client-Side Class Diagram	17
Figure 20 - Server-side Class Diagram.....	18
Figure 21 - Database Organisation Description	19

1 Vision and Scope

All quoted information that follows represents an extract of the requirements document provided directly by the customer for the Drivestats application (DVT). These are neither modified nor abstracted and are presented directly as specified (DVT, 2015).

1.1 Project Background

In South Africa over 10000 people die on the roads annually (Road Traffic Management Corporation, 2011). Students at the University of Pretoria (Team Antz) were approached by the company DVT with the intent of addressing this problem.

The problem of reckless driving has become so severe that, in 2011, the South African province of Limpopo has claimed that 32% of all deaths in that year were related to transport and road-safety (Department Of Statistics, 2011). With this in mind, DVT tendered a proposal to the University of Pretoria for the creation of a mobile application to increase road safety across the country.

As a team of enthusiastic and conscientious drivers, we are proud to be part of the design and implementation of the DVT DriveStats app which is designed to raise awareness for the quality of a user's driving.

1.2 Project Vision

The DVT DriveStats application aims to address the issue of road safety through enabling drivers to improve the quality of their driving. We intend to provide an application that will allow users to monitor and improve their driving skills. The app must have 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 use 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 according to a formula.

1.3 Project Scope

The scope of this project is the design and development of a mobile application which will use the built in sensors of Android devices to monitor users driving and report on the safety level of the driving activity in a vehicle. This is achieved via the use of an Android mobile application and communication with a privately hosted Azure Server.

The algorithm designed 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.

The rating should incorporate all of the above factors, weighted according to a formula which will determine an objective metric of safety of the driving session.

The deliverables for the project will be¹:

- *Mobile Application*
- *REST service API*

¹ A third part to the proposed project was initially slated to be the creation of a rich web-interface through which a user could examine their driving history. This was, however, removed from the project scope via agreement with the university of Pretoria and the DVT company.

1.4 Architectural Diagram

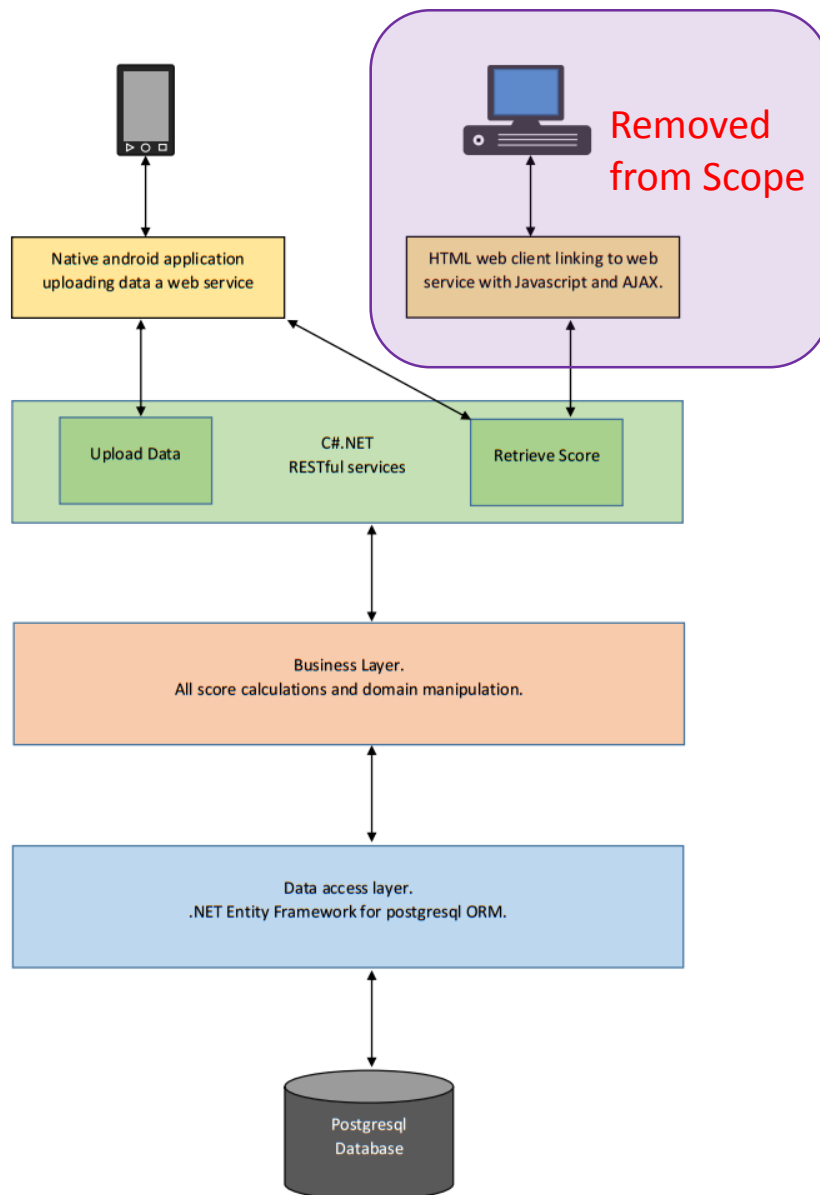


Figure 1 - Diagrammatic representation of the architectural layout of the DriveStats application.

2 Application requirements and design

This section covers the functional requirements as well as the process designs for the use cases and the domain models. The non-functional requirements are covered in the section on Quality Requirements.

2.1 High Level Requirements

Our understanding of the DVT requirements are:

- 1 A mobile app that has the capability of objectively measuring the level of safety of a trip taken by a driver in a motor vehicle*
- 2 The application should be suitable for 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.*
- 3 The app should be able to calculate an overall safety rating ("score") for each trip measured (as a score out of 10 with one decimal place)*
- 4 The app should use the sensors embedded in the mobile device, such as the GPS and accelerometer to determine different measures of driving safety*
- 5 The safety rating should incorporate all of the available sensors, weighted according to a formula which will determine an objective metric (measure) of safety of the driving session*
- 6 Measures required are:*
 - 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*

2.2 Use Case Prioritisation

Critical	Important	Nice-To-Have
userRegistration	DisplayTripInformation	viewComparedResults
userLogin		
TripMonitorState		

Table 2 - Use Case Prioritisation of the DVT DriveStats Application

3 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.

3.1 Critical

3.1.1 userRegistration

3.1.1.1 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.

3.1.1.2 Use Case

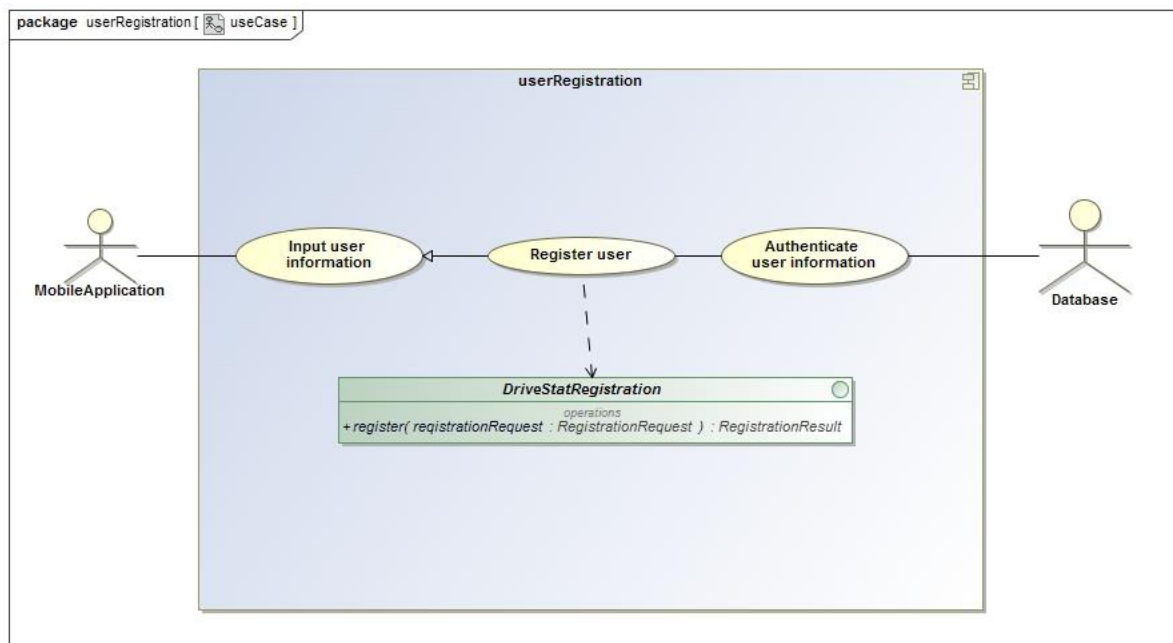


Figure 3- User Registration use case diagram

3.1.1.3 Service Contract

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

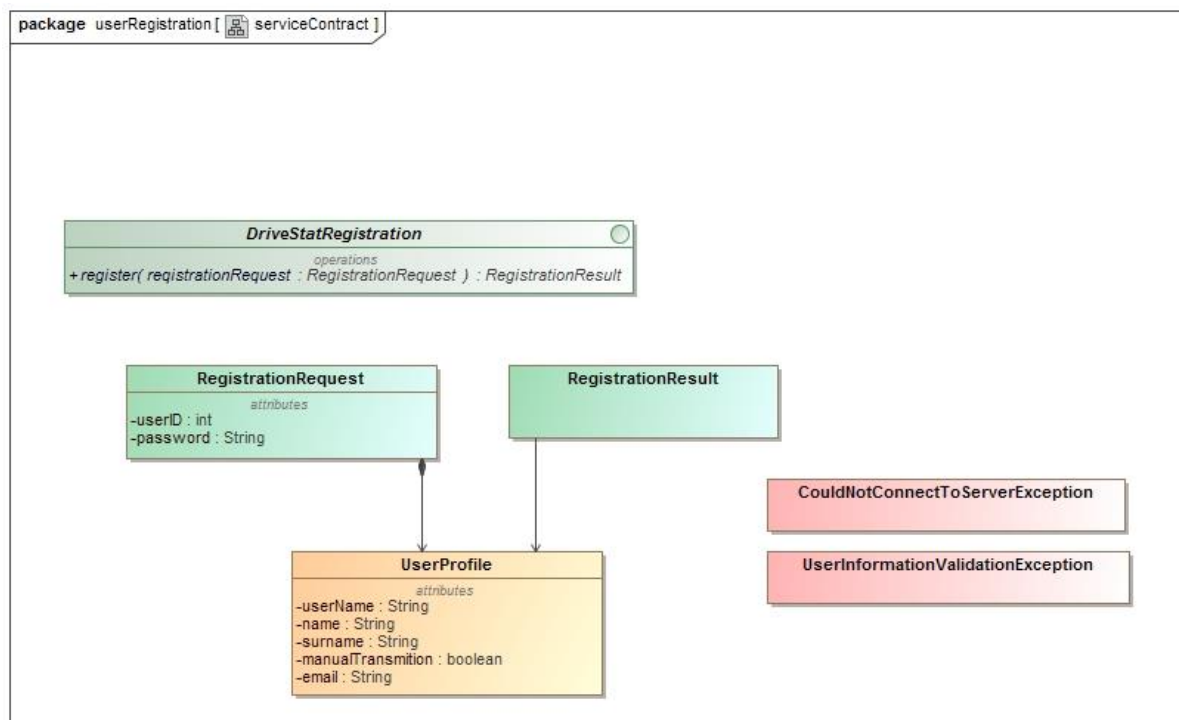


Figure 4- User Registration service contract

3.1.1.4 Process Specification

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

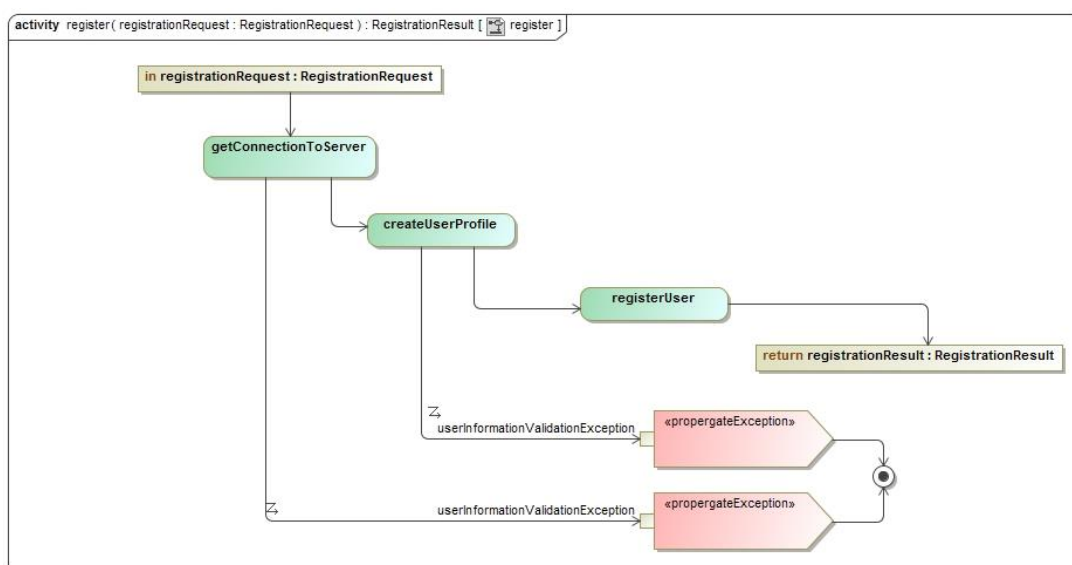


Figure 5 User Registration process specification

3.1.2 userLogin

3.1.2.1 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

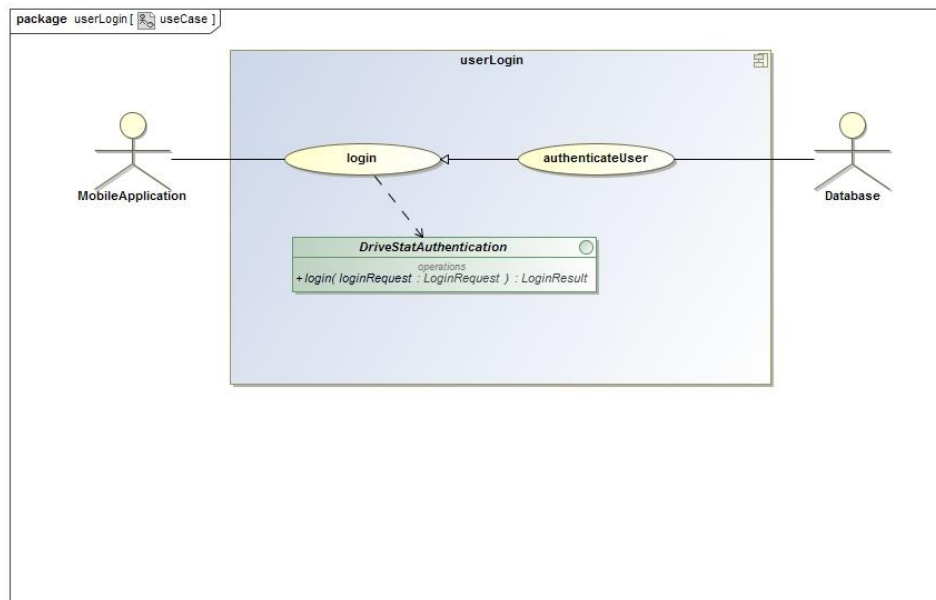


Figure 6 - User Login use case

3.1.2.2 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.

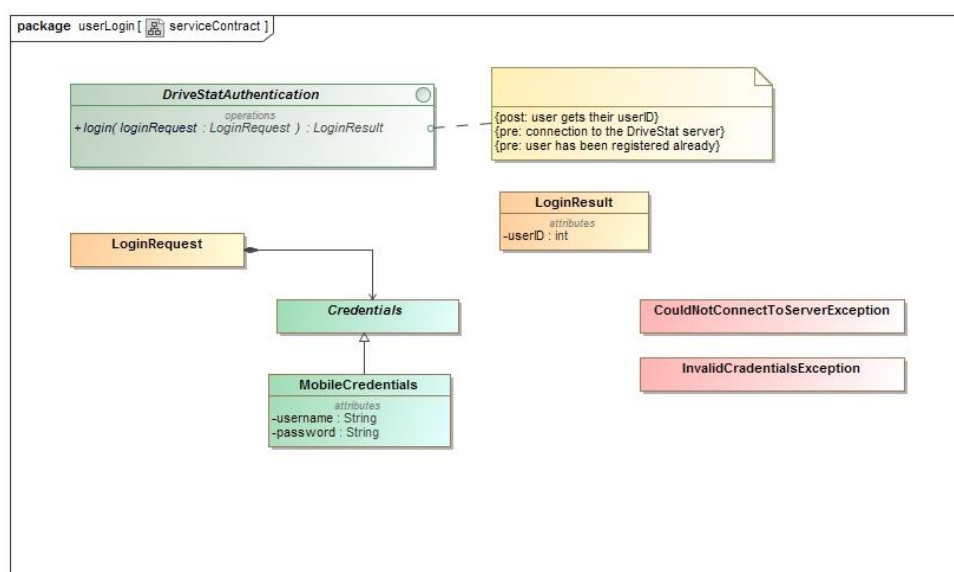


Figure 7- User Login service contract

3.1.2.3 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.

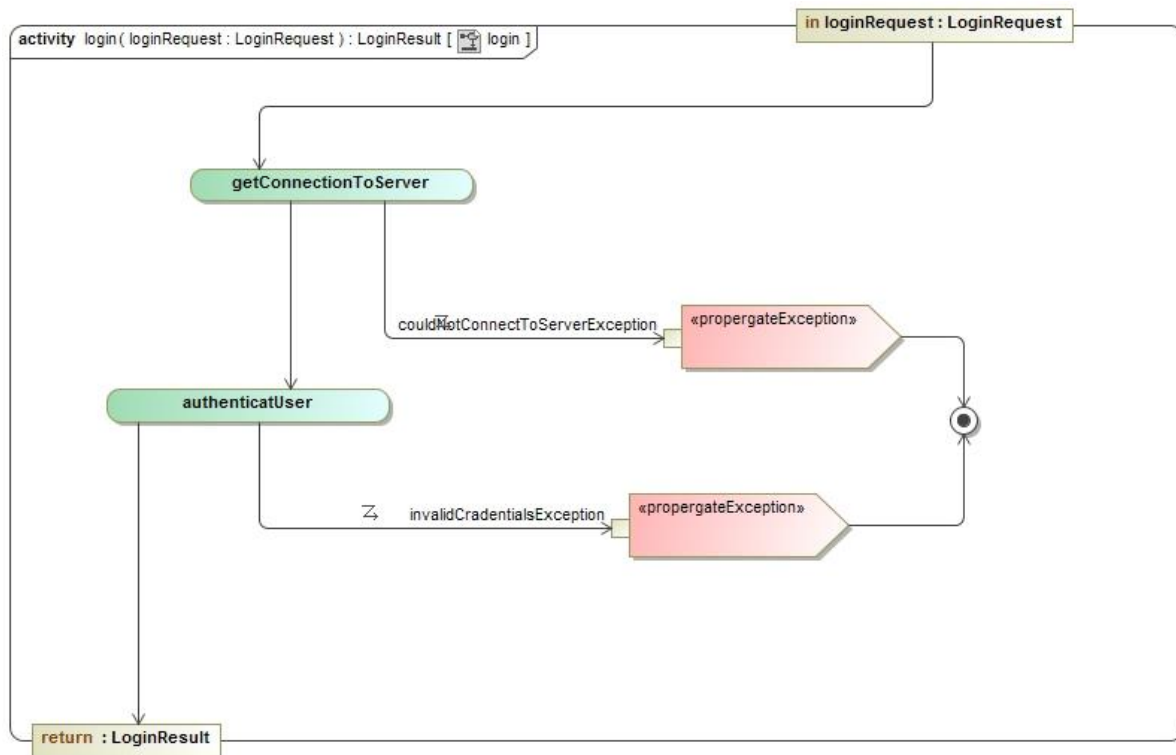


Figure 8 - User Login process specification

3.1.3 TripMonitorState

3.1.3.1 Description

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

3.1.3.2 Use cases

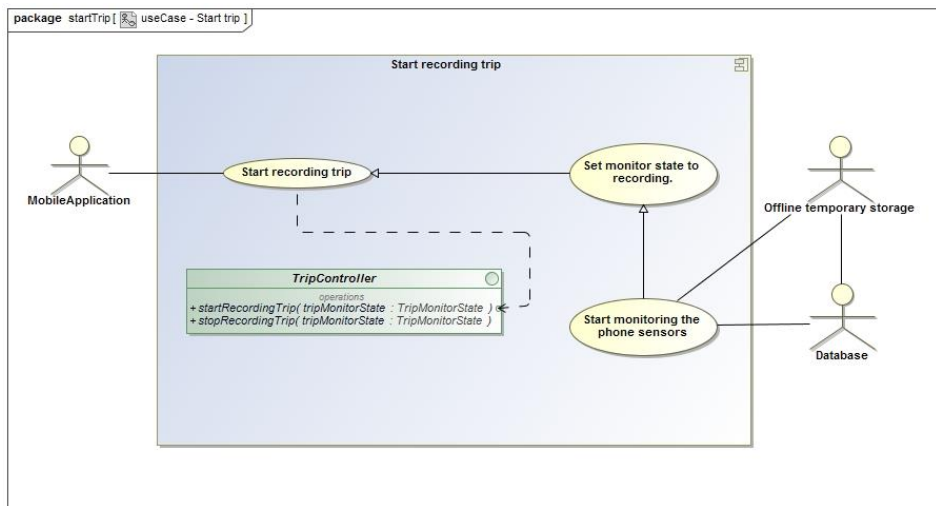


Figure 9 - Start Recording Trip use case

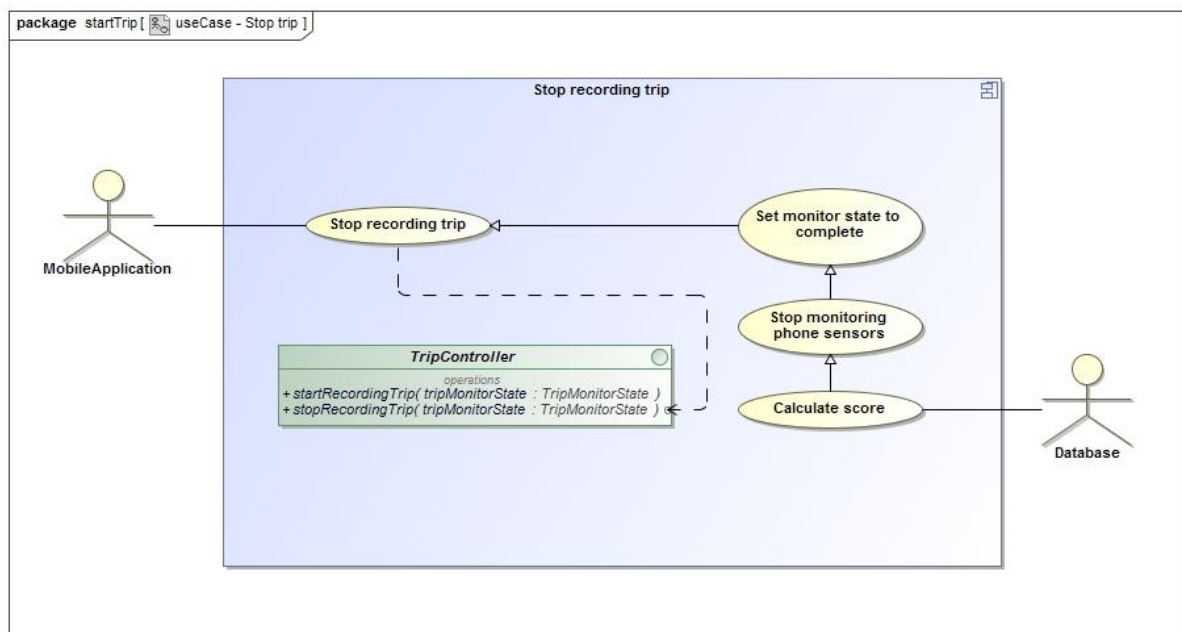


Figure 10 - Stop Recording Trip use case

3.1.3.3 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.

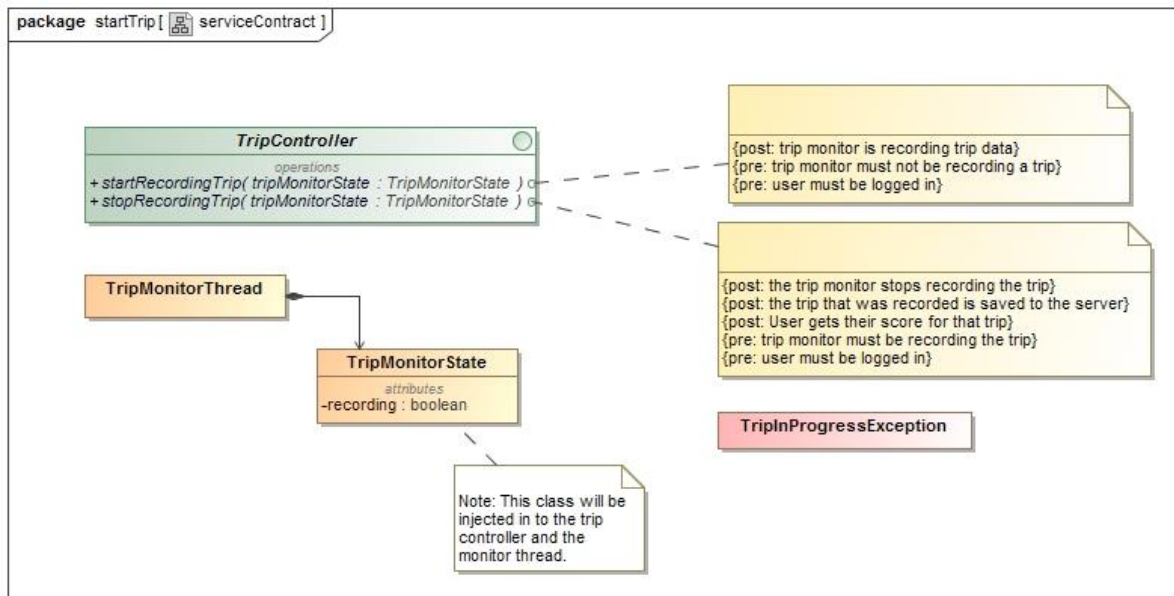


Figure 11 - Start Trip service contract

3.1.3.4 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.

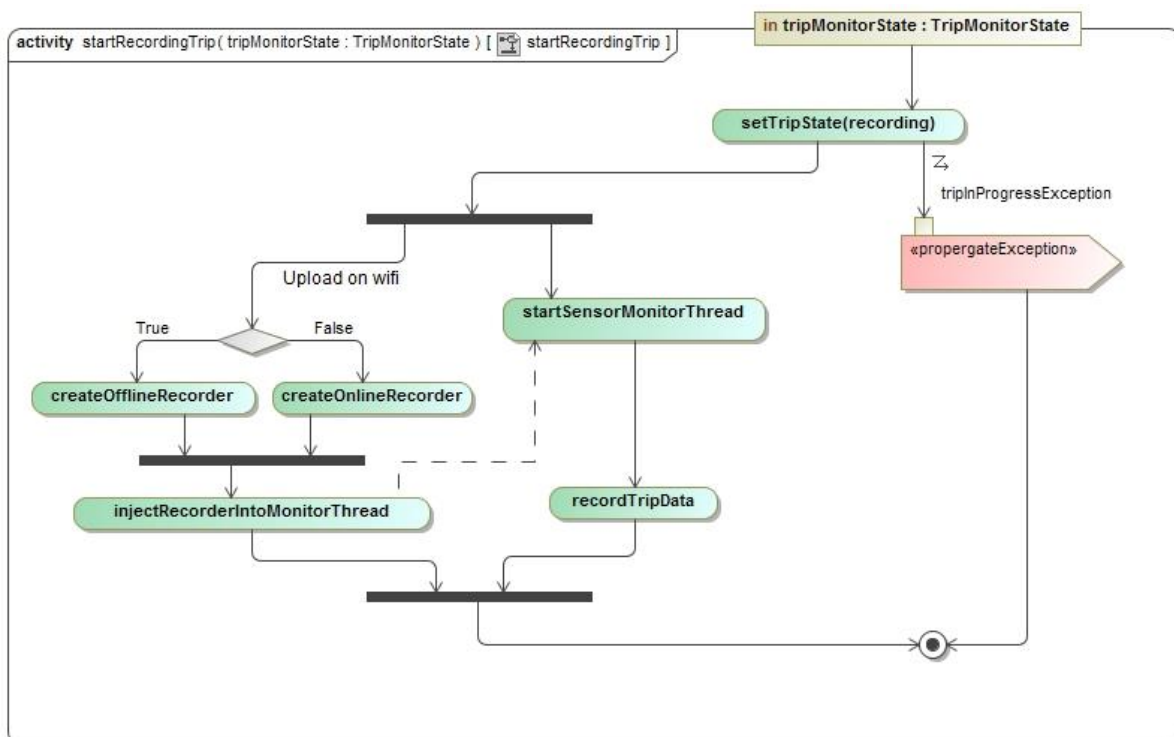


Figure 12 - Start Recording Trip service contract

3.2 Important

3.2.1 DisplayTripInformation

3.2.1.1 Description

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

3.2.1.2 Use case

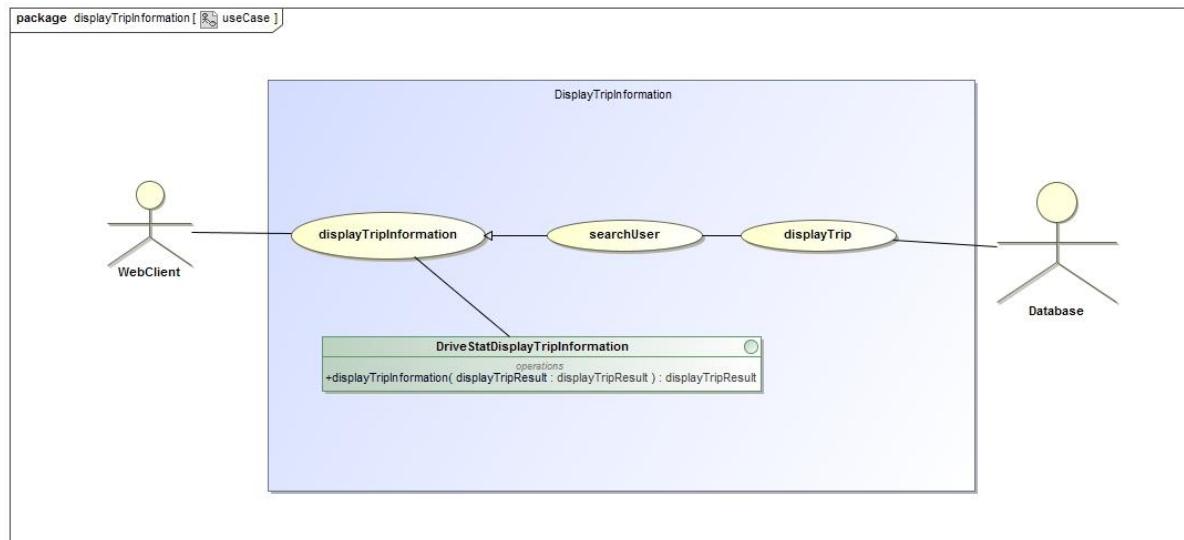


Figure 13 - Display Trip Information use case

3.2.1.3 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.

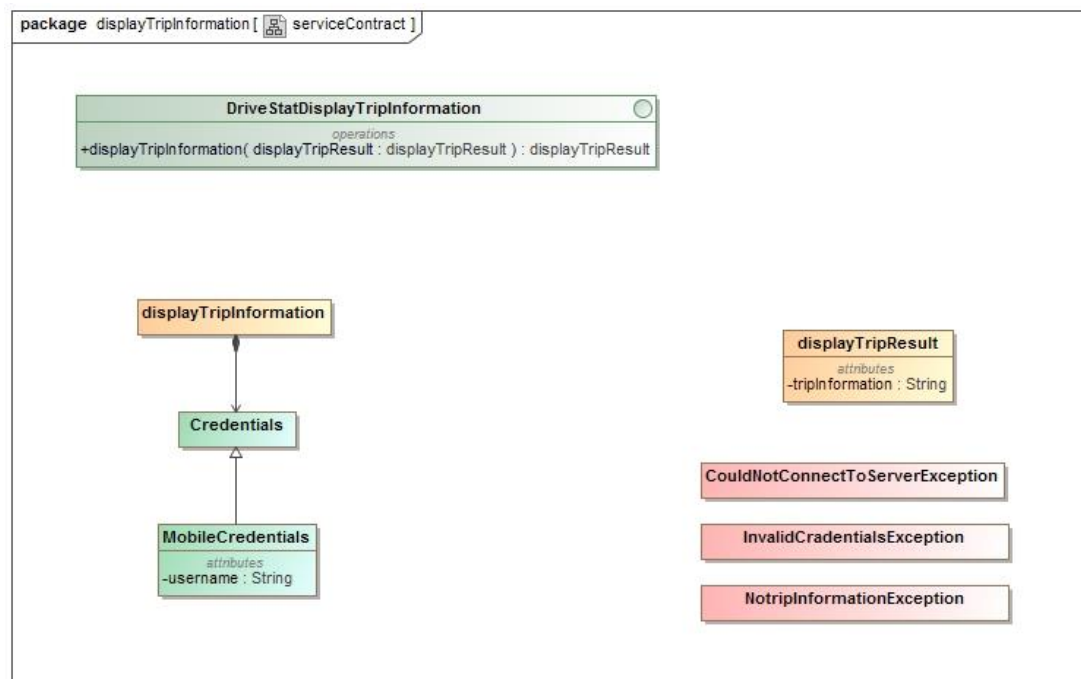


Figure 14 - Display Trip Information service contract

3.2.1.4 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.

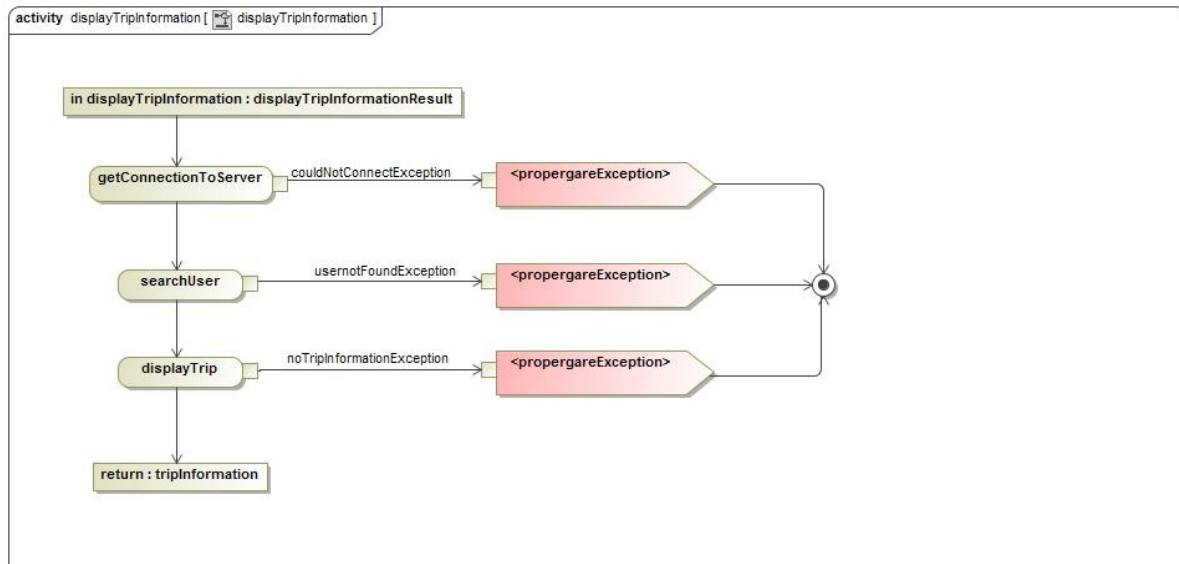


Figure 15 - Display Trip Information process specification

3.3 Nice-To-Have

3.3.1 viewComparedResults

3.3.1.1 Description

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

3.3.1.2 Use case

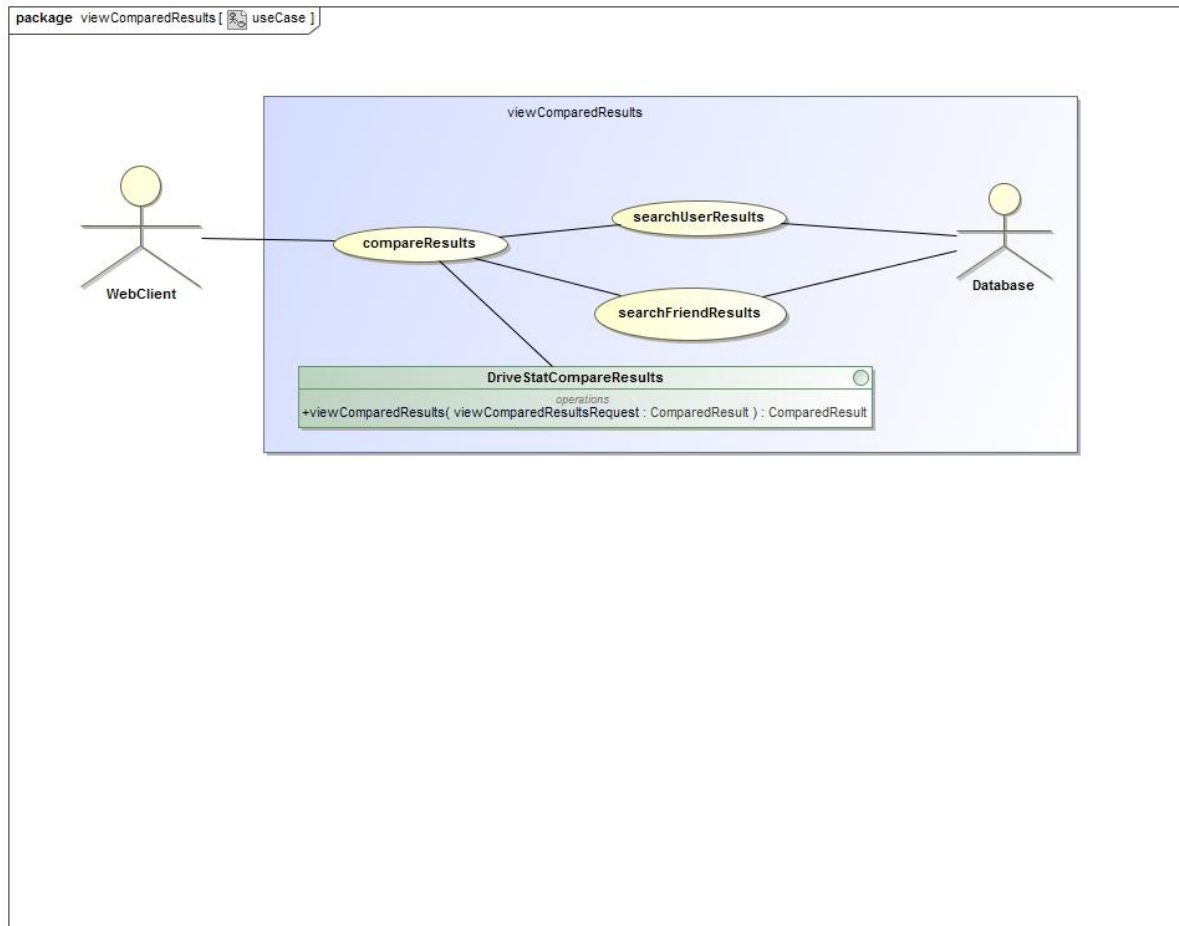


Figure 16 - View Compare Results use case

3.3.1.3 Service contract

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

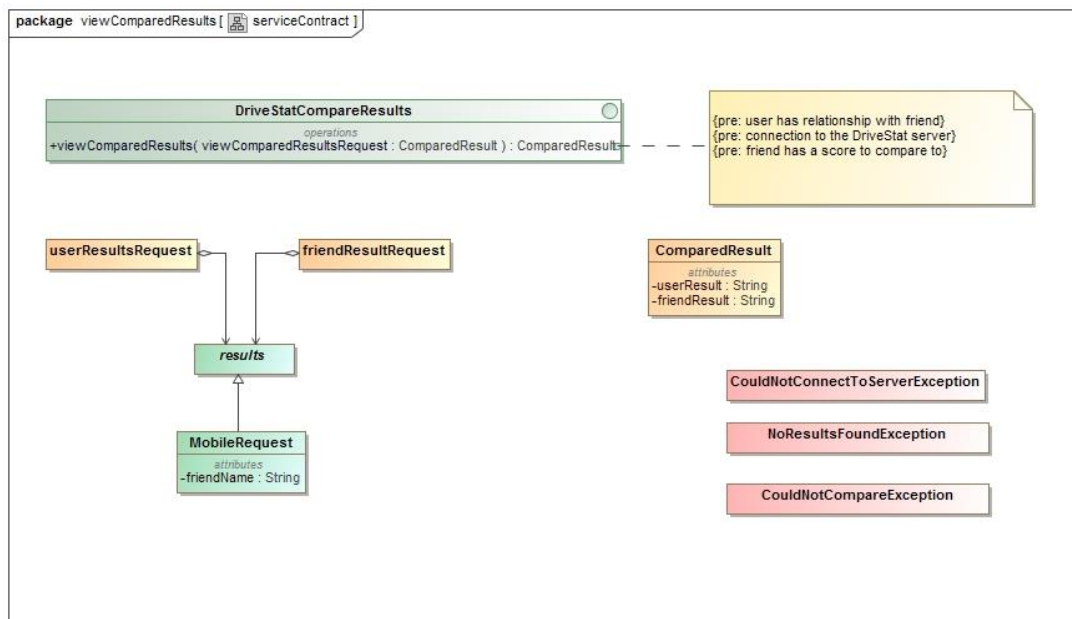


Figure 17 - View Compare Results Service Contract

3.3.1.4 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”.

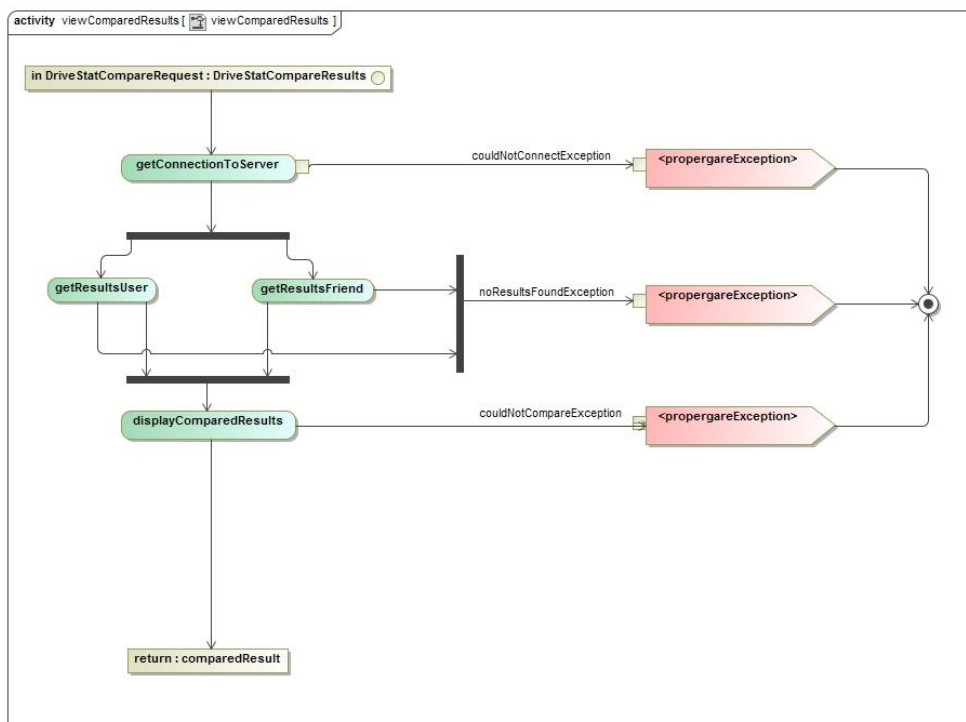


Figure 18 - View Compare Results Process specification

4 Domain Model

The Domain for the DVT DriveStats Application Spans 2 distinct ranges of technology. For this Reason the Domain Model of the Project is divided into two distinct sections and are connected via well-described API's.

4.1 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.

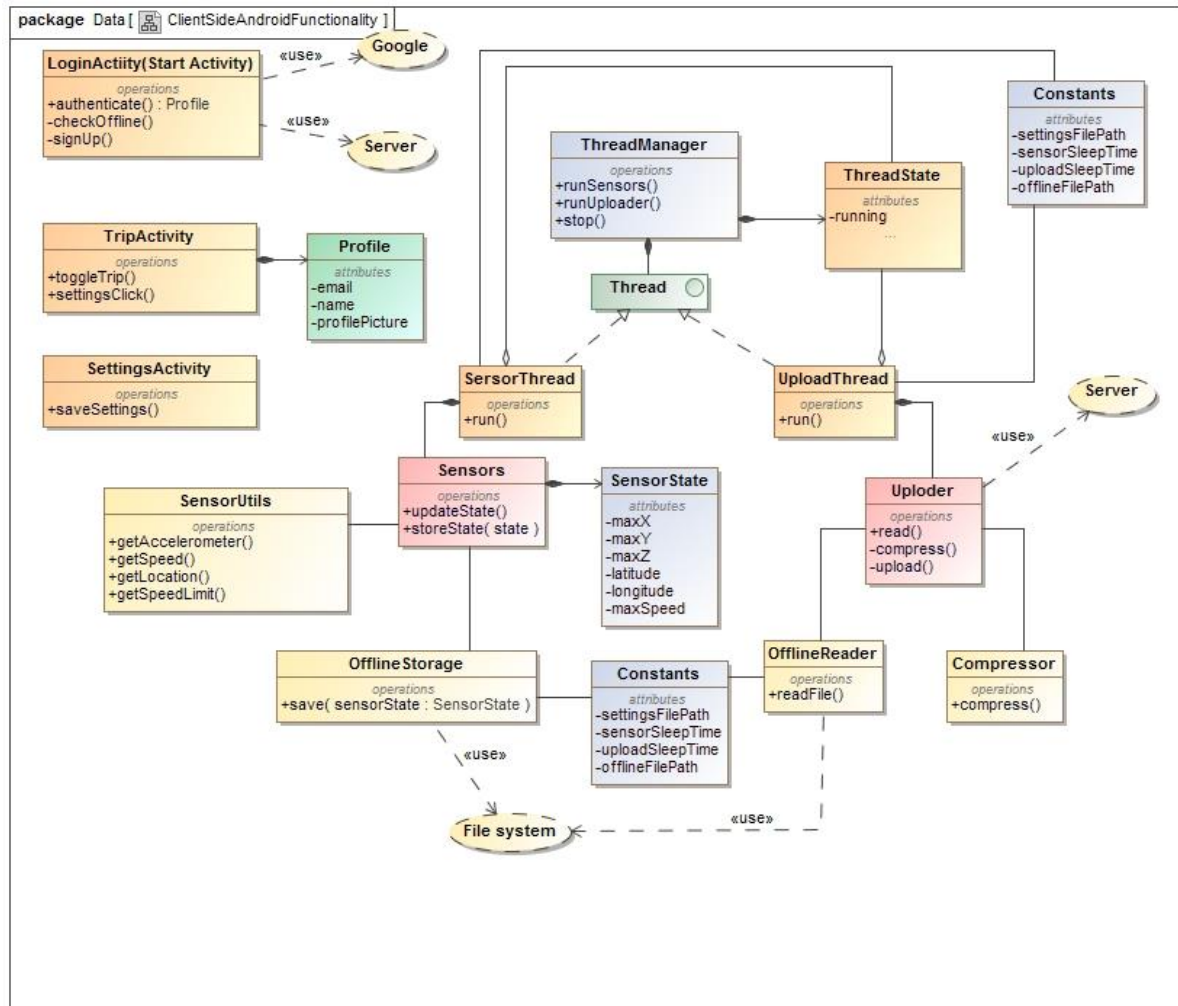


Figure 19- Client-Side Class Diagram

4.2 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

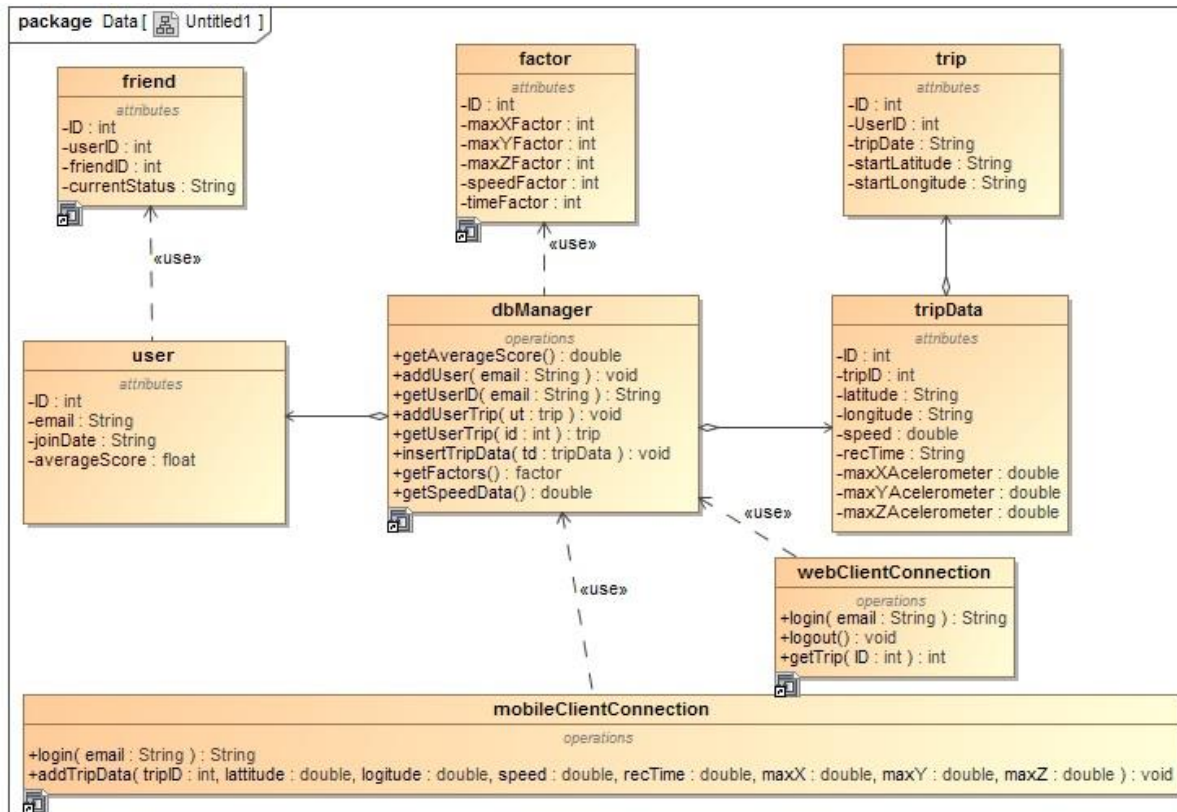


Figure 20- Server-side Class Diagram

4.3 Database Diagram

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

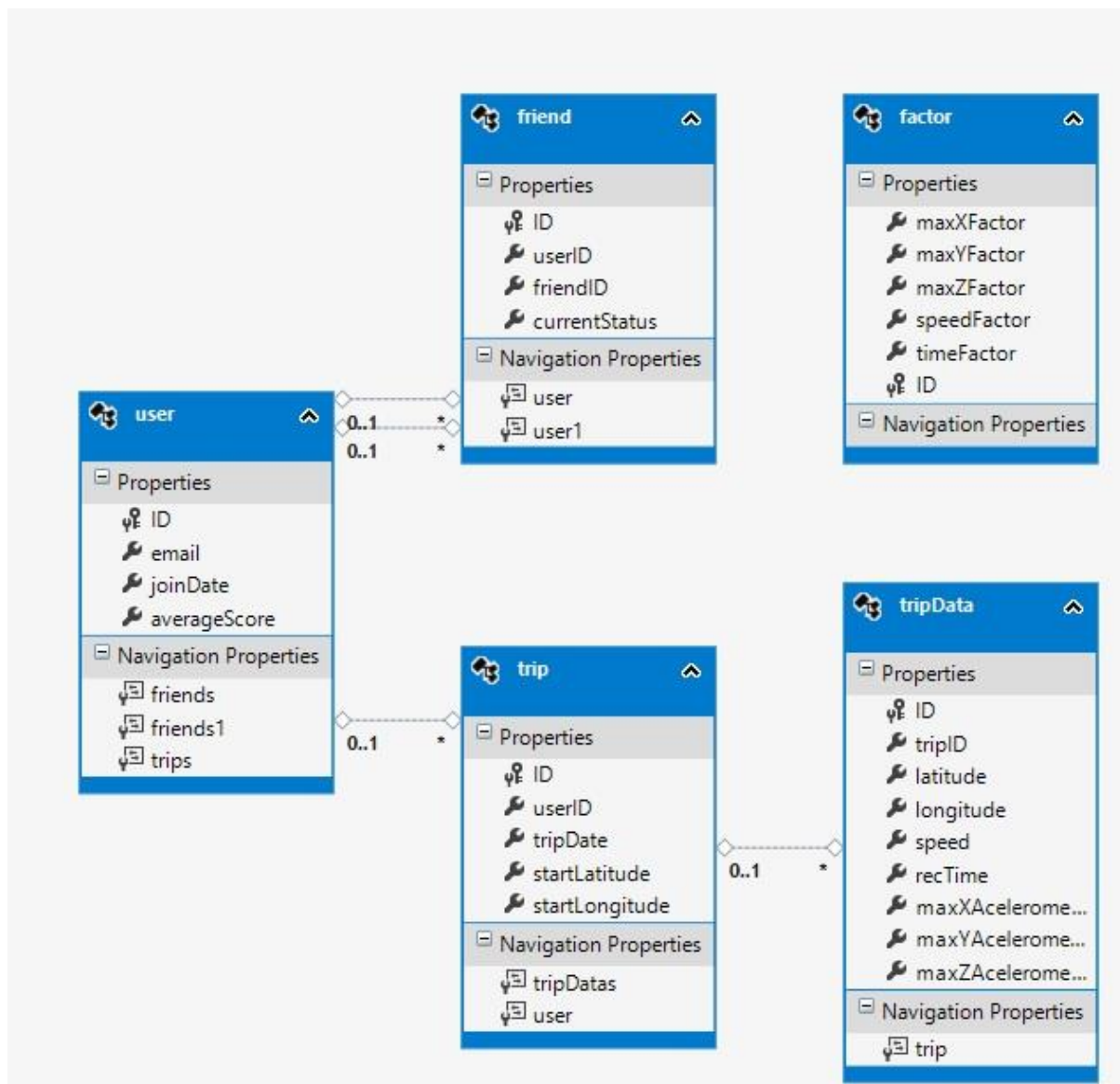


Figure 21- Database Organisation Description

5 Access Integration Channels

5.1 Access Channels

5.1.1 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.

5.1.2 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.

5.2 Integration Channels

The key integration requirements for Drivestats is the ability to efficiently and effectively make use of Microsoft Azure as the cloud component of the service. Furthermore, the application should remain compliant with the Google+ API used for the login functionality of the application.

6 Quality Requirements

6.1 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.

- 6.1.1.1.1.1 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.
- 6.1.1.1.1.2 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.

6.2 Performance Requirements

6.2.1 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.

6.2.2

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.

6.2.3

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.

6.3 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.

6.4 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.

6.5 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.

6.6 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 quantitative 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.

6.7 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.

6.8 Integratability

Integratability 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.

7 Architectural Responsibilities

- This system depends heavily on its open source access to the Google API used for login.
- The architecture must make use of a RESTful API (as requested by client).
- The use of libraries for device calibration requires an up-to-date and effective Android-side implementation.

8 Architecture 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.

9 Technologies

9.1 Build Technologies

- 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)
- Android Studio

9.2 Implementation Technologies

- Azure Server for database and server-side hosting considerations
- Android JellyBean
- Google API's for login