FIT5140 Assignment 3 Project Documentation

1. Architecture

Due to the strongest reliability of Firestore and S3, data can be persisted with these cloud providers instead of mobile phones. In addition, as users may not put the application in the foreground, Raspberry Pie is better suited to connecting to multiple sensors and collecting data, as it can always handle the task. Raspberry Pie is designed in such a way that it can be more easily integrated with other modules, where speed, GPS and images are correlated to form a relatively complete picture of the user's driving behaviour.

Finally, this is a server-side client program. In the project, the mobile phone is the recipient of the content most of the time, Raspberry Pie is the producer of the data, and the cloud service (Firestore, S3) guarantees the persistence of the data.

2. Database design

Raspberry Pie collects data and presents it in a non-relational way. The advantage of this is that different driving environments and vehicles will produce different data for the processing of vehicle driving information. The use of non-relational data is therefore a good way of dealing with this situation.

In the storage of the data, the key will still exist to locate the unique document and the key will also be applied to the different collections to form some sort of relationship between them.

Thus, non-relational data provides scalability, keys to constrain data relationships and thus ensures robustness.

3. Raspberry programme

The programs in Raspberry Pie are written in Python, as most of the library methods required by IoT programs can be found in python. The programs in Raspberry Pie are written in Python, as most of the library methods required by IoT programs can be found in python.

Another viable language to communicate with other sensors is C, which is much more efficient than Python. However, for this project, the request would not have been completed quickly enough using C, so Python became the preferred language.

Most of the task processes will be encapsulated in class form, which will be useful for future modifications and additions.

4. iOS application design

The most common function of the iOS software in a project is to communicate with the various service interfaces. Therefore, a multi-tier access data processing process has been designed. In this process, a multi-level data cache will be applied in order to increase the efficiency of the execution.

The execution process of a network request provides API in a pipeline-like fashion, so that any caller wishing to make a network request can implement a corresponding protocol to achieve a uniform request call.

On the other hand, the number of pages and design elements increased. xib combined with the view UI design pattern was introduced, in which views with similar functions can be completely reused. This reduces duplication of code. Most of the communication between the key API is designed as a protocol, so that the caller is unaware of the implementation, thus ensuring low coupling.

5. Data persistence

Currently, the service consists of Firestore, which is the store of data, and Amazon S3, which is the store of static files. Firestore provides a friendly non-relational data model and performance. S3 provides a good way to make things like images and videos easily viewable via http URL.

Due to the fact that the view controller is accessed on the basis of protocol in iOS applications. As a result, the application has no knowledge of how the data is organised in the data provider. This offers the possibility to migrate in the future when the performance of Firestore and s3 is no longer sufficient for the application.