Contents

[1. Preliminary software design 1](#_Toc502915939)

[1.1. The Exception Reporter 2](#_Toc502915940)

[1.2. The Controller 2](#_Toc502915941)

[1.3. The Taximeter Client 2](#_Toc502915942)

[1.4. The Engine 3](#_Toc502915943)

[1.1.1. XML Library 3](#_Toc502915944)

[1.1.2. Communications library 3](#_Toc502915945)

[1.1.3. The data store 4](#_Toc502915946)

[1.1.4. The Data Library 4](#_Toc502915947)

[1.1.5. Encryption Library 4](#_Toc502915948)

[1.5. The Exception Reporter 4](#_Toc502915949)

[1.6. Security requirements for the Disclosure System 5](#_Toc502915950)

[2. The preliminary production infrastructure 6](#_Toc502915951)

[2.1. Platform for the engine 6](#_Toc502915952)

[2.2. Platform for the data store 6](#_Toc502915953)

[2.3. Platform for the Exception reporter 7](#_Toc502915954)

[2.4. Platform for the controller 7](#_Toc502915955)

[2.5. Platform for the Taximeter Client 7](#_Toc502915956)

[2.6. The Production network Environment 8](#_Toc502915957)

[3. Test environment 8](#_Toc502915958)

[3.1. Security of the test environment 9](#_Toc502915959)

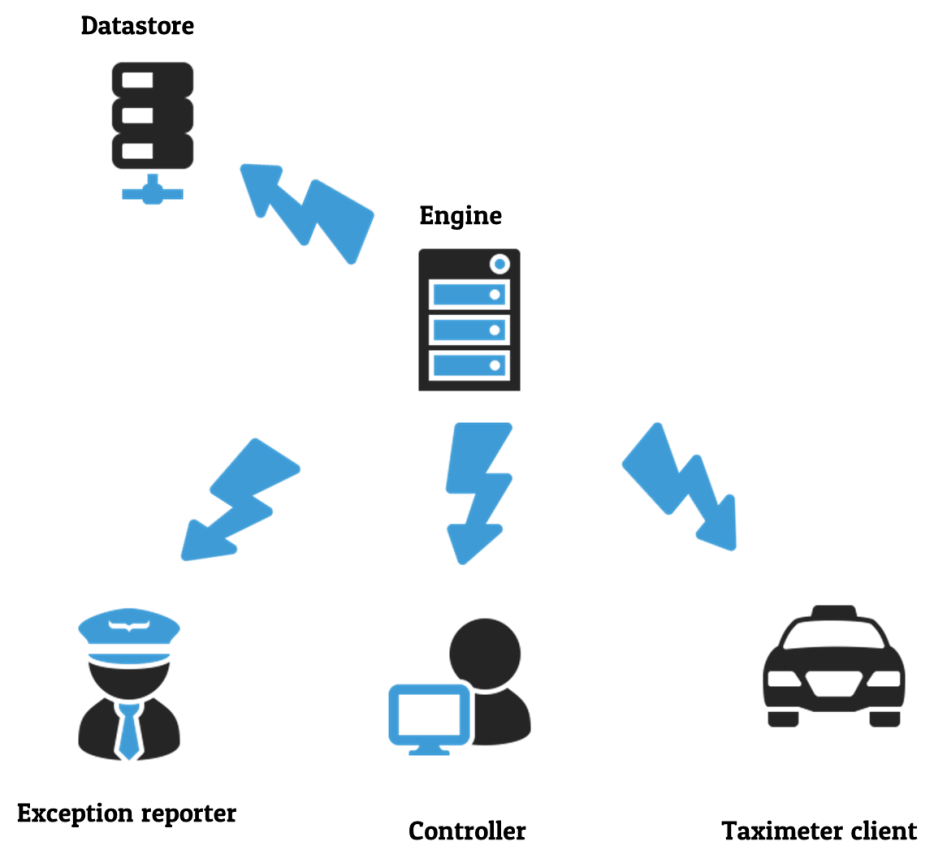
[4. The development environment 9](#_Toc502915960)

[4.1. Security of development environment 10](#_Toc502915961)

# Preliminary software design

The purpose of the preliminary design is to get an overview of the project. This overview is necessary to break up the system into manageable pieces and to begin to prioritize the design/development work

In the figure below the project has been broken down into its major constituents.



## The Exception Reporter

The exception reporter is a piece of software that will be used by taxi companies to:

* Report exceptions
* View alarms pertaining to their company
* Download data about their company that has been reported to 3 parties

## The Controller

The controller is used by the employees of the reporting center running the Disclosure System and will be used to:

* Set up users and privileges
* Verify users and ensure that they are only given access to data and functionality that they have privileges to.
* Register taxi companies and their vehicles.
* Review all data in the data store
* Create logs of all user actions
* Present alarms in case of failed or omitted transitions
* Search for data selections of taximeter data and exception reports
* Initiate the creation of reports to third parties
* Register data requests from third parties

## The Taximeter Client

The taximeter client is the software that will be running in taxis that are reporting to the disclosure system. The Taximeter client will need to:

* Store vehicle parameters and protect these parameters from being tampered
* Covert raw taximeter data to XML and send it asynchronously and encrypted to the Engine.
* Buffer and encrypt data for later transmission in the case communications breakdown with the engine and ensure that this data is not tampered with or deleted while it is stored.

## The Engine

The engine is the sever software that will contain the bulk of the business logic of the system. The server must maintain secure communications and stay available to the reporters, controllers and clients in the system.

The Engine is the only piece of software in the system that will be able to directly read or white data to or from the data store. As such the server is the guardian of the data and must prevent malicious transactions from harming the data.

The server should also be able to see the central components of the system and monitor them and itself for failures such as network down & database down.

The server must monitor the incoming data from the Taximeter Clients and generate appropriate alarms in case of deficient or omitted transmissions.

Create correctly formatted files for third parties who request information.

The server’s role is so complex that we will break down the server design to some major libraries or function sets that can be developed independently to simplify the development of the server.

### XML Library

The final output of the data when being reported to third parties is XML so it makes sense to use XML as the overall system format for transporting data between the software components over the network. We will develop an XML library that facilities the simple conversion of relational data to XML and back again.

### Communications library

Communication is vital in a system that will be very distributed. The communication is not only vital it also poses a security risk. The communication which will be enabled via TCP sockets will have to take place in an asynchronous non-blocking manner as it allows all software components to user the same socket meaning the software will be able to communicate with the engine without any configuration for the network (a synchronous configuration would require a unique port number for each client to ensure stability.) For security purposes all TCP communication will be encrypted.

The communications functionality will be a relatively complex piece of code and will therefore be developed as a library covering both server and client functionality.

### The data store

This is the database that will store all data. The data store has no business functionality or any kind of intelligence whatsoever it is just a data store. As mentioned before the only user of the database will be the engine. This means that no database connection information is stored in the controller, exception reporter or the taximeter client which is good for security. Neither will the database need a public IP, as in the case of the server, protecting the database further.

From TFS 2016:47 we can deduce that a minimum of the following tables will be needed

* Users
* Taxi companies
* Vehicles
* Exception Reports (from customer)
* Taximeter data
  + Batch header
  + Taximeter report header
  + Trip data
* Data requests from third parties
* Data outputs sent to third parties
* Transmission deficiency reports
* Transmission omission reports
* Alarms
* User change logs

### The Data Library

The Data Library will handle transactions between the server and the database. The database library should accept only system specific commands and parse these commands to SQL for interaction with the database. The Database Library should abstract implementation specific information about the database and also abstract the use of SQL commands from the rest of the system to guard against SQL injection attacks.

### Encryption Library

For security the network traffic needs to be encrypted to ensure strong, simple and standardized encryption methods across the different clients.

This library should also handle the encryption of data that needs to be buffered on for example a hard drive.

## The Exception Reporter

The Exception Reporter allows taxi owners to report in exceptions. An exception is a reason for a car not to report in taximeter data. Valid reasons are specified by TSRYT 2016:46

If an exception has been reported then an alarm should not be raised if the taxi does not report in data for the exception.

As well as being able to report in data the Exception Reporter should allow the user to review his data pertaining to his company such as:

* Reported exceptions
* Data reported by taximeters
* Alarms raised due to missing or incomplete transmissions
* Requests for data from the Tax authorities or Transport Agency

## Security requirements for the Disclosure System

Each software component of the system will need to authenticate itself as legitimate software. This will be accomplished by all network communications and data stored on must be encrypted. The encryption will require a system wide certificate (essentially a password) and an initialization vector which will not be constant but calculated using a system wide algorithm for this purpose. Legitimate software is defined as software that has the correct password and the capability of creating the dynamic initialization vector. Illegitimate software will not be able to exchange well-formed data or commands between the disclosure systems server or clients. The encryption ensures that any data in transport on the Internet or on buffered local storage cannot be read or manipulated.

Now that only legitimate users of that software are connected to the network. Every user connecting to the system need to be authenticated using a unique identifier and a strong password.

No user save the disclosure server should be able to interact with the database.

Any client connecting the disclosure system should not need configuration with information about the system as a whole such as the server IP address, communication port or password information this is to be handled confidentially by the software.

All commands must have a specific purpose and be constructed to be safe. For example it is if a data row such as a Taximeter report is to be written to the database from the Taximeter Client then the client need to send a discreet comment that can do nothing else other than this task. It is forbidden to create a general server command like “ExecuteSQL” to accept an SQL command to write to the database. The generic server command could be exploited to execute a DELETE or UPDATE command thereby damaging or manipulating stored data.

Data may only be entered into the system. No client or server function should be able to delete or edit saved data. All entered data should be stamped with the identity of the authenticated user, the place of entry (defined by the devices external IP) and the time of entry.

# The preliminary production infrastructure

We wish to implement a 3 tier client server architecture this will allow us to scale performance where it is needed, meaning that data, business logic and client hardware can be upgraded independently of each other.

We have already pointed out that the database should not have a public IP and should therefore be housed on its own machine for security reasons.

Given the background of the development team primarily in the dotNet development environment we will be using windows operating systems where ever appropriate and there is no reason not to use the latest widely adopted version Windows10 in its 64bit version.

## Platform for the engine

As regard to which Windows version we feel that there is no need to implement the server version. The security of the Disclosure System is handled by the Disclosure System as well as the communication being encrypted even on LAN communication negating the need for a domain controller. Using a Win10 Pro operating system saves cost due to less hardware and electricity for the same reason with both environmental benefits and will increase the uptime of the system if it needs to run off UPS.

Regarding performance this system is not viewed as a real time system. Whether an operation takes 1-2 seconds or 7-8 seconds is not critical for the correct function of the system. The only important factor speed wise is that the interface for the Exception Reporter and Controller needs to be sufficiently responsive such that the user does not become frustrated. Therefore the server need not need the scalability offered by the Windows Server Environment.

The physical server should have the following minimum specifications which should be sufficient for at least 5000 connected taximeters.

* Intel i5 processer
* 12 GB RAM
* 500 GB SSD

Add one core, 2GB RAM and 50GB HDD to your configuration per 1000 taxi clients over 5000 clients.

## Platform for the data store

We can further save resources by not using Windows Server / SQL Server for the database environment. The development team has most experience with development in Microsoft environments however as the Data Store is not going to have any intelligence and is basically a software package that is being used out of the box free open source software can be considered. Therefore we propose PostgreSQL 9.6 on Debian 9 as a perfectly capable database on a very lightweight operating system. Again this design decision will cut licensing and hardware costs as well as power consumption.

The physical server should have the following minimum specifications

* Intel i5 processer
* 16 GB RAM
* 2TB SSD

## Platform for the Exception reporter

This was one of the more difficult design decisions. The first decision regarding the platform for the Exception Reporter was to host it on a webserver, the advantage being that users could use the system from a variety of different platforms with a web browser. However when considering the consequences of this decision the choice seems less than optimal. A webserver would be required. The web server would need to be hosted on a separate machine to preserve the three tier client server architecture with increased hardware and licensing costs. A new programming language would have to be introduced such as asp.net. Finally a webserver involves its own set of security issues.

As the taxi client is to be run in on android device we can safely make the assumption that all taxi owners will have access to an android device. We therefore select this as the platform for the exception reporter as well.

Minimum requirements for the Exception Reporter device are:

* Intel Atom x3 C3200 or equivalent
* 1GB RAM
* 16GB storage

## Platform for the controller

The platform for the controller is Win10 Pro 64bit.

The physical PC should have the following minimum specifications

* Intel i5 processer
* 6 GB RAM
* 250 MB HDD

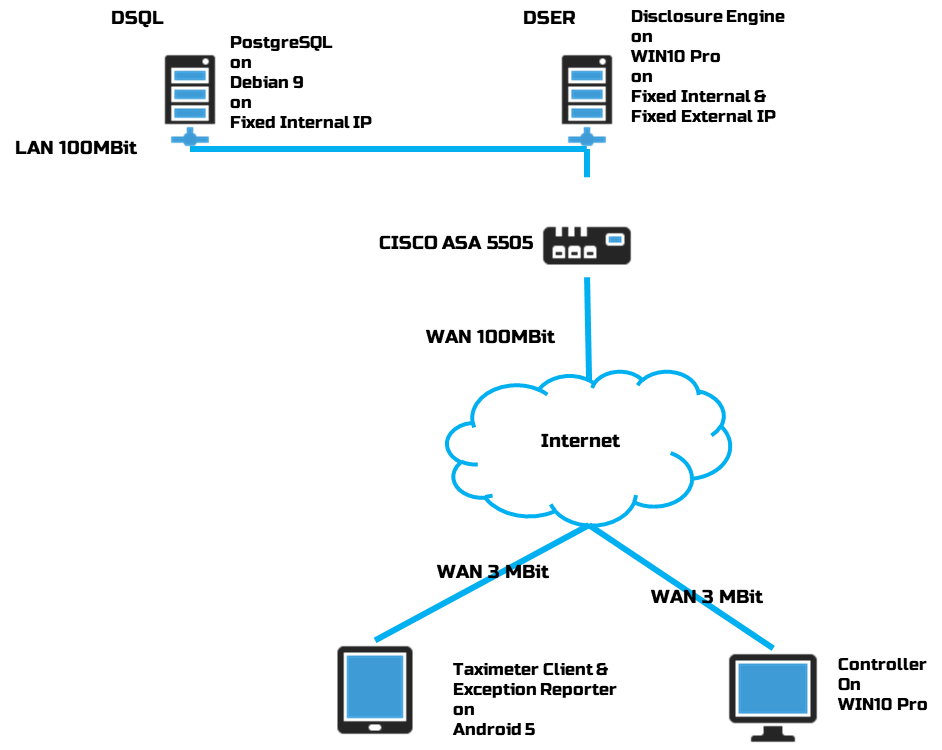
## Platform for the Taximeter Client

For the taximeter client we will select the Android platform version 5 and upwards. This application will take very little resources from the tablet meaning any tablet or smartphone manufactured after 2016 should be fine but we will set some minimum specs to be sure.

* Intel Atom x3 C3200 or equivalent
* 1GB RAM
* 16GB storage

## The Production network Environment

Again no component of this system is real time and in the production design serves to not exaggerate the need for overly powerful hardware rather efficient code and lightweight operating systems. The same goes for the LAN where a 100MBit connection will suffice as will a 100MBit internet connection. For the firewall / router we select the CISCO ASA 5505.



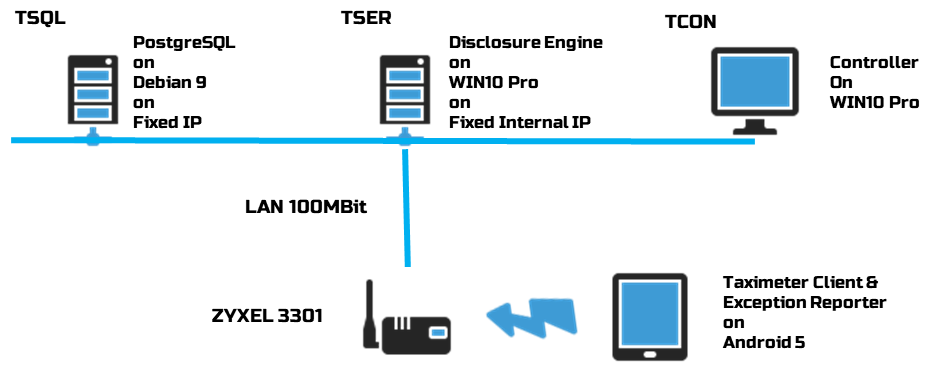
# Test environment

The test environment is designed to run on highly under dimensioned hardware the point of this is to expose and performance related issues as quickly as possible. The test environment will however have enough machines to mirror the 3 tier client server architecture to check this is working properly over the network. All software products (such as operating systems and database) will be identical to the production environment.

The test environment will consist of:

* A machine (TSQL) with an Intel Atom Processor and 2 GB RAM with Debian 9 and PostgreSQL 9.6 for the DataStore
* A machine (TSER) with an Intel Atom Processor and 2 GB RAM with Win10Pro 64 bit for the Disclosure Server
* A machine (TCON) with an Intel Atom Processor and 2 GB RAM with Win10Pro 64 bit for the Controller
* An Asus ZenPad C7 with an Intel Atom processor and 1GB RAM with Android 5.1

For the test network we will create an isolated LAN (no internet) using a Zyxel LTE 3301 for a router and wireless access point for the tablet.



## Security of the test environment

Access to the test environment will be physically controlled by a door lock with keys only being provided to the project team.

Access to the computers will be restricted by a strong project password on each machine.

Access to the computers via the network will be restricted maintaining the project environment on an isolated LAN and password securing the WIFI connection to the router. WIFI is necessary for testing the android device.

All executables shall be obfuscated before leaving the development environment hence any software in the test environment cannot be decompiled to gain insight to the system.

Any executables that are to be run on the test environment should represent a release on Github so that test can be rerun.

# The development environment

Development will be conducted in C# with Visual Studio 2017 Community Edition as the IDE both for Windows and Android environments. We will use Devarts dotConnect for PostgreSQL 7.10 Express to allow easy access to the Postgres database from .Net. We will also use Infragistics Windows Forms controls to quickly provide a polished user interface. All software will target .Net 4.6.

Each developer will have a development machine (DEV1 and DEV2) running Win10 Pro 64 bit with the following specifications:

* Intel i5 processer
* 16 GB RAM
* 500 GB SSD
* 1 TB

The development environment will also need a development database (DSQL) which will be identical to TSQL. The development environment will be on the office LAN but behind its own router for added security

## Security of development environment

Access to the development environment will be physically controlled by a door lock with keys only being provided to the project team.

Access to the computers will be restricted by personal password on each PC. The PCs will further be protected by Kaspersky Total Security.

Access to the computers via the network will be restricted by a project router / firewall. The firewall will be set up to block unnecessary ports and connections to the development environment PCs from outside the project LAN.

All project documentation, 3rd party software and code will be uploaded to Github to protect against data loss and also to track versions of all project deliverables.