Contents

1 Introduction

1.1 Purpose

The Purpose of this document is to address the non functional requirements and not the functional requirements. This includes the ability to provide the specifications of an appropriate software infrastructure within which the application is specified, deployed and executed. Such that the functionality is accessible through the appropriate access channels, as well as to ensure that integrated channels are supported and that the functionality is available with the required qualities.

1.2 Document Conventions

- Crow's foot notation used for entity relationship diagram
- Heading and subheading adhere to latex's conventions
- UML 2 used for use case modelling
- Comments that we create will be indicated in green.
- if we request something to be deleted we will highlight the work in red.
- We used Kruchten's 4+1 architectural standard on our designs.
- The 7 ISO OSI network standards have been followed.
- We used to MVC design pattern.

1.3 Project Scope

The project aims to design a system to keep track of times allocated to tasks and resources that each job requires. the system will replace the current job cards system by a digital system which allows workers to log resources expended on each task, and also for clients to sign off on tasks after a worker's time segment is done for the day, or when the task is done. The system will also allow for automatic billing of clients by a pluggable Billing API, and will restrict the creation of tasks to workers authorised to create tasks.

1.4 References

- http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.90.9484
- http://www.scribd.com/doc/106789883/4/Kruchten
- http://support.microsoft.com/kb/103884
- http://www.bpsharma.in/eLearning/Networking/OSI_Reference_Model.htm
- http://ezinearticles.com/?Seven-Layers-of-ISO-OSI-Model&id=349951
- http://www.inetdaemon.com/tutorials/basic_concepts/network_models/osi_model/
- http://ist.berkeley.edu/as-ag/pub/pdf/mvc-seminar.pdf
- http://www.dossier-andreas.net/software_architecture/pipe_and_filter.html
- http://support.microsoft.com/kb/103884
- http://www.bpsharma.in/eLearning/Networking/OSI_Reference_Model.htm
- http://ezinearticles.com/?Seven-Layers-of-ISO-OSI-Model&id=349951
- http://www.inetdaemon.com/tutorials/basic_concepts/network_models/osi_model/
- http://ist.berkeley.edu/as-ag/pub/pdf/mvc-seminar.pdf
- http://www.wordnetweb.princeton.edu/perl/webwn
- http://www.uml-diagrams.org/communication-diagrams.html
- Jaco Kroon - Ultimate Linux solutions

1.5 Related Documents

- The Requirement Specification

2 System Description

The intent of the system is to provide for time management, for both client and workers in a task oriented environment. I will achieve this by replacing an existing job card system with a digital system that allows resources expended on a task to be easily recorded and viewed later.

Services requested by clients are represented as tasks. A task is then assigned to a worker, or placed in a list of unassigned tasks, and a worker who has been assigned to a task, is responsible for completion of the task. Times that worker spends on a task are represented as time segments.

The Client has a unique digital pin that he/she can use to sign of a worker's job card upon task completion or end of day time segment. Once the entire task is complete, the client signs off tasks completion by entering the pin code on the system, and then, the client gets automatically billed by using the pluggable Billing API

3 Overall Architecture

The overall architecture of Kruchten's 4+1 architectural view model uses five coexisting views that deal with a particular concerns to create a description of the software architecture. The Kruchten 4+1 architectural models works as follows, it use the four views –Logical View, Development View, Process View, Physical View– to capture data and then it uses the fifth(+1) view - scenarios as a means to illustrated the data as well as a way of authenticating the four views.

3.1 Architectural Strategies

This is to conceptually specify how we are aiming to cater for the quality requirements. (Scalability, reliability, performance, security, auditability, exibility/extensibility, usability)

3.1.1 Thread Pooling:

This will be use to improve scalability because by making use of a thread pool you inturn have more threads that can execute tasks concurrently which will be useful when your system gets larger. This will allow for more efficiency when the task loader gets larger.

3.1.2 Clustering:

This will be used to improve scalability and reliability. This will be achieved because clustering is a set of loosely connected computers that work together and thus intern can be viewed as a single system. To improve scalability one can always add more computers to a system if the option seems viable and this improves reliability because if one computer were to go down the entire system is still capable of running independently.

3.1.3 Interception:

To acomplish this you wil have to ensure that no authentication informations is done at client side. The user log in details should be incripted before it is sent to the server. The server should then grant access and not send the authentication information back to the client.

3.1.4 Run-Time Lookups:

This will be used for pluggability and flexibility because if the system is able to do run-time lookups a new system can be plugged in and will be executed at run time. Thus in turn making the system more flexible so that parts of the system can undergo development without having to effect the system.

3.1.5 Queuing:

This will be used for reliability and scalability.The reason why queuing is used is because if every task that needs to be done is queued if the system

crashes or there are any issues, the tasks will still remain in the queue and be processed once the system is back up and running. This way no data is lost when the system is down or changes are being made to the system such as upgrading.

3.2 Architectural Patterns

High level infrastructural constraints on the software architecture.

3.2.1 Layering:

A network following the ISO OSI model uses 7 Layers. These are divided into two sections "Upper Layers" and "Lower Layers". The former is also referred to as the Application Section and can be broken down to the following layers:

- **Application (Layer 7)**In this layer it all comes down to the user interface, so it is responsible for representing data to the user in a meaningful way.
- **Presentation (Layer 6)**In this layer the data is translated so that the it can be displayed in the Application layer.
- **Session (Layer 5)**This layer is in control of session establishment, maintenance and terminations. It controls the session that is running between processors on different stations.

Where the latter is also referred to as the Data Transport Section and can be broken down into the following layers:

- **Transport (Layer 4)**This layer is on control of ensuring that the packages are delivered to the correct place and that the appropriate response is given based on the feedback it receives.
- **Network (Layer 3)**This layer is in control of which path the packet will follow to insure that if it is high priority that it will get there in the quickest time possible.
- **Data-Link (Layer 2)**This layer is in control of error checking.
- **Physical (Layer 1)**This layer is the the lowest layer in the ISO OSI model and deals with transmissions and receptions on a bitstream level.

3.2.2 Model-View-Controller(MVC:

The MVC is a design pattern for web applications that separates representation from user interactivity. The role that each of the components play are:

- **Model:** application data, logic and functions.
- **View:** representation of data.
- **Controller:** mediates input for model and view(Navigation)

This intern helps make the system more flexible and allows for more work to be done concurrently. This allows the Work to be split up and does not prevent any group (Designers, Engineers, Programmers.) from depending on another component (MVC) to have been completed before they are able to complete their work. This makes development more flexible, the designers of the UI can rework the design as much as the client requires without slowing down the backend developers.

3.2.3 Pipes and Filters:

Stateless processing units connected by pipes providing a high level responsibility of distribution. This is a process of dividing up a larger task into smaller subtasks.A filter transforms the data or simply filters through it. The pipes are connectors between the different filters.

3.3 Reference Architectures

- JEE

3.4 Technology and Framework Selection

The technologies that will be considered are:

3.4.1 Programming Languages

- Java
- PHP
- Javascript
- JQuery
- HTML 5
- CSS & CSS3
- Jason
- Ajax
- XML

3.4.2 Database Servers

- Java DBMS
- Sql
- PHP
- XML

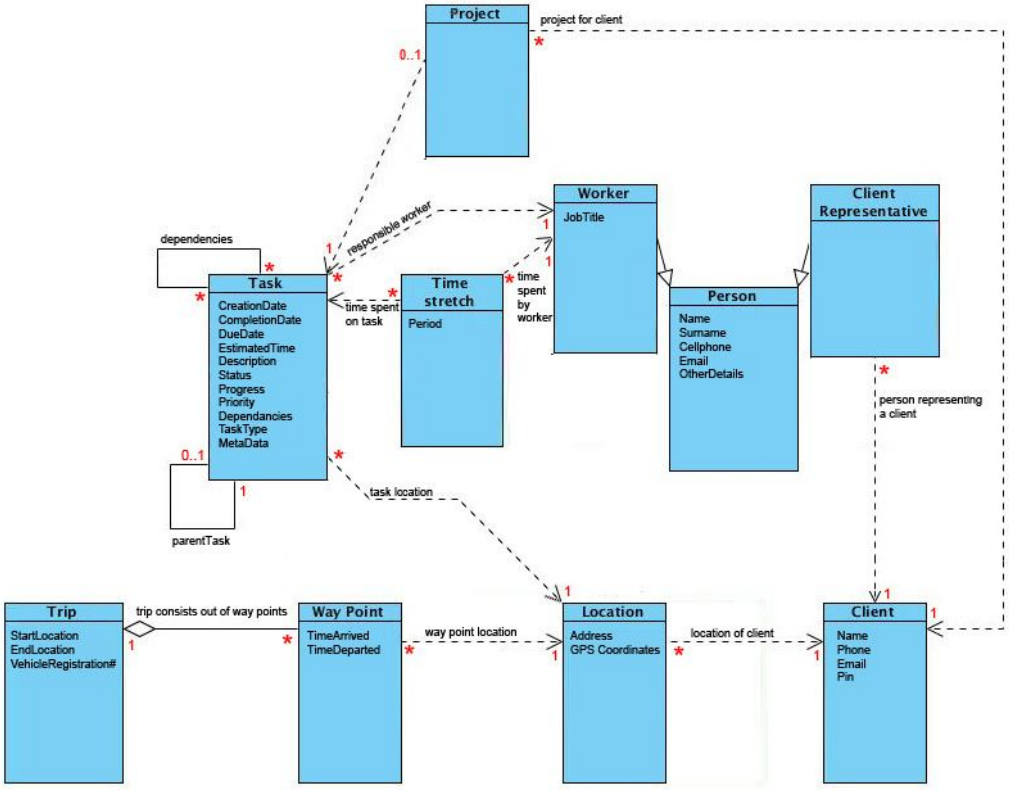
3.4.3 Web Server

- HTTP
- HTTPS

4 Details of System

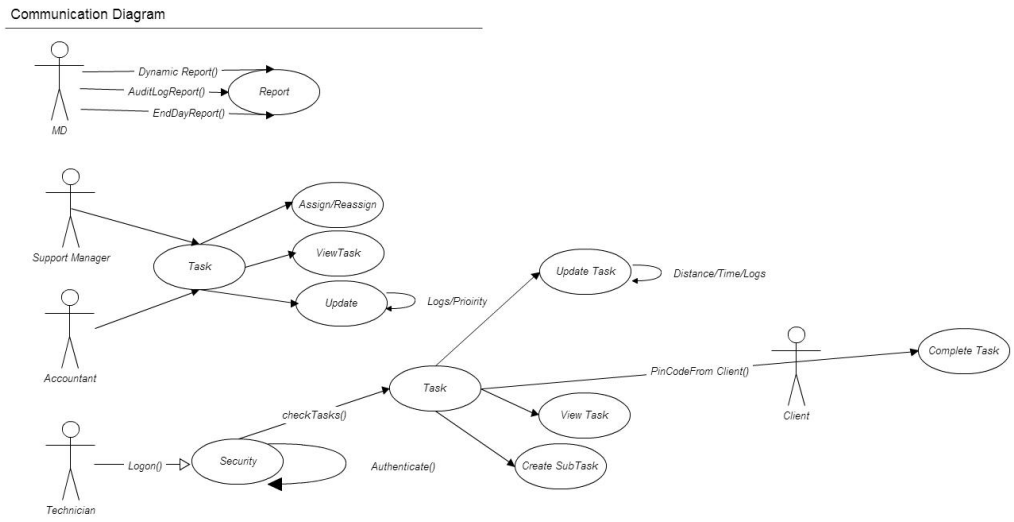
4.1 Logic View

4.1.1 Class Diagram



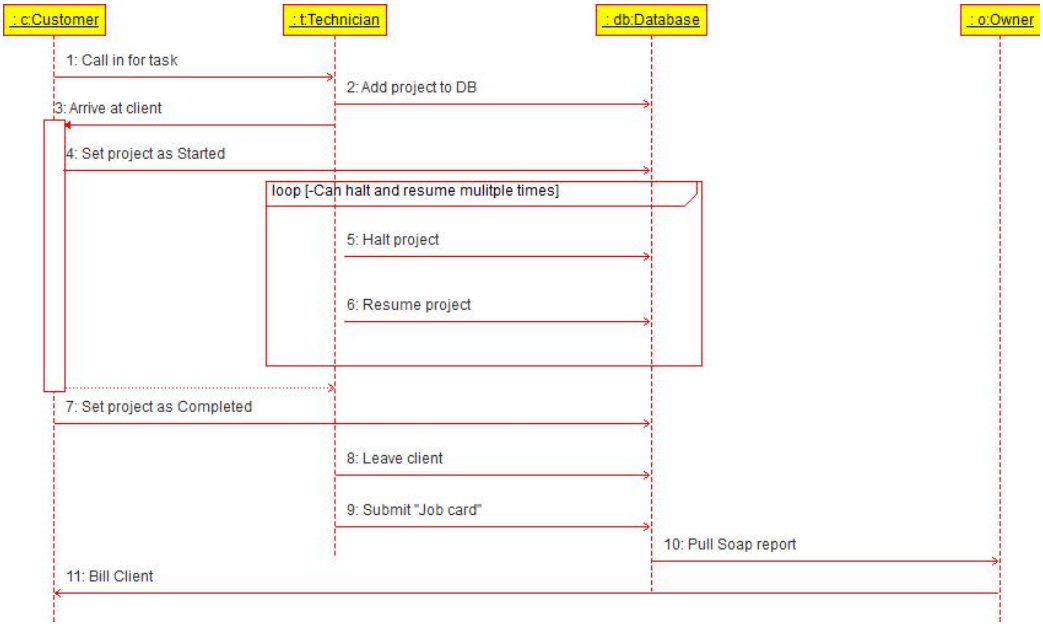
4.1.2 Communication Diagram

The Communication diagram models the communication between objects and tasks of the work flow. Manager Director(Jaco Kroon) would communicate with reports to execute 3 commands namely (Dynamic Report,Audit Log Report and End of the Day report). Support Manager and Accountant will be able to create new tasks and assign these tasks to technicians. They can View tasks at how far it is from finished or any new updates. They can also update the logs or specify a priority on the tasks. Technician should logon to his phone/system and will be authenticated before it carries on. Tasks can be updated locally and then submitted to the system when in range. Subtasks can be created. For tasks. Completion of tasks can only happen when the Pincode from the client is provided



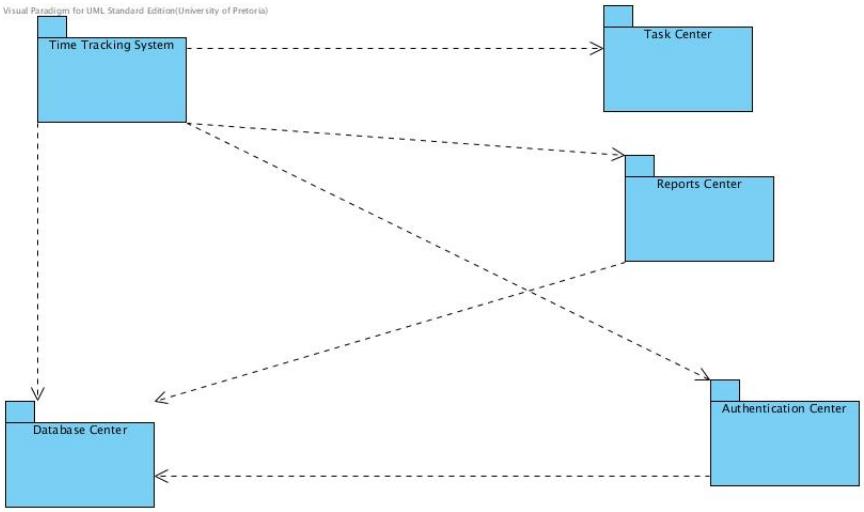
4.1.3 Sequence Diagram

The sequence diagram is used to show the tasks used to complete a specific task in our software. A customer will call in and talk to a technician, from here he will record the task as a project within the database. It will be sent to a pool from where it will be distributed to be completed. A technician will receive the job and when arriving at the client it will be marked as started by the customer. While at a client a task can be halted and resumed to not cause extra billing for time not spent working for client. After work is completed, client stated task as completed. The technician will return to the office and complete a "job card" to submit to the database. A soap diagram will be drawn and will will be done and sent out to the client.

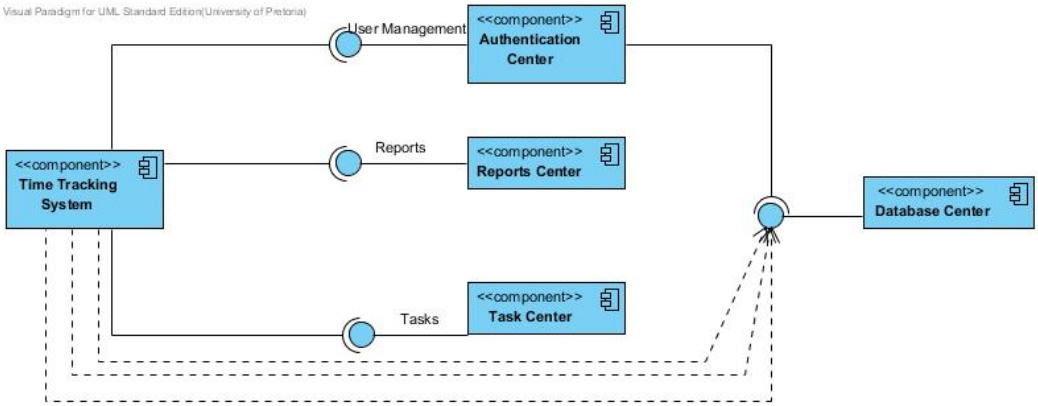


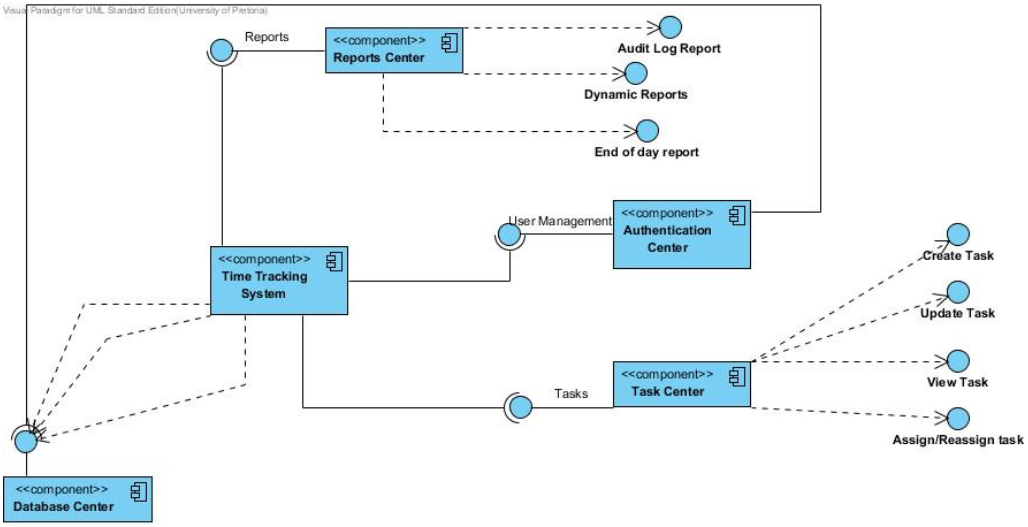
4.2 Development View

4.2.1 Package Diagram

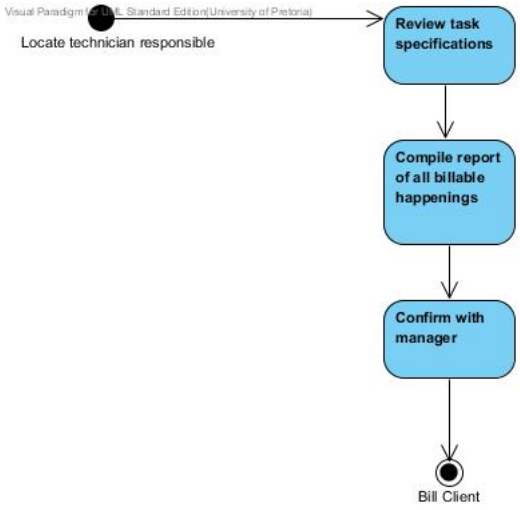


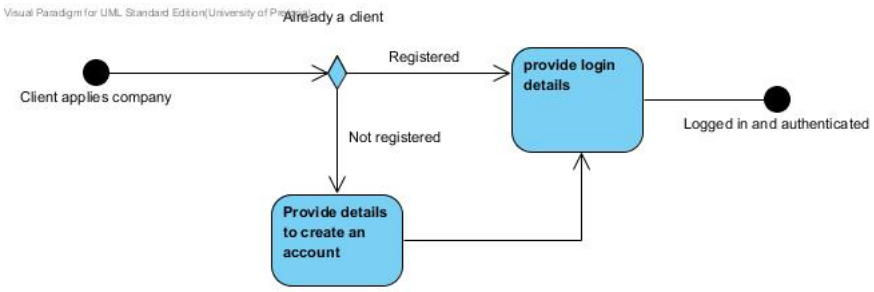
4.2.2 Component Diagram

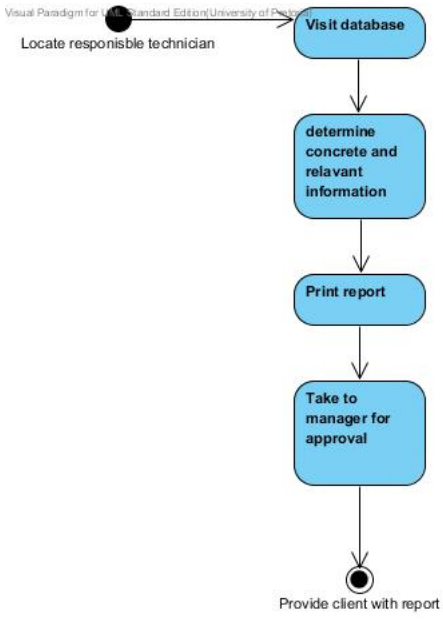


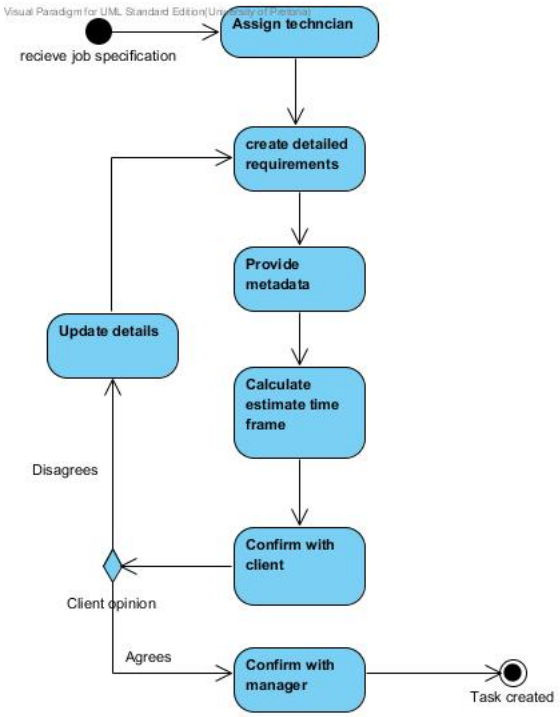


4.3 Process View




Already a client
Client applies company
Registered
Not registered
provide login details
Logged in and authenticated
Provide details to create an account


Locate responisble technician
Visit database
determine concrete and relavant information
Print report
Take to manager for approval
Provide client with report



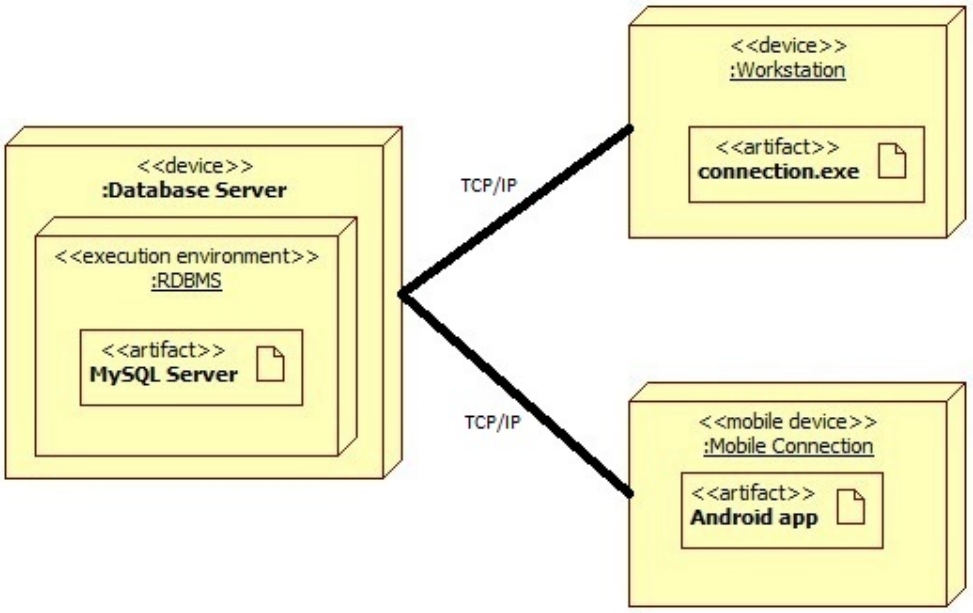
4.4 Physical View

The process view is responsible to graphically depict and provide insight about the system process to provide clarification on these processes and the way they communicate with one another. The process view focuses on the dynamic facet of the system and therefore depicts the system and its activities in a runtime fashion. The process view provides a clear layout of concurrency, distribution, performance and scalability of the system as well as the parts of the system that are active in theses aspects. An Activity diagram is the ideal UML diagram to represent processes in a system because it graphically represents activities and actions from the very start of a process to the end in a step by step manner to ensure that the workflows of components can be clearly observed and evaluated. All activity diagrams have an initial node – indicating the start of a process – and an activity final node – signalling the end of a process. There are 2 types of flows in an activity diagram namely control flow (Starts an activity node once the previous one has finished) and object flow (Can have objects or pass data along it). Decisions are depicted through the decision node while a fork node followed by a join node clearly

shows concurrent process. Clearly the activity diagram makes it easy to find faults in tasks and processes, thus helping you to improve your processes and in turn makes your system as efficient as possible.

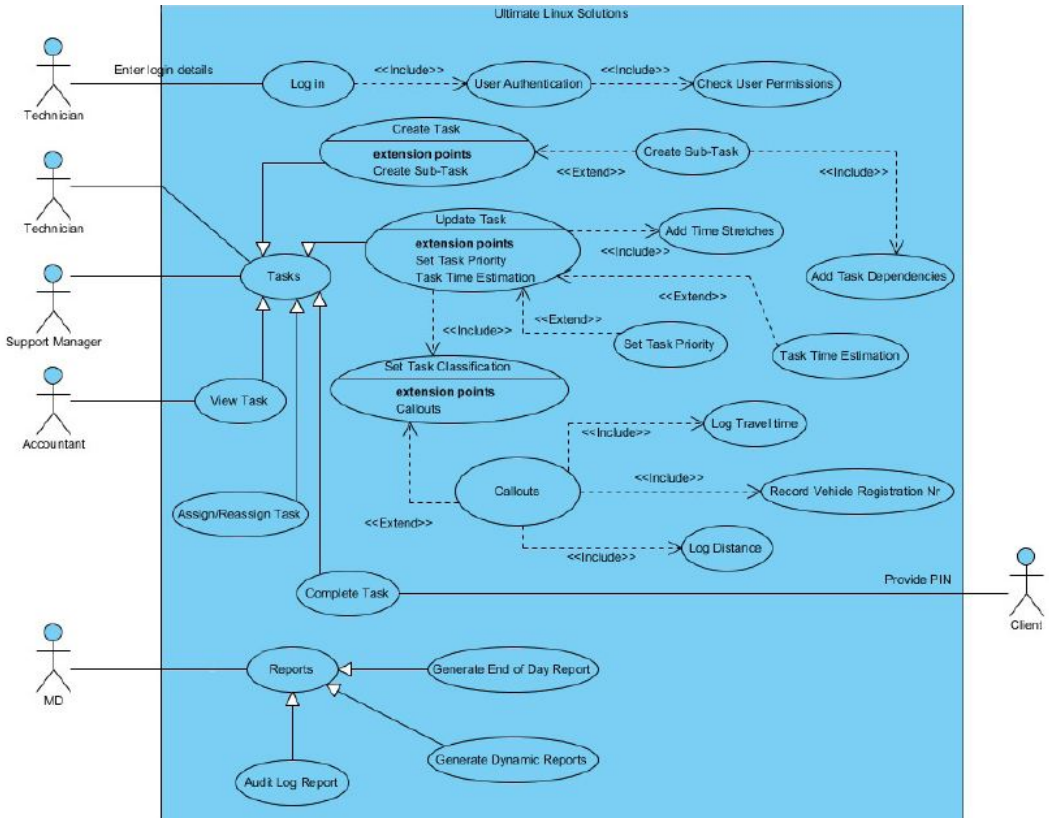
4.4.1 Deployment Diagram

The deployment diagram is used to show the physical components of the software. Here we show that a database server will be set up with will use the RDBMS to connect to MySql. Thought this we will be able to do queries from workstation programs of a mobile device application.



4.5 Scenarios

4.5.1 Use Cases Diagram



4.6 Component Diagram

The component diagram depicts how the different components of a system will interact with one another and how they are related. Within the diagram you have the different components and interfaces that components interface with, furthermore you also have the type of relationship that the components have with the different interfaces.

- **Scenarios:**
- **Logical View:**

- **Development View:**
- **Process View:**
- **Physical View:**

5 Tracibility Matrix

Requirement Traceability Matrix (Functional Requirements)												
	FRQ1	FRQ2	FRQ3	FRQ4	FRQ5	FRQ6	FRQ7	FRQ8	FRQ9	FRQ10	FRQ11	FRQ12
Test cases												
Test_case_1												
Test_case_2												
Test_case_3												
...												
Done												
Acceptance testing												
Acceptance_test_1												
Acceptance_test_2												
Acceptance_test_3												
...												
Accepted												

Requirement Traceability Matrix (Non-Functional Requirements				
	FRQ1	FRQ2	FRQ3	FRQ4
Test cases				
Test_case_1				
Test_case_2				
Test_case_3				
...				
Done				
Acceptance testing				
Acceptance_test_1				
Acceptance_test_2				
Acceptance_test_3				
...				
Accepted				

6 Policies

6.1 Performance

- **Response Times** The software should not take too long to load, it should take 2-3 seconds to load and refresh pages. It should also take the same amount of time when navigating through other pages.
- **Processing Times** Performing key functions like retrieving information and logging information should also take around 2-3 seconds.
- **Query and Reporting Times** Technicians should report to work during office hours and should a technician wish to work overtime, they should query within reasonable times.

6.2 Capacity and Scalability

- **Throughout** The system should be able to handle more than 5000 users using it at the same time.
- **Storage** The system's database should store more than 5000 different client's information
- **Growth Requirements** Depending on how often the system is used, it should be possible to extend it as time goes on

6.3 Availability

- **Hours of operation** The system should be available 24hour, 7 days a week and any person who needs to use it at any time should be able to use it.
- **Locations of operation** Since it has a mobile version, it should be used at any location that has a sufficient network signal for it to complete the necessary services.

6.4 Maintainability

- **Coding Standards**
  - **Names** Variable names should describe exactly what the variable is used for. E.g. studentName shows that the variable is used to store the name of a student.
  - **Comments** This is the internal documentation written by programmers next to the source code.The comments should describe exactly what each line of code or each method does.
  - **Format** The size of indentation should be consistent throughout the source code. The vertical alignment of open and closing braces should be the same. E.g.

```
for (i = 0; i < 100; i++){
            ...
}
```

6.5 Recovery

- **Restore time** When disaster strikes and affects the system's functionality, it should take 5-24 hours to recover from that particular disaster and get the system back to normal.

- **Backup time** Data should be backed up every 6 hours of each day if there are any updates to the data and the backing of data should take 1-2hours depending on how large the data that needs to be backed up is.

7 Glossary

- **Application:** Software that runs on a computer.
- **Thread:** Part of a program that can be executed independently and concurrently of other parts.
- **Infrastructure:** Physical hardware used to interconnect computers and users.
- **MVC:** Model View Controller
- **Network:** A horizontal or vertical connection between entities.
- **Policies:** Steps taken to reach a goal.
- **Scope:** A short overview with limitations and.or restrictions of the project.
- **Subsystem:** A system that is part of some larger system.
- **System:**Underlying processes working together to form a greater process.
- **Session:** Unique identifier created by the server.
- **UML:** Uniform modeling language.