Semi-Automated Querying & Statistical Reporting Web Application

A PROJECT REPORT

Submitted in partial fulfilment for the award of the degree of

MS

in

Software Engineering

By

V. Shravan, 10MSE0236

Under the Guidance of

R. Kiruba Thangam,

Assistant Professor (Sr.)

SCHOOL OF INFORMATION TECHNOLOGY AND ENGINEERING

VIT UNIVERSITY

VIT
UNIVERSITY
(Estd. u/s 3 of UGC Act 1956)
Vellore - 632 014, Tamil Nadu, India

DECLARATION BY THE CANDIDATE

I hereby declare that the project report entitled "**Semi-Automated Querying & Statistical Reporting Web Application"** submitted by me to VIT University, Vellore in partial fulfilment of the requirement for the award of the degree of **M. S. Software Engineering** is a record of bona fide project work carried out by me under the guidance of **Professor. R. Kiruba Thangam.** I further declare that the work reported in this project has not been submitted and will not be submitted, either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university.

Place: Vellore   Signature of the Candidate

Date:

VIT
UNIVERSITY
(Estd. u/s 3 of UGC Act 1956)
Vellore - 632 014, Tamil Nadu, India

SCHOOL OF INFORMATION TECHNOLOGY AND ENGINEERING [SITE]

SOFTWARE ENGINEERING DIVISION

BONAFIDE CERTIFICATE

This is to certify that the project report entitled **"Semi-Automated Querying & Statistical Reporting Web Application"** submitted by **Shravan. V (10MSE0236)** to Vellore Institute of Technology University, Vellore, in partial fulfilment of the requirement for the award of the degree of **M. S. Software Engineering** is a record of bona fide work carried out by him/her under my guidance. The project fulfils the requirements as per the regulations of this Institute and in my opinion meets the necessary standards for submission. The contents of this report have not been submitted and will not be submitted either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university.

Prof. R. Kiruba Thangam
Assistant Professor (Sr.)
Internal Guide
SITE, VIT University

Internal Examiner(s) **External Examiner(s)**

PayPal India Private Limited
Futura IT Park, Block A, First Floor,
No.334, Old Mahabalipuram Road, Sholinganallur,
Chennai - 600 119. India
Ph.: +91-44-66348000 Fax: +91-44-66348075

13 May 2015

BONAFIDE CERTIFICATE

This is to certify that the project work entitled **"Semi-Automated Querying & Statistical Reporting Web Application"** was duly carried out by **Mr. Shravan Venkataraman (10MSE0236),** in PayPal India Pvt. Ltd., towards the partial fulfilment of the requirement for the award of the degree of **MS Software Engineering** awarded by VIT University, is a bonafide work carried out by him under our supervision. The project fulfils the requirement as per the regulations of this Institute and in my opinion meets the necessary standards for submission. The contents of this report have not been submitted and will not be submitted either in part or in full, for the award of any other degree or diploma in this Institute or any other Institute or University.

We would like to inform that we are issuing this certificate only as a proof of internship and the undersigned/company is not responsible for any financial dealings of the above employee.

Should you require any clarification, please do not hesitate to contact the undersigned +65 6510 4560.

Thank you.

Yours faithfully,
For and on behalf of
PAYPAL INDIA PVT LTD

Mr. Rajeev Gupta
MTS-1, Project Lead

Acknowledgement

I wish to express my heartfelt gratitude to **Dr. G. Vishwanathan**, Chancellor, VIT University, Vellore for providing me with the facilities for the tenth semester project. I am highly grateful to our Vice Presidents, **Dr. V. Raju,** Vice chancellor, **Dr. K. Narayanan,** Pro-Vice Chancellor for providing necessary resources.

My sincere gratitude to **Prof. K. Ganesan**, Dean, School of Information Technology and Engineering, for giving us the opportunity to undertake the project. I wish to express my sincere gratitude to **Prof. R. Srinivasa Perumal**, Programme Chair of M. S. Software Engineering, School of Information Technology and Engineering and our project coordinator **Prof. Krishna Chandramouli** for providing me an opportunity to do my project work in the ***industry.***

I would like to express my special gratitude and thanks to my internal guide **Prof. R. Kiruba Thangam,** Assistant Professor (Sr.), School of Information Technology and Engineering whose esteemed guidance and immense support encouraged to complete the project successfully. I would also like to express my sincere gratitude to my team's manager **Ms. Ursula Chiang** and my on-site project lead & supervisor **Mr. Rajeev Gupta** and my mentor **Ms. Preethi Paramasivam** for helping me with the project.

I thank the Management of VIT University for permitting me to use the library resources. I also thank all the faculty members of VIT University for giving me the courage and strength I needed to complete my goals.

Place: Vellore

Date: V. SHRAVAN

Executive Summary

PayPal is one of the leading payment systems in the world. The regional merchants use PayPal APIs and integrate PayPal in their website for creation of business accounts, to check PayPal balance, etc. Managers help the regional merchants in all their technical issues like PayPal API failures, custom reporting, etc. To understand API Failures they need to query the database with corresponding tables and create reports for a particular date range. They have to run SQL queries to achieve the results repeatedly. Also, the retrieved data is in the SQL output format and it is hard for them to understand and analyse. The new web-app will simplify the job of the managers in helping the regional merchants solve the problems of API failure. It also makes repetitive tasks into semi-automated and more streamlined processes. The new web-app will provide a web interface to retrieve business data from PayPal Database. The web-app essentially allows the Managers to query the database through its web interface. The queries that are run are also recorded in local database log based storage. These queries that are recorded form the history module where-in a user can access the history of queries that were run. These historical queries can be re-run again. The history of queries ran by the other business managers can be monitored.

TABLE OF CONTENTS

List of Figures

List of Tables

CHAPTER-1

INTRODUCTION

PayPal is one of the leading payment systems in the world. The focus of my team is to interact with the business stakeholders and develop custom tools & plugins for their usage.

1.1 PROBLEM DEFINITION

The regional merchants use PayPal APIs and integrate PayPal in their website for creation of business accounts, to check PayPal balance, etc. Managers help the regional merchants in all their technical issues like PayPal API failures, custom reporting, etc. To understand API Failures they need to query the database with corresponding tables and create reports for a particular date range. They have to run SQL queries to achieve the results repeatedly. Also, the retrieved data is in the SQL output format and it is hard for them to understand and analyze.

1.2 PROPOSED WORK

The work proposed is to provide a web interface to retrieve business data from PayPal Database. The proposed web-app will consist of two main modules. The web-app will make use of two main classes of user roles namely administrators and end-users. The web-app essentially allows the managers to query the database through its web interface. The queries that are run are also recorded in local database log based storage. These queries that are recorded form the history module. Using this history module, a user can access the history of queries that were run. The app will also provide a facility to re-run a past query. The administrators will be provided with the capability to monitor the history of queries run by the other business managers, to add new queries, and to edit or delete existing queries.

The web-app also provides an email feature using which the end-users can provide their email so that the results of the queries can be emailed to them in the form of a '.csv' file. The web-app will provide the administrator access only to managers and other users will be given only end-user access. The path of the resultant csv files is stored in the local database table. The administrators will be provided with a search facility to search the business users through their LDAP (Local Directory Access Protocol) login email.

1.3 SCOPE

The user of the web-app should be a manager of a technical team in PayPal. The user should be logged in with corporate LDAP access. The app should be connected to the production database of PayPal with valid credentials.

1.4 PURPOSE

The purpose of this web-app is to provide web-interface for administrators and business users to query the live database and get periodical reports.

1.5 GOALS & OBJECTIVES

The goal of the web-app is to help the managers who use this app query the database and get the reports accordingly. The various objectives of the web-app are as follows.

i. To enable the managers to query the database for different parametric performance related data.

ii. To give managers an ability to download the results in the form of an excel file.

iii. To provide a feature for the managers to receive the results in their email as an excel attachment.

iv. To keep a history of queries run and to provide an ability to re-run the historical queries.

1.6 HARDWARE CONFIGURATION

The hardware configuration of the proposed system is as follows.

Processor/RAM/HDD	:	Intel i5/8GB RAM/ 240GB SSD
Web Server	:	Linux Server Version 12.04
Database Server	:	Teradata Database Server

1.7 SOFTWARE CONFIGURATION

The software configuration of the proposed system is as follows.

Operating System	:	Linux 14.04
DBMS	:	SQLite Version 3
Third Party Software	:	Putty, Win-SCP
IDE	:	Eclipse 4 or more with PyDev Plugin for Eclipse

1.8 FEASIBILITY STUDY

Feasibility study is carried out to realise if the project on completion will serve the purpose of the organization for the amount of work, effort and the time spent on it. Feasibility study lets the developer foresee the future of the project and the usefulness. A feasibility study of a system proposal is according to its workability, which is the impact on the organization, ability to meet their user needs and effective use of resources. Thus when a new application is proposed it normally goes through a feasibility study before it is approved for development.

The document provide the feasibility of the project that is being designed and lists various areas that were considered very carefully during the feasibility study of this project such as Technical, Economic and Operational feasibilities. They are as follows.

i. Technical Feasibility:

Technical feasibility study tests the level of technology used in the development of the system. To develop the **Teradata Query Web-App,** the developer should know Python based web-programming. He/she should be familiar with the Django web-framework built for python. Proper level-2 credentials are needed to test the app in its testing stage.

ii. Operational Feasibility

Operational feasibility studies the operational scope of the system. It is a measure of how well a proposed system solves the problems, and takes advantage of the opportunities identified during scope definition and how it satisfies the requirements identified in the requirements analysis phase of system development. **Teradata Query Web-App** retrieves the data from the production database and sends the same as a report in email to the user. This is feasible by making a connection between the web-app and the corporate email.

iii. Economic Feasibility

Economic feasibility evaluates the cost of developing the system. **Teradata Query Web-App** is developed in Eclipse ADT which is an open source tool. All the other technologies used in the creation of the web-app are open source too. So this project is economically feasible.

iv. Legal Feasibility

Legal feasibility determines whether the proposed system conflicts with legal requirements. E.g. a data processing system must comply with the local Data Protection Act. Data of the users are not accessible to anyone other than the users themselves and the administrators. So the legal precautions are taken care of and the project is deemed legally feasible.

v. Schedule Feasibility

A project will fail if it takes too long to be completed before it is useful. Typically this means estimating how long the system will take to develop, and if it can be completed in a given time period using some methods like payback period. Schedule feasibility is a measure of how reasonable the project timetable is. **Teradata Query Web-App** is estimated to be developed within 3 months.

1.9 MODELLING TECHNIQUES FOLLOWED

Teradata Query Web-App uses agile development model for software development process and follows RAD approach. Agile development model and RAD approach was chosen for development project. The benefits of the agile methodology are as follows.

i. Production of quality software
ii. Software that is delivered on time.

iii. Cost within the budget.
iv. Satisfies all requirements.

As the complete set of requirement of the application is not given initially and since there is a possibility of further changes in requirements, agile development model has been chosen. Agile development model is been chosen for following reasons:-

i. Changes in requirements over the development period.
ii. Deliverables will be refined at each stage.
iii. Rapid Development and client feedback.

1.10 REQUIREMENTS SPECIFICATION

The requirements specification of the proposed system is as follows.

1.10.1. Functional requirements

FR 1 – Request Module Functionality

FR 1.1 Run query

User should be able to select a query from the available list of queries and run that query.

FR 1.2 Download query results

User should be able to download the query results in the excel format.

FR 1.3 Receive results in email

The results of the query that is run should be sent as an attachment to the user's email id.

FR 2 – History Module Functionality

FR 2.1 Re-run query

The user should be able to re-run historical queries.

FR 2.2 Download past query results

The user should be able to download the historical query results.

FR 2.3 Run Query

The user should be able to modify the parameters of a historical query and run it again.

FR 2.4 Download re-run query results

The user should be able to download the results of the re-run queries.

FR 3 – User Management Functionality

FR 3.1 Request access

The users should be able to request for access to the web-app.

FR 3.2 Approve/Reject access

The administrator should be provided a facility to approve or reject a user that has requested access to the web-app.

FR 3.3 Add/Edit/Delete user

The administrator should be given a facility to explicitly add a user, edit the access level of a user and to delete a currently existing user.

1.10.2. Requirements out of scope

The project scope does not include

- Storage of personal details of the business users that use this system.

- Deleting the local storage files that are older than 90 days.

1.10.3. Non Functional Requirements:

NFR 1. Performance Requirements

- The average response time should be optimal

- The project should follow model-view-controller architecture

- The app should respond fast while accessing the request and history modules.

NFR 2. Safety Requirements

Safety precautions should be taken to make sure that the required rows are fetched from the live database, and should not be updated. READ ONLY

NFR 3. Security Requirements

Security precautions should be taken for the following:
- The Teradata access credentials should be kept encrypted.

- The Teradata Query Web-app is intended for PayPal internal usage.

- A configuration file should be maintained by the technical account managers to add or delete admins.

NFR 4. Software Quality Attributes

- Availability: The app should be available only to PayPal internal users.

-Maintainability: The design of the app should be cohesive and should have loose coupling.

- Reusability: The project modules should follow object oriented methodology to make them reusable.

- Usability: The app is intended to be used by the managers with good technical knowledge as well as the business stakeholders with less technical expertise.

1.10.4. External System Interface Requirements

External interface requirements specify the requirements related to the interface provided in the application to interact with the systems external to it. The table 1.1 lists the external system interface requirements.

Table 1.1 External System Interface Requirements

External Systems name	**Deliverables to External Systems**	**Deliverables From External Systems**
LDAP Email Directory	Valid login id and password	Authentication message as valid or invalid

1.11. CHALLENGES FACED

One of the challenges that a user might face in this application is with regarding the live production database. The credentials keep changing every 90 days and because of that, the maintenance end has to take the web-app down for a day or so in order to add the new set of credentials and test the system according to the new set of credentials.

CHAPTER-2

LITERATURE SURVEY

2.1 BACKGROUND

Managers at PayPal regularly query the PayPal live production database for data related to the performance of merchants, the PayPal APIs that the merchants have integrated with their business sites, performance of other managers under them, etc. This querying process now takes place as a command line operation wherein the manager types in the raw SQL query and obtains the results in a raw SQL output format. This process is very tedious and is also very time consuming. Over a period of time, this task gets very repetitive.

This application uses the MVC design pattern. There are different applications to automate the process of querying, but, with respect to PayPal's internal system of querying and reporting is different. Therefore, the system that should be developed in order to achieve the automation.

2.2 SURVEY OF EXISTING WORKS

The Model-View-Controller (MVC) design pattern is very useful for architecting interactive software systems. This design pattern is said to be partition-independent as it is expressed in terms of an application running in a single address space. The web-app development is complicated by the fact that the developers are all advised to partition the application as early as in the design phase [12].

The MVC Architecture divides the system in three different layers that take care of the interface control logic and data access individually resulting in facilitating easier maintenance of the web-app modelled using the MVC architecture [11].

Most frameworks enforce MVC separation partially. They do not enforce MVC end-to-end. Django focuses on the server side. The programmer has to write glue code in order to propagate model changes to views and to co-ordinate front-end UI changes with back-end models [5].

Django has an interesting example of a core library; it auto generates an administrative section for a website project. Django uses "Fixtures" which are sample data sets used for the tests which are easy to make. Django allows the usage of third party unit testing libraries [2].

Users can gain insights and trust into service applications by leveraging trust in a neutral third party: a cloud provider that hosts application services on an infrastructure and platform that it controls. A trusted cloud provider will act as a source of trust and help the clients to host their services in the cloud [9].

The large number of database systems having query and transaction processing eventually and naturally led to the need for data analysis and understanding. Hence, due to this increasing necessity data mining started its development. The process reveals hidden patterns that can’t be detected using traditional query and OLAP tools. So there is “Teradata Solution”, unique approach to counselling and our optimal use of data warehouse technology [8].

2.3 PROPOSED SYSTEM

The work proposed is to provide a web interface to retrieve business data from PayPal Database. The proposed web-app will consist of two main modules. The web-app will make use of two main classes of user roles namely administrators and end-users. The web-app essentially allows the Managers to query the database through its web interface. The queries that are run are also recorded in local database log based storage. These queries that are recorded form the history module where-in a user can access the history of queries that were run. The app will also provide a facility to re-run a past query. The administrators will be provided with the capability to monitor the history of queries run by the other business managers, to add new queries, and to edit or delete existing queries.

The web-app also provides an email feature using which the end-users can provide their email so that the results of the queries can be emailed to them, in the form of a ‘.csv’ file. The web-app will provide the administrator access only to Managers and other users will be given only end-user access. The path of the resultant csv files is stored in the local database table. The administrators will be provided with a search facility to search the business users through their LDAP (Local Directory Access Protocol) login email.

2.4 ASSUMPTIONS

The assumptions are as follows:

i. The user is accessing the web-app from within the corporate internet.
ii. The user has valid access credentials to login into the internal corporate internet.
iii. Valid credentials are used to access the corporate production database.
iv. The server cloud is up and running.

2.5 LIMITATIONS

i. A user can run only one query at a time.

ii. If the report file exceeds 10mb, it will only be downloadable and the result file will be split into different parts when sent in email.

CHAPTER-3

SYSTEM ARCHITECTURE

3.1 DESIGN DESCRIPTION

The design of the actual web-app follows the Model View Controller (MVC) design pattern. The user interfaces are prone to change requests. The user interface of a long-lived system is a moving target. Building a system with the required flexibility will be expensive and error-prone if the user interface is tightly interwoven with the functional core. This can result in the development and maintenance of several substantially different software systems one for each user interface implementation. When developing such an interactive software system, the changes to the user interface should be easy and possible at run- time. Also, adapting or porting the user interface should not impact code in the functional core of the application.

To solve the problem, the MVC pattern divides an interactive application into three areas such as processing, output and input. The model component encapsulates core data and functionality. The model is independent of specific output representations or input behavior. The view components display information to the user. A view obtains the data it displays from the model. There can be multiple views of the model. Each view has an associated controller component. Controllers receive input, usually as events that denote mouse movement, activation of mouse buttons or keyboard input. Events are translated to service requests, which are sent either to the model or to the view. The user interacts with the system solely via controllers.

The separation of the model from the view and controller components allows multiple views of the same model. If the user changes the model via the controller of one view, all other views dependent on this data should reflect the change. To achieve this, the model notifies all views whenever its data changes. The views in turn retrieve new data from the model and update their displayed information.

3.2 OMT CLASS DIAGRAM

The MVC Pattern is used as the basic framework on which the web-application is built. It contains three parts namely Models, Views and Controllers also called as Templates with respect to the Django architecture. The observer interacts with the views using the templates. The fig 8.1 of Appendix A describes the classes in the MVC Pattern and their interaction functions.

3.3 DJANGO MVC ARCHITECTURAL DIAGRAM

The fig 8.2 of Appendix A describes the interaction between various components in the Django system. The Django system's HTTP Request handler takes care of all the protocol requests and communicates with the MVC related files and retrieves the response for the user.

3.4 MVC PATTERN ARCHITECTURE

The View Part consists of the user interface design of the web-app. The model part consists of the database and the database tables used in the web-app. The controller part consists of the business logic written to act as a mediator in controlling the data (model part) as well as updating the changes in the views. The system design of Django based on MVC is given in the fig 8.3 of Appendix-A.

3.5 HIGH LEVEL DESIGN

In the Teradata Query Web-App, the application basically runs the queries against the Teradata live database and pulls the results. Once the results are all acquired, it then populates the excel file and sends an email to the user along with that excel file as an attachment.

You can find the high level design of the system being represented by the fig 8.4 of Appendix-A.

3.6 DATA FLOW DIAGRAM

DFD stands for Data Flow Diagram is a preliminary step used to create a system which can be further elaborated. DFD is the graphical representation of the flow of data in the system.

The data flow diagrams for the Teradata Query Web-App can be found in fig 8.5, 8.6 and 8.7 of Appendix-A.

3.7 UML DIAGRAMS

UML stands for Unified modelling Language. It is one of the powerful design methodologies used to design the system. UML diagram provides a standard way to visualise the system. This document contains the different UML diagrams which show the interaction between different components of the system.

3.7.1 Use Case Diagram

The fig 8.8 of the Appendix-A denote the use case diagram for the web-app. The Business users log in through LDAP login, chooses the query they want to run, and passes the required field parameters. Then they choose run query. The results are returned to them in e-mail in the form of a CSV File. The administrators can add, edit, and delete the queries that business users run.

The queries that are run will fetch the data from the database, while the log will be stored in the local storage SQLite3 database. Any addition, deletion, edit done to the allowed queries will be saved in the local database table.

The use case catalogue is given in the table 3.1 below.

Table 3.1 Use-case catalogue

No.	Use Case	Complexity	Priority
1.	**Name**: Login **Description**: User must log into the web-application	Medium	High
2.	**Name**: Choose query **Description**: Chooses one of the various queries available.	Medium	High
3.	**Name**: Supply field parameters **Description**: Supply the necessary parameters for the query field values.	Low	High
4.	**Name**: Run query **Description**: Runs the query	Medium	High
5.	**Name**: Add query **Description**: Administrator adds a query.	Low	High
6.	**Name**: Edit query **Description:** Administrator edits an existing query.	High	High
7.	**Name**: Delete query **Description**: Deleting an existing query.	Low	High

3.7.2 Class Diagram

The class diagram is represented as given in the fig 8.9 of appendix-A. There are three main classes – the Web-App class, the user class and the data-store class. They are all further divided into many other classes. The Web-App class also consists of an authenticator class. The data store class is specifically divided into Production database and Local database. User class consists of Administrators and Business Users.

3.7.3 Activity Diagram

To add query, the administrator should supply field names and their types and submit the addition. Delete query deletes the existing query while edit query allows the administrator to edit the query parameters and data types of those query parameters represented in the form fields. The fig 8.10 of Appendix-A represents the various activities and the flow associated with those activities.

3.7.4 State chart Diagram

A state chart diagram represents the various states that the system can have and predominantly used to show the dynamic nature of the system. Refer the fig 8.11 of Appendix-A for the state chart diagram of the Teradata Query Web-App.

3.7.5 Collaboration Diagrams

The fig 8.12 and 8.13 give you the sequence and collaboration diagrams for the Teradata Query Web-App. Collaboration diagrams basically are used to represent the collaboration between various objects and actors in the system while sequence diagrams show the sequence of the actions performed in the system by the user.

3.7.6 Package Diagram

The fig 8.14 shows the package diagram for the Query Web-App. Since the application follows the MVC pattern, the packages are also divided into models, views and controllers and the sub-packages are separated accordingly.

CHAPTER-4

EXPERIMENTAL EVALUATION

4.1. TECHNOLOGY USED

4.1.1 Python

Python is a widely used general-purpose, high-level programming language. Its design philosophy emphasizes code readability, and its syntax allows programmers to express concepts in fewer lines of code than would be possible in languages such as C++ or Java. The language provides constructs intended to enable clear programs on both a small and large scale. Python supports multiple programming paradigms, including object-oriented, imperative and functional programming or procedural styles. It features a dynamic type system and automatic memory management and has a large and comprehensive standard library.

This project utilizes python because of few very important factors. First factor that played a huge role in the choice of python is the ease of web-programming with different other features. The second factor is the ability to utilize cross platform python based web-framework called as Django.

4.1.2 SQLite

SQLite is an Open Source Database which is embedded into Django. SQLite is preinstalled in the corporate cloud linux server. SQLite require very less memory at runtime (approx., 250 KByte). Externally SQLite database creates file system so this may be slow, Therefore it is recommended to perform database operations inside the Django's models.

4.1.3 Django

Django is a web development framework that saves development time and gives the developers the ability to create and maintain high-quality web applications with minimal fuss. Django lets you focus on the crux of the web application and also provides high level abstraction of common web development patterns.

4.2. IMPLEMENTATION

4.2.1 Query Manipulation Module

This module is made use of, by the administrator to add/edit/delete queries to make them available for the business users. The business users can then use this module to run the desired query for their purpose. The snapshot of the module is shown below in fig 4.1.

4.2.1.1 Download Module

A download option is given with the Run-Query module which gives the user an option to download the report created out of the query output. Downloads can be in two formats - .csv or .xls.

4.2.1.2 E-Mail Module

The application also contains an e-mail module (the email field given in fig 4.1). The users can use this feature to get the generated reports directly in their inbox as an attachment to the email, with the details and parameters of the query that was run.

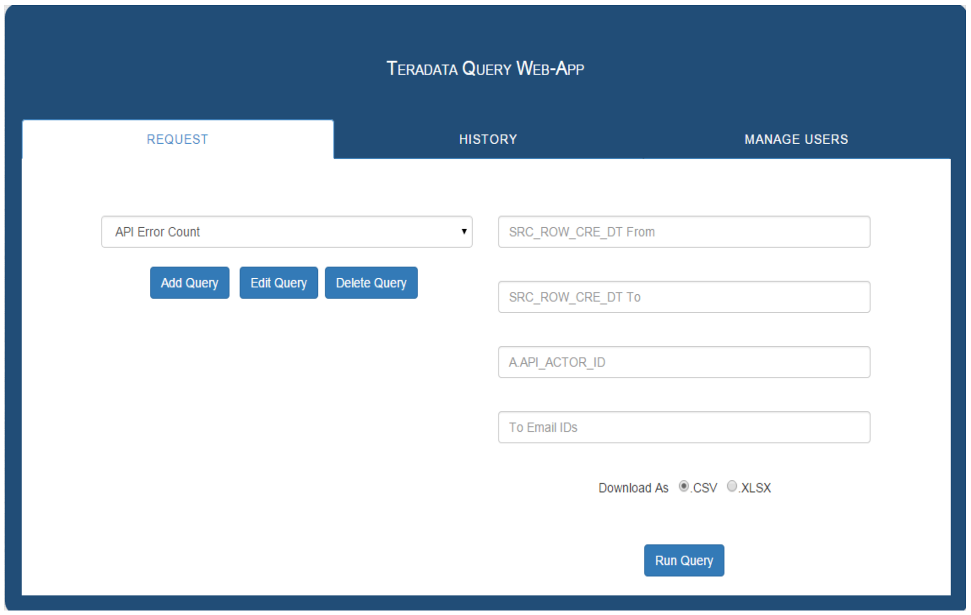
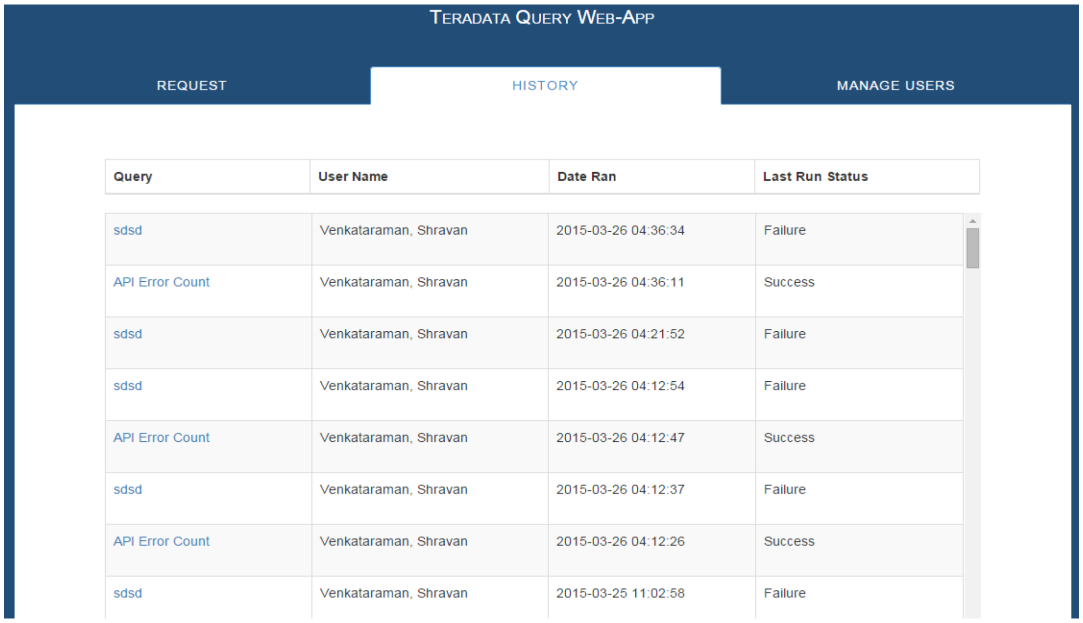


fig 4.1 – Query Manipulation Module

4.2.2 History Module

The history module displays the history of all queries run by a particular business user. A business user can only see his own history while the administrator can see everyone’s history of queries that they have run. The snapshot of the module is shown below in fig 4.2.



Query	User Name	Date Ran	Last Run Status
sdsd	Venkataraman, Shravan	2015-03-26 04:36:34	Failure
API Error Count	Venkataraman, Shravan	2015-03-26 04:36:11	Success
sdsd	Venkataraman, Shravan	2015-03-26 04:21:52	Failure
sdsd	Venkataraman, Shravan	2015-03-26 04:12:54	Failure
API Error Count	Venkataraman, Shravan	2015-03-26 04:12:47	Success
sdsd	Venkataraman, Shravan	2015-03-26 04:12:37	Failure
API Error Count	Venkataraman, Shravan	2015-03-26 04:12:26	Success
sdsd	Venkataraman, Shravan	2015-03-25 11:02:58	Failure

fig 4.2 History Module

4.2.3 User Management Module

This is the module exclusively available only to the administrators. The administrators will use this module to add/edit/delete business users. They can also use this module to add other administrators. The snapshot of the module is shown below in fig 4.3.

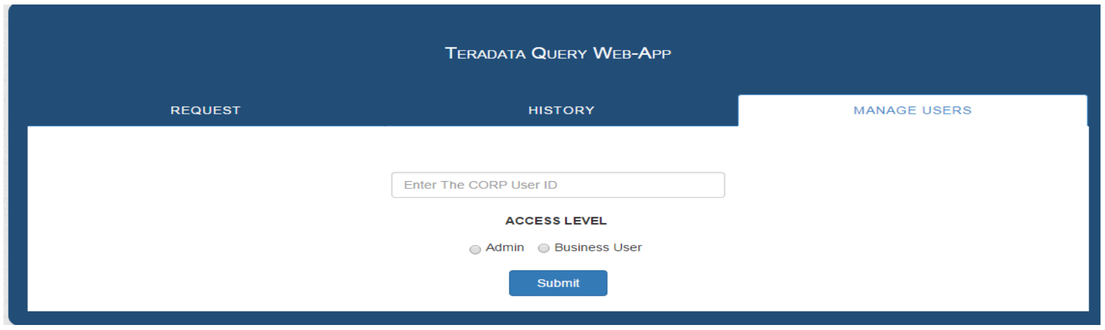


fig 4.3 – User Management Module

4.2.4 Authenticator Module

This module is used to authenticate the users and check the user access level. Based on the user access level, the user will be redirected to the appropriate interface. The snapshot of the module is shown below in fig 4.4.

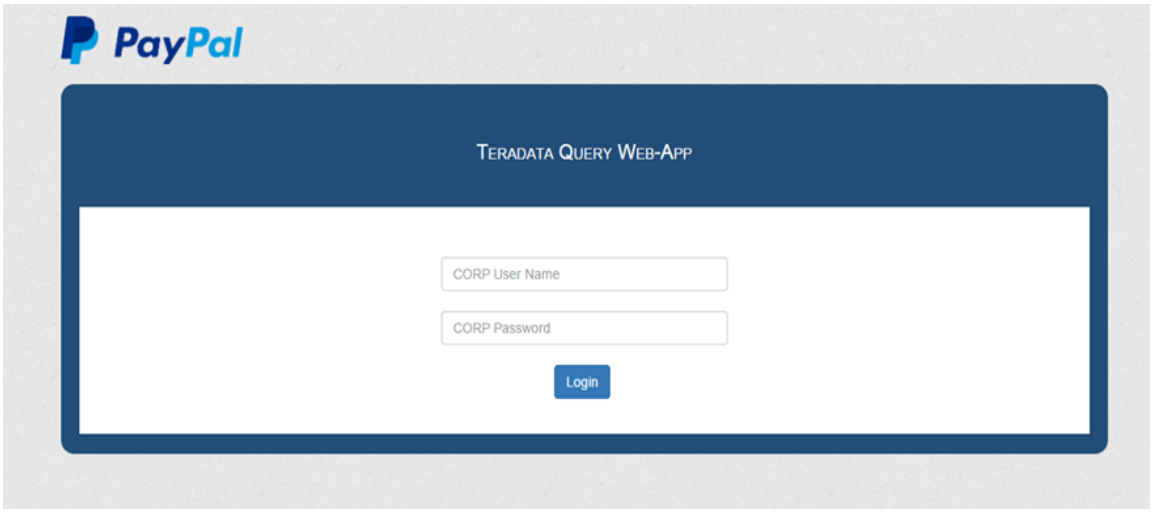
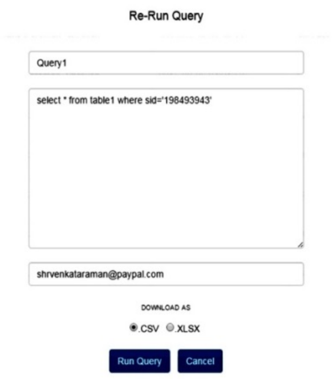


fig 4.4 Authenticator Module

4.2.5 Re-Run Module

The Re-Run module helps facilitate the re-run of the queries with the parameters exactly as they ran in the past by the user. This option is available with the history module which can be made use of to retrieve reports from the earlier run queries. The snapshot of the module is shown below in fig 4.5.


Re-Run Query

Query1

```
select * from table1 where sid='198493943'
```

shrvenkataraman@paypal.com

DOWNLOAD AS

[✓ .CSV] .XLSX

fig 4.5 Re-Run Module

4.3. LAYERING AND PARTITIONING

Teradata Query Web-App is divided into 3 partitions (functionalities) namely request functionality, history functionality and user management functionality.

In the request functionality, a user can choose and run a query from the available list of queries. Once the query is run, the user can download the result file. The web-app automatically sends an email to the user with the result file as an attachment to it.

In the history functionality, a user can view the queries that were already run by him/her and can also download the result files of those queries. The user can also re-run those queries after editing the parameters and obtain fresh results.

In the user management functionality, an administrator can approve or reject user access requests, add new user, edit user access level or delete an existing user.

4.4. CODING AND UI STANDARDS

The following coding standards and UI standards are expected.

i. Usage of Camel Casing wherever possible.

ii. Usage of comments wherever required.

iii. Usage of Indent Style Conventions.

iv. Usage of code tuning techniques.

v. Usage of software engineering practices for development of the application.

4.5. SYSTEM TEST PLANNING

The Teradata Query Web-App is tested based on the system test plan prepared. All the functions of the application are integrated and tested as a whole.

4.6. DEVELOPMENT ENVIRONMENT

The Teradata Query Web-App is a Django based web-application which is developed using Eclipse Juno and the database is developed using SQLite3.

4.7. SOFTWARES USED

The softwares used include Eclipse Juno, SQLite version 3 and Pydev plugin for Eclipse.

4.8. DEPLOYMENT

The various deployment activities to be included are given below.

4.8.1 Release

The Teradata Query Web-App will be released at the time of generation of its transfer into the corporate cloud server.

4.8.2 Install and Activate

The Teradata Query Web-App will be available for access in the server address that it is deployed to.

4.8.3 Update

The Teradata Query Web-App's updates will be released at successful response of the application.

4.9. OPERATIONAL MANAGEMENT

4.9.1. Performance Engineering

There are two roles namely administrator and business user. Administrator manages the other users, and the business users are only allowed to access the app for querying purposes. The above set of roles and the related activities of the Teradata Query Web-App are performed at every phase of the systems development life cycle.

4.9.2. Exception Handling / Logging Mechanism

Exception Handling is done to handle the exceptions and print the appropriate exception messages at the time of execution of the Teradata Query Web-App and for errors an error logger file is updated periodically with appropriate error messages.

4.9.3. Security Mechanism

Since this web application is internal to the corporate, it has security mechanisms intact in order to make sure that the user who accesses the web-app are validated before they could exercise their access roles.

4.9.4. State & Session Management

The user's session is created whenever the user logs into the web-application. User's session and state are captured and stored each time the user accesses the web-application.

4.9.5. Data Access Mechanism

All the user data and information is persistent into a database. The Django framework used to perform the data access mechanism is the Django File system.

CHAPTER-5

TESTING

5.1 TEST PLAN

A test case in software engineering is a set of conditions or variables under which a tester will determine whether an application or software system is working correctly or not. The mechanism for determining whether a software program or system has passed or failed such a test is known as a test oracle. In some settings, an oracle could be a requirement or use case, while in others it could be a heuristic. Test cases are often referred to as test scripts, particularly when written. Written test cases are usually collected into test suites.

5.1.1 Unit Test Plan

The unit test plan specifies the plan for the testing of the individual units of the web-application. The following tables 5.1, 5.2, 5.3, 5.4, 5.5 represent the different unit tests devised for testing the different units of the web-application.

Table 5.1 Test Case 1

Test Case ID: Case_1	**Test Designed by:** Shravan
Test Priority (Low/Medium/High): High	**Test Designed date:** March 3, 2015
Module Name: PayPal Login Screen	**Test Executed by:** Ramakrishnan
Test Title: Verify login with CORP username and CORP password	**Test Execution date:** March 3, 2015
Description: Test the Web-App login page	
Pre-conditions: User has valid CORP username and password	
Dependencies: Dependency on CORP Local Directory.	

Step	Test Steps	Expected Result	Actual Result	Status
1.	Navigate to login page	User should be able to login	User is navigated to the respective dashboard. The	Pass
2.	Provide valid username			

3.	Provide valid password		dashboard shown is according to the access level of the user	
4.	Click on Login button			

Post-Conditions
After successful login of the user, the session details are logged in the local database.

Table 5.2 Test case 2

Test Case ID: Case_2	**Test Designed by:** Shravan
Test Priority (Low/Medium/High): High	**Test Designed date:** March 3, 2015
Module Name: History	**Test Executed by:** Ramakrishnan
Test Title: Test History Module	**Test Execution date:** March 3, 2015
Description: Test the history page	
Pre-conditions: User has logged into the web-app dashboard.	
Dependencies: Dependency on local history database and on prior queries run by the user.	

Step	**Test Steps**	**Expected Result**	**Actual Result**	**Status**
1.	Navigate to history page	Users are shown their query history according to their access level	The users are shown their history alone if the person accessing is a business user and everyone's history if it is an administrator.	Pass
2.	Hover on query name	Display the query	The query is displayed	
3.	Click on the query	Query re-run popup is displayed	The Query re-run popup is displayed	

Post-Conditions
The query re-run popup displays with the auto-populated values of that particular query.

Table 5.3 Test Case 3

Test Case ID: Case_3	**Test Designed by:** Shravan
Test Priority (Low/Medium/High): Medium	**Test Designed date:** March 3, 2015
Module Name: SQL Parser sub-module	**Test Executed by:** Ramakrishnan
Test Title: Test Add Query SQL Parser	**Test Execution date:** March 3, 2015
Description: Test the add query page	
Pre-conditions: User has logged into the web-app dashboard.	
Dependencies: Dependency on local query database. Only administrators can add a query.	

Step	Test Steps	Expected Result	Actual Result	Status
1.	Click on Add Query	User is taken to the add query page	The user is taken to the add query page.	Pass
2.	Enter query to add	Disable add button until the query is a valid select query	The add button is enabled when the user enters a valid select query.	
3.	Click on add button	User is taken to the field-generator page.	The query is parsed and fields are generated in the field-generator page.	
4.	Choose field types and click on the submit button in the page.	The query is submitted along with the fields and success message is displayed.	Query submission success message is displayed.	

Post-Conditions
The user should be able to see the added query reflected in the homepage list of queries.

Table 5.4 Test Case 4

Test Case ID: Case_4	**Test Designed by:** Shravan
Test Priority (Low/Medium/High): High	**Test Designed date:** April 3, 2015
Module Name: History Re-Run	**Test Executed by:** Ramakrishnan
Test Title: Test history re-run functionality	**Test Execution date:** April 4, 2015
Description: Test the history re-run popup	
Pre-conditions: User has navigated to the history page and clicked on a query.	
Dependencies: Dependency on local history query database.	

Step	Test Steps	Expected Result	Actual Result	Status
1.	Click on the Cancel Button	The re-run pop-up must close.	The re-run popup closes.	Pass
2.	Edit the query fields values and click on the re-run button.	The query is re-run with the updated parameter values.	Query is successfully re-run.	
3.	Click on download results button of the query	Historical result file of the selected query must be downloaded.	The result file of that query is downloaded.	
4.	Leave one of the fields empty in the re-run popup and click re-run.	Error message is thrown under the empty field to be filled.	An error message is shown for the field to be filled.	

Post-Conditions
After the re-run, the user should be able to see the button to download the fresh results of the query-rerun and the user should also receive an e-mail of the fresh query re-run.

Table 5.5 Test case 5

Test Case ID: Case_5	**Test Designed by:** Shravan
Test Priority (Low/Medium/High): High	**Test Designed date:** April 3, 2015
Module Name: User Management	**Test Executed by:** Ramakrishnan
Test Title: Test user management functionality	**Test Execution date:** April 4, 2015
Description: Test the manage users page	
Pre-conditions: The administrator has logged into the web-app and navigated to the user-management page.	
Dependencies: Dependency on the local database where the user details are stored.	

Step	Test Steps	Expected Result	Actual Result	Status
1.	Select requested users and click on approve.	Users are added to the approved users list and the user is notified of the addition.	The approved users are added to the database, are given access, and also notified to their mails.	Pass
2.	Select requested users and click on reject	Users are not added to the approved users list and the user is notified of the rejected access.	Users are removed from the request database and notified through email.	
3.	Type a valid user id and click on add	User is added and notified	The user added is reflected and is also sent a mail.	

4.	Select the users and click on delete.	The user is deleted and notified of the removal.	The user is deleted and is notified	

Post-Conditions
After any user management operation has happened, both the user and the admin will be sent a notification of what happened in their email inbox.

5.1.2. Integration Test Plan

5.1.2.1. Integration Test Environment

i) Hardware

Processor/RAM/HDD	: Intel Pentium/ 8GB RAM / 240GB SSD
Web Server	: The corporate network connection is recommended
Database Server	: PayPal Live Database Server

ii) Software

Operating System	: Windows/Linux/Mac
DBMS	: SQLite Version 3

5.1.3 Integration Test Parameters

5.1.3.1. Critical Modules

The critical modules for the integration test are as follows.

i) Sending the results of the queries in email as an excel attachment

ii) Giving the user an option to download the results in the form of a excel file.

5.1.3.2. Interfaces among modules

The interfaces among the various modules of the web-app are as given below.

i) Interface between adding and removing users, approving and rejecting users.

ii) After running a query, the user is redirected to the home page.

iii) After re-running a historical query from the re-run popup, the user is taken to the history page.

5.1.3.3. External interfaces

The data of the query results are obtained from the PayPal production database and sent to the user in the form of mail.

5.1.4 Integration Test Procedures

5.1.4.1. Order of integration

The request module is first integrated together. Then, the request module is integrated with the history module. Once the entire request and history part is tested, the authenticator module is integrated alongside and the application is tested as a whole.

5.1.4.2. Activities, techniques, tools

It involves addition and removal of users manually from the application. A test user is given the admin access for the purpose of testing the web-application.

5.1.4.3. Test Execution procedures

Once the application is deployed in the test server environment, the application will be tested using the test applications and the results will be identified.

5.1.4.4. Test result checking method

The observed results will be compared with the desired results and any discrepancy will be noted down.

5.1.5 Defect Tracking and Management

The various steps involved in defect tracking and management are as follows.

i) Ensure that the web-app is up and running after it is deployed into the final corporate cloud server.

ii) Check whether the initial administrators have received an email giving them their access credentials.

iii) Checking if the web-app is accessible only from the internal corporate network and not from internet outside the corporate environment.

5.1.6 Test stop criteria

Testing is stopped when the predefined test criteria are fulfilled and all the test cases are executed as expected. The calibrations on improvements will be done once the site is rolled out as the feature releases in the next version.

5.2 TEST REPORT

5.2.1. Test Plan Coverage

80 percent of test plan completed.

5.2.2. Code Coverage

The execution of Teradata Query Web-Application has been monitored and the degree of coverage at the statement, branch, or path level is tested.

5.2.3. Requirement Coverage

Monitoring and reporting on the number of requirements tested, and whether or not they are correctly implemented. All the requirements of Teradata Query Web-Application has been tested and correctly implemented.

5.2.4. Test Status Metrics Used

The following test status metrics are used to analyze the system performance during the testing phase.

i) Metrics Unique to Test are Defect Removal Efficiency, Defect Density, and Mean Time to Last Failure.

ii) Complexity Measurements are the quantitative values accumulated by a predetermined method, which measure the complexity of a software product.

iii) Project Metrics represent the status of project including milestones, budget and schedule variance and project scope changes.

iv) Size Measurements are methods primarily developed for measuring the software size of information systems, such as lines of code, and function points. These can also be used to measure software testing productivity. Sizing is important in normalizing data for comparison to other projects.

v) Defect Metrics which include the values associated with numbers or types of defects, usually related to system size, such as “defects/1000 lines of code” or “defects/100 function points”; severity of defects, uncorrected defects, etc.

vi) Product Measures are those metrics that represent the measures of a product’s attributes such as performance, reliability, failure, usability.

5.2.5 Quantitative Analysis

In order to present the findings of the testing to the customer, the team has to do a lot of analysis on the various data collected during the testing. So it is considered better that they be presented in the report to support conclusions, reduce rework effort on the same kind of analysis that the development team may end up doing and to implicitly suggest the areas/process that require improvement.

5.2.5.1 Defects Related

i) Defect distribution by status, build, severity and resolution.
ii) Defect analysis based on stage injected (Requirements, HLD, DLD, Build etc.)
iii) Defect analysis based on defect category (like functional, regression, performance, GUI, etc.)

The defects in the web-application are tested according to the defects related metrics mentioned above and the results of the defects testing are tabulated. Refer the table 5.6 for the results of the defects testing process.

Table 5.6 Defects testing status

Sl #	Closing Resolution	State	Count
1.	Software Changed	Analyze	1
2.	Software Changed	Closed	30
3.	Defer to next version	Closed	4
4.	Not to be Fixed	Closed	10
5.	Cannot reproduce	Closed	2
6.	Configuration change	Closed	4
Total			**51**

5.2.5.2 Final problem resolution status

The testing is done and the problems that were found in the web-application during the process of testing were fixed iteratively. Refer to fig 5.1 for the defects testing problem resolution results.

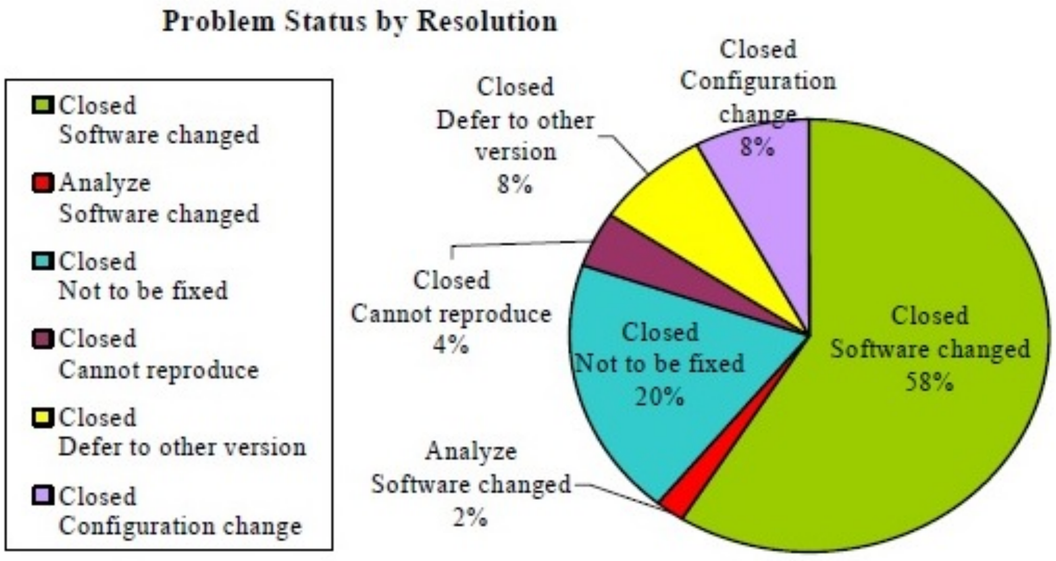


fig 5.1 Defects testing problem resolution chart

5.2.5.3 Query execution performance report

The following charts report the performance of the web-app during load testing and response time testing. The fig 5.2 represents the performance comparison between load and execution time. The fig 5.3 represents the performance comparison between the response time and execution time.

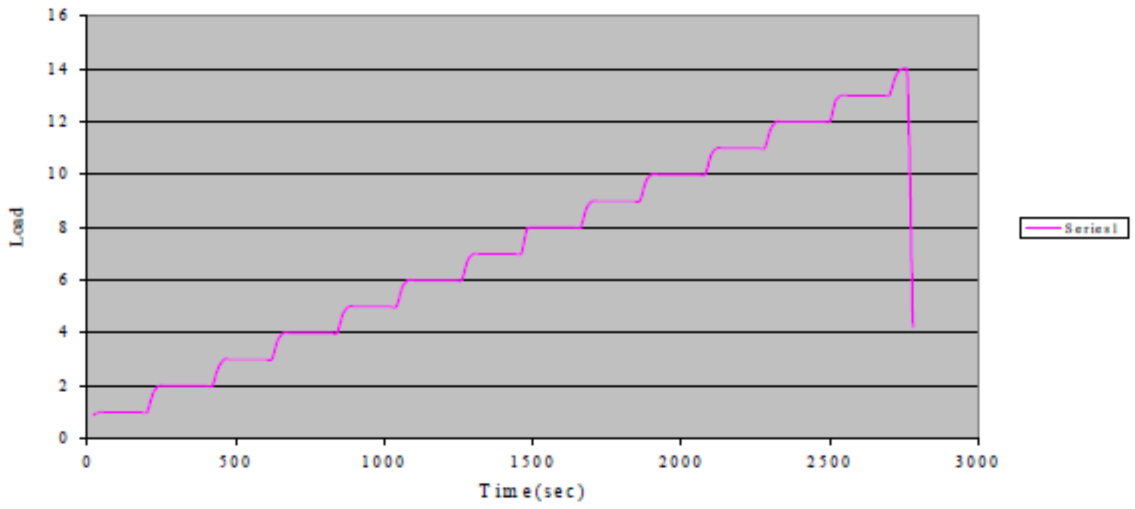


fig 5.2 Load Vs Execution Time

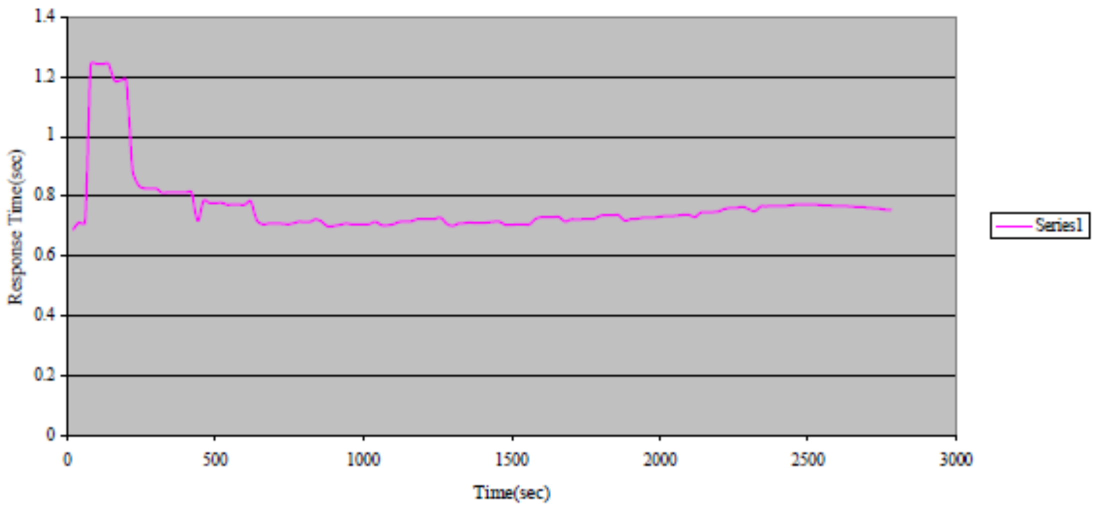


fig 5.3 Response Time vs Execution Time

5.2.6 Integration Test report

5.2.6.1 Subsystem Name

Teradata Query Web-App has three parts- the request part, the history part, and the user management part. These are all integrated and tested accordingly.

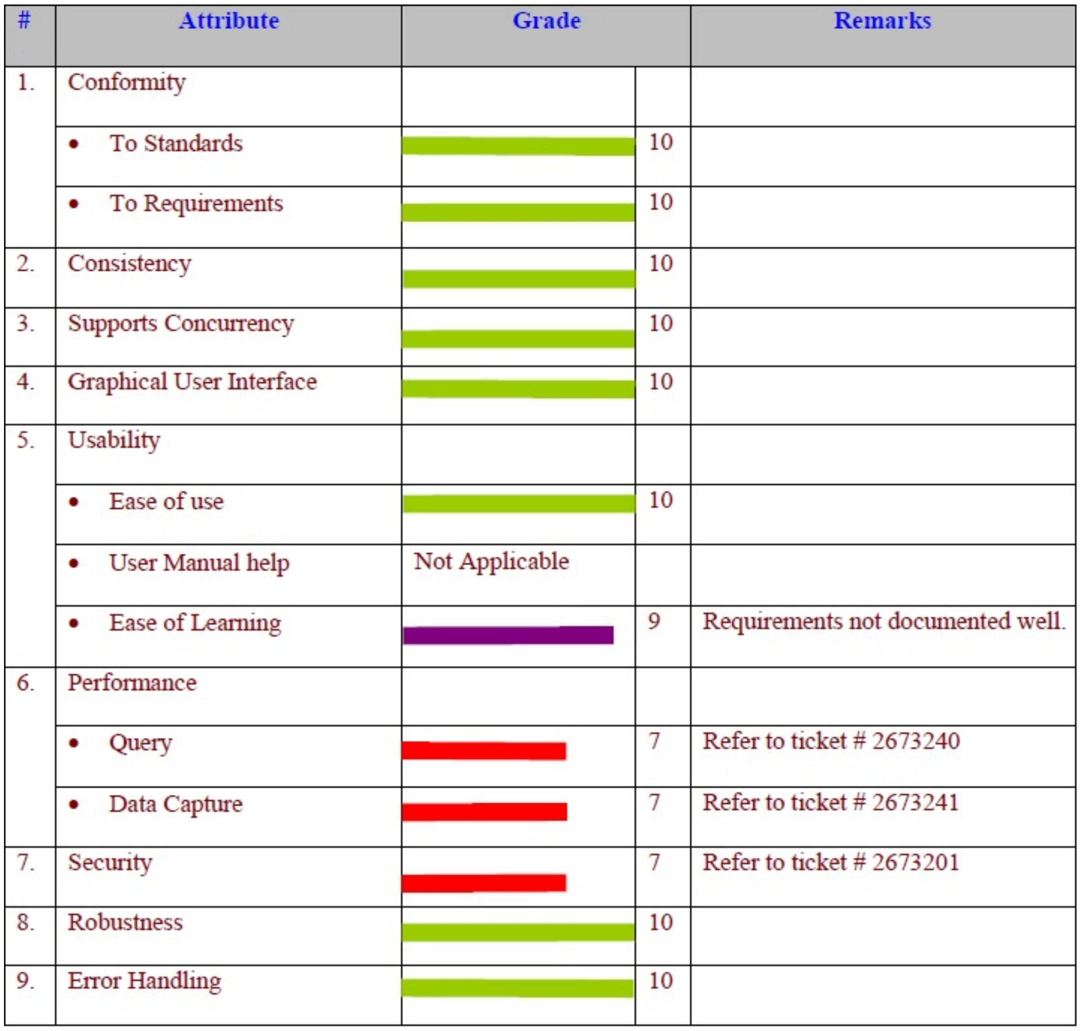
5.2.6.2 Functionality Tested

The functionalities tested include the downloading and emailing part, history re-run part and the user add/edit/delete functionalities.

5.2.7 Qualitative Grading

Once the testing is done, there is a need to know which features work and which features don't. Once this aspect is tested, the test results are qualitatively analysed and graded. Refer table 5.7 for the qualitative grading results.

Table 5.7 Qualitative Grading Report



#	Attribute	Grade		Remarks
1.	Conformity			
	• To Standards	🟩	10	
	• To Requirements	🟩	10	
2.	Consistency	🟩	10	
3.	Supports Concurrency	🟩	10	
4.	Graphical User Interface	🟩	10	
5.	Usability			
	• Ease of use	🟩	10	
	• User Manual help	Not Applicable		
	• Ease of Learning	🟪	9	Requirements not documented well.
6.	Performance			
	• Query	🟥	7	Refer to ticket # 2673240
	• Data Capture	🟥	7	Refer to ticket # 2673241
7.	Security	🟥	7	Refer to ticket # 2673201
8.	Robustness	🟩	10	
9.	Error Handling	🟩	10	

CHAPTER – 6
CONCLUSION & FUTURE WORK

6.1 Conclusion

The ultimate goal of this web-application is to make the job of the managers easier with respect to performance reporting of the merchants' integration that they are responsible for and it achieved that. By automating the tasks of querying and reporting, the managers are given the ability to focus on analysing the performance data and taking effective measures with that data in order to improve the performance.

6.2 Future Work

This web-app runs queries and fetches results individually for each query. Still, the managers have to copy the results and create the report themselves manually. We can overcome this by creating a batch file that runs every month which will run all the queries that a manager wants, for a particular merchant, and auto-populate the final report and send the report to the manager in email. By automating the complete reporting task using an automated batch file that runs on the beginning of every month on its own, the managers need not have to go through waiting period to fetch millions of rows of results and also won't have to manually input all the resulting data into the report.

CHAPTER – 7
REFERENCES

1. Ariel Ortiz, “Web Development with Python and Django”, ACM Journal, vol.4, no.1, pp.175-187, 2012.

2. Daniel Walker and Ali Orooji, “Metrics for Web Programming Frameworks”, CiteSeerX, vol.5, no.3, pp.97-104, 2011.

3. David Day, Joo Tan, Kyle Wamsley, "Designing an interactive personal assistant web application system", Journal of Computing Sciences, vol.2, no.7, pp.77-85, 2015.

4. Jazayeri. M, "Some Trends in Web Application Development", Future of Software Engineering Journal, vol.10, no.11, pp.199-213, 2007.

5. Nishant Sinha, Rezwana Karim, Monika Gupta, "Simplifying Web Programming", IBM Research Journal, vol.7, no.3, pp.132-145, 2015.

6. Reuven M. Lerner, “At the forge: twitter bootstrap”, Linux Journal, vol.2012 no.218 pp.6-13, 2012.

7. Selfa, D.M., Carrillo, M., Del Rocio Boone, M., "A Database and Web Application Based on MVC Architecture," Journal of Electronics, Communications and Computers, vol.48, no.48, pp.27-41, 2006.

8. Shubhangi Pharande, Simantini Nalawade, Ajay Nalawade, "Data Mining: Approach Towards The Accuracy Using Teradata”, International Journal of Computer Applications Technology and Research, vol.21, no.4, pp.107-125, 2015.

9. Andrew Brown, Jeffrey S. Chase, “Trusted platform-as-a-service: a foundation for trustworthy cloud-hosted applications”, ACM Journal on Cloud Computing Security, vol.9, no.7, pp.178-194, 2011.

10. Askins, Ben, Gree, Alan, “A Rails/Django Comparison”, International Journal of Open Source Software and Processes, vol.3, no.2, pp.7-19, 2011.

11. J. Plekhanova, “Evaluating web development frameworks: Django, ruby on rails and cakephp”, International Journal of Open Source Software and Processes, vol.3, no.7, pp.140-159, 2010.

CHAPTER – 8

APPENDIX A

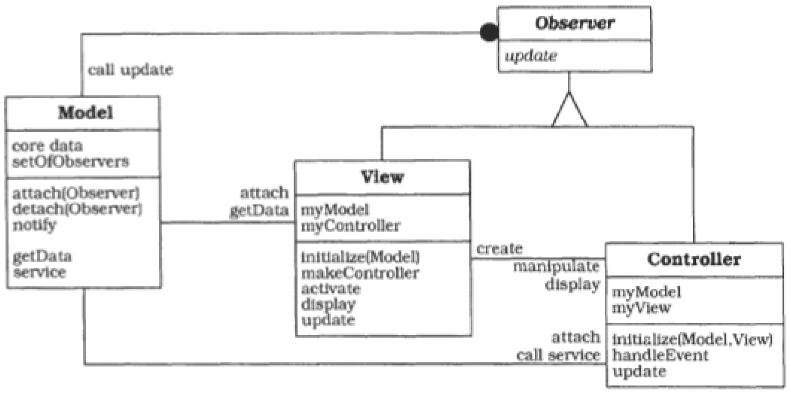


fig 8.1 – OMT Class Diagram for MVC Pattern

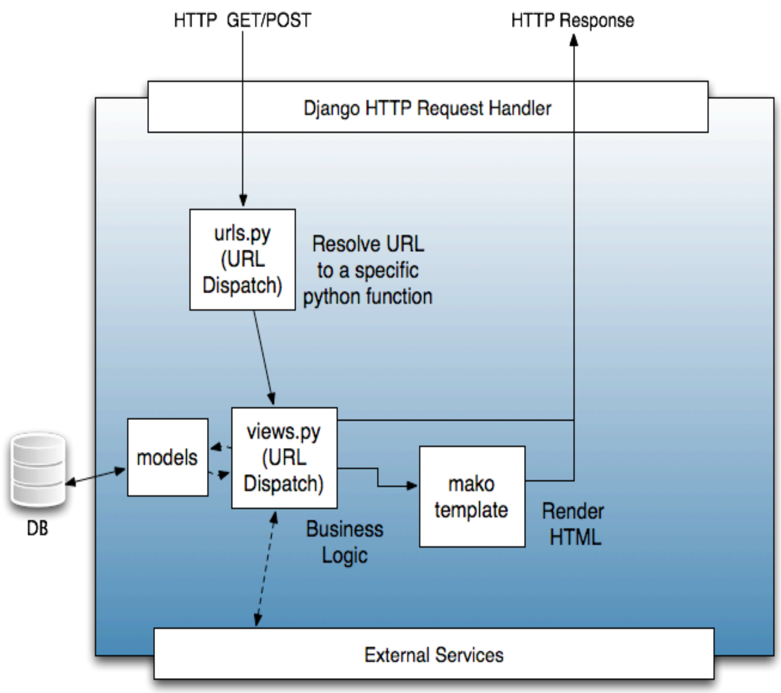


fig 8.2 – Django Architecture Diagram

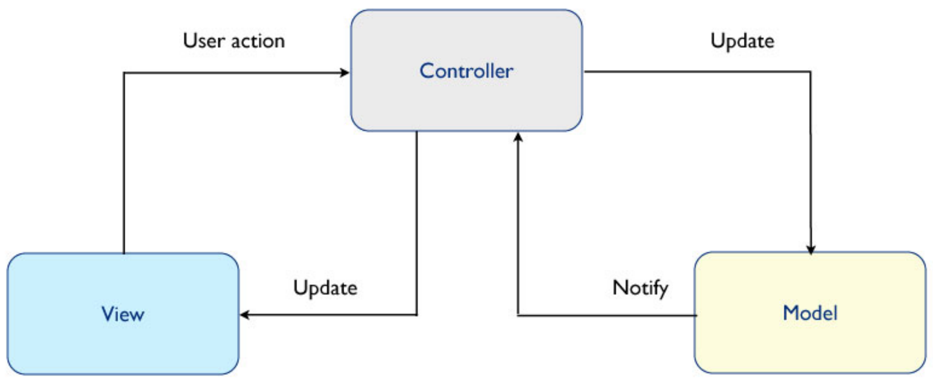


fig 8.3 MVC Based Django System Design

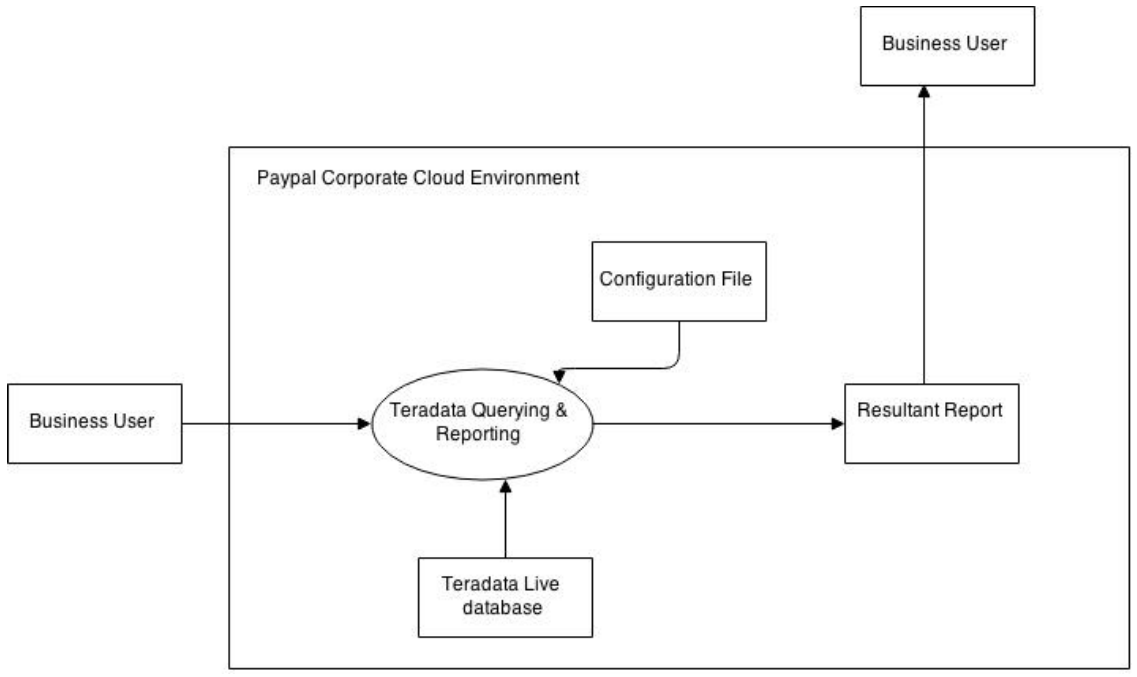


fig 8.4 High Level Design

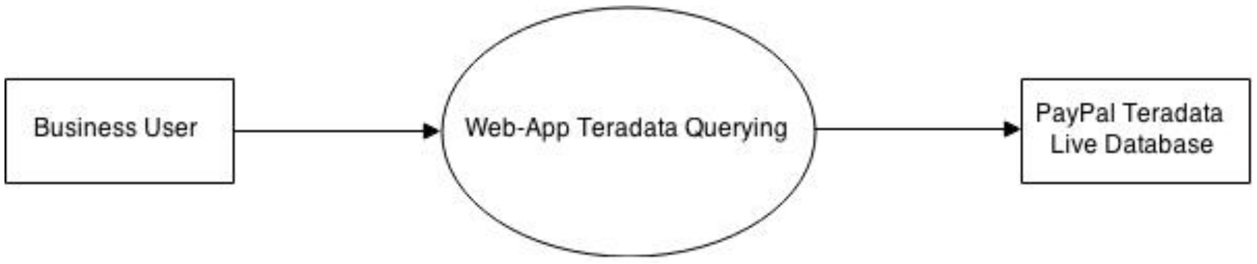


fig 8.5 Context Level DFD

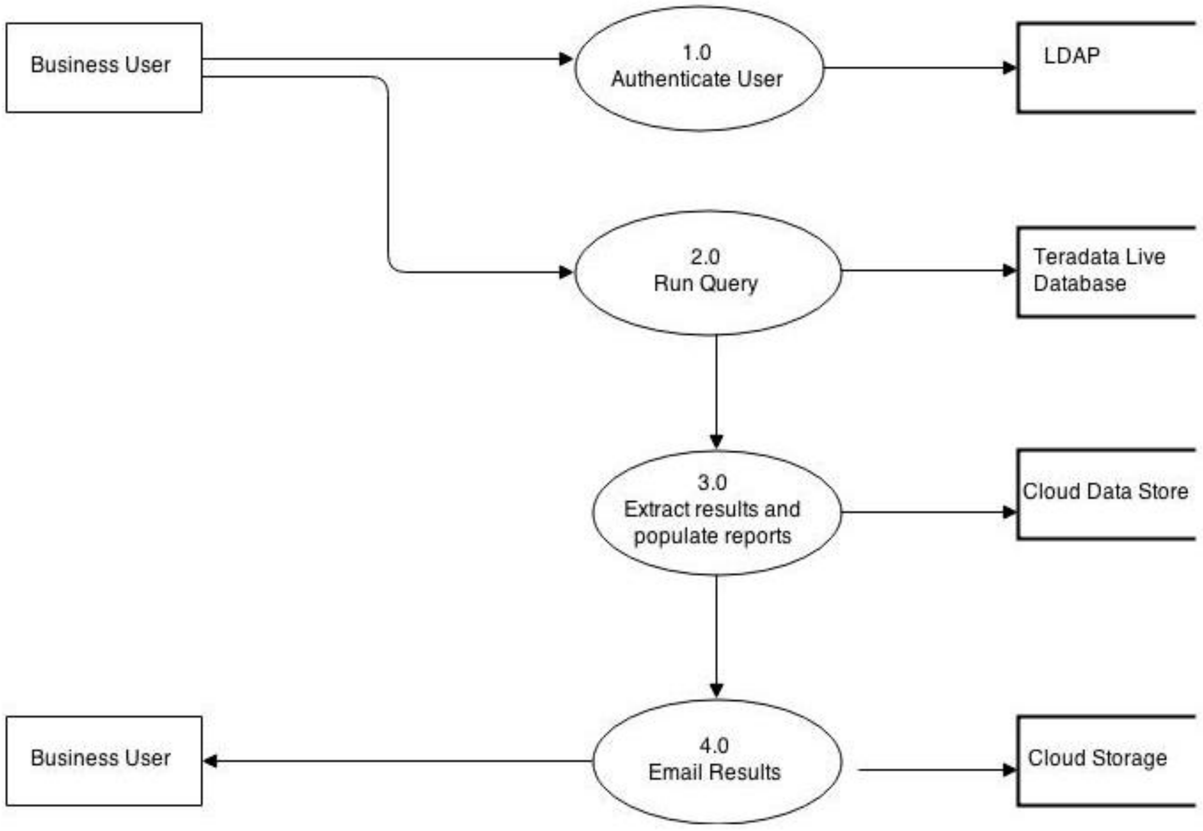


fig 8.6 Level 1 DFD

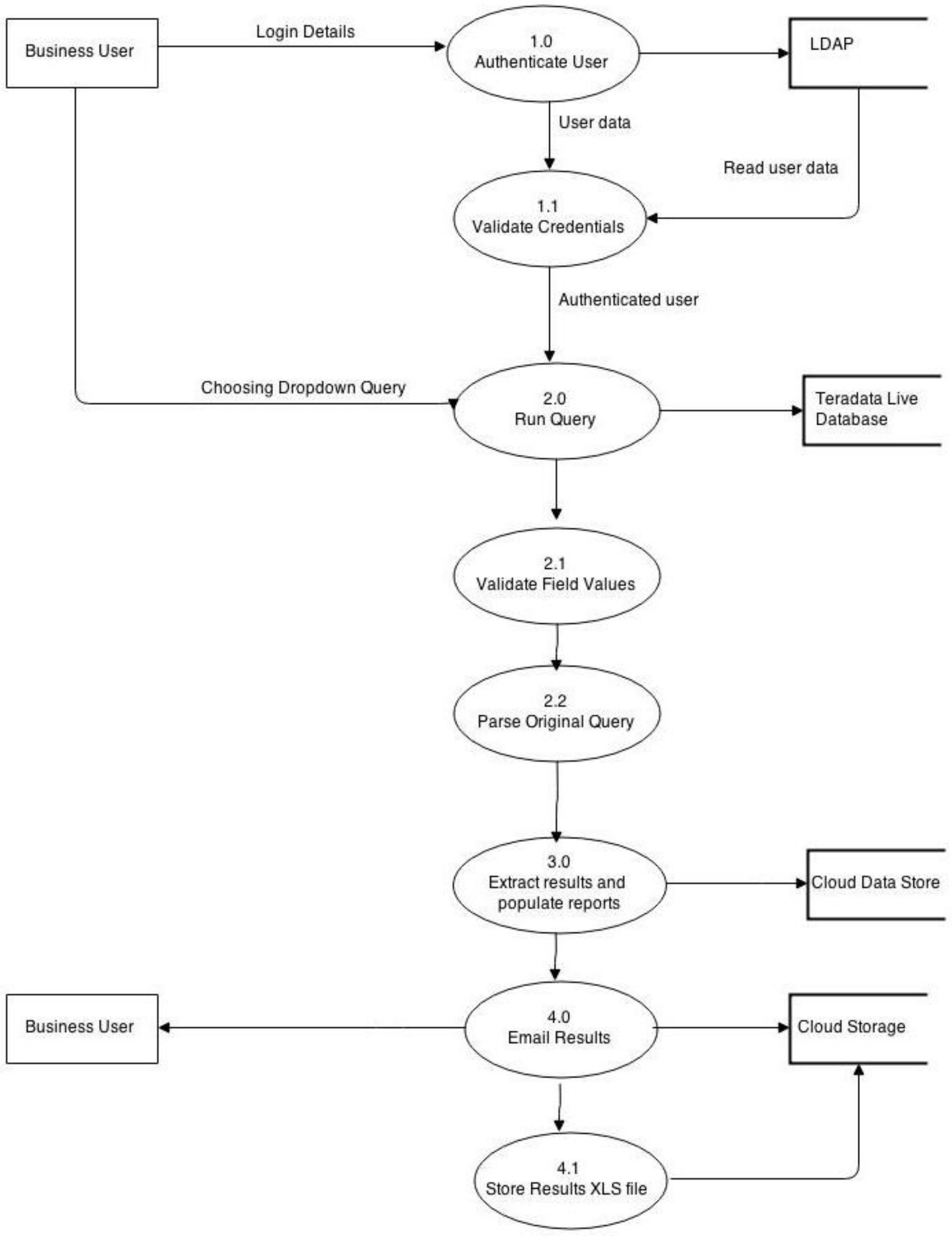


fig 8.7 Level 2 DFD

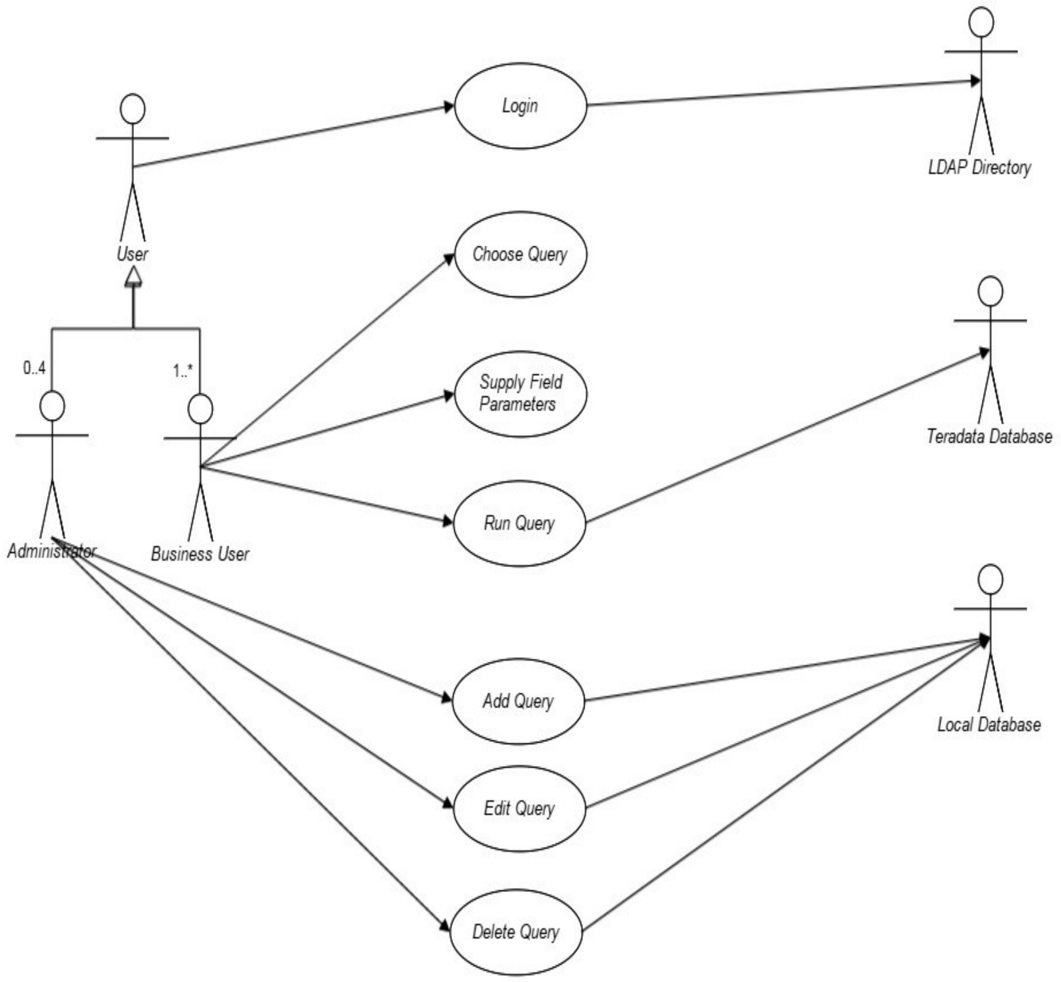


fig 8.8 Use-Case Diagram

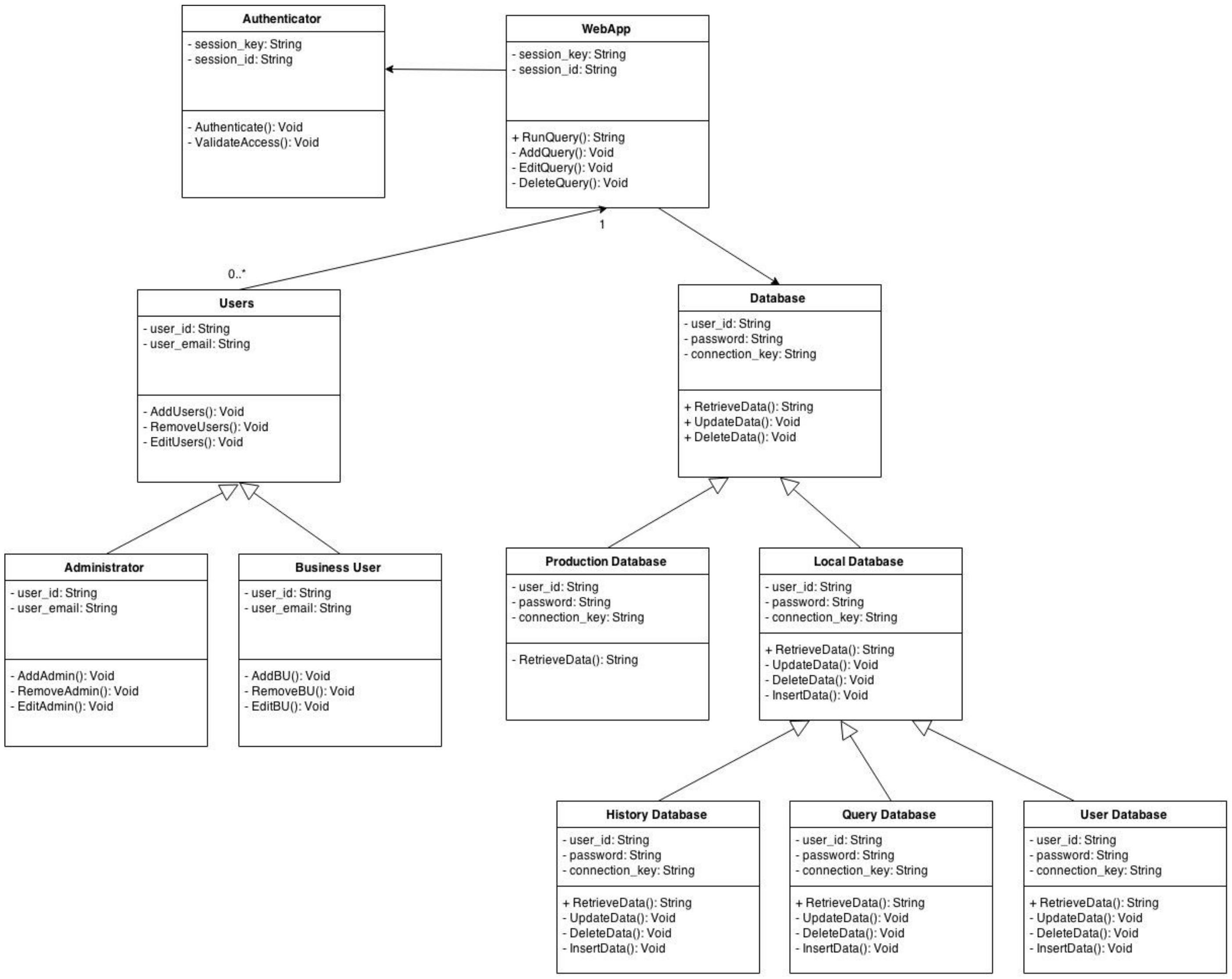


fig 8.9 Class Diagram

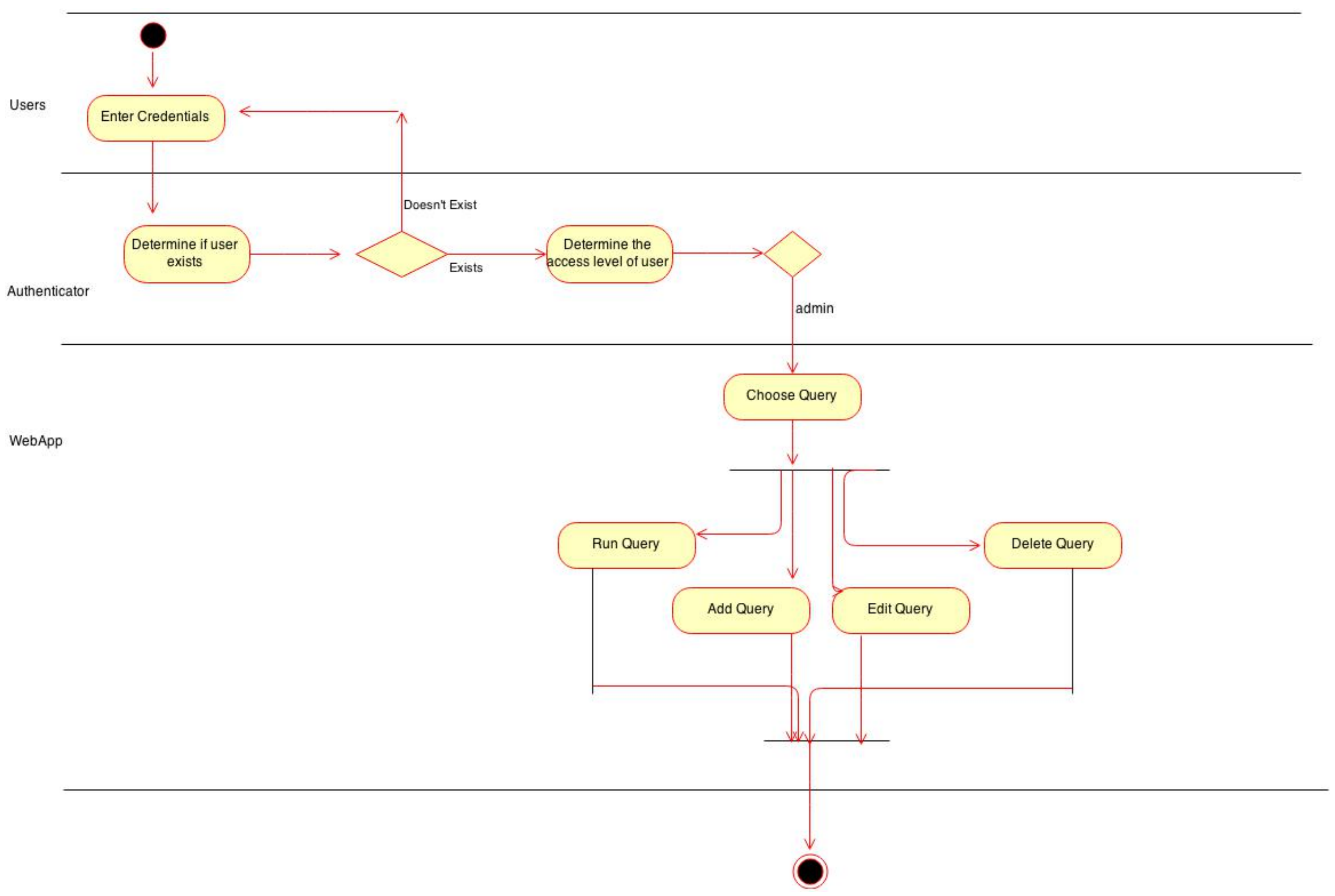


fig 8.10 Activity Diagram

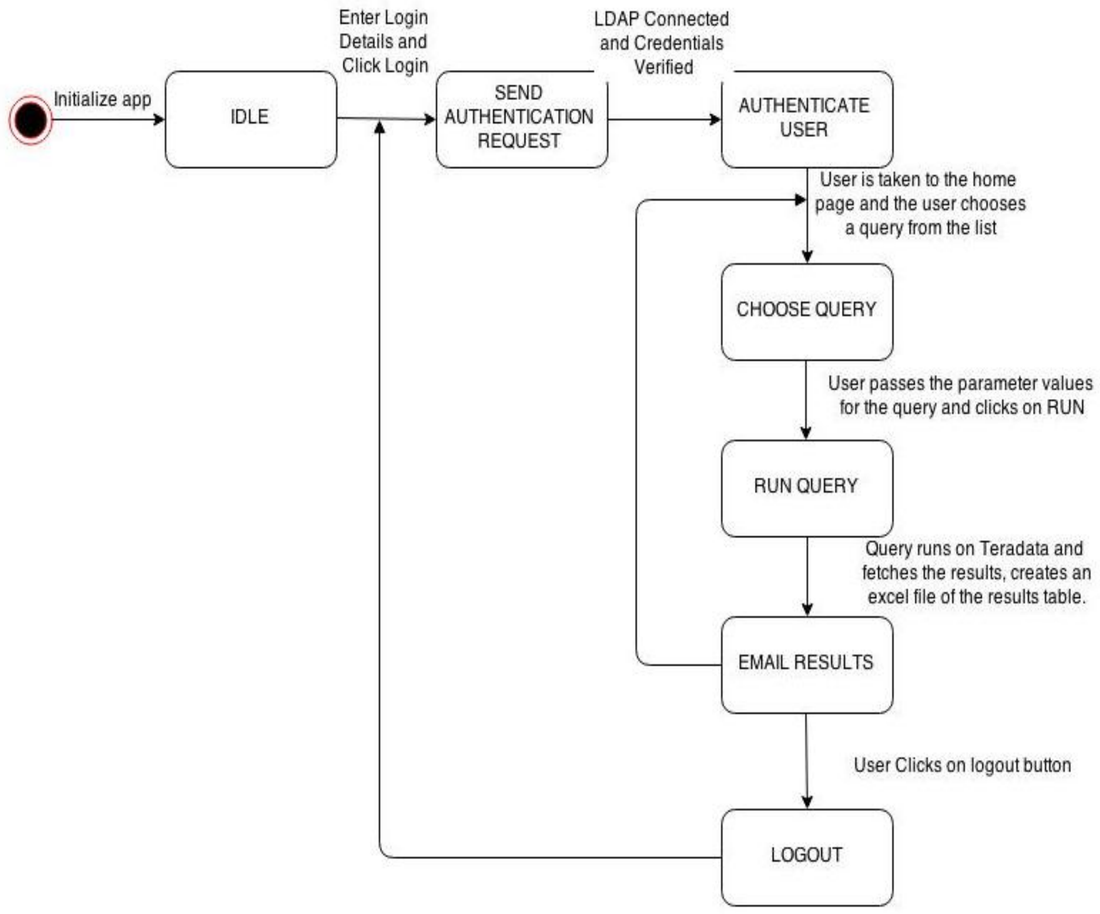

Initialize app
IDLE
Enter Login Details and Click Login
SEND AUTHENTICATION REQUEST
LDAP Connected and Credentials Verified
AUTHENTICATE USER
User is taken to the home page and the user chooses a query from the list
CHOOSE QUERY
User passes the parameter values for the query and clicks on RUN
RUN QUERY
Query runs on Teradata and fetches the results, creates an excel file of the results table.
EMAIL RESULTS
User Clicks on logout button
LOGOUT

fig 8.11 State chart Diagram

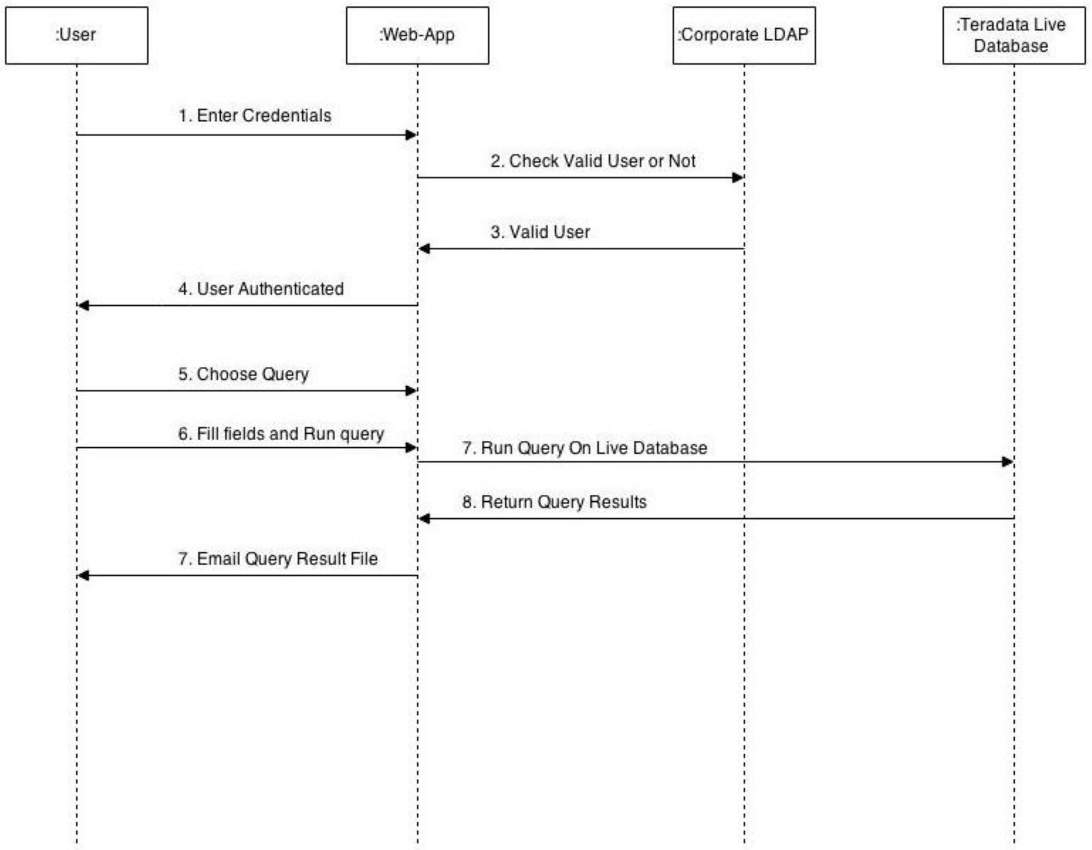


fig 8.12 Sequence Diagram

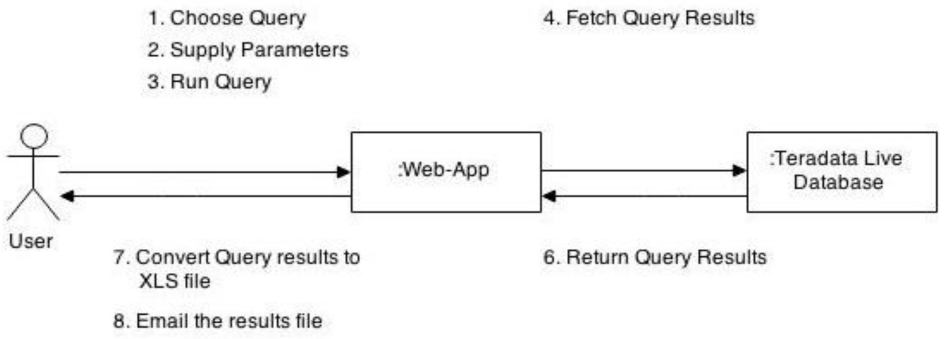


fig 8.13 Collaboration Diagram

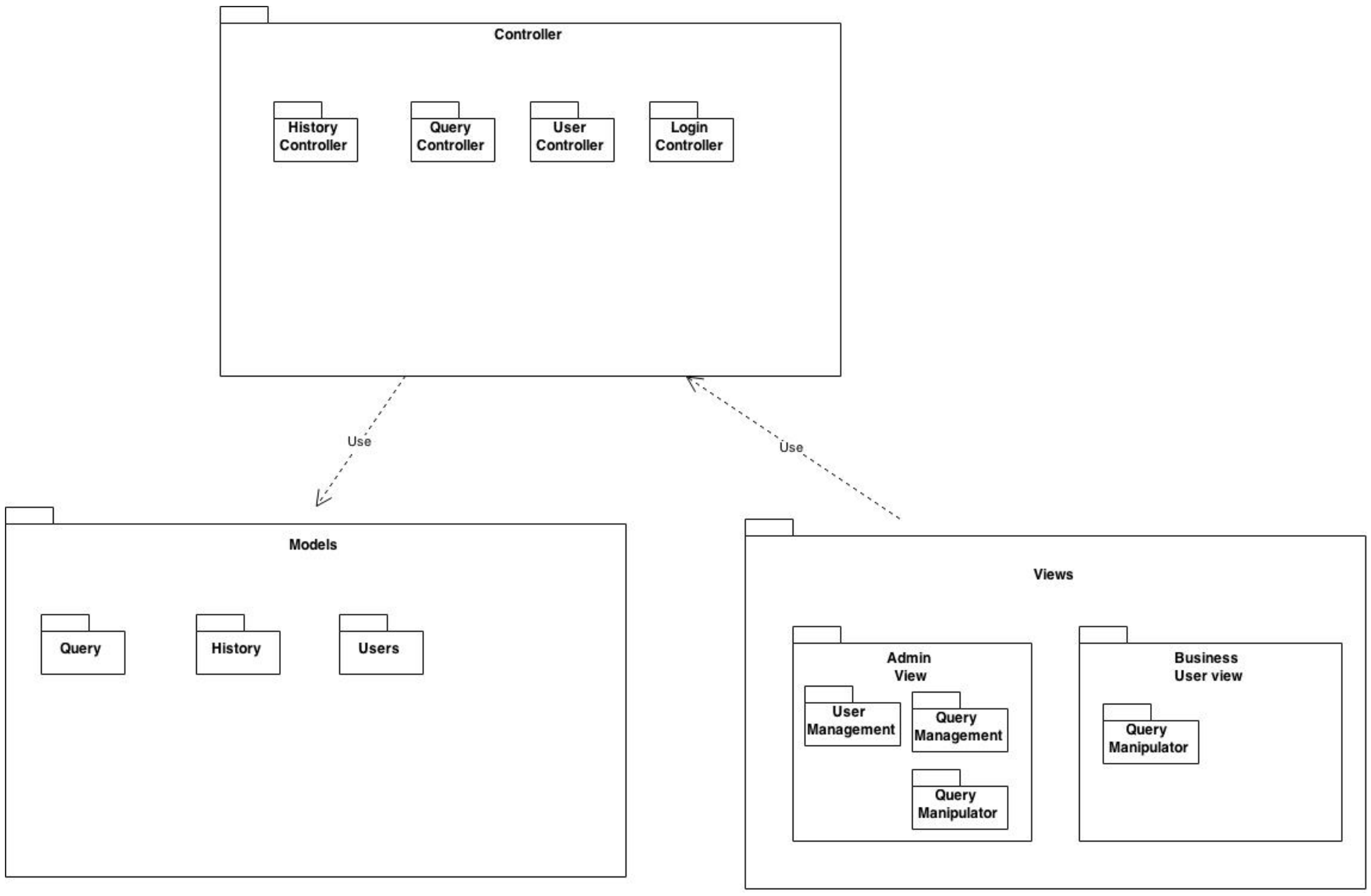


Fig 8.14 Package Diagram