SRS Document

**Project Name**: - DB Version Tracking System

**Project Manager**: - Nikhil Saxena

**Author**: -

* Ashish Chavan
* Trupti Bhandari
* Suyash Wagh

**Reviewer**: - Kishor Sen

**Organisation**: - SE2

**Date Created:** - 26 August 2020

Table of Contents

[1. Introduction 3](#_Toc49414727)

[1.1 Purpose 3](#_Toc49414728)

[1.2 Intended Audience and Reading Suggestions 3](#_Toc49414729)

[1.3 Product Scope 3](#_Toc49414730)

[2. Overall Description 3](#_Toc49414731)

[2.1 Product Perspective 3](#_Toc49414732)

[2.2 Operating Environment 3](#_Toc49414733)

[2.3 Design and Implementation Constraints 3](#_Toc49414734)

[2.4 User Documentation 4](#_Toc49414735)

[2.5 Assumptions and Dependencies 4](#_Toc49414736)

[3. External Interface Requirements 4](#_Toc49414737)

[3.1 User Interfaces 4](#_Toc49414738)

[Request Page 4](#_Toc49414739)

[3.2 Software Interfaces 5](#_Toc49414740)

[3.3 Communications Interfaces 5](#_Toc49414741)

[3.4 Hardware Interfaces 5](#_Toc49414742)

[4. Workflow 6](#_Toc49414743)

[5. Non-Functional Requirements 9](#_Toc49414744)

[1. Performance: 9](#_Toc49414745)

[6. Future Scope 9](#_Toc49414746)

# Introduction

## Purpose

Database Version Tracker project is to track each query executed by user during release. This project will help track database changes and to trace error during any defects or errors. This project is basically git of database queries.

The User will register query which Approver will check. Approver will Approve or Reject query.

## Intended Audience and Reading Suggestions

This document is intended for developers. Scope of application is restricted to SE2 workplace.

## Product Scope

This software will help to maintain logs of change request which will help to track database version history.

# Overall Description

## Product Perspective

There is nightly database execution. In which .sql file is executed. There is no application to maintain and track query defects and enhancements. The queries are approved and stored in .sql file which will be executed by a shell script.

* **Request Page:**

User registers request i.e. Query here. After registering request is stored in request.csv file.

* **View Page:**

User can view Tickets.

* **Approve/Reject Page:**

Approver enters email id and then Approver or reject request.

* **Shell Script:**

Queries from .SQL file should be executed in database and logs should be generated.

## Operating Environment

Software will operate on

* spring boot
* Angular.js 1.5.
* Database used will be Oracle 12C.
* Shell environment is UNIX.

## Design and Implementation Constraints

Application will not have database access. So queries are to be stored on disk and accessible from outside project folder.

Python and Node.js are not available on environment so script should be in Shell Script.

## User Documentation

Documentation is expected at end of project.

## Assumptions and Dependencies

Mail credentials to send mail.

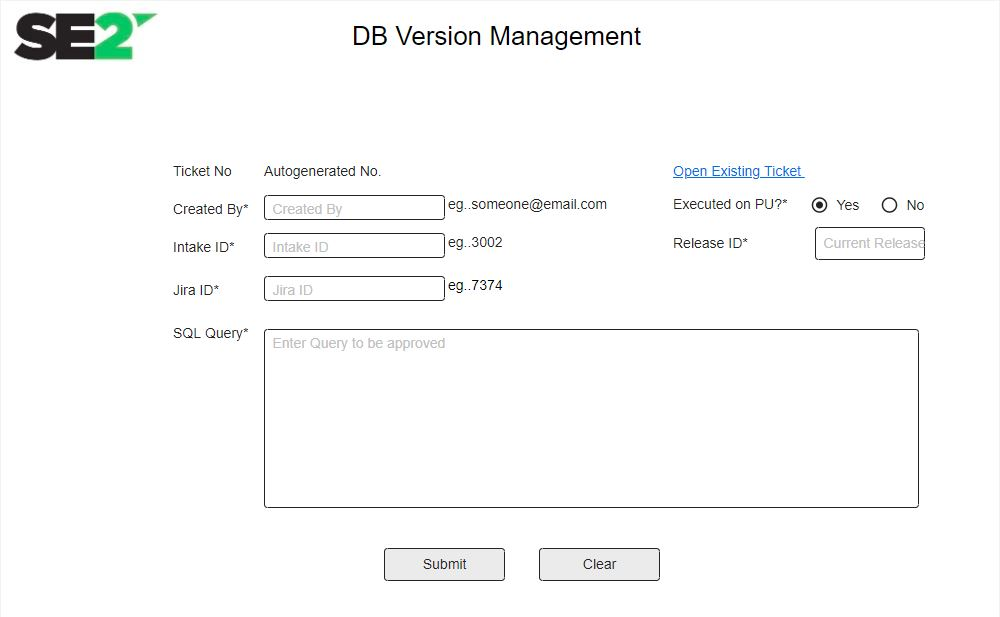
# External Interface Requirements

## User Interfaces

# Request Page

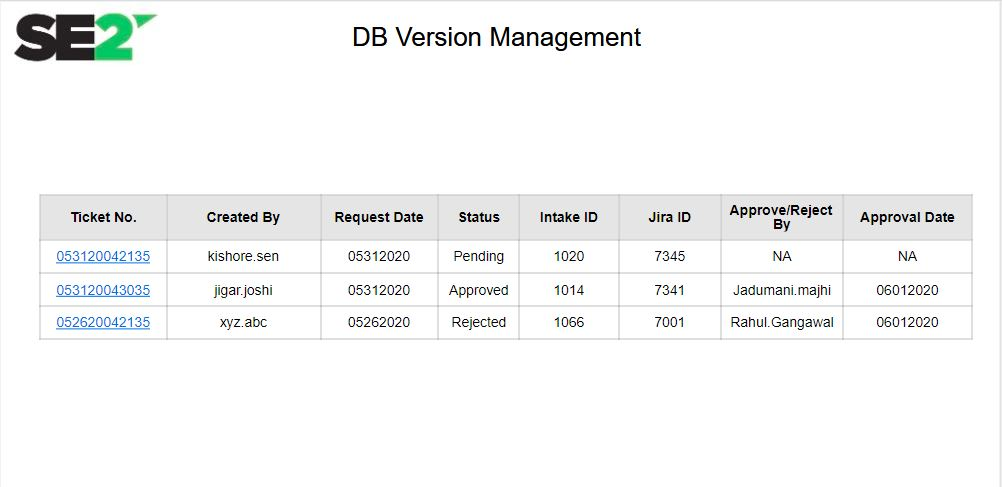
This page to register query in application.

* User will enter all required fields
* Validation will be done and then user can submit
* Clear Button will clear all fields



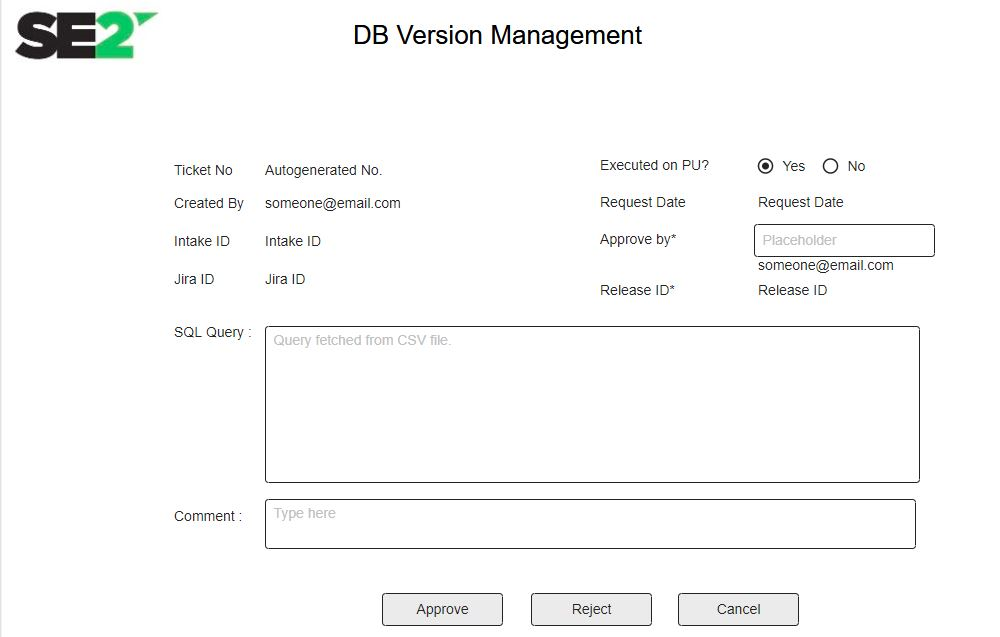
**View Page**

All queries are visible. Approver can visit ticket id by clicking on ticket no hyperlink.



**Request Approval Page**

This page is intended for Approver. He will enter his approver mail id and approver or reject query respectively.



## Software Interfaces

* Angular.js 1.5 – Front End
* Spring boot, Java 8 - Backend
* Oracle 12C - Database

## Communications Interfaces

Spring Boot SMTP server is used to send email

.

## Hardware Interfaces

* 1GB Ram
* 40 GB HDD

# Workflow

There are 3 main pages in application. Request Page, View Page and Approver page. Main landing page is Request Page.

1. **Request Page**

On this page user Registers request.

Form fields:

|  |  |
| --- | --- |
| Field Name | Description |
| Ticket No | Auto Generated Ticket No.  Logic:  Auto ticket no generation will be based on  Today’s Date (MMDDYY)+TSTAMP(HHMMSSMS) For Ex : #052920042135100.  Field is read only, user not able to edit. |
| Created By\* | Text box.  Validation  User entry current format of email id.  Field is mandatory. |
| Intak Id\* | Text box.  Validation  Field is mandatory. |
| Jira Id\* | Text box.  Validation  Field is mandatory. |
| Executed on PU\* ? | Radio button. (Yes/No)  Validation  Field is mandatory. |
| Release ID\* | Text box.  Validation  Field is mandatory. |
| SQL Query\* | Text box.  Validation  Field is mandatory. |
| Open existing ticket | Hyperlink  On click user view all ticket, redirect to page 2 (View) |
| Submit | Button |
| Clear | Button |

**Control Flow :**

**Front End:**

1. User enters fields on form.
   1. User Clicks Submit Button
      1. Validate all fields are correctly entered.
      2. After successfully validation save all data in .cvs file.
      3. Send notification to approval list. (approval list maintains in property file)
      4. Set message: “Request save successfully”
   2. User Clicks Clear Button

2.2.1 Clear all fields.

**Back End:**

* Fetch approvers from properties file and notify them with mail.
* Store request in .csv format

1. **View Page**

Displaying Request queries in table format.

|  |
| --- |
| * Ticket No – Hyperlink: Takes to Approve Page |
| * Created By\* |
| * Intak Id\* |
| * Jira Id\* |
| * Release ID\* |

1. **Approve Page**

|  |  |
| --- | --- |
| Field Name | Description |
| Ticket No | Read Only |
| Created By | Read Only |
| Intak Id | Read Only |
| Jira Id | Read Only |
| Executed on PU ? | Read Only |
| Release ID | Read Only |
| SQL Query\* | Populate save data.  Editable field |
| Approved by\* | Text box.  Validation  User entry current format of email id.  Field is mandatory. |
| Comment\* | Text box.  Validation  Field is mandatory. |
| Approve | Button |
| Reject | Button |
| Cancel | Button |

**Control Flow:**

**Front End :**

1. Approver enters email on form (Proceed if email belongs to approver).

* 1. Approver Clicks Approve Button

1. Validate all field entered by user.
2. After successfully validation save all data in .csv file & .sql file. (Below format)
3. Update status “Approved”
4. Send user to view page.
   1. Reject Button Clicked
      1. Update status with ‘Reject’
      2. Send user to view page.
   2. Cancel Button Clicked
      1. Send user to view page.

**Back End:**

* Send email to if approved or rejected respectively.
* If Approved query in .sql file and update in *request.csv* file.
* If Rejected update in *request.csv* file.

**Script**

Get Queries from “.sql” file and executed one by one on “T\_DB\_VERSION”.

T\_DB\_VERSION format

* Ticket number
* Intake Id
* Created By
* Approved by
* System time

Validation

* One Ticket number from .sql file should be executed exactly once.
* .log Filename format should be “DBSCript\_TIMESTAMP\_(Log\_file\_count).log”.
* Error should be logged in log file
* New Log file should be created if log size exceeds 1MB.

# Non-Functional Requirements

# Performance:

Mail should be sent immediately after creating request and approval and rejection.

# Speed, Request:

API request should be fast and responsive

# Future Scope

* Authentication System to users. Approve Page will be accessible to only authorized user.