TABLE OF CONTENTS

1. [Introduction](#INTRO) 2

1.1 Purpose 2

1.2 Scope 2

1.3 Overview 3

2. System Overview 4

2.1 System Architecture 5

2.2 Configuration Files and 5

2.3 User Interface and APIs Over View 10

3. System Design 19

3.1 Design Method and Standards 19

3.2 Naming conventions 19

3.3 Programming Standards 19

3.4 Software used 20

3.5 System Requirement 20

4. Document COntrol 21

1. **INTRODCUTION**
   1. **Purpose**

This document is a generic Technical Design Document used by USFOODS project. It provides guidance which is intended to assist the relevant technical supports to client and developer, in producing a project‑specific technical assistant .

This introductory section set out an approach to designing systems that may be developed under USFOODS ECOM Project.

It attempts to set standards and create a consistent approach to the design and development of systems across the spring boot layered architecture.

Contemporary approaches to design (Object Oriented Design), aims for code re-use and the need to develop systems that will work on an operational basis over many years and the associated desire to make such systems easily supportable and affordable.

This software is designed for Automation purpose only. many things we are depend on product but product having their own limitation, so keep this thing in mind. we have developed “**STOS**” application in java for automate our process.

USFOOD ECOM team working on frequent fixes and enhancements which necessitate a full regression and functional testing with the help of QA team. It will require an enormous amount of manual effort, and it is falling under Expenses Category for the clients. Due to this, the ratio between expenditure vs Capital cost is increasing. HCL has provided a solution to overcome this challenge to reduce the testing expenditure cost for the client.

This solution has implement at project level as a value add to the clients. Clients will be getting more beneficial toward the yearly test execution on ECOM fixes.

1. STOS tool can run/deploy any machine without any problem with help of Spring boot application. It saved lot of time and increased productivity of the testing team.
2. Shareable object repository and reusable code modules for efficient test creation and reduced maintenance.
3. Customizable test report with video capturing of test execution for bugs tracking.
4. Run distributing and running tests on several machines and manage multiple environments from a central point with this tool
5. Integrates with tools like Jira, Jenkins, Bamboo, Azure and AWS.
6. Consume Rest client API service

.

* 1. **Scope**

**STOS** application will optimize the ECOM Manually automation work load and automate some manual work in Batch/Group Optimization Jobs.

As of now we have Automation category:

1] Manual automation Run

2] Group wise automation Run with Job

3] Job Wise Health check automation

The benefits behind this Job automation is that we will safe from erroneous business practice, avoid subordinate co-worker and timing consumption.

* 1. **Overview**

“**STOS” ( Selenium Test Automation Suite )** concept come in picture when **ECOM** automation POM based tool become error prone, there was no use of best practices and no more scalable/reusable.

"**STOS**” is Web based Graphical User Interface Tool developed in Java & Spring Boot that helps to execute the automation test scripts based on the user need.

It has features such as test suite/batch/individual execution, test execution reports generation and report viewing which can be seen in one-click.

Our team is part of e-commerce automation testing, Legacy Automation Tools is to be more complex and bound to run test script in Eclipse Ide. When end-user/business user need to run the regression script they need java and eclipse ide setup, which was cumbersome and time taking. Then we started with this tool for end-user so he would use it without any ability on technical stuff.

This tool provides simplicity, efficiency as per organizations need. it can deliver a GUI experience that's optimized for a specific script execution out of box of STOS environment so that end-user can run the test script on to the user interface without knowing any ability on automation

1. **SYSTEM OVERVIEW**

**2.1 System Architecture**

This application has been developed in Java Web based client server architecture. The Architecture

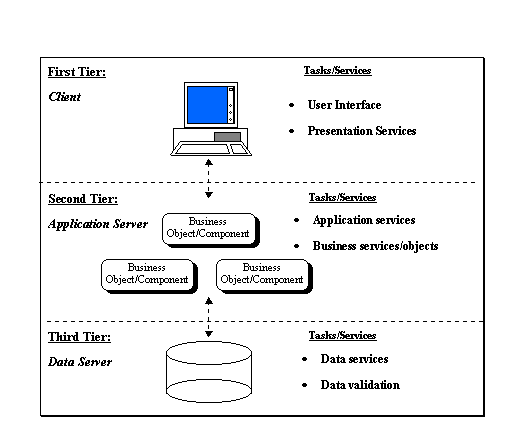
may be a simple client-server system in which web technologies are used to provide forms from a

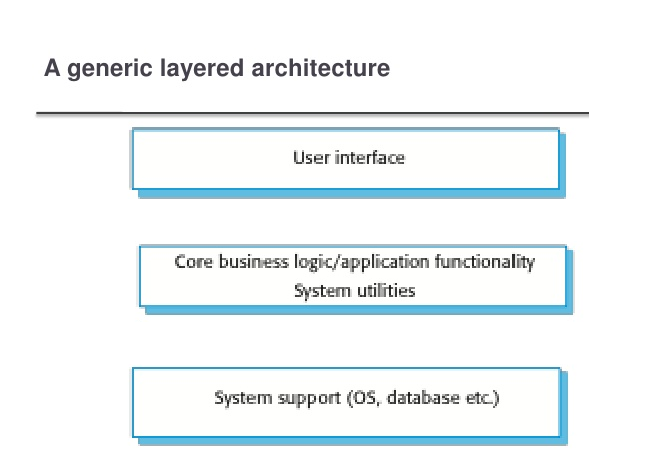
simple server that can be filled in remotely by someone.

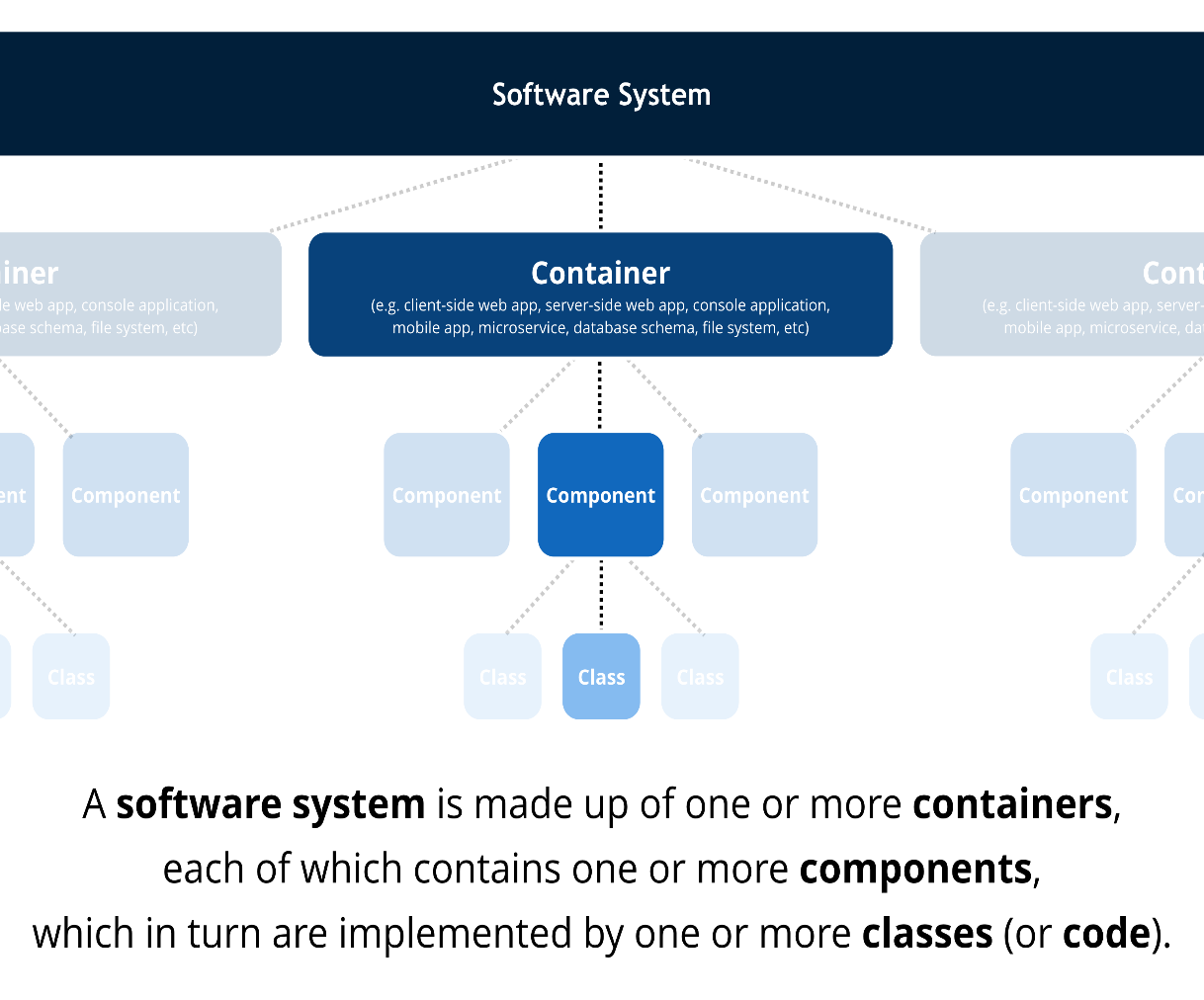
**Today n-tier architectures can often be divided into several tiers as follows:**

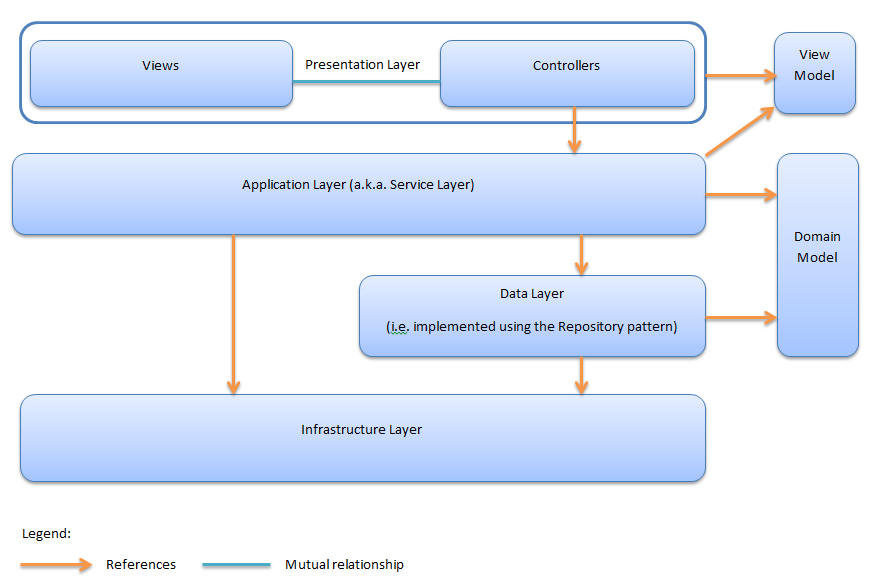
1. **client or ‘access’ tier** – which facilitates access by both external and internal clients through technologies such as web browsers
2. **presentation tier** – the layer that accesses information from the other systems as required, and can contain its own ‘business rules’ for simple processes
3. **business tier** – contains the business information layer
4. **persistence tier** – the database layer, which provides the facilities to store data.

Depending upon the solution proposed for the system in question there may be a balance of functions placed into each of these layers. We believe that in the design process it is vital, however, to have a layered architecture as the basis of the solution to facilitate change and adaptation of the system in the future – in other words to protect the investment that has been made in the system as technologies develop in the future. The degree of isolation afforded to elements of the system through adopting a layered approach is significant and can help immunise the project from having to be replaced in full in years to come – some key elements will survive technological developments.

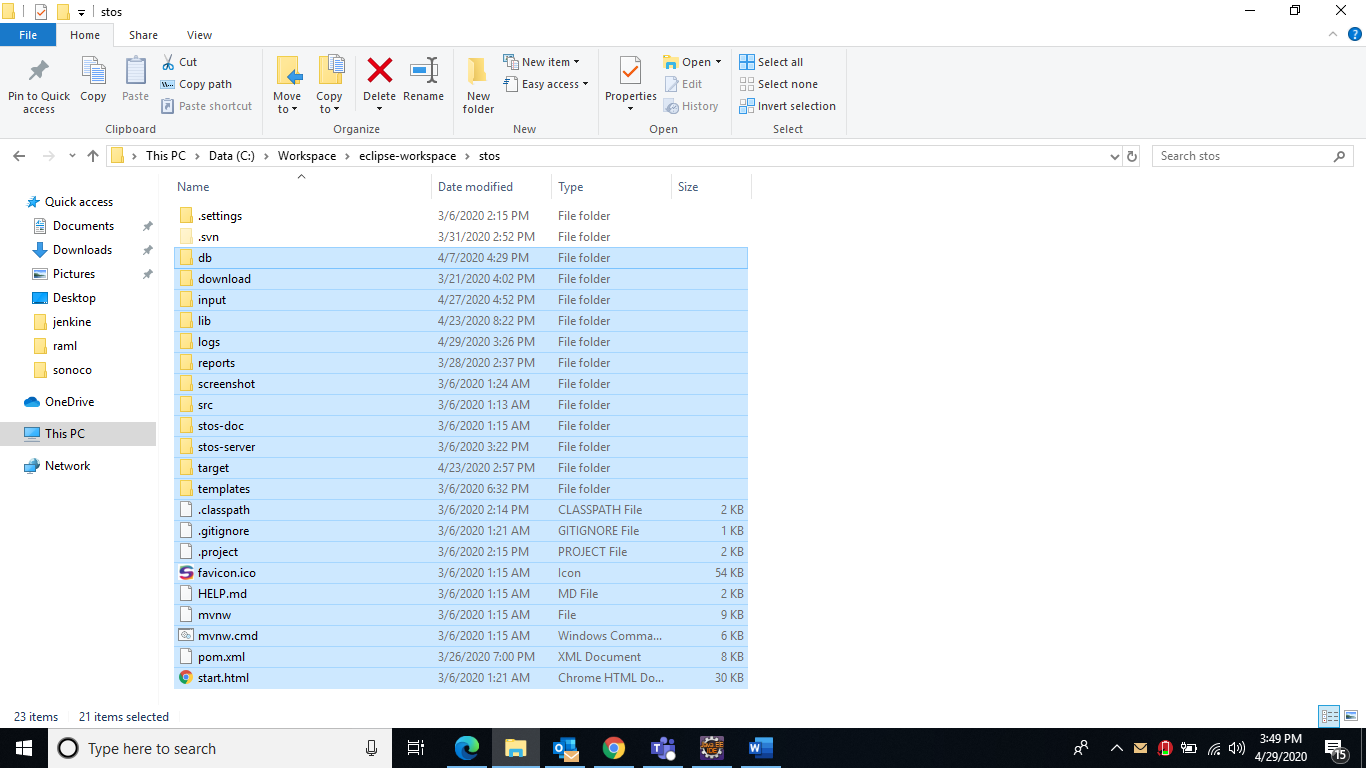




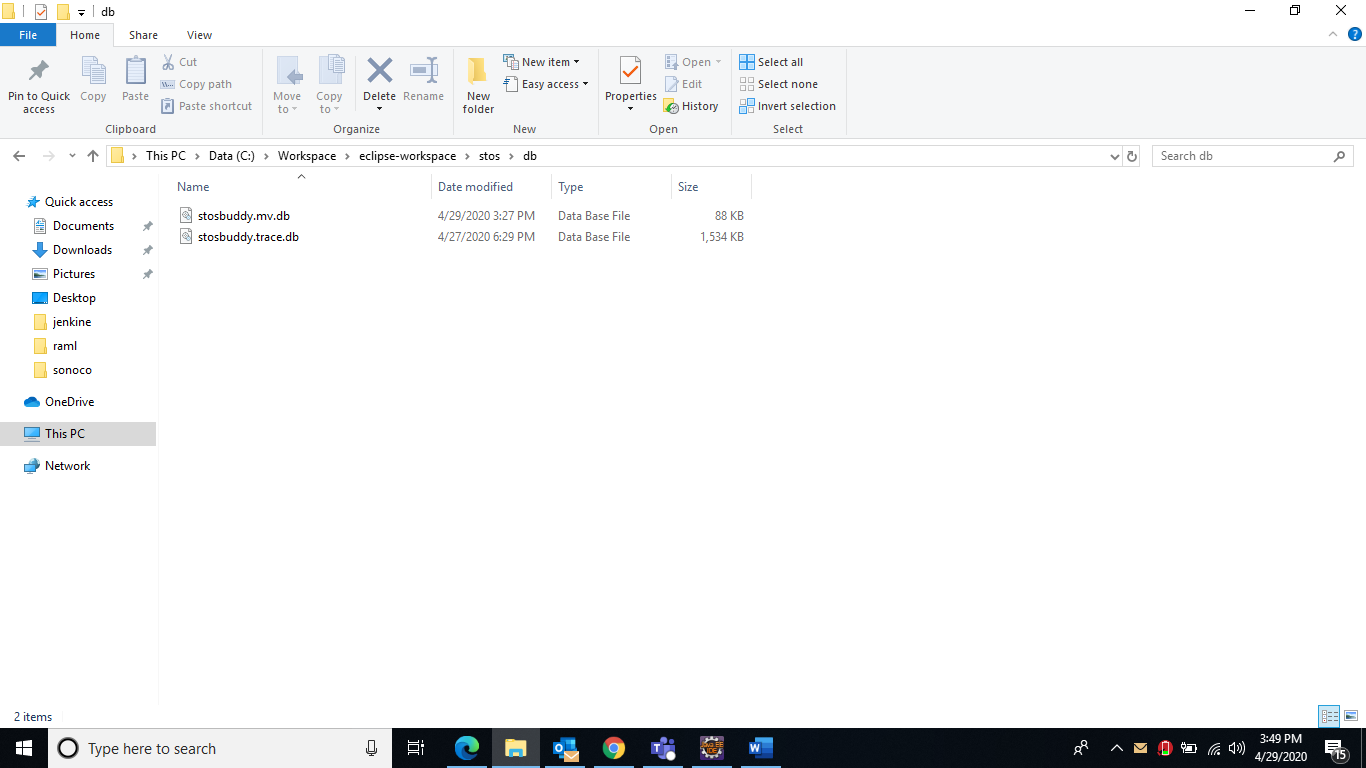




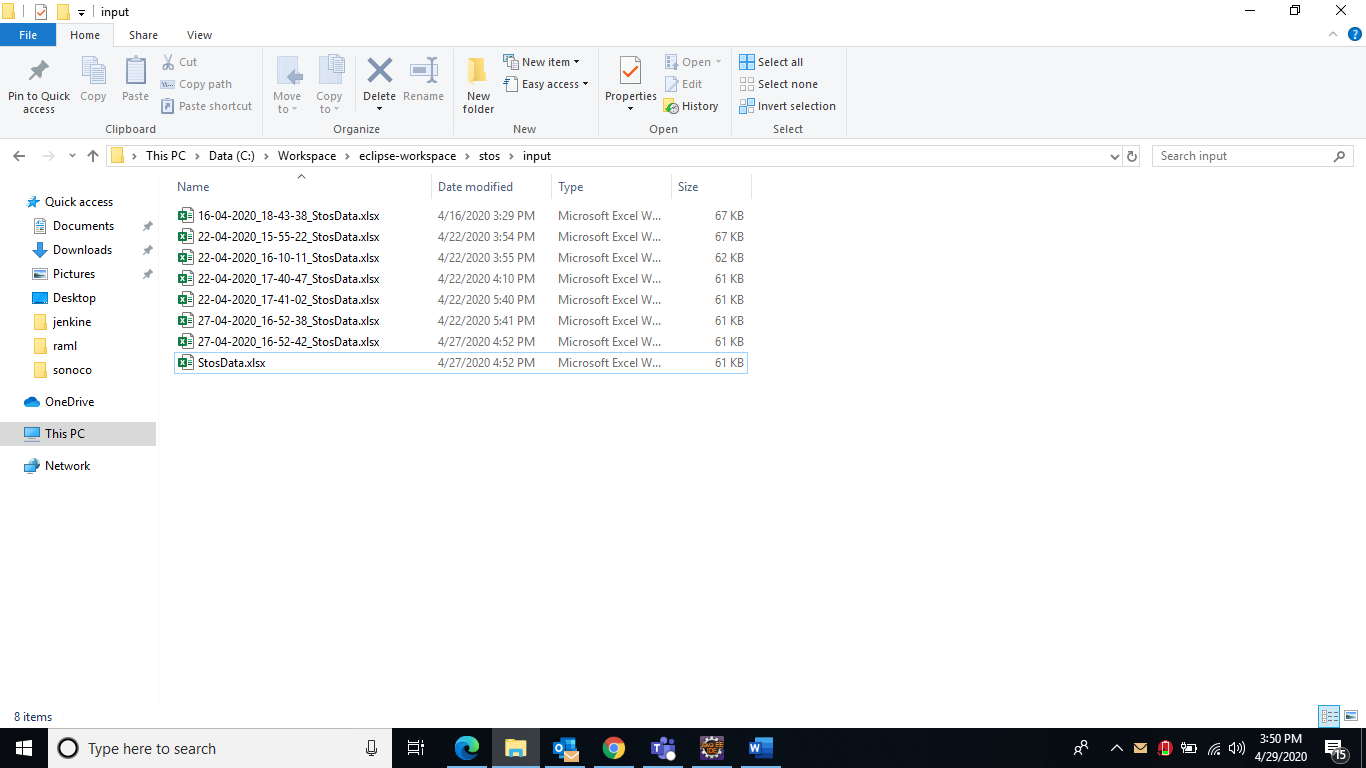
* 1. **Configuration File**



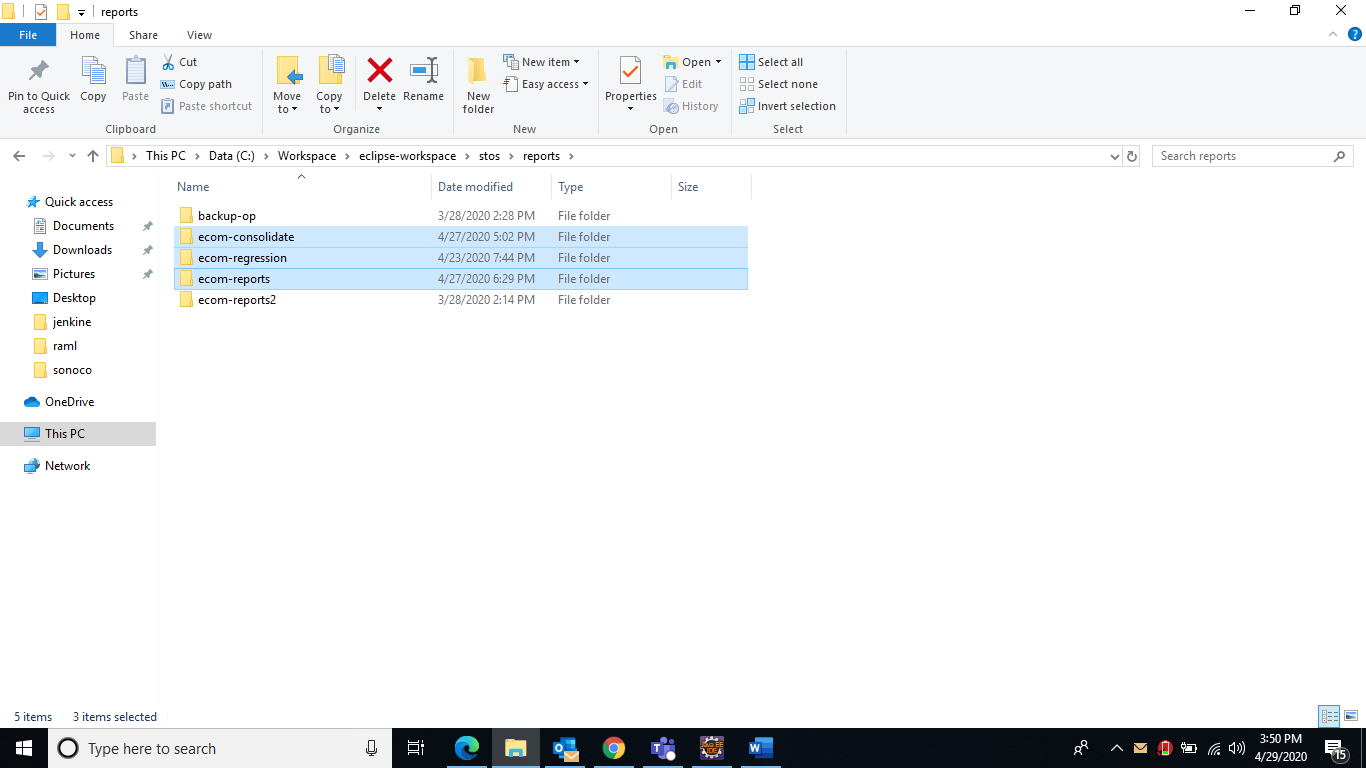
**DB-**

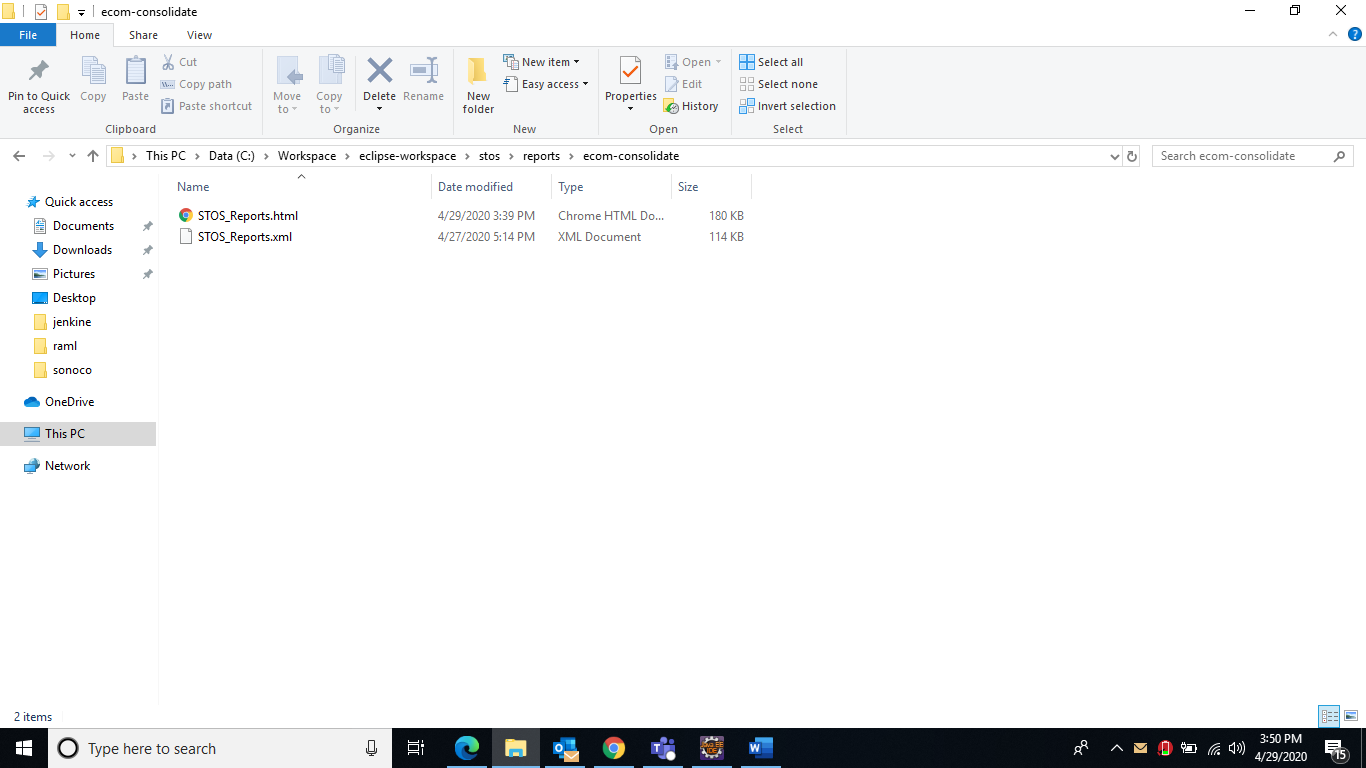


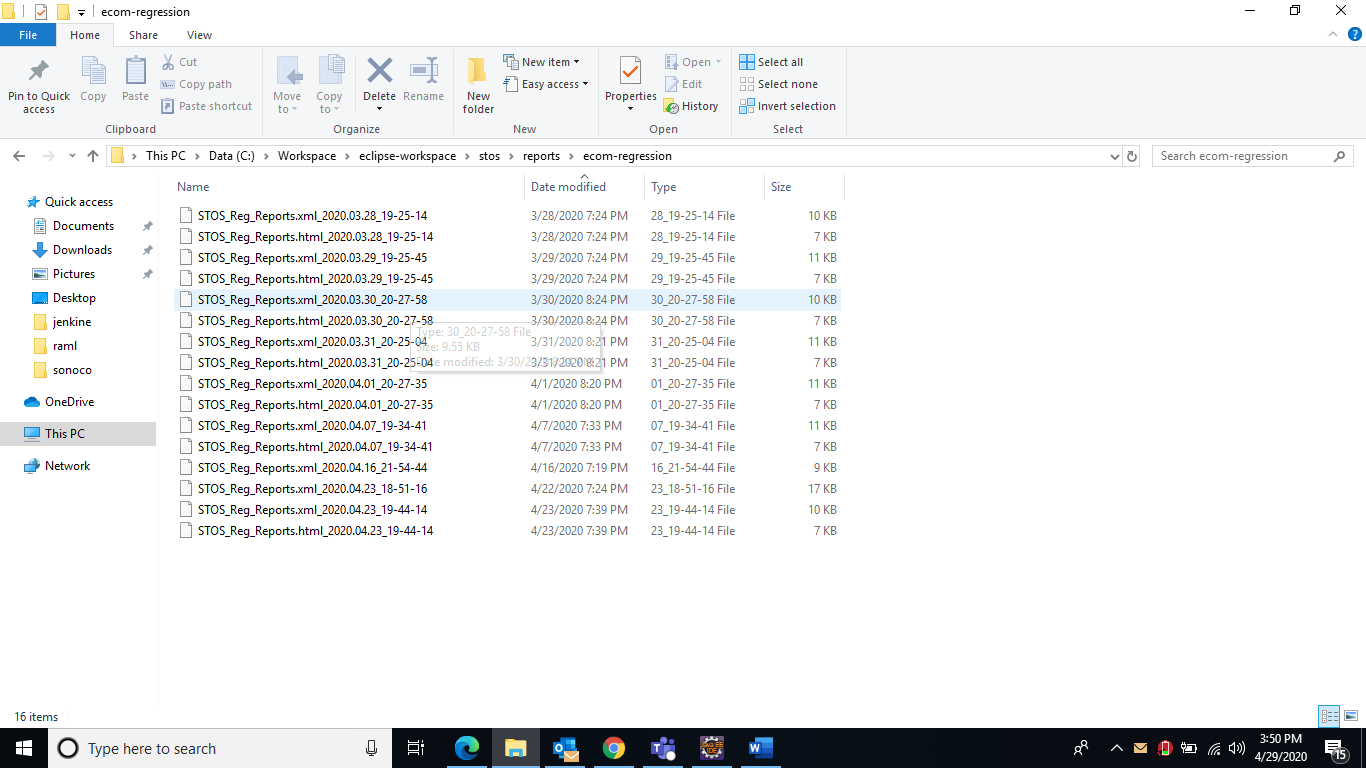
**Excel-**

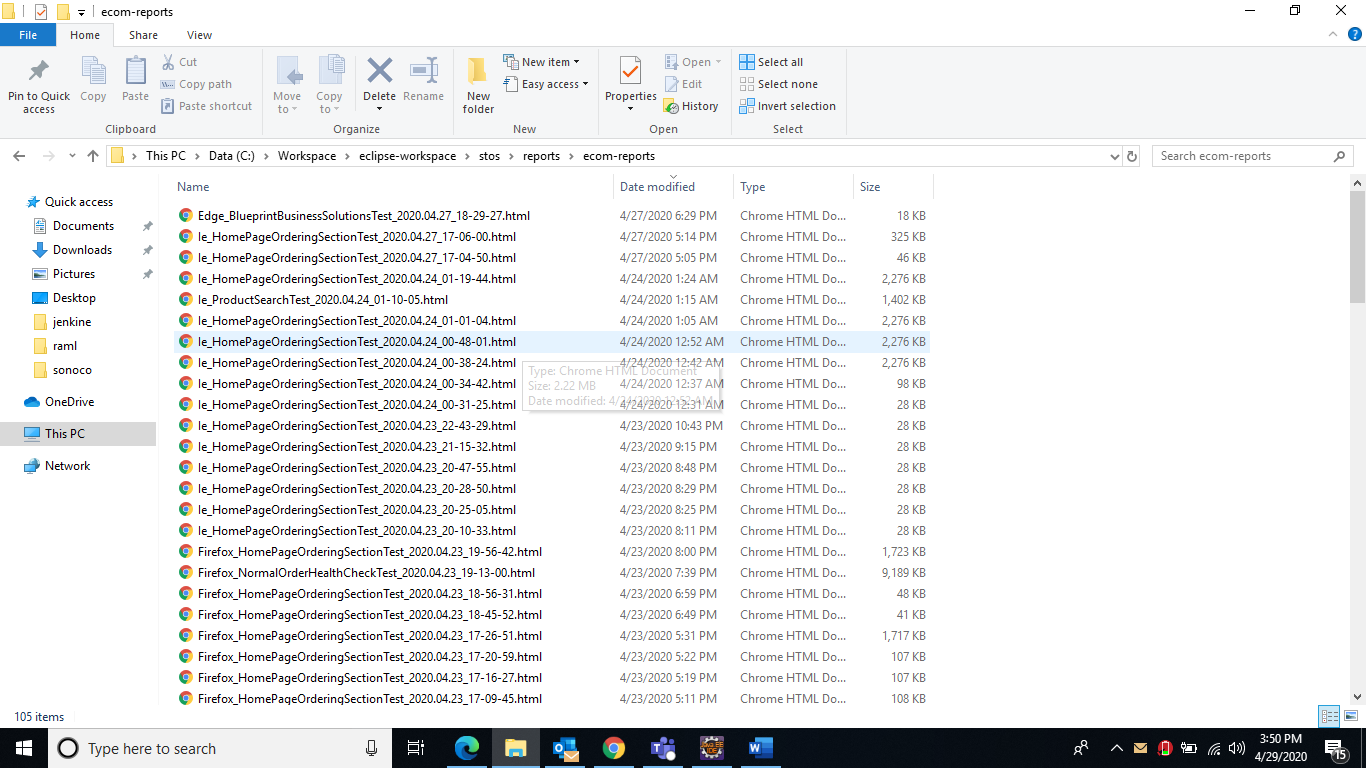


**Reports**

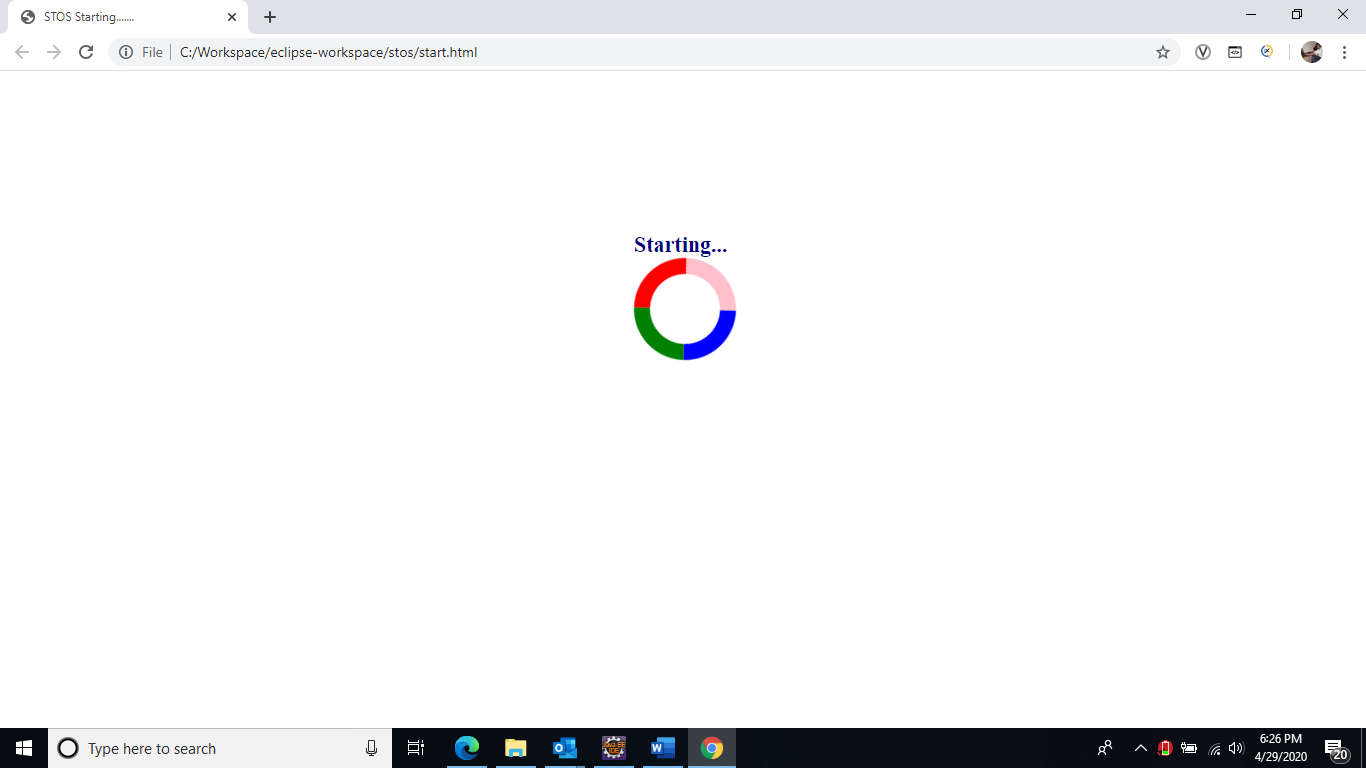




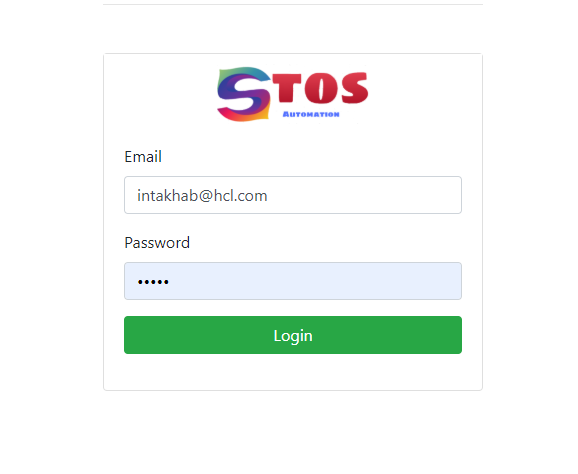




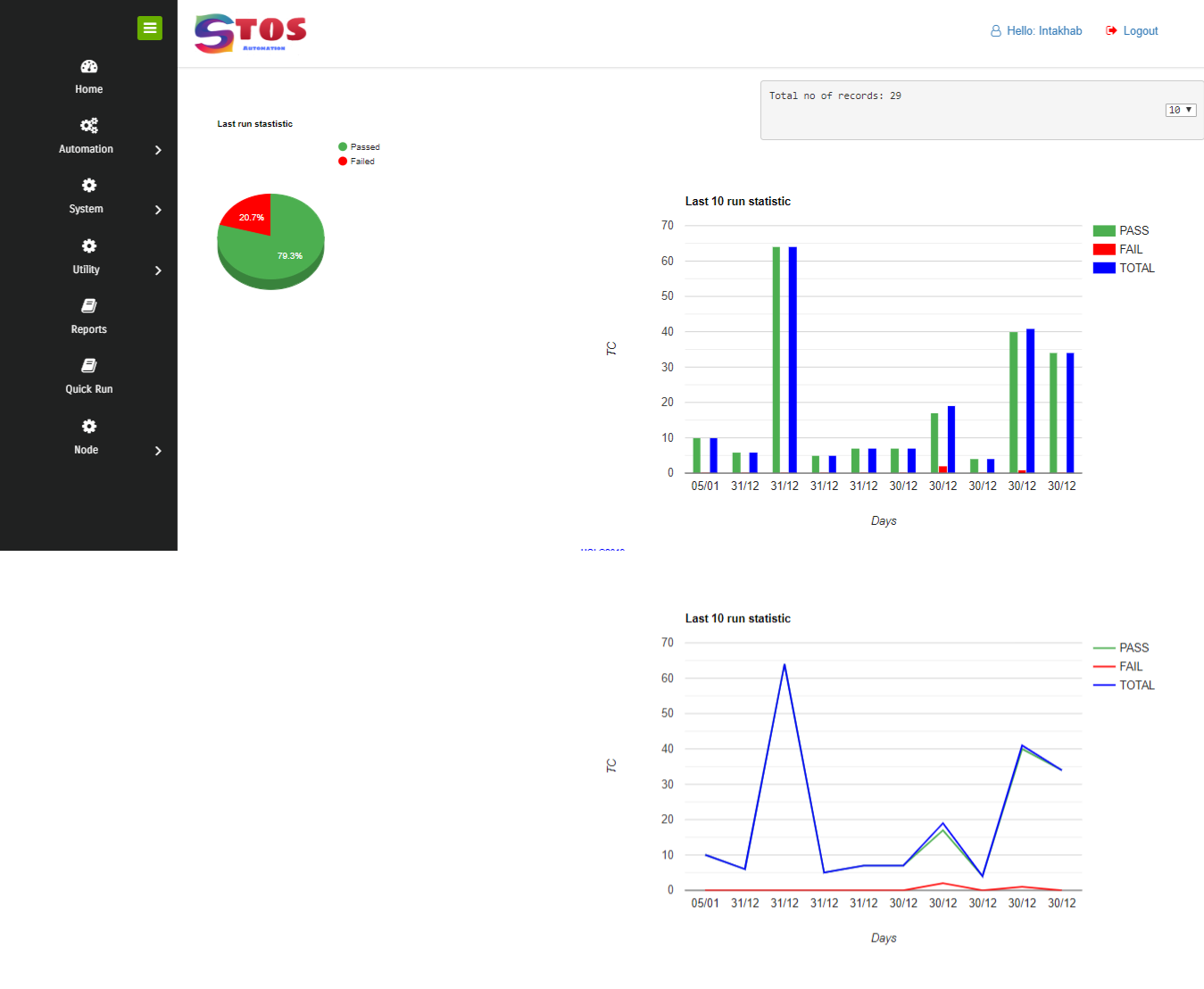
**2.3 User Interface and APIs Over View**



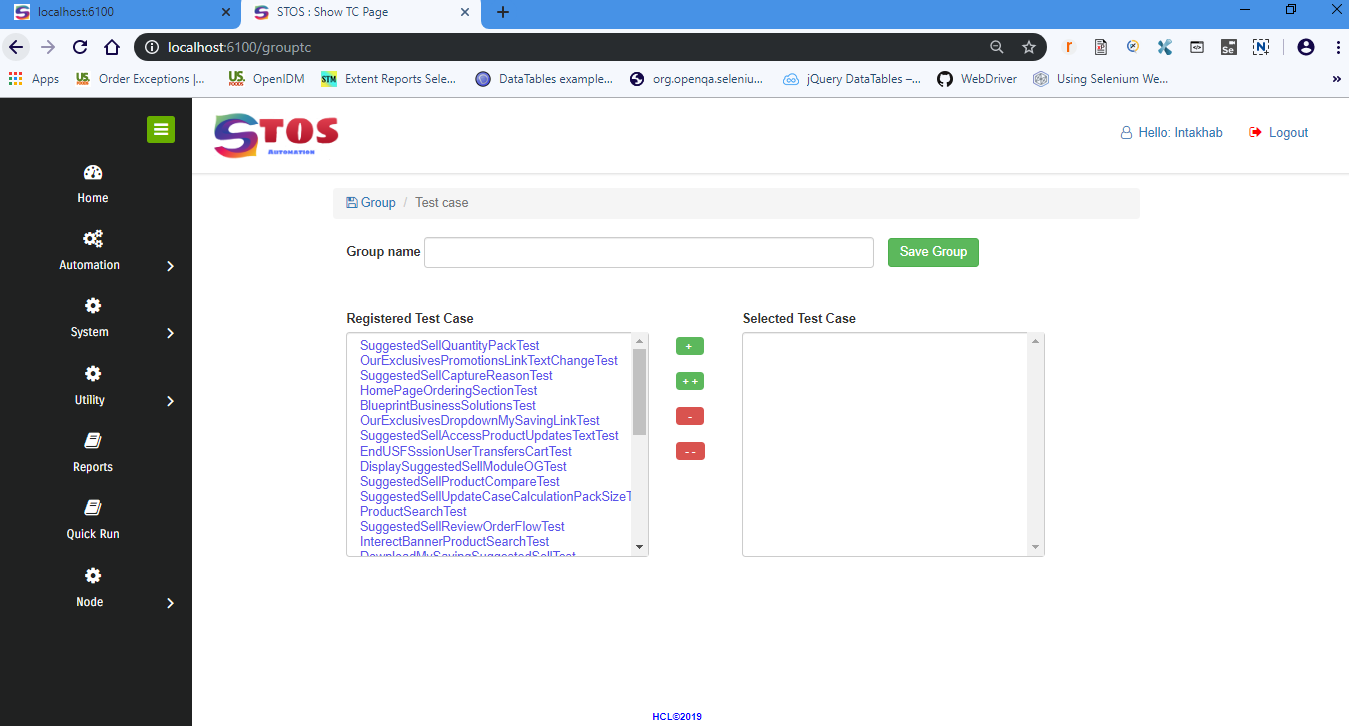
Login



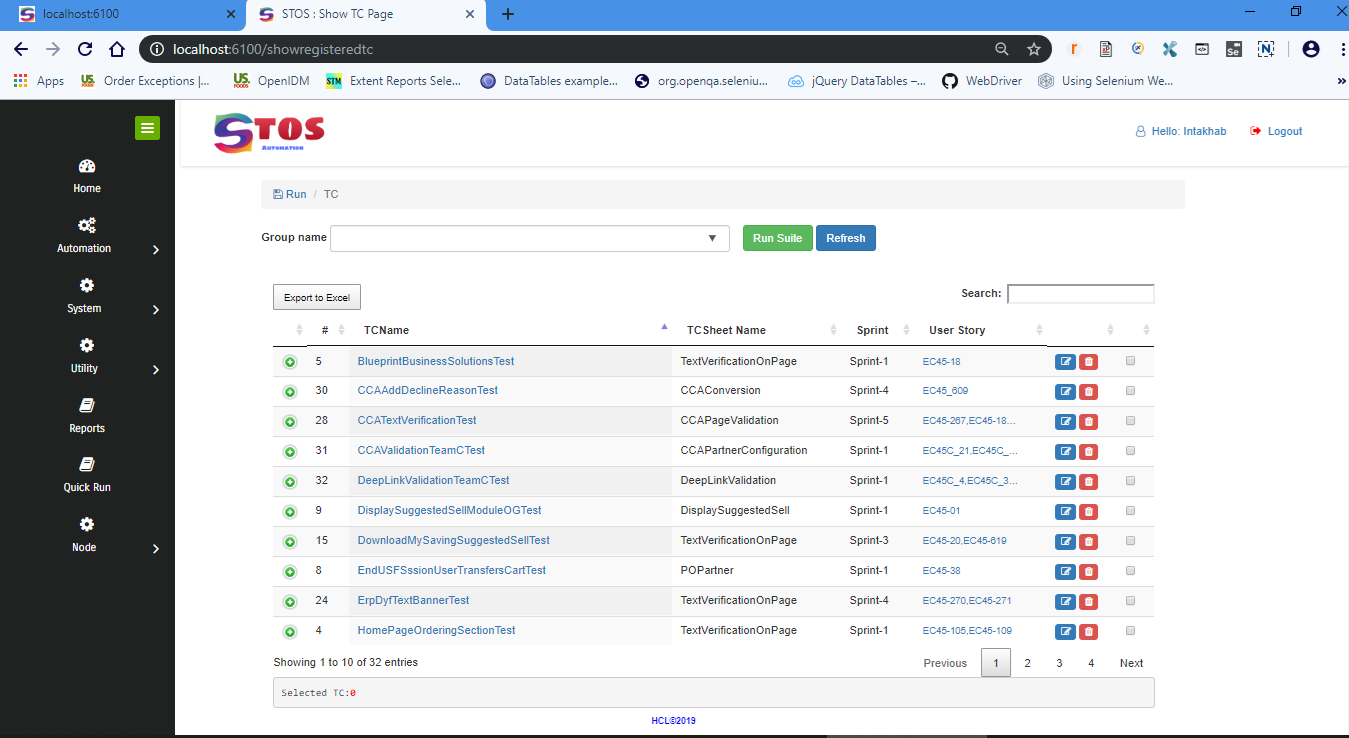
Dashboard(High level/Low Level Reports)



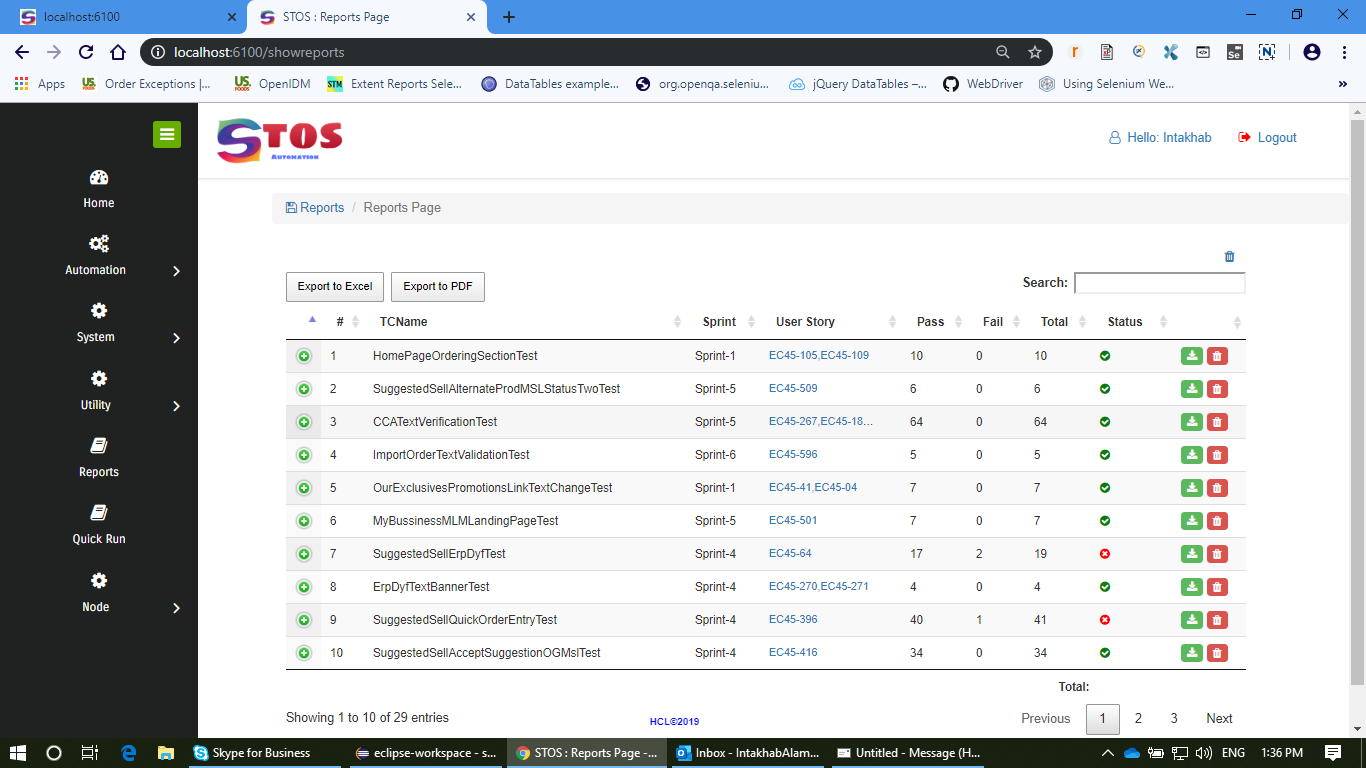
Group TC

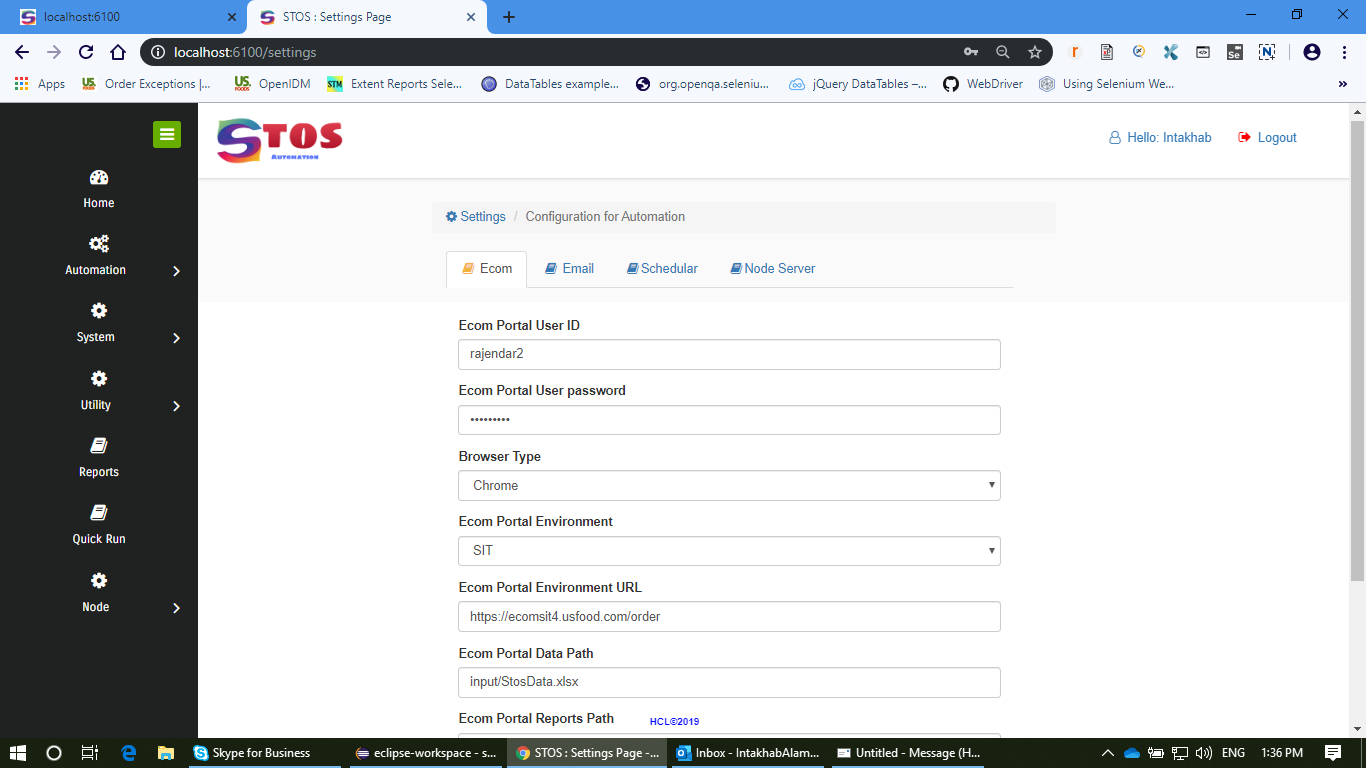


Run TC

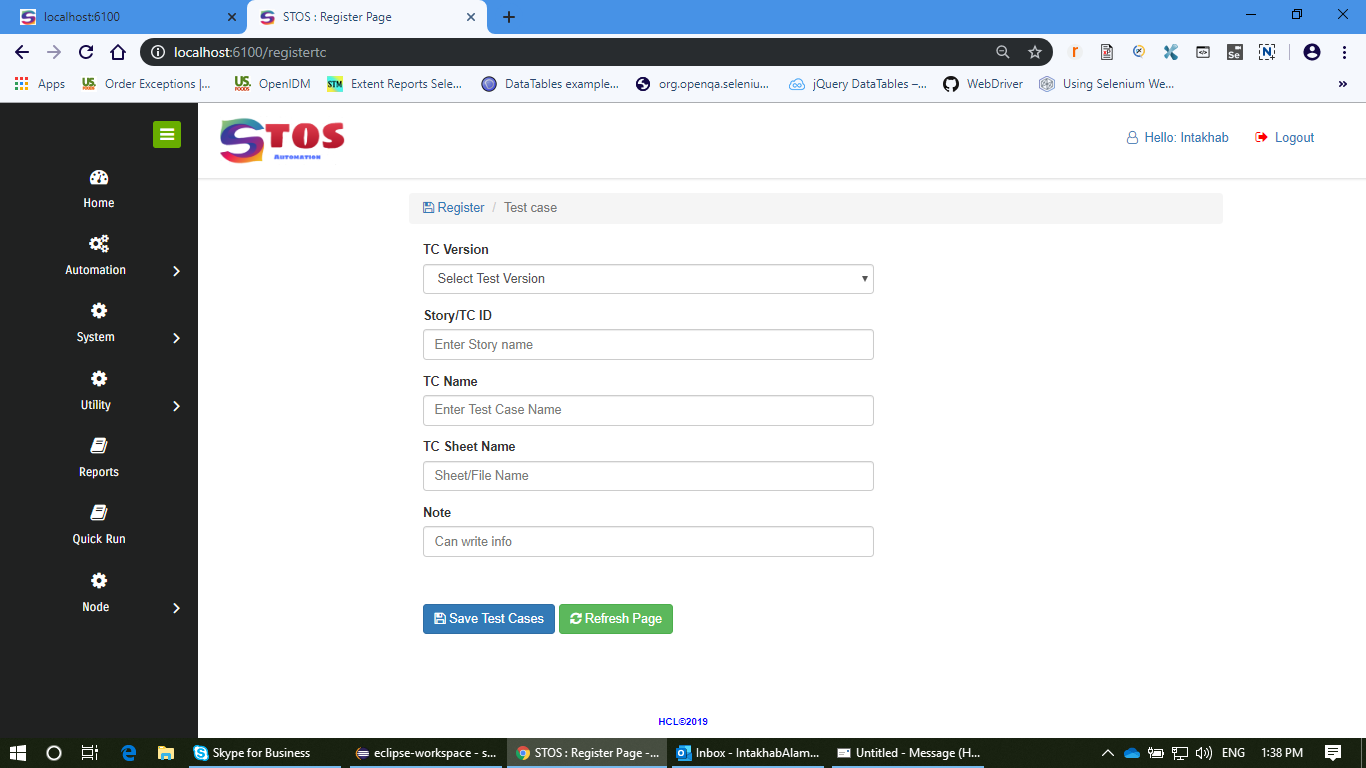


Reports

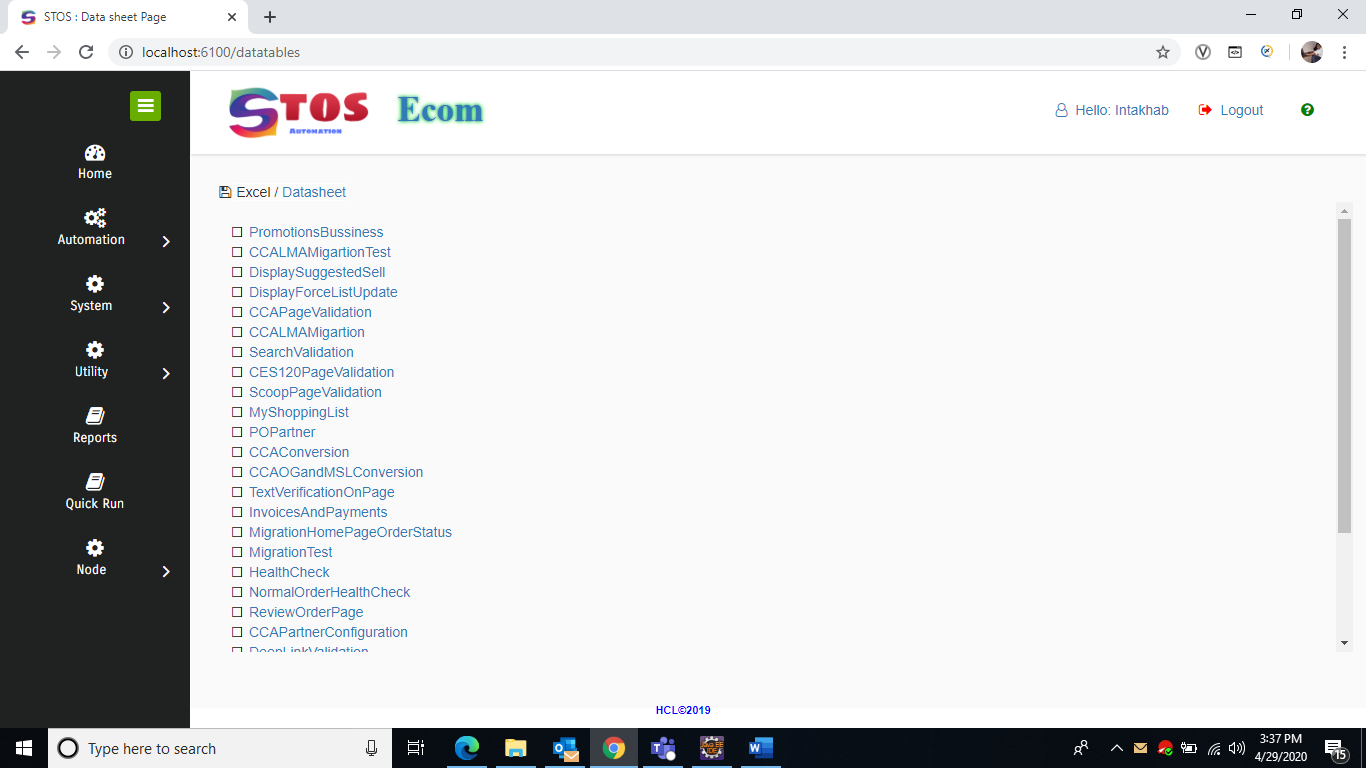


Settings

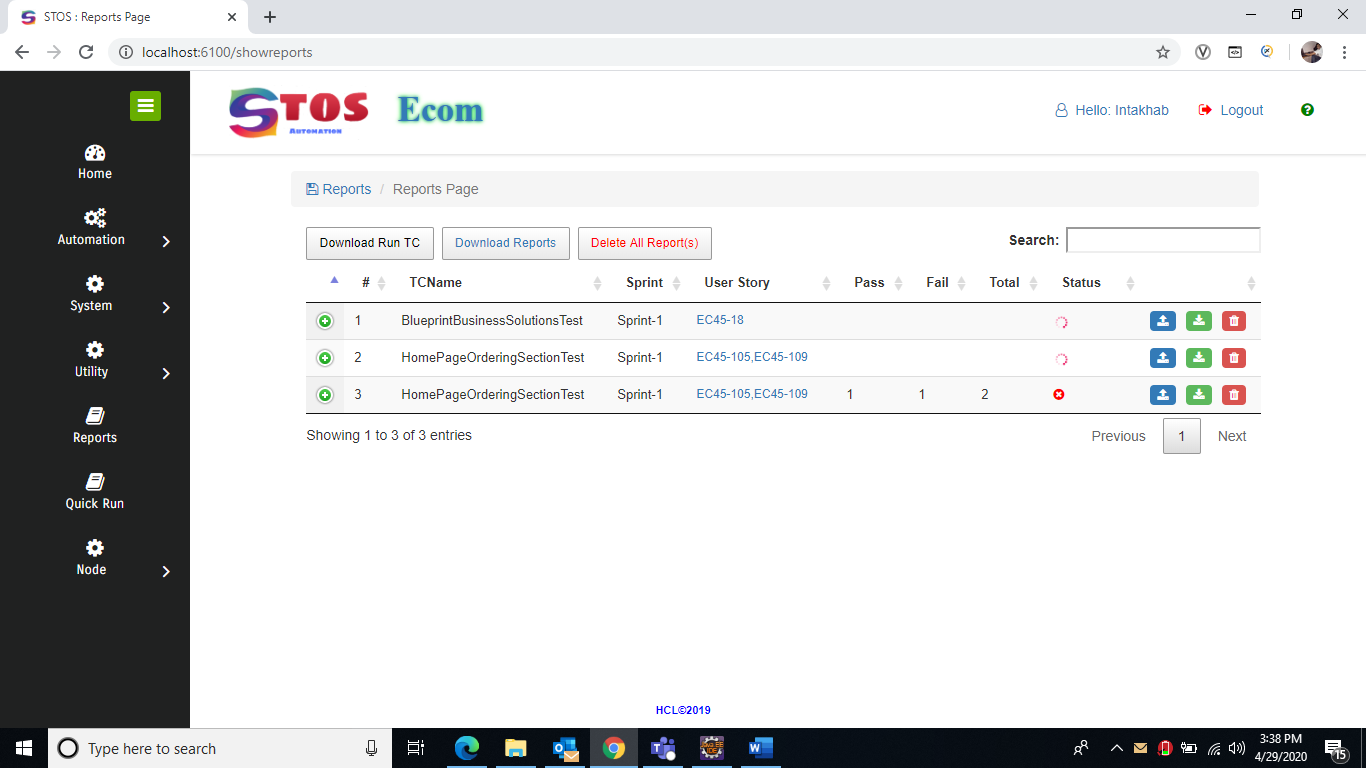
Register TC



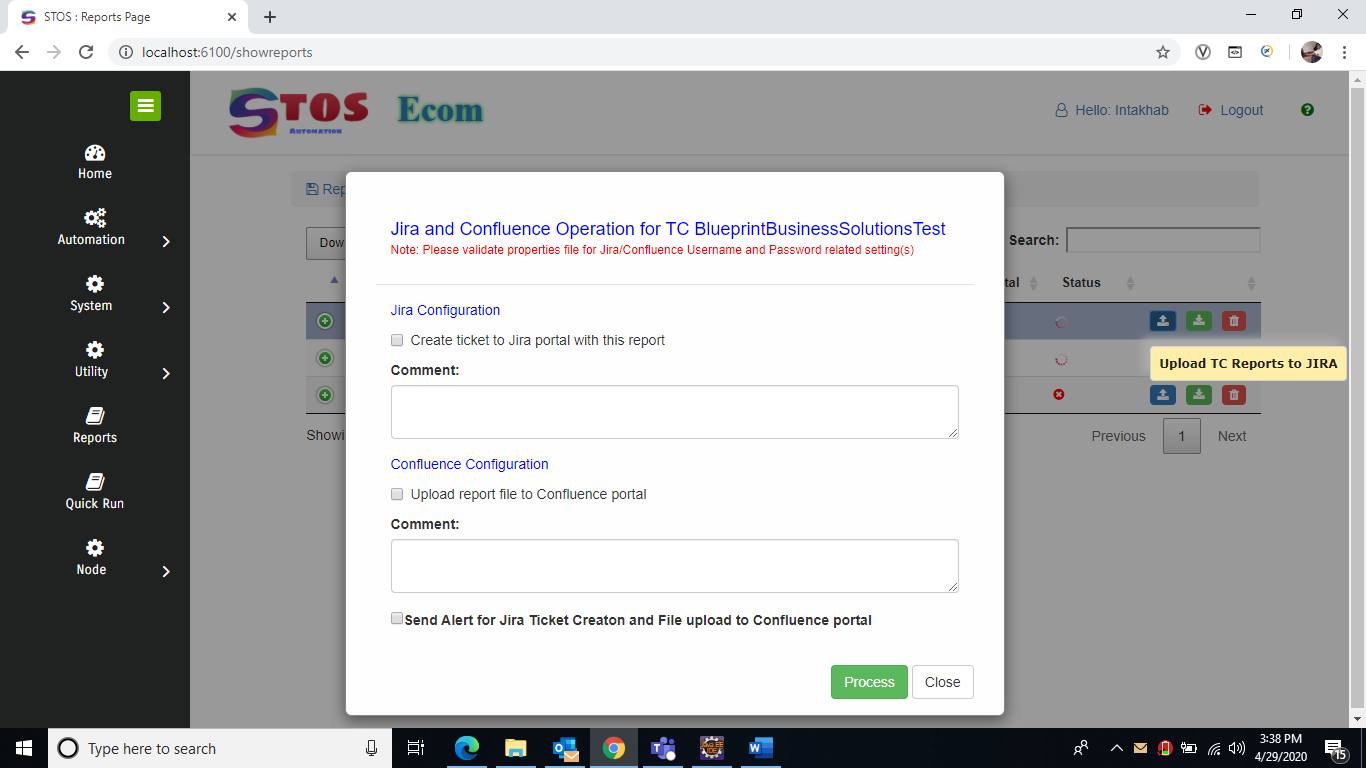
Excel Data sheet



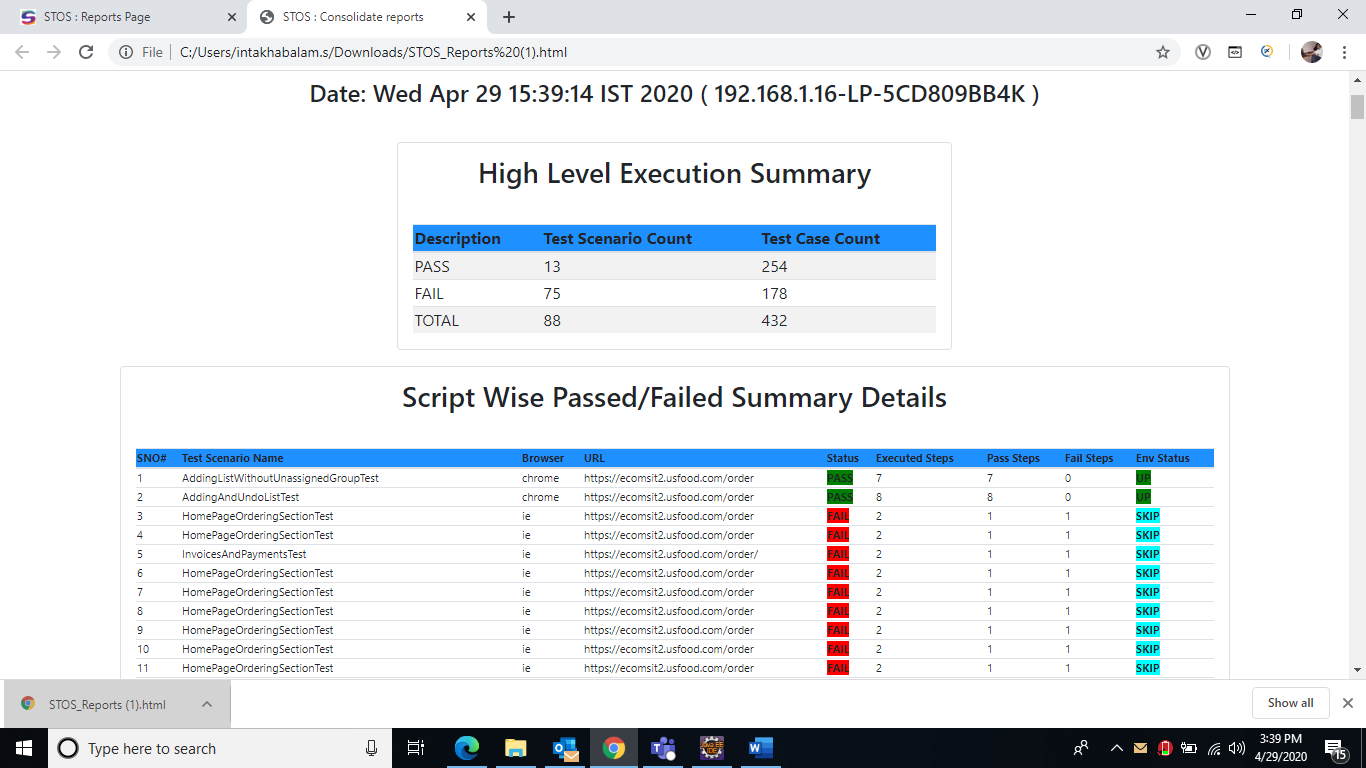
Reports

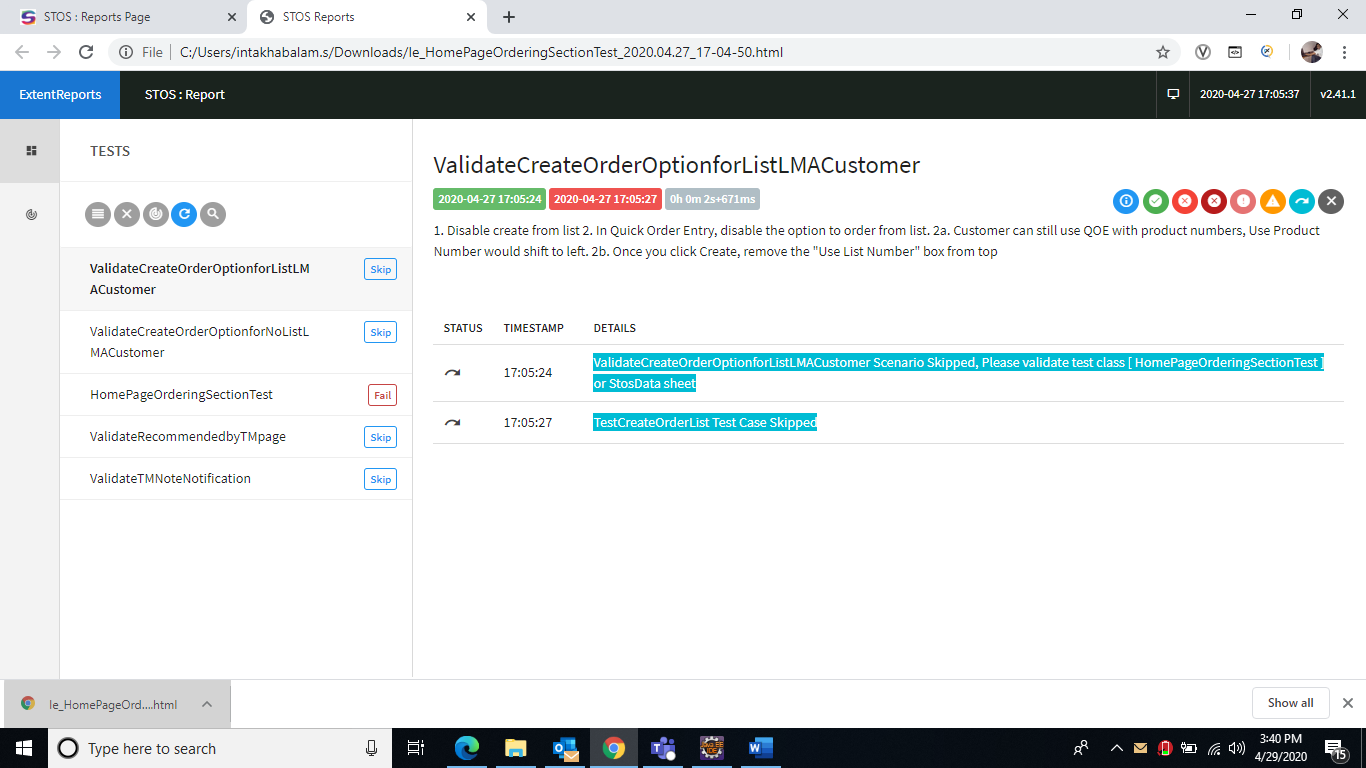


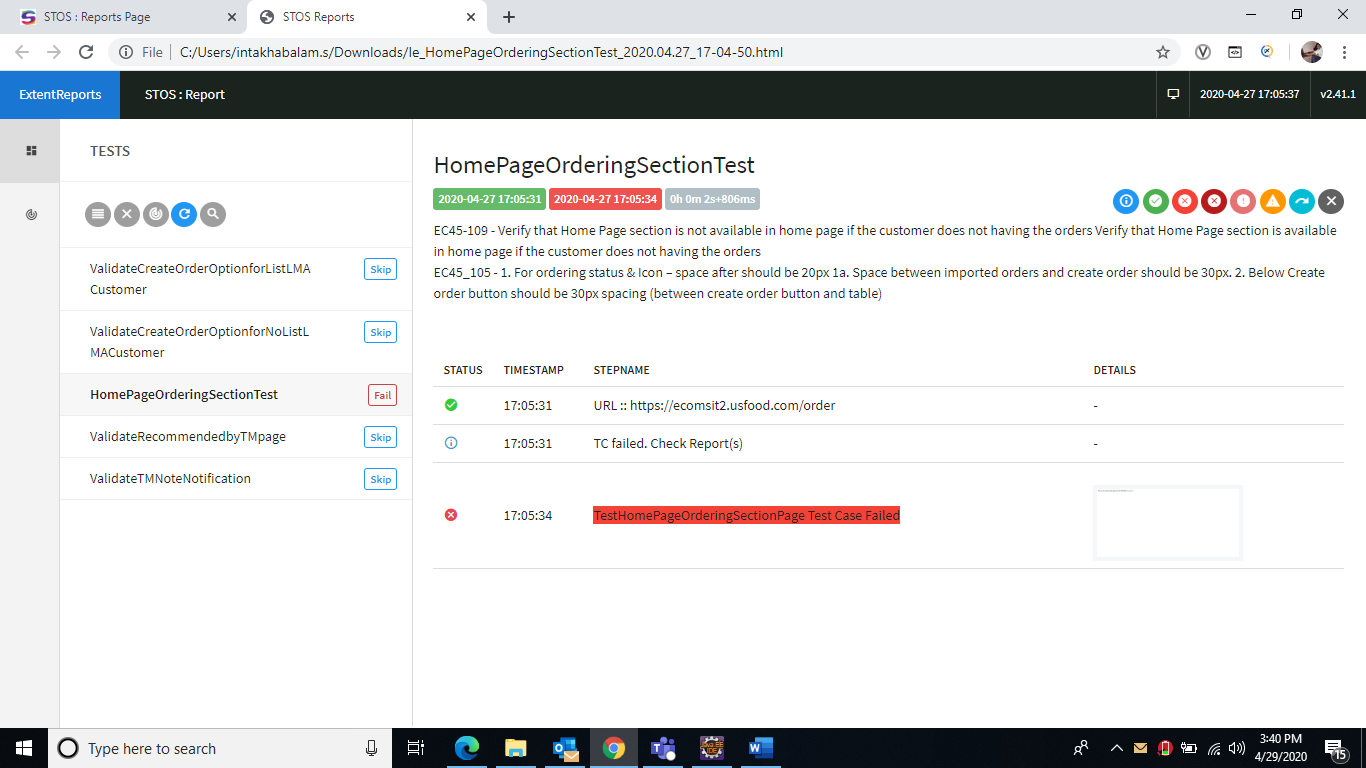
Upload Reports to JIRA/Confulence



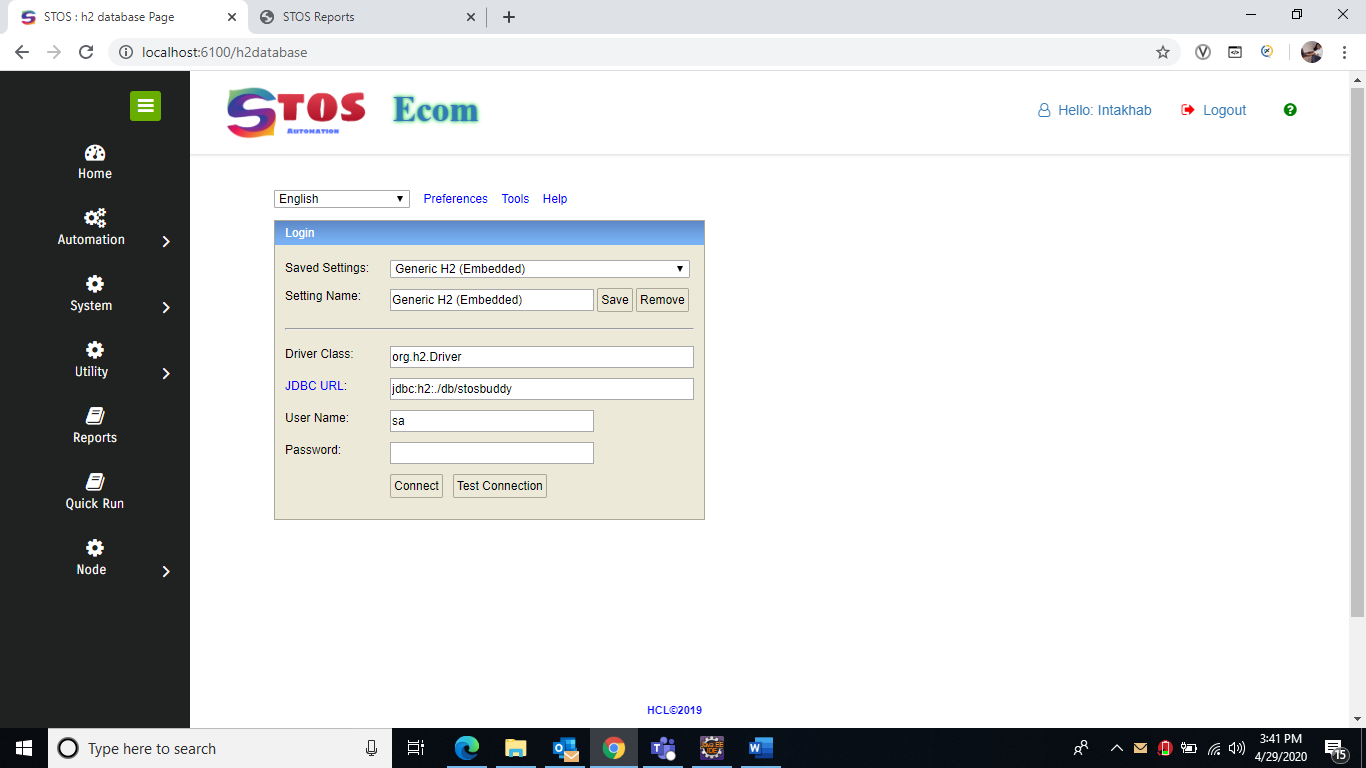
Reports

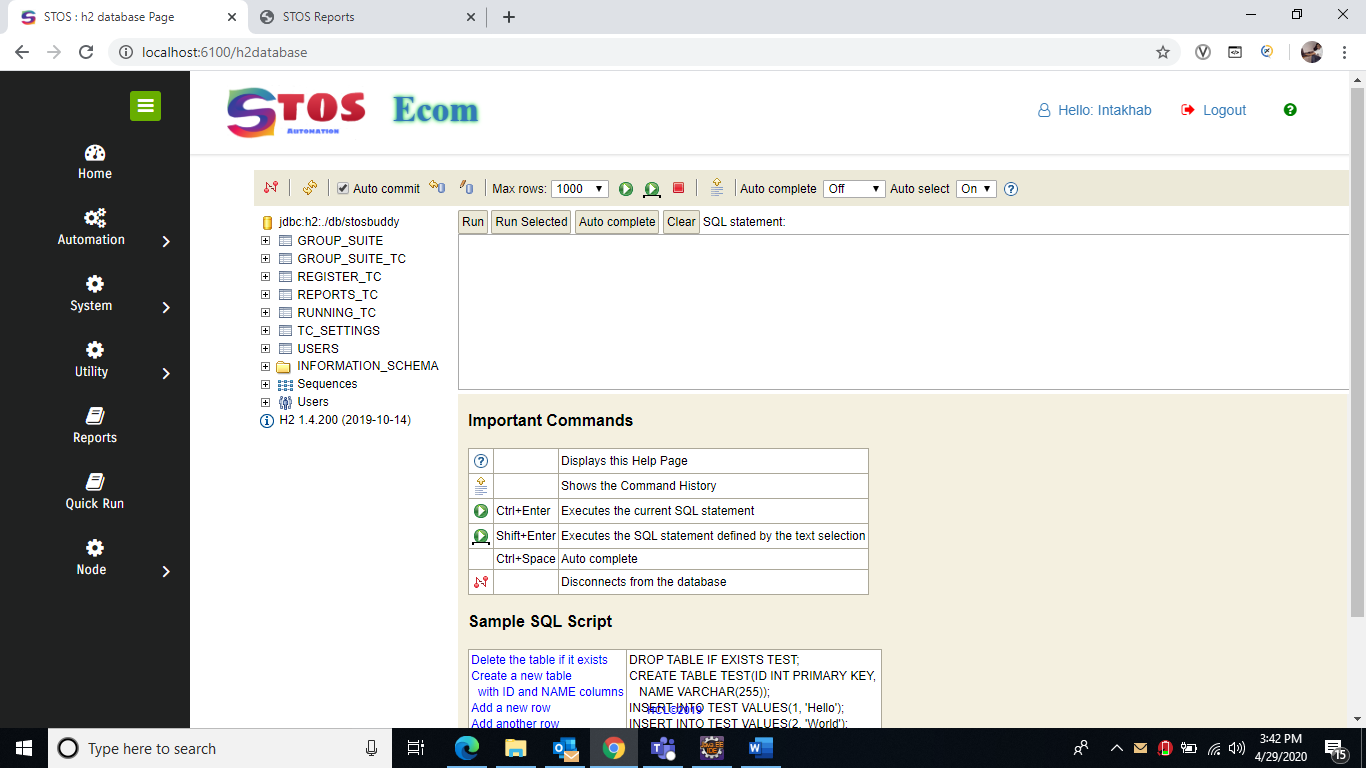


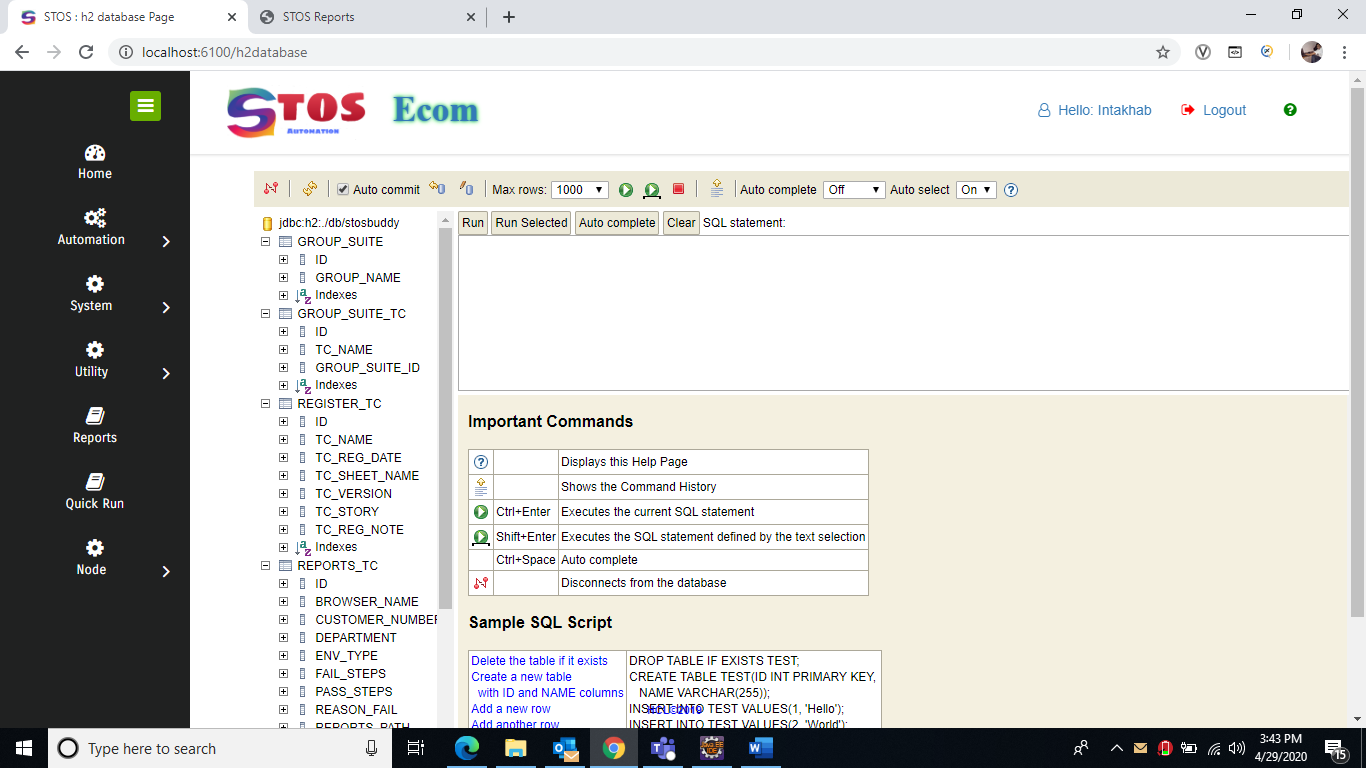




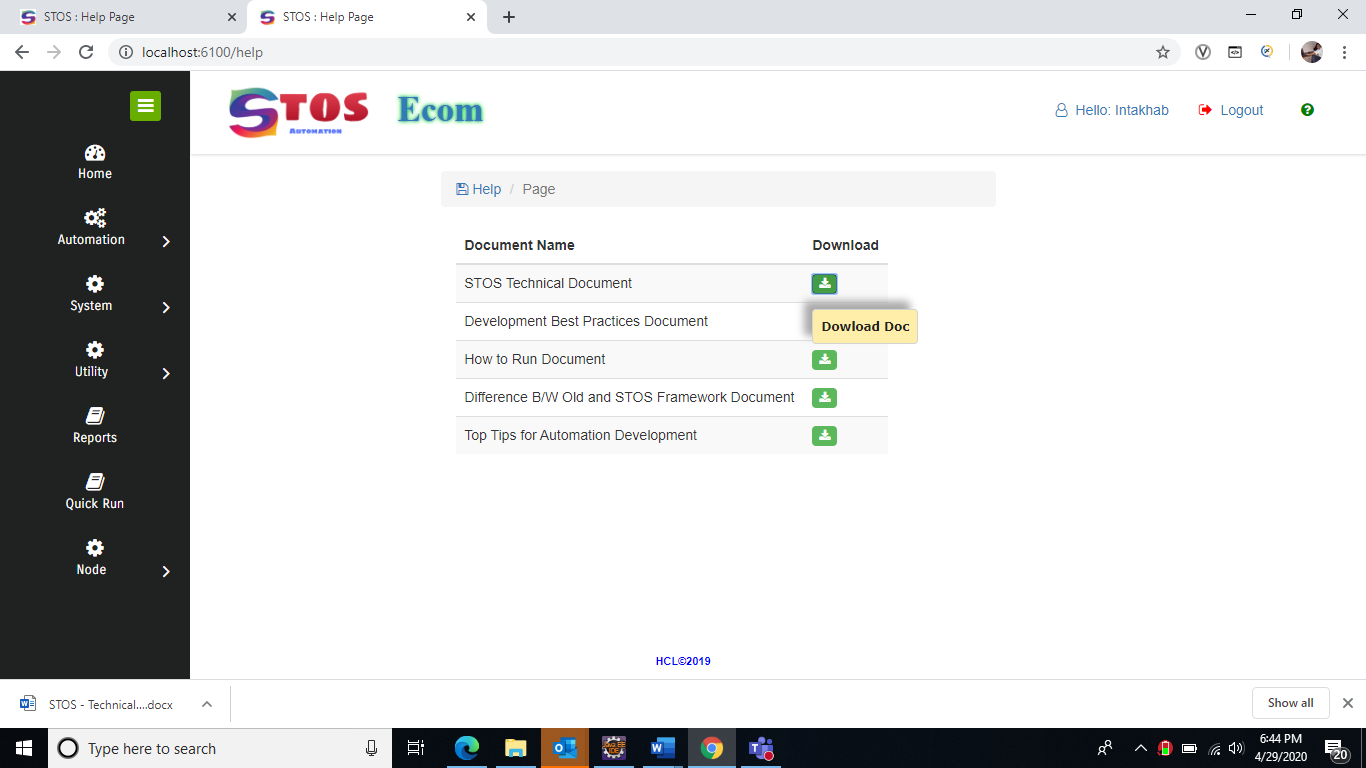
Database (Path copy paste this path ) in JDBC URL=🡺 **jdbc:h2:./db/stosbuddy**







I have uploaded all document at Help Page



1. **SYSTEM DESIGN**

Depending upon the approach to be taken the software may be produced using coding statements or it may be automatically generated by an application development tool or a mixture of both.

**3.1 Design Method and Standards**

In designing objects there are four important ‘cardinal points’ to be remembered. These are:

1. Attributes – what the object knows
2. Methods – what the object does
3. States – the changes that occur due to process flow
4. Events – responses to the outside.

One area where careful attention to design can reap rewards in the longer term is in the interface

between the client and the so-called presentation layer. In implementations based on web

technologies, such as SpringBoot there is much to be gained from separating out elements of the

application server architecture. One approach to this is given in an implementation of the MVC

paradigm using SpringMVC and Spring Core.

**3.2** **Naming conventions**

This section should explain all naming conventions used, and draw attention to any points a

maintenance programmer would not expect. A table of the file types and the permitted names or

extensions for each is recommended for quick reference. Conventions for naming files, programs,

modules, and possibly other structures such as variables and messages, should all be documented

here.

|  |  |
| --- | --- |
| Best Practices |  |
| How to connect JIRA/Confulence |  |

**3.3** **Programming Standards**

Whatever languages or standards are chosen, the aim should be to create a convenient and easily usable method for writing good-quality software.

If an application development tool is used there may be other conventions that need to be defined, e.g. colour schemes. In general, the programming standard should define a consistent and uniform programming style. Specific points to cover are:

* 1. Modularity and structuring
  2. Headers and commenting
  3. Indenting and layout
  4. Library routines used

**3.4 Software used**

1. Java 8
2. Bootstrap 4.
3. jQuery
4. SpringBoot 2.0
5. Spring MVC
6. Tomcat webserver
7. JMX (Java Management Services)
8. H2 Database
9. Java Mail
10. Quartz

**3.5 Requirement and Specification**

1. Win/Linux operating system with minimum 8 GB RM
2. Java 8 in System Class Path
3. Modern Browser (Chrome/Mozilla and Microsoft Edge)
4. **DOCUMENT CONTROL**

|  |  |
| --- | --- |
| **Title:** | Technical Design Document |
| **Issue:** | Issue 1 |
| **Date:** | 28th April 2020 |
| **Author:** | Intakhab Alam Siddiqui |
| **Distribution:** | @USFOODS ECOM Project Team |

Document Signoff

|  |  |  |  |
| --- | --- | --- | --- |
| **Nature of Signoff** | **Person** | **Date** | **Role** |
| Authors | Intakhab Alam Siddiqui |  | Senior Technical Lead |
|  |  |  |  |
|  |  |  |  |

Document Change Record

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Author** | **Change Details** |
| 28th April 2020 | V1 | Intakhab Alam Siddiqui | First complete draft |