**1.INTRODUCTION**

**1.1 Introduction To The Company – iNAUTIX**

**1.1.1 Introduction**

iNautix Technologies India Private Limited is a group company of Bank of New York Mellon - a leading financial services provider. iNautix provide technology development, business & technology operations and remote infrastructure management services for BNY Mellon and its subsidiaries. iNautix also develops and delivers comprehensive technology solutions and software development products for customers of BNY Mellon.

**1.1.2 Founders**

The Bank of New York Mellon was founded by Alexander Hamilton in 1784. Gerald Hassell is an American bank executive and is the Chairman and CEO of the Bank of New York Mellon from 2011. Suresh Kumar is the CEO of iNautix Technologies.

**1.1.3 About Company**

             iNautix Technologies, established in the year 2000,provides technology solutions to BNY Mellon. As an innovation-focused company, iNautix specializes in providing insightful and new age technology solutions to help meet clients' business needs end-to-end. Since its inception in 2000, iNautix has maintained a relentless commitment to develop products and solutions that are designed to improve operational efficiencies, make planning more dynamic and realistic, monitoring and evaluation simple, risk-free and methodical, increase productivity, enhance delivery performance and ensure success for our clients.

**1.2 Background of the Project**

In the corporate world, each organization has as much as thousands of applications; these would have been typically written in numerous languages which in turn run in diversified platforms. Despite of these complications, if we are still able to maintain these applications and provide the user, the best experience it is because of logging. The process of making an entry in the log file is called logging. These logs generated during the execution of an application, can be recorded in multiple destinations such as a local disk, console, into other sockets or files. Analysis of these log files is necessary later during debugging or enhancement of the respective application.

Logging is crucial for a developer, who has to modify the application, to add new functionality, or modify the existing one, which has been developed years ago. The main components of logging are: Logger, Formatter, and Appender. The Logger captures the information with metadata that has to logged. Formatter formats the object. Appenders publish the log information into a variety of ConsoleAppender, FileAppender, RollingFileAppender, and SocketAppender.

A logger can be associated with different levels of severity such as fatal, error, warning, info, debug, trace in the descending order. So a logger if configured to warning level would log all levels above it. Logger has overhead associated with it, so developer has to be judicious on what is being logged. Log manager creates the logger and manages corresponding configuration files.

In this application, extraction of log files and corresponding analysis is done. In the corporate world, each organization has its datacenters in different geographical locations far from each other, in order to withstand any natural calamities. Each datacenter in turn has numerous severs associated with it. Each application runs in some hosts of the server, produces log files as the application is being executed.

Our application helps the developer and the support team to derive the relevant files from loads of log files by specifying the severity level and the keywords, which can be converted into pie charts, bar graphs, and text files for further debugging, and analysis. The developer can also add searches which are frequently used. It also displays the recently used queries. The developer can also search for any particular type of error on any application, it is termed as raw event searches . Thus this application can handle sparse searches and also dense searches.

This flexible modular application can handle complex queries by matching patterns and can be customized to organizational needs. This application can be invoked whenever necessary.

**2. SOFTWARE PROJECT PLAN**

**2.1 Existing System**

Splunk is a log analyser tool. It has both **free and paid plans**. Spunk light, the free plan, allows the user to log up to 500MB data per day. The pricing for paid plans varies based on the amount of data that an user is processing.  It provides facilities like indexing, searching, visualizing, analyzing and reporting of log data. But the system is not customised for any organization so it is dearth of any application specific filters.

* + 1. **Drawbacks**

**Price:**

The costs of the third party vendor applications are cost wise higher and even though the organization gets the service, these organizations cannot be able to work independently on its own pace. Any modifications in those applications can be done only with the vendor support.

**Generic Purpose:**

The existing system is more general, it lacks specialization for any organization. Dynamic customization to an organization is not possible.

**2.2 Proposed System**

In the proposed system, we are going to create a customized log analyser tool which can get rid of the complexity of maintaining the application specific logs. It also provides a way to identify and solve the bugs. The system will be able to filter the logs in real time. The system receives the log files, generated during the user interaction with the application. The log files are analysed and the results are filtered in accordance to the user demand.

An application will be configured once initially. The process includes linking the application, with the servers which assists in running it and with the locations in which the log file will be stored.

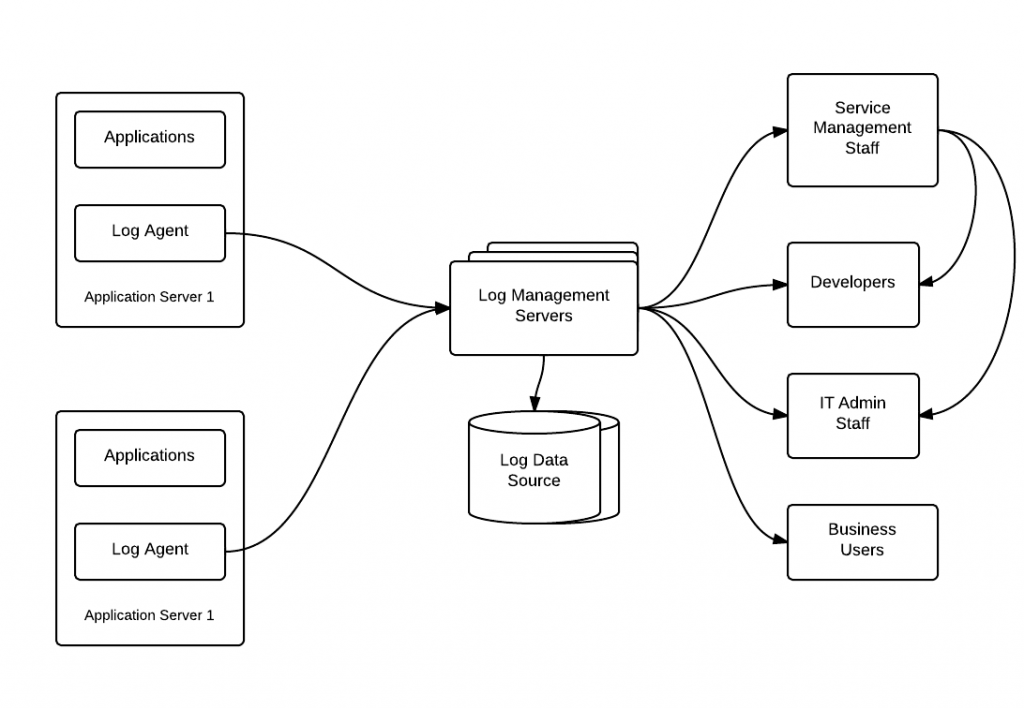
Search is the primary way which enables user to navigate data. Search retrieves events from log file, use statistical commands to calculate metrics and generate reports, search for specific conditions within a rolling time window, identify patterns in your data, predict future trends, and so on. Searches can be saved as reports and used to power dashboard panels. The system supports various ways of searching, such as searching for a particular level of severity of error among multiple applications, or searching a single application for the presence of some keyword. It also allows you to save a frequently used search.

The application indexes the incoming data. It generates numerous files which can be categorized into: Raw data, data in its compressed form; Indexes that point to these files and its metadata.

Alerts are triggered when conditions are met by search results for both historical and real-time searches. Alerts can be configured to trigger actions such as sending alert information to designated email addresses, posting alert information to an RSS feed, or running a custom script, such as one that posts an alert event to syslog.

Reports are saved searches and pivots. You can run reports on an ad hoc basis, schedule them to run on a regular interval, or set scheduled reports to generate alerts when the results of their runs meet particular conditions. You can add reports to dashboards as dashboard panels.

These results can be transformed into multiple formats which include excel, pdf,csv,txt, JSON and can be sent to the support team members for further actions. The trend of the application usage can be visualized through various visual aids such as graph, pie chart, bar graph.

Fig 1.1 : Log Management Architecture

The above figure represents the log management architecture, log agents assist us in the aggregation of log files. Log management server processes the users request for the reports. The developers, Service Management Staff, Support Team can access these log files to get meaningful insights.

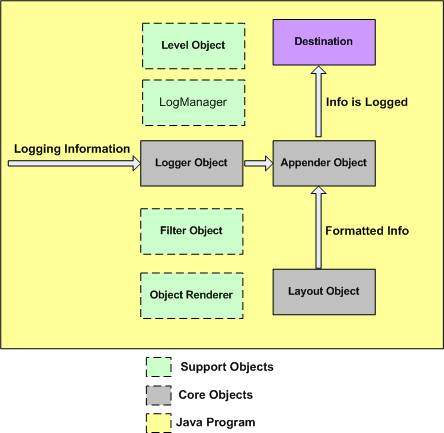


Fig 1.2 : Java Logging Framework

The above figure represents java logging framework ,

**2.3 Modules**

1. Generating Regular Expression
2. Searching
3. Generating reports

**2.3.1 Generating Regular Expression**

This module consists of obtaining the keyword, application name from the user and generates their corresponding regular expression. The regular expression acts as a search pattern for analyzing the list of application names. The regular expression thus generated can be a very generic pattern or a more specific pattern. Depending on the level of the pattern the number of results generated varies. If the pattern is more generic it matches with more number of applications, compared to a more specific pattern.

3

***2.3.2 Searching***

The keyword, which may be a session id, tracker id, module name or the severity level of the log, is obtained from the user is used to search the log file. The Boyer-Moore string searching algorithm is being used to search the large number of text files. The algorithm processes the keyword but not the log files which is being searched. Thus this algorithm best suites the purpose as the length of the keyword is very small when compared to the log files, thus can easily be processed.

***2.3.3 Reporting***

The logs which matches the keyword are produced by the algorithm are displayed in the user interface. The whole log associated with each of the matched result is being displayed which gives the user the complete knowledge necessary for analyzing the issue or arriving at usage analysis, perform audit. This result can also be imported in multiple formats such as pdf, csv, text which provides facility of sending it across organization and also saving it for future reference, it helps in identifying whether the issue is a recurring issue.

**3. LITERATURE REVIEW**

**3.1 Bootstrap framework**

Angular Js 1 uses ng-app directives to point that Angular content has started whereas Angular 2 uses Bootstrapper framework.

The main module contains the bootstrapping code for your Angular 2 application.

**Bootstrap** is an open-source collection of tools which is used in creating websites and web applications. It contains HTML and CSS based design templates for forms, buttons, and navigation and so on, as well as optional Java Script extensions. This aims to ease the development of dynamic websites and also web applications.

Bootstrap is a front end web framework which is an interface for the user, unlike the server-side code which lies on the "back end" or server. Bootstrap comes with several JS(JavaScript) components in the form of jQuery plugins. They provide additional UI(user interface) elements such as dialog boxes, tooltips, and so on. They also extend the functionality of existing interface elements, including auto-complete function for input fields. The following JavaScript plugins are supported by bootstrap: Dropdown, Scroll spy, Modal, Tab, Tooltip, Popover, Collapse, Alert, Button, Carousel and Type ahead.

**3.1.1 Advantages Of Bootstrap**

* Ease of Use
* Highly Flexible
* Responsive Grid
* Comprehensive List of Components
* Leveraging JavaScript Libraries
* Frequent Updates
* Detailed Documentation and Vast Community
* Consistency

**3.2 Maven Dependency**

Maven is a powerful tool that allows users to import dependencies into their software projects and also automatically manage transitive dependencies. In order to use Maven, it is necessary to explicitly add dependencies to the Maven pom.xml file. Once added to the Maven pom.xml file, dependencies will be automatically downloaded, updated, and have their sub-dependencies managed by Maven. The dependencies can be dynamically added and removed from the list and the version of the jar files can be modified at any time during the development of an application. This also ensures uniformity of the versions of files that every developer uses in their module of the same application, thus maintaining consistency

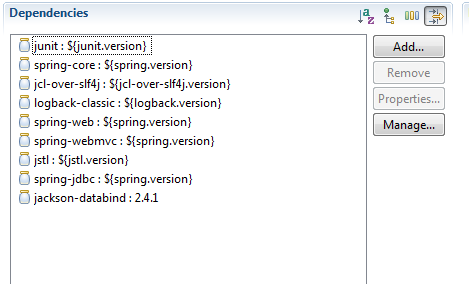


Fig 3.1 : Managing Dependencies

The above screenshot contains some of the core dependency injected and each one is responsible for importing some specific packages. For example, Jackson-databind will import the packages at run time which will convert list into JSON objects and vice versa.

**3.3 Spring MVC Architecture:**

The spring web MVC framework provides model-view-controller architecture and in-built components that could be used to develop flexible and loosely coupled web applications. The MVC pattern helps in separating the different parts of the application (input logic, business logic, and User Interactive logic), by providing a loose coupling between these elements.

**Model -** encapsulates the application data and in general they will consist of POJO.

**View -** responsible for rendering the model data and in general it generates HTML output that the client's browser can interpret.

**Controller -** responsible for processing user requests and building appropriate model and passes it to the view for rendering.



Fig 3.2 : Spring Framework RuntimeDiagram

**3.3.1 Benefits of Spring Framework:**

* Spring helps developers to develop enterprise-class applications using POJOs. The advantage of using only POJOs is that you need have an EJB container product such as an application server instead use only a robust servlet container such as Tomcat Server.
* Spring is organized in a modular structure.
* Spring truly makes use of some of the existing technologies like ORM frameworks, logging frameworks, JEE, Quartz and JDK timers.
* Spring applications are easy to test and simple because environment dependent code is moved into framework. In addition, by using Java Bean-style POJOs, it becomes easier to use dependency injection for injecting test data.
* Spring provides a suitable and compatible API to translate technology-specific exceptions (thrown by JDBC, Hibernate, or JDO) into consistent, unchecked exceptions.
* Lightweight IoC containers are useful for developing and deploying applications on systems with limited memory and CPU resources.
* Spring provides consistent transaction management interface which scales down to a local transaction (using a single database, for example) and scales up to global transactions (using JTA, for example).
* Spring Session provides an API and implementations to manage user's session information. It also provides transparent integration with:

1. Http Session -replaces the Http Session in an application container (i.e.Tomcat)
2. Clustered Sessions - Spring Session support clustered sessions without being connected to application container specific solution.
3. Multiple Browser Sessions - Spring Session supports managing multiple users' sessions in a single browser instance.
4. RESTful APIs - Spring Session provides session ID in headers to work with RESTful APIs.
5. Web Socket - keeps the Http Session alive when receiving Web Socket messages

**3.3.2 Spring IOC Containers:**

The spring container is the core element of Spring Framework. The container creates objects, wire them together, configure them, and manage their complete lifecycle from creation till destruction.

The spring container uses dependency injection (DI) to manage the components which constitute an application. These objects are called Spring Beans .The container gets information on what objects to create, configure, and assemble from configuration metadata provided which can be represented either by XML, Java annotations, or Java code. The following diagram is a high-level view of how spring works. The Spring IoC container makes use of POJO classes and configuration metadata to produce a completely configured and executable application.



Fig 3.3 : Spring IoC Container Layout Diagram

**3.3.3 Dependency Injection(DI):**

The spring framework is recognized and used widely for having Dependency Injection (DI) as flavour of Inversion of Control. Dependency Injection is a concrete example of Inversion of Control.

When we build complex Java application, DAO classes should be independent as possible of other Java classes to increase the reusability and to test independently Dependency Injection helps in connecting these classes together and same time keeping them independent.

Dependency is something which translates into an association between two classes. For example, class A is dependent on class B. Now, Injection is that class B will get injected into class A by the IoC. Dependency injection can be as of passing parameters to the constructor or by post-construction using setter methods.

**3.3.4 Aspect-Oriented Programming:**

Aspect oriented programming (AOP) is one of the key components of spring framework. The functions which span multiple points of an application are called cross-cutting concerns and they are theoretically separate from the application's business logic. There are numerous examples of aspects including logging, declarative transactions, security, and caching.

The key aspect of modularity in OOP is the class, whereas in AOP the unit of modularity is the aspect. The AOP module provides aspect-oriented programming implementation which enables you to define method-interceptors and point cuts to clearly decouple code which implements functionality that should be separated.

* Aspect: modularization of a concern that cuts across multiple classes. Eg :Transaction management
* Join point: a point which occurs during the execution of a program or the handling of an exception. In Spring AOP, it always represents a method execution.
* Advice : Action taken at a particular joinpoint occurred. Different types of advice are “after”, ”before” and “around”. Spring model an advice as an interceptor.
* Pointcut: a predicate which is matched with join points. Advice is associated with a pointcut expression and runs at any join point matched by the pointcut.



Fig 3.4 : Aspect Oriented Programming Layout

**3.3.5 Dispatcher Servlet:**

The Spring Web(MVC) framework is designed with a DispatcherServlet that handles all the HTTP requests and responses. Therequest processing workflow of DispatcherServlet is described in the following diagram

Sequence of events for an incoming HTTP request to DispatcherServlet:

* After receiving an HTTP request, DispatcherServlet goes to handler mapping to call the appropriate Controller.
* The Controller processes request and calls the appropriate service methods based on GET or POST methods. The service method will set model data based on business logic defined and view name is returned to the DispatcherServlet.



Fig 3.5 : Spring MVC Flow Diagram

* With the help of ViewResolver, DispatcherServlet picks defined view for the request.
* DispatcherServlet passes model data to view when finalized which is rendered on the browser.

**3.4 JavaScript**

JavaScript is a lightweight programming language which is used to create network-centric applications. It is complimentary to Java programming language. It is open and cross-platform.

**3.5 JSP**

The Java Server Pages Standard Tag Library (JSTL) is a collection of useful JSP tags which hold together the core functionality common to many JSP applications.

JSTL has support for common applications such as formatting, database access, iterating, manipulating XML documents, structural tasks such as iteration and conditionals, tags for manipulating XML documents, internationalization tags, and SQL tags. It also provides a framework for integrating existing custom tags with JSTL tags.

**3.6 jUnit**

Unit Testing is a software testing method where each components are tested individually before integration. Several tools are available in market for testing the components. Test cases are put into a test suite and they are tested sequentially. jUnit, a unit testing framework has been helpful in test-driven development. jUnit has annotations that facilitate invocation of test methods.

A formal written unit test case is characterized by a known input and an expected output, which is worked out before the test is executed. The known input should test a precondition and the expected output should test a post-condition.

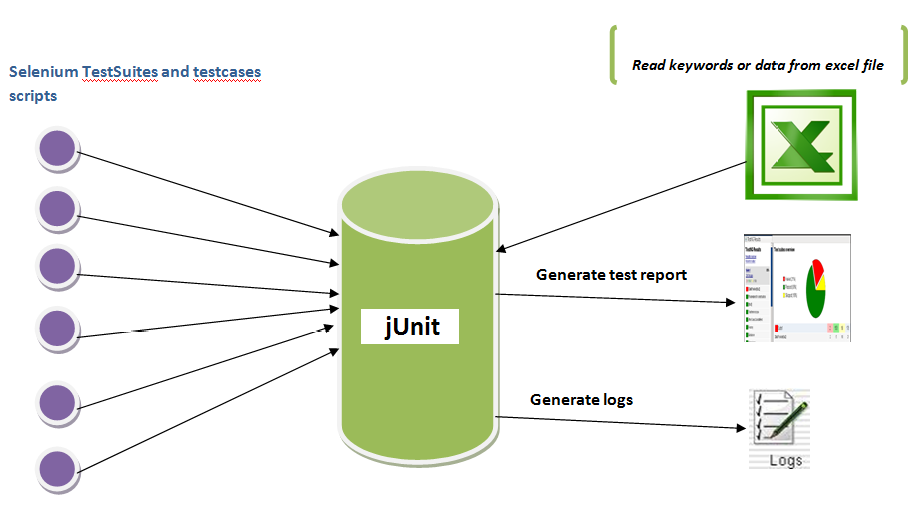


Fig 3.6 : jUnit Architecture

**4. SOFTWARE REQUIREMENT SPECIFICATION**

**4.1 Functional Requirements**

The application’s functional requirements include the number of required input and its format, functions that an application can do and the format of the output.

**4.1.1 Input**

1. If the application is to be configured for the first time then, application name, the list of the servers in which the application will be running, and their location where the log files generated will be stored, and the severity level associated with it and the rollout type of the log file will be provided as input.
2. If the log is to be analyzed then the application gets the application name, for which the log file has to be searched, keyword(s) for segregation of those log files.

**4.1.2 Output**

The system displays the complete segregated logs and also provides an option to import the result as text files or csv and stored in the local disk or in a network drive, which is a storage device located on Local Access Network(LAN) and can be accessed across the organization. This helps in the communication between various departments such as development, support and maintenance easier and tracking the issues faster.

**4.2 NON FUNCTIONAL REQUIREMENTS**

***4.2.1 Performance Requirements***

The key objective is to enable the user an instant access to loads of log file. It helps the developer and the support team to spot the errors and warnings if any and hence proceed with the following debugging.

***4.2.2 Interface Requirements***

The command prompt is used as an interface between the user and system where the user enters the initial key and the input file or text to be encrypted is given.

***4.2.3 Resource Requirements***

* Software Requirements:

Operating System - Windows,Linux

Front End - Web Application

Language - Java

System Type - 32, 64 bit Operating System

Memory - 8 GB RAM

Speed - 3 GHz

* Hardware Requirements:

Processor - Pentium IV or more

RAM - 512 Mb or more

Hard Disk - 50GB or Higher

Monitor - Display Panel (640 x 480)

***4.2.4 Security Requirements:***

The application needs a high level of security because it accesses critical data which includes tracker id, thread id.

***4.2.5 Quality and Reliability Requirements:***

Reliability is a vital requirement of this application as it assists in the further debugging of it.

**5. SYSTEM ANALYSIS AND DESIGN**

**5.1 System Analysis**

**5.1.1 Use Case Diagram**

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals, and any dependencies between those use cases.

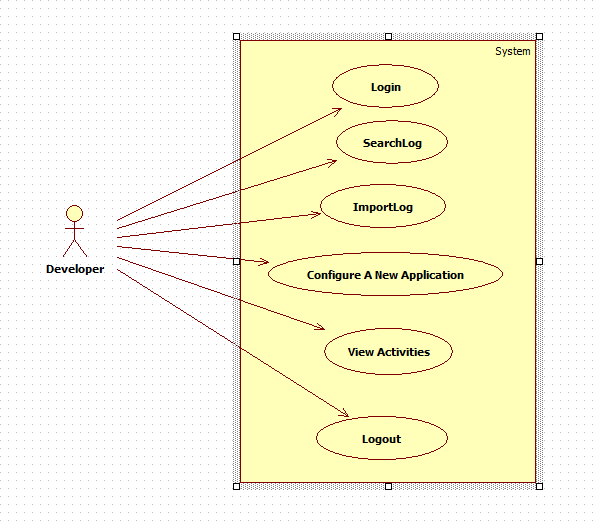


Fig. 5.1 : Use Case diagram

**5.1.2 Class Diagram**

Class Diagrams portrays the static structure of any project. Thus, it is easy for the developers to create plain old java objects using class diagram. Class Diagrams also has a facility to include association, composition and aggregation. Multiplicity or cardinality can also be represented in the class diagram.

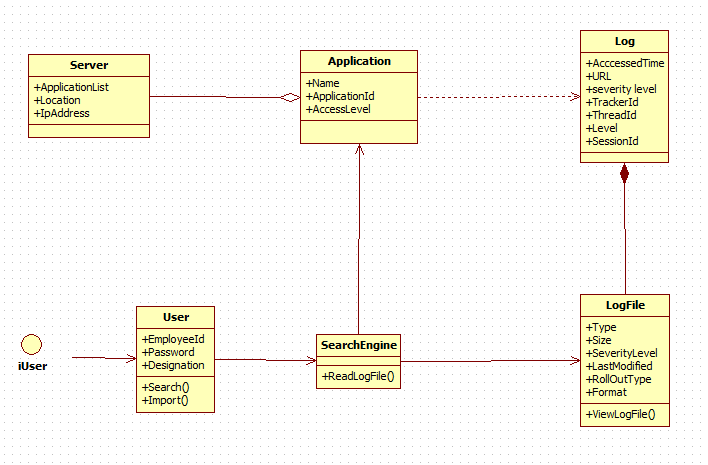


Fig. 5.2 : Class diagram

**5.1.3 Sequence Diagram**

Sequence diagram (also called as event diagram) depict how various objects interact with each other and in which order. In other words, it shows the occurrence or sequence of objects in the application. It displays the object interaction in time sequence.

.

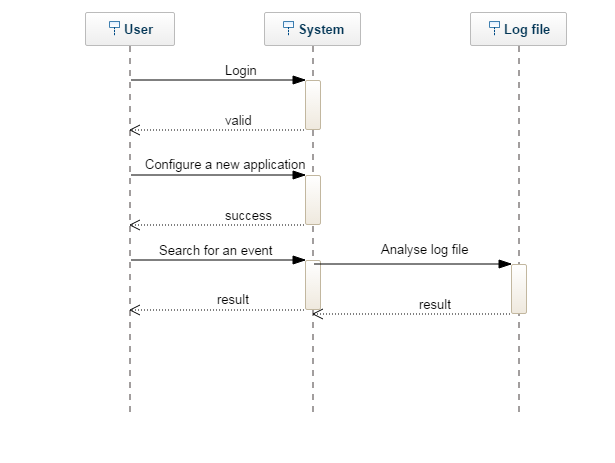
****

Fig. 5.3 : Sequence diagram

The above diagram depicts the sequence of activities that an user will be able to do once he logins into the system.

**5.1.4 Activity Diagram**

An activity diagram is a graphical representation of the flow of activities that an user will go through. It captures the dynamic flow of control. . Activity diagram describes change from one state to another. Constraints, Association and other relationships has to be identified before starting with the activity diagram. The specific usage is to model the control flow from one activity to another. This control flow does not include messages.

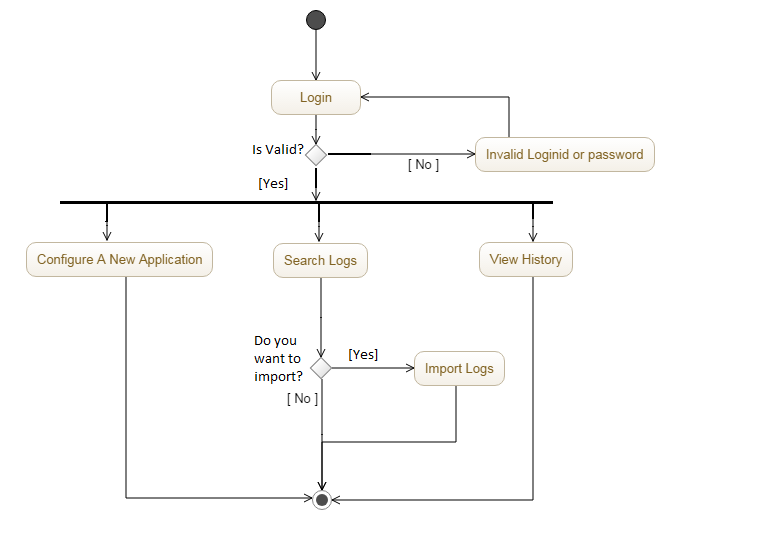
****

Fig. 5.4 : Activity diagram

The user will be able to configure a new application, search the log files and also view the recent activities.

**5.2 System Design**

System design is the process of defining the elements of a system such as the architecture, modules and components, the different interfaces of those components and the data that goes through that system. It is meant to satisfy specific needs and requirements of a business or organization through the engineering of a coherent and well-running system. Systems design could see it as application of systems theory to product development. There is some overlap with the discipline of the system analysis, architecture and engineering. Conceptual design is not the end of design process. It serves as the basis for a brief design.

**5.2.1 Front End Design**

The front end is built using various web technologies which include HTML, CSS, JavaScript, Bootstrap, and Angularjs. HTML helps in creating the basic web pages. Bootstrap is a framework which contains various templates like html forms, navigation and buttons. Angularjs is developed based on JavaScript.CSS is used to add styles that modify the layout of the web pages. JavaScript is used to specify the behavior of the web pages.

**5.2.2 Back End Design**

Apache derby database is used to store the data. It is a relational database management system (RDBMS) developed by Apache Software Foundation which is embedded in Java programs. It has a 2.6 MB disk-space footprint.

**6. CODING**

**6.1 Sample Coding**

***6.1.1 POM.xml:***

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4\_0\_0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>ira.training.xeccq49</groupId>

<artifactId>SpringStudentRestWebService</artifactId>

<packaging>war</packaging>

<version>0.0.1-SNAPSHOT</version>

<name>LogAnalyser Maven Webapp</name>

<url>http://maven.apache.org</url>

<properties>

<jdk.version>1.7</jdk.version>

<spring.version>4.1.1.RELEASE</spring.version>

<jstl.version>1.2</jstl.version>

<junit.version>4.11</junit.version>

<logback.version>1.0.13</logback.version>

<jcl-over-slf4j.version>1.7.5</jcl-over-slf4j.version>

</properties>

<dependencies>

<!-- Unit Test -->

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>${junit.version}</version>

</dependency>

<!-- Spring Core -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-core</artifactId>

<version>${spring.version}</version>

<exclusions>

<exclusion>

<groupId>commons-logging</groupId>

<artifactId>commons-logging</artifactId>

</exclusion>

</exclusions>

</dependency>

<dependency>

<groupId>org.slf4j</groupId>

<artifactId>jcl-over-slf4j</artifactId>

<version>${jcl-over-slf4j.version}</version>

</dependency>

<dependency>

<groupId>ch.qos.logback</groupId>

<artifactId>logback-classic</artifactId>

<version>${logback.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-web</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-webmvc</artifactId>

<version>${spring.version}</version>

</dependency>

<!-- jstl -->

<dependency>

<groupId>jstl</groupId>

<artifactId>jstl</artifactId>

<version>${jstl.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-jdbc</artifactId>

<version>${spring.version}</version>

</dependency>

<!-- Jackson JSON Processor -->

<dependency>

<groupId>com.fasterxml.jackson.core</groupId>

<artifactId>jackson-databind</artifactId>

<version>2.4.1</version>

</dependency>

</dependencies>

<build>

<finalName>LogAnalyser</finalName>

</build>

</project>

**6.1.2 spring-servlet.xml**

beans xmlns=*"*[*http://www.springframework.org/schema/beans*](http://www.springframework.org/schema/beans)*"*

    xmlns:xsi=*"*[*http://www.w3.org/2001/XMLSchema-instance*](http://www.w3.org/2001/XMLSchema-instance)*"* xmlns:context=*"*[*http://www.springframework.org/schema/context*](http://www.springframework.org/schema/context)*"*

    xmlns:mvc=*"*[*http://www.springframework.org/schema/mvc*](http://www.springframework.org/schema/mvc)*"*

    xsi:schemaLocation=*"http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd*

[*http://www.springframework.org/schema/context*](http://www.springframework.org/schema/context)[*http://www.springframework.org/schema/context/spring-context.xsd*](http://www.springframework.org/schema/context/spring-context.xsd)

[*http://www.springframework.org/schema/mvc*](http://www.springframework.org/schema/mvc)[*http://www.springframework.org/schema/mvc/spring-mvc.xsd*](http://www.springframework.org/schema/mvc/spring-mvc.xsd)*"*>

    <context:component-scan base-package=*"main.services"* />

    <mvc:annotation-driven />

     <mvc:default-servlet-handler/>

      <bean id=*"dataSource"* class=*"org.springframework.jdbc.datasource.DriverManagerDataSource"*>

              <property name=*"driverClassName"* value=*"org.apache.derby.jdbc.ClientDriver"* />

              <property name=*"url"* value=*"jdbc:derby://172.24.18.16:1527/book"* />

              <property name=*"username"* value=*"user"* />

              <property name=*"password"* value=*"pwd"* />

       </bean>

     <bean id=*"issuesDAO"* class=*"main.services.IssuesDAO"* />

</beans>

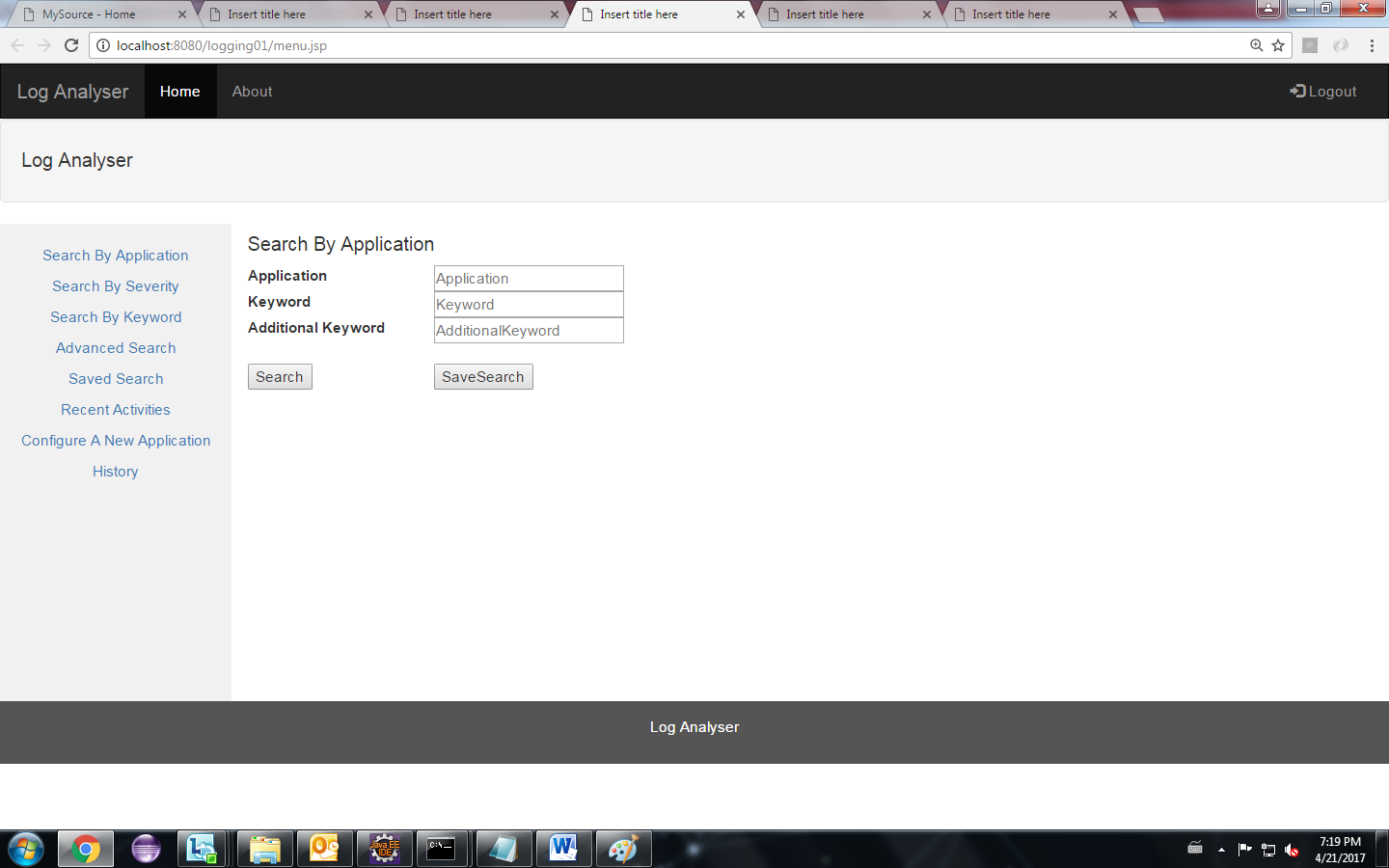
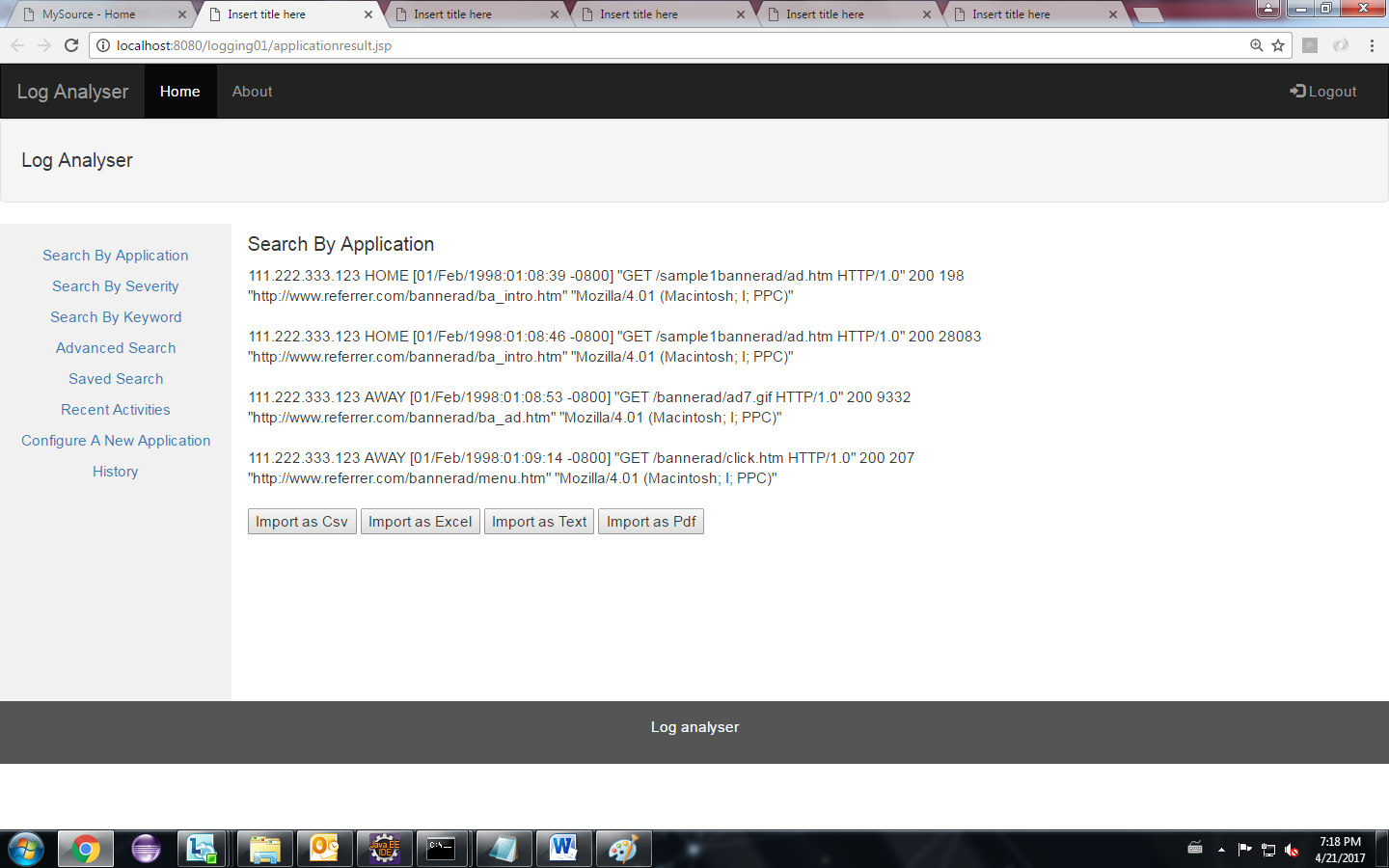
**7. OUTPUT SNAPSHOTS**

Fig 7.1 : Search by application

* This search page is used to segregate the logs of any single application.
* The application name, keyword, and an optional additional keyword is obtained. The keyword may be a tracker id, session id, session id and so on.
* The save search enables the user to save the search for future use.

****Fig 7.2 : Search Result

* This page displays the result obtained from search query that has been made in the previous page.
* It also provides the option of saving the result as pdf, csv, text, excel.

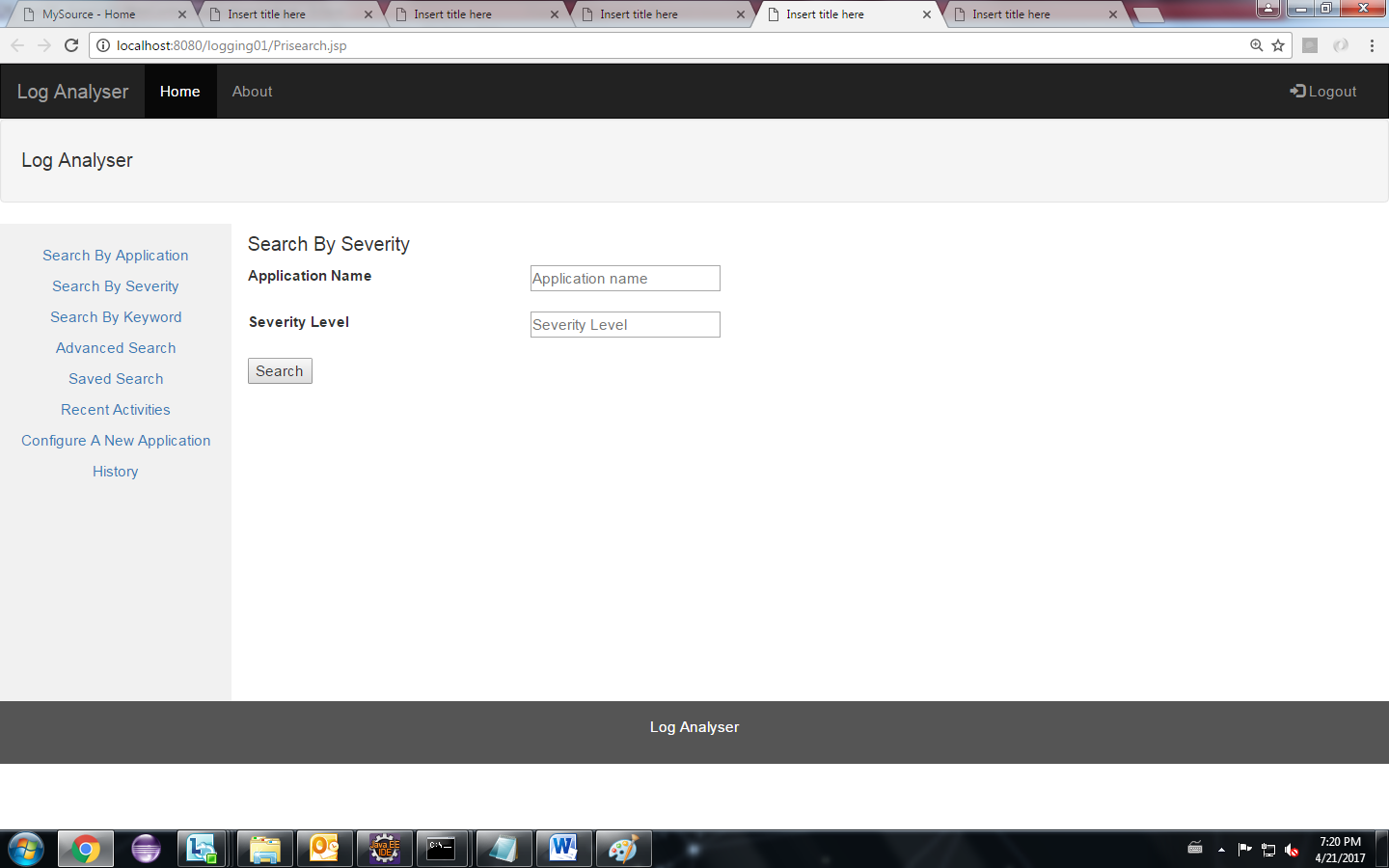
****

Fig 7.3 : Search by Severity

* The search by severity page allows the user to search an application for a particular severity level.
* The logs for that particular seveirty level will be displayed.

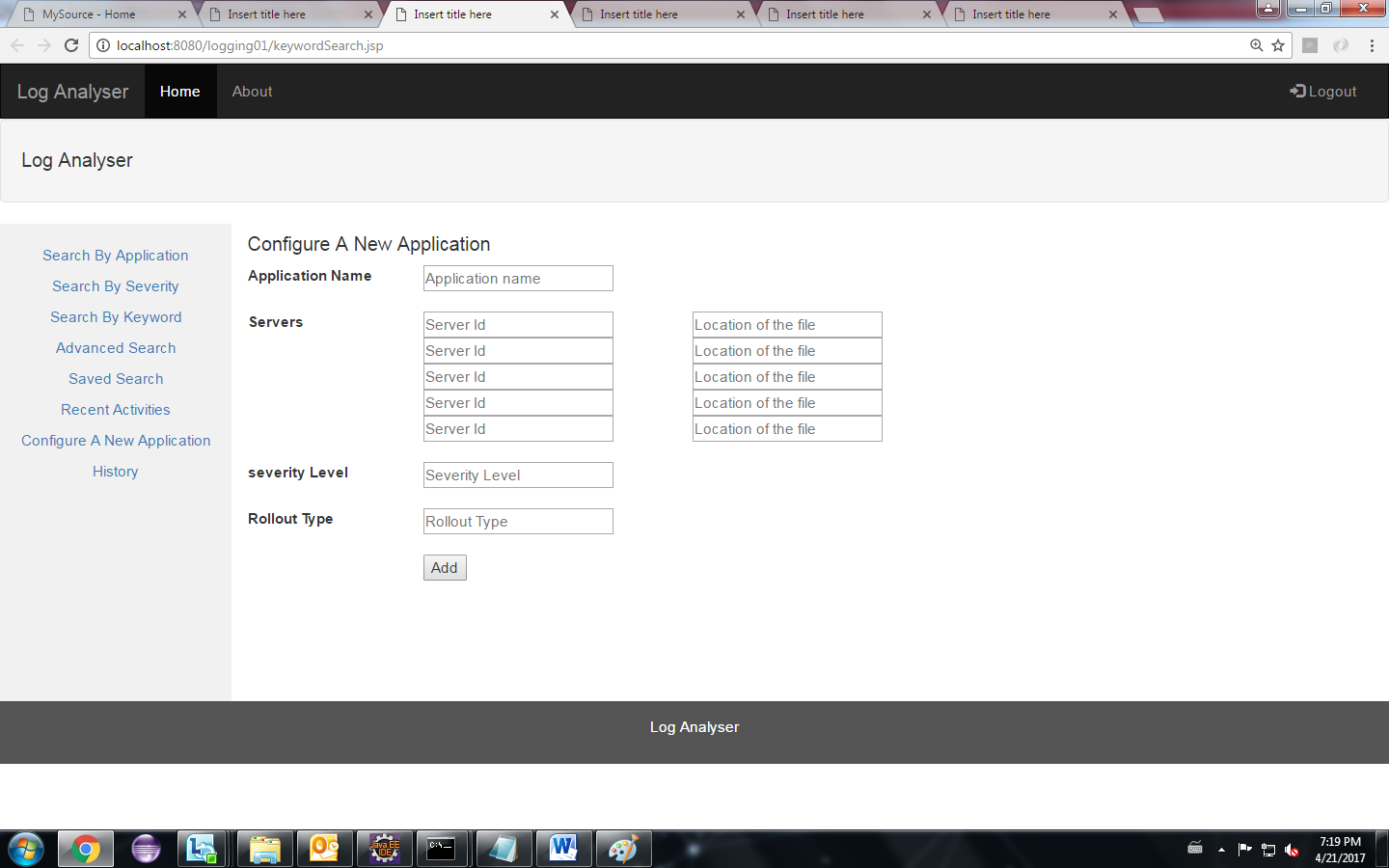
****

Fig 7. 4 : Configuring a New Application

* This page is used to configure a new application.
* The inputs for configuring a new application are application name, server id and their corresponding location of their log files, severity level, Rollout type.

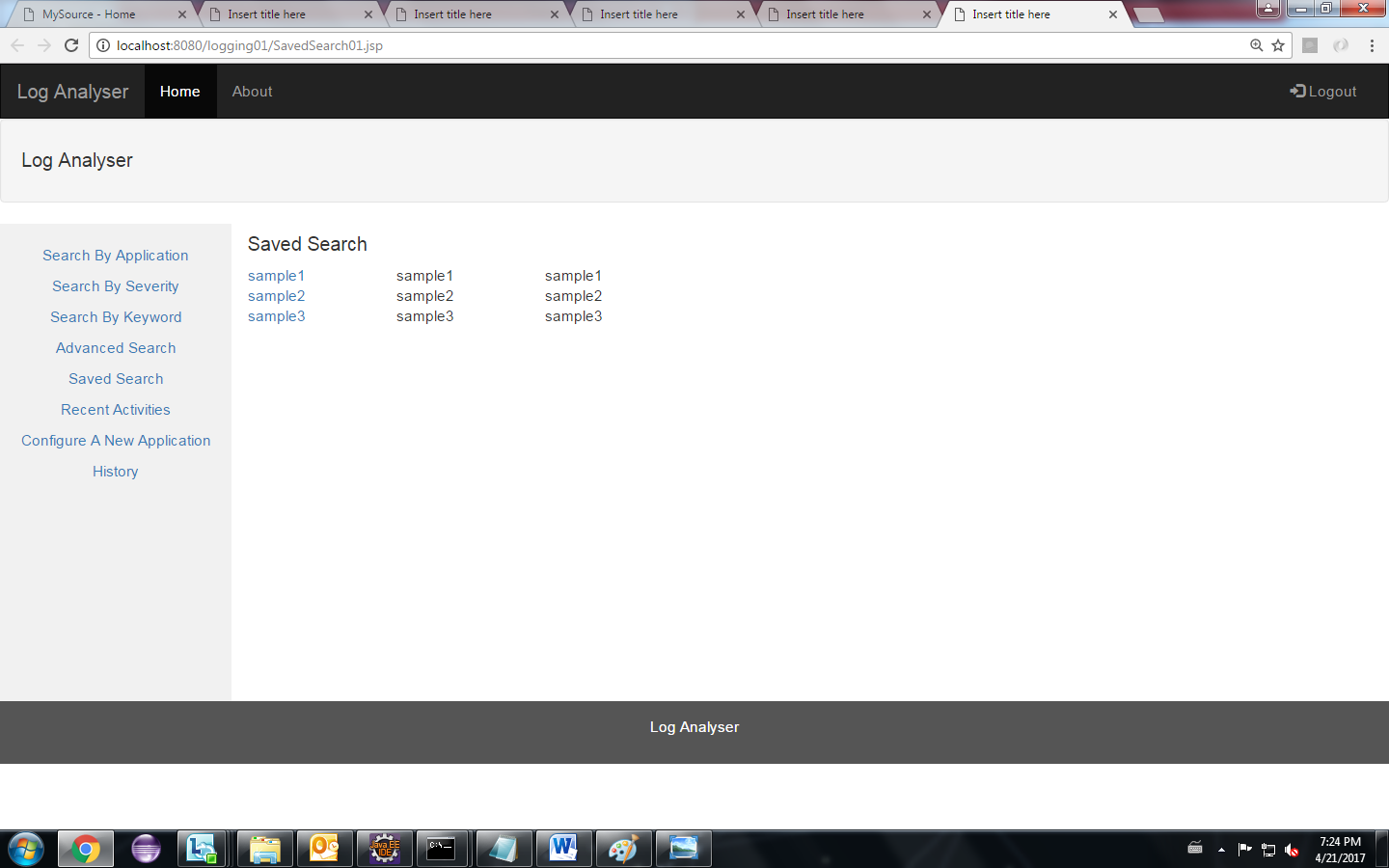
****

Fig 7.5 : Saved Searches

* This page displays the list of saved searches that has been made so far.
* If the user clicks on any of the search then the same query is made again and the new result will be updated.

**8. TESTING**

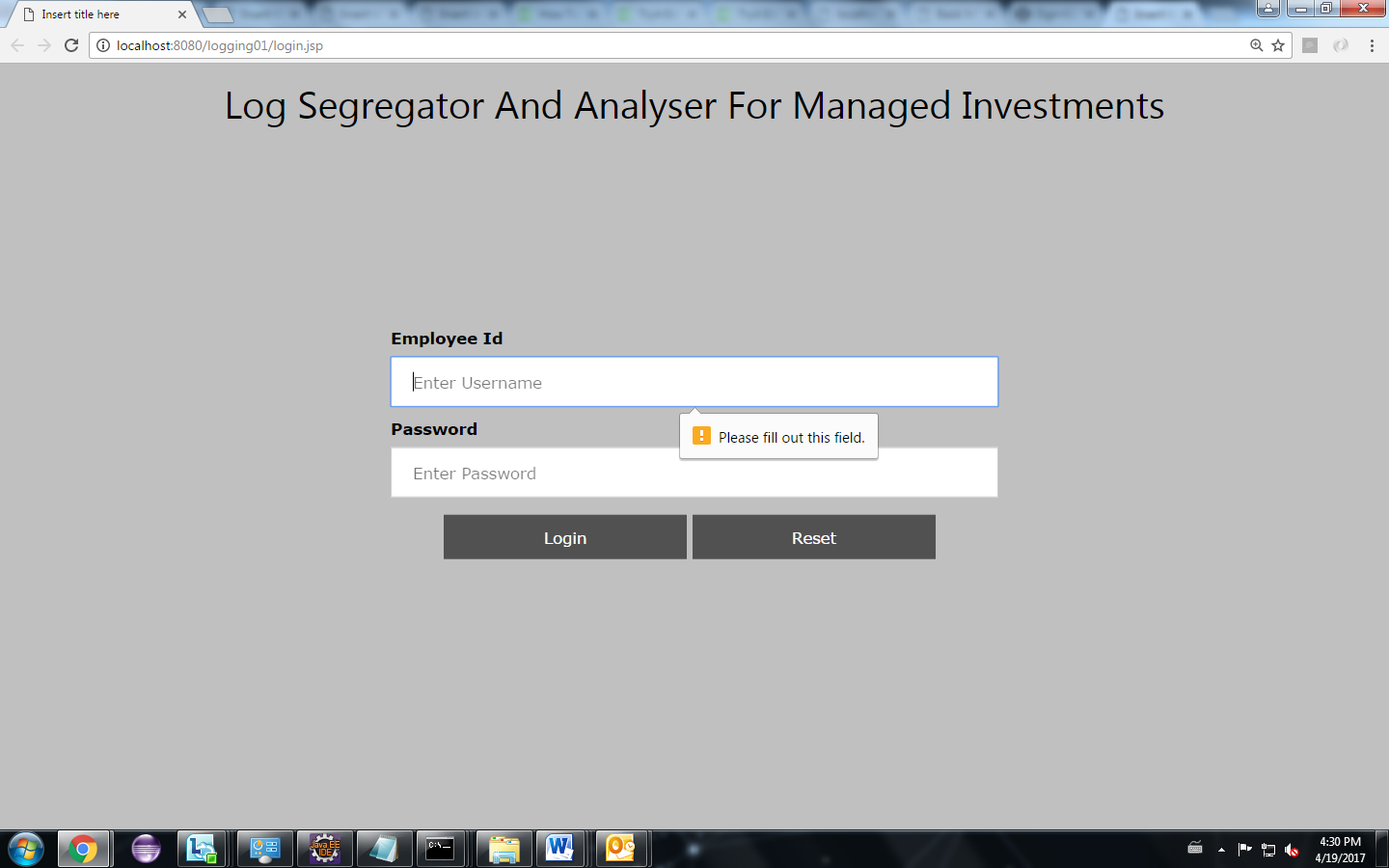
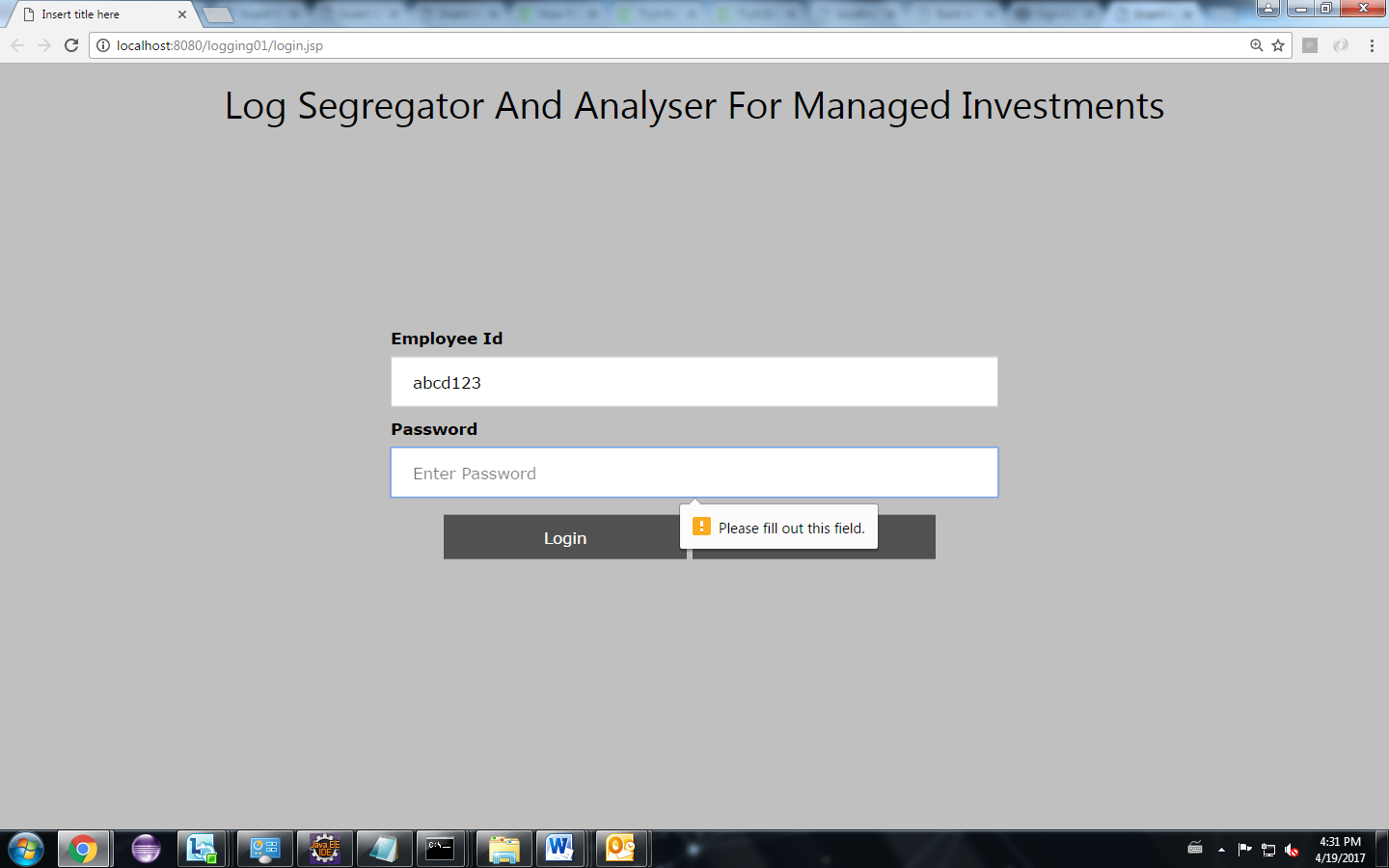
**

Fig 8.1 : Validation Testing**

* This is the login page in which the user enters their employee id and their password to login.
* These credentials are then matched with the database, if it matches the user will be able to login.
* If any of the fields are not typed in during submission of the form then it displays an alert message to the user.

**9. IMPLEMENTATION**

**9.1 Problems Faced:**

During development of any application, a developer would come across various issues which might be encountered in any of the stage of the software development life cycle. Initially there was a problem in reading the large data, which would be stored in multiple locations. Exporting as pdf posed a challenge initially.

Typically an application is developed by thousands of developers over a number of months, each developer would have different view about what to log and how it should be done and where it is to be done. The developers for same application may log error case as “ERROR”, “error”, or “err” and so forth, in this case, the search for error status should produce all the types of error logs.

**9.2 Lessons Learnt:**

Retrieving a huge data from various server, and their corresponding searching and filtering is learnt. Generation of log file and their subsequent storage mechanism are learnt in depth. The importance of logging data in server was inferred. The efficient usage of log file has been understood. Various severity levels associated with each logger logs different data into the log files. Detailed knowledge about these severity levels was gained.

**10. CONCLUSION & FUTURE SCOPE**

**10.1 Conclusion**

Thus a customized log segregator and analyser has been developed which meets the industrial needs. It enables the organization to configure new applications so as to analyse the log files to trace the bugs or errors if any. The facility of importing as pdf, csv provide ease of use and it can be saved for future references. The searching of log files by providing keywords, is done by pattern matching, the results thus extracted provides us valuable insights to the developer for the further improvement of the application as log files are the integral part of any application. Thus this system act as a google for the log files.

**10.2 Future Scope**

The system can be further enhanced by introducing the feature of spell check, this could play a vital role when the typed keyword does not match with any of the word in the log file. The user may type in a keyword in haste, which would lead to a misspelled word, in that case, the system can suggest words which are related to the keyword typed in. Thus a user can pick a word from the list, if the user misspelled it at the first time. This would save a lot of time and human effort. This feature can further be improvised by adding a statistical machine learning algorithm in which the system learns from the users’ previous searches.

**11.REFERENCES**

<http://www.vogella.com/tutorials/Logging/article.html>

<http://logging.apache.org/log4php/docs/loggers.html>

<http://www.tutorialspoint.com/bootstrap/>

<http://www.tutorialspoint.com/angularjs/>

<http://www.tutorialspoint.com/jsp/>

<https://docs.splunk.com/Documentation/Splunk/6.5.3/Overview/AboutSplunkEnterprise>

<http://www.tutorialspoint.com/log4j/>

<https://en.wikipedia.org/wiki/Boyer%E2%80%93Moore_string_search_algorithm>