CHAPTER- I

1. INTRODUCTION

Nowadays, managing data in an efficient manner has become an important aspect of any organization. If their data is not stored using proper mechanisms, then retrieving, managing and querying the data becomes burdensome. BH Global Marine India Pvt. Ltd. (BHI) is a multi-national company providing solutions to the Marine & Offshore, and Oil & Gas industries. This company maintains data regarding tender, sales, clients etc... To perform its operations. Right now, the company is storing and managing its data using document files and spreadsheets. When a client issues a tender or when BHI wants the customer details for delivering the products, then the employees of BHI manually search for the respective information from data present in their documents, emails etc.. There is no synchronization of data, which is shared among employees. Retrieval of data is a tedious and time taking process. Process of tender searching is not automated in existing system. There is no dynamic updating of data. Performing multiple operations becomes chaotic. The BHI website i.e. a web based application for customer information management and tender search overcomes all the problems mentioned above. It provides a business solution.

1.1 Existing System:

BH Global Marine India Pvt. Ltd. (BHI) is a multi-national company providing solutions to the Marine & Offshore, and Oil & Gas industries. The company has to maintain data about tender, sales, clients etc… for its operations. At present, the company is storing and maintaining its data using document files and spreadsheets. Whenever the client issues a tender or when BHI wants the customer details for delivering the products, then the employees of BHI search for the respective information from data present in their documents or they manually go through numerous email conversations that took place between BHI and their clients earlier. They sometimes write notes and they also execute commands in spreadsheets in order to get the required information. The data, which is shared among employees, is not in sync. . Most of the operations are not performed in an effective and efficient way. Also, there is no authentication or access control, hence the data is less secure and its management becomes chaotic. In order to search tenders, the employees have to manually check each website of shipyard companies, which is a time consuming process.

1.2 Disadvantages:

1) Data is not validated upon entry, hence there is a higher probability of presence of errors in data.
2) Retrieval of data is a tedious and time taking process.
3) Process of tender searching is not efficient.
4) Dynamic updating of data is not possible.
5) Performing different operations on data cannot be done effortlessly.

1.3 Proposed System:

According to the client's requirements, we have proposed to implement a client-server web based application for Customer Information Management and Tender Search. This is done by designing a web based application, which is implemented using web technologies (HTML/CSS) at client-side and a combination of a database (MySQL) and scripting language (PHP) at server-side. The data is stored on the server and it is accessed, updated and modified by the user through our web interface. This approach simplifies the work to be done by the employees of BHI when they deal with the data. For this purpose, we have also been commissioned to carry out the purchasing, configuration and installation of a local server on the company's premises. In this way, we are able to overcome the challenges of the existing system and provide much-needed security for their data via authentication.

1.4 Limitations:

1) The user should first get activated, only then he/she can use this web based application.
2) Firstly the company/contact person details should be added, only then the user can refer them in future.
3) The tender search performed, are limited to few websites.
4) This web based application is developed only for the BHI i.e. it can be used only by employees, admin and few people associated with BHI.

1.5 Scope:

The system here is a Web based Application for Customer Information

Management and Tender Search. This system will be designed to maximize scope for the employees of BH Global Marine India Pvt. Ltd (BHI) to perform various operations with ease. This is done by designing a web based application which performs efficient management of customer information and also automates the tender searching process. It has an effectual user interface. This web based application also provides access control and user authentication features. It is aimed to provide a well-organized and secure way of managing data which will optimize the operations performed by BHI.

1.6 Outline:

The proposed system i.e. a web based application for customer information management and tender search has multiple modules that allows BHI to perform various operations with ease. The modules are: user management, login, customer information management and tender search. Each of these modules consist of multiple operations. It provides an effectual data management. This web based application overcomes the problems of existing system. The proposed system saves the time and effort taken by BHI and it overcomes the problems of existing system.

CHAPTER- II

2. LITERATURE SURVEY

2.1 Back ground:

Business users like spreadsheets because they are simple to set up, easy to navigate and offer speed and freedom in completing a task. However this freedom is dangerous because it allows spreadsheets to be easily modified and errors can be built into them without much difficulty. It is not always possible to validate or control the contents of spreadsheets and historically quality control measures to validate spreadsheets have been insufficient.
These risks and losses highlight the importance of embracing new technologies. Today new BI tools offer many alternatives to these legacy systems. BI software removes many manual processes and creates automation by retrieving data directly from the source and displaying it for the business user. Those most familiar with the data are able to manipulate and analyze it directly through easy to use dashboards or interactive interfaces. Business users can focus in on the requirements driven by their roles, responsibilities, daily tasks, clients and internal process, and most importantly without trumping the needs of IT for managed risk and security, standardized processes, and operational efficiencies.

Automation and BI systems address the most important industry problems and challenges; minimizing enterprise and systemic risk, achieving economies of scale and creating consolidated views of data by removing silos. The entire life cycle of the data is accounted for, from the source to the end user and the ability to respond to market activity is improved. Financial services have become a network of systems and creating an effective strategy to manage these systems and the data within them is key to improving overall business performance and profitability in the long run.

Therefore to provide an efficient way for business organizations (here BHI) to perform various operations, we decided to develop a web based application. In computing, a web based application or web app is a client–server software application which the client (or user interface) runs in a web browser. Web based applications are popular due to the ubiquity of web browsers, and the convenience of using a web browser as a client to update and maintain web based applications without distributing and installing software on potentially thousands of

client computers is a key reason for their popularity, as is the inherent support for cross-platform compatibility. With web-based applications, users access the system via a uniform environment—the web browser. While the user interaction with the application needs to be thoroughly tested on different web browsers, the application itself needs only be developed for a single operating system. There's no need to develop and test it on all possible operating system versions and configurations. This makes development and troubleshooting much easier, and for web based applications that use a flash front end, testing and troubleshooting is even easier.

The user interface of web-based applications is easier to customize than it is in desktop applications. This makes it easier to update the look and feel of the application, or to customize the presentation of information to different user groups. Using internet technologies based on industry-wide standards, it's possible to achieve a far greater level of interoperability between applications than with isolated desktop systems. Increasing processor capacity also becomes a far simpler operation. If an application requires more power to perform tasks, only the server hardware needs to be upgraded. Any of three core technologies can be used for building web-based applications, depending on the requirements of the application.

2.2 Previous System Problems:

Business Financial and services firms are stuck in a tangled web of data and legacy systems. This model will not sustain today's business environment, which is largely defined by new regulations and the changes firms need to make to comply. The industry is also challenged with low trading volumes, shortened settlement cycles, and traders looking to new markets for revenue. Within this environment, firms are under pressure to cut operational costs, increase profitability, and better manage risk; however, effectively managing data often becomes a limiting factor in accomplishing these goals. Using spreadsheets as short term solutions to business problems continues to hamper long term performance.

Financial services firms need to be smart in their deployment of technology. Legacy systems and processes must be traded in for innovative business intelligence (BI) tools that allow firms to automate processes and implement effective data strategies.

Manual processes lead to operational risk:

Manual forms of data entry and manipulation pose inherent risk to any crucial enterprise function. Used at the core of almost all business processes –reporting, analysis, risk, and as the

basis of decision making–the financial services are heavily spreadsheet dependent. Due to the abundance of manual processes involved in creating, maintaining and updating them, spreadsheets are highly susceptible to operational risk and spreadsheet based calculations can result in massive errors with devastating consequences.

Spreadsheets are adjusted regularly to reflect market fluctuations, price adjustments and foreign exchange currencies or to take into account new strategic positions. They must also be manipulated in order to drill down on information for deeper analysis or in response to specific requests, such as price derivations. Spreadsheet changes, especially those with multiple formulas, are prone to many undetected errors that will have an overarching effect on the data. When data is entered manually, the link to the original data source is broken. With these types of changes, certain properties of the data will be lost, and unless specifically noted, they become hard to trace back to the original source. When additional calculations are added or records are deleted, quality control becomes further jeopardized.

Common spreadsheet changes and errors:

- Cell or name references changed to a static value. In this case cells are set as constants when they should be computed or derived
- When the parameters of a built-in function are changed to static values, 'what if' scenarios will be affected if a computed value is not used or referenced
- Failing to include certain spreadsheet cells in calculations will over or understate totals
- Risk comes from simple name changes to spreadsheets, especially when there are multiple users and all the users are not notified of the change
- Mechanical errors – The integrity of a spreadsheet and its data will be lost if improper sorting is applied or if formulas are overwritten
- Misinterpreting the situation to be modelled
- Logic errors – Choosing the wrong formula or creating the wrong function
- As spreadsheet complexity increases, so do the potential for error and enterprise risk.

Spreadsheets are real culprits for enterprise risk:

Spreadsheets are real culprits for enterprise risk and can create major financial losses for banks, as in the case of the London Whale. In 2012 a leading American bank lost approximately 2 billion dollars and faced over 900 million dollars in penalties due to errors in

a new trade strategy designed to hedge the firm's overall risk exposure. The chairman of the company blamed the trading strategy to be "flawed, complex, poorly reviewed, poorly executed and poorly monitored", however internal investigations into the cause also revealed the role of faulty spreadsheets. Spreadsheets are real culprits for enterprise risk.

The abundance of manual processes contained in the bank's spreadsheets left the bank exposed to operational risk and error. Specifically affected were those spreadsheets containing VAR computations, these computations were performed using a manual process that involved copying and pasting large amounts of data analytics into the spreadsheets. Spreadsheet based calculations were conducted with frequent formula and code changes and additional changes were made to a spreadsheet that inadvertently produced material calculation errors.

Manual forms of data entry and manipulation pose inherent risk and make it difficult to trace data back to the source; in turn certain properties of the source data are often irretrievable. Spreadsheet errors are common, yet in the absence of a thorough audit, they are often hard to spot. In the case of a leading American bank, insufficient spreadsheet controls, combined with a multitude of code changes and the lack of a vetting process, allowed faulty spreadsheets to slip through the cracks.

2.3 Approaches of a web based application:

A web based application can be developed in various ways. For e.g. a web based application is developed using servlets which has database connectivity using jdbc and runs on Tomcat web server. Here the BHI web based application is built using Php and MySQl is the database (database application), Apache is the web server application, HTML, CSS and JavaScript are front end technologies.

[1] Steven Holzner, McGraw Hill education, PHP: The complete Reference, edition 2008.

Summary: PHP development began in 1994 when Rasmus Lerdorf wrote several Common Gateway Interface (CGI) programs in C which he used to maintain his personal homepage. He extended them to work with web forms and to communicate with databases, and called this implementation "Personal Home Page/Forms Interpreter" or PHP/FI. PHP/FI could be used to build simple, dynamic web based applications. To accelerate bug reporting and improve the code, Lerdorf initially announced the release of PHP/FI as "Personal Home Page Tools (PHP

Tools) version 1.0" on the Usenet discussion group on June 8, 1995. This release already had the basic functionality that PHP has as of 2013. Early PHP was not intended to be a new programming language, and grew organically. A development team began to form and, after months of work and beta testing, officially released PHP/FI 2 in November 1997. Zeev Suraski and Andi Gutmans rewrote the parser in 1997 and formed the base of PHP 3, changing the language's name to the recursive acronym PHP: Hypertext Preprocessor. Afterwards, public testing of PHP 3 began, and the official launch came in June 1998. PHP is a server-side scripting language designed for web development but also used as a general-purpose programming language. Originally created by Rasmus Lerdorf in 1994 the PHP reference implementation is now produced by The PHP Group. PHP originally stood for Personal Home Page but it now stands for the recursive acronym PHP: Hypertext Preprocessor. PHP code may be embedded into HTML code, or it can be used in combination with various web template systems, web content management system and web frameworks. PHP code is usually processed by a PHP interpreter implemented as a module in the web server or as a Common Gateway Interface (CGI) executable. The web server combines the results of the interpreted and executed PHP code, which may be any type of data, including images, with the generated web page. PHP code may also be executed with a command-line interface (CLI) and can be used to implement standalone graphical applications. The standard PHP interpreter, powered by the Zend Engine, is free software released under the PHP License. PHP has been widely ported and can be deployed on most web servers on almost every operating system and platform, free of charge. PHP includes various free and open-source libraries in its source distribution, or uses them in resulting PHP binary builds. PHP is fundamentally an Internet-aware system with built-in modules for accessing File Transfer Protocol (FTP) servers and many database servers, including PostgreSQL, MySQL, Microsoft SQL Server and SQLite (which is an embedded database), LDAP servers, and others. Numerous functions familiar to C programmers, such as those in the stdio family, are available in standard PHP builds. There are two primary ways for adding support for PHP to a web server – as a native web server module, or as a CGI executable. PHP has a direct module interface called Server Application Programming Interface (SAPI), which is supported by many web servers including Apache HTTP Server, Microsoft IIS, Netscape etc… The LAMP architecture has become popular in the web industry as a way of deploying web based applications. PHP is commonly used as the P in this bundle alongside Linux, Apache and MySQL, although the P may also refer to Python, Perl, or some mix of the three. PHP can be deployed on most web servers, many operating

systems and platforms, and can be used with many relational database management systems (RDBMS). Most web hosting providers support PHP for use by their clients. It is available free of charge, and the PHP Group provides the complete source code for users to build, customize and extend for their own use.

[2] Michael McLaughlin, McGraw-Hill Education, Oracle Press, MySQL Workbench: Data Modeling & Development, April 30, 2013.

Summary: MySQL was created by a Swedish company, MySQL AB, founded by David Axmark, Allan Larsson and Michael "Monty" Widenius. The first version of MySQL appeared on 23 May 1995. It was initially created for personal usage from mSQL based on the low-level language ISAM, which the creators considered too slow and inflexible. They created a new SQL interface, while keeping the same API as mSQL. By keeping the API consistent with the mSQL system, many developers were able to use MySQL instead of the (proprietarily licensed) mSQL antecedent. MySQL is an open-source relational database management system (RDBMS);in July 2013, it was the world's second most widely used RDBMS, and the most widely used open-source client–server model RDBMS. It is named after co-founder Michael Widenius's daughter, My. The SQL acronym stands for Structured Query Language. The MySQL development project has made its source code available under the terms of the GNU General Public License, as well as under a variety of proprietary agreements. MySQL was owned and sponsored by a single for-profit firm, the Swedish company MySQL AB, now owned by Oracle Corporation. For proprietary use, several paid editions are available, and offer additional functionality. MySQL is a popular choice of database for use in web based applications, and is a central component of the widely used LAMP open-source web based application software stack (and other "AMP" stacks). LAMP is an acronym for "Linux, Apache, MySQL, Perl/ PHP/ Python". Free-software open-source projects that require a full-featured database management system often use MySQL. The applications that use MySQL database are: TYPO3, MODx, Joomla, WordPress, phpBB, MyBB, Drupal etc…. MySQL is also used in many high-profile, large-scale websites, including Google (though not for searches), Facebook, Twitter, Flickr and YouTube. Though MySQL began as a low-end alternative to more powerful proprietary databases, it has gradually evolved to support higher-scale needs as well. It is still most commonly used in small to medium scale single-server deployments, either as a component in a LAMP-based web based application or as a standalone

database server. Much of MySQL's appeal originates in its relative simplicity and ease of use, which is enabled by an ecosystem of open source tools such as phpMyAdmin. In the medium range, MySQL can be scaled by deploying it on more powerful hardware, such as a multi-processor server with gigabytes of memory. phpMyAdmin is a free and open source tool written in PHP intended to handle the administration of MySQL with the use of a web browser. It can perform various tasks such as creating, modifying or deleting databases, tables, fields or rows; executing SQL statements; or managing users and permissions. The software, which is available in 78 languages, is maintained by the phpMyAdmin Project.It can import data from CSV and SQL, and transform stored data into any format using a set of predefined functions.

[3] Ben Frain, responsive web design with HTML5 and CSS3, second edition, 24th Aug, 2015.

Summary: HyperText Markup Language, commonly referred to as HTML, is the standard markup language used to create web pages. Along with CSS, and JavaScript, HTML is a cornerstone technology, used by most websites to create visually engaging web pages, user interfaces for web based applications, and user interfaces for many mobile applications. Web browsers can read HTML files and render them into visible or audible web pages. HTML describes the structure of a website semantically along with cues for presentation, making it a markup language, rather than a programming language. HTML elements form the building blocks of all websites. HTML allows images and objects to be embedded and can be used to create interactive forms. It provides a means to create structured documents by denoting structural semantics for text such as headings, paragraphs, lists, links, quotes and other items. HTML can embed scripts written in languages such as JavaScript which affect the behavior of HTML web pages. Web browsers can also refer to Cascading Style Sheets (CSS) to define the look and layout of text and other material. The World Wide Web Consortium (W3C), maintainer of both the HTML and the CSS standards, has encouraged the use of CSS over explicit presentational HTML since 1997. CSS stands for Cascading Style Sheets. CSS describes how HTML elements are to be displayed on screen, paper, or in other media. CSS saves a lot of work. It can control the layout of multiple web pages all at once. External style sheets are stored in CSS files.

[4] Apache contributors, Apache HTTP Server 2.4 Reference Manual 1/3: Volume 1, pg. 100-250, Samurai Media Limited (16 November 2015)

Summary: The Apache HTTP Server Project is a collaborative software development effort aimed at creating a robust, commercial-grade, feature-rich and freely available source code

implementation of an HTTP (Web) server. The project is jointly managed by a group of volunteers located around the world, using the Internet and the Web to communicate, plan, and develop the server and its related documentation. This project is part of the Apache Software Foundation. In addition, hundreds of users have contributed ideas, code, and documentation to the project. The Apache HTTP Server, colloquially called Apache is the world's most used web server software. Most commonly used on a Unix like system (usually Linux), the software is available for a wide variety of operating systems besides Unix. Apache supports a variety of features, many implemented as compiled modules which extend the core functionality. These can range from server-side programming language support to authentication schemes. Some common language interfaces support Perl, Python, Tcl and PHP.

2.4Methodologies:

In this project we employed an object oriented approach to write the Php code. Creating classes and use them effectively. Simplifying database access by developing MySQL database and result set classes. Taking full advantage of advanced OOP features in PHP. Integrating PHP with AJAX applications. The following are few methodologies used in our web based application. (BHI web based application).

Self-service password reset (SSPR): It is defined as any process or technology that allows users who have either forgotten their password or triggered an intruder lockout to authenticate with an alternate factor, and repair their own problem, without calling the help desk. It is a common feature in identity management software and often bundled in the same software package as a password synchronization capability.

AJAX Live Search: Live search has many benefits compared to traditional searching:
Results are shown as you type. Results narrow as you continue typing. If results become too narrow, remove characters to see a broader result.

Index Terms:

- Web based application
- PHP- Hypertext Preprocessor
- HTML- Hypertext Markup Language
- Apache web server
- CSS – Cascading Style Sheet

CHAPTER- III

3. SYSTEM ANALYSIS

3.1 Introduction:

We perform system analysis in order to have a detailed understanding of the system i.e. a Web based Application for Customer Information Management and Tender Search. It will explain the purpose and features of the system, the interfaces of the system, what the system will do, the constraints under which it must operate. System analysis also helps the developers and the stakeholders to have a detailed understanding of the system.

3.2 Overall Description:

3.2.1 Product Perspective

BHI is a multi-national company providing solutions to the Marine & Offshore, and Oil & Gas industries. The company has to maintain data about tender, sales, clients etc. for its operations. At present, the company is storing and maintaining its data using document files and spreadsheets. In order to search tenders, the employees have to manually check each website of shipyard companies, which is a time consuming process. The existing system has no security, access control or efficient management of data. It does not employ any automated process. Hence, we proposed a solution to the above mentioned problems. We proposed to develop a web based application i.e. Web based Application for Customer Information Management and Tender Search. This will alleviate the problem of data management, tender search process, retrieving customer information, etc… It also provides user authentication and access control. It is developed using modern web technologies. It saves both time and effort taken by employees of BHI to perform different operations.

3.2.2 Product Functions

The system is intended to perform the following functions:

- It provides user authentication.
- It also provides access control.
- It allows the user to easily switch to different pages within the web based application.
- It performs the tender search process automatically.
- It performs tender search process on a daily basis using cron scheduler.
- It automatically updates the database if there are any new tenders.
- The user can also view the previous tenders.
- The users can also mark a tender as inactive.
- This system also provides password reset feature to its users.
- The employees can view the detailed information of various contacts/companies.
- The users can add, delete and edit the contacts/company details.

3.2.3 Operating environment

This product is a web-based application and will be hosted by a web server. This product can be viewed by any web browser supporting the latest web technology standards.

Hardware Requirements

The hardware requirements for this web based application are as follows:

1. RAM	:	2 GB
2. Dual Core processor	:	2 Ghz
3. Hard disk	:	500 GB

Software Requirements

The software requirements for this web based application are as follows:

Operating System	:	Ubuntu 14.04 LTS

Server	:	Apache 2.4
Backend technologies	:	PHP 5.5, Python 2.7
Database	:	MySQL 5.5
Frontend technologies	:	HTML 5, CSS 3, JavaScript

3.2.4 Design and Implementation Constraints:

The intention of the system is to be a web-accessible Customer Information System and Tender Search system. The customer data can be manually entered by an employee along with any relationships between the customers. However, Tenders are posted on several different websites. The system should automatically download the webpages and provide updates to all the users of the system.

This product is being developed for a small business organization with a limited budget, and therefore is constrained to low-cost methods of implementing the system. Since the system will be implemented using open source technologies the development team will be locked into using these technologies to do the majority of the implementation work. It comes with the added advantage of being updated and patched frequently by the open-source community. Open source technologies are entirely free to use and modify as the company sees fit. This provides future scope for flexibility of the system however, open source technologies are sometimes abandoned by the community when interest in such projects wanes.

We have selected PHP language to implement the business logic of the application. The team will be using Atom Editor and WAMP/LAMP development environments to edit and create the application. The team will be using MySQL as the DBMS for storing all the data of this application. Additionally, some web-scraping scripts will be written using Python programming language.

The customer's organization will be solely responsible for maintaining the software after final product has been delivered. Since the organization will not have a large budget for maintenance, strict and thorough documentation needs to be generated to provide a third party with a method of understanding and maintaining the internal aspects of the system.

3.2.5 Assumptions and Dependencies:

- It is assumed that the system will be developed using the PHP language.
- It is assumed that the system will be able to interface with an email server in order to send notification emails to users.
- It is assumed that the system will operating on Apache HTTP Web Server.
- It is assumed that the system will interface with a MySQL database.
- It is assumed that BH Global Marine India employees will have an email address for transmitting updated information.
- It is assumed that the browser utilized will support modern HTML5 and CSS 3.0 standards.
- It is assumed that the customer data will be manually inserted by an employee of the organization.

3.3 External Interface Requirements:

3.3.1 User Interfaces:

UI-1: The first interface is a login page, the user shall be able to input the username and password here. He shall be able to use the forgot password feature, when he/she forgets his/her password.

UI-2: After successful login, the user shall be able to see the dashboard (home) page. It consists of a menu with options like Dashboard, Users (only seen by admin and manager), Tenders, Customer Information System, Settings, Help and Logout. The user shall be able to choose any of those options by clicking on them.

UI-3: When a user clicks on the “Users” option, then the system navigates to “Users” page and the user shall be able to view and utilize the functionalities of “Users” page using the elements present in “Users” page.

UI-4: If a user clicks on the “Customer Information System” option, then the system navigates to “Customer Information System” page and the user shall be able to view and utilize the functionalities of “Customer Information System” page using the elements present in “Customer Information System” page.

UI-5: When a user clicks on the “Tenders” option, then the system navigates to “Tenders” page and the user shall be able to view and interact and utilize the functionalities of “Tenders” page using the elements present in “Tenders” page.

UI-6: If a user clicks on the “Settings” option, then the system navigates to “Settings” page and the user shall be able to view and utilize the functionalities of “Settings” page using the elements present in “Settings” page.

UI-7: When a user clicks on the “Help” option, then the user shall be able to view the help documentation.

UI-8: When a user clicks on “Logout” option, then the user shall be able to exit from the web based application.

3.3.2 Hardware Interfaces:

There is no interfacing with the hardware, other than the system must make use of a network connection.

3.3.3 Software Interfaces:

Along with a network connection, the system makes indirect use of a web browser which renders the html document with the help of Cascading Style Sheet and provides interactivity on client side through JavaScript. Apache uses CGI to interface with php and generate web pages dynamically. php communicates with the MySQL DBMS using MySQLi extension.

3.3.4 Communications Interfaces:

This web based application will be using the https protocol for communication over the internet and intranet.

3.4 System Features:

The following are the core features of the project which are implemented in order to have fully functioning application.

3.4.1 User Authentication:

3.4.1.1 Description and Priority:

In this web based application the user authentication is provided by assigning login credentials to users. The purpose of logging in is to let the application know who you are, so it can grant you the necessary privileges or restrict you from accessing certain things throughout your session. Only after a successful login i.e. after entering the correct username and password, the user can access the contents. Priority: HIGH

It is given high priority as it is an essential for a web based application to provide user authentication.

3.4.1.2 Stimulus/Response Sequences:

1. The user enters the username and password.
2. The system validates the entered data.
3. The system checks whether the user exists and that the password matches. If both conditions are met then a cookie is set on the user's browser, else appropriate error message is generated.

3.4.1.3 Functional Requirements:

REQ-1: The system shall allow the user to enter the username and password, it should provide a submit button in the login page.

REQ-2: The system shall view the relevant messages to user as a part of validation process.

REQ-3: The system shall check whether the user entered username and password are present in the database.

REQ-4: The system shall set a cookie, if the user is authenticated.

REQ-5: The system shall be able to view an error message to unauthenticated users.

3.4.2 Self-service password reset:

3.4.2.1 Description and Priority:

Self-service password reset (SSPR) is defined as any process or technology that allows users who have either forgotten their password to authenticate with an alternate factor, and repair their own problem, without calling the help desk. In our web based application we will use password notification email technique.

Priority: MEDIUM

It is a feature provided by a proficient web based application, but it is not a mandatory feature for a normal web based application. Therefore assigned to medium priority level.

3.4.2.2 Stimulus/Response Sequences:

1. The user enters the username.
2. The system validates and verifies the entered username.
3. If the user exists an email is sent to the associated email id.
4. The user click on the link provided in his password notification email.
5. The user enters the new password.
6. The system updates the user password in the database.

3.4.2.3 Functional Requirements:

REQ-1: The system shall allow the user to enter the username and password, it should provide a submit button in the login page.

REQ-2: The system shall be able to validate and verify the entered username.

REQ-3: The system shall be able to send a password notification email to verified user.

REQ-4: The system shall allow the user to enter the new password.

REQ-5: The system shall be able to update new in the database.

3.4.3 Customer Information System

3.4.3.1 Description and Priority:

It is one of the modules of our web based application. It allows the users to view, edit, delete, add and search the company/contacts details. A company can be assigned a Role and Type of the organization. A company can be related to another company with different types of relationships. A company can also have a relationship with a contact

person. A contact person has all the contact details of a person working at a company along with some other additional information.

Priority: HIGH

3.4.3.2 Stimulus/Response Sequences:

1. The user selects “Customers” from the menu.
2. The system displays the list of companies.
3. The user can add, update, delete or search a company.
4. The user clicks on the “add” option and fills in the required information and any relationships between that company and other entities.
5. The system updates the database with the entered data.
6. The user clicks on the “update” option and fills in the required information and any relationships between that company and other entities.
7. The system updates the database with the entered data.
8. The user click on the “delete” option.
9. The system deletes that company/customer and associated relationships from the database.
10. The user can search for any company.
11. The system provides a page with all the related companies and associated contact persons.

3.4.3.3 Functional Requirements:

REQ-1: The system shall allow the user to select “Customers” option present in the menu.

REQ-2: The system shall display the list of companies.

REQ-3: The system shall provide multiple operations to the users such as add, update, delete and search the details of contact/company.

REQ-4: The system shall allow the users to perform the above mentioned operations.

REQ-5: The system shall be able to update the database constantly based on various operations performed by the users on the details of contact/company.

3.4.4 Tender Search:

3.4.4.1 Description and Priority:

This is an essential module in our web based application, which will perform tender searching process automatically on daily basis.

Priority: High

This will save lot of time and work to be done by employees at BHI. Therefore it is given high priority.

3.4.4.2 Stimulus/Response Sequences:

1. The system displays the list of currently active tenders.
2. The user marks a tender as inactive.
3. The system will hide that tender from the current page.
4. The user clicks on “View all Tenders” option
5. System displays all tenders: current and inactive.
6. The user clicks on “Delete” option.
7. The system deletes the tender from the database.

3.4.4.3 Functional Requirements:

REQ-1: The system shall allow the user to view the contents present in the Tenders page.

REQ-2: The system shall allow the user to mark a tender as inactive.

REQ-3: The system shall be able to hide the inactive tenders from the current page.

REQ-4: The system shall be able to view all the tenders when the user clicks on “View all Tenders” option.

REQ-5: The system shall be able to delete the tenders when the user clicks on “Delete” option.

3.5 Other Nonfunctional Requirements:

3.5.1 Performance Requirements:

The performance of this web based application depends on the router used, the broadband connection and the number of requests made to the server. The number of requests that the server can handle at once is 10.

3.5.2 Safety Requirements:

There are no essential safety requirements. No safety requirements have been identified.

3.5.3 Security Requirements:

SEC-1 – The system shall only allow unauthorized users to access the login, and lost password page.
SEC-2 – The system shall require a login and password to gain access to the site.
SEC-3 – The system requires a valid email id for secure password reset and account activation.
SEC-4 – When the application is accessed from the internet, a Captcha is placed on the pages exposed to unauthorized users to thwart brute force attacks and hacking attempts.
SEC-5 – The system does not allow more than one email being sent every 15 minutes to prevent abuse of the activation and password reset systems.
SEC-6 – The system shall only accept connections through the HTTPS protocol. This prevents Man-in-the-middle attacks.
SEC-7 – The admin or authorized user of sufficient privilege can activate/deactivate another user account and control their permissions.
SEC-8 – The system stores passwords securely in hashed form and uses a random, unique salt to prevent rainbow table attacks.
SEC-9 – The systems utilizes cookies that expire every 24 hours. This provides a balance between convenience and security.

3.5.4 Software Quality Attributes:

- Flexibility: The system layout is in such a way that it can be accessed via both

mobile and tablets.

- Usability: This system can be used on-the-go by the users. It provides a high degree of usability.
- Reliability: The user can completely rely on this system for effective learning of the objective type questions.

3.6 System Architecture:

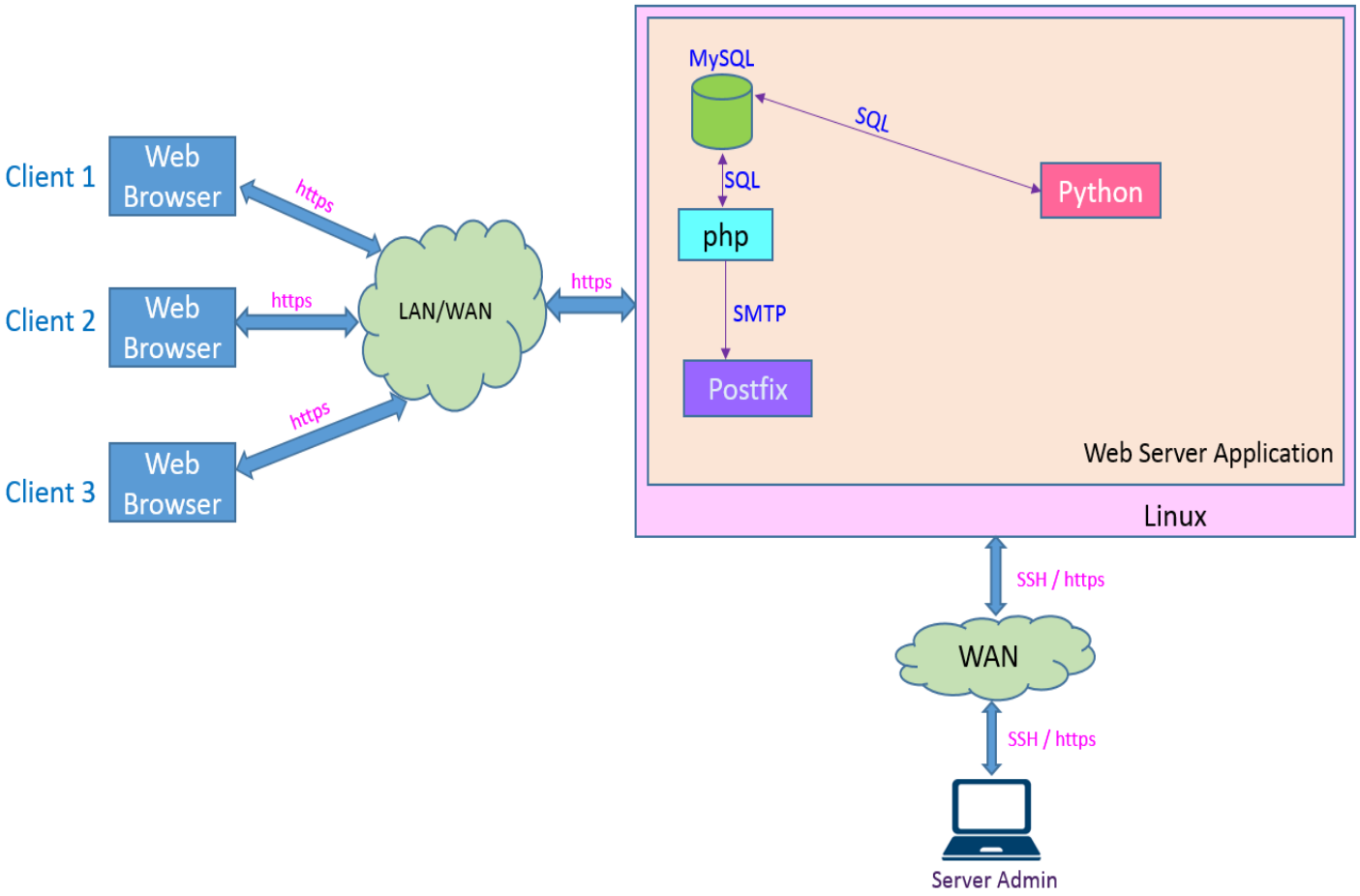


Fig 3.6 Block diagram for BHI web based application

The clients here are employees of BHI, they send request to sever and get response from server using HTTPS protocol and the connection can either be a LAN or WAN. The admin sends request to server and get response from server using HHTPS/SSH protocol. The web server application used in this project is Apche 2.4. MySQL is a database (application) that will intake/provide data from/to web based application using SQL queries. The web based application logic is majorly written in PHP. The tender search process makes use of Python scripts.

CHAPTER- IV

4. DESIGN

4.1 IMPORTANCE OF DESIGN:

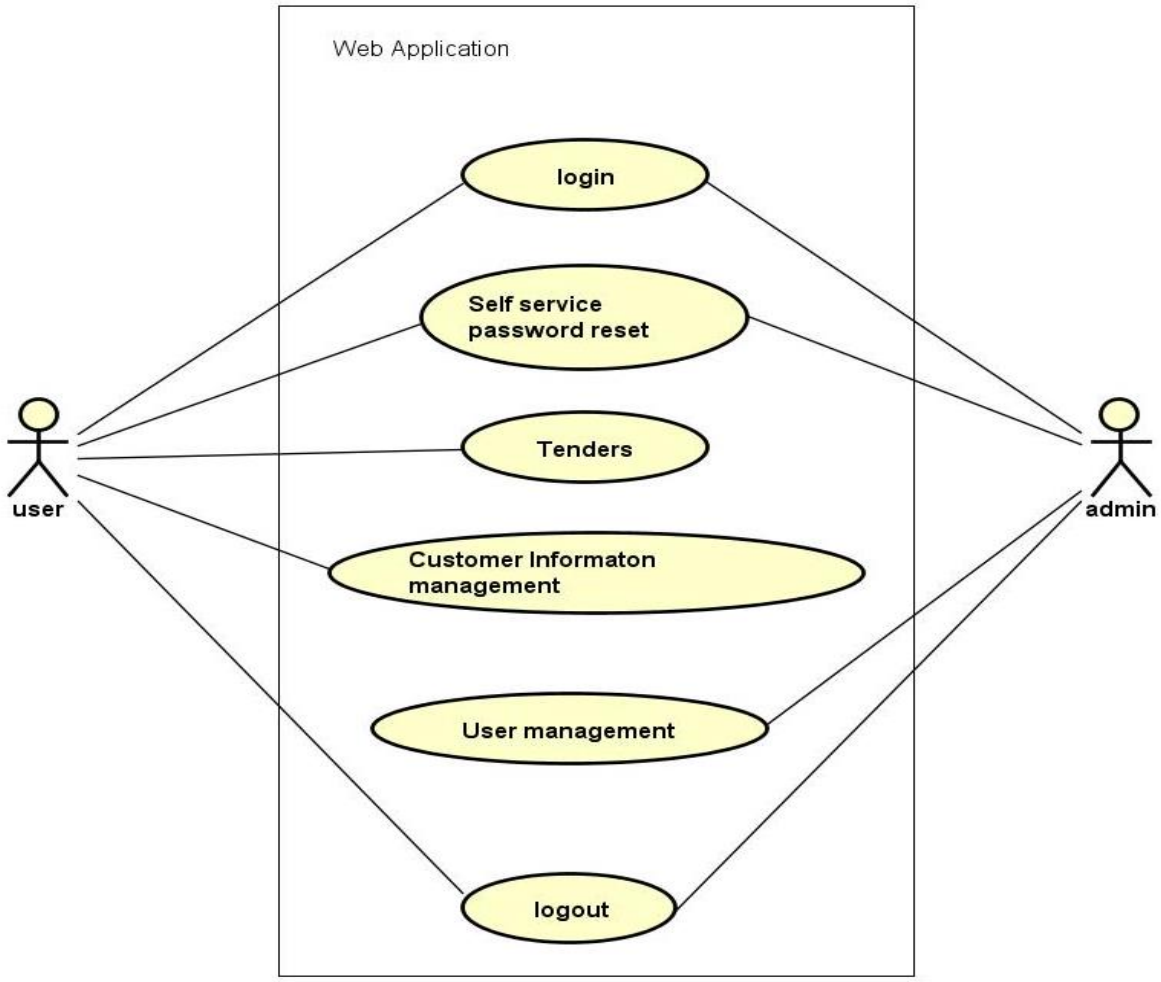
Software design is the process by which an agent creates a specification of a software artifact, intended to accomplish goals, using a set of primitive components and subject to constraints. Software design may refer to either "all the activities involved in conceptualizing, framing, implementing, commissioning, and ultimately modifying complex systems" or "the activity following requirements specification and before programming, as a stylized software engineering process. Software design usually involves problem solving and planning a software solution. This includes both low-level component and algorithm design and high-level, architecture design.

Software design is the process of implementing software solutions to one or more set of problems. One of the important parts of software design is the software requirements analysis (SRA). It is a part of the software development process that lists specifications used in software engineering. If the software is "semi-automated" or user cantered, software design may involve user experience design yielding a storyboard to help determine those specifications. If the software is completely automated (meaning no user or user interface), a software design may be as simple as a flow chart or text describing a planned sequence of events. Language and Fundamental modeling concepts. In either case, some documentation of the plan is usually the product of the design.

Furthermore, a software design may be platform-independent or platform-specific, depending upon the availability of the technology used for the design. In terms of the flexibility, one of the main objectives of this phase is that it is intended to design such a system which can be dynamic in nature and responsive to the changes if required. Another important objective is that the phase of system designing is concerned with creating the system which can work very effectively and efficiently providing the required output and being responsive to the time within a given time limit. The aspect of reliability and physical security of data cannot be ignored. With this respect, the system designing phase ensures security measures of the system effectively and efficiently.

4.2 UML DIAGRAMS:

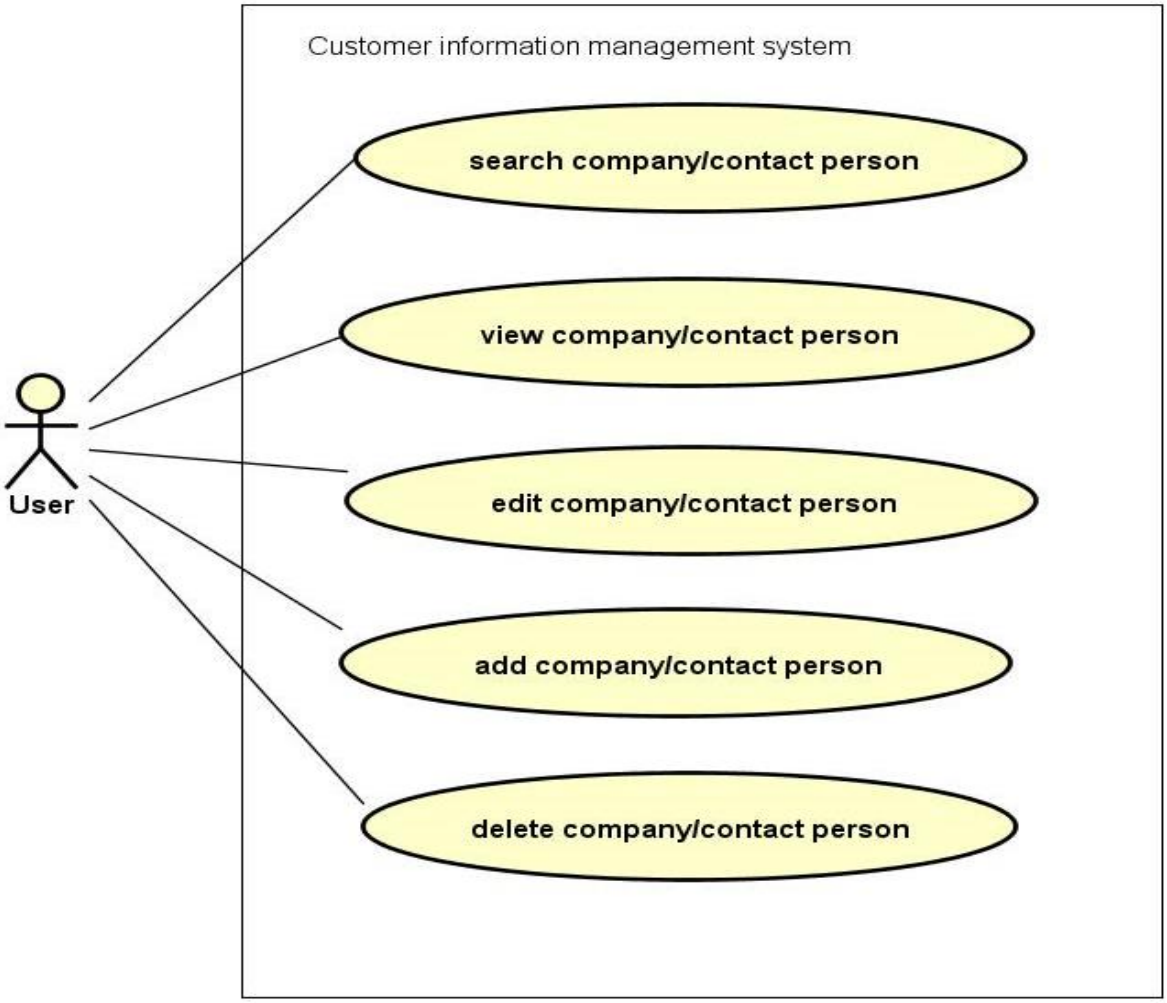
4.2.1 USE CASE DIAGRAM:



4.2.1.1 Use case diagram of a web based application for BHI

Description

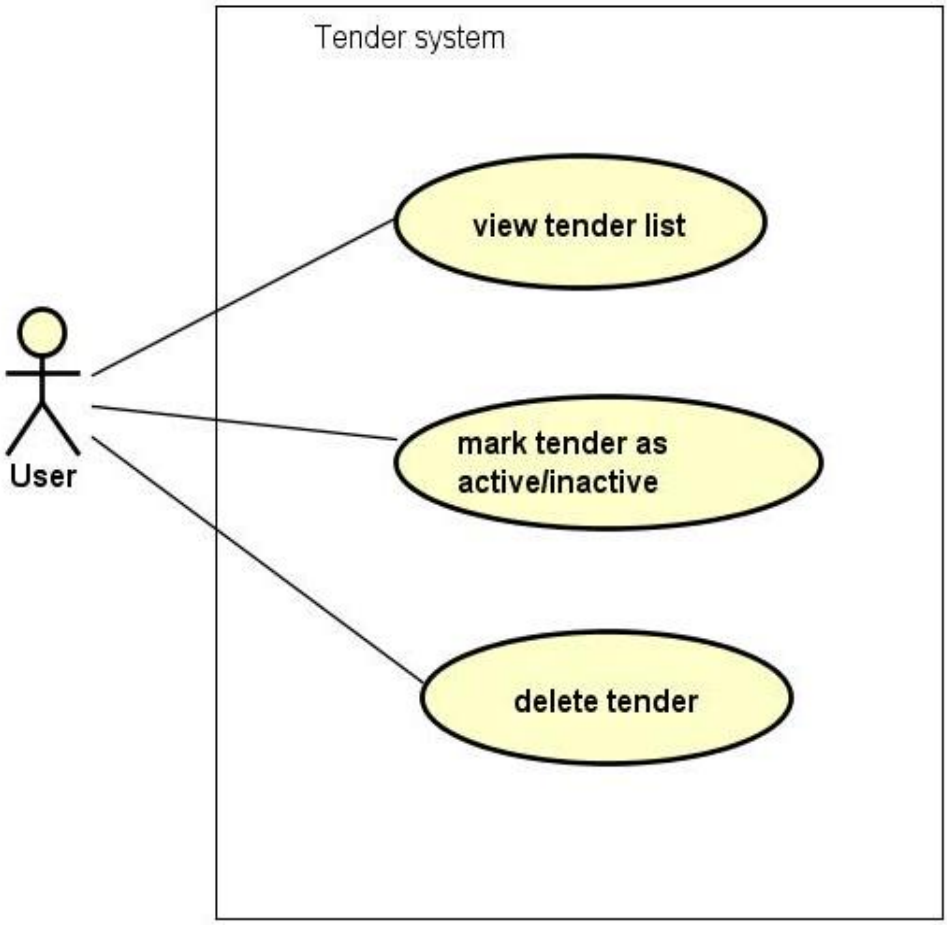
In the above use case diagram, the primary actor is user and the secondary actor is admin. The system here is a web based application. The user is associated with login, logout, SSPR, tenders and customer information management use cases. The admin is associated with login, logout, SSPR and user management use cases.



4.2.1.2 Use case diagram of customer information management

Description

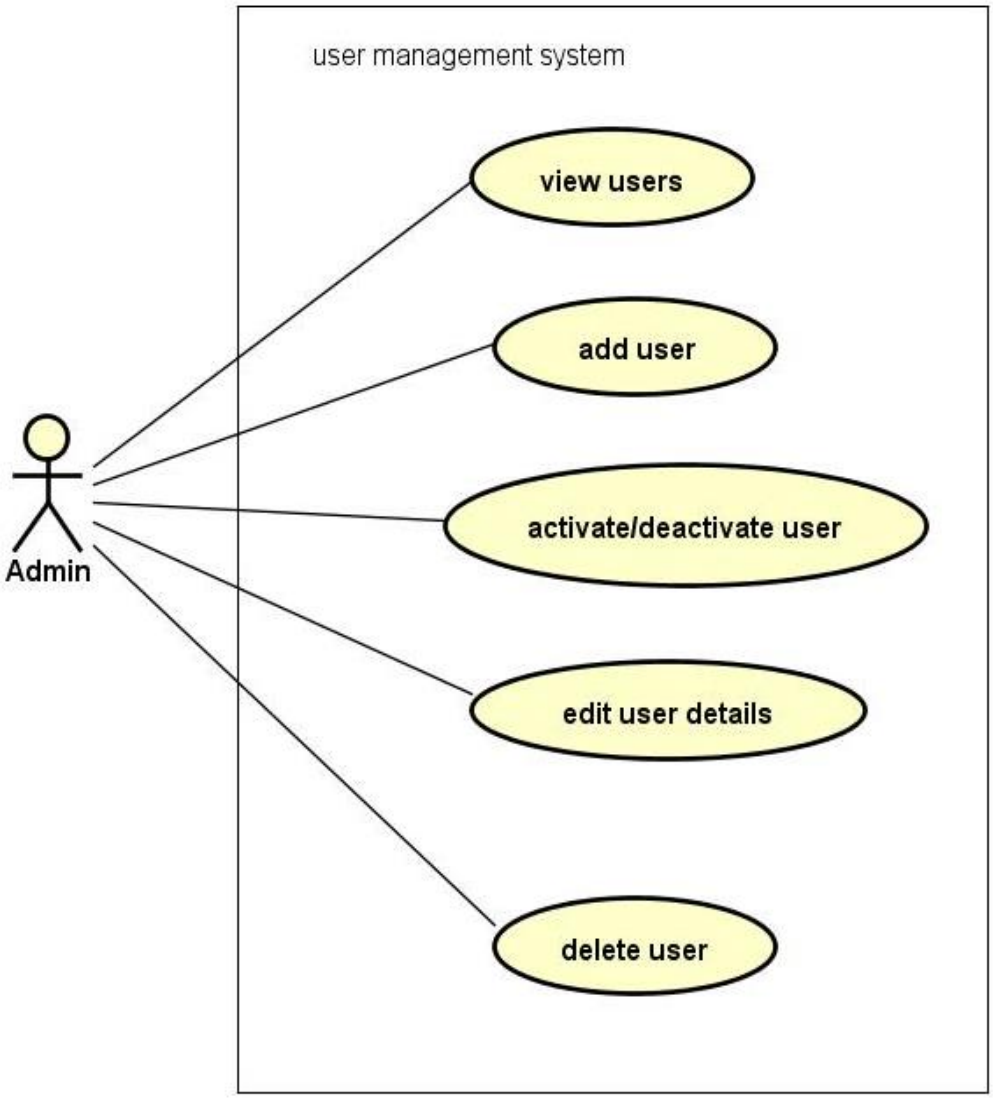
In the above use case diagram the actor is user and the system is customer information management system. The user is associated with add company/contact person, search company/contact person, view company/contact person, edit company/contact person and delete company/contact person use cases.



4.2.1.3 Use case diagram of tender system

Description

In the above use case diagram the actor is a user and the system is tender system. The user is associated with view tender list, mark tender as active/inactive and delete tender use cases.

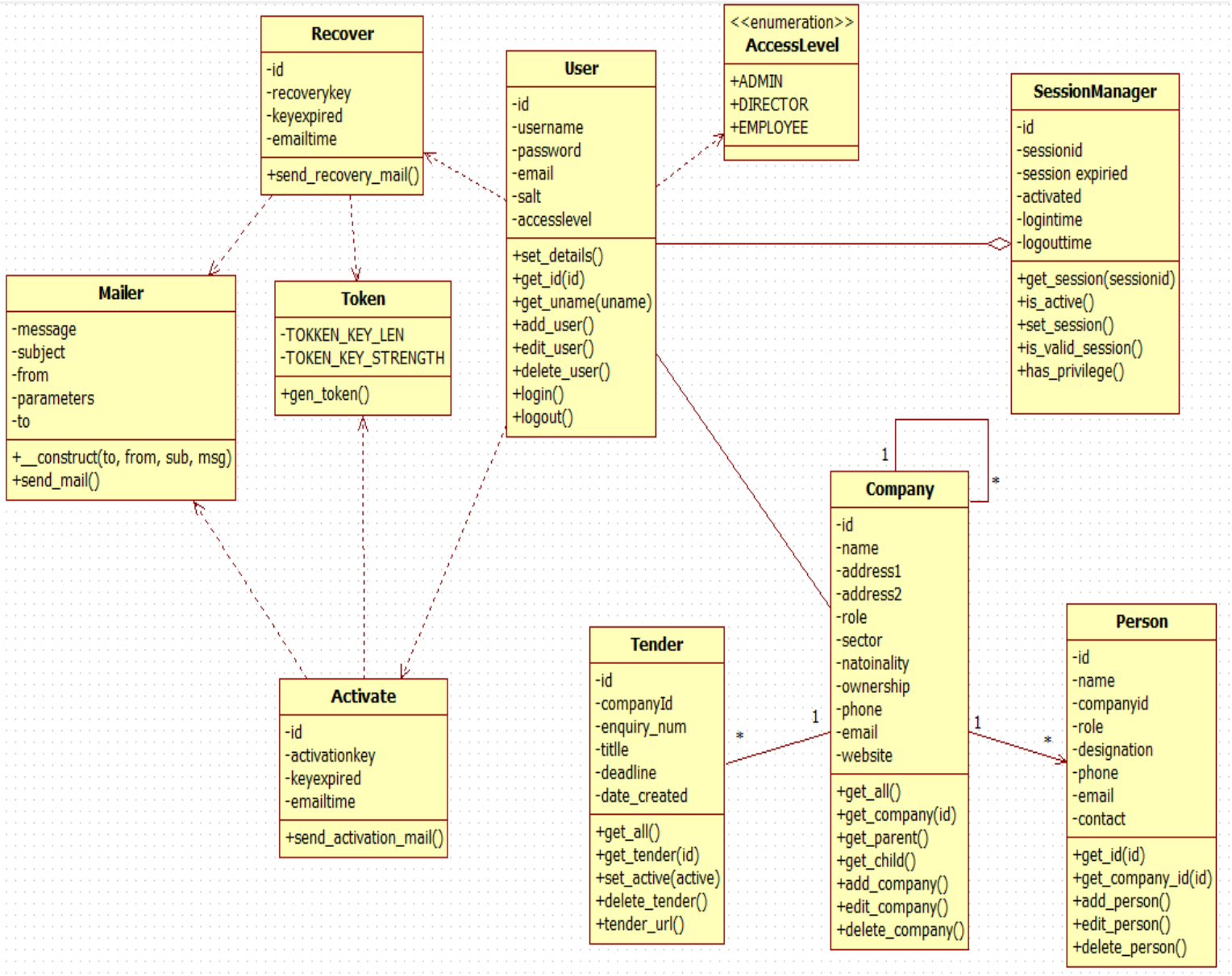


4.2.1.4 Use case diagram of user management

Description

In the above use case diagram the actor is admin and the system is user management system. The admin is associated with view users, add user, activate/deactivate user, edit user and delete user use cases.

4.2.2 CLASS DIAGRAM:

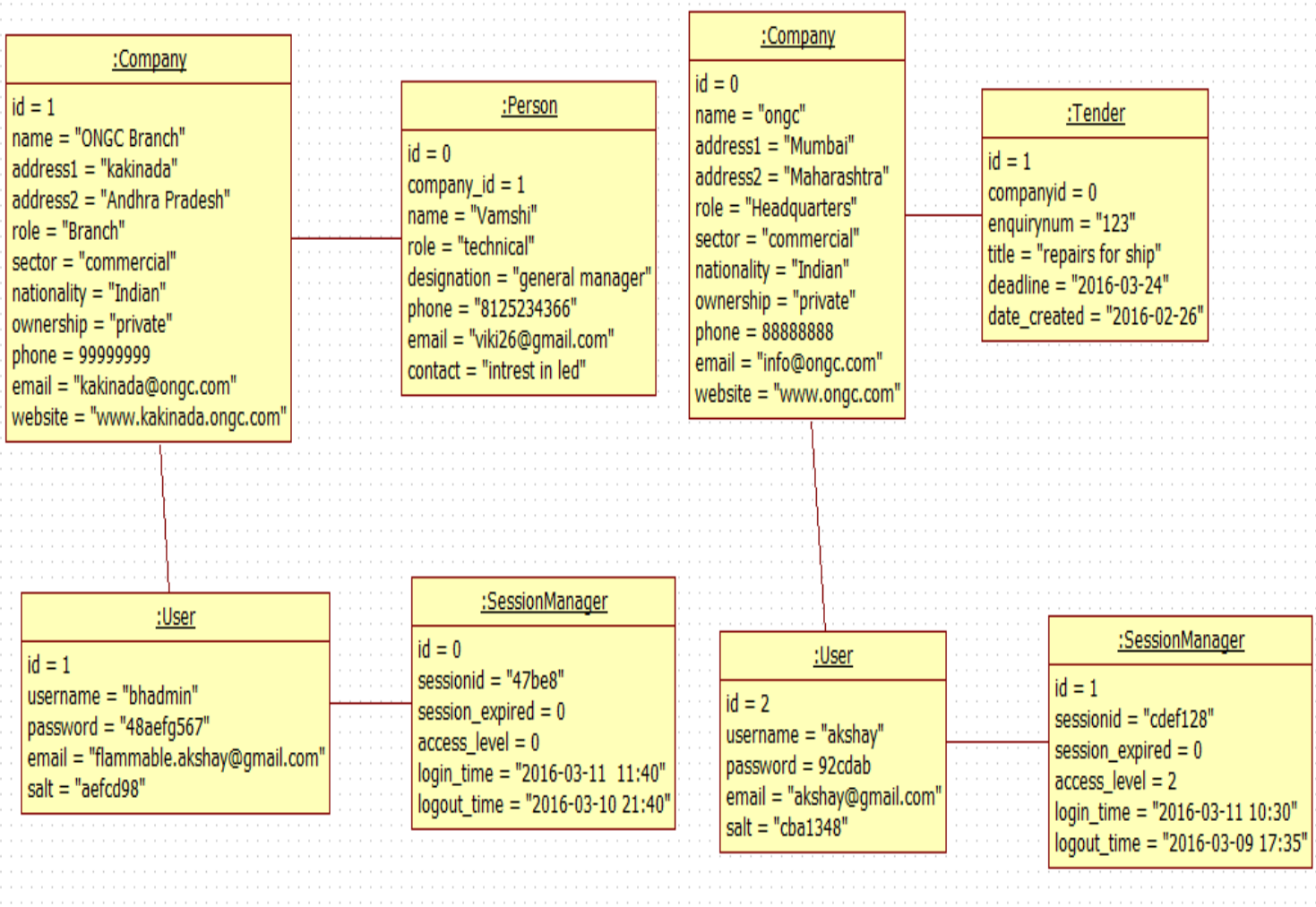


4.2.2 Class diagram of web based application for BHI

Description

The User class uses Recovery class, AccessLevel class and Activate class. It is also associated with Company class. The SessionManager has a User. The Company class is associated to itself and also associated with Tender class and Person class. Recovery class and Activate class uses Token class and Mailer class.

4.2.3 OBJECT DIAGRAM:

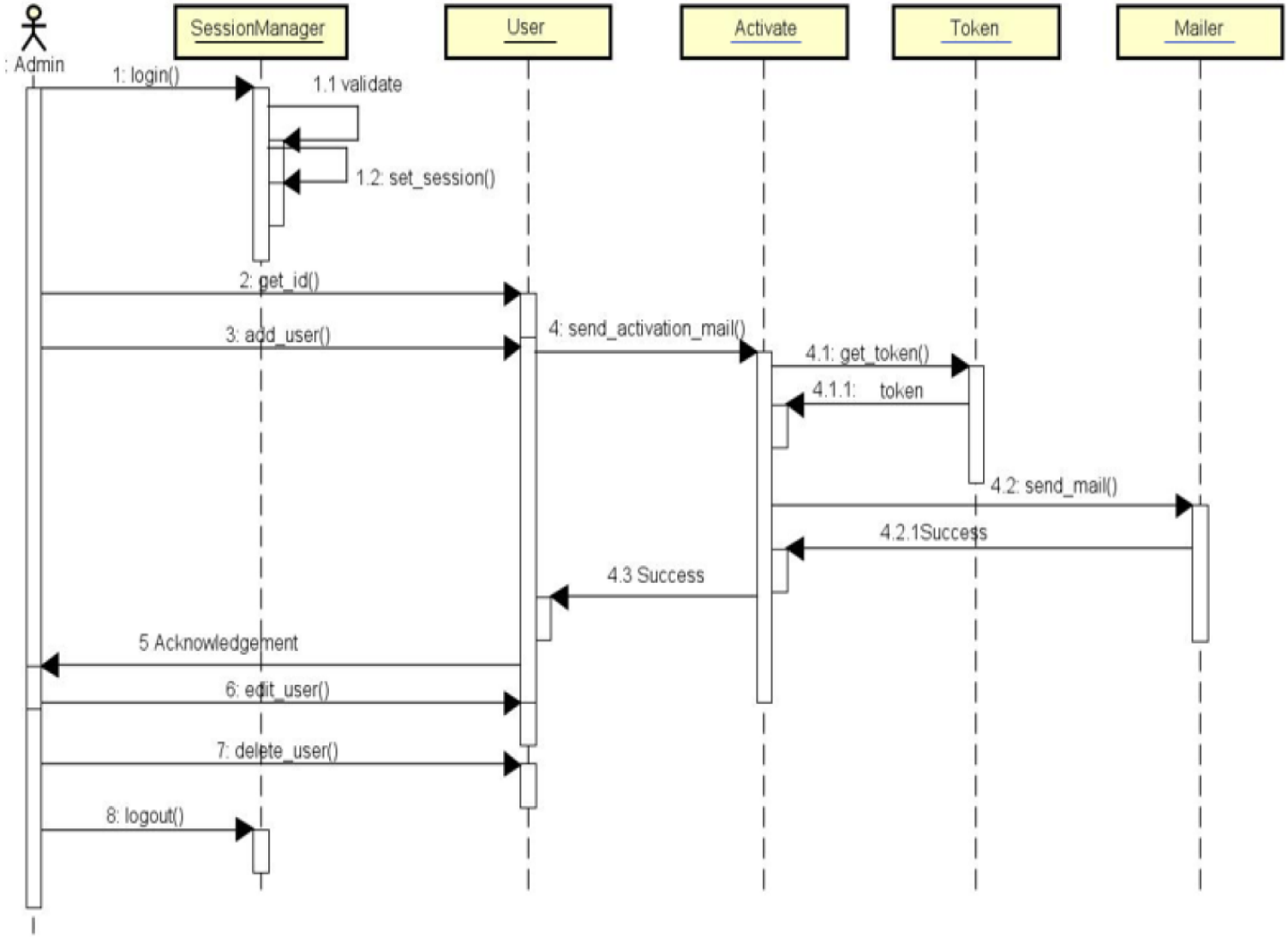


4.2.3 Object diagram of web based application for BHI

Description

The above diagram illustrates the static view of the system at a particular moment. Company, Person, User, Tender and SessionManager are objects. The various attributes of each object and the links between those objects are shown clearly in the above diagram.

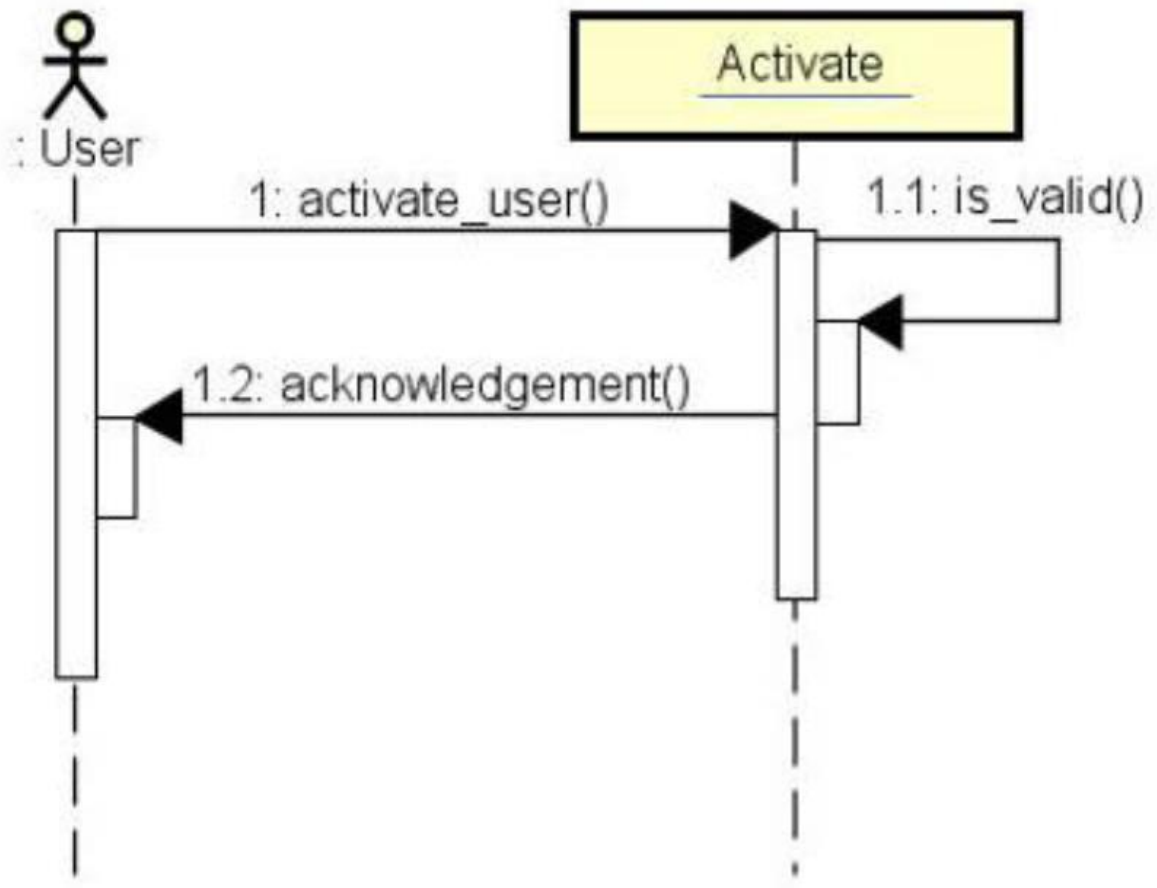
4.2.4 SEQUENCE DIAGRAM:



4.2.4.1 Sequence diagram of user management

Description

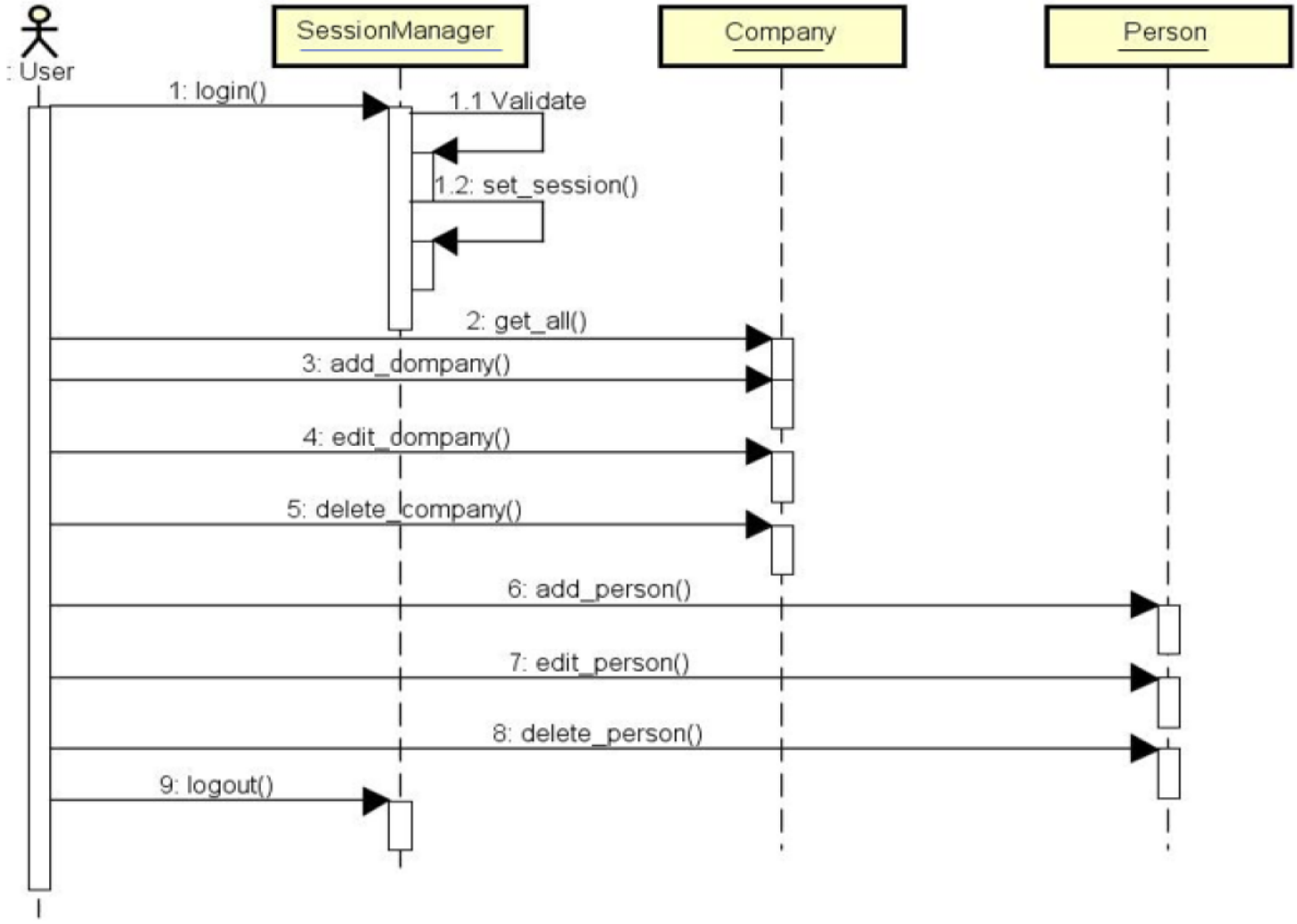
The admin will login and then SessionManager will call validate( ), once validation is done, it will call set_session( ) . Now the admin can perform required operations by calling various methods as shown above. The interaction between the objects of User class, SessionManager class, Activate class, Mailer class and Token class is shown clearly in the above diagram. Once the admin is done performing the required operations, he/she will logout from the system.



4.2.4.2 Sequence diagram of user activation

Description

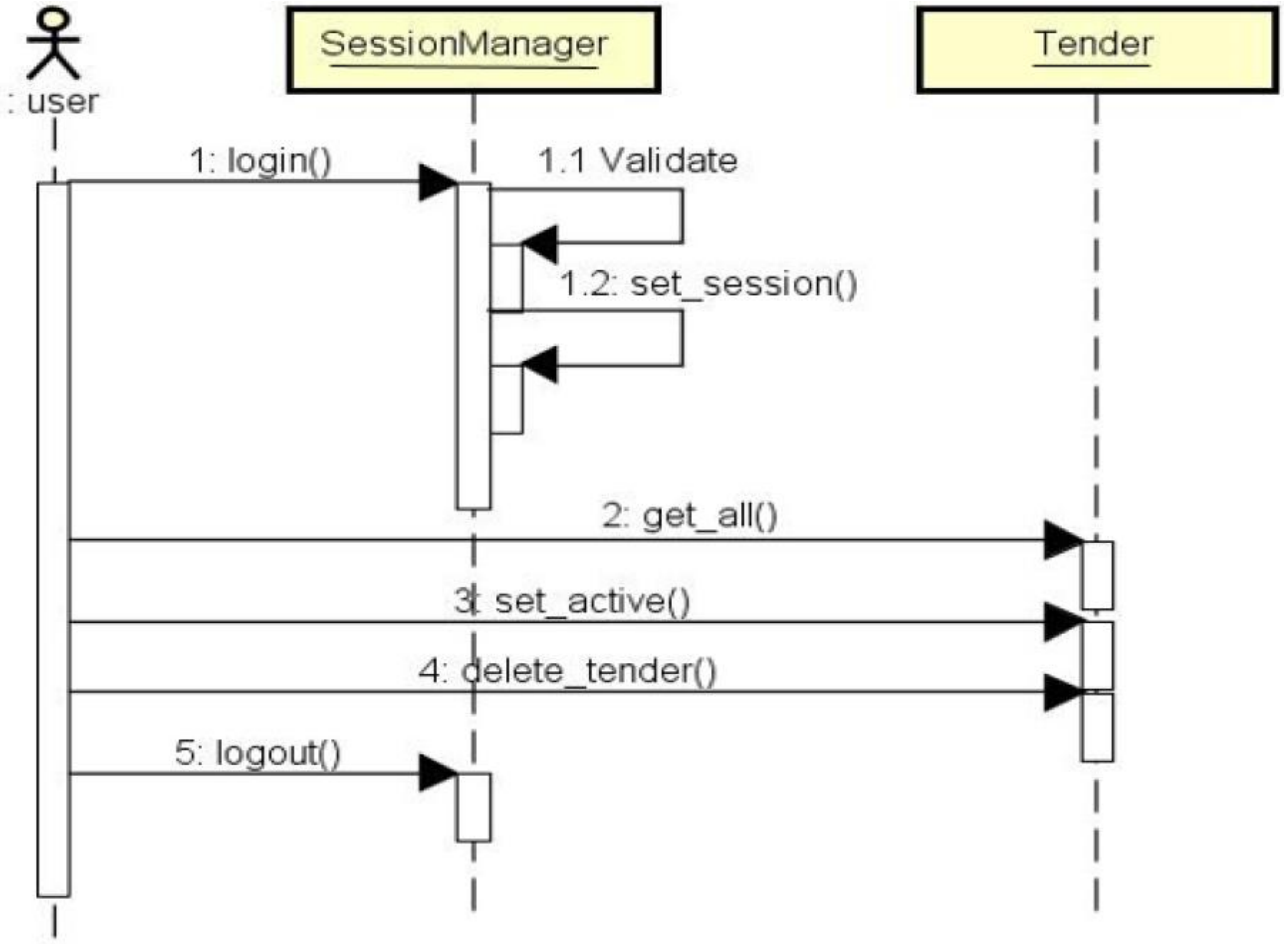
In the above diagram, user is an actor. The user receives an activation mail and once the user clicks on the link provided in the mail, the activate_user( ) is called. The object of Activate class calls is_valid( ) and later sends an acknowledgement to user.



4.2.4.3 Sequence diagram of customer information management

Description

The user will login and then SessionManager will call validate( ), once validation is done ait will call set_session( ). Now the user can perform required operations by calling various methods as shown above. The interaction of User with the objects of SessionManager class, Activate class and Company class is shown clearly in the above diagram. Once the user is done performing the required operations, he/she will logout from the system.

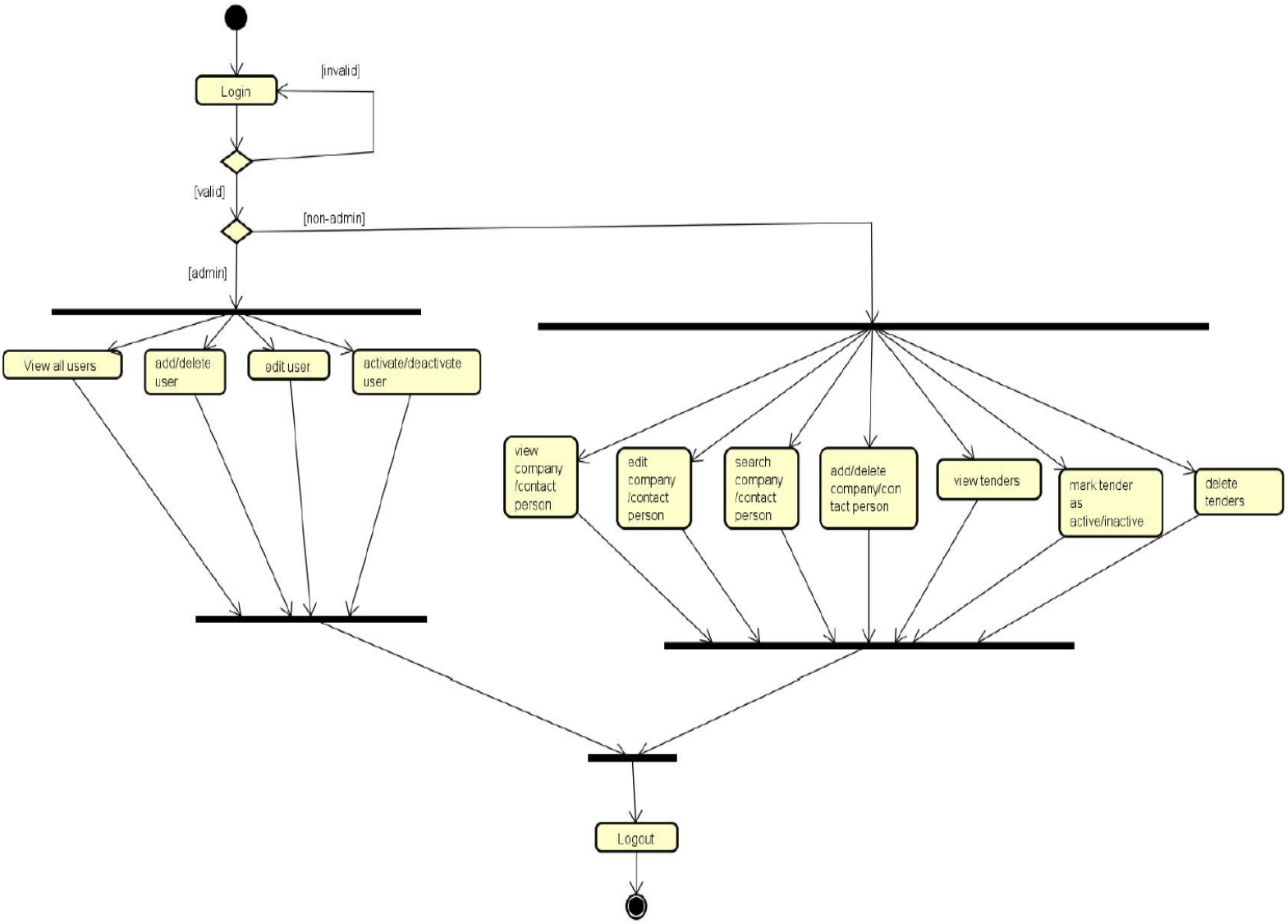


4.2.4.4 Sequence diagram of tender

Description

The actor in the above diagram is user. User will login and then SessionManager will call validate( ), once validation is done ait will call set_session( ). Now the user can perform required operations by calling various methods as shown above. The interaction of user with the objects of SessionManager class and Tender class is shown clearly in the above diagram. Once the user is done performing the required operations, he/she will logout from the system.

4.2.5 ACTIVITY DIAGRAM:

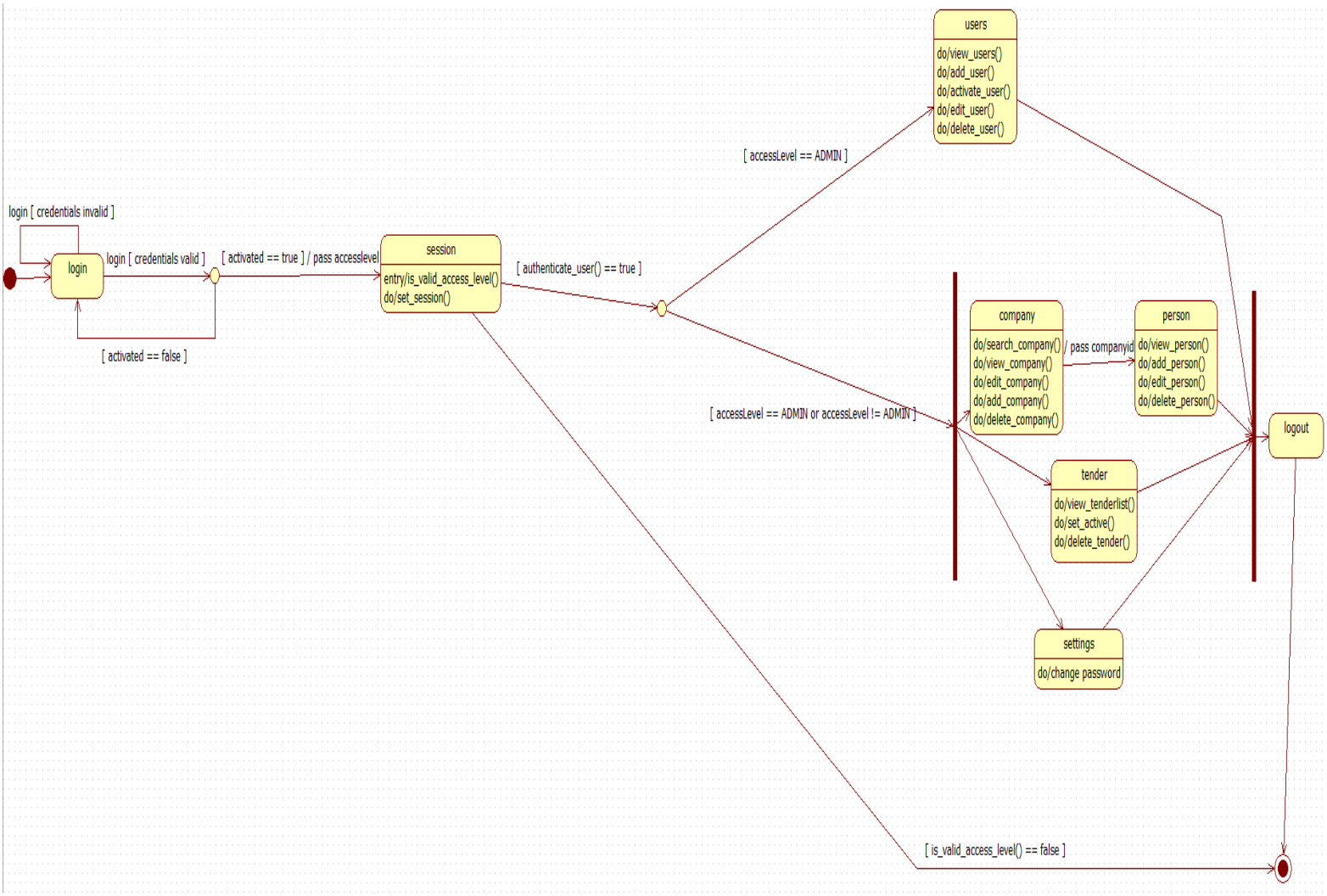


4.2.5 Activity diagram of a web based application for BHI

Description

The flow from one activity to other activity in the system is shown in the above diagram. The person will login into the system and the system checks if the person is a valid or invalid user. If valid it checks whether the user is an admin or non-admin, else it navigates to login page. If the person is admin then he can perform various operations in user management module. If the person is non admin then he can perform various operations in tender module and customer information management module. When the admin and non-admin are done performing their required operations, they will logout from the system.

4.2.6 STATECHART DIAGRAM:

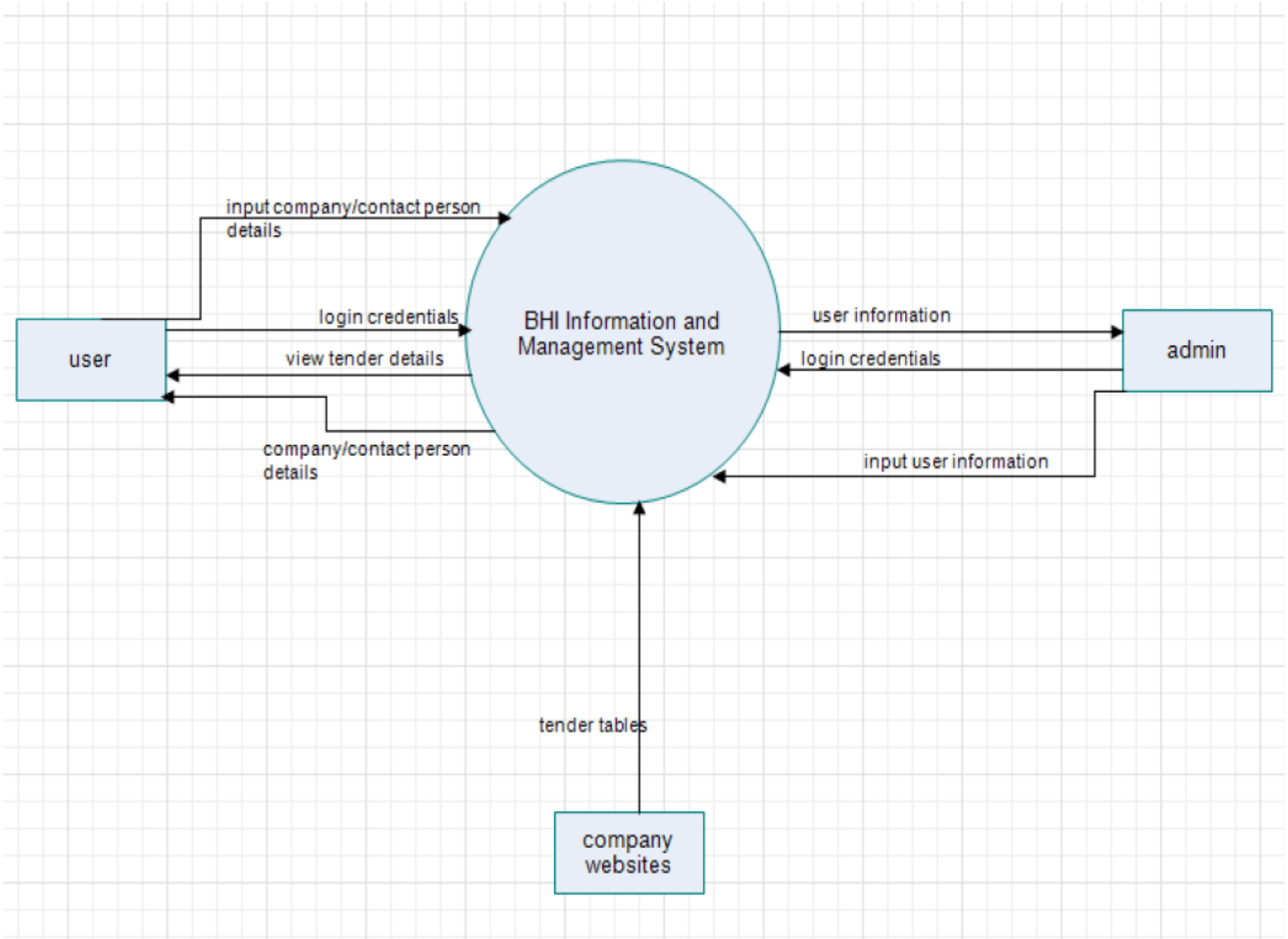


4.2.6 Statechart diagram of a web based application for BHI

Description

The states and transitions obtained from initial to final state of the object is depicted in the above diagram. Login and logout are states. Session, company, person, tender, users and settings are internal transitions. The various guard conditions that enables transitions are shown in the above diagram.

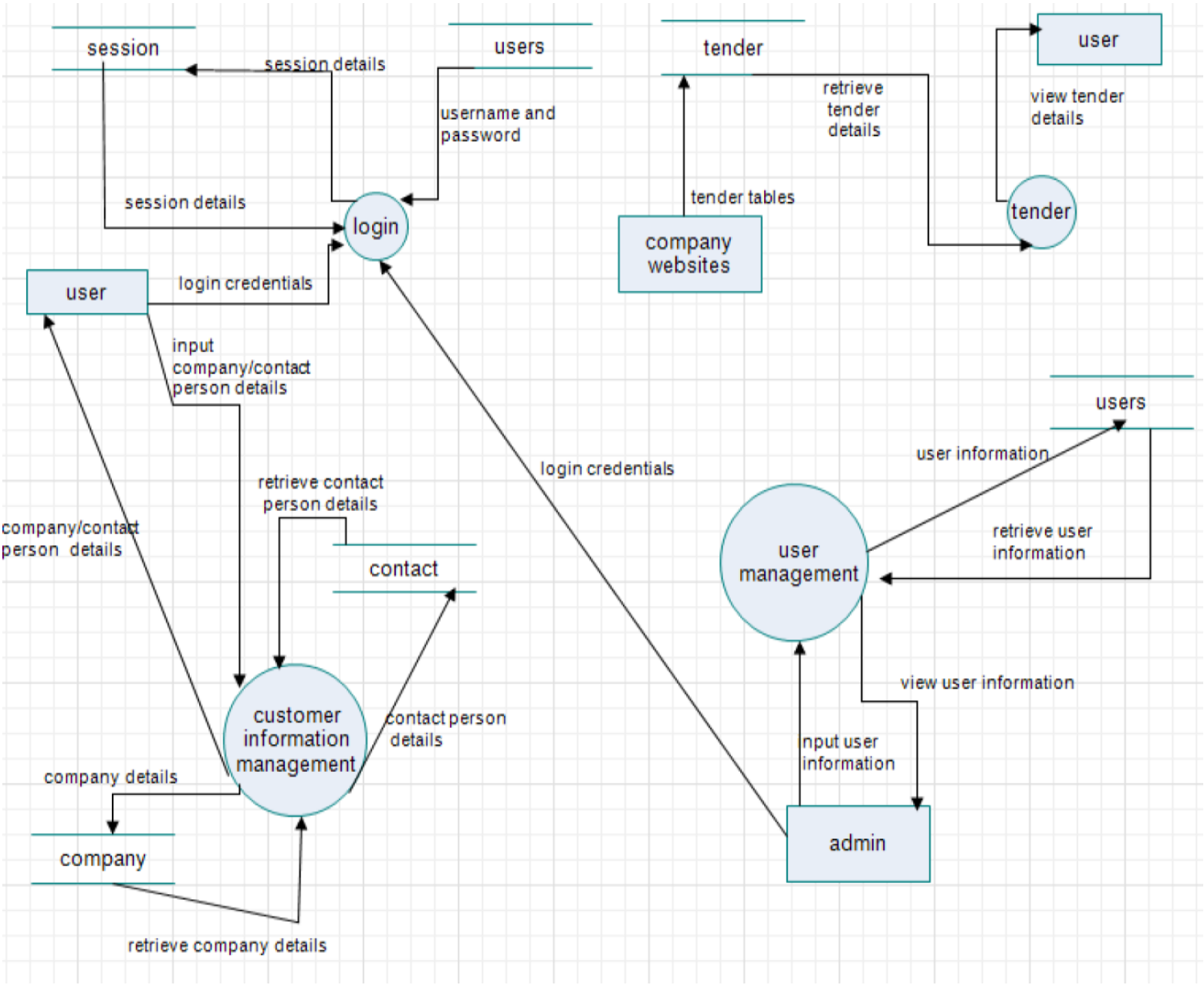
4.2.7 DATAFLOW DIAGRAM:



4.2.7.1 Dataflow Level 0 Diagram of a web based application for BHI

Description

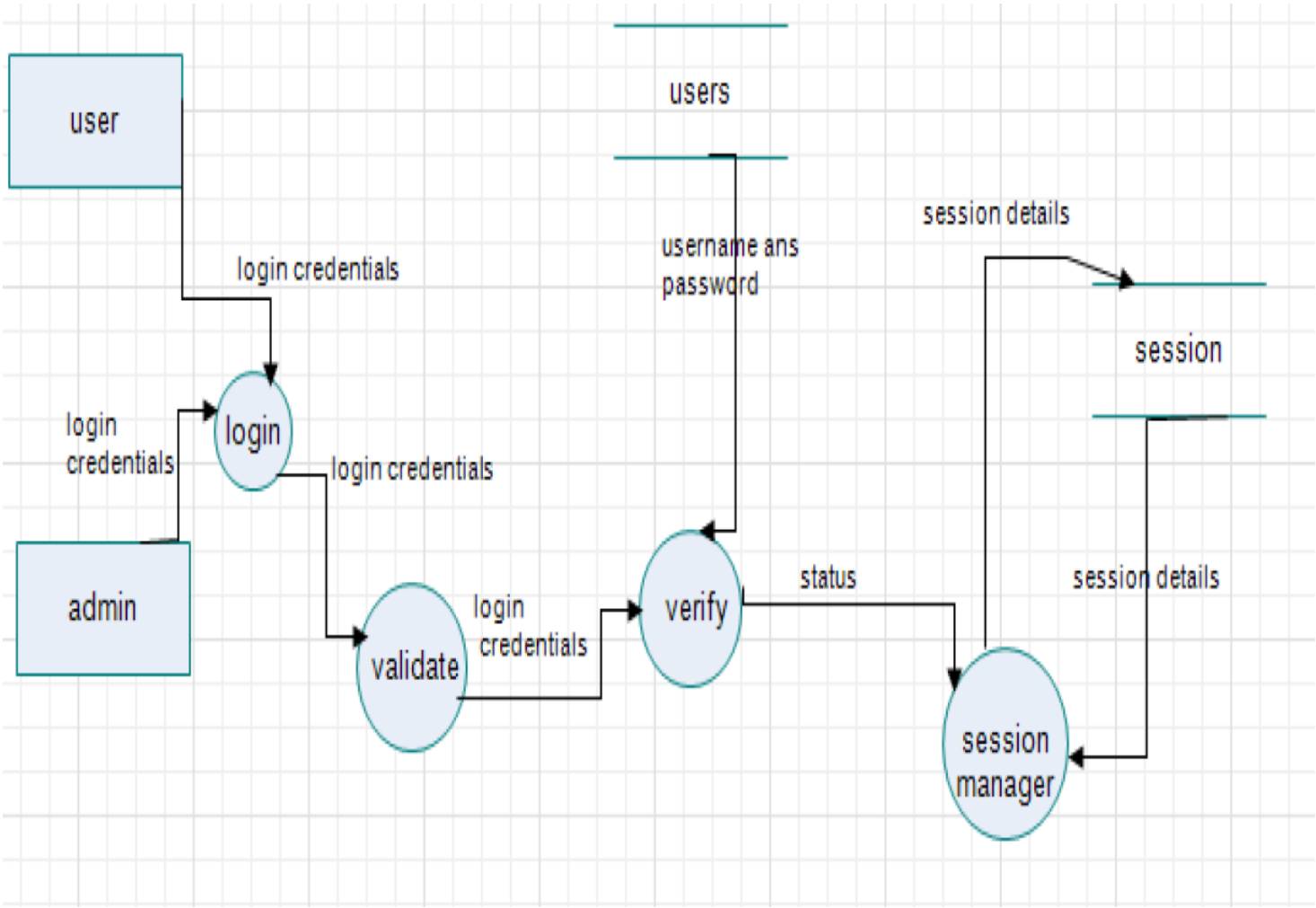
Dataflow Level 0 Diagram is known as context diagram. In this diagram, the entire system is shown as single process. User, admin and company websites are external entities. The data flow between the process and external entities are depicted in the above diagram.



4.2.7.2 Dataflow Level 1 Diagram of a web based application for BHI

Description

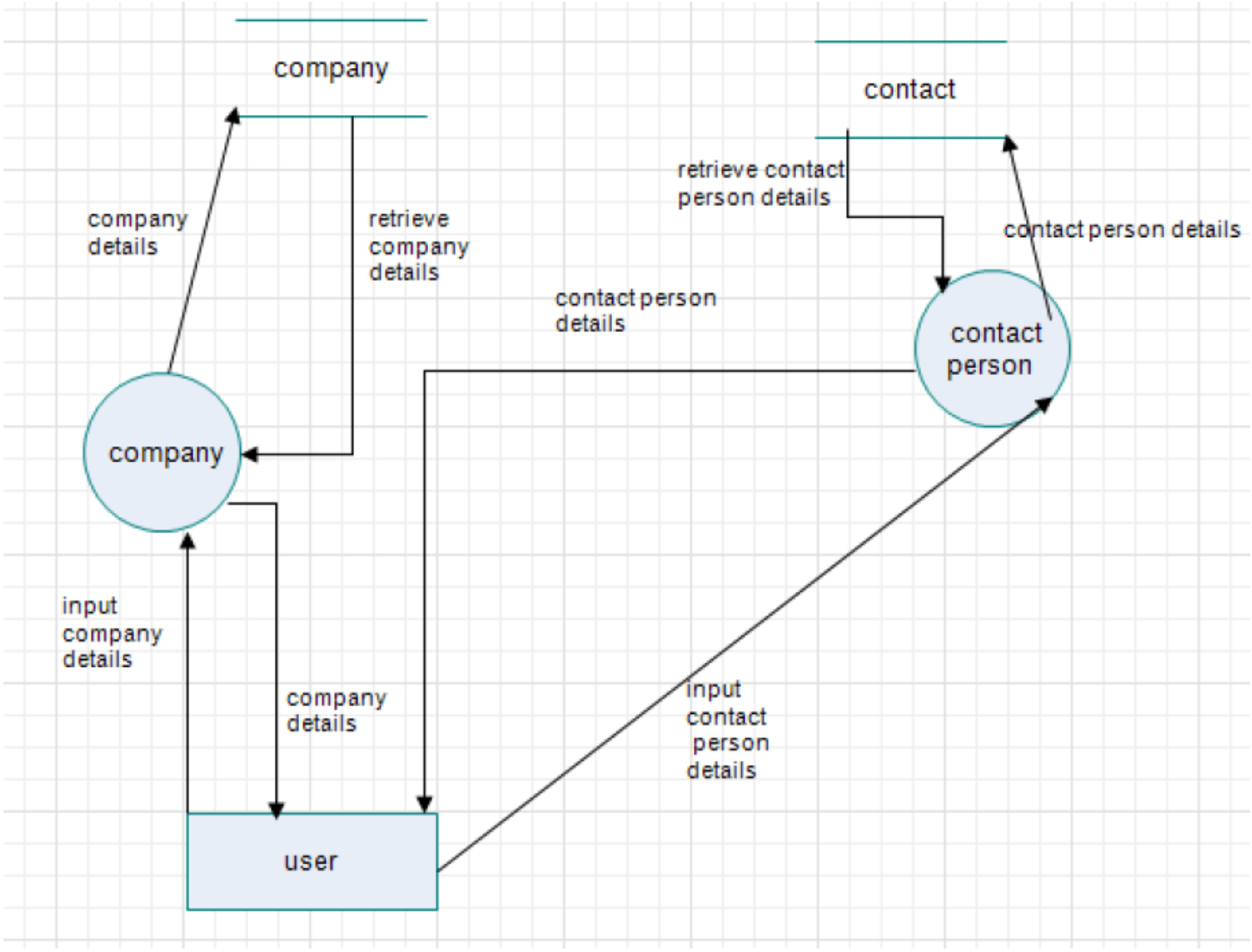
The single process in DFD level 0 diagram is decomposed into multiple processes in DFD level 1 diagram. Session, admin, users, company, contact and tender are data stores. Though there are multiple processes and data stores in DFD level 1 diagram, the data flow is balanced.



4.2.7.3 Dataflow Level 2 Diagram of a web based application for BHI

Description

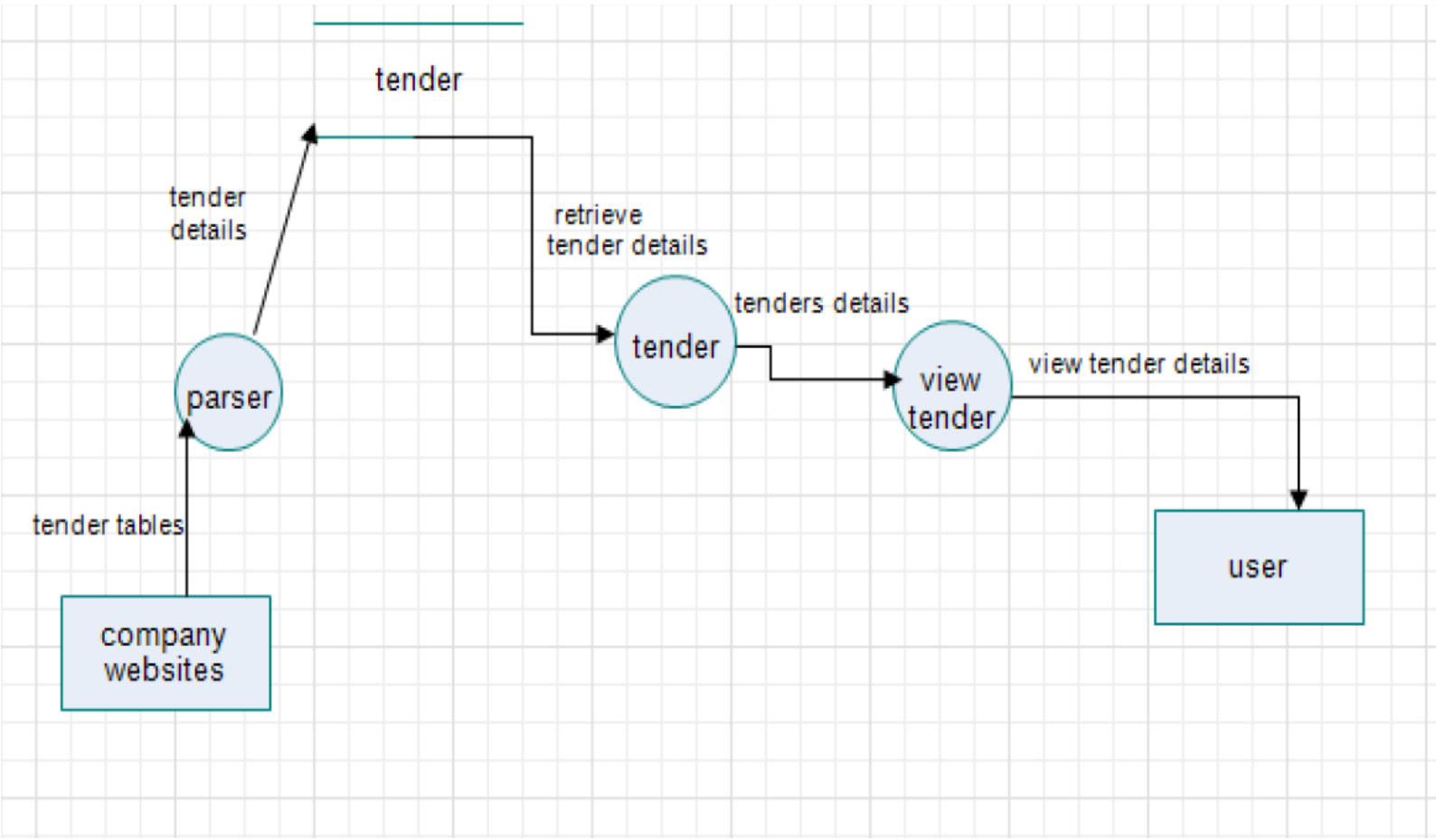
In DFD level 2 diagram we take each process present in DFD level 1 diagram and we further decompose them into sub processes. In the above diagram, we decomposed the login process present in DFD level 1 diagram. Login, session manager, validate and verify are the sub processes of login process present in DFD level 1 diagram. The data flow is also balanced in DFD level 2 diagram.



4.2.7.4 Dataflow Level 2 Diagram of a web based application for BHI

Description

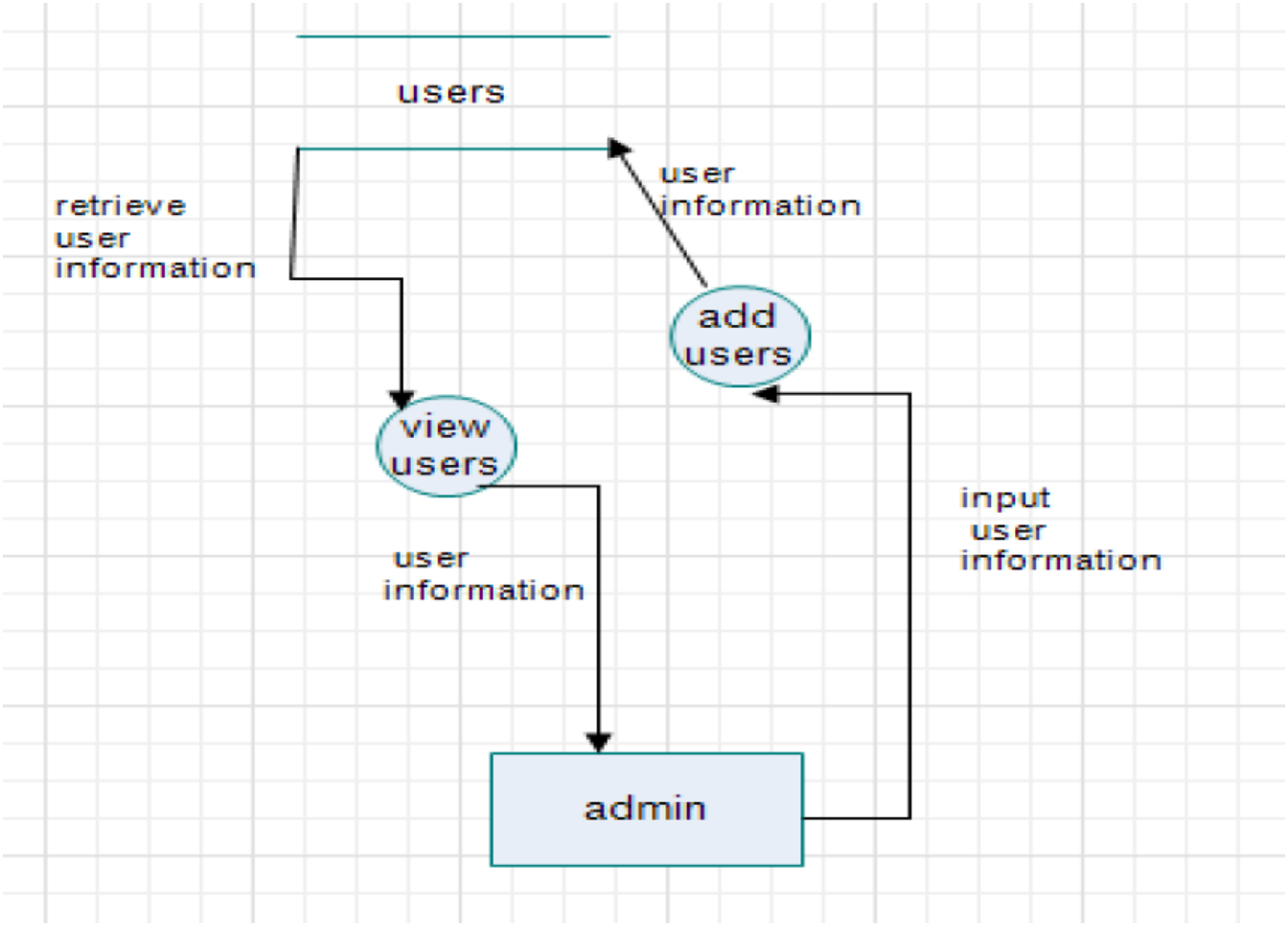
In the above diagram, we decomposed the customer information management process present in DFD level 1 diagram. Company and contact person are the sub processes of customer information management process present in DFD level 1 diagram. There are no new external entities or data stores added in DFD level 2 diagram.



4.2.7.5 Dataflow Level 2 Diagram of a web based application for BHI

Description

In the above diagram, we decomposed the tender process present in DFD level 1 diagram. Tender, view tender and parser are the sub processes of tender process present in DFD level 1 diagram. There are no new external entities or data stores added in DFD level 2 diagram. The data flow is balanced in the above diagram.



4.2.7.6 Dataflow Level 2 Diagram of a web based application for BHI

Description

There are no new external entities or data stores added in DFD level 2 diagram. The data flow is balanced in the above diagram. In the above diagram, we decomposed the user management process present in DFD level 1 diagram. View users and add users are the sub processes of user management process present in DFD level 1 diagram.

4.4 DATA DICTIONARY:

With PHP, you can connect to and manipulate databases. MySQL is the most popular database system used with PHP. MySQL is a database system used on the web. MySQL is a database system that runs on a server..MySQL is ideal for both small and large applications. MySQL is very fast, reliable, and easy to use. MySQL uses standard SQL. The data in a MySQL database are stored in tables. PHP combined with MySQL are cross-platform (you can develop in Windows and serve on a Unix platform) query is a question or a request. In MySQLi extension (I stands for improved). Earlier versions of PHP used the MySQL extension. However, this extension was deprecated in 2012. The following are few of the tables used in BHI web based application.

1) Activate

   Activate table has aid, uid, activation_key, key_expired, key_generated_on and last_email_time as its attributes.

2) Company

   Company table has id, name, type, sector, ownership, phone and website as its attributes.

3) Dashboard

   Dashboard table has did, name, value and last_modified as its attributes.

4) Person

   Person table has id, company_id, name, designation, role, phone, email, skype, notes, contact and updates as its attributes.

5) Recover

   Recover table has rid, uid, recovery_key, key_expired, key_generated_on and last_email_time are its attributes.

6) Relationships

   Relationships table has id, parent_id, child_id and type as its attributes.

7) Sessions

   Sessions table has sid, uid, session_id, session_expired, login_time and logout_time as its attributes.

8) Users

   Users table has uid, username, password, salt, email, access_level and activated as its attributes.

CHAPTER- V

5. IMPLEMENTATION

5.1 MODULE DESCRIPTION

The following are the modules of this project:

5.1.1 Login Module

5.1.2 Customer Information Management Module

5.1.3 Tender Search Module

5.1.4 User Management Module

5.1.1 Login Module:

Login module is an essential module of BHI web based application. It provides user authentication feature. It also employs a captcha to make sure that no program can login to the system. If the user forgets his/her password, then the user can click on "forgot password" link present in Login page. It navigates to another page wherein the user can follow few steps and reset his/her password. This module makes sure that only authenticated user can login to the system.

5.1.2 Customer Information Management Module:

This module allows the user to add company/contact person (details), delete company/contact person details and edit company/contact person details. This module saves a lot of time for BHI. It also specifies various relationships among companies. It specifies different types of relationships among companies for e.g.: X company is a subsidiary of Y company, R company is a parent of Q company etc… It also specifies whether a company is a public or a private sector company. It displays all the contact person associated with each company. It also provides a "live search" feature in order to search company/contact person. This module provides effective management of customer information.

5.1.3 Tender Search Module:

The Tender Search module allows the user to view tenders, delete tenders and mark a tender as active/inactive. This module is developed to perform automated tender search i.e. it

will automatically go through company websites on a daily basis and it parses the html content and adds the new tenders to the database and it views the tender details to the user in the form of a tender list table. This module saves the effort and time taken by BHI to search and manage tenders.

5.1.4 User Management Module:

This module can be accessed only by the admin. It allows the admin to manage the user (employee) information. The admin can add user, edit user, activate/deactivate user and delete user. This module is aimed to provide efficient management of user information.

5.2 ALGORITHMS/PRINCIPLE:

The core logic of this web based application is written in Php. Apart from that we also used few Python scripts in the "Tender Search" module. It is used to parsing purpose. We employed an object oriented approach for coding in Php. OOP offers encapsulation, inheritance, polymorphism and composition, it helps you build much more extendable, reusable, maintainable, and organized application. The advantages are more evident in larger programs, which are pretty much impossible to work on with procedural style, especially when teamwork is required. It also reduces the number of lines of code and it provides better extensibility. The diagram below explains the complete process involved in tender search.

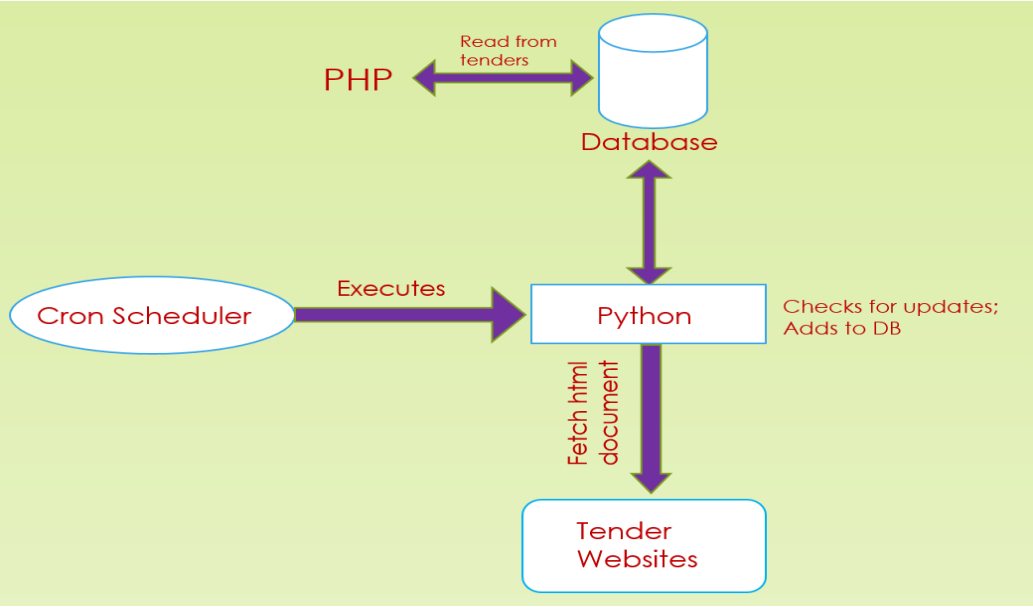


Fig 5.2 Process of Tender search

5.3 SAMPLE CODE:

```
<?php
 class User {
    const PASSWORD_MIN_LENGTH = 8;
    const PASSWORD_MAX_LENGTH = 200;
    const USERNAME_MIN_LENGTH = 3;
    const USERNAME_MAX_LENGTH = 30;

    private $uid;
    private $username;
    private $password;
    private $salt;
    private $email;
    private $accessLevel;
    private $activated;
    public $hasDetails;
    public $conn;

    function __construct() {
     //do nothing
     $this->conn = new Database();
     $this->hasDetails = false;
    }

    private function has_details() {
     if($this->hasDetails)
       return true;
     else
       throw new AppException("User not loaded", AppException::USER_ERROR);
    }

    public function get_all_ids() {
     $stmt = $this->conn->prepare("SELECT uid FROM users");
     $stmt->execute();
     $result = $stmt->get_result();

     if($result->num_rows > 0) {
      for($i=0;$i<$result->num_rows;$i++) {
       $row = $result->fetch_assoc();
       $uid[$i] = $row["uid"];
```

```
   }
   return $uid;
  }
  else {
   throw new AppException("No users found!", AppException::USER_ERROR);
  }
}

public function get_access_level() {
  if($this->has_details())
   return $this->accessLevel;
}

public function get_id() {
  if($this->has_details())
   return $this->uid;
}

public function get_activated() {
  if($this->has_details())
   return $this->activated;
}

public function get_password() {
  //returns hashed password
  if($this->has_details())
   return $this->password;
}

private function gen_hashed_password() {
  if($this->has_details())
   return hash("sha512", ($this->password . $this->get_salt()));
}

public function get_username() {
  if($this->has_details())
   return $this->username;
}

public function get_email() {
```

```
    if($this->has_details())
      return $this->email;
    }

    public function get_salt() {
    if($this->has_details())
      return $this->salt;
    }

    private function set_details($row) {
    //Parameter is the returned row from result in query
    if(isset($row)) {
      $this->uid = $row["uid"];
      $this->username = $row["username"];
      $this->password = $row["password"];
      $this->email = $row["email"];
      $this->salt = $row["salt"];
      $this->accessLevel = $row["access_level"];
      $this->activated = $row["activated"];

      $this->hasDetails = true;
    }
    else {
      throw new AppException("Must provide user details: set_details()",
AppException::USER_ERROR);
    }
    }

    public function load_id($uid) {
    //get details using ID
    $this->uid = sanitize_input($uid);

    $stmt = $this->conn->prepare("SELECT * FROM users WHERE uid = ?");
    $stmt->bind_param("s", $this->uid);
    $stmt->execute();
    $result = $stmt->get_result();

    if($result->num_rows > 0) {
      $row = $result->fetch_assoc();
      $this->set_details($row);
```

```
  }
  return $this->hasDetails;
}

public function load_username($username) {
  //get details using username
  $this->username = sanitize_input($username);

  $stmt = $this->conn->prepare("SELECT * FROM users WHERE username = ?");
  $stmt->bind_param("s", $this->username);
  $stmt->execute();
  $result = $stmt->get_result();

  if($result->num_rows > 0) {
                        //sessionid found in db
                        $row = $result->fetch_assoc();
                        $this->set_details($row);
  }
  return $this->hasDetails;
}

private function gen_salt() {
  $token = new Token();
  return hash("sha512", $token->gen_token());
}

public function gen_hashed_pass($password) {
  return hash("sha512", $password . $this->get_salt());
}
private function user_exists($username) {
  //check if user exists
  $username = strtolower(sanitize_input($username));
  if(!empty($username)) {
    $stmt = $this->conn->prepare("SELECT * FROM users WHERE username = ?");
    $stmt->bind_param("s", $username);
    $stmt->execute();
    $result = $stmt->get_result();
    if($result->num_rows > 0) {
      return true;
    }
```

```
        }
        return false;
    }

    private function validate_details($username, $email, $accessLevel) {
        $this->username = strtolower(sanitize_input($username));
        $this->email = strtolower(sanitize_input($email));
        $this->accessLevel = sanitize_input($accessLevel);

        //Validate input
        if(empty($this->username)) {
            throw new AppException("Username cannot be empty", AppException::USER_ERROR);
        }
        if(strlen($this->username) < self::USERNAME_MIN_LENGTH || strlen($username) > self::USERNAME_MAX_LENGTH) {
            throw new AppException("Username should be at least 3 characters and not greater than 30 characters", AppException::USER_ERROR);
        }
        if(empty($this->email)) {
            throw new AppException("Email cannot be empty", AppException::EMAIL_ERROR);
        }
        if(empty($this->accessLevel)) {
            throw new AppException("Access Level needs to be set", AppException::ACCESS_ERROR);
        }
        else {
            (new AccessLevel)->is_valid_access_level($this->accessLevel);
        }
        return true;
    }

    public function add_user($username, $email, $accessLevel) {
        //add a new user
        //validates user details
        $this->validate_details($username, $email, $accessLevel);

        //check if user already exists
        if($this->user_exists($this->username)) {
            throw new AppException("User already exists!", AppException::USER_ERROR);
```

```
    }
    //Insert user details into DB
    $this->activated = 0;
    $this->salt = $this->gen_salt();

    $stmt = $this->conn->prepare("INSERT INTO users(username, salt, email, access_level, activated) VALUES (?, ?, ?, ?, ?)");
    $stmt->bind_param("sssii", $this->username, $this->salt, $this->email, $this->accessLevel, $this->activated);
    $result = $stmt->execute();

    if(!$result) {
     throw new AppException("Cannot insert user into DB!", AppException::USER_ERROR);
    }
    //Activation token
    if($this->load_username($this->username)) {
     $activate = new Activate($this);
     try {
      if($activate->send_activation_mail()) {
       return true;
      }
     }
     catch(AppException $e) {
      //Rollback user
      $this->delete_user();
      throw new AppException($e->message(), $e->get_code());
     }
    }
    else {
     $this->delete_user();
     throw new AppException("Cannot load user!", AppException::USER_ERROR);
    }
   }

   public function edit_user($username, $email, $accessLevel) {
    //edit existing user
    if($this->has_details()) {
     $prevEmail = $this->email;
     $prevUser = $this->username;
```

```
        //validates user details
        $this->validate_details($username, $email, $accessLevel);

        //check if username is taken by someone
        if(!strcmp($prevUser, $this->username) == 0) {
          if($this->user_exists($this->username)) {
            throw new AppException("Another user is using this username!",
AppException::USER_ERROR);
          }
        }
        //checks if it should send activation mail
        //echo $prevEmail . " : " . $this->email . " : " . $this->activated;
        if((strcmp($prevEmail, $this->email) == 0) && $this->activated == 1) {
          $this->activated = 1;
        }
        else
          $this->activated = 0;
        $this->salt = $this->gen_salt();

        //Insert user details into DB
        $stmt = $this->conn->prepare("UPDATE users SET username = ?, salt = ?, email = ?,
access_level = ?, activated = ? WHERE uid = ?");
        $stmt->bind_param("sssiii", $this->username, $this->salt, $this->email, $this-
>accessLevel, $this->activated, $this->uid);
        $result = $stmt->execute();

        if(!$result) {
          throw new AppException("Cannot update user details!",
AppException::USER_ERROR);
        }

        if($this->activated == 0) {
          //Activate user
          $activate = new Activate($this);
          try {
            if($activate->send_activation_mail()) {
              return true;
            }
          }
          catch(AppException $e) {
```

```
            throw new AppException($e->message(), $e->get_code());
           }
          }
          else {
           return true;
          }
         }
         else {
          throw new AppException("User does not exist!", AppException::USER_ERROR);
         }
        }

        public function delete_user() {
         //delete existing user
         if($this->has_details()) {
          $stmt = $this->conn->prepare("DELETE FROM users WHERE uid = ?");
          $stmt->bind_param("i", $this->uid);
          $result = $stmt->execute();
          if($result == false)
           throw new AppException("Could not delete user", AppException::USER_ERROR);
          else
           return true;
         }
        }
       }
?>
```

CHAPTER- VI

6. TESTING

6.1 IMPORTANCE OF TESTING:

Software testing is a process to determine the quality of the software developed by a developer or programmer. It is a methodological study intended to evaluate the quality-related information of the product. Understanding of the important features and advantages of software testing helps businesses in their day-to-day activities.

Software testing is the is the process used to identify the correctness, completeness and quality of developed computer software .it includes set of activities conducted with the intent of finding errors in software so that it could be corrected before the product is reached to the end users .

In simple words software testing is an activity to check whether the actual results match the expected results and to ensure that the software system is defect free

A number of rules that act as testing objectives are:

- Testing is a process of executing a program with the aim of finding errors.
- A good test case will have a good chance of finding an undiscovered error.
- A successful test case uncovers a new error.

6.2 TYPES OF TESTS:

6.2.1 Unit testing:

Unit testing, also known as component testing refers to tests that verify the functionality of a specific section of code, usually at the function level. In an object-oriented environment, this is usually at the class level, and the minimal unit tests include the constructors and destructors. Unit testing alone cannot verify the functionality of a piece of software, but rather is used to ensure that the building blocks of the software work independently from each other.

Unit testing is a software development process that involves synchronized application of a broad spectrum of defect prevention and detection strategies in order

to reduce software development risks, time, and costs. It is performed by the software developer or engineer during the construction phase of the software development lifecycle. Rather than replace traditional quality assurance focuses, it augments it.

Unit testing aims to eliminate construction errors before code is promoted to quality assurance; this strategy is intended to increase the quality of the resulting software as well as the efficiency of the overall development and quality assurance process. Depending on the organization's expectations for software development, unit testing might include static code analysis, data-flow analysis, metrics analysis, peer code reviews, code coverage analysis and other software verification practices.

6.2.2 Integration testing:

Integration testing is any type of software testing that seeks to verify the interfaces between components against a software design. Software components may be integrated in an iterative way or all together ("big bang"). Normally the former is considered a better practice since it allows interface issues to be located more quickly and fixed.

Integration testing works to expose defects in the interfaces and interaction between integrated components (modules). Progressively larger groups of tested software components corresponding to elements of the architectural design are integrated and tested until the software works as a system.

6.2.3 Functional Test:

Functional testing refers to activities that verify a specific action or function of the code. These are usually found in the code requirements documentation, although some development methodologies work from use cases or user stories. Functional tests tend to answer the question of "can the user do this" or "does this particular feature work."

6.2.4 System Test:

System testing of software or hardware is testing conducted on a complete, integrated system to evaluate the system's compliance with its specified requirements.

System testing falls within the scope of black-box testing, and as such, should require no knowledge of the inner design of the code or logic.

As a rule, system testing takes, as its input, all of the "integrated" software components that have passed integration testing and also the software system itself integrated with any applicable hardware system(s). The purpose of integration testing is to detect any inconsistencies between the software units that are integrated together or between any of the assemblages and the hardware. System testing is a more limited type of testing; it seeks to detect defects both within the "inter-assemblages" and also within the system as a whole.

6.2.5 White Box Testing:

White-box testing (also known as clear box testing, glass box testing, and transparent box testing and structural testing, by seeing the source code) tests internal structures or workings of a program, as opposed to the functionality exposed to the end-user. In white-box testing an internal perspective of the system, as well as programming skills, are used to design test cases. The tester chooses inputs to exercise paths through the code and determine the appropriate outputs.

While white-box testing applied at the unit, integration and system levels of the software testing process, it is usually done at the unit level. It can test paths within a unit, paths between units during integration, and between subsystems during a system–level test. Though this method of test design can uncover many errors or problems, it might not detect unimplemented parts of the specification or missing requirements.

6.2.6 Black Box Testing:

Black-box testing treats the software as a "black box", examining functionality without any knowledge of internal implementation, without seeing the source code. The testers are only aware of what the software is supposed to do, not how it does it.

Black-box testing methods include: equivalence partitioning, boundary value analysis, all-pairs testing, state transition tables, decision table testing, fuzz testing, model-based testing, use case testing, exploratory testing and specification-based testing.

6.3 TEST CASES:

6.3.1 TS_1 Test scenario for user login:

Test case #ID	Test Description	Test case steps	Test data	Expected output	Actual output	Result	Remarks
TC_1	Login with correct username and incorrect password	1) Go to Login page 2) Enter correct username 3) Enter incorrect password 4) Click on Submit	Username: xxxxx Password: xxxxxxxx	Displays an error message “Invalid password ”	Displays an error message “Invalid password”	Pass	When you click on submit button it displays the error message in the same page.
TC_2	Login with incorrect username and correct password	1) Go to Login page 2) Enter incorrect username 3) Enter correct password 4) Click on Submit	Username: xxxx Password: xxxxxxxx	Displays an error message “Invalid Username”	Displays an error message “ Invalid Username ”	Pass	When you click on submit button, it displays error message in the same page.
TC_3	Login with correct username and correct password	1) Go to Login page 2) Enter correct username 3) Enter correct password 4) Click on Submit	Username: xxxxx Password: xxxxxxxx	Login successful and navigates to Dashboard	Login successful and navigates to Dashboard	Pass	Navigates to Dashboard page, which doesn’t have “Users” module.

6.3.2 TS_2 Test scenario for admin login:

Test case #ID	Test Description	Test case steps	Test data	Expected output	Actual output	Result	Remarks
TC_4	Login with correct username and incorrect password	1) Go to Login page 2) Enter correct username 3) Enter incorrect password 4) Click on Submit	Username: xxxxx Password: xxxxxxxx	Displays an error message “Invalid password ”	Displays an error message “Invalid password ”	Pass	When you click on submit button it displays the error message in the same page.
TC_5	Login with incorrect username and correct password	1) Go to Login page 2) Enter incorrect username 3) Enter correct password 4) Click on Submit	Username: xxxx Password: xxxxxxxx	Displays an error message “Invalid Username”	Displays an error message “Invalid Username”	Pass	When you click on submit button it displays error message in the same page.
TC_6	Login with correct username and correct password	1) Go to Login page 2) Enter correct username 3) Enter correct password 4) Click on Submit	Username: xxxxx Password: xxxxxxxxx	Login successful and navigates to Dashboard	Login successful and navigates to Dashboard	Pass	Navigates to Dashboard page, which has “Users” module.

6.3.3 TS_3 Test scenario for Dashboard page (home page):

Test case #ID	Test Description	Test case steps	Test data	Expected output	Actual output	Result	Remarks
TC_7	Verify the "Tender Search" link	1) Login to BHI Website 2) Click on "Tender Search" link present in Dashboard page		Navigates to Tender Search page	Navigates to Tender Search page	Pass	none
TC_8	Verify the "Customers" link	1) Login to BHI Website 2) Click on "Customers" link present in Dashboard page		Navigates to Customers page	Navigates to Customers page	Pass	none
TC_9	Verify the "Users" link	1) Login to BHI Website 2) Click on "Users" link present in Dashboard page		Navigates to Users page	Navigates to Users page	Pass	none
TC_10	Verify the "Settings" link	1) Login to BHI Website 2) Click on "Settings" link present in Dashboard page		Navigates to Settings page	Navigates to Settings page	Pass	none
TC_11	Verify the "Logout" link	1) Login to BHI Website 2) Click on "Logout" link present in Dashboard page		The user will be logged out from the website	The user will be logged out from the website	Pass	none

6.3.4 TS_4 Test scenario for Tender Search page:

Test case #ID	Test Description	Test case steps	Test data	Expected output	Actual output	Result	Remarks
TC_12	Verify "View all" button (function)	1) Login to BHI Website 2) Click on "Tender Search" link 3) Click on "View all" button present in Tender Search page		All the tenders are viewed in tender list.	All the tenders are viewed in tender list.	Pass	A Tender list consisting of active tenders is viewed in Tender Search page. By clicking on "View all" button we can even see inactive tenders in Tender list.
TC_13	Verify "Delete" button (function)	1) Login to BHI Website 2) Click on "Tender Search" link 3) Click on "Delete" button present in Tender Search page		The tender is deleted	The tender is deleted	Pass	Each tender in Tender list table has an active/inactive button and delete button. When we click on delete button of a tender, only that particular tender is deleted.
TC_14	Verify "active" button (function)	1) Login to BHI Website 2) Click on "Tender Search" link 3) Click on "active" button present in Tender Search page		The Tender becomes active	The Tender becomes active	Pass	When we click on active button for a tender in tender list, only that particular tender becomes active. In a similar way, we can also mark a tender as inactive.

6.3.5 TS_5 Test scenario for Customers page:

Test case #ID	Test Description	Test case steps	Test data	Expected output	Actual output	Result	Remarks
TC_16	Verify "Search" function	1) Login to BHI website 2) Click on "Customers" link in Dashboard page 3) Enter the name of company/ contact person in search box 4) Click on search button	xxxxxxx	The company/contact person details are viewed	The company /contact person details are viewed	Pass	This website provides live search feature.
TC_17	Verify "Add" function	1) Login to BHI website 2) Click on "Customers" link in Dashboard 3) Click on Add button	Name: xxxxx Address: xxx Role: xxxx Sector: xxxx Nationality: xx Ownership: xx Phone: xxxx Email: xxx Website: xxx	The company details are added	The company details are added	Pass	The contact person details can also be added in a similar way.
TC_18	Verify "Delete" function	1) Login to BHI website 2) Click on "Customers" link in Dashboard page 3) Click on Delete button		The company (company details) is deleted	The company (company details) is deleted	Pass	It also deletes the contact person associated with the company

6.3.6 TS_6 Test scenario for Users page:

Test case #ID	Test Description	Test case steps	Test data	Expected output	Actual output	Result	Remarks
TC_19	Verify "Add" function	1) Login to BHI Website 2) Click on "Users" link in Dashboard page 3) Click on "Add User" button	Username :xxx Password: xxxxxxx Email: xxxxx	The user is added	The user is added	Pass	When we click on "Add" button, it navigates to page where the user details are filled.
TC_20	Verify "Edit" function	1) Login to BHI Website 2) Click on "Users" link in Dashboard page 3) Click on edit button	Username :xxxxx Password: xxxxxxx Email: xxxxxx	The user (details) is edited	The user (details) is edited	Pass	When we click on "Edit" button, it navigates to page where the user details are edited.
TC_21	Verify "Activate" function	1) Login to BHI Website 2) Click on "Users" link in Dashboard page 3) Add a user and then click on activate button	Username :xxx Password: xxxxxxx Email: xxxx	An activation mail is sent to user's email id	An activation mail is sent to user's email id	Pass	When user opens the activation mail and clicks on activation link, only then the user gets activated.
TC_22	Verify "Delete" function	1) Login to BHI Website 2) Click on "Users" link in Dashboard page 3) Click on delete button		The user is deleted	The user is deleted	Pass	none

6.4 SCREEN SHOTS:

Login page

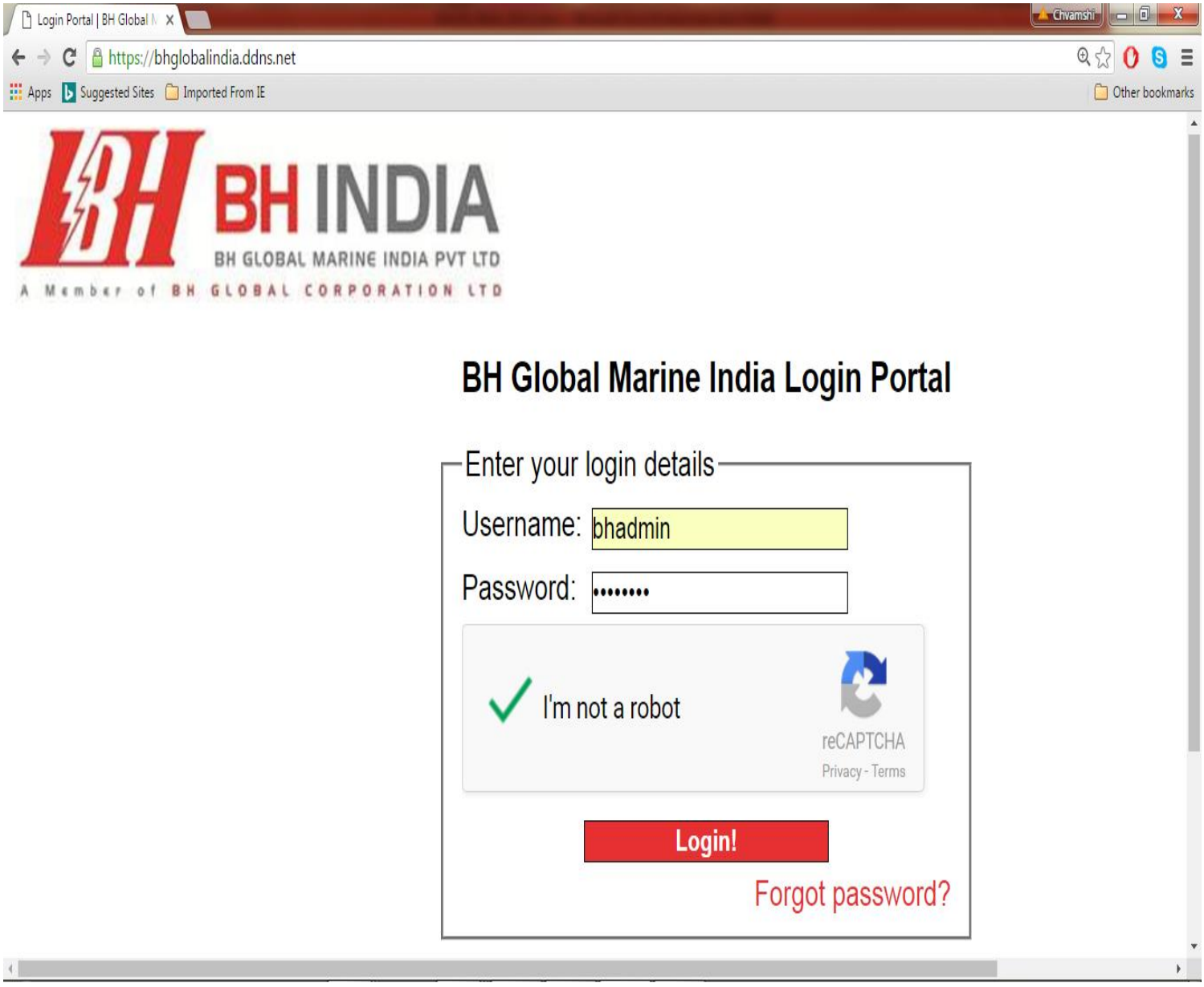


Fig: 6.4.1 Login page

Dashboard page (home page)

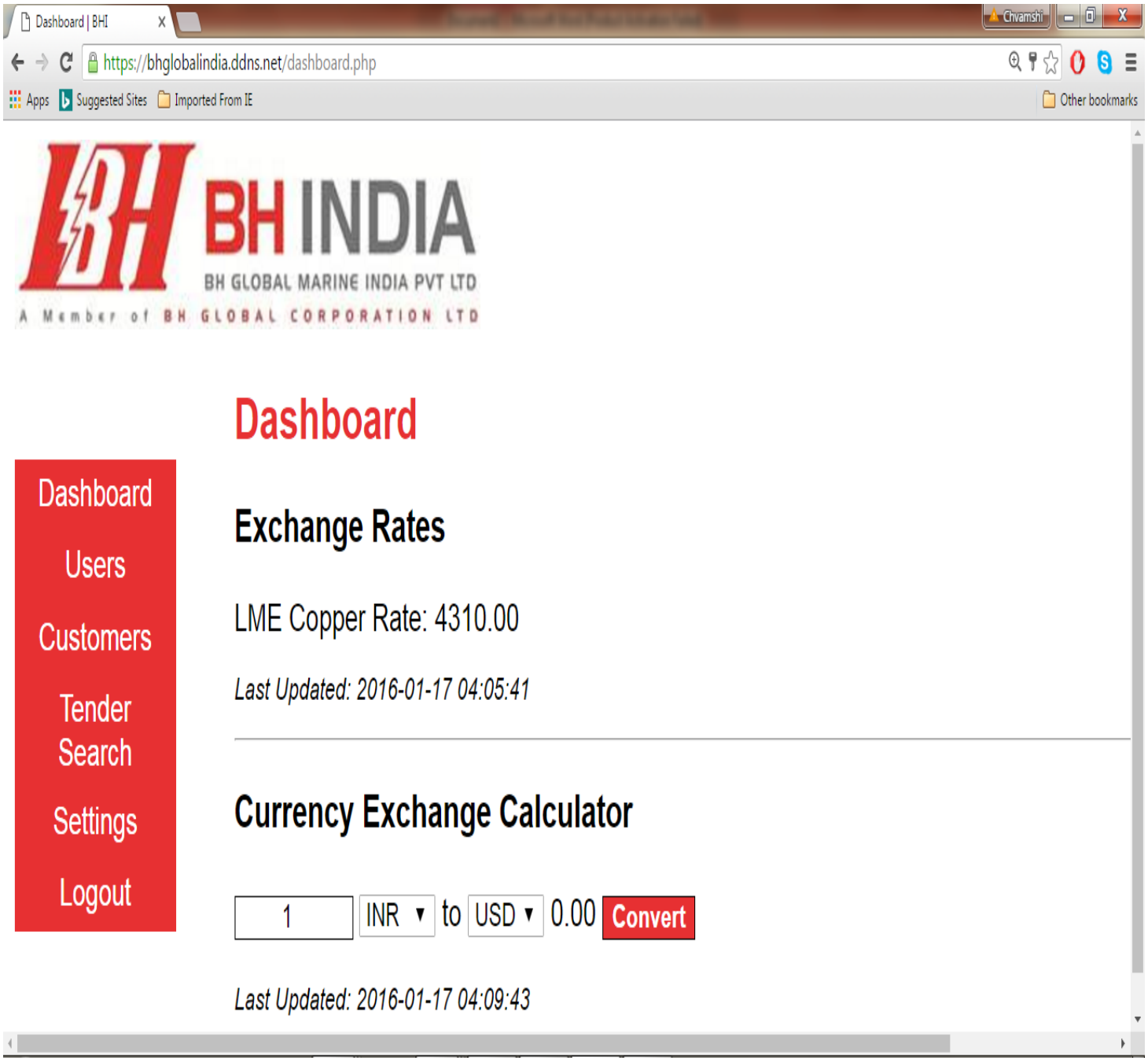


Fig: 6.4.2 Dashboard page (home page)

Users page

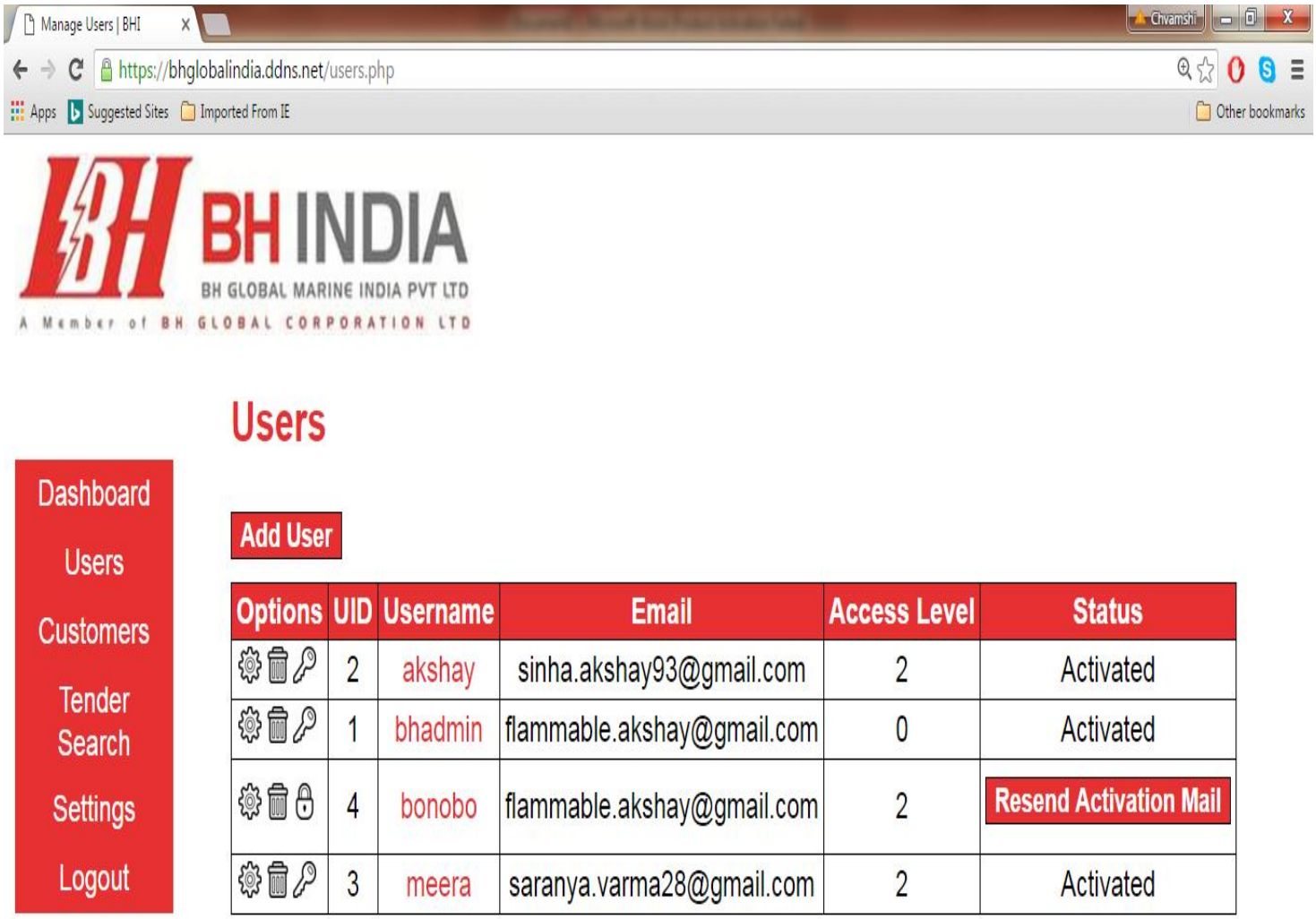


Fig: 6.4.3 Users page

Add user

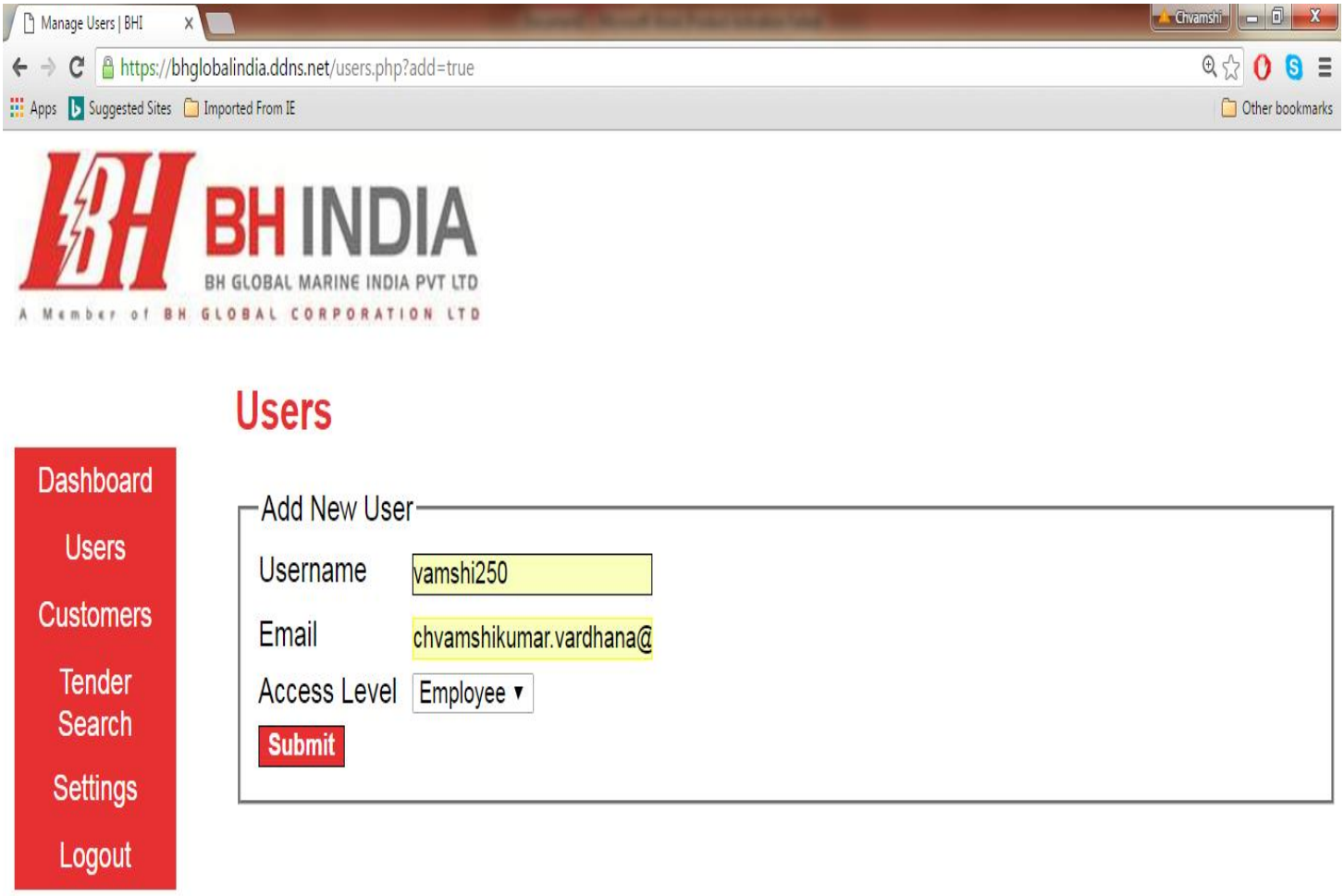


Fig: 6.4.4 Add user

Email for user activation

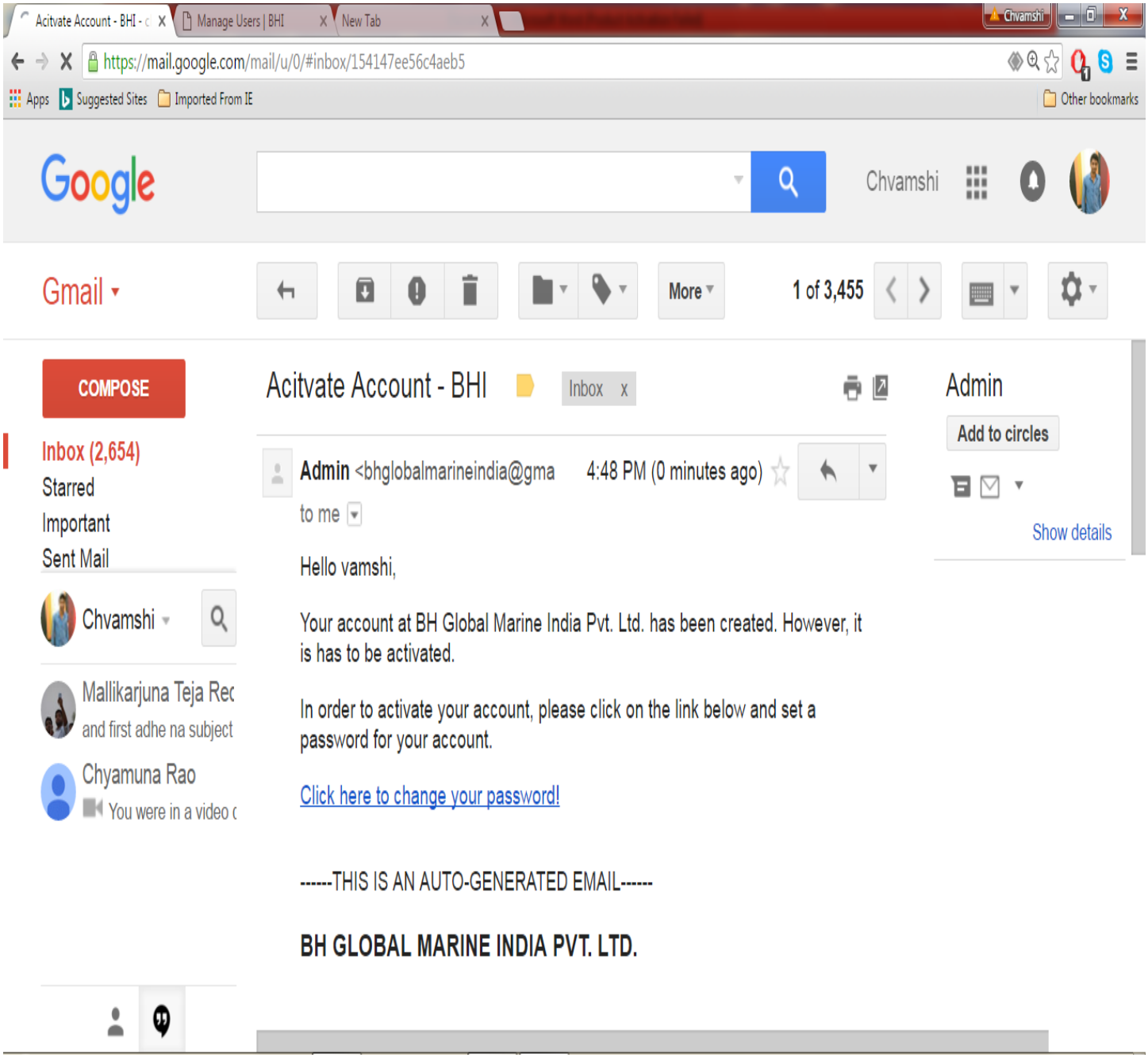


Fig: 6.4.5 Email for user activation

Setting password for user account

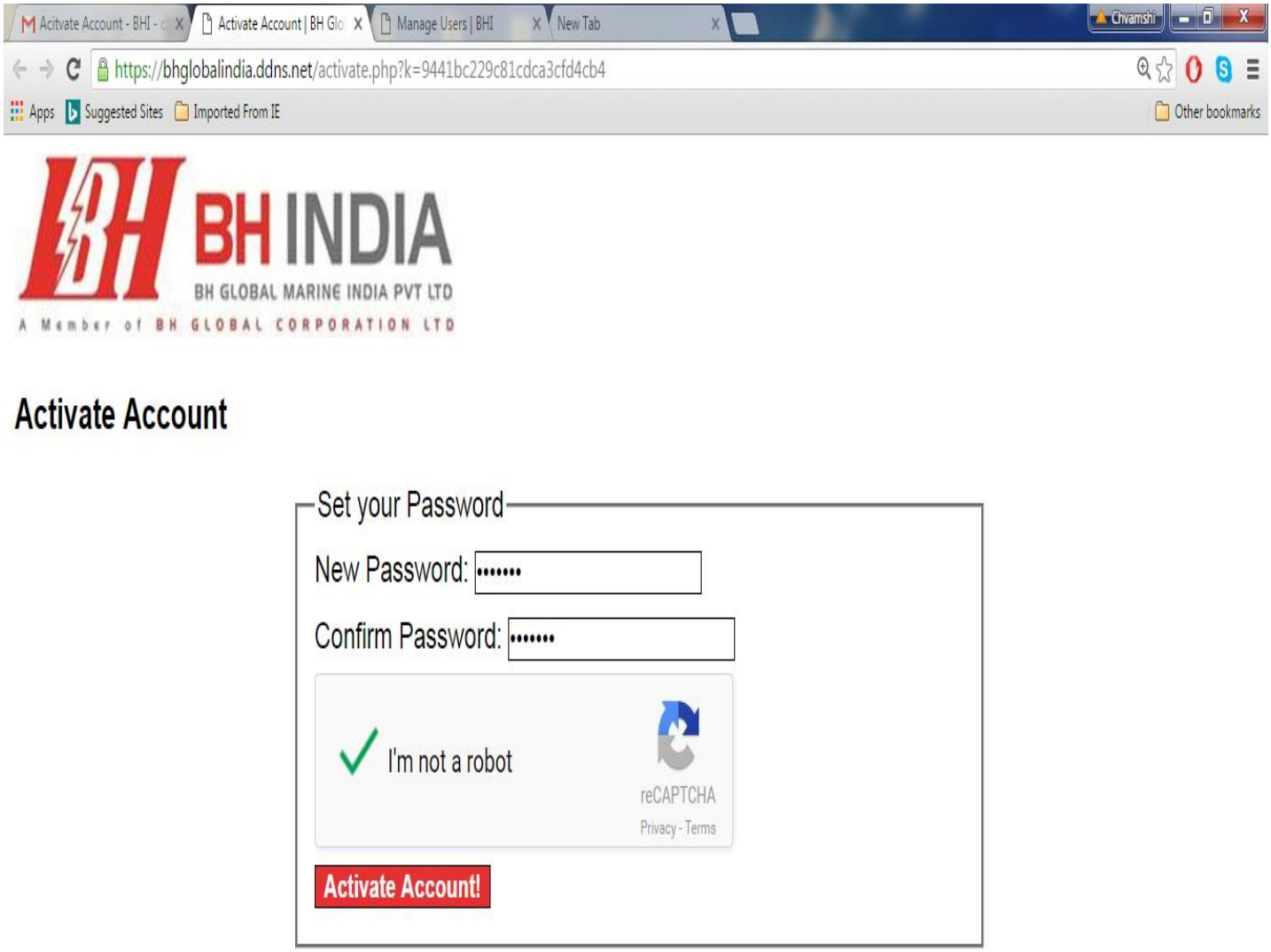


Fig: 6.4.6 Setting password for user account

Successful activation of user

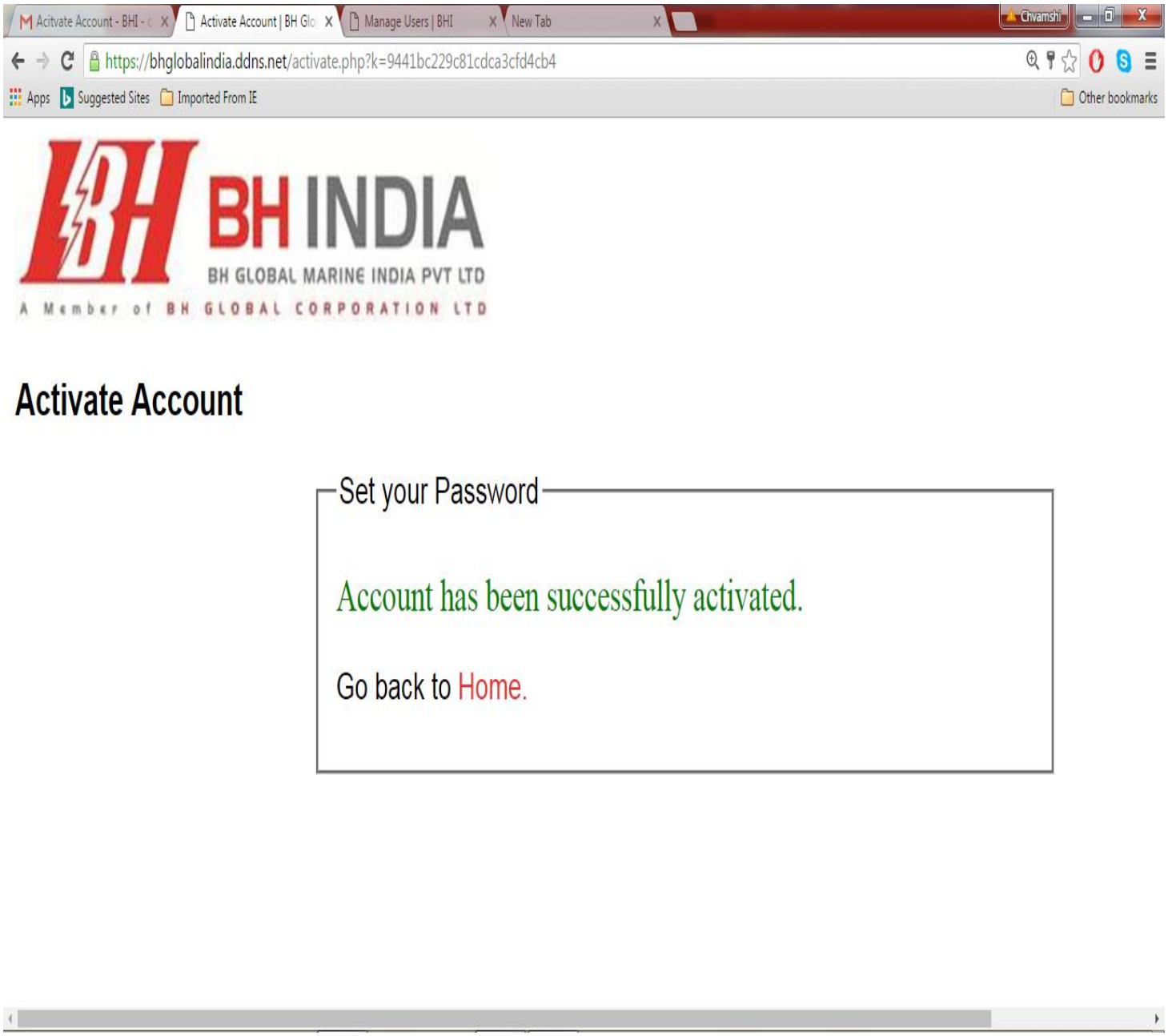


Fig: 6.4.7 Successful activation of user

Editing the existing user

Fig: 6.4.8 Editing the existing user

Customers page

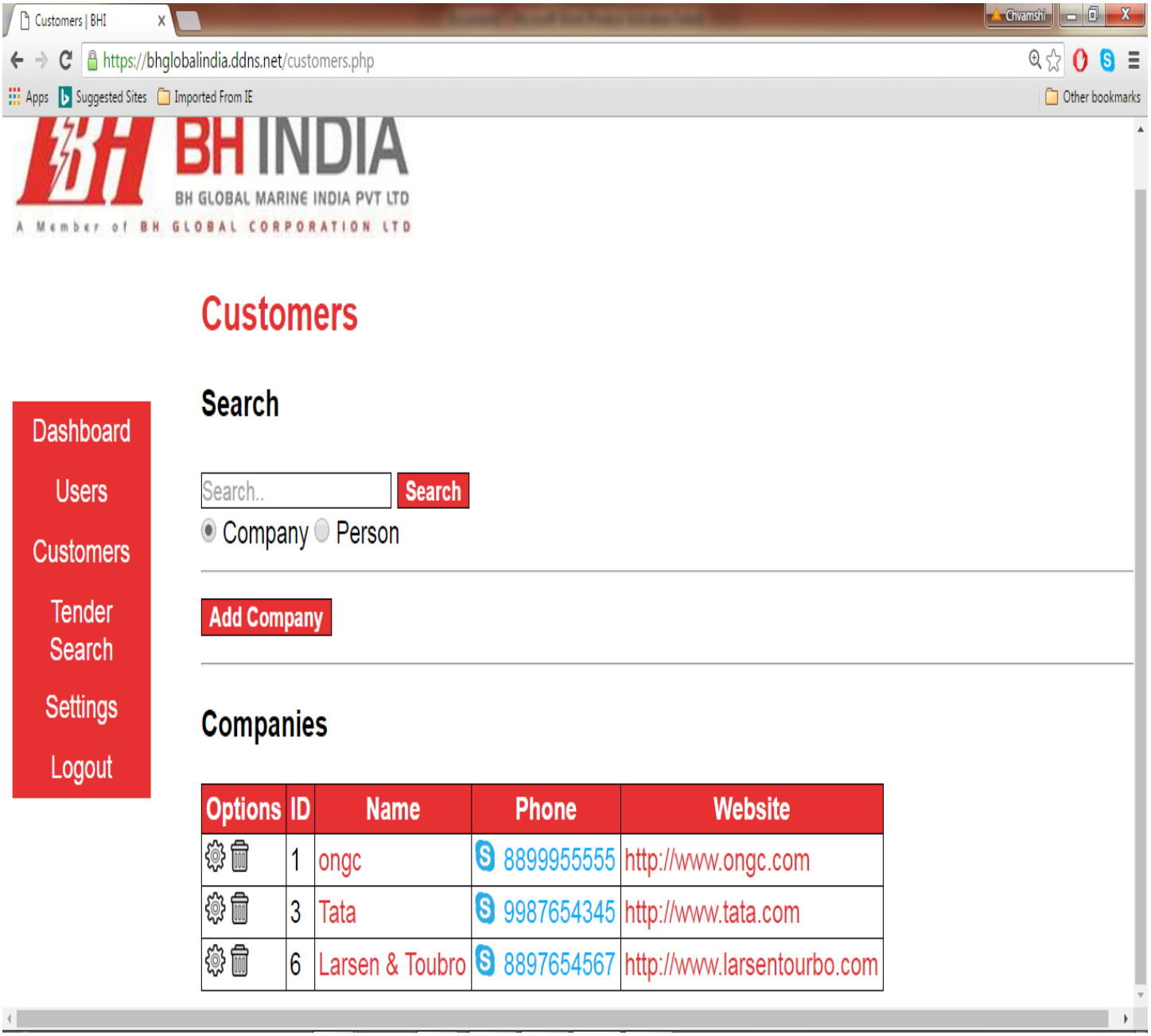


Fig: 6.4.9 Customers page

Editing the existing company

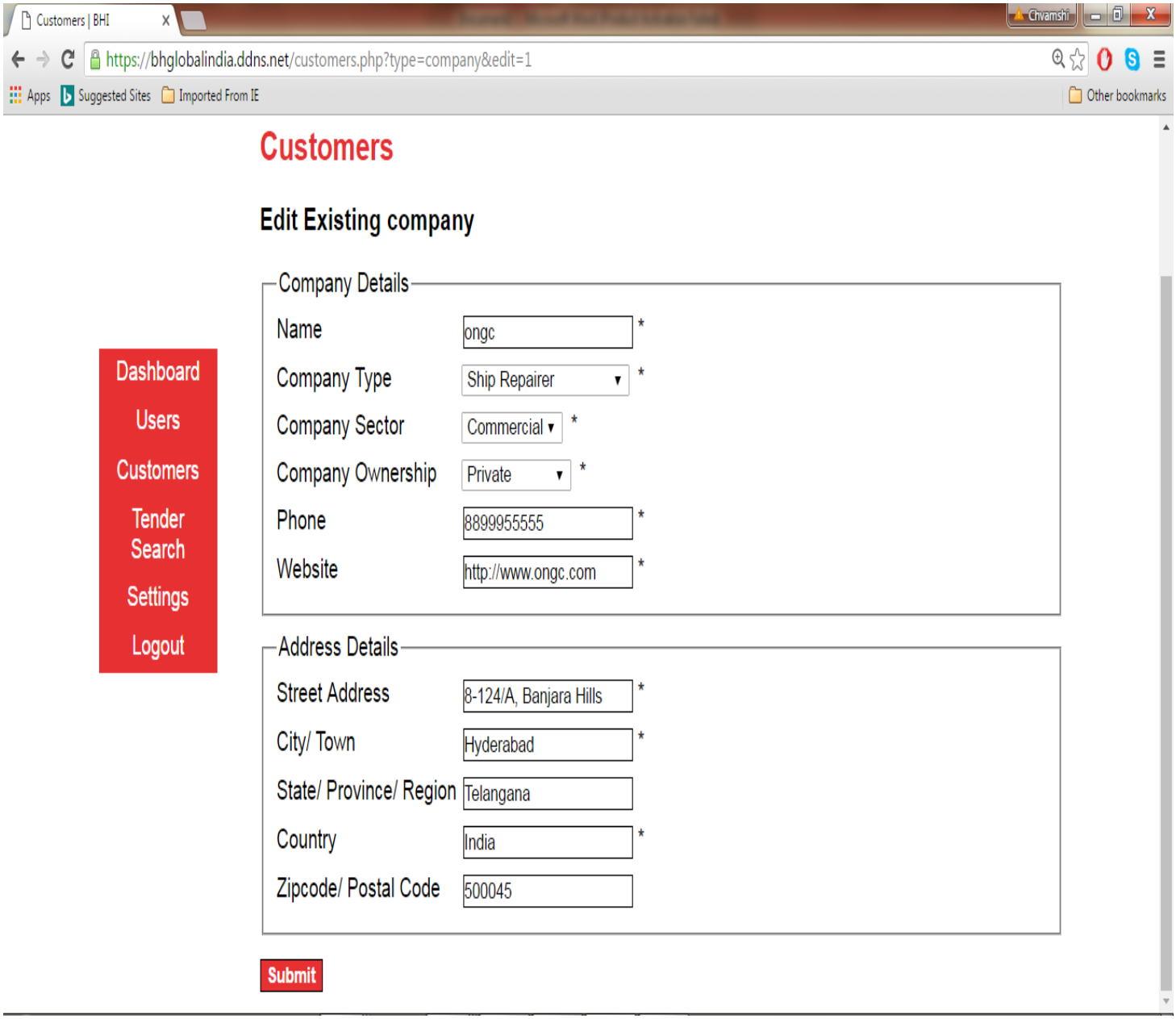


Fig: 6.4.10 Editing the existing company

Company details and its associated employee details

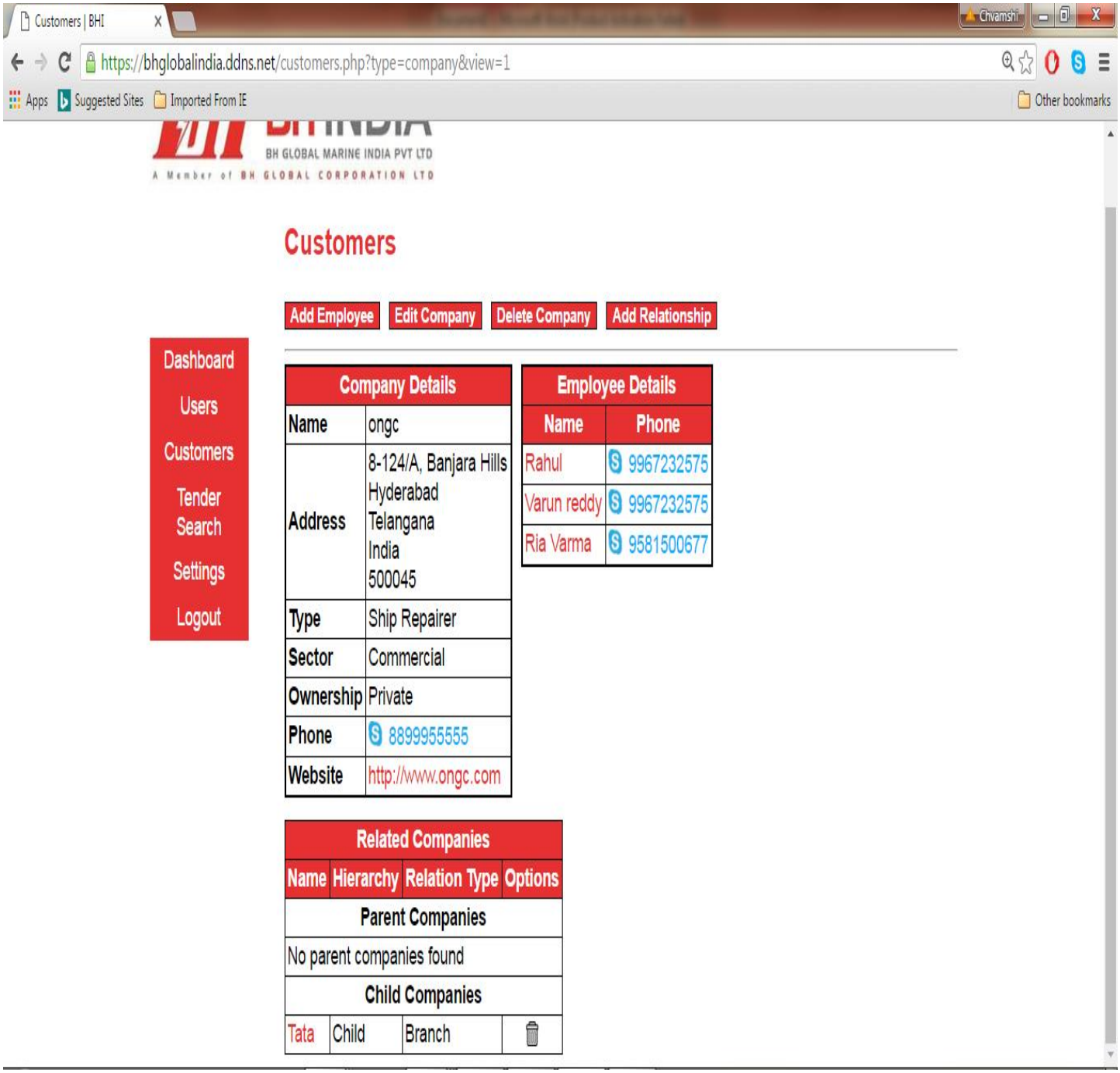


Fig: 6.4.11 Company details and its associated employee details

Tenders page

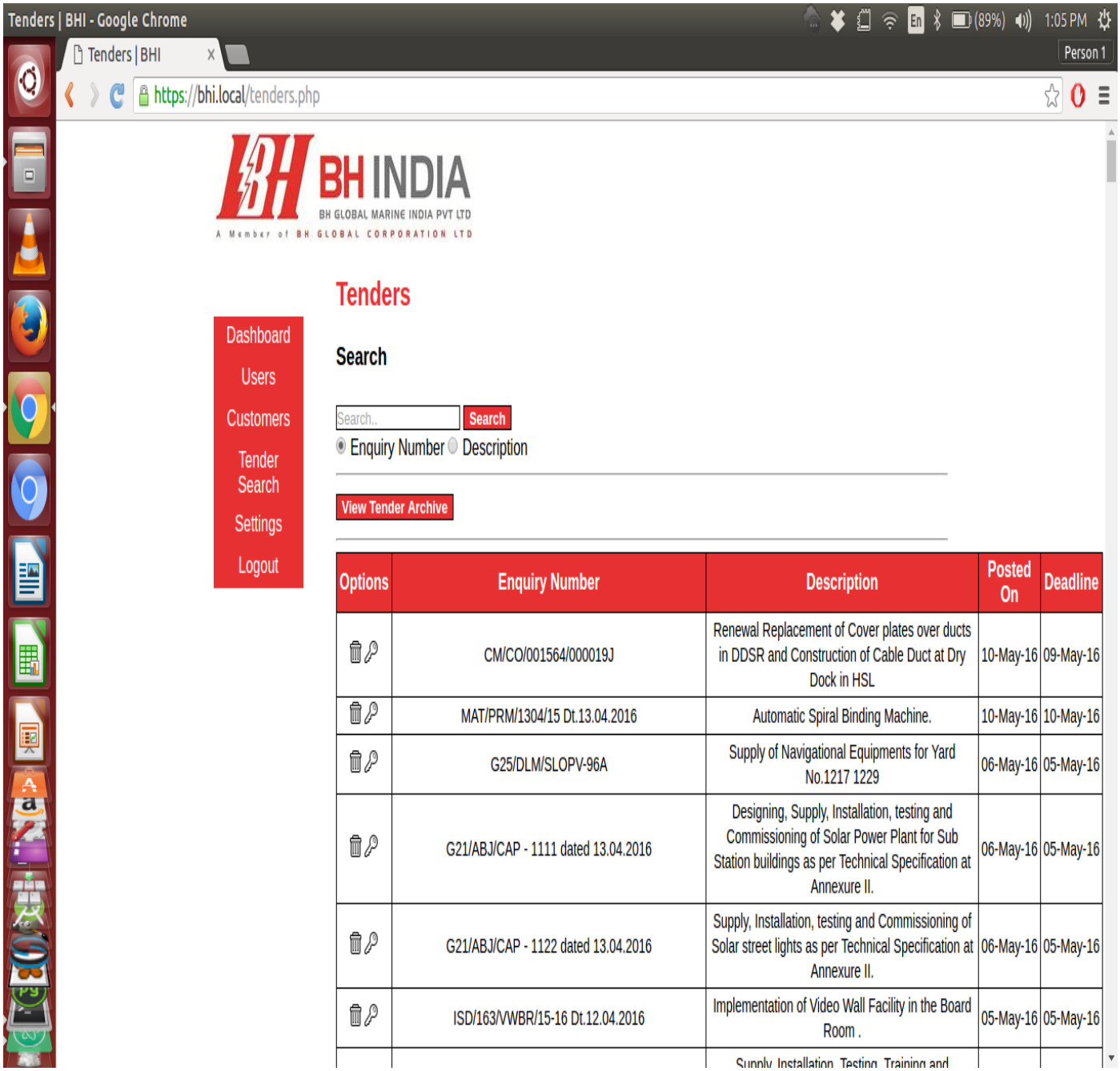


Fig: 6.4.12 Tenders page

Settings page

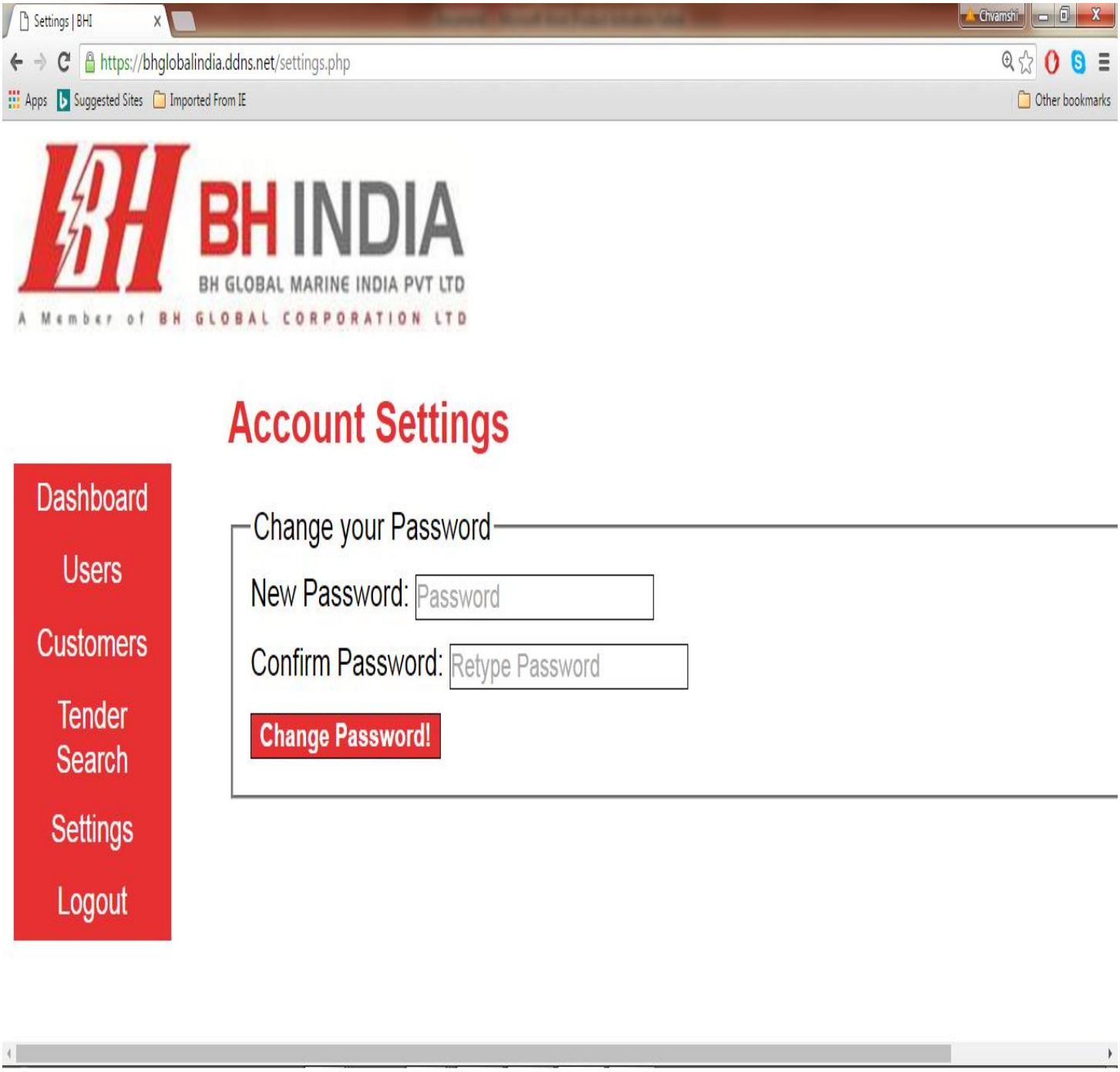


Fig: 6.4.13 Settings page

CHAPTER- VII

7. CONCLUSION AND FUTURE SCOPE

This project is developed in order to help BHI perform various operations in an efficient way. It is a business solution for BHI. It automated the tender search process which saves lots of time and effort taken by BHI. It provides an effectual data management. The customer information management is no more a tough task for BHI, as our web based application allows them to add, edit and delete company/contact person in an efficient way. The manager (admin) is provided with a modern way of managing the employees (users). This web based application overcomes all the problems faced by existing system. It allows BHI to perform their operations with ease, which in turn contributes to growth and success of BHI. This application can perform tender search only on few websites. In future, based on the growth and expansion of BHI, this web based application shall be able to perform tender search on a large number of websites. This web based application shall be able to work efficiently even when it has to handle huge amount of data and large number of users.

CHAPTER- VIII

8. BIBLIOGRAPHY

[1] Robin Nixon, O'Reilly Media; 4 edition, Learning Php, MySQL, JavaScript, CSS & Html5, 300-628pg, December 14, 2014.

[2] Michael McLaughlin, McGraw-Hill Education, Oracle Press, MySQL Workbench: Data Modeling & Development, April 30, 2013.

[3] Apache Contributors, Apache HTTP Server 2.4 Reference Manual, 16 Nov 2015**.**

[4] Ben Frain, responsive web design with HTML5 and CSS3, second edition, 24th Aug, 2015.

[5] Steven Holzner, McGraw Hill education, PHP: The complete Reference, edition 2008.

[6] Jon Duckett, HTML and CSS, design and build websites, 18 Nov 2011.

[7] Christopher Schmitt, O Reilly, HTML 5 COOKBOOK, 21 Nov 2011.

[8] Richard Petersen, Mc Graw Hill Ubuntu: The complete Reference 25 Sep 2008.

[9] Gabriele Lanaro, python high performance programming, 2014.

[10] Clyde Bank Technology, Clydebank media, PHP: quick start guide, April 2015.

[11] Elizabeth Castro, HTML and CSS, visual quick start guide, eighth edition, 2014.

[12] Ben laurie, O Reilly, Apache The definitive guide, Third edition, 10 Jan 2003.

[13] Sklar D, O Reilly, PHPcookbook: solutions and examples for PHP programmers, 2014.

[14] Josh Lockhart Reilly, modern PHP, 20 Feb 2015.