

**PROJECT REPORT**

**ON**

**SALESFORCE-OUTLOOK(64-bit) CONNECTOR**

**FOR**

**PERSISTENT SYSTEMS Ltd.**

**BY**

**SHUBHANKAR RAJ**

**UNIVERSITY OF PUNE**

**MASTER OF COMPUTER APPLICATION**

**M.E.S's**

**INSTITUTE OF MANAGEMENT AND CAREER COURSES**

**(IMCC), PUNE-411029**

**2012-13**



Apr 23, 2013

**To Whomsoever it May Concern**

This is to certify that **Shubhankar Raj** is undergoing industrial training at Persistent Systems Ltd. Pune under the guidance of Mr. Dineshsing Girase.

The details are as follows: -

Duration: 09<sup>th</sup> January 2013 to 05<sup>th</sup> July 2013

Project Title: "**Salesforce – Outlook (64-bit) Connector**"

**For Persistent Systems Limited**

A handwritten signature in black ink, appearing to read 'Sanjay', with a stylized flourish extending to the right.

**Sanjay Karmarkar**  
**Associate Senior Manager-Human Resources**

## Acknowledgement

I express heartfelt gratitude to **Persistent Systems Ltd** and **Mr. Dineshsing Girase** for helping me to carry out the project. The other team members have also played an important part in this project.

I am grateful to **Dr. V. H. Inamdar, Director of IMCC** for providing us with all the resources needed for the project.

I would also like to thank **Dr. Santosh Deshpande, Head of Department** for his support.

I would like to thank **Mrs. Manasi Bhatte**, my internal project guide, for his invaluable guidance during the course of the project. I am indebted to her for giving me her valuable time and cooperation.

Shubhankar Raj

## Index

|      | <b>Contents</b>                              | <b>Page No.</b> |
|------|--|-----------------|
|      | <b>1:INTRODUCTION</b>                        |                 |
| 1.1  | Company Profile                              | 1-3             |
| 1.2  | Existing System and Need for System          | 4-7             |
| 1.3  | Scope of Work                                | 8-10            |
| 1.4  | Operating Environment- Hardware and software | 11              |
| 1.5  | Details Description of Technology Used       | 12-18           |
|      | <b>2:PROPOSED SYSTEM</b>                     |                 |
| 2.1  | Proposed System                              | 19-20           |
| 2.2  | Objective of System                          | 21              |
| 2.3  | User Requirements                            | 22              |
|      | <b>3:ANALYSIS &amp; DESIGN</b>               |                 |
| 3.1  | Object Diagram                               | 23              |
| 3.2  | Class Diagram                                | 24              |
| 3.3  | Use Case Diagram                             | 25-28           |
| 3.4  | Activity Diagram                             | 29-31           |
| 3.5  | Sequence Diagram                             | 32              |
| 3.6  | Module Hierarchy                             | 33              |
| 3.7  | Component Diagram                            | 34              |
| 3.8  | Deployment Diagram                           | 35              |
| 3.9  | Module Specification                         | 36-38           |
| 3.10 | User Interface Design                        | 39-45           |
| 3.11 | Test Procedures and Implementation           | 46-55           |

| <b>4:USER MANUAL</b> |   |       |
|----------------------|---|-------|
| 4.1                  | User Manual                             | 56-67 |
| 4.2                  | Operations Manual                       | 68-70 |
| 4.3                  | Program Specifications                  | 71-77 |
|                      | <b>Drawbacks and Limitations</b>        | 78    |
|                      | <b>Proposed Enhancements</b>            | 79    |
|                      | <b>Conclusions</b>                      | 80    |
|                      | <b>Bibliography</b>                     |       |
|                      | <b>Annexure 1:User Interface Screen</b> |       |
|                      | <b>Annexure 2:Output Report</b>         |       |
|                      | <b>Annexure 3:Sample Code</b>           |       |

# **CHAPTER 1: INTRODUCTION**

## **1.1 Company Profile**

**Name:** Persistent Systems Limited

**Website:** [www.persistentsys.com](http://www.persistentsys.com)

### **Persistent Systems Ltd.**

#### **1.1.1 Company Information**

Established in 1990, Persistent is a global company specializing in software products and technology services. For more than two decades, Persistent has been an innovation partner for world's largest technology brands, leading enterprises and pioneering start-ups. With a global team of 7,000+ employees, Persistent has 300+ customers spread across North America, Europe and Asia.

#### **1.1.2 Core Competencies**

Today, Persistent focuses on best-in-class solution in four key next generation technology areas: Cloud Computing, Analytics, Mobility and Collaboration, for telecommunication, life sciences, consumer

packaged goods, banking & financial services and health care verticals.

### **1.1.3 Persistent Foundation**

Persistent has been contributing to local and regional Health and Education institutions since 1995. In 2009, the Persistent Foundation was formally established as a public charitable trust to aid in the company's charitable activities. Persistent earmarks 1% of net profit for Persistent Foundation programs.

Persistent Foundation is primarily involved in the following three key areas:

- Healthcare
- Education
- Community Development



### **1.1.4 Persistent Services**

Persistent has created a professional services practice based on its well established product knowledge, deep technical expertise and experience working with product teams.

Persistent delivers integrated end to end solutions and services for our professional services customers. In order to deliver best in class solutions we leverage our skills and expertise across major industry domains as well as IT infrastructure. This helps our clients make their businesses more efficient, agile and responsive to their customers' needs.

#### **Persistent provides professional services in several key areas:**

- Security
- Big Data
- BI & Analytics
- Cloud
- Mobility

## **1.2 Existing system and Need for System**

### **Existing system**

Contact specific data that is in Outlook is pretty much visible only to the owner of the Outlook account and it is not reflected in the Company CRM and vice-versa. In the same way Events and Tasks scheduled in Outlook is also not reflected in the Company CRM and vice-versa. These encounters following limitations:-

- I) Users don't have a single view of the relationship.
- II) Other people on your team don't have access to the information.
- III) Hard to assist in activity tracking

Existing Salesforce for Outlook doesn't facilitates user to schedule synchronization time period according to their need. Existing Synchronizing time period is 1 hr.

Existing System doesn't provide log information to the user.

Additionally, there are several connectors available in the market place for syncing the Contacts between the personal

Outlook Account and Company CRM, but all connector doesn't operate on 64-bit Operating System and 64-bit Outlook.

### **Need for System**

The key reason to create integrations between Outlook and CRM system is to ensure that Contact specific data that is in Outlook (and pretty much visible only to the owner of the Outlook account) is reflected in the company CRM system. In the same way when a task and event is scheduled to a CRM account than it will be automatically reflected in the personal Outlook Account and vice-versa. So that

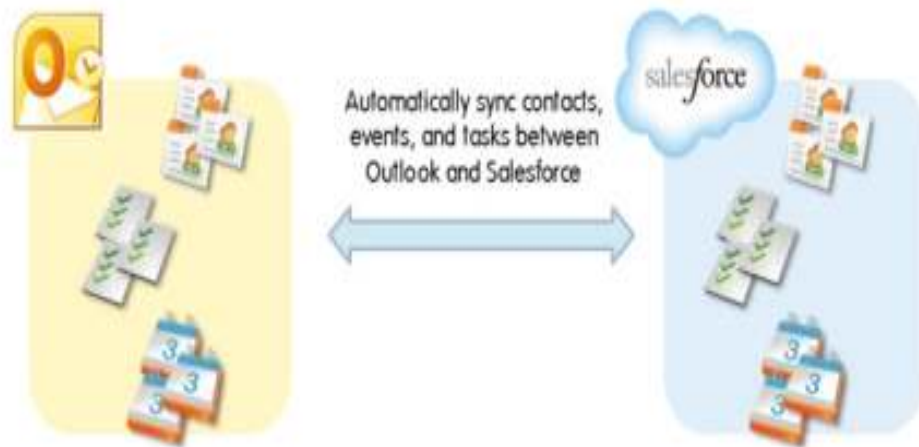
- (i) User have a single view of the relationship
- (ii) Other people on User's team can have access to the information.
- (iii) To assist in activity tracking.

The application should facilitate user to schedule synchronization time period according to their need.

The application should provide information of the history of Synchronization.

Additionally, the application will operate on 32-bit as well as 64-bit Operating System and 32-bit and 64-bit Outlook.

### 1.3 Scope of Work



This Application will work as a middleware Application between Salesforce and Outlook for the Synchronization of Contacts, Events and Tasks. The Application provides various Options for Synchronization of Contacts, Events and Tasks between SalesforceCRM and Outlook. Options are as follows:-

### 1. Selection of Items for Synchronization:

User can select the Items they want to Synchronize.

The Item are:

- a). Contacts.
- b). Events.
- c). Tasks

### 2. Select the direction of Synchronization:

User can set the Synchronization settings according to their need , Synchronization settings provides following Options to the users:

- a). Salesforce CRM To Outlook
- b). Both directions.

### 3. Set the Timer time period on which the Synchronization will be done according to the users settings.

4. User's different configuration settings :
  - a). Outlook Configurations.
  - b). Salesforce CRM Configurations
  - c). Proxy Configurations
  - d). Synchronize Configurationsare stored in the Properties Files and used by the application when the application is executed.
5. User can update different configuration settings except Outlook UserId and SalesforceUserId, Once these two informations are added to the corresponding properties files they can't be modified further.
6. All the Configurations information will be stored in the encrypted format.
7. Application provides logs for all the activities.



## **1.4 Operating Environment - Hardware and Software**

### **Hardware Requirement:**

**Processor** : -Intel Pentium IV or More

**RAM** : - 512 MB.

**Hard Disk** : - 80GB.

### **Software Requirement:**

- Internet Explorer6.0 or later version
- Windows XP or Later
- JDK 1.7, Eclipse
- MS-Outlook 2007 or Later

## **1.5 Detail Description of Technology Used**

This connector application has developed using Java. Java is purely object oriented technology and provides high level of flexibility.

Java is a set of several [computer software](#) products and specifications from [Sun Microsystems](#) (which has since merged with [Oracle Corporation](#)), that together provide a system for developing [application software](#) and deploying it in a [cross-platform](#) computing environment. Java is used in a wide variety of [computing platforms](#) from [embedded devices](#) and [mobile phones](#) on the low end, to [enterprise servers](#) and [supercomputers](#) on the high end. While less common, [Java applets](#) are sometimes used to provide improved and secure functions while browsing the [World Wide Web](#) on [desktop computers](#).

Writing in the [Java programming language](#) is the primary way to produce code that will be deployed as [Java bytecode](#). [Java syntax](#) borrows heavily from [C](#) and [C++](#), but object-oriented features

are modelled after [Smalltalk](#) and [Objective-C](#). Java eliminates certain low-level constructs such as [pointers](#) and has a very simple memory model where every object is [allocated on the heap](#) and all variables of object types are [references](#). Memory management is handled through integrated automatic [garbage collection](#) performed by the JVM.

Swing is the primary [Java GUI widget toolkit](#). It is part of [Oracle's Java Foundation Classes](#) (JFC) — an [API](#) for providing a [graphical user interface](#) (GUI) for Java programs.

Swing was developed to provide a more sophisticated set of GUI [components](#) than the earlier [Abstract Window Toolkit \(AWT\)](#). Swing provides a native [look and feel](#) that emulates the look and feel of several platforms, and also supports a [pluggable look and feel](#) that allows applications to have a look and feel unrelated to the underlying platform. It has more powerful and flexible components than AWT. In addition to familiar components such as buttons, check boxes and labels, Swing provides several advanced

components such as tabbed panel, scroll panes, trees, tables, and lists.

Unlike AWT components, Swing components are not implemented by platform-specific code. Instead they are written entirely in Java and therefore are platform-independent. The term "lightweight" is used to describe such an element.

This Connector uses java APIs provided by Microsoft and Salesforce.

An application programming interface (API) is a [protocol](#) intended to be used as an [interface](#) by [software components](#) to communicate with each other. An API is a library that may include specification for [routines](#), [data structures](#), [object classes](#), and variables. An API specification can take many forms, including an International Standard such as [POSIX](#), vendor documentation such as the [Microsoft Windows API](#), the [libraries](#) of a programming language, e.g., [Standard Template Library](#) in [C++](#) or [Java API](#).

### **EWS Java API :**

Java implementation of the **Exchange Web Services (EWS)** API. This API gives developers programmatic access to Exchange Server 2007 SP1 and above.

Exchange Web Services (EWS) provides the functionality to enable client applications to communicate with the Exchange server. EWS Java Proxy application simplifies the use of EWS in Java client applications, by providing API which is dramatically simpler for developers. EWS Java Proxy provides access to much of the same data that is made available through Microsoft Office Outlook. EWS clients can integrate Outlook data into Line-of-Business (LOB) applications. SOAP provides the messaging framework for messages sent between the client application and the Exchange server. The SOAP messages are sent by HTTP.

### **WSC API:**

The Force.com **Web Service Connector** (WSC) is a high performing web service client stack implemented using a streaming parser. WSC also makes it much easier to use the Force.com API (Web Services/SOAP or Asynchronous/REST API). WSC can be used to invoke any doc literal wrapped web service

The Force.com Web Services Connector (WSC) is a code-generation tool and runtime library for use with Force.com Web services. WSC uses a high-performing Web services client stack implemented with a streaming parser. It is the preferred tool for working with salesforce.com APIs. You can write Java applications with WSC that utilize the Force.com SOAP API, Bulk API, and Metadata API.

### SOAP:

**SOAP**, originally defined as **Simple Object Access Protocol**, is a [protocol](#) specification for exchanging structured information in the implementation of [Web Services](#) in [computer networks](#). It relies on [XML Information Set](#) for its message format, and usually relies on other [Application Layer](#) protocols, most notably [Hypertext Transfer Protocol](#) (HTTP) or [Simple Mail Transfer Protocol](#)(SMTP), for message negotiation and transmission.

SOAP can form the foundation layer of a [web services protocol stack](#), providing a basic messaging framework upon which web services can be built. This XML based protocol consists of three parts: an envelope, which defines what is in the message and how to process it, a set of encoding rules for expressing instances of application-defined datatypes, and a convention for representing procedure calls and responses. SOAP has three major characteristics: Extensibility (security and WS-routing are among the extensions under development), Neutrality (SOAP can be used over

any transport protocol such as [HTTP](#), [SMTP](#), [TCP](#), or [JMS](#))  
and Independence (SOAP allows for any programming model)



# **CHAPTER 2: PROPOSED SYSTEM**

## **2.1 Proposed System**

- **SALESFORCE-OUTLOOK (64-bit) CONNECTOR** is proposed to perform Synchronization between Salesforce and Outlook Account for Outlook 64-bit.
- Application facilitates user to select the items to be synced according to their need. Such as- Contacts, Events, Tasks.
- Application facilitates user to select the Synchronization direction. Such as Salesforce to Outlook and Bi-directional.
- Application facilitates user to schedule the synchronization time for automatic synchronization.
- Application should provide Configuration settings for users. Such as-Outlook Configuration, Salesforce Configuration, Proxy Configuration and Synchronize Configurations. These Configuration Settings are stored in separate properties files for different settings.

- Application stores Sync information in the log files for future use.
- Application is supposed to provide next Synchronize time schedule after each synchronization.
- Application is supposed to provide proper message for the items which are not synced.
- Application is supposed to provide a good User Interface which can be easily handled by the user.

## **2.2 Objective of the System**

- The main objective of the system is to reduce the workload of creating or updating contacts, events and tasks on the opposite sides of the accounts if any new item is created.
- Provide facility to synchronize the items in single click either Salesforce to outlook or Bi-directional.
- Provide facility to schedule the automatic sync time so that user do not have to select sync option manually.
- Helps user to maintain consistency in Salesforceaccount's item information and Outlook account information of items.

## **2.3 User Requirement**

**User Interface :**Application provides user friendly interface so that user will interact easily with the application.

**Consistency:** Since Application is designed for Synchronizing items both ways (Outlook &Salesforce) the item information will be Consistent on both the sides.

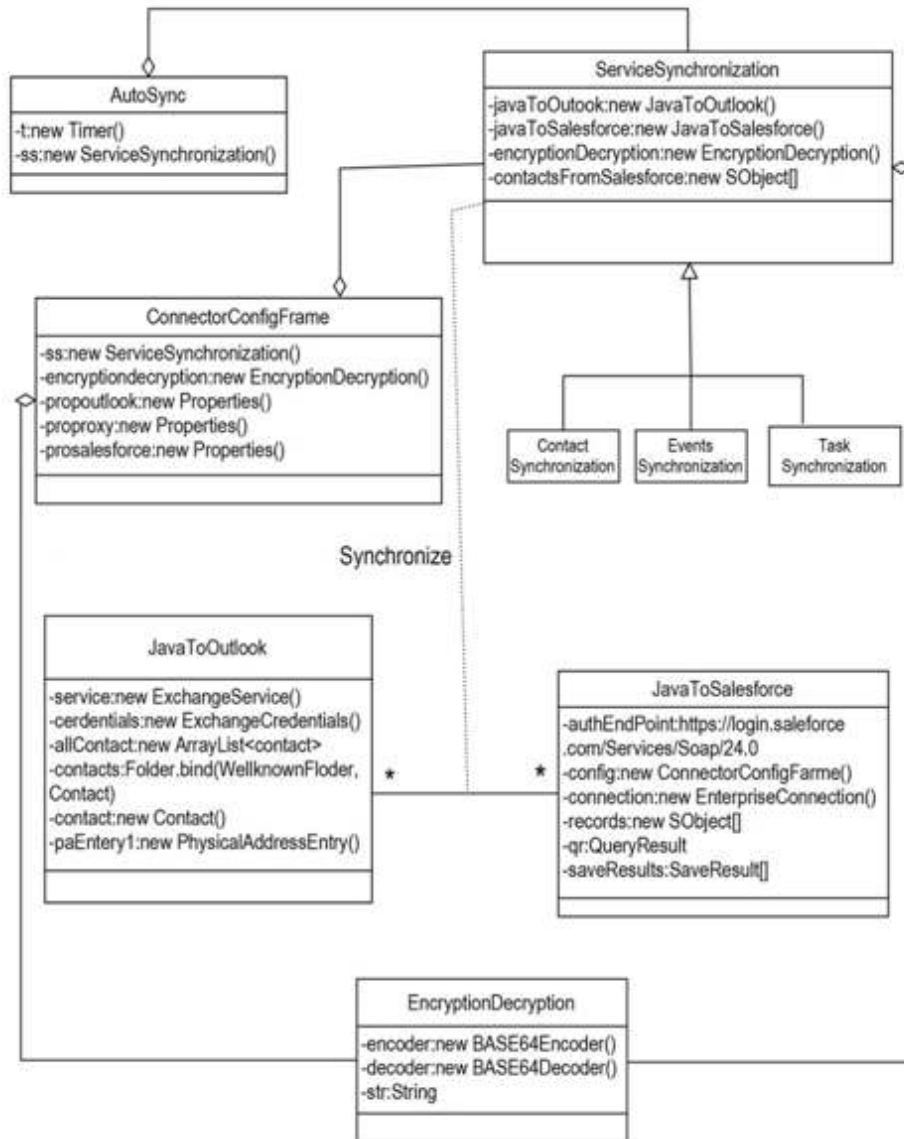
**Flexibility:** Application provide facility to user to select the Sync direction as well as to schedule the Synchronization automatically.

**Automatic:** Application provide facility for automatic sync by setting the timer. So that user don't have to select the sync option manually.

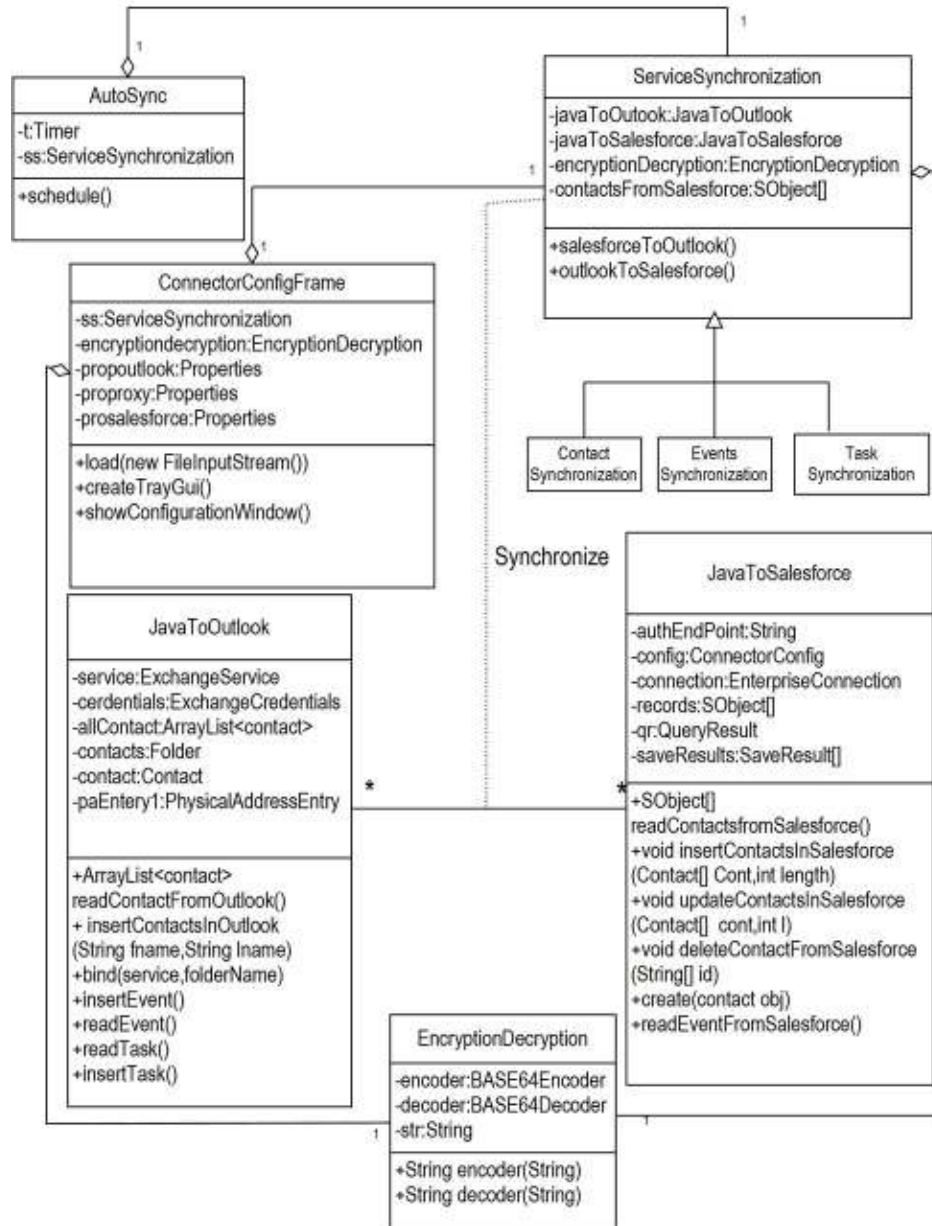
**Item choice:**Application facilitates user to select the items they want to synchronize.

**CHAPTER 3:**  
**ANALYSIS & DESIGN**

### 3.1 Object Diagram



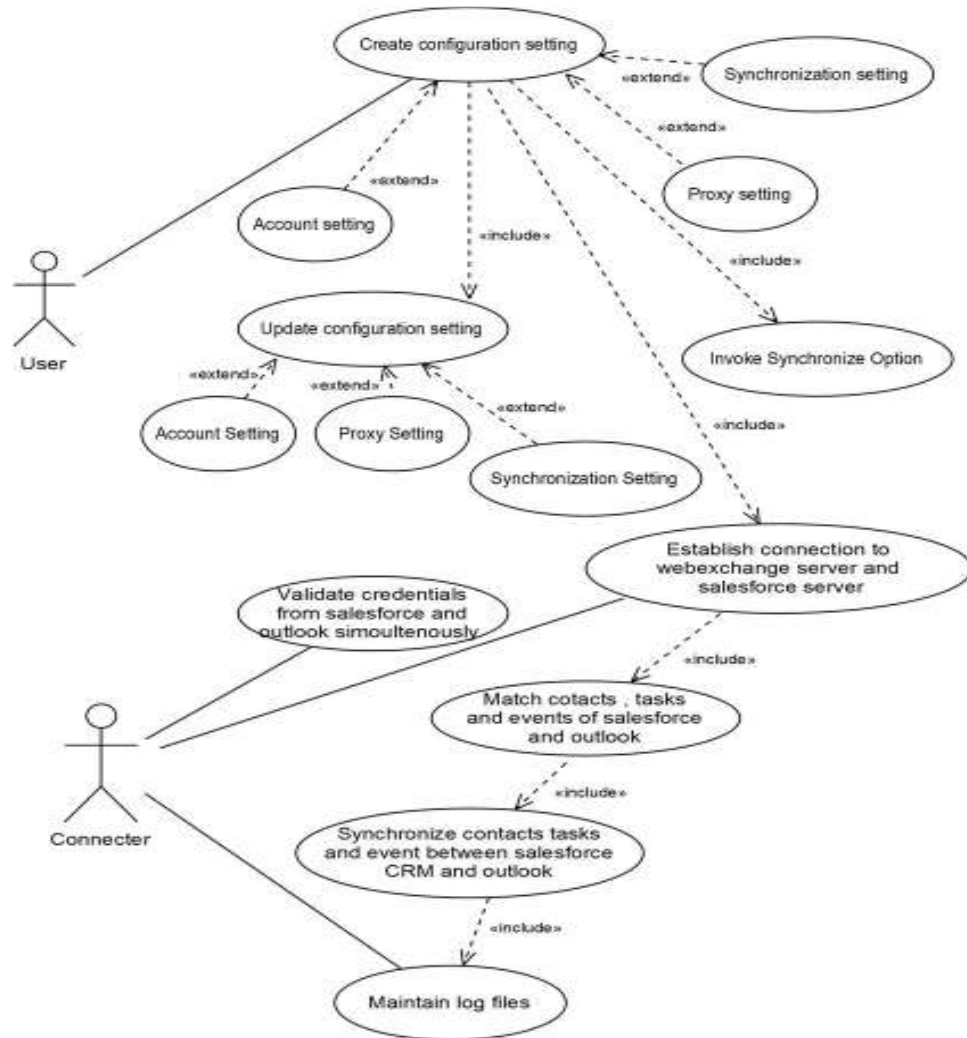
### 3.2 Class Diagram



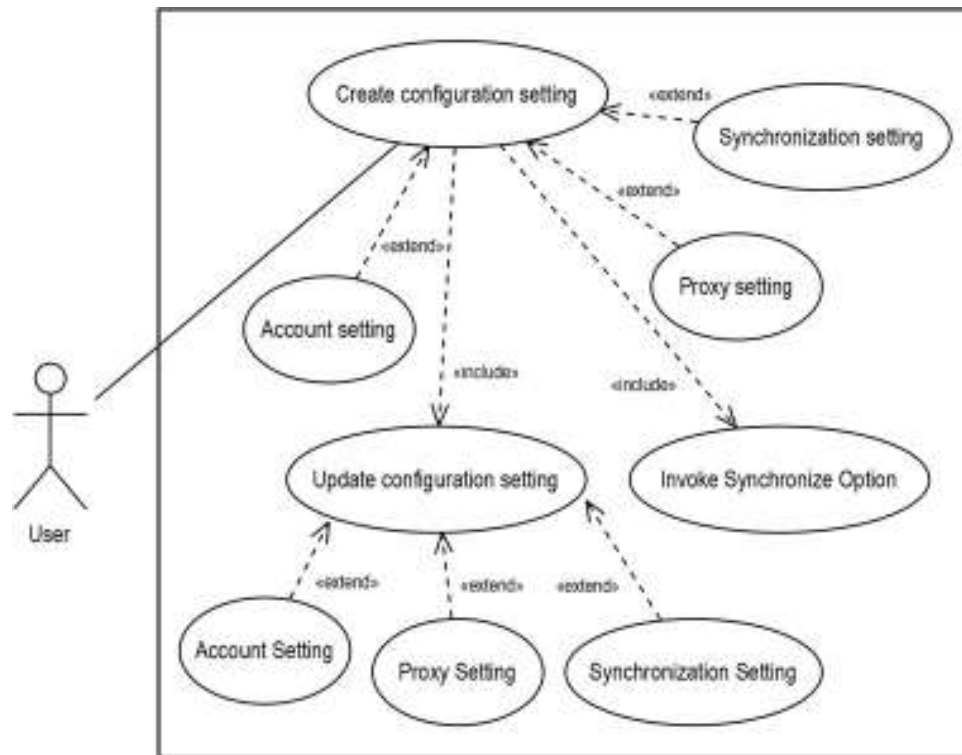


### 3.3 Usecase Diagrams

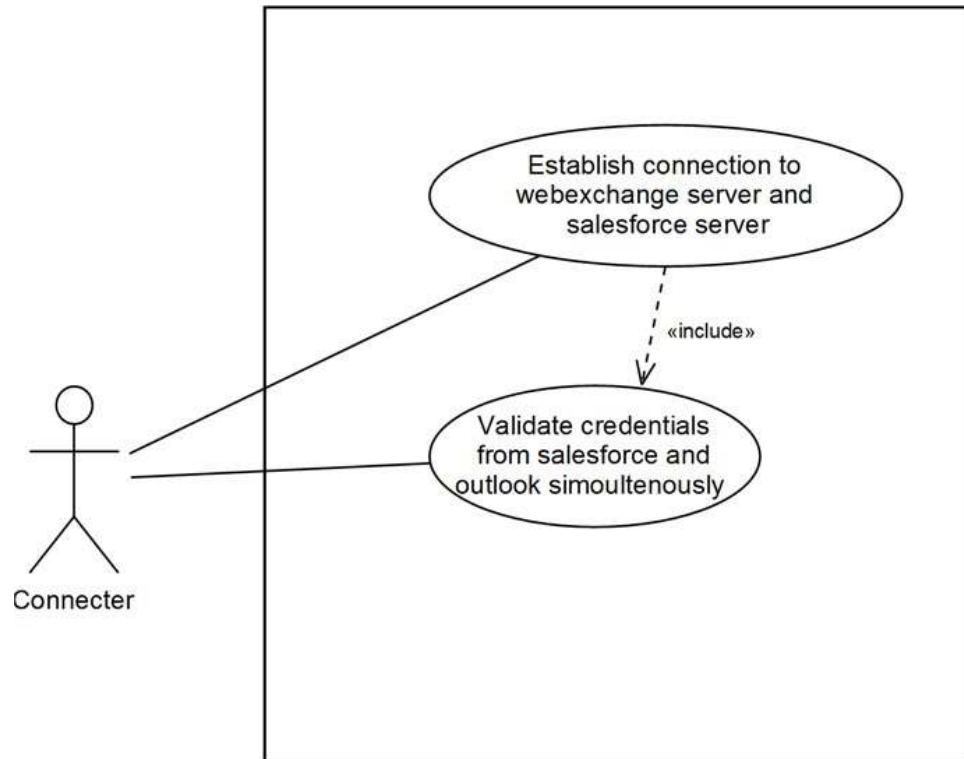
#### 3.3.1 Business Usecase Diagram



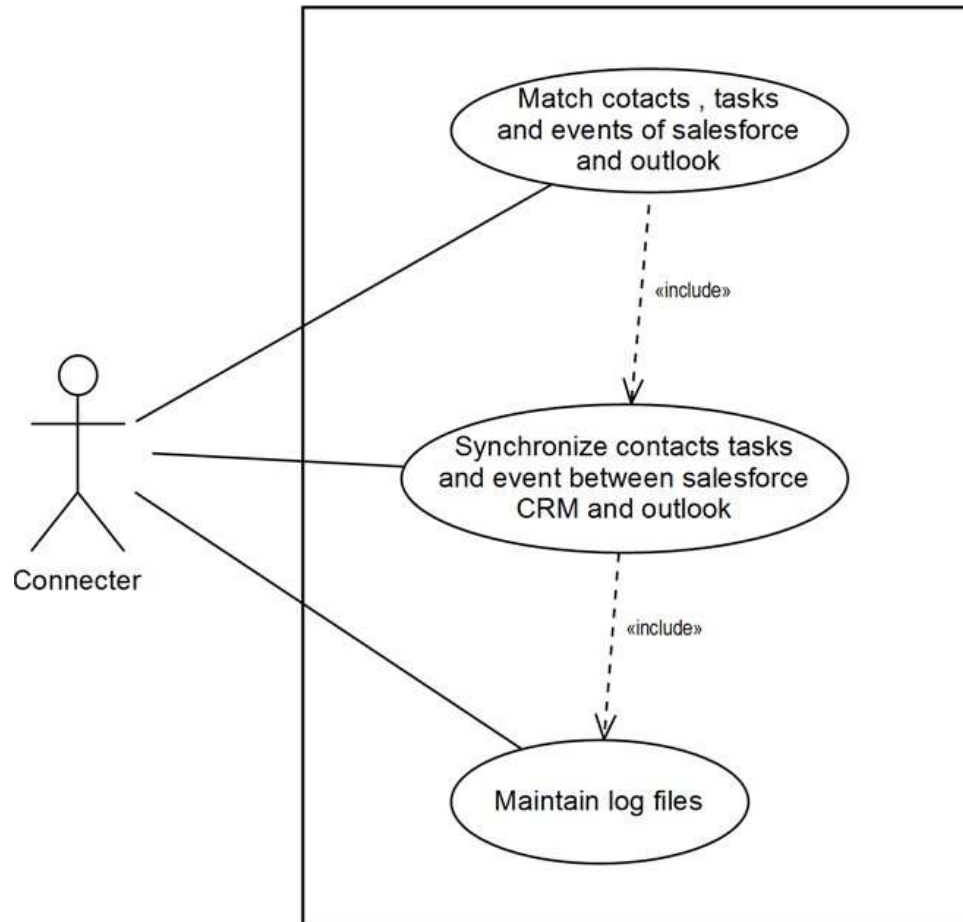
### 3.3.2 Usecase Diagram for User's Configuration Settings



### 3.3.3 Usecase Diagram for Establishing Connection

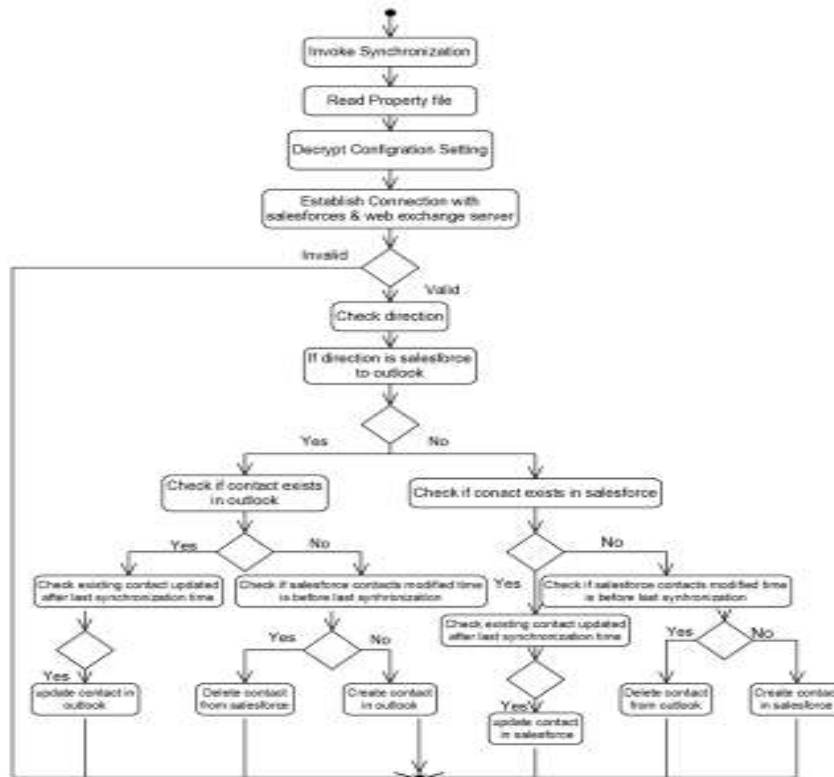


### 3.3.4 Usecase Diagram for Synchronization

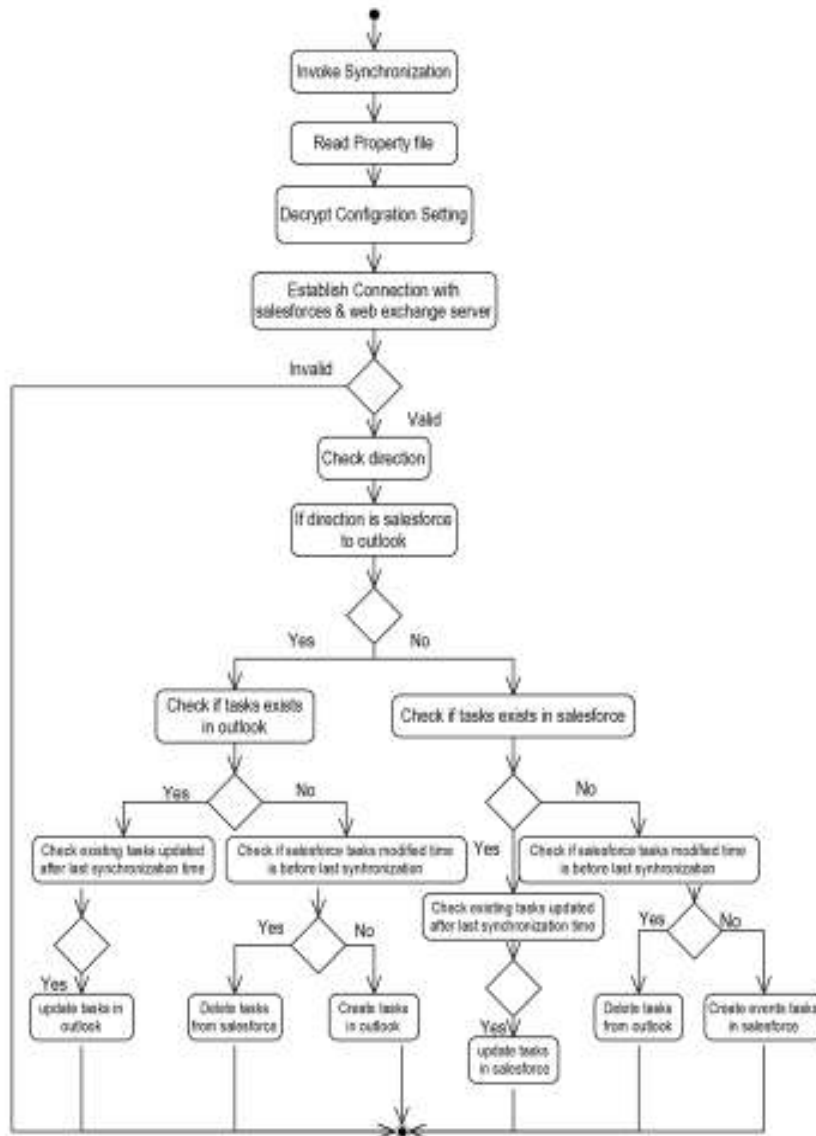


### 3.4 Activity Diagrams

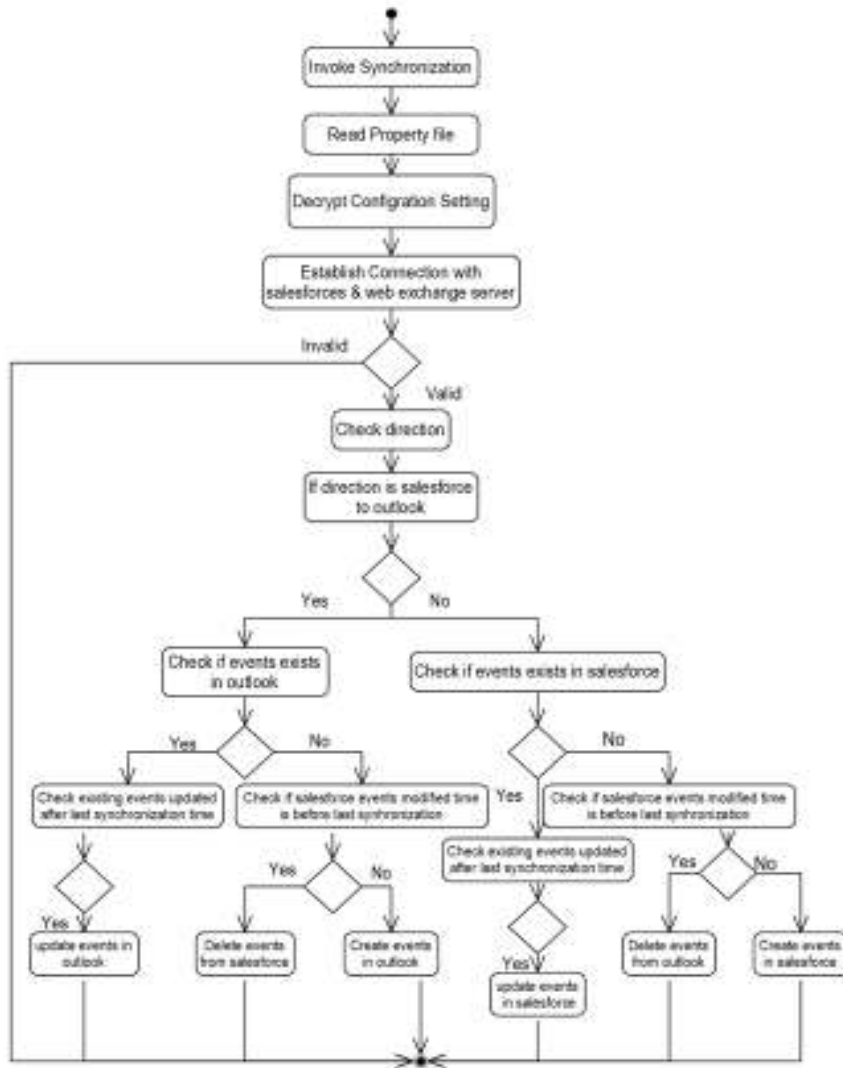
#### 3.4.1 Activity Diagram for Contact Synchronization



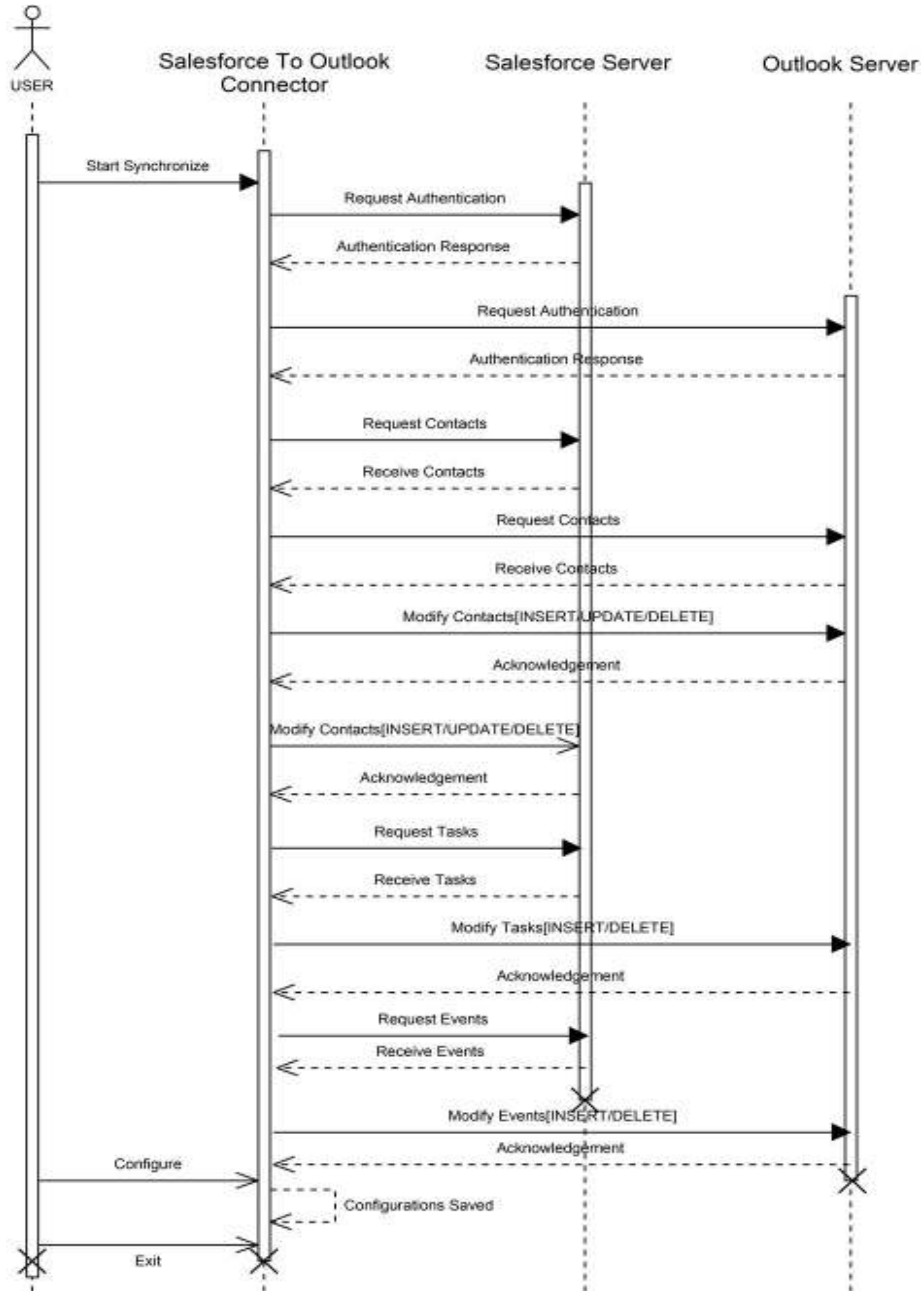
**3.4.2 Activity Diagram for Task Synchronization**



### 3.4.3 Activity Diagram for Event Synchronization

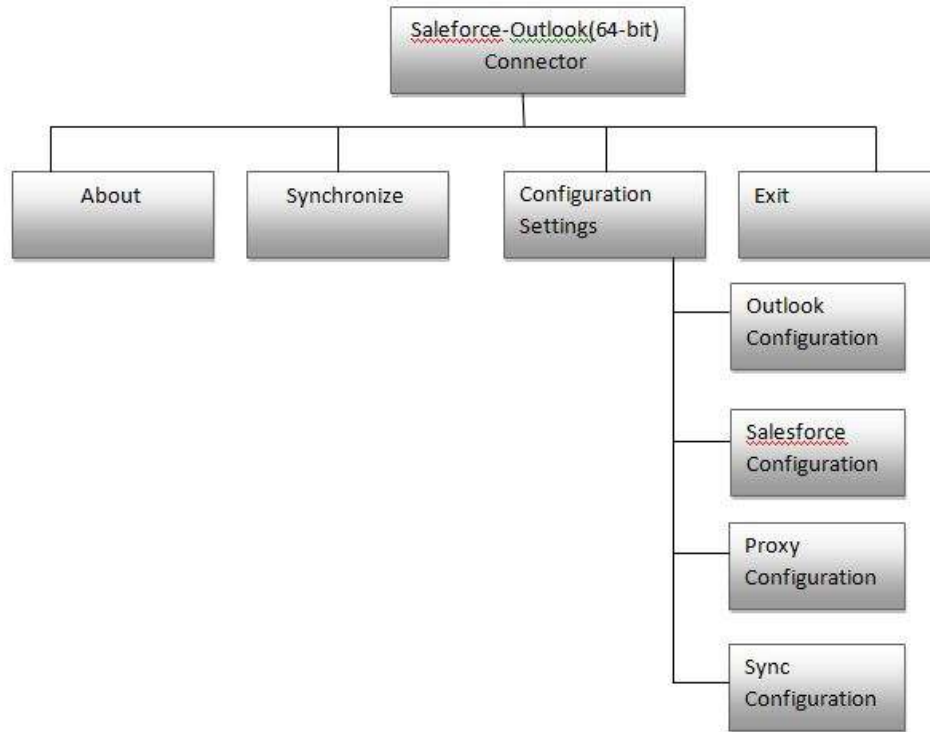


### 3.5 Sequence Diagram

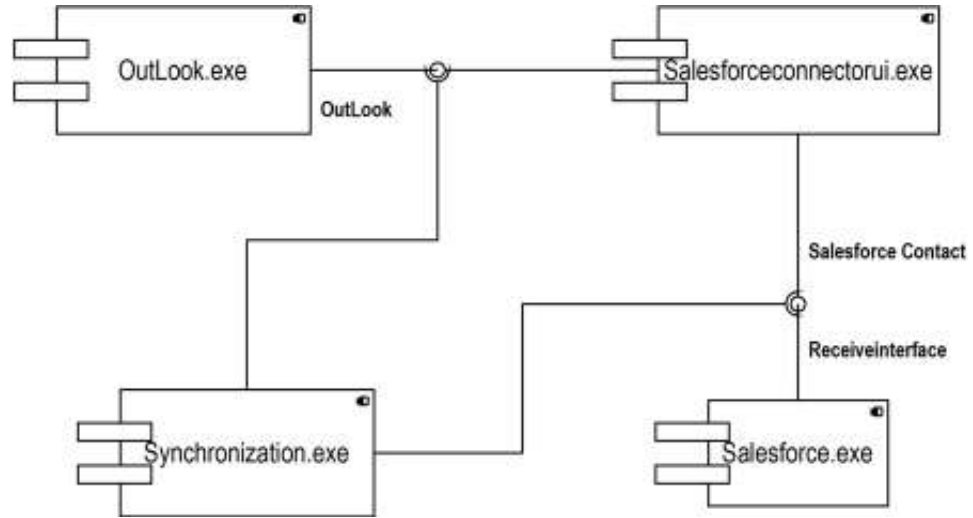




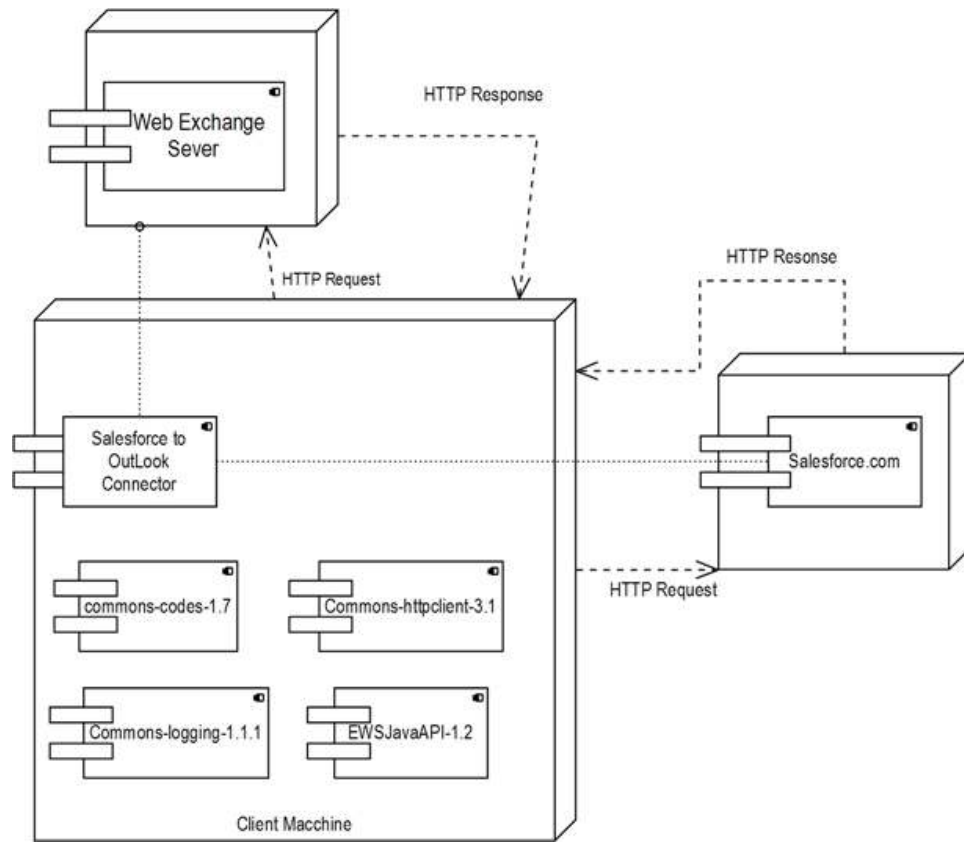
### 3.6 Module Hierarchy



### 3.7 Component Diagram



### 3.8 Deployment Diagram



### **3.9 Module Specification**

**This project contains following Modules:-**

- Connector Frame
- (Application)Java to Outlook
- (Application)Java to Salesforce
- Synchronization
- Encryption-Decryption

#### **Connector Frame**

1. This module deals with designing the Configuration setting windows and the window for providing sync options.
2. The information filled through the interface is stored in the properties files.

### **(Application)Java to Outlook**

- 1 Establish connection with Microsoft web exchange server using the user's credential stored in the properties file.
- 2 Read Contacts from the Microsoft web exchange server.
- 3 Create Contacts on Microsoft web exchange server.
- 4 Read events from Microsoft web exchange server.
- 5 Create events on Microsoft web exchange server
- 6 Read tasks from Microsoft web exchange server.
- 7 Create tasks on Microsoft web exchange server.

### **(Application)Java to Salesforce**

1. Establish connection with Salesforce server using the user's credential stored in the properties file.
2. Read Contacts from the Salesforce server.
3. Create Contacts on Salesforce server.
4. Read events from Salesforce server.
5. Create eventson Salesforce server.
6. Read tasks from Salesforce server.

7. Create tasks on Salesforce server

### **Synchronization**

1. Implements algorithm by using modules (Java to Outlook and Java to Salesforce) when to create a new item, update item or delete item on either sides.
2. This module provides two functionality :-  
Outlook to Salesforce.  
Salesforce to Outlook.

### **Encryption-Decryption**

1. Encrypt the Configuration information before storing it to the properties files.
2. Decrypt the Configuration information when it is retrieved from the properties file for use.

### **3.10 User Interface Design**

#### **Application Icon**



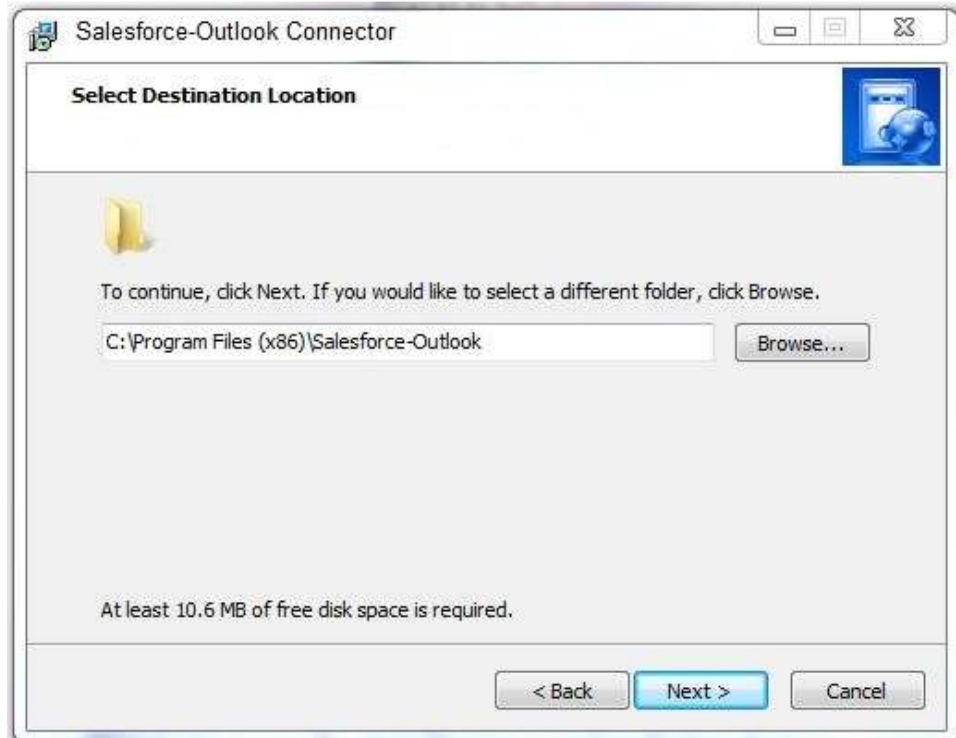
## Installation Steps

### Step 1:

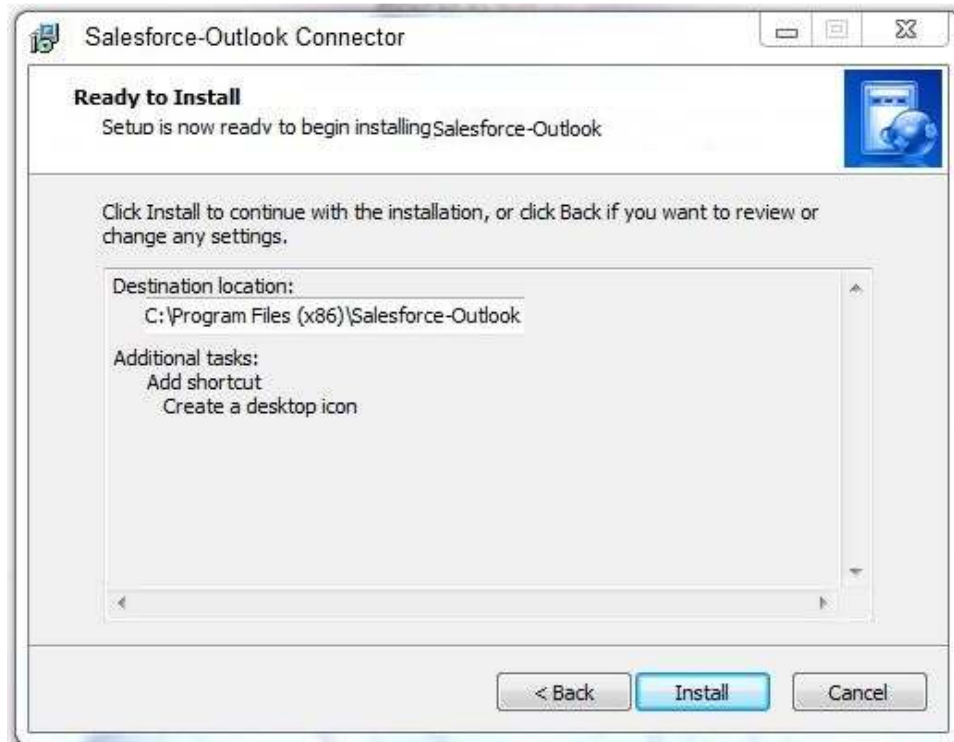




## Step 2:



### Step 3:



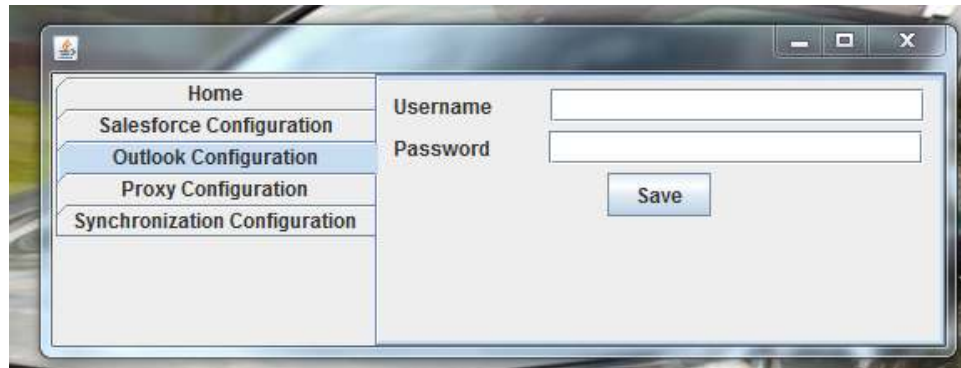
**Step 4:**



### Salesforce Configuration settings



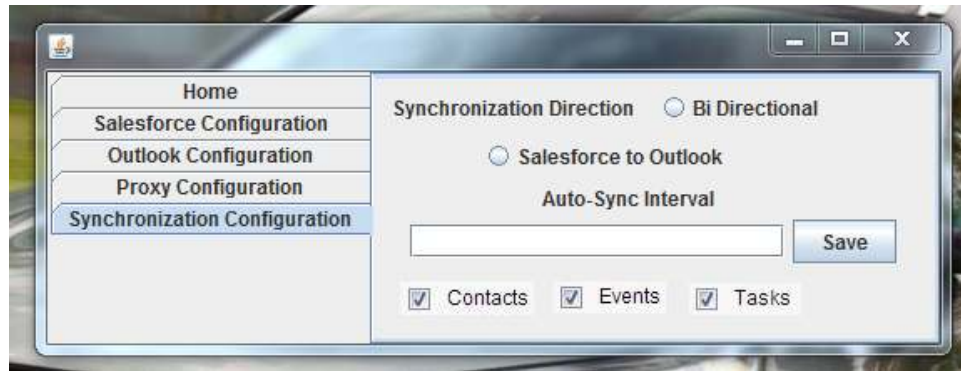
### Outlook Configuration settings



### Proxy Configuration settings



### Synchronize settings



### **3.12 Test Procedures and Implementation**

#### **What is software testing?**

Software testing is a critical element of software quality assurance and represents the ultimate review of specification, design and code generation. It is a process of executing a program with a primary objective of finding errors. Testing gives the guarantee that the software does not fail and runs according to its specifications and in the way the end user expects. This can be done by various software testing techniques which provide a systematic guidance for designing tests that exercise the input and output domains of the program to uncover errors in program function, behavior and performance.

The following software testing techniques were used in order to uncover errors in the system:

- Unit testing
- Integration testing

- White box testing
- Black box testing
- Acceptance tests (Alpha and Beta testing)

### **1: Unit Testing**

Unit testing is normally considered as an adjunct to the coding step. It is the test for the small units of code, e.g. programs, modules or procedures, in order to ensure that they perform their intended functions. All possible paths through the control structure are exercised to ensure that all statements in a program are executed at least once. Unit testing is also done to test the data flow across a module interface.

The following errors are uncovered during unit testing:

- Comparison of different data types.
- Incorrect logical operators or precedence.
- Incorrect comparison of variables.
- Improper or nonexistent loop termination.

- Improperly modified loop variable.

### **2: Integration Testing**

Integration testing is a systematic technique for constructing the program structure while at the same time conducting tests to uncover errors associated with interfacing. During this activity, unit tested components are taken and a program structure is built as per the design. Then incremental integration is performed on the system. This means that programs are constructed and tested in small increments instead of testing the entire program as a whole. This is done because correction of errors becomes difficult in case of whole program testing as many errors were detected and it is not easy to correct them at one go. Thus, through incremental integration testing, any error uncovered could be easily noted and corrected and interfaces are tested completely.

### **3: White Box Testing**

White box testing is also called as glass box testing. It is related with the structure (internal logic) of the program. It helps in uncovering



many errors that black box testing cannot. During white box testing activity, every statement of programs is executed at least once. All independent paths are also executed. Every logical decision is executed to check both true and false conditions. All loops are executed at their boundaries and within their operational bounds. Validation checks are also done during this process.

#### **4: Black Box Testing**

Black box testing, also known as behavioral testing, focuses on the functional requirements of the software. It is related to input and output only and not related with the internal structure of the program. This testing is also done so as to find errors such as:

- Initialization and termination errors
- Behavior and performance errors
- Incorrect or missing functions
- Interface errors
- Errors in data structures and external database access
- Performance errors

### **5: Acceptance Testing (Alpha & Beta Testing)**

An acceptance test is a test carried out by the customer or end user rather than the developer in order to enable the customer to validate all requirements. Alpha testing and beta testing are two types of acceptance tests that are conducted.

### **6: Alpha Testing**

Alpha test is conducted in a controlled environment. As a matter of fact, the end user conducts alpha test at the developer's site. During the course of the system development, the end user is operating the software in front of the developer and the errors and other problems are recorded. Rectification is made accordingly.

### **7: Beta Testing**

Beta testing is also conducted by the end user, but in the absence of the developer. Here, the end user himself records all the problems that he encounters during testing the system and then reports them to

the developer at regular intervals. As a result of problems reported during beta testing, modifications are made to overcome the problem

**Test Cases**

|                       |   |   |                                      |                           |
|-----------------------|---|---|--------------------------------------|---------------------------|
| <b>Test Case ID #</b> |   | 1   |                                      |                           |
| <b>Test Case Name</b> |   | To test functionality of Synchronization from Salesforce to Outlook.  |                                      |                           |
| <b>Prerequisite</b>   |   | In Sync Configuration Direction should be set to salesforce to Outlook and item to Sync should be Contacts. |                                      |                           |
| <b>Objective</b>      |   | To find out bugs in Syncing the Contacts from Salesforce to outlook.  |                                      |                           |
| <b>Sr.No</b>          | <b>Steps to be executed</b>   | <b>Expected Result</b>  | <b>Actual Result</b>                 | <b>Pass/Fail Criteria</b> |
|                       |   |   |                                      |                           |
| 1                     | 1.Salesforce contact without e-mail.<br>2.Outlook contacts do not have same name as Salesforce contact. | It should create new contact in outlook.  | Created contact reflected in outlook | Pass                      |
| 2.                    | 1. Salesforce contact with e-mail.<br>2. Outlook contacts have same                                     | It should create new contact in outlook..   | Created contact reflected in outlook | Pass                      |

## Salesforce-Outlook (64-bit) Connector

---

|    |  |  |   |      |
|----|--|--|---|------|
|    | name as Salesforce contact but do not have e-mail. |  |   |      |
| 3. | 1. Outlook Contact deleted.                        | Corresponding Salesforce contact should be deleted.  | Corresponding Salesforce is deleted.          | Pass |
| 4. | 1. Salesforce Contact is deleted.                  | Corresponding Outlook contact should not be deleted. | Corresponding Outlook contact is not deleted. | Pass |
| 5. | 1. 520 new contacts are synced                     | Contacts should be created in outlook                | Contacts should be created in outlook         | Pass |
| 6. | 1. delete all contacts from outlook and sync       | All salesforce contact should be deleted.            | All salesforce contacts are deleted.          | Pass |

## Salesforce-Outlook (64-bit) Connector

---

| <b>Test Case ID#</b>  |  | 2   |   |                  |
|-----------------------|--|---|---|------------------|
| <b>Test Case Name</b> |  | To test functionality of Synchronization Bi-directional i.eSalesforce to Outlook and outlook to salesforce. |   |                  |
| <b>Prerequisite</b>   |  | In Sync Configuration Direction should be set to Bi-Directional and item to Sync should be Contacts.        |   |                  |
| <b>Objective</b>      |  | To find out bugs in Syncing the Contacts Bidirectional  |   |                  |
| <b>Sr. No</b>         | <b>Steps to executed</b>                           | <b>Expected Result</b>  | <b>Actual Result</b>  | <b>Pass/Fail</b> |
|                       | <b>First name Textbox Test cases</b>               |   |   |                  |
| 1.                    | 1.create a new contact in outlook without surname. | It should display message “Contact could not sync due to missing surname”                                   | It displays message “Contact could not sync due to missing surname” | Pass             |
| 2.                    | 1. create a blank new contact in outlook           | Contact should not be synced to salesforce.   | Contact not synced to salesforce.                                   | Pass             |
| 3.                    | 1.delete contact from Salesforce                   | Corresponding outlook contact should not be deleted.  | Corresponding outlook contact is not deleted from outlook contact.  | Pass             |
| 4.                    | 1.Update   | Contacts should   | Updated   | Pass             |

## Salesforce-Outlook (64-bit) Connector

---

|  |   |                       |                                    |  |
|--|---|-----------------------|------------------------------------|--|
|  | contact in salesforce.<br>2.Update other contact in outlook | be updated both sides | contacts are reflected both Sides. |  |
|--|---|-----------------------|------------------------------------|--|

**CHAPTER 4:**  
**USER MANUAL**



#### **4.1 User Manual**

The User Manual is Prepared reflexively because it is an item that must accompany every system. Manual is given so that there is quick reference about the system package.

This Manual will help user to Navigate through the Application and help user out whenever there is any trouble while using application.

User manual provides user with the ease of operating the system. If intended user of the system feels any problem in operating the system, they can refer to this manual to clear their doubts. This user manual guides the users of the system and provides them the briefing of the system to how to use the system.

**Setup of Salesforce-Outlook(64-bit) Connector**



This is Setup of the Application.

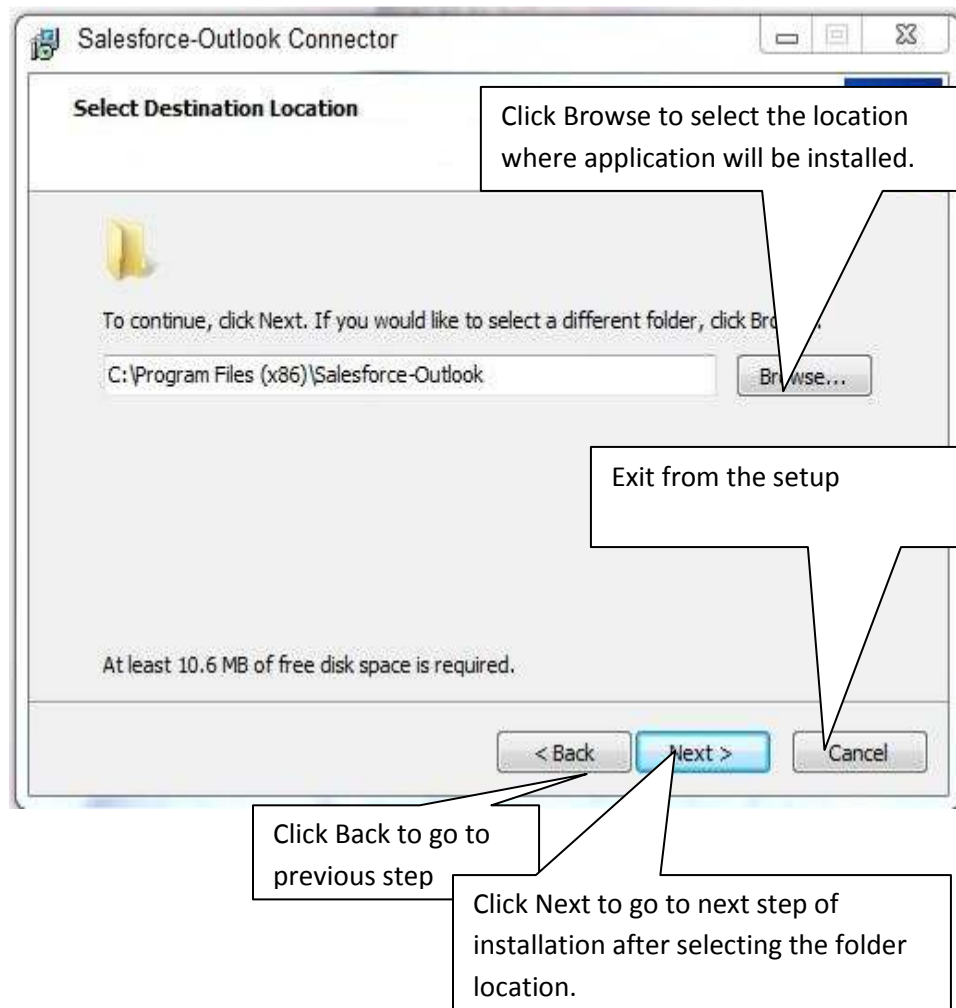
Double Click on the setup to install the application.

## Installation steps

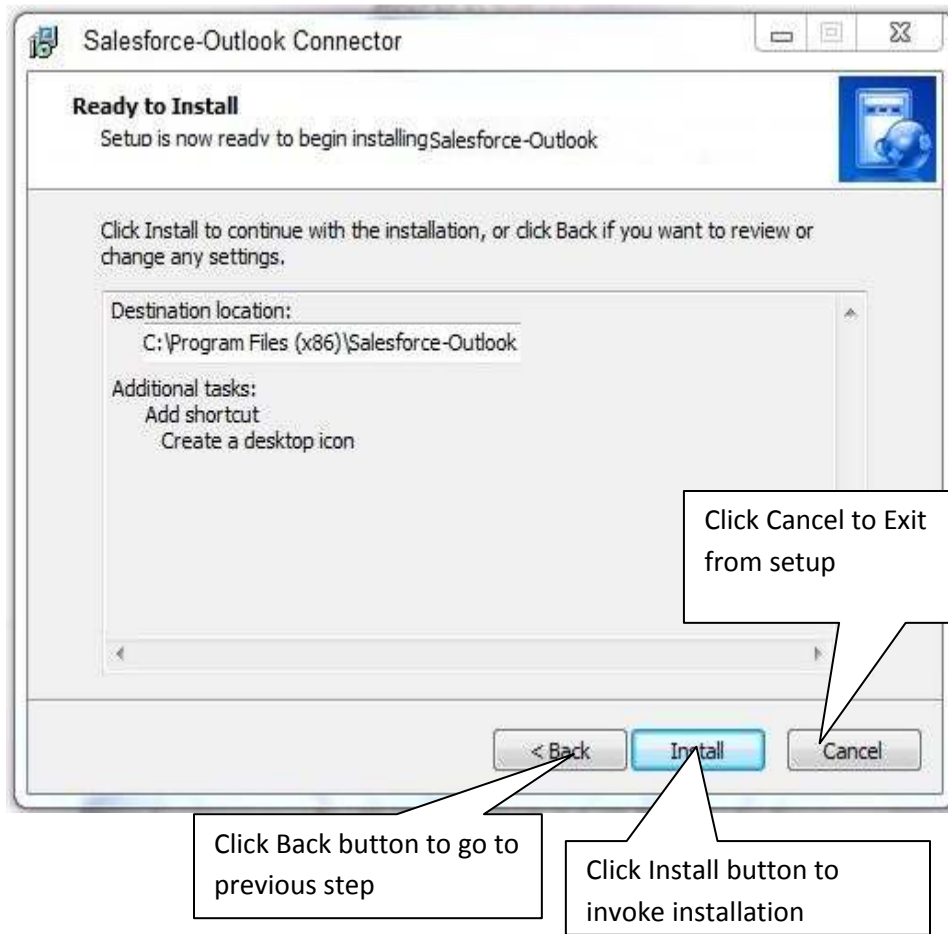
General Information before installation starts



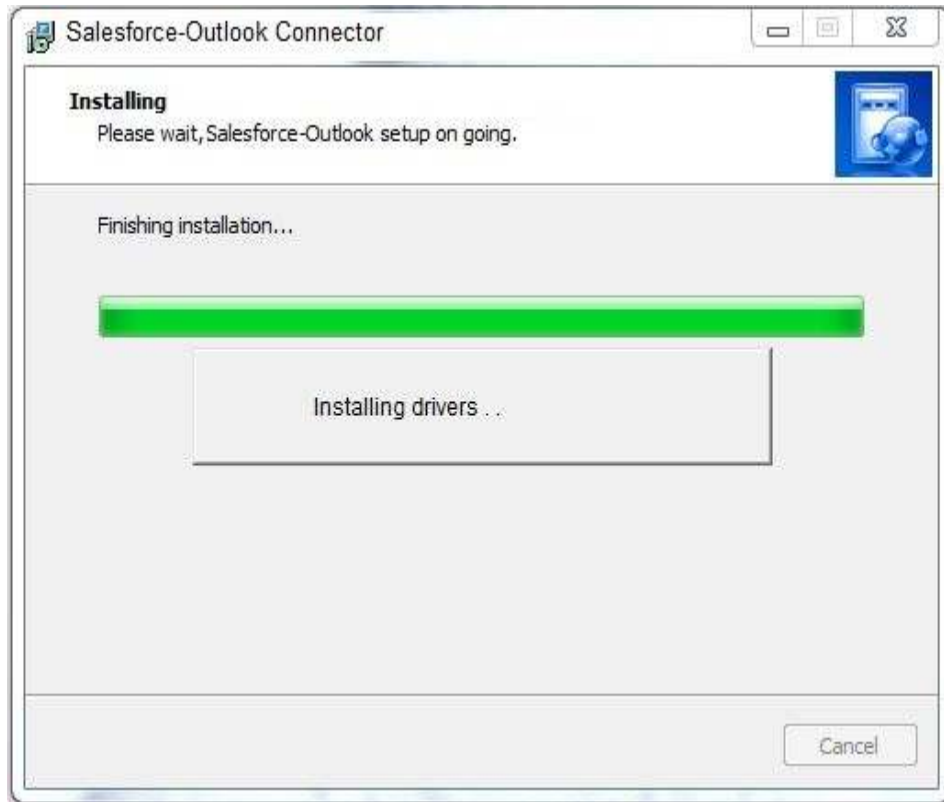
Select the location for installing the Application:



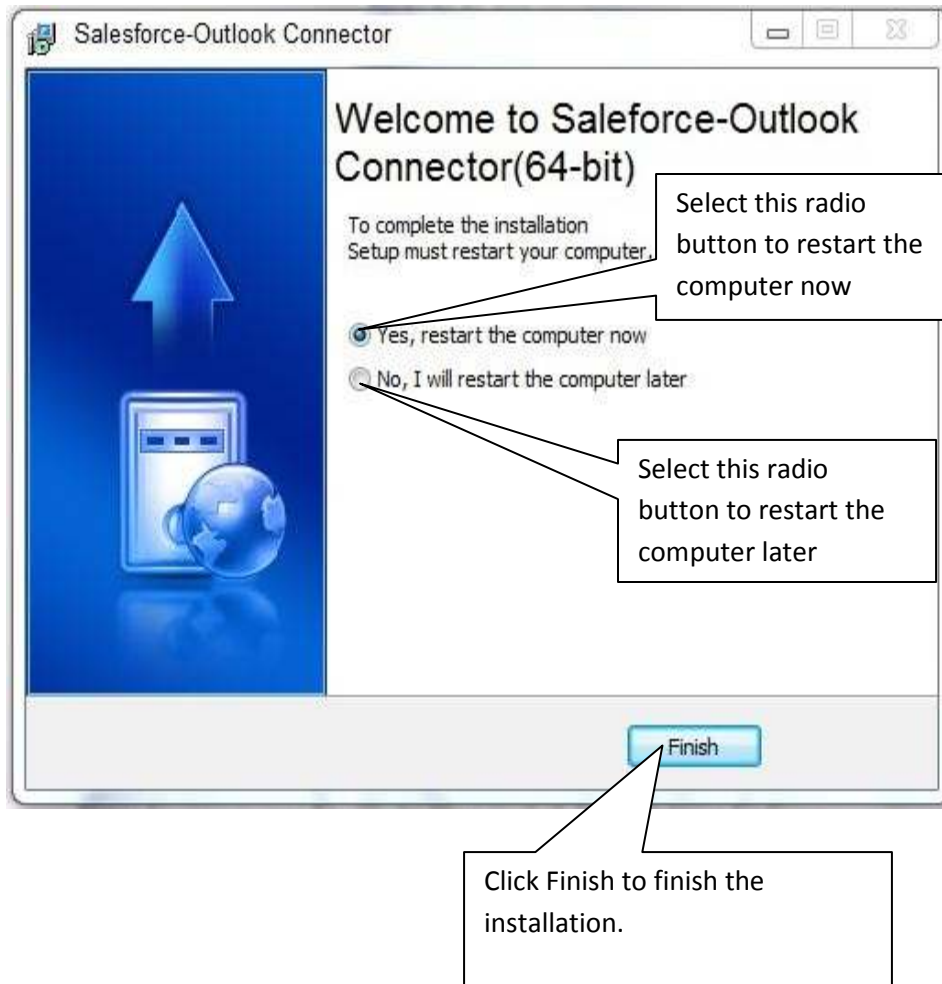
## Installation confirmation



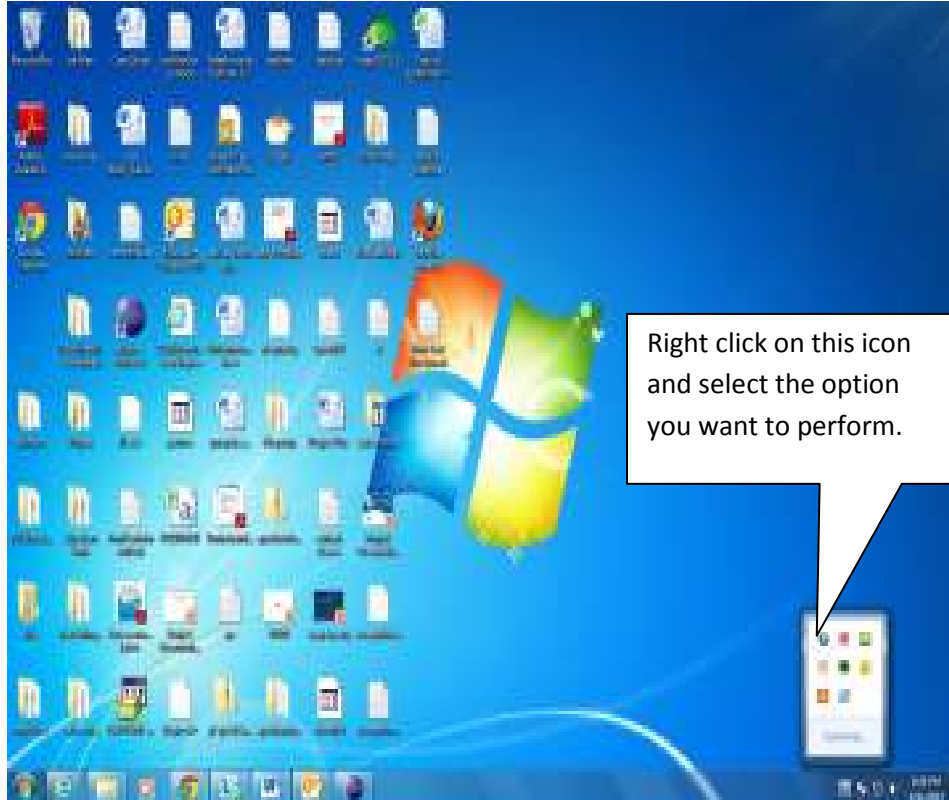
**Showing installation progress**



**Asking user to select the computer restart and to finish the installation**

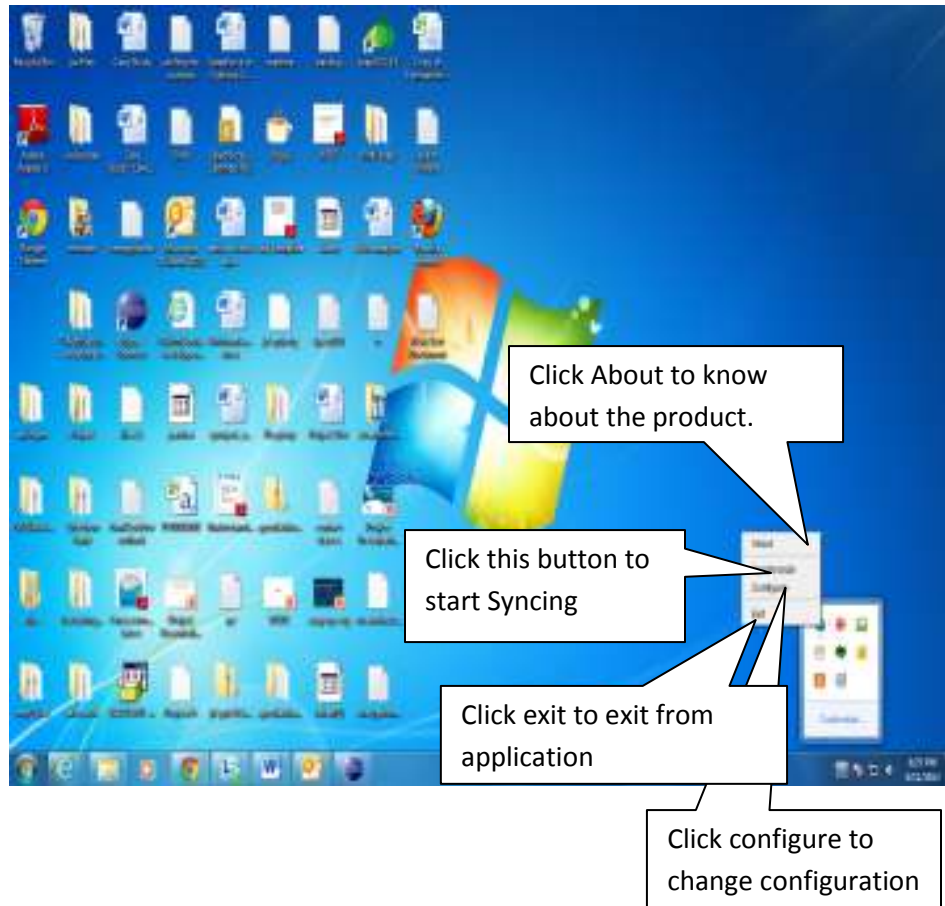


### Execute the Application

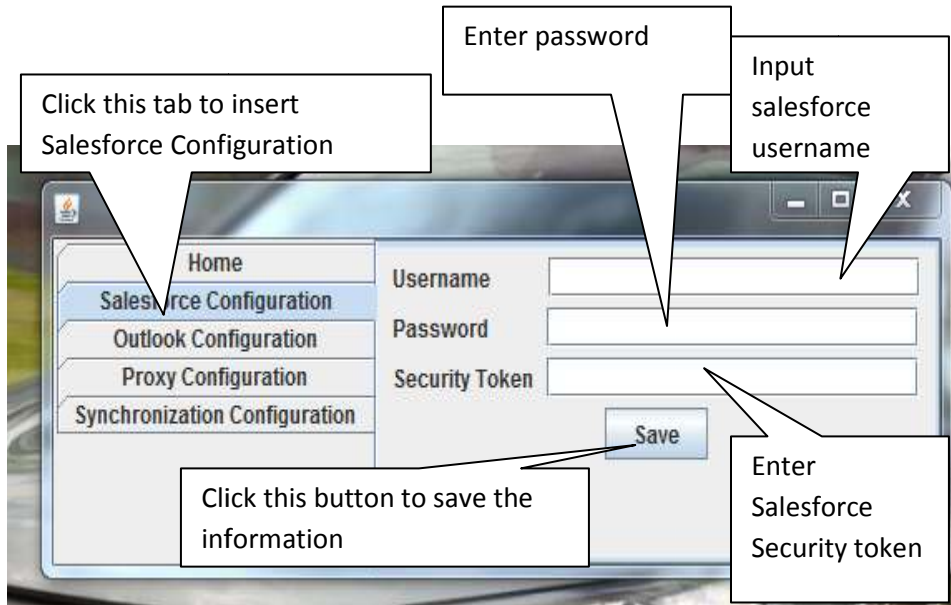
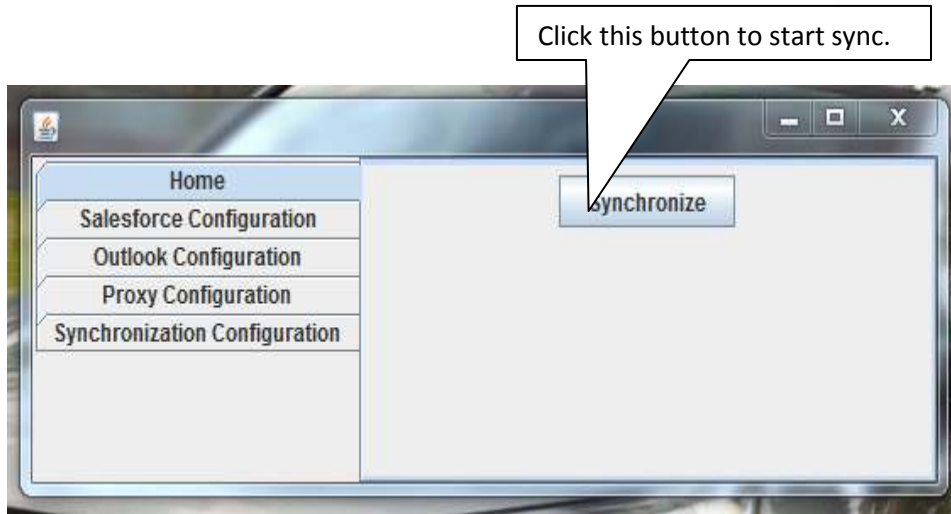


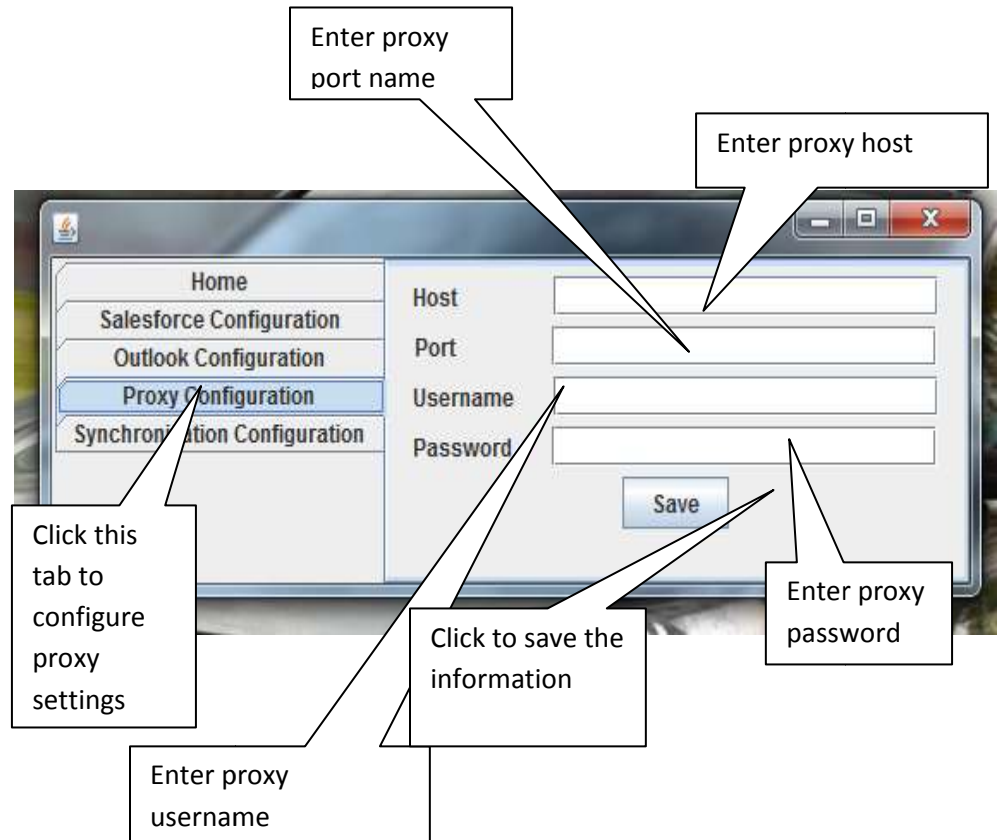
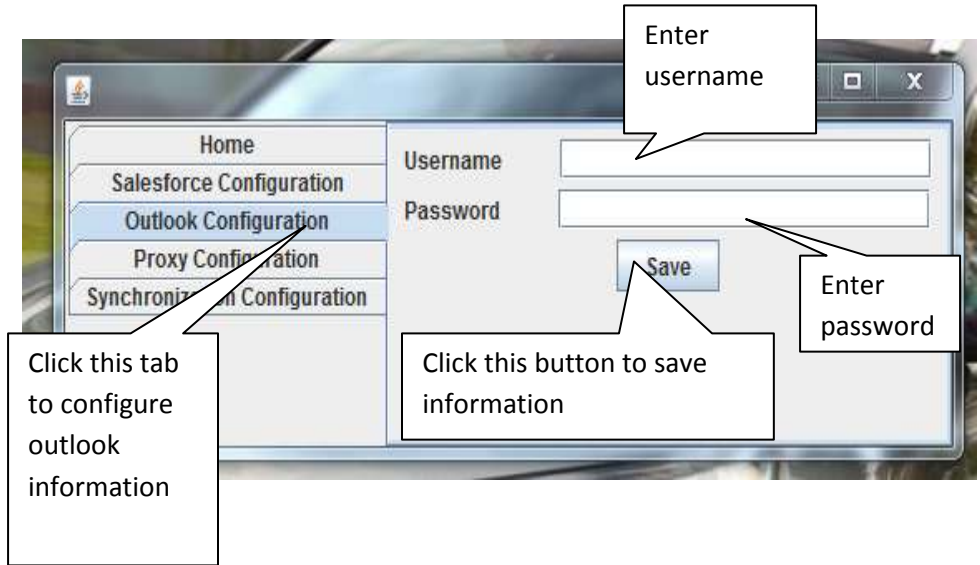


**Options provided by the application**



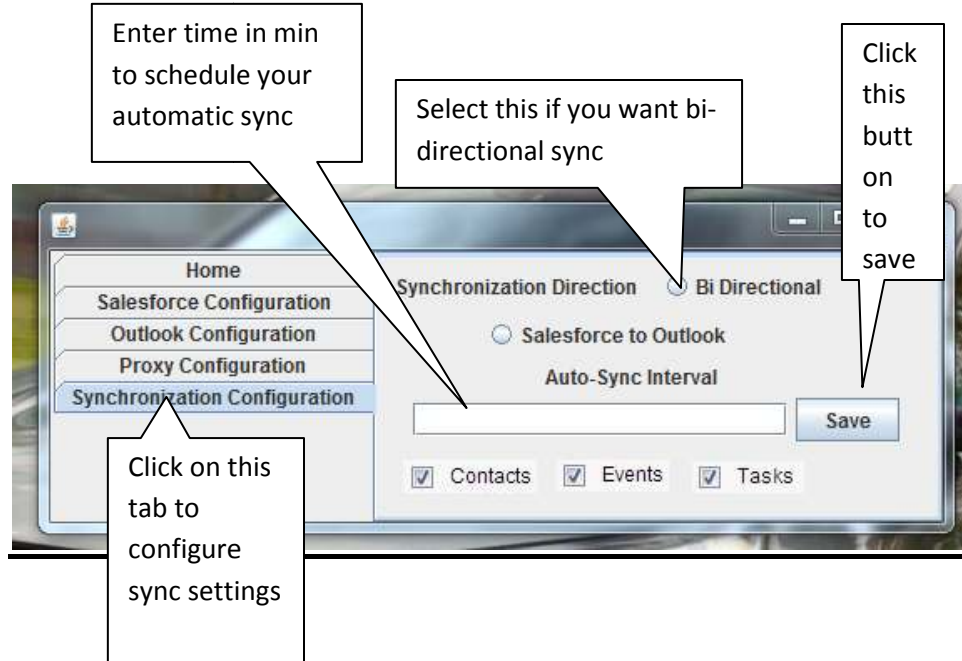
**Option provided by clicking on Configuration**





## Salesforce-Outlook (64-bit) Connector

---



## **4.2 Operation Manual/Menu Explanation**

Menu screen provide user with ease of access to various part of the application.

Menu explanation describes the various menus used in the system.

Different menus

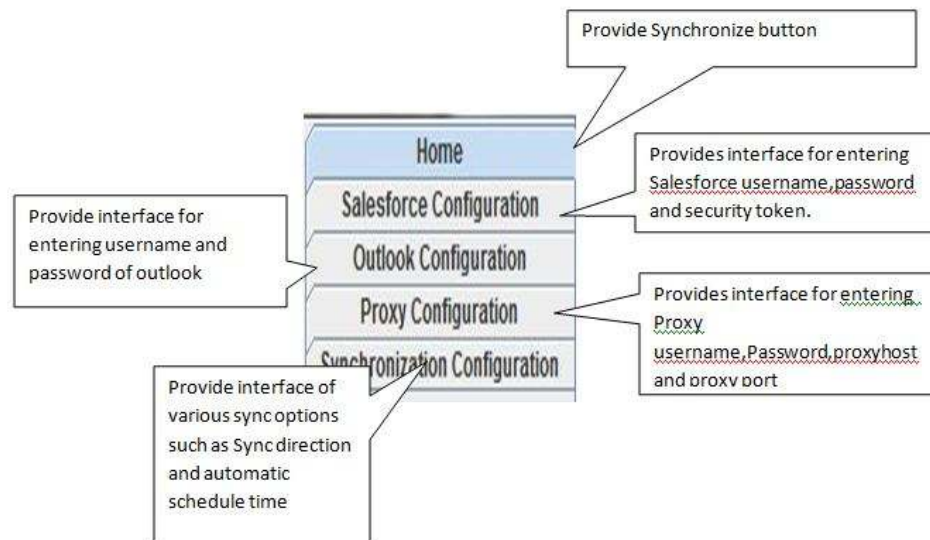
are as follows.



Based on the selection of above menu there are more menus with different options.

Menu After Clicking on Configure button.

## Menu for Configure button

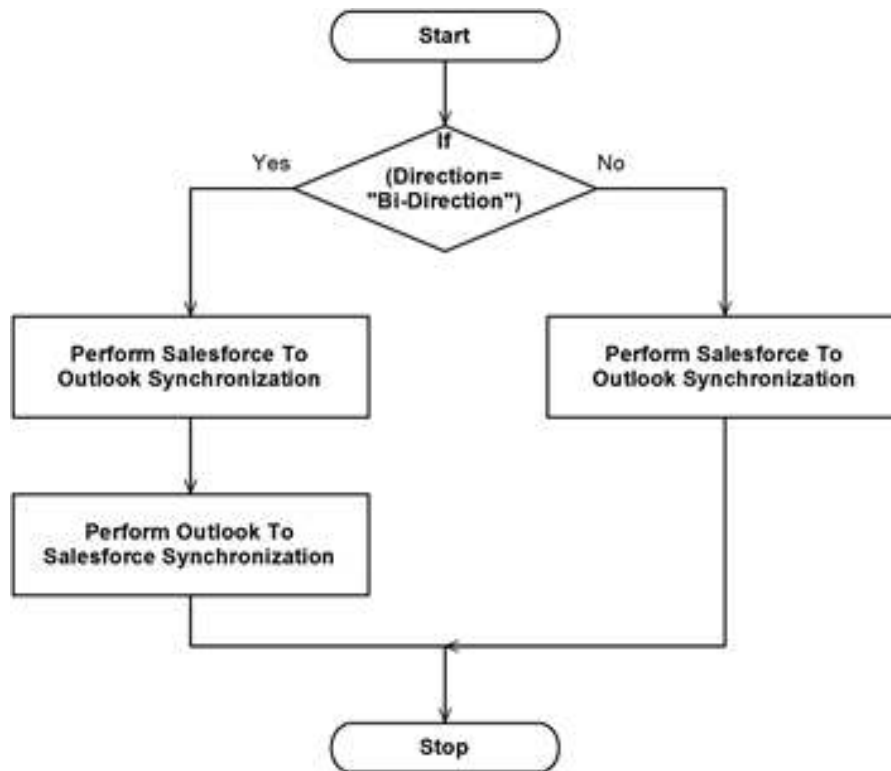


### **4.3 Program Specification/Flow Chart**

The system is developed in JAVA .

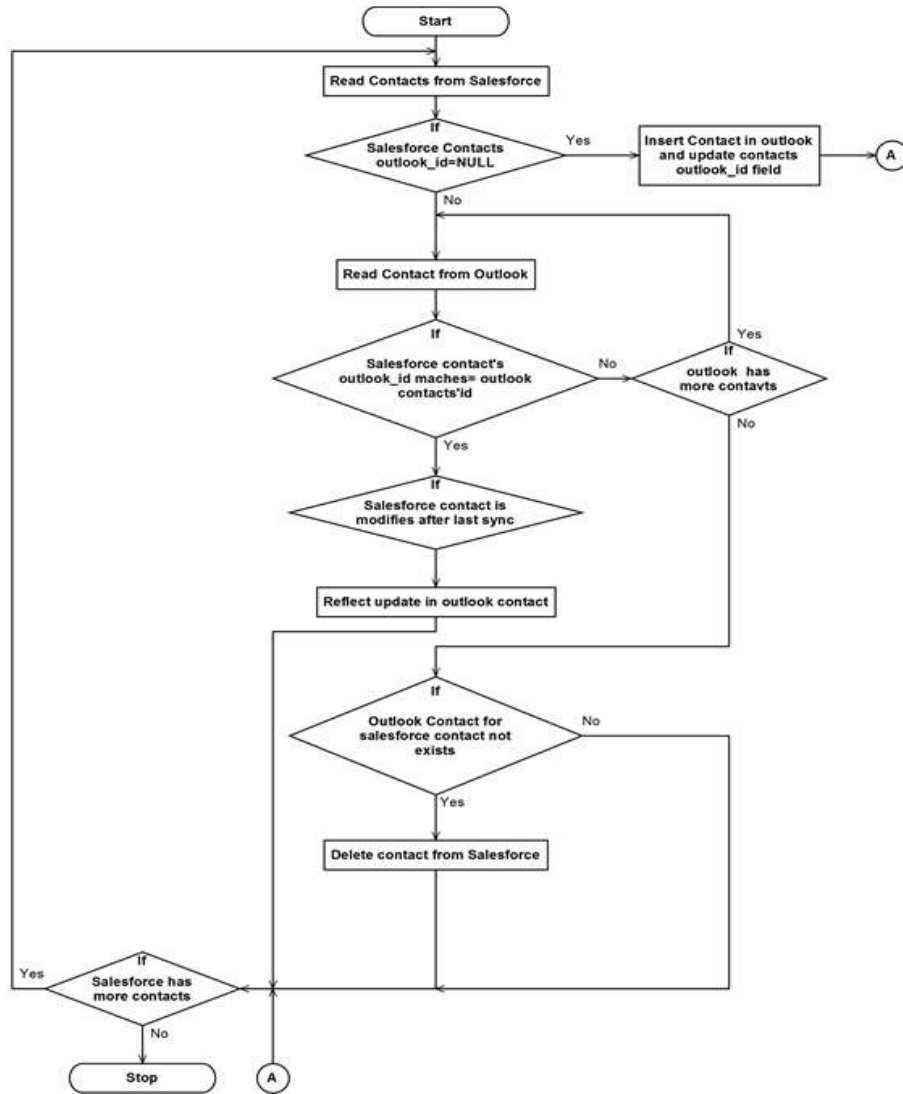
Following are the program specification used in the development process explained with the help of flowcharts.

#### **4.3.1 Main Flowchart**

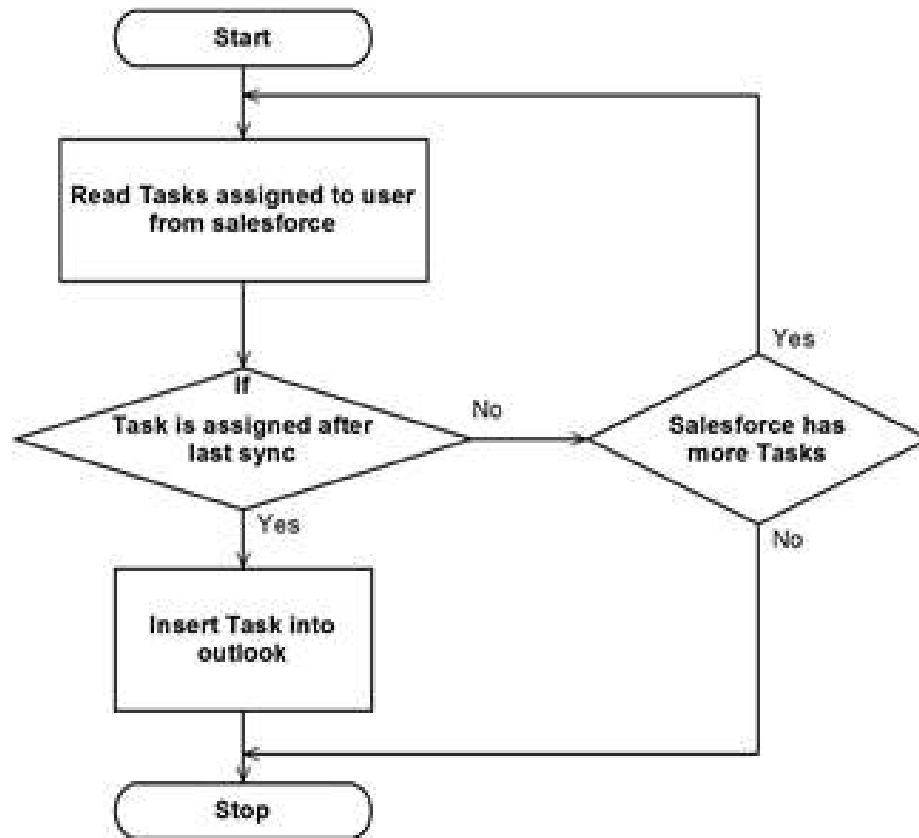




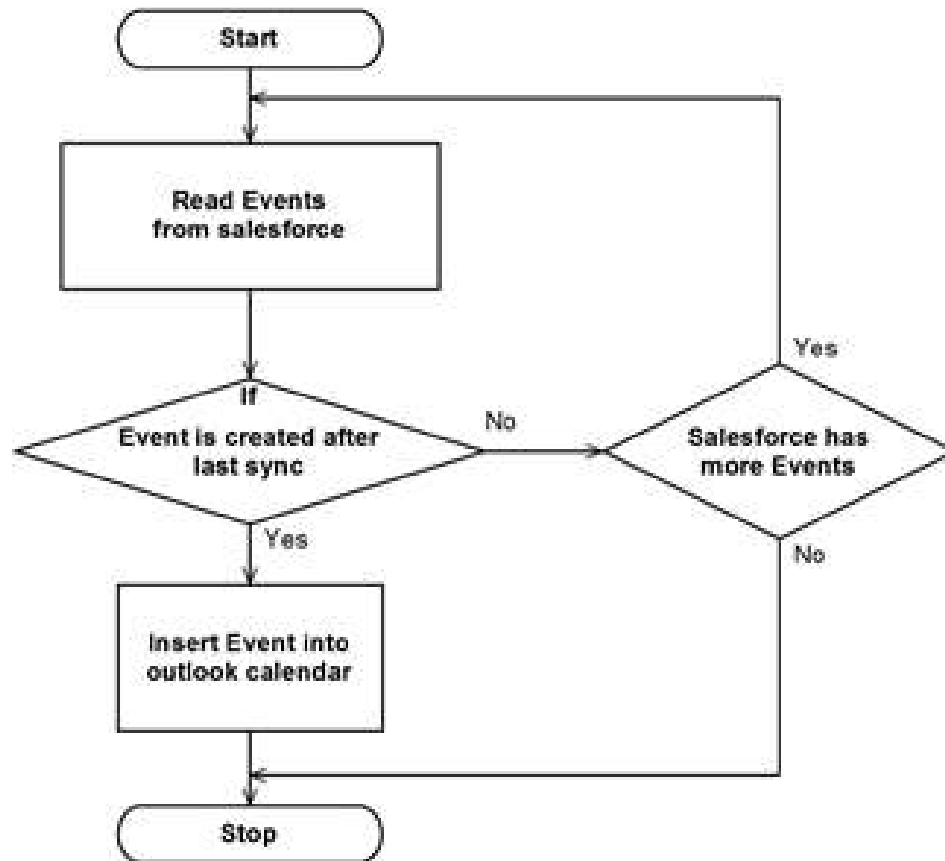
### 4.3.2 Flowchart for Salesforce To Outlook Contact Synchronization



### 4.3.3 Salesforce To Outlook Task Synchronization

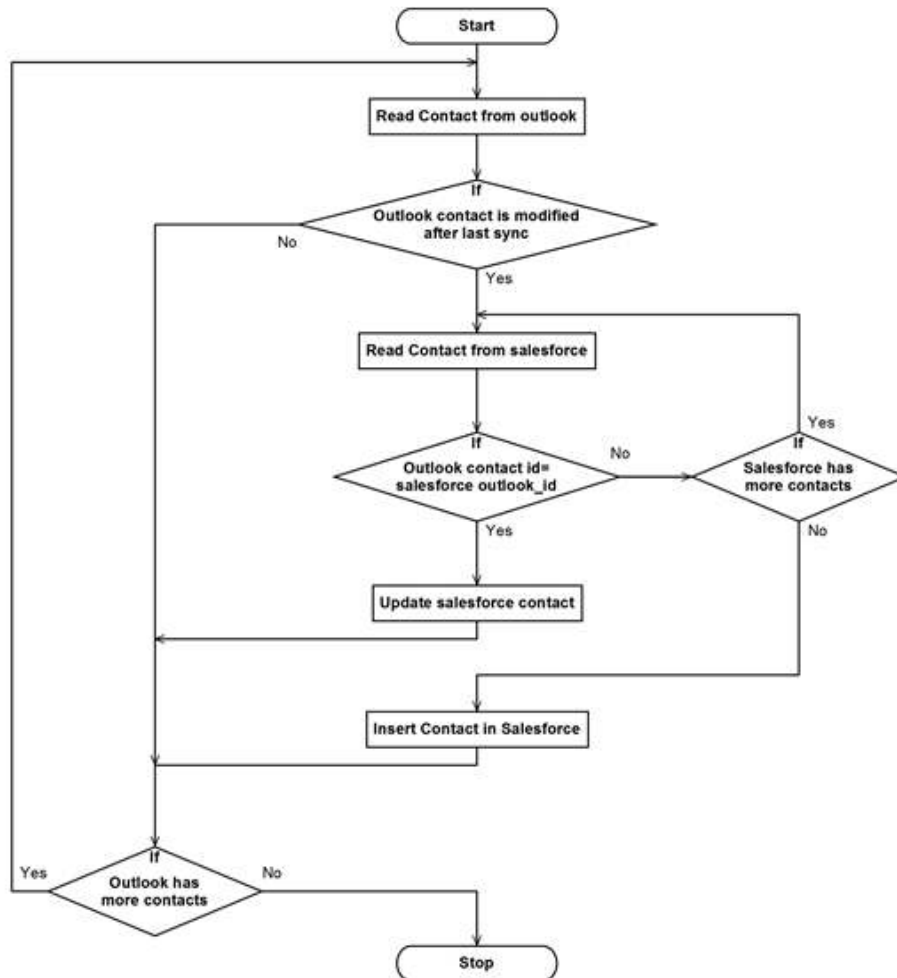


#### 4.3.4 SalesforceTo Outlook Event Synchronization



### 4.3.5 Outlook to salesforce Synchronization

(Note : In bi-directional synchronization, first salesforce to outlook synchronization is done and then outlook to salesforce synchronization is completed)



## Field Mapping for Synchronization

### Field mapping for contact item

| OUTLOOK             | SLAESFORCE           |
|---------------------|----------------------|
| Given Name          | First Name           |
| SurName             | Last Name (Not Null) |
| Department          | Department           |
| JobTitle            | Title                |
| Business Fax        | Fax                  |
| Business Home Phone | Home Phone           |
| Business Mobile     | Mobile               |
| Business street     | Mailing street       |
| Business city       | Mailing city         |
| Business state      | Mailing State        |
| Business Country    | Mailing country      |
| Business Postal pin | Mailing pin          |
| Email1              | Email                |

### Field mapping for Task

| OUTLOOK        | SLAESFORCE     |
|----------------|----------------|
| Subject        | Subject        |
| <u>DueDate</u> | <u>DueDate</u> |
| Priority       | Priority       |
| Status         | Status         |

### Field Mapping for Event

| OUTLOOK          | SLAESFORCE       |
|------------------|------------------|
| Subject          | Subject          |
| <u>StartDate</u> | <u>StartDate</u> |
| <u>EndDate</u>   | <u>EndDate</u>   |

**DRAWBACKS**  
**&**  
**LIMITATIONS**

### **Drawbacks & Limitations**

- a) All fields are not mapped between salesforce and outlook account.
- b) 2000contacts can be inserted at once in salesforce account.
- c) Blank Contact of Outlook can't be synced to the salesforce.
- d) User's userid can't be modified once inserted.
- e) Panel is not attached to the outlook.



**PROPOSED  
ENHANCEMENT**

## **Proposed Enhancement**

User requirements keep changing as the system is being used.

Some of the future enhancements that can be done to this system are:-

- a) All fields will be mapped between salesforce and outlook account.
- b) Panel will be attached to the outlook.

# **CONCLUSIONS**

## **Conclusions**

All the requirements stated by the company have been addressed in this application.

As we know that, any project even on completion requires constant improvement and changes which gives way for release of new version. We made this application user friendly.

For developing this application we used technologies such as Java, Exchange web services , SOAP which are widely used now-a-days.

# **BIBLIOGRAPHY**

## **Bibliography**

### **Books:**

The Complete Reference java.

Salesforce Developer Guide

### **URLs:**

[www.sun.java.com](http://www.sun.java.com)

[www.google.com](http://www.google.com)

[www.help.salesforce.com](http://www.help.salesforce.com)

# **ANNEXURES**

## **ANNEXURE 1:User Interface Screens**

### **Application Set up Icon**



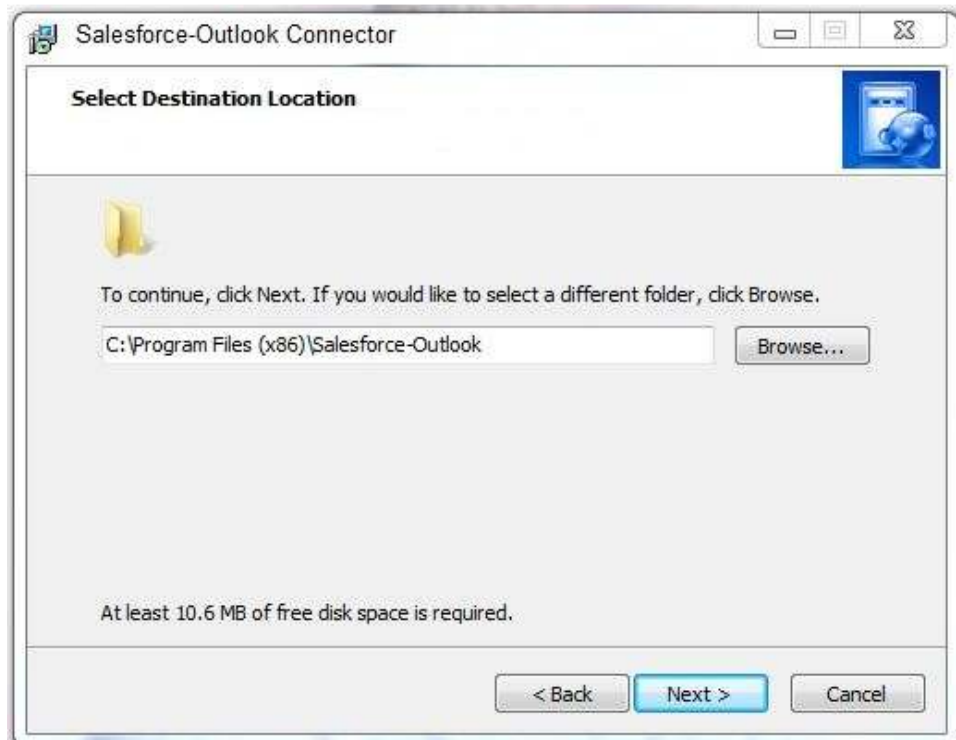


## Installation Steps

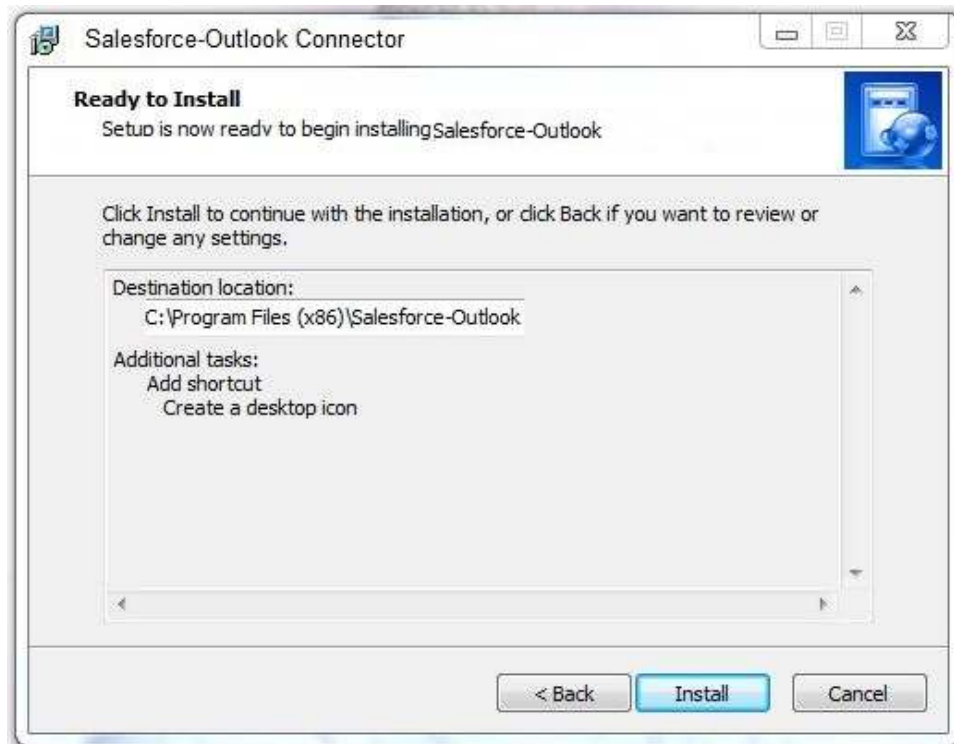
### Step – 1



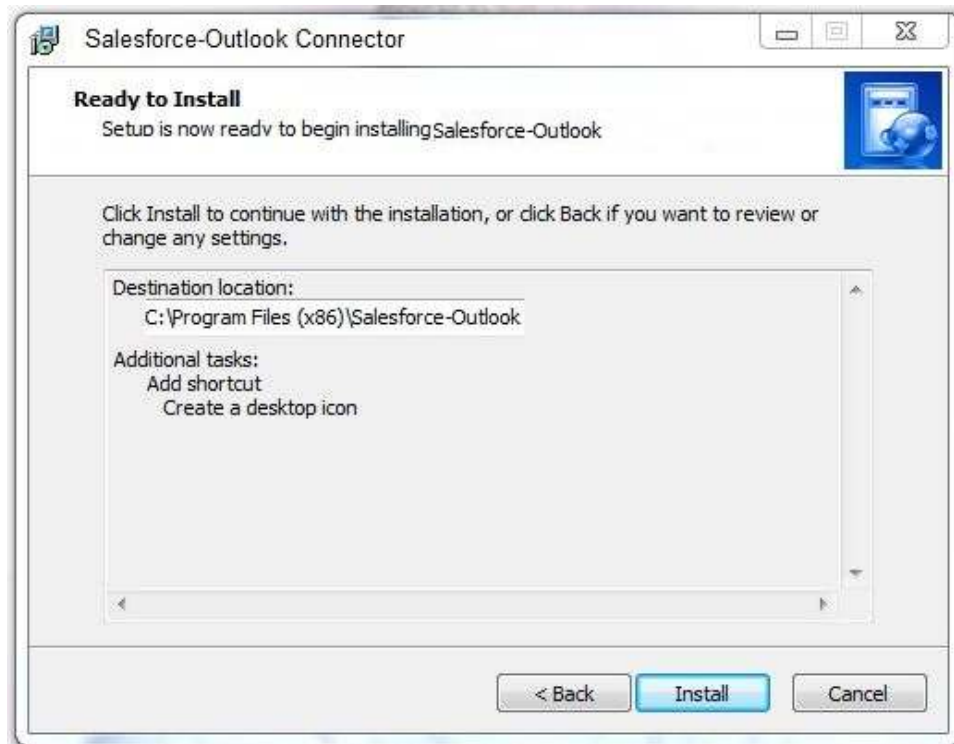
## Step-2



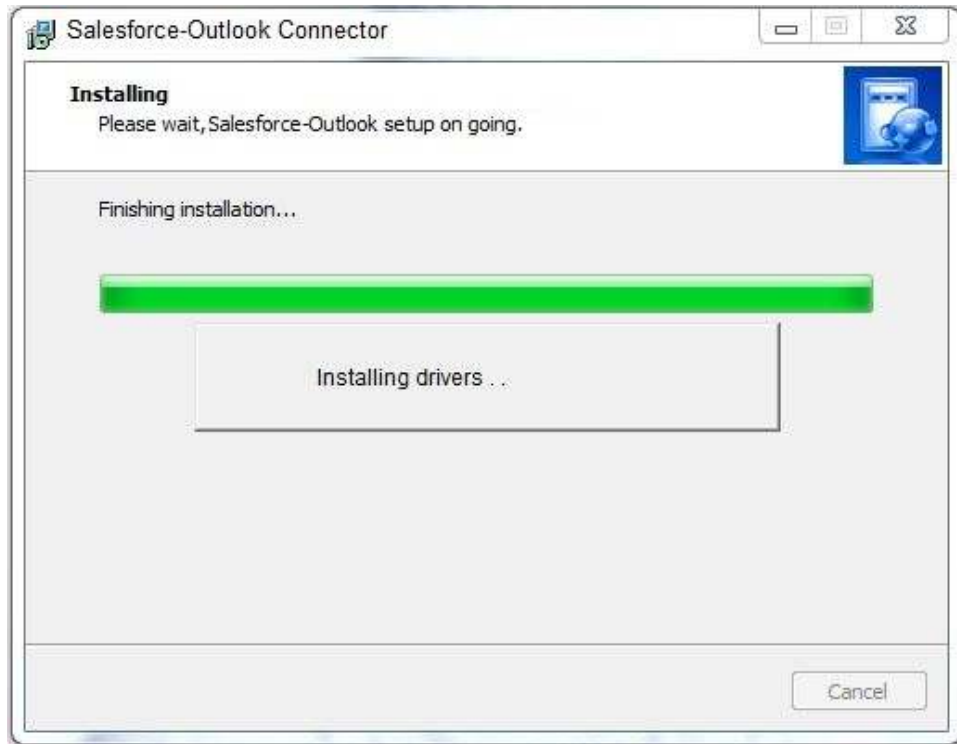
### Step-3



## Step-4



## Step-5



## Step-6



## Configuration settings input screens

### Input Screen for Salesforce Configuration



A screenshot of a web application window titled "Salesforce Configuration". The window has a sidebar on the left with a menu containing "Home", "Salesforce Configuration", "Outlook Configuration", "Proxy Configuration", and "Synchronization Configuration". The "Salesforce Configuration" option is selected and highlighted in blue. The main content area contains three input fields: "Username" with the value "shubhankar22@gmail.com", "Password" with masked characters "••••••••", and "Security Token" with the value "FgYIsN59zPcqxPLIubbsyjDE4". A "Save" button is located below the input fields.

|                               |                |                           |
|-------------------------------|----------------|---------------------------|
| Home                          | Username       | shubhankar22@gmail.com    |
| Salesforce Configuration      | Password       | ••••••••                  |
| Outlook Configuration         | Security Token | FgYIsN59zPcqxPLIubbsyjDE4 |
| Proxy Configuration           |                |                           |
| Synchronization Configuration |                |                           |

Save

### Input Screen for Outlook Configuration

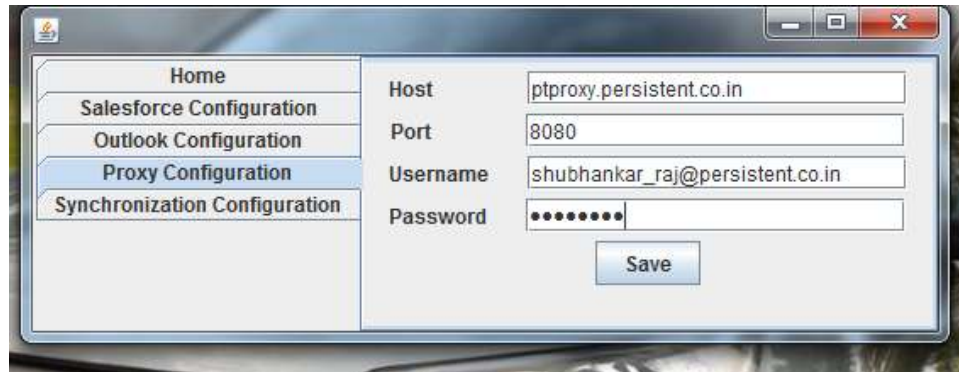


A screenshot of a web application window titled "Outlook Configuration". The window has a sidebar on the left with a menu containing "Home", "Salesforce Configuration", "Outlook Configuration", "Proxy Configuration", and "Synchronization Configuration". The "Outlook Configuration" option is selected and highlighted in blue. The main content area contains two input fields: "Username" with the value "shubhanakar\_raj@persistent.co.in" and "Password" with masked characters "••••••••". A "Save" button is located below the input fields.

|                               |          |                                  |
|-------------------------------|----------|----------------------------------|
| Home                          | Username | shubhanakar_raj@persistent.co.in |
| Salesforce Configuration      | Password | ••••~•••                         |
| Outlook Configuration         |          |                                  |
| Proxy Configuration           |          |                                  |
| Synchronization Configuration |          |                                  |

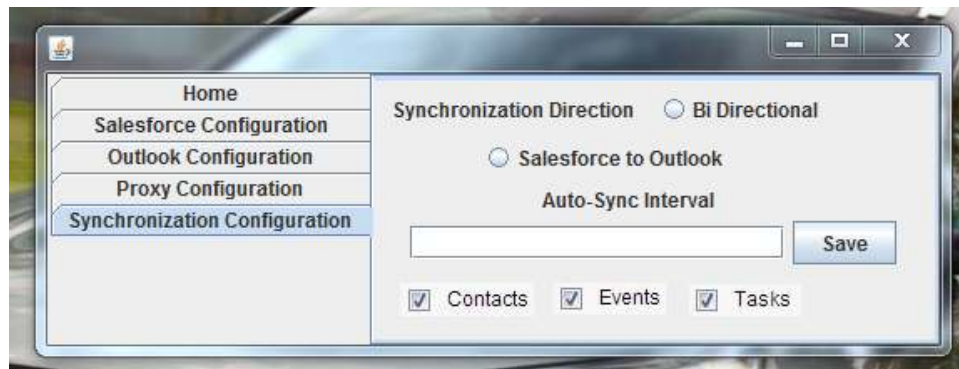
Save

## Input Screen for Proxy Configuration



A screenshot of a software window titled "Proxy Configuration". On the left, there is a vertical menu with five items: "Home", "Salesforce Configuration", "Outlook Configuration", "Proxy Configuration" (which is highlighted in blue), and "Synchronization Configuration". The main area of the window contains four input fields: "Host" with the value "ptproxy.persistent.co.in", "Port" with the value "8080", "Username" with the value "shubhankar\_raj@persistent.co.in", and "Password" with a masked field of ten dots. A "Save" button is located at the bottom right of the main area.

## Input Screen for Sync Configuration



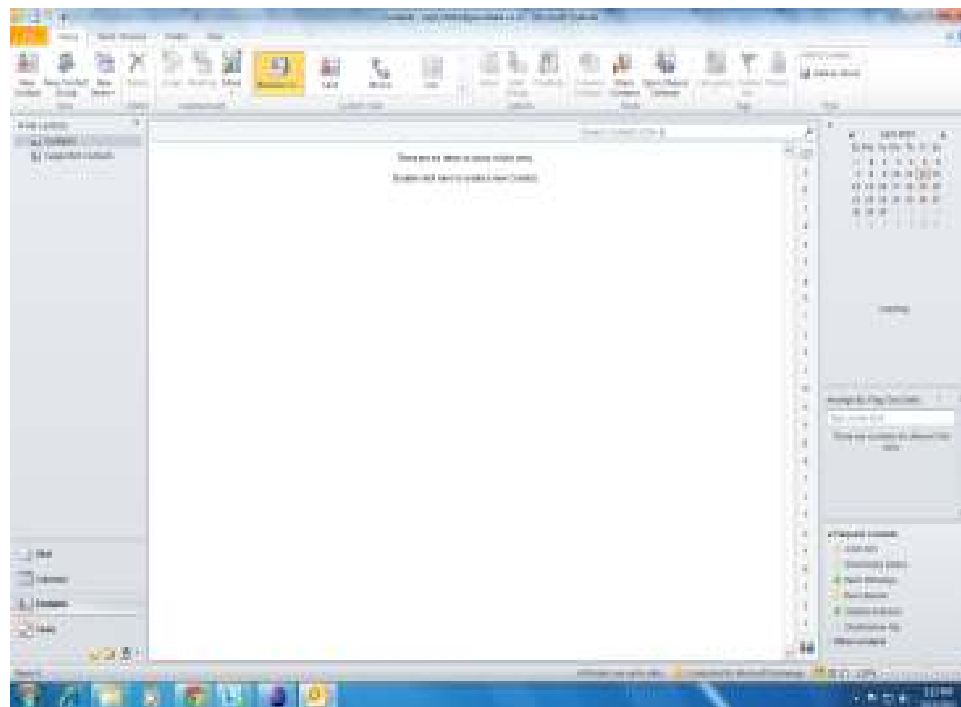
A screenshot of a software window titled "Sync Configuration". On the left, there is a vertical menu with five items: "Home", "Salesforce Configuration", "Outlook Configuration", "Proxy Configuration", and "Synchronization Configuration" (which is highlighted in blue). The main area of the window contains the following configuration options: "Synchronization Direction" with two radio buttons, "Bi Directional" (selected) and "Salesforce to Outlook"; "Auto-Sync Interval" with an empty input field; and three checked checkboxes: "Contacts", "Events", and "Tasks". A "Save" button is located at the bottom right of the main area.



## **ANNEXURE 2: Output Report with Data**

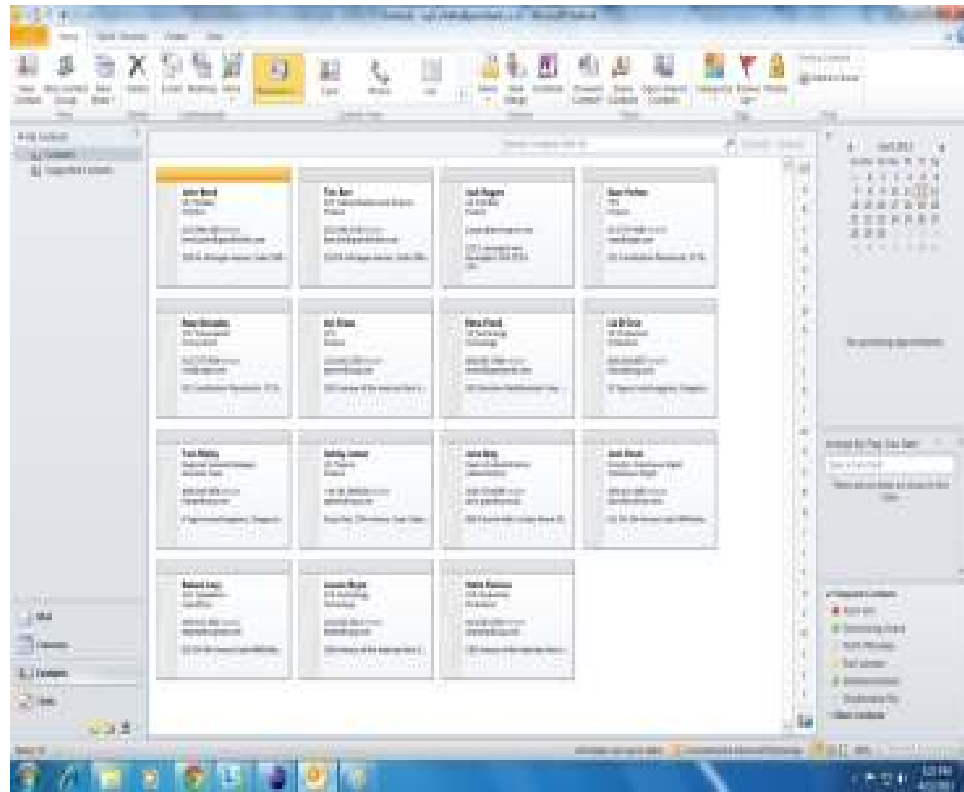
### **Contacts Synchronization**

**Outlook without any Contact (Before Synchronization)**

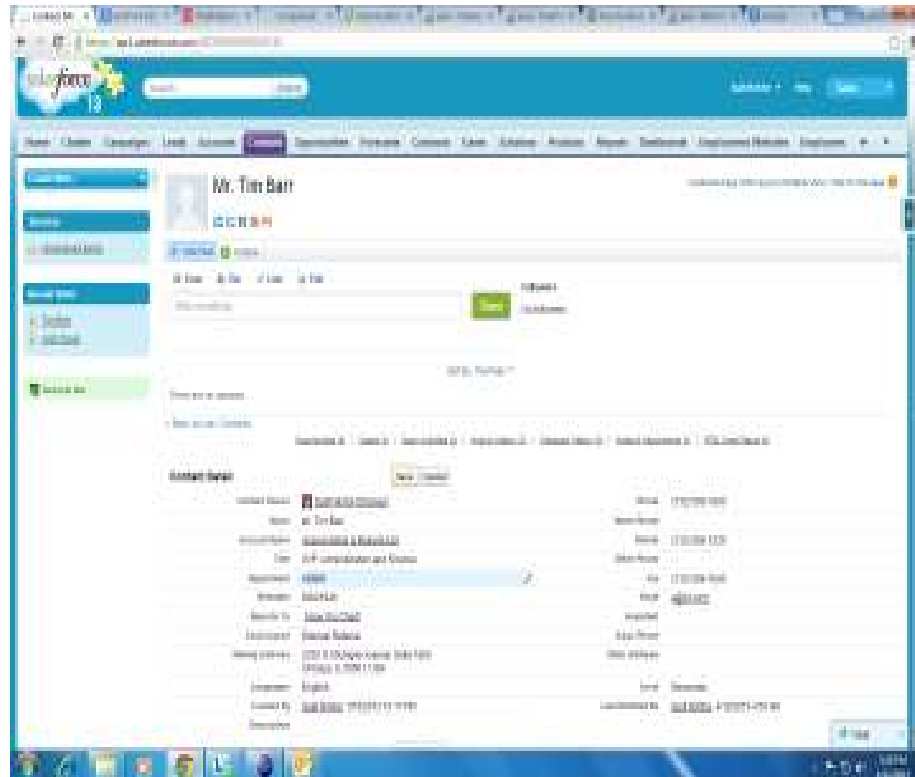




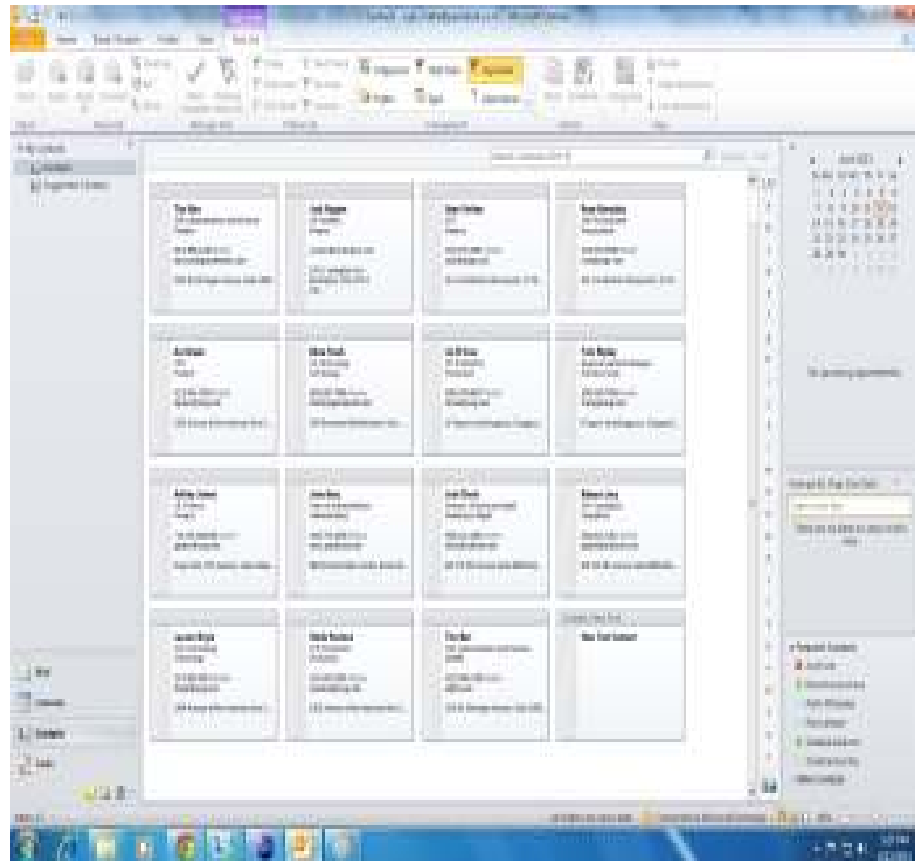
## Outlook Account after Synchronizing from slaesforce to outlook



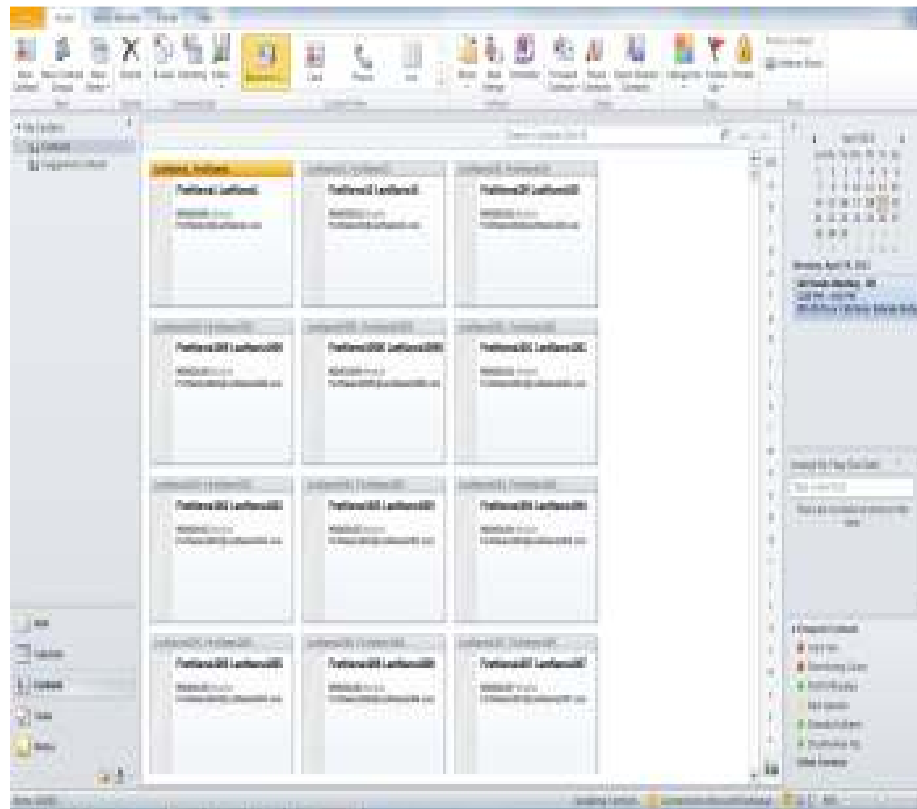
## Updating a salesforce contact



## After updation sync result in outlook

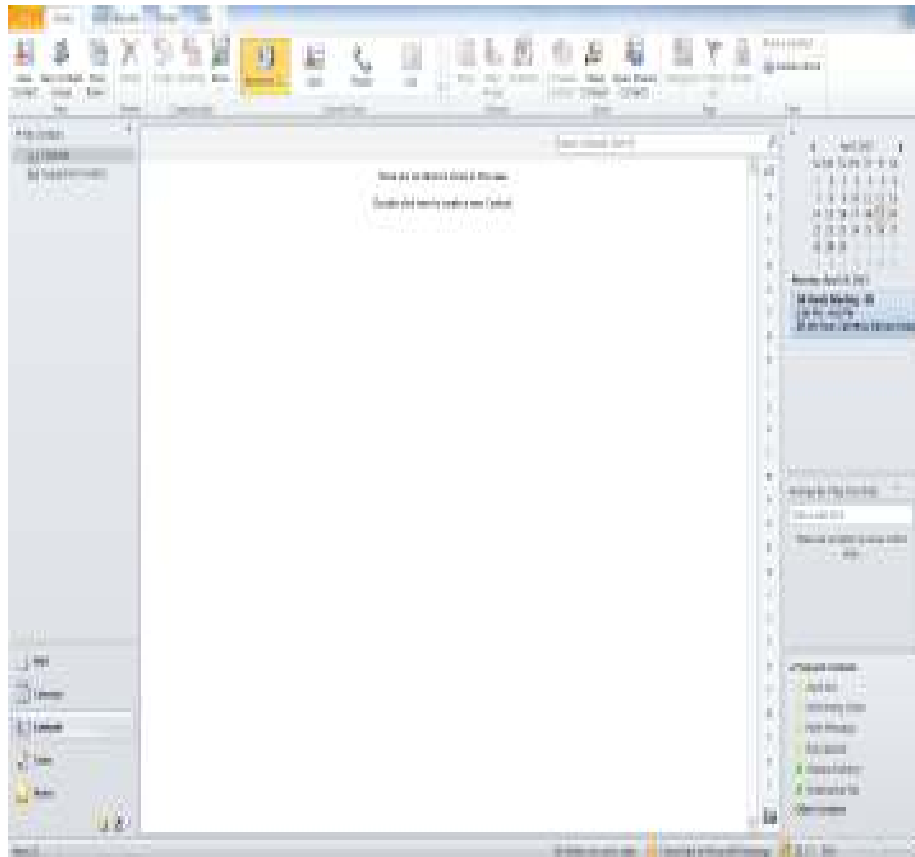


## Inserted 10,000 contacts in Outlook



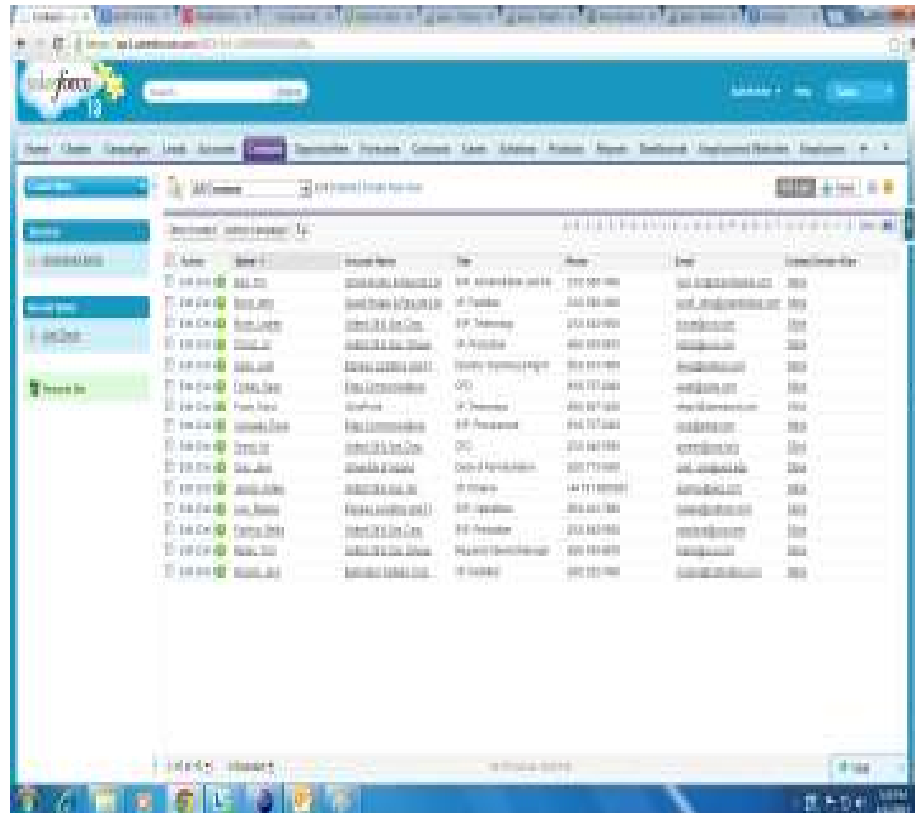


## Deleted all Contacts in Outlook





**Deleted all Contacts from Salesforce except those 15 contacts which do not have delete permission**



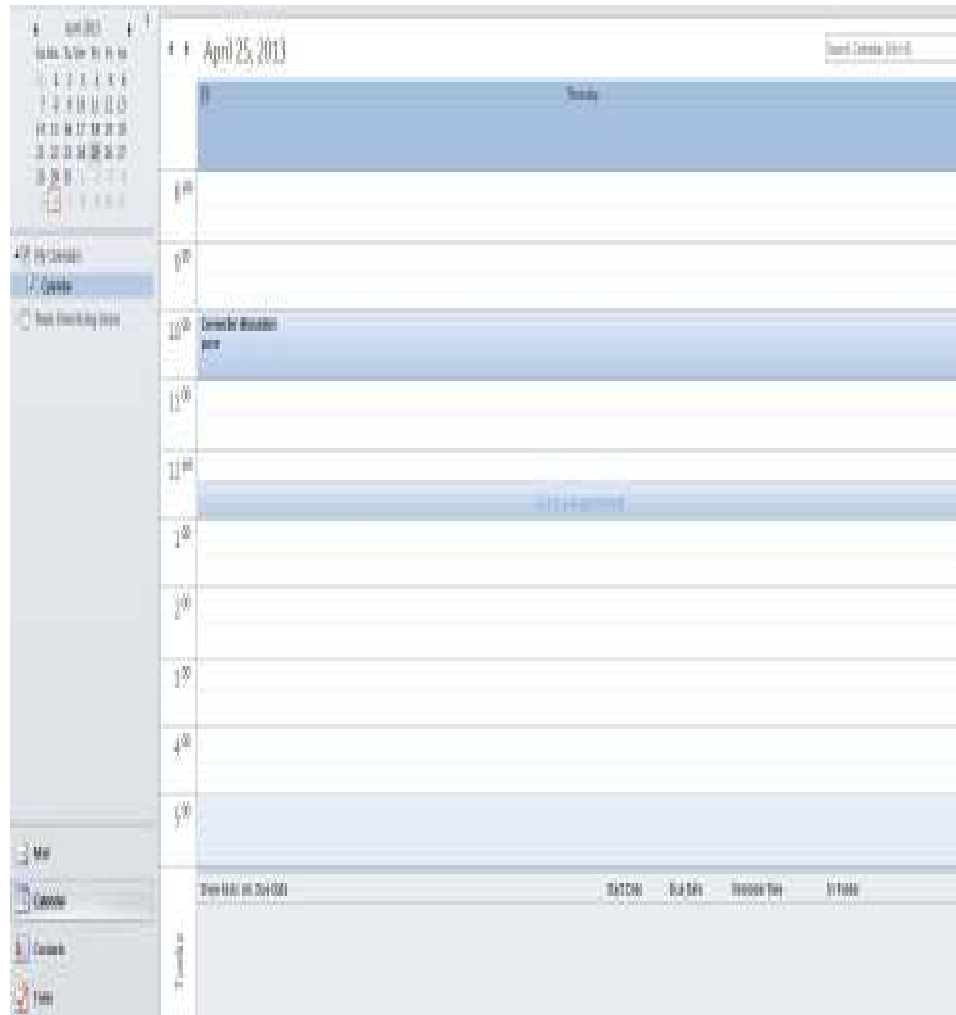
## Event Synchronization

### Event in Salesforce Before Synchronization

The screenshot displays the Salesforce user interface for a calendar. At the top, the Salesforce logo is on the left, and a search bar and user profile are on the right. Below the header, a navigation bar contains links for Campaigns, Leads, Accounts, Opportunities, Contacts, Products, and Sites. The main content area shows a calendar for Thursday, 24 April 2013. The calendar grid is mostly empty, with a tooltip over the 10:00 slot that says "(Event occurs here)". On the right side, there is a "My Tasks" sidebar with a table of tasks. The table has columns for "Task", "Subject", "Name", and "Status". One task is listed with the subject "MULTI-PHASE CAMPAIGN".

| Task | Subject              | Name | Status |
|------|----------------------|------|--------|
| 1    | MULTI-PHASE CAMPAIGN |      | 0      |

## Event of Salesforce reflected in Outlook after Sync



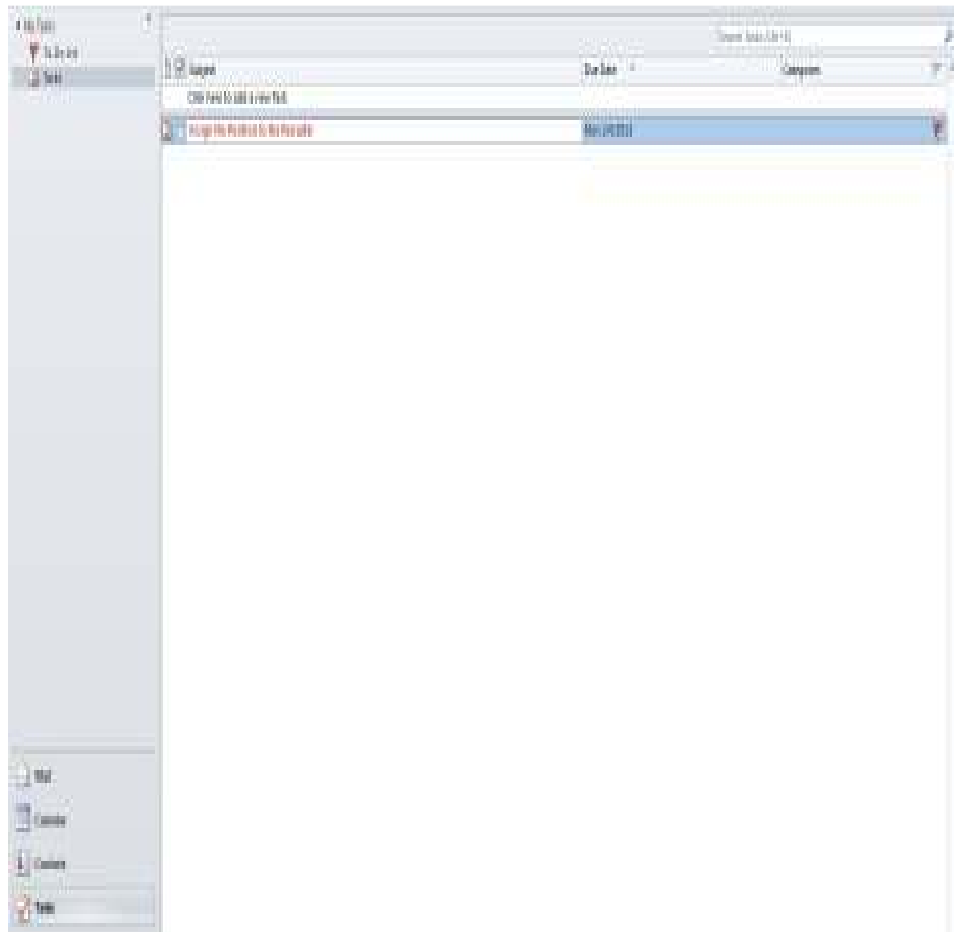
# Task Synchronization

## Task in Salesforce

The screenshot displays a Salesforce dashboard with several key sections:

- Calendar:** Shows a calendar for May 2013. It includes tabs for 'Scheduled Meetings' and 'Assigned Meetings'. A message states: 'Today 9:00 AM: You have an event scheduled to be met 7 days.' A calendar grid shows the current date (May 13, 2013) highlighted in blue.
- My Tasks:** A table listing tasks. The table has columns for 'Complete Date', 'Status', 'Name', and 'Assign To'. One task is visible: 'Migrate the Profiles to the Profile 1' with a status of 'In Progress'.
- Items to Approve:** A section with a 'Manage All' link. It displays the message: 'No items to approve.'
- Dashboard:** A summary section with three widgets:
  - Open Profiles by Functional Area:** A bar chart showing the distribution of open profiles across different functional areas.
  - Profiles Open Longer Than 60 Days:** A gauge chart showing the percentage of profiles that have been open for more than 60 days. The needle is positioned in the red zone, indicating a high percentage.
  - Profile Performance:** A summary card showing 'Profile Open Rate' and 'Average Days Open'. The 'Profile Open Rate' is 97.9% and the 'Average Days Open' is 14.

## Reflected Task in Outlook after Synchronization



## **ANNEXURE 3: Sample Code**

### **Sample Code of Configuration Frame**

```
packagepersistent.salesforceforoutlook;

importjava.awt.Dimension;

importjava.awt.FlowLayout;

importjava.awt.event.ActionEvent;

importjava.awt.event.ActionListener;

importjava.io.FileInputStream;

importjava.io.FileNotFoundException;

importjava.io.FileOutputStream;

importjava.io.IOException;

importjava.util.Properties;

importjavax.swing.*;

public class ConnectorConfigFrame extends JFrame

{

    private static final long serialVersionUID = 1L;

    public static void main(String[] args)

    {
```

```
        @SuppressWarnings("unused")
        ConnectorFramecFrame= new ConnectorFrame();
    }
    publicConnectorFrame()
    {
        JTabbedPaneconnectorTabbedView = new
        JTabbedPane();
        connectorTabbedView.setPreferredSize(new
        Dimension(100, 200));
        setSize(550,200);
        connectorTabbedView.addTab("Home", new
        HomePanel());
        connectorTabbedView.addTab("Salesforce
        Configuration", new SalesforceConfigPanel());
        connectorTabbedView.addTab("Outlook
        Configuration", new OutlookConfigPanel());
        connectorTabbedView.addTab("Proxy Configuration",
        new ProxyConfigPanel());
        connectorTabbedView.addTab("Synchronization
        Configuration", new SyncConfigPanel());
```

```

        connectorTabbedView.setTabPlacement(JTabbedPane.LEFT)
        add(connectorTabbedView);
        setVisible(true);
    }
}

class HomePanel extends JPanel
{
private static final long serialVersionUID = 1L;

    private JButton sfSync = new JButton("Synchronize");

    public HomePanel()
    {

        this.setLayout(new
            FlowLayout(FlowLayout.CENTER));

        add(sfSync);

        sfSync.addActionListener(new ActionListener()
        {

            public void actionPerformed(ActionEvent e) {

                Properties propSync = new Properties();

```



```
try {  
  
    propSync.load(new  
        FileInputStream("synconfig.properties"  
        ));  
    } catch (FileNotFoundException e1) {  
        e1.printStackTrace();  
        //log this into log file  
    } catch (IOException e1) {  
        e1.printStackTrace();  
        //log this into log file  
    }  
}  
  
if("BiDirectional".equalsIgnoreCase(propSync.getProperty("syncBiDirectional"))){  
  
    Synchronization synchronize=new Synchronization();  
  
    synchronize.salesforceToOutlook();  
  
    synchronize.outlookToSalesforce();  
    }else{  
  
System.out.println("Salesforce to outlook");  
    Synchronization synchronize=new Synchronization();
```

```

        synchronize.salesforceToOutlook();
    }
}
});
}
}
class SalesforceConfigPanel extends JPanel
{
private static final long serialVersionUID = 1L;
public SalesforceConfigPanel()
{
    this.setLayout(new
    FlowLayout(FlowLayout.CENTER));

    JLabelsfUsername_Label=new
    JLabel("Username ");

    JLabelsfPassword_Label=new
    JLabel("Password ");

    JLabelsfSecurityToken_Label=new
    JLabel("Security Token ");

    finalJTextFieldsfUserName_TextField=new
    JTextField(20);

```

```

finalJTextFieldsfPassword_TextField=new
JTextField(20);

finalJTextFieldsfSecurityToken_TextField=new
JTextField(20);

JButtonsfSave=new JButton("Save");

add(sfUsername_Label);

add(sfUserName_TextField);

add(sfPassword_Label);

add(sfPassword_TextField);

add(sfSecurityToken_Label);

add(sfSecurityToken_TextField);

add(sfSave);

sfSave.addActionListener(new ActionListener()

{

public void actionPerformed(ActionEvent e) {

Properties prop = new Properties();

try {

//set the properties value

```

```

/*****SALESFORCE*****/

```

```
        prop.setProperty("salesforceUsername",sfUserName_TextField.getText());
```

```
        prop.setProperty("salesforcePassword",sfPassword_TextField.getText());
```

```
        prop.setProperty("salesforceToken",sfSecurityToken_TextField.getText());
```

```
                //save properties to project root folder
```

```
                prop.store(new  
                FileOutputStream("salesforceconfig.properties",true),  
                null);
```

```
                } catch (IOException ex) {
```

```
                    ex.printStackTrace();
```

```
                }
```

```
            }
```

```
        });
```

```
    }
```

```
}
```

```
class OutlookConfigPanel extends JPanel
```

```

{
    private static final long serialVersionUID = 1L;
    public OutlookConfigPanel()
    {
        this.setLayout(new
        FlowLayout(FlowLayout.CENTER));

        JLabel outlookUsername_Label = new
        JLabel("Username");

        JLabel outlookPassword_Label = new
        JLabel("Password");

        final JTextField outlookUserName_TextField = new
        JTextField(20);

        final JTextField outlookPassword_TextField = new
        JTextField(20);

        JButton outlookSave = new JButton("Save");

        add(outlookUsername_Label);

        add(outlookUserName_TextField);

        add(outlookPassword_Label);

        add(outlookPassword_TextField);

        add(outlookSave);

        outlookSave.addActionListener(new ActionListener()
        {

```

```

public void actionPerformed(ActionEvent e) {
    Properties propOutlook = new Properties();
    try {
        //set the properties value

/*****OUTLLOOK*****/

        EncryptionDecryptionencryptionDecryption=new
        EncryptionDecryption();

        propOutlook.setProperty("outlookUsername
        ",outlookUserName_TextField.getText());

        propOutlook.setProperty("outlookPassword
        ",encryptionDecryption.encode(outlookPassword_Text
        Field.getText()));

//save properties to project root folder

        propOutlook.store(new
        FileOutputStream("outlookconfig.properties",true),
        null);

    } catch (IOException ex) {
        ex.printStackTrace();
    }
}

```

```

        });
    }
}

class ProxyConfigPanel extends JPanel
{
private static final long serialVersionUID = 1L;

public ProxyConfigPanel()
{
    this.setLayout(new
    FlowLayout(FlowLayout.CENTER));

    JLabel proxyHost_Label = new JLabel("Host
");

    JLabel proxyPort_Label = new JLabel("Port
");

    JLabel proxyUsername_Label = new
    JLabel("Username ");

    JLabel proxyPassword_Label = new
    JLabel("Password ");

    final JTextField proxyHost_TextField = new
    JTextField(20);

```

```
finalJTextFieldproxyPort_TextField=new  
JTextField(20);
```

```
finalJTextFieldproxyUsername_TextField=new  
JTextField(20);
```

```
finalJTextFieldproxyPassword_TextField=new  
JTextField(20);
```

```
JButtonproxySave=new JButton("Save");
```

```
add(proxyHost_Label);
```

```
add(proxyHost_TextField);
```

```
add(proxyPort_Label);
```

```
add(proxyPort_TextField);
```

```
add(proxyUsername_Label);
```

```
add(proxyUsername_TextField);
```

```
add(proxyPassword_Label);
```

```
add(proxyPassword_TextField);
```

```
add(proxySave);
```

```
proxySave.addActionListener(new ActionListener() {
```

```
public void actionPerformed(ActionEvent e) {
```

```
Properties propProxy = new Properties();
```



```
        try {  
            //set the properties value  
  
            /*******PROXYSETTING******/  
  
            propProxy.setProperty("proxyhost",  
                proxyHost_TextField.getText());  
  
            propProxy.setProperty("proxyport",proxyPort_TextField.getT  
                ext() );  
  
            propProxy.setProperty("proxyUsername",proxyUsername_Te  
                xtField.getText());  
  
                propProxy.setProperty("proxyPassword  
                ",proxyPassword_TextField.getText());  
  
                //save properties to project root folder  
  
                propProxy.store(new  
                FileOutputStream("proxyconfig.properties",true), null);  
  
        } catch (IOException ex) {  
            ex.printStackTrace();  
        }  
    }  
});
```

```

    }
}
class SyncConfigPanel extends JPanel
{
    private static final long serialVersionUID = 1L;
    public SyncConfigPanel()
    {
        this.setLayout(new
        FlowLayout(FlowLayout.CENTER));

        JLabel syncDirection_Label = new
        JLabel("Synchronization Direction ");

        JLabel syncInterval_Label = new JLabel("Auto-
        Sync Interval ");

        final JRadioButton syncBiDirectional = new
        JRadioButton("Bi Directional ");

        final JRadioButton syncSftoOutlook = new
        JRadioButton("Salesforce to Outlook
        ");

        ButtonGroup group = new ButtonGroup();
        group.add(syncBiDirectional);
        group.add(syncSftoOutlook);

        final JTextField syncInterval_TextField

```

```

=new JTextField(20);

        JButtonoutlookSave=new JButton("Save");
        add(syncDirection_Label);
        add(syncBiDirectional);
        add(syncSftoOutlook);
        add(syncInterval_Label);
        add(syncInterval_TextField);
        add(outlookSave);
        outlookSave.addActionListener
(newActionListener() {
        public void actionPerformed(ActionEvent e) {
            Properties propSync = new Properties();
            try {
                //set the properties value

                /*******Synchronization setting*****/

                if(syncSftoOutlook.isSelected())
propSync.setProperty("syncDirection",syncSftoOutlook.getText());
            else

```

```
        propSync.setProperty("
syncBiDirectional", syncBiDirectional.getText() );

        propSync.setProperty("syncInterval",syncInterval_TextField.
getText() );

//save properties to project root folder

        propSync.store(new
FileOutputStream("synconfig.properties",true), null);
        } catch (IOException ex) {
        ex.printStackTrace();
        }
    }
});
}
}
```