



READY-TO-GO WEB SALES & SMALL BUSINESS KIT

Technical Report

<https://github.com/horaceli/bizwebsys>

This report is submitted as part requirement for the UCL COMP2014 Systems Engineering II module (Group 10). It is substantially the result of the group's own work except where explicitly indicated in the text. The report may be freely copied and distributed provided the source is explicitly acknowledged.

Jing Li, Horace Li, Eeren Tan, Weiwei Liang, Tim Szeto
jing.li.11@ucl.ac.uk; me@horaceli.com; zcesh69@ucl.ac.uk; vvianessa@hotmail.com; tins820@live.hk

Contents

Introduction	7
Foreword.....	7
Management.....	8
Research.....	9
Background	9
Progress.....	9
Market.....	9
Email Client	9
Website (Marketing)	9
Web Store	10
ERP	10
Social Networks.....	11
Technology.....	12
Web Language.....	12
Framework.....	12
Hosting	12
Requirements.....	13
Data Gathering	13
Requirements List	17
Use Cases	19
Design.....	27
Architecture	27
Data Stores.....	27
SQL	27
LDAP	27
Mail	28
Cloud	28
ERP Session Control	29
Login control	29
User data.....	29
Reset password	30

POS.....	32
Inventory.....	35
Contacts and Employees.....	37
Sales and Purchases	38
Architecture	38
Purchases	39
Sales	40
Details	41
Social Networks.....	41
Production Architecture	41
Component Description	41
Implementation	42
Architecture	42
Authentication	42
Data Stores.....	42
Mail	44
Cloud	45
POS.....	45
Item object.....	45
Ajax and Json.....	46
Print function	47
Item object.....	47
Ajax and Json.....	48
Inventory.....	49
Contact and Employee	49
Sales and Purchases	51
Social Networks.....	53
Testing.....	58
Different Types of Testing.....	58
Functionality Testing.....	58
Usability Testing.....	58
Compatibility Testing	58
Performance Testing.....	58

Security Testing.....	58
Testing Toolkits We used	58
CodeIgniter Unit Testing	58
Selenium	58
Exploratory Testing	58
Inventory.....	58
Test Type of Items.....	59
Test Number of Items	59
Automating	59
Contacts	59
Test Retrieval Type of Contacts	59
Test Number of Contacts	60
Automating	60
POS.....	60
Test new order	60
Automating	61
Sales and Purchases	61
Test Retrieval Type of Sales	61
Automating	62
Employee	62
Test Retrieval Type of Employee	62
Test Number of Employee	62
Automating	63
Social Networks.....	63
Testing for Posting a Status onto Facebook.....	63
Testing for Posting a Status onto Twitter	65
Testing for Posting a Status onto Both Facebook and Twitter	67
Testing for Error Message for not Checking Any Checkboxes	69
Automating	71
Review.....	72
Design evaluation.....	72
Cohesion and Decoupling: MVC.....	72
Appropriate division into components and sub-systems	73

Robustness and stability	73
Scalability	73
Usability	73
Portability.....	73
Standards	73
Aesthetic and minimalist design	73
Things to be improved	74
Single database design.....	74
Security	74
SSO	74
Future Development	74
Mobile apps	74
Online Shopping.....	74
Calendar	74
Newsletter.....	74
SEO	74
Customizable app panel.....	74
Project management evaluation.....	75
Conclusion	75
System Manual	77
Get system files.....	77
Domain name and hosting.....	77
For windows server.....	77
Setup WampServer	77
Install system	77
Setup LDAP server.....	77
For Linux server.....	78
Apache installation	78
PHP installation	78
MySQL installation	78
Get PHP to work with MySQL	79
Install system	79
Setup LDAP server.....	79

Setup Mail Servers	79
Other Subsystems	79
User Manual	80
Login page	80
Workspace panel	80
ERP	80
Inventory	81
Inventory home screen	81
Create an new item in inventory	81
Display an item.....	82
Edit item	83
Contacts	83
Contacts home panel	83
Display detail.....	83
Edit detail	84
POS.....	84
Pos home panel.....	84
Select items	85
Make a order.....	85
Make a payment	86
Show receipt	87
Sales and Purchases	87
Sales panel	87
Create new invoice	87
Display invoice	88
Purchases panel	88
Employee	89
Edit employee	89
Apps	89
Mail	89
Owncloud	91
Webmaster tools.....	91
Connecting to the public.....	92

Settings..... 93

 Change password 93

 Company Settings 94

Introduction

Foreword

Many small businesses are launched every single day here in the United Kingdom. These small businesses are like cafes, bars, restaurants, stationery shops, hairdressers and etc. They traditionally do not use computers in facilitating their daily activities. Our aim for this project is to investigate how a thin client web solution can be suitably integrated into these small business to improve their businesses.

Our idea is to build a generic yet customizable app panel for various small businesses. This app panel contains HTML5 apps which include contact management, inventory, social network integration, emails, sales and purchase management and others. These tools generally help a small business's employees to carry out their daily jobs, such as documentation and handling transactions. To identify what apps are needed, we had carried out interview sessions with two small businesses and extract information we needed on determining what apps our system should have.

Besides the main app panel, which the users are employees of a small business, and which we call it as the "Workspace", our system also host a website of the small business, potentially generating dynamic webpages based on the data with have on the database. For example, using the products a small business has in their inventory app, we can dynamically generate webpage of products offered by the small business. To investigate on how this could be done for any business, we have explored several web applications that allow users to set-up their websites and potentially online stores as well. The initial idea is to create another app under the workspace which allow employees to choose the template of their website and using this template, generate webpages with information from the database. Potentially, we can also generate a sales model based on the information kept in database for sales, to enhance the website of our client.

Two small businesses who acted as our direct clients are a restaurant and a stationery shop. We held interviews with them, and scrutinized their daily activities and record what documents are needed to be kept. Then, we developed a set of basic apps that assist them in their businesses. The whole system is designed to be a single deployment at this moment. However, we used specialization and generalization approach in development the core structure of our system. Notably, the environment, for example the workspace is design with a generic approach on our mind so that it fits any business, while the specialization happens in app level where we create different HTML5 apps for different purposes.

This project is interesting because there is a big market for the system which we are developing. London is one of the world's largest economy hubs of Europe and one of the most populated cities as well. We have access to a lot of small businesses as well as startups projects launch by the students in our university. To come up with something which helps all these small businesses to grow is certainly exciting because this project can change the way small businesses operate and raise the awareness on the importance of introducing technologies into their businesses.

This project has introduced numerous unprecedented challenges to most of the group members. Firstly, more than half of the group members have had no experience in developing web application nor any knowledge on the underlying architecture of a web application. In terms of competition as a web application, we are facing a huge challenges as there are many web applications out there which are doing something similar. Thus, we have to come up with idea to make our system special and practical to the small businesses who are our main clients. For example, there is OpenERP, Salesforce, WiX and others.

After all, we don't want to be developing an application which does not practically help any small businesses at all. We also faced problems such as how to make this system useful to the small businesses. What we aim to do is not migrating what small businesses are already doing to a computer system, but using the power of the system, discover possibility to enhance their business by growing their market or expanding their customers.

Overall, the unique selling point of the system we are developing is a suite of tools for small businesses to further grow their business by means of introducing more potential customers, increase publicity and improve work efficiency among workers. Small businesses usually have more regular customers then randomly drop by customers and thus, we want to help these businesses to be found by their potential customers using social network and various search engines as well.

Management

For project management, our team has two joint leaders and one meeting manager. Each decision made on development is discussed and agreed by the two joint leaders. At the same time, one leader is mainly in charge of assigning tasks while the other leader is in charge of quality control and reviewing code commits. The meeting manager is responsible for arranging meetings and booking meeting rooms.

For project meetings, we normally have three meetings a week during term time. The meetings are taken place in lab session so that we can discuss issues with professors and TAs. During holiday, we would have meeting once a week, either book a meeting room or Skype meeting. During the meeting, each team member will present what they have done, discuss the problem they have, check our progress according to milestones and assign new tasks.

For team communication, we use various online media for different purpose of communication. To assign task, we used do.com, which is a web app for project management, from there we are able to assign task to team member with details and deadline, also track unfinished and finished tasks. For code comments and bug reports, we use the Github issues panel, where we can assign issues to specific team member and track closed issues. For file sharing, we use dropbox and Google doc. Besides, our team use Facebook group for normal discussion.

For task assignment, we would first consider the interest of team member. If they ask to work on some part, then we assign the part to him first. We would also consider the strength and weakness of each member, and make sure everyone has work to do.

For progress control, we have milestones for whole project development and also task deadlines for individual tasks assigned to each team member. Team leaders will check if tasks finished on time. If there are tasks that cannot be finished on time, we will try to find the problem together, assess the difficulty and either help work on it or change the solution.

Research

Background

Although this report is a follow on from the original prototype report detailing initial efforts to create a Web Sales app, the overlap is very limited. As such, this can be considered a fresh start, without many of the original constraints imposed by assumptions, such as the operation model, hardware constraints such as the use of RPi. Indeed, for this attempt at creating a working prototype, we opted to concentrate on the server side and web technologies, rather than optimizing the ArchLinux installation which constituted the bulk of our final product in our previous attempt. While our architecture potentially allows for limited integration with our linux client workstation, no explicit provisions have been made in this aspect.

While functional requirements were being gathered with small businesses, work had already begun defining the high level architecture. Certain functionality was required regardless of the outcome of the requirements gathering. These included email, file storage, and the ERP. Together, they formed the basic outline of our project, as briefly outlined in the presentation.

Progress

While the requirements gathering was in progress for the ERP section of the solution, research was underway for requirements where a consensus had already been reached. This primarily included market research of other contenders or preexisting packages in the area, as well as technical decisions that had to be made before any functionality could be implemented.

Market

Email Client

The major decision that had to be made was whether this was to be developed and hosted in house or outsourced. Current contenders in the market include Microsoft's Office 365 and Google Apps, complemented by numerous smaller offerings such as 20 Mail St. Since our hosting would not be limited to web services, we had the option of keeping this in-house which would allow for deeper integration.

In terms of the actual email UI, the most popular choices are SquirrelMail, RoundCube, and Horde. Horde was dropped from our candidates due to its large feature set - while we appreciated its extensive functionality, it was preferable for us to develop as much as possible ourselves, and hence opt for a lighter-weight webmail client. In the end, RoundCube was settled upon for its much improved UI over SquirrelMail.

Website (Marketing)

In terms of marketing, WiX is our greatest competitor. Its rich and user-friendly features allow users to create and host their websites with ease using 100+ design templates.

Strengths

- Secure, reliable and dedicated web hosting
- 24/7 support center
- Allow user to set up and online store
- Integrate social network services such as Facebook
- User-friendly drag-and-drop website building feature
- Uses Google Analytics for site statistics

- Step-by-step video tutorials to assist user in creating and hosting websites
- Guidance on building a website which is search engine optimized
- The tools provided to build a website is completely free



Weaknesses

- Doesn't have other value propositions as a marketing tool beside creating and hosting a website with online store
- Online store only provides payment options such as PayPal, Google Checkout (also known as Google Wallet) and PagSeguro
- Websites created via free account contain Wix ads

Web Store

This was one of the areas which we considered outsourcing, due to the difficulty in its implementation and the amount of work required to integrate it properly with the rest of the system. Unlike email, which would only require a common user database and address book, any packages which we could adapt for our own use (of which there are many, such as osCommerce) would require heavier integration with regards to stock, price, product listings, payment, customer help and CRM, delivery dates, and so on.

The market for such SaaS is still in its infancy. Numerous startups exist, including many that grew in recent years to take advantage of new opportunities to reach out to consumers such as Facebook storefronts and increasing numbers of consumers who shop online, but none have yet become a dominant player in this niche market. We briefly reviewed Miiduu, a relatively settled player in the market, and TicTail, a startup barely a year old. However, we failed to find any way to integrate its product listing and storefront with our own product stock and inventory backend. Since there isn't an industry standard for storing stock information and prices, these services require manual input of product data, which has to be kept in sync with the inventory in the main system manually.

ERP

As for the business management, Compiere and Openerp are two major competitors which are offering full-featured and fully integrated business management system comprising many tools.

OpenERP

Strengths

- Essentially a web application, no download is needed

- Includes management tools to handle CRM, invoicing and payments, POS, project management, accounting and finance, sales management, warehouse management, purchase management, MRP, employee directory, recruitment process, leave management, expense management etc.
- Reporting tool allows user/business to analyze its data from sales, manufacturing, inventory and others, allowing tabulation of data graphically and generate reports for better decision making purpose
- User-friendly interface, with templates on various document types, such as invoices, sales orders, receipts and others
- Contacts, employees and other data can be imported using .csv files
- Administrator has total control on the access rights of other users
- Settings are clearly categorized and are tools independent
- It is open source and can be used commercially by developers or anyone
- It is free to set-up an account

Weaknesses

- Supported version of OpenERP is not free
- First user, also the administrator will need proper guidance
- Purely a business management tools, it doesn't offer any opportunities of connecting potential customers to the business

Compiere

Strengths

- Has similar tools and features offered by OpenERP, such as performance management tools like reports generator, business view later and ERP as well as CRM tools
- Integration with salesforce allows accurate data synchronization between Compiere ERP and Salesforce
- Community Edition is open source and it is distributed under Gnu General Public License

Weaknesses

- Community edition has limited features and it's not supported by Compiere
- It is a professional system for larger scales businesses

Social Networks

In our solution, we aim to include links with social networks, notably Facebook, Twitter, and Google+. Social networks presences have been proved to boost search engine rankings. For example, the existence of a Google+ profile has been shown to positively contribute to SEO of its affiliate site. Furthermore, a recent study by Deloitte has shown that tweets (and Twitter chatter) are directly correlated with the success of video games.

Facebook and Twitter both provide extensive APIs. Google+, however, proved to be impossible to integrate with on a level we'd have liked, due to their closed APIs.

Technology

Web Language

The primary contenders for this were PHP and ASP.NET, with the most resources available for learning and libraries. Since we don't have a non-HTTP backend to contend with, cross platform scripts such as Python/Java are not a necessity. With our philosophy of 'open beats closed' and objective of interoperability, PHP was chosen for its support of Linux and integration with other FOSS products. ASP.NET would have sufficed, but that would have largely limited us to Microsoft/Windows-based software packages. Mono would have proved a viable alternative, but it's generally advisable to stay within a single ecosystem (Linux vs Windows).

Framework

Naturally, a framework was to be chosen over writing custom code from scratch, both to ensure a common standard and to save time writing any overhead. The choice for this was substantially harder than the choice of scripting language, since we could choose from the whole library spectrum from the enterprise-level Zend Framework, through to heavyweights such as Symfony and Yii, 'mainstream' options such as CakePHP, and up-and-coming packages like Laravel. Ultimately, we opted for CodeIgniter, due to its maturity and more tolerant approach to MVC compared to the other frameworks.

Hosting

Microsoft Azure accounts were provided for cloud hosting, which provided IaaS and PaaS. We settled on the former, as we wanted to avoid API lock-in. The current XaaS market is not yet mature enough for a uniform standard. AWS and OpenStack are strong contenders, but until specifications have been standardized, dependence on Azure will prevent migration to alternatives.

With a mind to future proof the system, we wanted to allow businesses to locally host their cloud on a server of their choice, in the way that many small tech-savvy companies currently have an NAS or a small local server on their network. By deciding on using a virtual machine, this could be deployed onto any computer, not necessarily a cloud provider.

Requirements

Data Gathering

We have carried out interviews with two small businesses to gather data needed. One of them is Oriental Dragon Restaurant and the other is Tower Stationery, both located at Cleveland Street, London. Verbal interviews and written responses are recorded. The results of interviews will be shown side by side to compare and contrast the needs of different businesses.

	Oriental Dragon	Tower Stationery
What is the nature of your business?	We are mainly a restaurant but also provide KTV entertainment as a package	Stationery retail shop but also provide binding and copying services
Who are your main customers?	Well, it could be anyone, but mainly the working individuals nearby or students	Mainly students who live nearby. Secondary school or university students
How do you currently market your own business?	Customers tell their friend about it and we also offer CSSA (Chinese Students and Scholar Association) members discount so they advertise us on their newsletters to their members	It is mainly spread by words of customers
Do you have access to computer connected to the internet at home?	Yes	Yes
Do you have access to computer connected to internet at your working location?	No	Yes
Do you have your own website?	No	No
Do you think having your own website would help your company?	Yes	Yes

	Oriental Dragon	Tower Stationery
Do you do delivery service or online order?	No and no, but we take phone orders, it has to be collected by the customer	We don't do deliveries and we don't have an online platform to take orders. Service is only given after payment. Therefore, no.
How do you think having your own website/facebook page or twitter account would help improving your business?	I can post my menu and any student offers on the website. It would be good if anyone who is searching for an oriental restaurant nearby them through google would find us one way or the other.	I don't really know, but if I can make people follow my facebook page or twitter account, I think I can market my service or products easier.
What kind of document or paperwork do you deal with?	Receipts, invoices, purchasing, costing	Sales receipts, supplier invoices, purchasing
Do you use templates when you write on those documents?	For invoices, I just keep it and we do not provide sales orders, we handwrite customer receipts and they don't show the details either, so basically, no	Yes, the receipts printed out from the till has a format and I use papers with preset templates when sending supplier orders.
Do you have any record of your customers?	No	No
How do you store your supplier details?	In a contact book	In a contact book
Do you manage your own financial accounts?	No, it's handle by private accountant	No, it's handle by private accountant

The following tables, containing most functions, tools or documents a small business might be using or documenting is presented to the interviewee. They are asked to tick any boxes which corresponds to the functions, tools or documents related or useful to them. A "Yes" is written to replace a tick if and only if the corresponding feature is essential to the small business and if the feature itself is used by them.

Oriental Dragon Restaurant		
Tick box	Feature	Description
No	Financial Accounting	General Ledger, Fixed Asset, Payables, Receivables, Cash Management, Financial Consolidation
Yes	Management Accounting	Budgeting, Costing, Cost Management, Activity Based Costing
No	Human Resources	Recruiting, Training, Payroll, Benefits, 401K, Diversity Management, Retirement, Separation
No	Manufacturing	Engineering, Bill of Materials, Work Orders, Scheduling, Capacity, Workflow Management, Quality Control, Manufacturing Process, Manufacturing Projects, Manufacturing Flow, Product Life Cycle Management
Yes	Supply Chain Management	Supply Chain Planning, Supplier Scheduling, Order to Cash, Purchasing, Inventory, Product Configurator, Claim Processing
No	Project Management	Project Planning, Resource Planning, Project Costing, Work Break Down Structure, Billing, Time and Expense, Performance Units, Activity Management
No	Customer Relationship Management	Sales and Marketing, Commissions, Service, Customer Contact, Call Center Support - CRM systems are not always considered part of ERP systems but rather BSS systems . Specifically in Telecom scenario
No	Data Services	Various "self-service" interfaces for customers, suppliers and/or employees
Yes	Access Control	Management of user privileges for various processes

Tower Stationery		
Tick box	Feature	Description
No	Financial Accounting	General Ledger, Fixed Asset, Payables, Receivables, Cash Management, Financial Consolidation
Yes	Management Accounting	Budgeting, Costing, Cost Management, Activity Based Costing
Yes	Human Resources	Recruiting, Training, Payroll, Benefits, 401K, Diversity Management, Retirement, Separation
No	Manufacturing	Engineering, Bill of Materials, Work Orders, Scheduling, Capacity, Workflow Management, Quality Control, Manufacturing Process, Manufacturing Projects, Manufacturing Flow, Product Life Cycle Management
Yes	Supply Chain Management	Supply Chain Planning, Supplier Scheduling, Order to Cash, Purchasing, Inventory, Product Configurator, Claim Processing
No	Project Management	Project Planning, Resource Planning, Project Costing, Work Break Down Structure, Billing, Time and Expense, Performance Units, Activity Management
No	Customer Relationship Management	Sales and Marketing, Commissions, Service, Customer Contact, Call Center Support - CRM systems are not always considered part of ERP systems but rather BSS systems . Specifically in Telecom scenario
No	Data Services	Various "self-service" interfaces for customers, suppliers and/or employees
Yes	Access Control	Management of user privileges for various processes

Requirements List

ID	Details	Type	Priority
General			
RQ1	The system should be user friendly towards employees (no over-complicated training should be needed)	Non-functional	Should Have
RQ2	The system shall use an Internet browser as its user interface	Non-Functional	Must Have
RQ3	The system should display a web desktop with a panel of HTML5 apps as login homescreen	Non-Functional	Must Have
Account Setup			
RQ4	Each registered business should be given a sub-domain to host its website	Functional	Could Have
RQ5	One administrator is setup and only the administrator can add employees	Functional	Must Have
RQ6	Administrator has the right to edit the access rights of an invited user	Functional	Could Have
RQ7	Users should be able to change their password	Functional	Must Have
Marketing			
RQ8	One of the apps shall provide web building and editing feature to modify the company's website	Functional	Should Have
RQ9	Facebook page and Twitter account should be synchronized and accessible by members of the same business	Functional	Must Have
RQ10	Tools to generate newsletter to subscribers and customers	Functional	Should Have
RQ11	Generation of webpages from sales models or data	Functional	Should Have
RQ12	Online store setup for online shopping	Functional	Should Have
RQ13	Online payment system	Functional	Could Have

Business Management Tools			
RQ14	Customer eInvoicing and Payment system integrated with online store (customer invoices, refunds, sales receipts, customer payments)	Functional	Must Have
RQ15	Supplier Management to manage supplier invoices, refunds, purchase receipts, supplier payments	Functional	Must Have
RQ16	POS system integrated with Inventory	Functional	Must Have
RQ17	Personal calendar app	Functional	Must Have
RQ18	Inventory Control to handle physical inventory	Functional	Must Have
RQ19	Inventory Control to handle Incoming products and Deliver products	Functional	Could Have
RQ20	Employee management that keeps employee details	Functional	Must Have
RQ21	Customer and Supplier contact management	Functional	Must Have
RQ22	Basic office suite	Functional	Should Have
RQ23	Business account dependent cloud storage for documents sharing	Functional	Should Have
RQ24	Folder views with list selections based on accounts or date	Non-Functional	Could Have
Analysis			
RQ25	Sales Analysis to helps promote products	Functional	Should Have
RQ26	Inventory Analysis to improve product replenishing	Functional	Should Have
RQ27	Online sales analysis based on various identifiers, such as customer details	Functional	Could Have
Performance			
RQ28	The system should be available 24/7	Non-Functional	Should Have

Use Cases

The concept of the use case analysis below involves the client and the administrator. The general employee of a small business performs the most basic functions and has limited access to the entire system. The administrator, on the other hand, perform more advanced operations and actions on the system and have access to most of the system configurations and functions. The key administrator feature is that an administrator has the ability to create new employee and reset an employee's password while a basic user does not. We are presenting part of our overall use cases as there are too many apps in the workspace and each app, should ideally have different set of use cases. Following the list of use cases, we will have the use case specifications in detail.

ID	Use Case	Actors
1	CreateEmployee	Administrator, Employee
2	EditEmployee	Administrator
3	EditCompanySettings	Administrator
4	ResetEmployeePassword	Administrator
5	LogIn	Administrator, Employee
6	ChangePassword	Administrator, Employee
7	ViewItemList	Administrator, Employee
8	CreateItem	Administrator, Employee
9	DeleteMultipleItem	Administrator, Employee
10	PostStatus	Administrator, Employee

The use case specifications describe the main and alternative flows to achieve a specific use case, including the actors who are involved in the use case. Besides, each specification states the preconditions required for a use case to be performed as well as postconditions following the completion of a use case

Use Case : CreateEmployee
ID : 1
Brief Description : Administrator/Employee creates an employee
Primary Actors : Administrator, Employee
Secondary Actors : n/a
Preconditions : User has logged in to the system
Main Flow : The use case starts with system showing the workspace app panel The actor clicks on employee and is being shown a list of employees already exist The actor clicks on “Create” button on the top left corner The system displays a create new employee form The actor fills in the form The actor clicks save The system displays the new employee created if successful
Postconditions : An employee account is created and added to the system with information entered
Alternative Flow : The actor clicks save Validation and verification errors are shown if there exists any

Use Case : EditEmployee
ID : 2
Brief Description : Administrator edits an employee
Primary Actors : Administrator
Secondary Actors : n/a
Preconditions : User has logged in to the system
Main Flow : The use case starts with system showing the workspace app panel The actor clicks on employee and is being shown a list of employees already exist The actor clicks on an employee listed The system displays a an employee The actor clicks "Edit" button on the top eft The actor updates information on the form The actor clicks save on the top left corner The system displays the employee updated if successful
Postconditions : An employee account is updated in the system
Alternative Flow : 7. The actor clicks save 8. Validation and verification errors are shown if there exists any

Use Case : EditCompanySettings
ID : 3
Brief Description : Administrator changes company settings
Primary Actors : Administrator
Secondary Actors : n/a
Preconditions : User has logged in to the system

Use Case : EditCompanySettings

Main Flow :

The use case starts with system showing the workspace app panel
The actor clicks on the settings tap on the top
The actor is shown with the settings which are modifiable
The actor makes changes to the settings
The actor clicks save
The settings page refreshes with updated information

Postconditions :

Settings of the company/business are updated

Alternative Flow :

The actor clicks save
Validation and verification errors are shown if there exists any

Use Case : ResetEmployeePassword

ID : 4

Brief Description : Administrator changes/resets an employee's password

Primary Actors : Administrator

Secondary Actors : n/a

Preconditions : User has logged in to the system

Main Flow :

The use case starts with system showing the workspace app panel
The actor clicks on employee and is being shown a list of employees already exist
The actor clicks on an employee listed
The system displays a an employee
The actor clicks "Edit" button on the top eft
The actor changes password of employee
The actor clicks save on top left corner
The updated employee is displayed

Postconditions :

Password of the employee is updated and changed in the system

Alternative Flow :

The actor clicks save
Validation and verification errors are shown if there exists any

Use Case : LogIn
ID : 5
Brief Description : Administrator/Employee logs in to the system
Primary Actors : Administrator, Employee
Secondary Actors : n/a
Preconditions : n/a
Main Flow : The user is at the website homepage The user clicks sign in on the top right corner The user fills in username and password The user clicks "Sign In" The user is directed to workspace app panel if successful
Postconditions : User gains credentials and is displayed with the workspace app panel
Alternative Flow : The actor clicks save Error occurs and is displayed if the username or password is invalid

Use Case : ChangePassword
ID : 6
Brief Description : Administrator/Employee changes password
Primary Actors : Administrator, Employee
Secondary Actors : n/a
Preconditions : User is logged into the system
Main Flow : The user is at the workspace app panel The user clicks on the top right corner where a button shows the user's username A dropdown list occurs The user clicks the option "Change Password" The user is directed to a change password page The user enters current password, new password and re-enter new password The user clicks "Done" The user is displayed with a message saying the password has been successfully changed

Use Case : ChangePassword
Postconditions : User's password is changed and saved in the database
Alternative Flow : The actor clicks save Error occurs and is displayed if the current password is incorrect or the new passwords do not match

Use Case : ViewItemList
ID : 7
Brief Description : Administrator/Employee views item list
Primary Actors : Administrator, Employee
Secondary Actors : n/a
Preconditions : User is logged into the system
Main Flow : The user is at the workspace app panel The user clicks on the inventory app icon The user is directed to a webpage which shows items in the database in a table
Postconditions : User is displayed with list of items on a table
Alternative Flow : n/a

Use Case : CreateItem
ID : 8
Brief Description : Administrator/Employee creates item
Primary Actors : Administrator, Employee
Secondary Actors : n/a
Preconditions : User is logged into the system

Use Case : CreateItem

Main Flow :

The user is at the workspace app panel
The user clicks on the inventory app icon
The system shows list of item
The user clicks "Create" button on the top left corner
The user fills in item information
The user clicks "Save" button on the top left
The item created is displayed

Postconditions :

The new item is created and saved in the system's database

Alternative Flow :

n/a

Use Case : DeleteMultipleItem

ID : 9

Brief Description : Administrator/Employee deletes items

Primary Actors : Administrator, Employee

Secondary Actors : n/a

Preconditions : User is logged into the system

Main Flow :

The user is at the workspace app panel
The user clicks on the inventory app icon
The system shows list of item
The user selects item which user wants to delete by ticking the box on the left of each item
The user click "Delete" on the top left
The user is prompted to confirm the deletion
The items are deleted from the database and page is refreshed

Postconditions :

Selected items are deleted from the database of the system

Alternative Flow :

n/a

Use Case : PostStatus
ID : 10
Brief Description : Administrator/Employee post status
Primary Actors : Administrator, Employee
Secondary Actors : n/a
Preconditions : User is logged into the system, the Twitter and Facebook account
Main Flow : The user is at the workspace app panel The user clicks on the connecting to the public app icon The system displays a textbook for posting status and checkboxes to social networks The user writes something in the textbox The user selects social network to post the status The user clicks post A message to display status has been successfully been posted
Postconditions : Status is posted to the social network ticked
Alternative Flow : The user clicks post Error occurs if textbox is empty

Design

Architecture

Our conception of the architecture of the entire system was based around a slimmed down version of Microsoft Exchange Server, whereupon the employees could rely on a single server to manage everything related to the day to day running of the organization, from system authentication and email, to stock tracking and payment processing.

The original vision was to have a single server backend with multiple external interfaces (some internal services make use of the same interfaces, e.g. webmail):

- HTTP
 - Staff access to a secure portal that would provide webmail and account/stock management.
 - Customers access to a web store, which would utilize the same database backend. Ideally, this would share the same URI as the staff portal, but only display functionality designed for customers.
 - Authenticated WebDav interface for file, contacts, and calendar CRUD operations.
 - Authenticated custom-designed REST interface for use with dedicated mobile apps (both for staff and customers).
- SMTP/POP3/IMAP
 - Staff access to email using external clients of their choice.
- Kerberos
 - Single Sign-on integrated for certain workstations (possible an RPi as a thin client).

Data Stores

The original intention was to store as much as possible on the LDAP server, since that would allow the most compatibility with external systems. However, the RFC for LDAP specifies certain schemas, and as much as we wanted to offload data from the SQL server to LDAP, either 1) the data was better served in a relational database, or 2) the predefined RFC schemas available limited what data could be stored.

SQL

MySQL was chosen over other SQL implementations because almost every library, script, and PHP package had support for MySQL. Indeed, PHP and MySQL often go hand in hand, and it was felt that this would provide maximum compatibility. Furthermore, since the introduction of the mysqli extension, pairing PHP and MySQL has become a much better option than competing bundles from a coding, security, and performance perspective.

For the bulk of our data, notably those relating to the stock/transactions, we chose to store it in a MySQL database. Since individual orders, transactions, and inventory data are inherently interrelated, it would be unsuitable to move them into the LDAP server.

LDAP

We decided that LDAP would serve three main purposes – store the common address book, store employee data, and provide the backend authentication (using password lookups).

For the address book, the LDAP would store the contact details for each customer and supplier (treated as a 'contact', to allow for the fact that people can both be suppliers and customers). To keep track of customer details when dealing with orders and transactions, they would be linked to the database by use of a unique numeric 'contactID' in the database, which would correspond to a unique uid number in the LDAP directory.

For employee details, we took a similar approach, in that the individual details of every employee was stored in the LDAP server, and where data had to be 'joined' with tables within the database, we used a uid (also generated by MySQL auto incrementing the primary key) to keep track of employees in the database and match it with their details in the LDAP directory.

Using LDAP to store password details and authenticate users was the original aim of using LDAP at all. In the previous project where we worked on an RPi thin client, we briefly explored using SSO (despite not implementing it at the time). This time round, we wanted to work towards that. The first stepping stone was using an industry standard method of authenticating users, so users could log in using the same set of credentials.

Members of our team had seen SSO in operation before, for example on the UCL single sign on (used to log in to UCL library services, student union pages, and so on). In addition to SSO, there are services which doesn't natively support SSO across different technologies and protocols, such as logging onto a client computer (e.g. students are required to log in once onto a workstation, then log in again at a web interface for email etc.). It is for these use cases that, despite anticipated hurdles, we aimed to get LDAP running to provide a single point of authentication for email (mail servers, ERP, and external packages such as OwnCloud).

Mail

Briefly before we started working on the mail server, we explored our options. The easiest method would be to outsource it, but since we had control of our own VPS and domain name (bizwebsys.tk), there was little reason why we couldn't install our own mail servers and use our own domain name.

The email system consists of the MDA, MTA, and the mail client. The MTA does the sending and delivering using SMTP (and LMTP). The MDA does the storing of the mail until it's picked up by the client. The mail client can be web based, using SquirrelMail or RoundCube, or as we envisaged it, on a desktop/mobile phone if the individual employee wished.

Numerous options were available, including Postfix, Courier, Dovecot, and Exim (just a selection of the most commonly used MTA and MDAs). Although the original intention was to try a few to see which was the best, because the configuration and debugging was so time consuming, we just went with the first ones we tried, which happened to be a combination of Postfix, Dovecot, and Roundcube (prettier than SquirrelMail).

Cloud

In our project, 'cloud' primarily encompasses file storage and calendar. Having reviewed OwnCloud previously, it was believed that this would offer the most functionality for free, checking all the boxes. Although it had dawned on us that despite the software being FOSS, there was little chance we would actually be able to modify the source code for our ends given our time and ability constraints, as idealized previously. However, choosing OwnCloud to power the file storage and personal calendar by providing a

WebDav interface would meet the requirements, allow users to login with the same set of credentials using the LDAP backend, and potentially open up further opportunities for integrating in the future.

The original intention was to use OwnCloud to handle files, calendar/tasks, and manage contacts. The file storage was designed to be standalone from the rest of the system, using OwnCloud's own built in functionality to handle any file opening, image preview etc. The calendar, however, would 'synchronize' with a common calendar the main ERP system by means of a dedicated iCal stream listing payment due dates, delivery dates and notable events. However, this proved prohibitively difficult due to the time constraints and the fact that the OwnCloud was designed to be a purely 'personal' calendar.

With regards to the contacts, the brick wall was that while OwnCloud supported LDAP-based authentication, it doesn't allow for fetching public contacts via LDAP. While RoundCube supported manipulating contacts via LDAP, it was also extremely buggy and didn't compensate for the uid that was required to maintain a link between contacts in the LDAP directory and other records in the ERP database. The solution we ultimately worked towards was implementing the contacts CRUD operations in the ERP, rather than would be more appropriate in the 'cloud' section.

ERP Session Control

Session is used for login control, store user data and reset password.

Login control

After user login, data below will be stored in session:

username	string
email	string
employee_id	string
is_admin	boolean

php scripts can retrieve session data by:

```
$this->session->userdata( 'username' );
```

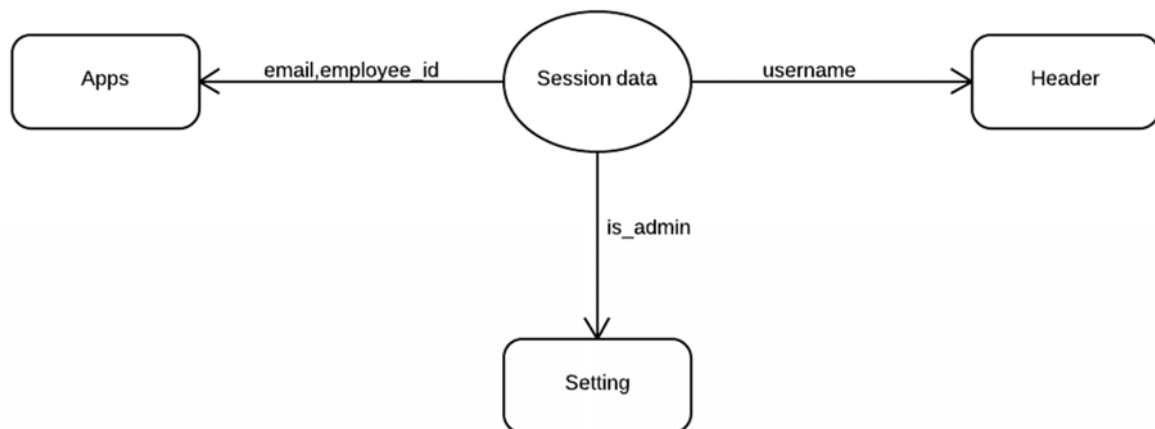
If user is logged in, then proceed to page. If not, redirect to home page.

User data

Header retrieve username from session data and display on screen.

Header nav retrieve is_admin from session data to determin whether to display setting panel.

Apps retrieve email and employee_id for various uses.

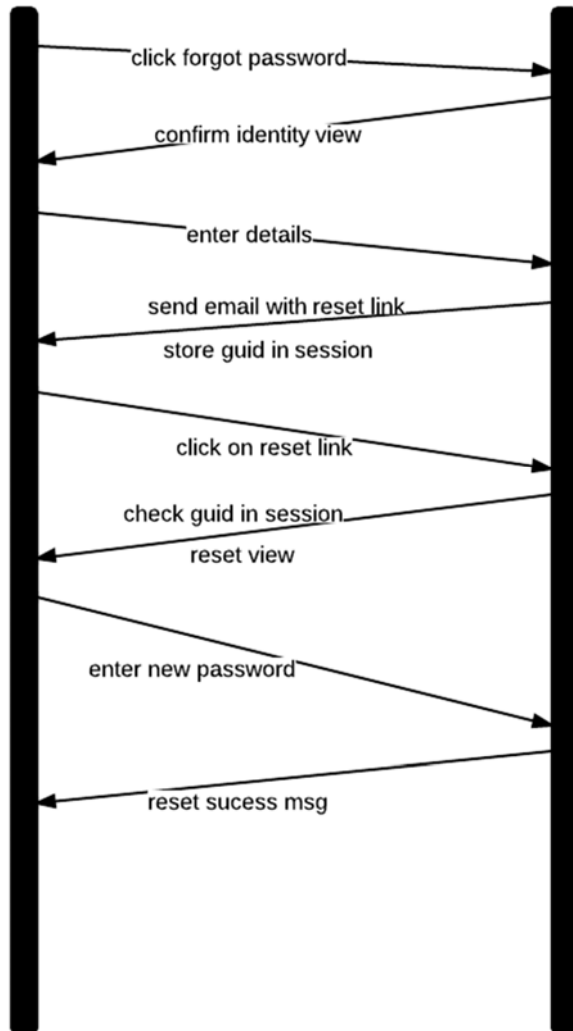


[Reset password](#)

Below shows the process of user reset their password. Session plays an important role in this process. After validating the user identity, system will generate a guid and store it in session data. Then a reset url with guid embeded in it will be sent to user email. When user clicks on the reset link, system will compare the guid in url with the guid in session. If unmatched or session has expired, then system will display error message. This enforce the user to use the latest reset link, and click on reset link within a certain amount of time.

User

System



The code below shows server generate a guid and store it in session:

```
$guid = com_create_guid();
$guid = substr($guid, 1, strlen($guid)-2);
$newdata = array(
    'guid' => $guid,
    'uid'   => $uid
);
$this->session->set_userdata($newdata);
```

The code below shows the server check guid against session data:

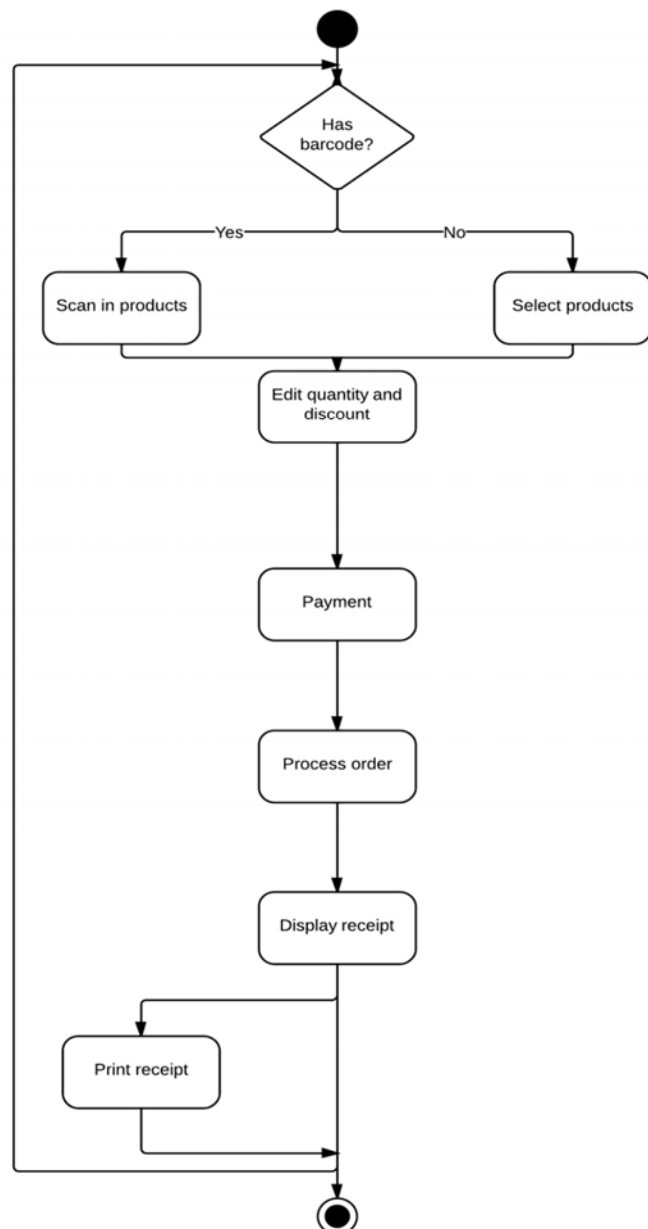

```

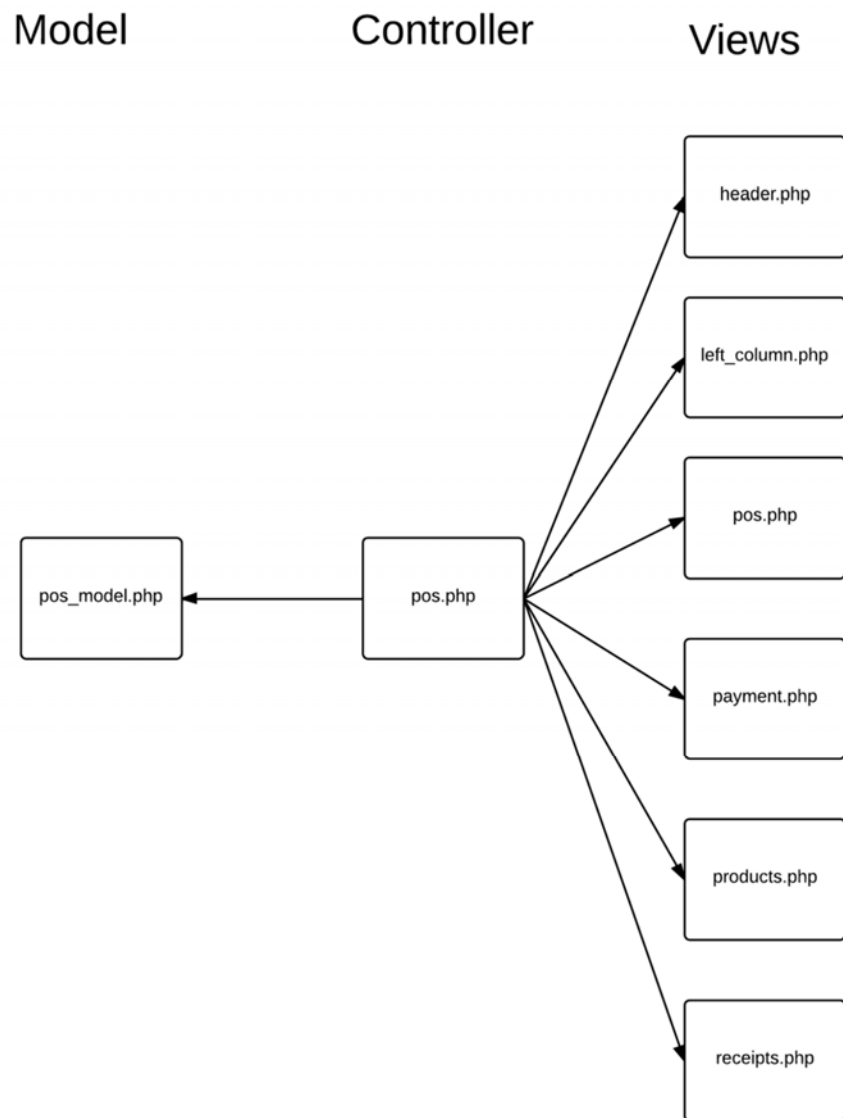
$session_guid = $this->session->userdata('guid');
if($session_guid!=$guid)
{
    echo "Link has expired or is invalid.";
    return;
}

```

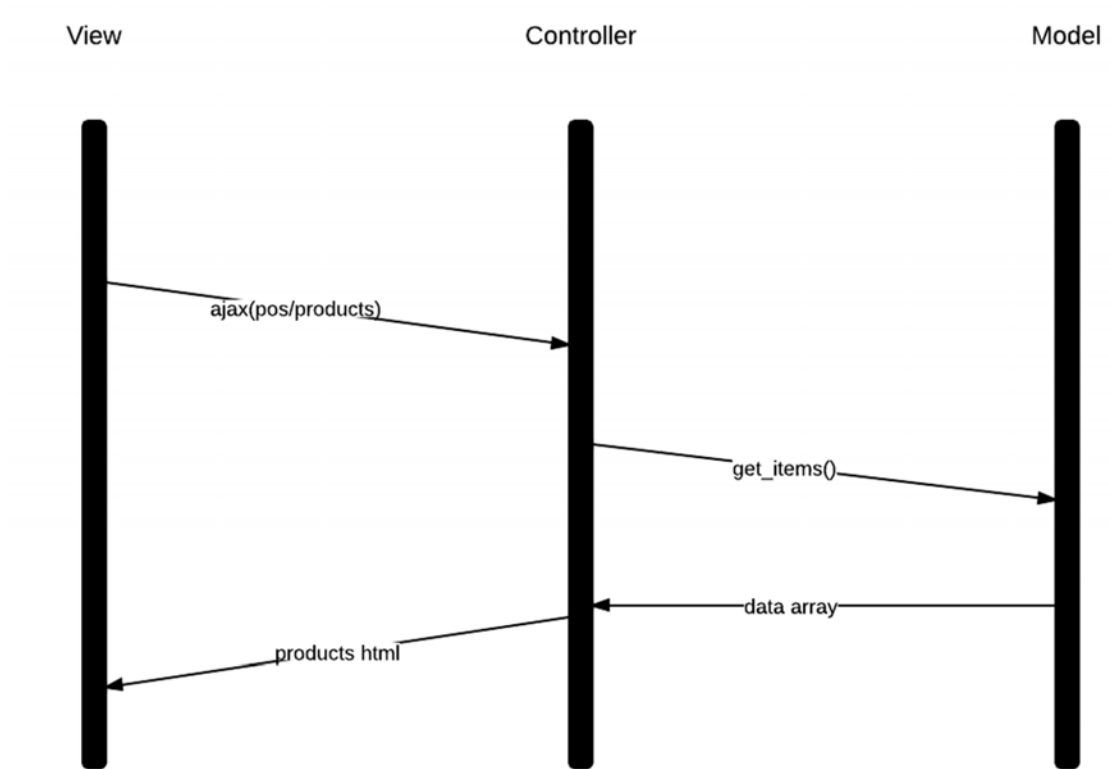
POS

POS is a sales terminal used during store selling. It is designed for touch screen. Sales person can scan in all the product barcode for an order, and produce a receipt for customer. Alternatively, instead of scan in product barcode, salesman can also select the product from screen.

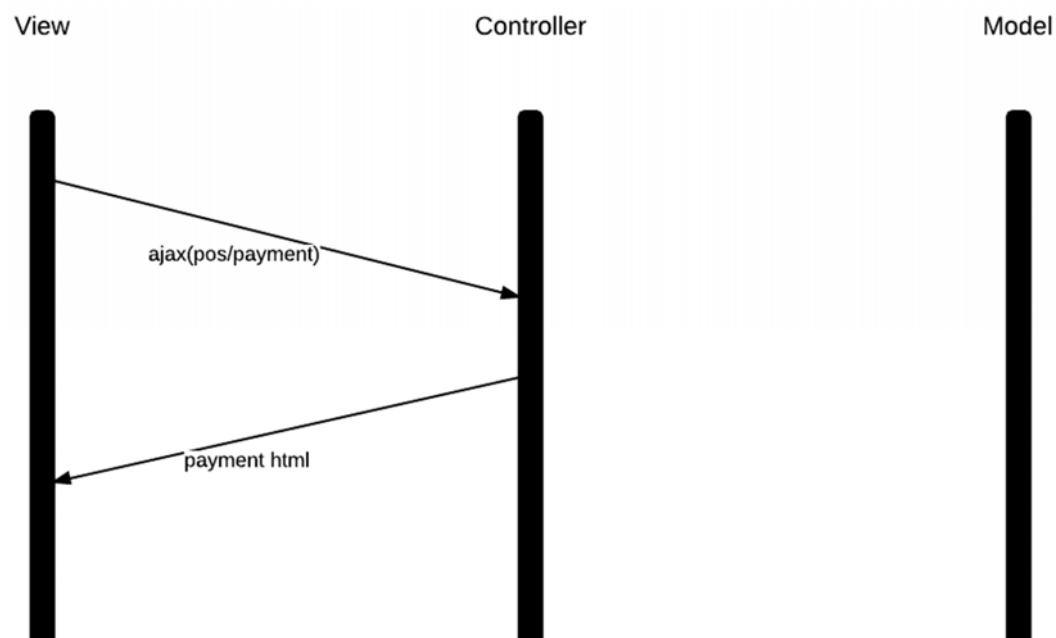




Show products - View uses jQuery to send an ajax request to POS controller. On receiving the request, the controller gets data of items from POS model, and returns products html to view.

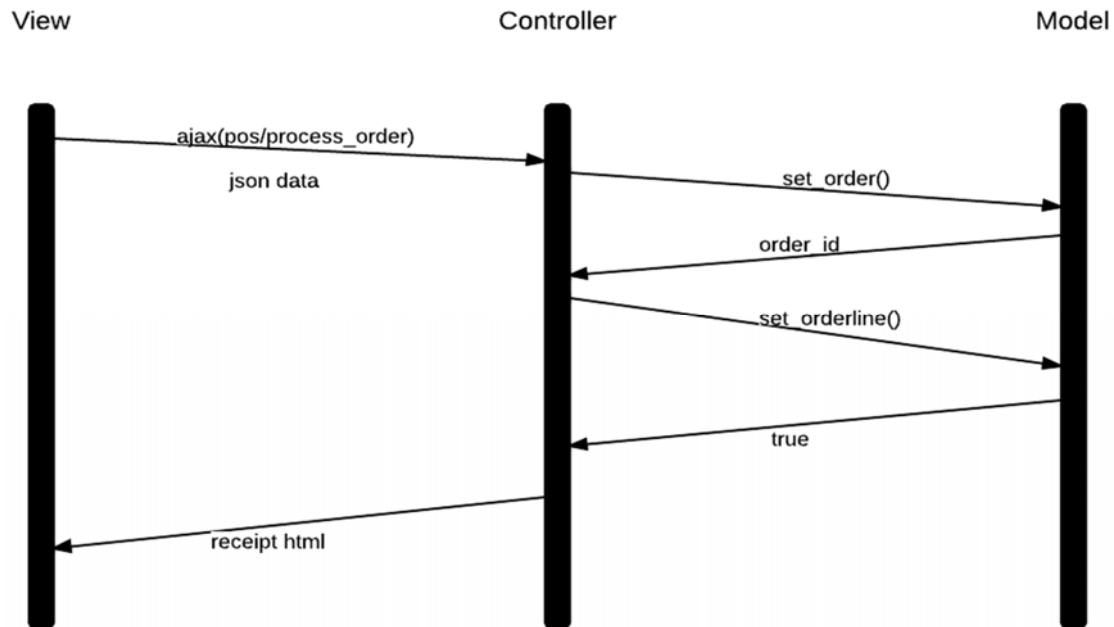


Show payment - View uses jQuery to send an ajax request to POS controller, together with data 'total'. On receiving the request, the controller returns payment html to view.



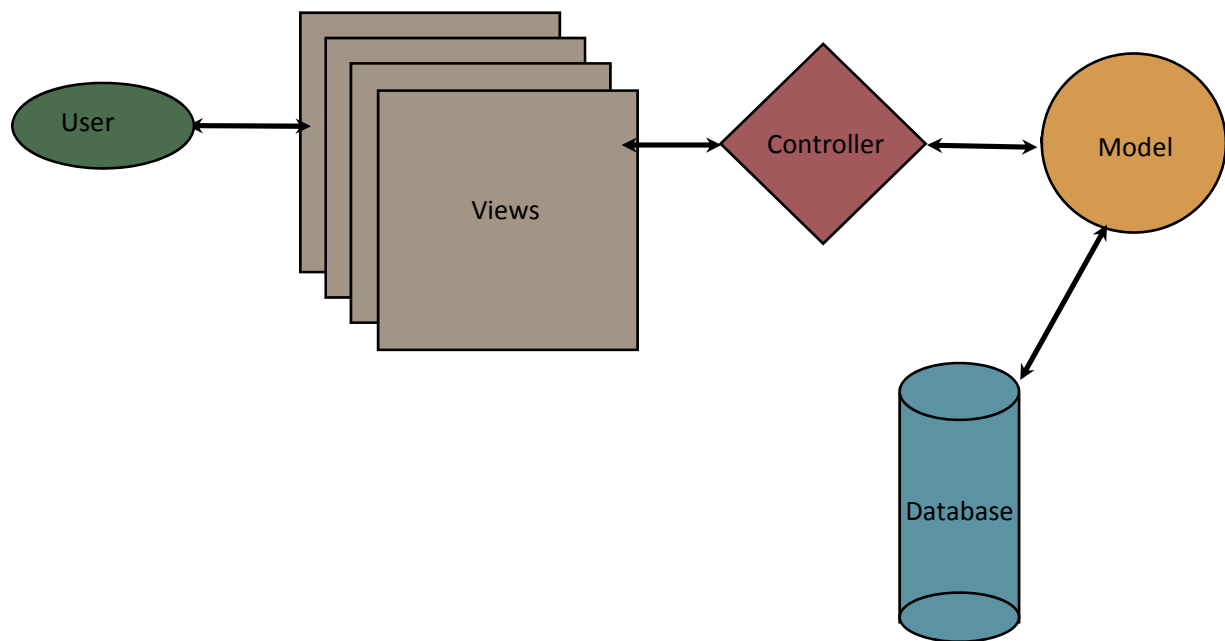
Process order - View uses jQuery to send an ajax request to controller, together with json data including array of 'items' and 'cash'. On receiving the request, the controller calls `set_order()` in POS model which

insert a new order record in SalesOrder table. The model returns the orderId of new order. Then controller calls set_orderline() in POS model which insert details of items of the order. If the process is successful, model returns true to controller. And then controller returns the receipt html to view.



Inventory

The inventory app follows the model-view-controller structure. Overall, there are four views components for the inventory, one controller and one model. The diagram below shows the architecture of the app. The four views include index item list, which is also the index page for inventory app, create item, edit item and display item. The index view shows a list of items, while the rest show each item in a form views.



The table below shows the Inventory table in our database.

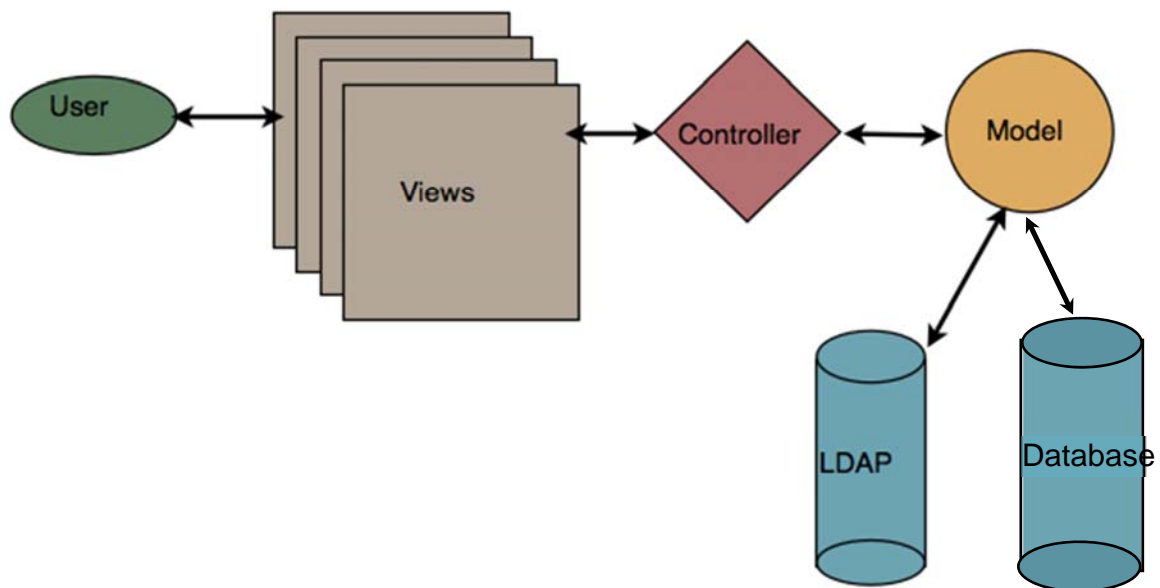
Field	Type	Length	Unsigned	Zerofill	Binary	Allow Null	Key	Default	Extra
ItemID	INT	10	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	PRI		auto_incr...
ItemType	VARCHAR	100	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>		NULL	None
ContactID	INT	10	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	MUL		None
SKU	VARCHAR	45	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	UNI	NULL	None
Name	VARCHAR	100	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>		NULL	None
Description	VARCHAR	200	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>		NULL	None
Stock	INT	10	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>		NULL	None
StockROP	INT	10	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>		NULL	None
GTIN	CHAR	14	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	UNI	NULL	None
Imagepath	VARCHAR	100	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>		NULL	None
Cost	DECIMAL	10,2	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>		NULL	None
NetPrice	DECIMAL	10,2	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>		NULL	None
VATRate	DECIMAL	10,2	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>		NULL	None
DiscountRate	DECIMAL	4,3	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>		1.000	None

- ItemID - The primary key for the table. A unique identification for each item record.
- ItemType - Specifies the type of item, such as Toys, Cars, Beauty and others.
- ContactID - Stores the contactID of the supplier for a particular item.
- SKU - Stores the unique identifier to a particular stock keeping unit.
- Name - Specifies the name of item.
- Description - Stores description for the item.

- Stock - Specifies number of stock.
- StockROP - Specifies the stock re-order point.
- GTIN - Stores the Global Trade Item Number.
- Imagepath - Stores the file name of the picture attached to this item.
- Cost - Specifies cost of the item.
- NetPrice - Specifies the net price of the item.
- VATRate - Specifies the vat rate of the item.
- DiscountRate - Specifies the discount rate of the item.

Contacts and Employees

The architecture for contact and employee apps are very similar except that they connect to an ldap server and a database. A record with only ID is created in the database tables. Each of the app has four views. The index view shows the list of contact or employee in thumbnails, a view to create new contact or employee, a view to display contact or employee and a view which edits contact or employee. The diagram below shows the structure of these apps.



For the LDAP server, there are two organization units created to hold the records for contacts and employee. Contacts are being referred to as contacts, whereas employees are being referred to as people in the directory. The lists below show the schemas for each organization unit.

Field	Type	Length	Unsigned	Zerofill	Binary	Allow Null	Key	Default	Extra
ContactID	INT	10	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	PRI		auto_incr...

Contact table in database

Field	Type	Length	Unsigned	Zerofill	Binary	Allow Null	Key	Default	Extra
EmployeeID	INT	10	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	PRI		auto_incr...

Employee table in database

Structure for Contacts in LDAP (ou = contacts)

```
*cn - Name
*sn - Surname
uid - contactID (from MySQL)
givenName - First Name
facsimileTelephoneNumber - Fax
telephoneNumber - (Work) Phone
mobile - Mobile Number
street - Street
st - County
l - Country
postalAddress - Delivery Address
postalCode - Postcode
jpegPhoto - Profile photo
mail - Email
o - Organization
```

Structure for Employees in LDAP (ou = people)

```
*uid - Username
*cn - Name
*sn - Surname
*givenName - First Name
employeeNumber - userID (from MySQL)
homePhone - Home Phone
mobile - Mobile Number
l - Country
postalAddress -Address (full address including street names, city)
postalCode - Postcode
jpegPhoto - Profile photo
mail - Business Email (derived from username)
title - Job Title
userPassword - Password (auto-generated based on username)
```

*Compulsory fields

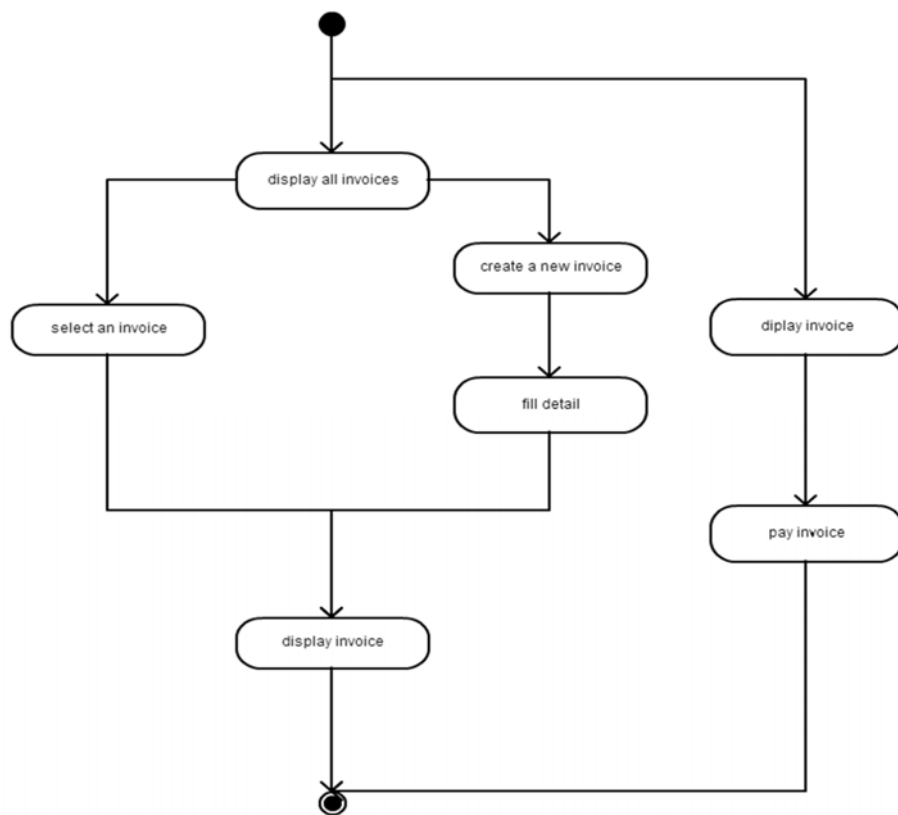
Sales and Purchases

Sales/Purchases part is designed to organize sales invoices for customers and purchases invoices for suppliers for the business. Employees can create, edit and record invoices for customers. Alternatively, they can also create, edit and record invoices for suppliers. And pay to suppliers as well.

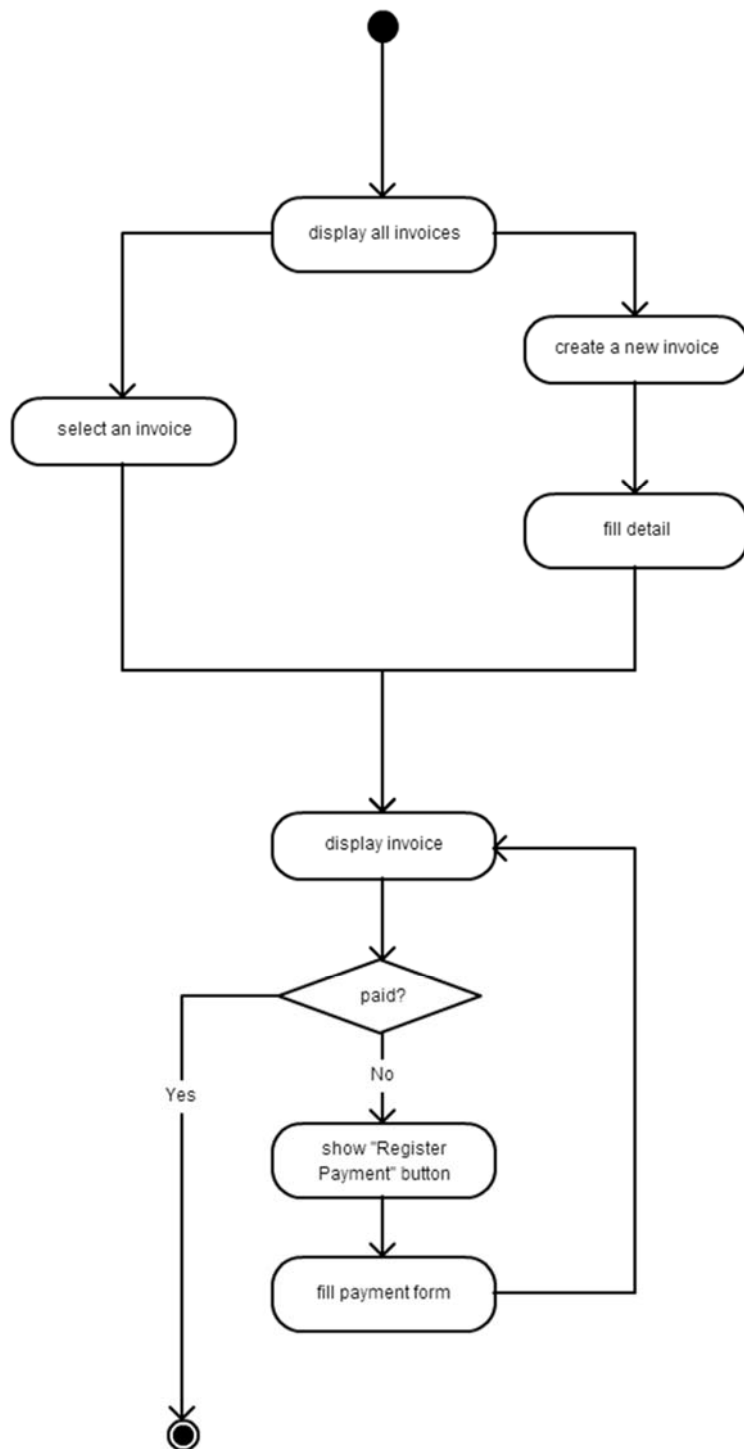
Architecture

In the Sales/Purchases panel, there are two components, sales and purchases. In sales part, it displays all the orders. It can create a new invoice and make a payment of it.

Purchases



Sales



Details

Sales/Purchases part is used PHP to get data from database to display. And the dynamic part, for example, add new sales invoice line in the new sales invoice, is using JavaScript.

Display all sales invoice

display orders from `display_order_by_id()` in sales controller, which is from `contact_list('cn')` in `contacts_model`, `employee_list('cn')` in `employee_model` and `get_orders()` in `sales_model`.

View sales invoice

Get data from `view_invoice()` from sales controller, which data is from `contact_list('cn')` in `contacts_model`, `employee_list('cn')` in `employee_model` and `get_orders()` in `sales_model`. For to pay the invoice, using `update_sales_payment()` in sales controller to update.

Display all purchases invoice

Get display orders in `display_cust_order_by_id()` in sales controller. `contact_list('cn')` in `contacts_model`, and `get_cust_orders()` from `sales_model`.

Social Networks

At first, we planned to create an application which let users post a status (or a 'tweet') onto the three popular social network: Facebook, Twitter and Google+. Therefore, I had to find the corresponding APIs for each of them and a protocol that allows the application to posting to the social networks.

Production Architecture

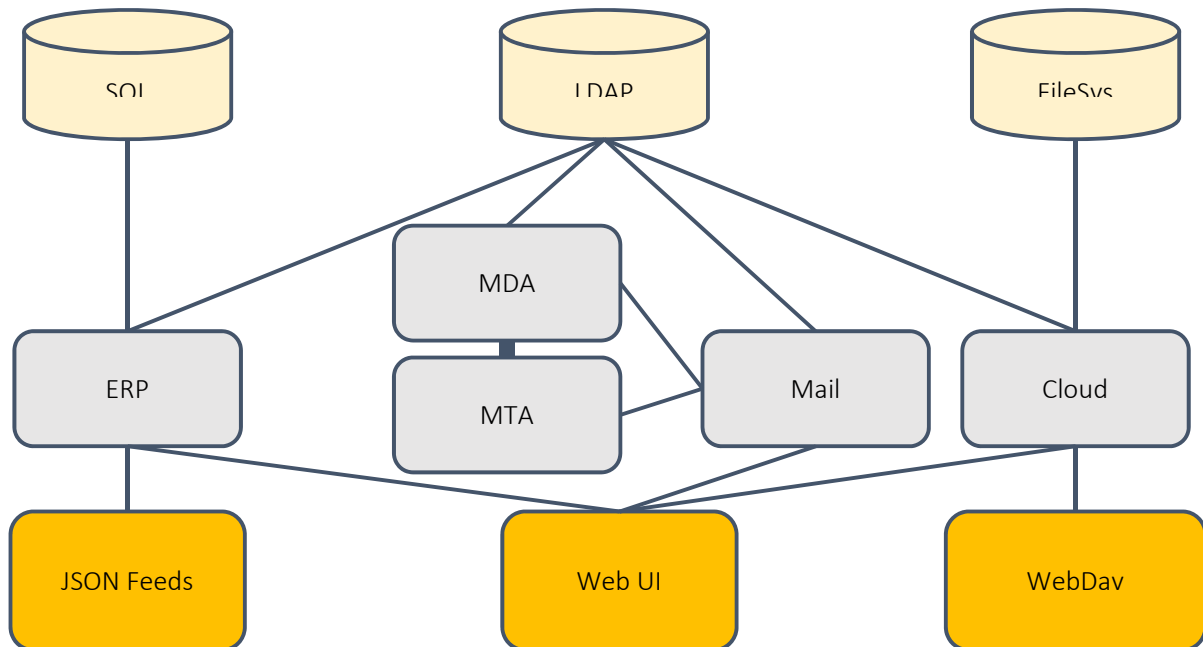
We used a simple form format to let users input the content that they wished to post onto the social network; and also three check boxes (Facebook, Twitter and Google+) that allow them to check one or more for choosing the social networks that they want to be posted on. Based on the checked boxes, the application should initialise the corresponding APIs. For Facebook and Twitter, I used the official APIs for posting. However, after some researches, I found out that the official API for Google+, it can be only used for reading but not posting. Therefore, the Google+ part has been taken out of the application.

Component Description

Except the Facebook and Twitter APIs that I have mentioned above. A protocol is necessary for the Facebook and Twitter sides know that who is posting onto their network which can also prevent the third side to knowing the account information (such as account and password) with a limited scope of information from the social network. For processing the values that the users type in (the status content and the check boxes) in the form, I write it in JavaScript. The Facebook API is also written in JavaScript, therefore, it does not require other more. However, the Twitter API is written in PHP. It means that I have to find a way to pass through the data (the status content) from the JavaScript part to the PHP part. I chose the Ajax to do this job that passes the data from JavaScript to PHP.

Implementation

Architecture



Authentication

For the sake of simplicity, rather than attempting to log in to LDAP using each user's credentials, the ERP logs in as an administrator, and binds according to the username and password. This is similarly the case with OwnCloud, RoundCube, and the mail servers.

We attempted to use Kerberos (SASL) for non-HTTP SSO. That would have allowed us to go one step further, integrating with an RPi linux operating system. However, difficulties as outlined in the blog (fairly extensively) as well as time constraints prevented us from pursuing this further.

The equivalent for HTTP services SSO is SAML, as is used by Google Apps to integrate with existing services. However, this again proved unfeasible as RoundCube and OwnCloud doesn't make provisions for this, and with the additional testing and extra code/extensions we needed to write (both as a SAML provider and to integrate the external packages we used), the proposition was not worth the time and effort.

Data Stores

The storage of data generally followed that of the original design. However, the actual attributes of the data stored was always in flux.

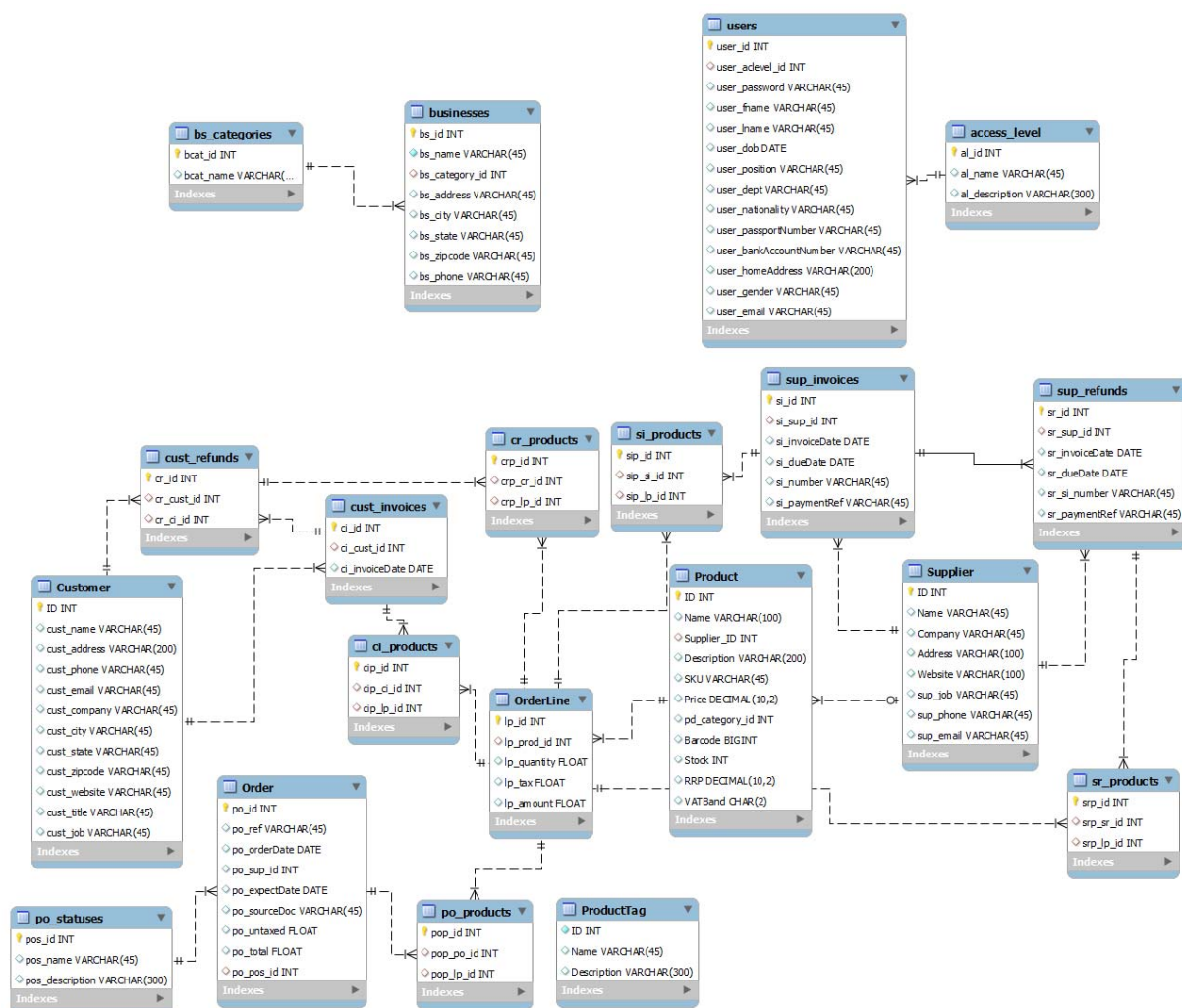
LDAP

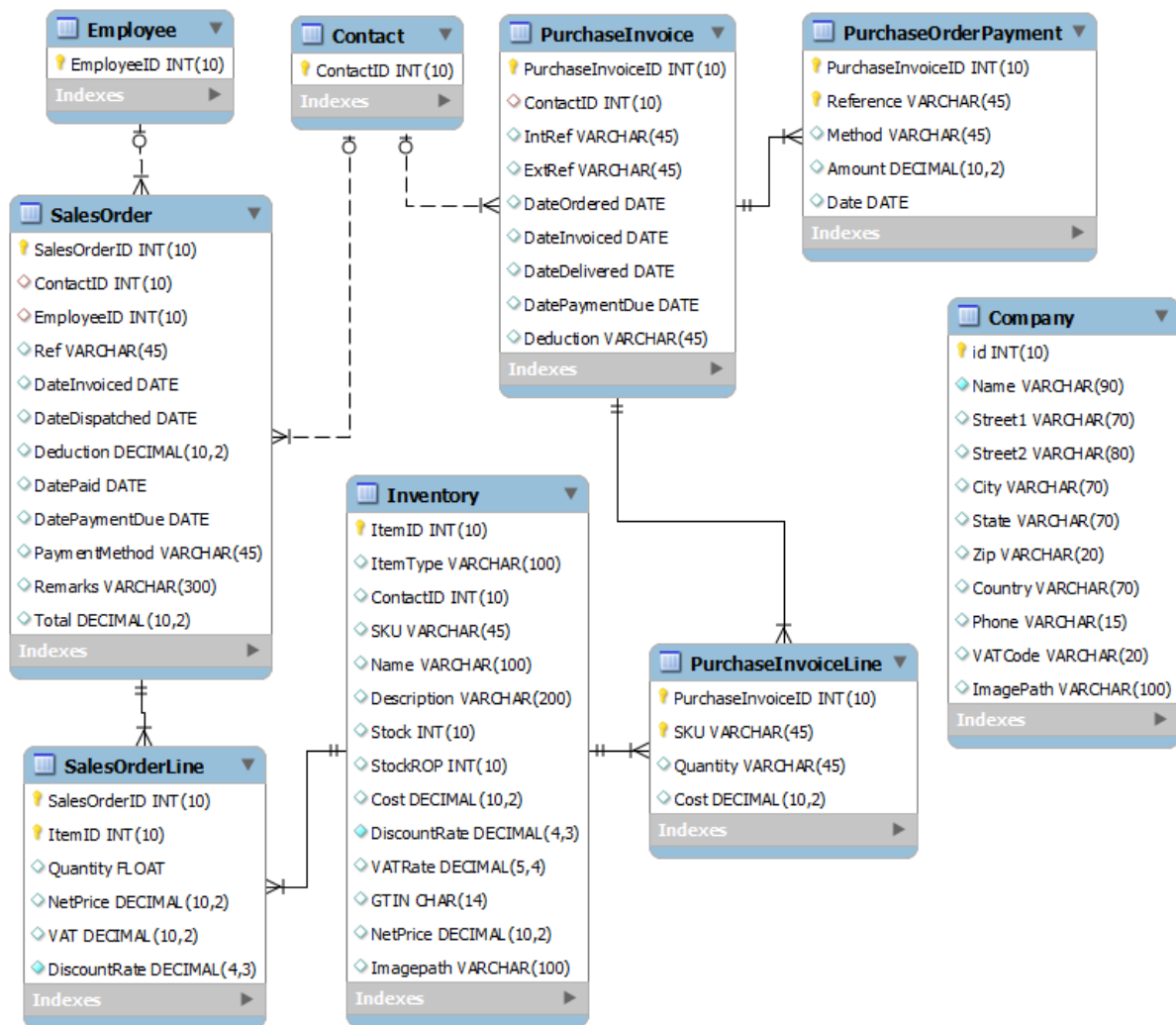
The choice of the LDAP server was a no brainer – OpenLDAP was by far the most common distribution. The attributes used for the contacts and employees were settled upon fairly late, as we encountered problems such as the choice of ObjectClass being 'person' or an 'organization', and how that would

impact the attributes we could store. The final attributes used are detailed under the respective sections on Contacts and Employees.

SQL

The original design of the SQL schema was derived from OpenERP, a model which we originally based the ERP on. Before we realized the difficulties of making a single deployment of our project a SaaS in itself, we'd hoped to be able to host multiple organizations on a single server. The original design of the database was by far the most comprehensive, before we worked towards streamlining it and making accommodations for LDAP. Listed below are the before and after ER diagrams.





Mail

Servers

Dovecot was used as the MDA. Since we weren't dealing with actual linux accounts, we used virtual accounts for each mail user, which were generated by probing (password lookups, as opposed to authentication binds) the LDAP server whenever a mail was delivered.

Postfix was chosen as the MTA. It is integrated with Dovecot for three purposes:

- By the nature of the SMTP protocol, anyone can send from an MTA and they are thus prone to spammers. Usage of the MTA must therefore be limited to certain (i.e. authenticated) users. Dovecot acts as an SASL provider in this instance, and instead of postfix connecting to LDAP to check if a user has permissions to send mail, it checks with Dovecot first.
- When Postfix detects mail are destined for our predefined domain (since we are using only one domain, bizwebsys.tk is defined as the canonical domain), instead of checking if the user exists via LDAP as would normally be the case, it forwards them to Dovecot anyway, probing Dovecot

for that user's existence (the details of which is irrelevant to Postfix). If Dovecot throws an error message, only then is the mail bounced.

- Postfix delivers mail to Dovecot via LMTP (the advantages of LMTP over LDA, which is also available for Dovecot/Postfix, will not be explained here).

Webmail Client

For sending and receiving/reading mail, RoundCube connects to Postfix and Dovecot respectively, both of which ultimately authenticates via the LDAP server. Although RoundCube doesn't authenticate users with the LDAP directory, it uses the LDAP directory (used by the ERP) for contacts info. In addition, an address book called 'Internal Contacts' was included, which has a similar functionality, but instead of listing contacts, it allows users to email their fellow colleagues.



Cloud

With regards to OwnCloud, our chosen 'provider', the only area where it was useful was providing a place for file storage. The calendar function, while available for the individual user, is far from the integrated solution (including common dates) that we envisaged. While this nominally meets our requirements, it's primarily used as proof of the extensibility of our architecture.

POS

The challenging part of POS implementation is item list. Items scanned in or selected will go onto item list, user can select a specific item from the list and edit its quantity and discount rate. To fulfill these functionalities, we implement an item object as below. The items on item list are stored in an array.

Item object

Every time an item being selected (or scanned in), a new item object will be created and added to items array. And then JavaScript method `renderItemList()` will be called to refresh the item list.

When user selects an item from the item list and edit its quantity and discount rate using keypad, the quantity and discount attribute of the selected item object will be edited. Then `renderItemList()` will be called to refresh the item list.

item
+ ProductID:int + name:string + NetPrice:double + VATRate:double + stock:int + quantity:int + discount:double + selected:bool + QuantityString:string + DiscountString:string
+ getDisplayPrice:double + getTax:double + getTotal:double

Ajax and Json

When it is time to process the order, client side will send an Ajax request with Json data to controller.

Client side JavaScript: send Ajax request. Use Json format to send array of item objects. On server returns, it renders the html to content panel.

```
function saveOrderShowReceipt()
{
    var cash = parseFloat($('#input_cash').val());
    var jsonItems = $.toJSON(items);
    $.ajax({
        url: '\'. site_url('pos/process_order') .'\',
        type: \'POST\',
        data: {items:jsonItems,cash:cash},
        success: function(response) {
            //load receipt to content
            $('#content').html(response);
            renderReceiptItemList();
            receiptButtonFunc();
        }
    });
}
```

Server side php: decoding Json data.

```
$items = json_decode($_POST["items"]);
```

Print function

The print function is used to print receipt for customer. The function write a new html file of receipt to a popup window and print it.

```
$('#btn_print').click(function() {  
    //var prtContent = document.getElementById('receipt');  
    var html="<html>";  
    html+="<head>";  
    html+="<style type='text/css'>.small-font {font-size:12px;  
    html+="</head>";  
    html+= document.getElementById('receipt').innerHTML;  
    html+="</html>";  
    var WinPrint = window.open('', '', 'left=0,top=0,width=  
    WinPrint.document.write(html);  
    WinPrint.document.close();  
    WinPrint.focus();  
    WinPrint.print();  
    WinPrint.close();  
});
```

The challenging part of POS implementation is item list. Items scanned in or selected will goes onto item list, user can select a specific item from the list and edit its quantity and discount rate. To fulfill these functionalities, we implement an item object as below. The items on item list are stored in an array.

Item object

Every time an item being selected (or scanned in), a new item object will be created and added to items array. And then JavaScript method `renderItemList()` will be called to refresh the item list.

When user selects an item from the item list and edit its quantity and discount rate using keypad, the quantity and discount attribute of the selected item object will be edited. Then `renderItemList()` will be called to refresh the item list.

item
+ ProductID:int + name:string + NetPrice:double + VATRate:double + stock:int + quantity:int + discount:double + selected:bool + QuantityString:string + DiscountString:string
+ getDisplayPrice:double + getTax:double + getTotal:double

Ajax and Json

When it is time to process the order, client side will send an Ajax request with Json data to controller.

Client side JavaScript: send Ajax request. Use Json format to send array of item objects. On server returns, it renders the html to content panel.

```
function saveOrderShowReceipt()
{
    var cash = parseFloat($('#input_cash').val());
    var jsonItems = $.toJSON(items);
    $.ajax({
        url: '\'. site_url('pos/process_order') .'\',
        type: \'POST\',
        data: {items:jsonItems,cash:cash},
        success: function(response) {
            //load receipt to content
            $('#content').html(response);
            renderReceiptItemList();
            receiptButtonFunc();
        }
    });
}
```

Server side php: decoding Json data.

```
$items = json_decode($_POST["items"]);
```

Inventory

Four main methods are written in the controller which render views to the four views which are index, create item, display item and edit item. Some methods have two parts where first part being the javascript code to be attached to the view and second part being the php code. There are also other methods which handle deletion, validation and verification. Besides, inventory model and contacts model are also loaded when a controller is instantiated.

For the index method, the javascript included mainly handles the checkboxes in the table which lists the items in the database and also the delete button which handles the deletion of items. When the delete button is clicked, it will post an ajax data to a php method in the same controller. The ajax post contains the list of items to be deleted. The rest of the index method makes preparation in order to render the index webpage, which includes getting the item records from the database using inventory model.

For the new item method, we followed the CodeIgniter tutorial practices while writing the method for the form to create new item. Form validation is done by using CodeIgniter's library methods. Our own validation methods are also written in order to carry out certain validation and verification. For an instance, the "gtin_check" method. It is written to compare a new item's gtin number with the existing records and if any existing record has the same GTIN, an error will occur. The method also supports image upload. If the user clicks "Save" and there are no validation or verification errors, the new item will be displayed using display item method.

The display item method has one parameter and it is the item ID of an item. Using the item ID, the method calls a model method to retrieve the record from the database. A call to contacts model is also being performed to retrieve the list of suppliers. In addition, some javascript is written to enable stocking up process to be done easily as well as to enable the deletion of the particular item being viewed.

Finally, the edit item method does two things in general. Firstly, it also takes in an argument, which is the item ID of an item to be edited. Using that argument, it displays a modifiable form to the user with current data of the item in the form. Validation and verification rules are also set similarly to the create new item method. However, the GTIN check and SKU check have to be implemented differently because it already has its own GTIN and SKU. Thus, when "Save" button is clicked, the GTIN check edit and SKU check edit methods will first identify if the GTIN and/or SKU has been modified, if they haven't been, then no check is necessary. Lastly, if the validation rules are passed, an inventory model is called to update the item in the database.

In the inventory model, the methods are generally written to carry out simple CRUP operations on the database table. For example, get the list of items, get item by id, update stock, update imagepath and delete item.

Contact and Employee

Contact and employee has rather similar implementations. Each app has four views in total and their operations are similar. One index view for rendering the main page that displays thumbnails of contacts or employees, one view for creating new contact or employee, one for displaying purpose and one for editing purpose.

The contact and employee controllers also have similar methods. The difference in implementation will be discussed later. For index method, we use the contact/employee model loaded to call a function which retrieve all contact/employee records in the LDAP. The function returns an array of records, where the first item in the array represent number of records in the array. This array is then passed to the index view to be populated on the screen. Each row on the content pane has 4 thumbnails. Using bootstrap css, this layout is also responsive in multiple dimensions. It works fine on mobile devices as well as tablets.

```
<?php foreach ($contacts as $k => $contact): ?>
    <?php if (!is_array($contact)) continue; ?>

    <?php if ($x % 4 == 0): ?>
        <ul class="thumbnails">
    <?php endif; ?>

        //code to populate the thumbnails with picture and names
etc.

    <?php $x = $x+1; ?>

    <?php if ($x % 4 == 0 || $x == ($contacts["count"])): ?>
        </ul>
    <?php endif; ?>

<?php endforeach ?>
```

Here is a snippet of code on how the layout is designed in the index view of contact. Employee index view has similar codes.

In both controllers, there are also methods which handle create new, display and edit contact/employee. There are handled the same way as inventory. The difference comes in the model implementations of contact and employee in comparison to the inventory model.

The inventory model carry out simple CRUD operations based on CodeIgniter Active Record class. Details of functions can be found at:

http://ellislab.com/codeigniter/user-guide/database/active_record.html

However, in contact and employee model, the model connects to database and also to an ldap server. The model loads MyLdap library when instantiated. In methods where ldap connection has to be established, it will instantiate a new connection to ldap using the constructor of MyLdap library. The model also involves the use of database because the Contact and Employee table keep track of the Contact ID and Employee ID. The tables are only modified when creating a new contact/employee or deleting a contact/employee. First, a record in contact/employee table is created before creating a new record in LDAP.

The challenges and difficulties when developing these apps also occur at the model side. LDAP does not support null values and therefore, the form must be checked when creating a new record as well as

modifying an existing record. When creating a new contact/employee, not all fields are compulsory, therefore we must check whether the post is true or false before putting them into the ldap directory as ldap does not accept null values. Also, during the modification of a contact/employee, if any field is left blank, the corresponding attribute existing in the ldap directory must be deleted. The update of ldap records actually involves deletion of attribute and possibly creation of attributes.

Sales and Purchases

Caused the limited time, unfortunately, Sale&Purchases part is incomplete.

In the sales part, it is incomplete. In the displaying sales orders view, the dispatched button doesn't work as expect. When click dispatch button, it can change to text loading...., jump an alert to confirm, change the dispatched time in database SalesOrders, but it cannot display the updated dispatched date in the table.

In create new sales invoice view, there is missing function in adding rows to new sales invoice. Adding row is used to add a new sales row, including item name, quantity, unit tax, unit price and untaxed amount, which item name and quantity is input by employee and unit tax, unit price and untaxed amount is calculated by name and quantity. Currently, it cannot delete the current row and update the quantity to when the item first time choosing when the item has been already added.

```
$('.quantity').on("keyup", function (event) {  
  
    var item_id = $(event.target).parentsUntil('tr').parent().find('.item_select').first().val();  
    var quantity = $(event.target).val();  
    var row = $(event.target).parentsUntil('tr').parent();  
    //alert(item_id);  
    if (false) {  
  
        if (confirm(raw_item_list[item_id] + " already selected, amend quantity?")) {  
            $(event.target).parentsUntil('table').find('tr').each(function (index, ele) {  
                if (ele.find('.item_select').val() == item_id) {  
                    ele.find('.quantity').focus();  
                }  
            });  
            row.remove();  
        } else {  
            $(event.target).val('');  
        }  
  
    } else {  
        update_order_line($(event.target).parentsUntil('tr').parent());  
        order_lines[item_id] = quantity;  
    }  
  
});
```

In addition, the total count, which used to count the untaxed amount, taxes and total amount of the sales order, which is not working. And the data in this page cannot post back to database.

Besides, edit sales invoice view is not exist. It designed attached with the sales invoice ID, and display format is similar as the new sales invoice page but with sales order detail.

BizWebSys Workspace www.lang

SALES

Invoices / New

Save Discard

Order

Customer Date

katy perry 05/23/2013

Invoice Lines

Product	Quantity	Unit Tax	Unit Price	Untaxed Amount
Guitar	9	12.00	120.00	1080.00
Guitar	7	12.00	120.00	840.00

Add an item

In the purchases part, both supplier payment and supplier invoices are not working. It has complete views but without enough functions. Purchases invoice payment page is designed for employee to either pay a purchases invoice to supplier or make a new payment to supplier. It including following data: supplier, invoice date, item name, due date, item's name, description, quantity, taxes, unit price, untaxed amount, and the calculation of total amount. Pay button is designed to pay for current purchase invoice. Currently, it can only display suppliers name and date chosen.

BizWebSys Workspace www.lang

Supplier Invoice / Internal Reference

SALES

Invoices

PURCHASES

Supplier Payment

Supplier Invoices

Pay

Save Discard

Invoice

Supplier Invoice Date :

Jude

Due Date :

Invoice

Product	Descriptions	Quantity	Taxes	Unit Price	Untaxed Amount
Add an item					

Untaxed Amount: 0.00

Taxes: 0.00

Deduction:

Total: 0.00

© 2013 Business Web System - UCL COMP2013 Group 10

The structure of supplier payments is very similar to sales invoice. It can display all the purchases invoices, create a new invoice and view an invoice. Unfortunately, currently, it can only display all purchases invoices.

Social Networks

By the structure of this part, it is not a complex and difficult to construct. However, to find a suitable and working APIs for our project was a large challenge. There is a lot of information and examples on the web but most of them are either not up-to-date or not suitable for our project. For first, since we are using PHP framework, so I try to use the Facebook API for PHP, unfortunately it did not seem work after days of work. Therefore, I tried it in another way, writing it in JavaScript. It did work quite smoothly.

For posting onto the Facebook, user has to put in the status content, check the Facebook boxes and press the 'Post' button. After that, the application will to initialise the Facebook SDK and API by sending the app ID and some of the setting (login status, cookie and parsing XFBML). If the application detects that there is no one already logged into Facebook in on this web browser, the application will pop up a new windows to let the user to log in by typing in the e-mail and the password. If the user is logged in but have not granted the publish stream permission, there will be also a pop-up window to ask for permission. If the user is already logged in and the account has already granted the publish stream permission, the status will be posted straight away. Here is the code for this part:

```
if ($("#facebook").prop("checked") == true)
{
    ...
    getLogin();
}
```

Here is the code for starting the process to post on Facebook if the 'Facebook' check box is checked.

```
{
    function(d)
    {
        var js, id = "facebook-jssdk", ref = d.getElementsByTagName("script")[0];
        if (d.getElementById(id)) {return;}
        js = d.createElement("script"); js.id = id; js.async = true;
        js.src = "//connect.facebook.net/en_US/all.js";
        ref.parentNode.insertBefore(js, ref);
    }(document)
};

window.fbAsyncInit = function()
{
    FB.init
    (
        {
            appId      : "444833315594772", // App ID
            channelUrl  : "//localhost/channel.html", // Channel File
            status      : true, // check login status
            cookie      : true, // enable cookies to allow the server to access the session
            xfbml       : true // parse XFBML
        }
    );
};
```

Here are the loading of the API and the SDK.

```
function getLogin()
{
    FB.getLoginStatus
    (
        function(response)
        {
            if (response.status === "connected")
            {
                publish();
            }
            else if (response.status === "not_authorized")
            {
                fbLogin();
            }
            else
            {
                fbLogin();
            }
        }
    );
};
```

From the coding above, if the user is already connected (is logged in and granted permission) the application will go to the publish() method.

If the user is either not logged in or have not granted permission, it will go to the fbLogin() method.

```
function fbLogin()
{
    alert("fbLogin");
    FB.login
    (
        function (response)
        {
            if (response.authResponse)
            {
                streamPublish();
            }
            else
            {
                // cancelled
            }
        }, { perms: "publish_stream" }
    );
};
```

Here is the coding for fbLogin() method that let user to login or granted permission or both if the user has not.

```

function publish()
{
    var params = {};
    params["message"] = $("#statusContext").val();

    FB.api(
        "/me/feed", "post", params, function (response)
        {
            if (!response || response.error)
            {
                alert("Error - Facebook");
            }
            else
            {
                alert("Sucessfully Posted on Facebook");
            }
        }
    );
};

```

Here is the publish() method that takes the value (what the user put into the status content textbox) and post onto the Facebook. If something goes wrong in the process, an error message will be pop-up as alert box to indicate. Otherwise, a successful message will be pop up as an alert box to indicate.

For the Twitter part, because at the time while I was starting to write the Twitter part, the majority of my coding is in JavaScript, therefore, I first started to write it in JavaScript. Unsurprisingly, it did not go so well for me. Therefore, I wrote in PHP at last.

Similar to Facebook, posting onto the Twitter, also require user to type in the content of the 'tweet' and check the 'Twitter' check box. In the construct class, the application has already loaded up the TwitterOauth library. Because for the form, I wrote in JavaScript. Therefore, I needed a way to pass the value (the content of the 'tweet') to the PHP function which does the Twitter posting. I chose the Ajax to handle the transferring. In the PHP method, it can only be posting. I ran out of time to complete the login part. Therefore, this part requires the user to be logged in already to be successfully posted onto Twitter.

```

function __construct()
{
    parent::__construct();
    // Loading TwitterOauth library. Delete this line if you choose autoload method.
    $this->load->library('twitteroauth');
}

```

This is the loading of the library.


```

$("#button_post").click(function()
{
    var status_note = $("#statusContext").val();

    if ($("#twitter").prop("checked") == true)
    {
        var ajaxurl = "http://" + (document.location.hostname) + "/socialnetwork/twitter";
        var ajaxdata = status_note;
        $.ajax
        (
            {
                type: "POST",
                url: ajaxurl,
                data: {ajaxdata : ajaxdata},
                success: function(results)
                {
                    alert("Sucessfully Posted on Twitter");
                },
                error: function(xhr, textStatus, error)
                {
                    alert(xhr.statusText);
                    alert(textStatus);
                    alert(error);
                    alert("AJAX ERROR");
                }
            }
        );
    }
});
}

```

Then, this is Ajax posting after the application detected the 'Twitter' check box is checked by the user.

```

public function twitter()
{
    $connection = $this->twitteroauth->create('qsyeIajydJgHRfLI9T4A',
        '9V5rqgTbwUdCvnlBiyF9eMegaaPJJWTlX05URm6cuU', '1327423556-7yRfZmvInpYu$DbbroXllupBntln7zjjtAdoyiF',
        'T6KeloIKVj0zKzVjVHyjFdwjLCNgTCIDpmHfz8jsNuk');

    $content = $connection->get('account/verify_credentials');

    if(isset($content->error))
    {
        return false;
    };

    $status = $_POST["ajaxdata"];

    $data = array
    (
        'status' => $status,
    );

    $result = $connection->post('statuses/update', $data);
}

```

This the PHP twitter() method. First, it initialises by passing the consumer key, consumer secret, the access token and the access token secret (which are the random string of numbers and alphabets

above). After that, the PHP will get the Ajax data (the content of the 'tweet'). Then, it just simply posts onto Twitter.

For the Google+, as I mentioned in the design, the Official API for Google+ is read-only access. Therefore, it is not possible to finish the Google+ by the time I had left at that time.

Beside the main functions, I also wrote some checking functions for checking empty string and none of the check box has been checked before posting as below.

```
if (status_note === "")
{
    alert("The content cannot be empty");
}

if (($("#facebook").prop("checked") == false) && ($("#twitter").prop("checked") == false))
{
    alert("Nothing is done due to none of the checkbox is selected");
}
```

Testing

Different Types of Testing

Functionality Testing

Testing for the links in the webpages and testing for the connection of databases is being tested as we working on the project.

Usability Testing

Due to we cannot find a suitable user to test on our website, therefore, members of our group carry out this test and try to improve the usability as we working on the project throughout the whole project. Basing on the final testing and writing the user manual, we have some final adjustments for a better usability.

Compatibility Testing

For our project, we design the website in a way that should be working on the popular web browsers, like IE, Opera, Chrome and Safari. Therefore, my website is worked fine in all of the browsers that I have mentioned.

Performance Testing

The setting between the website and the server and the database is much more complicate than we expected when we started the project. We can only manage to make sure that what they supposed to do. However, as the target we aimed for, the performance should be enough for a small-scale of users as it is for a small starting-up business.

Security Testing

As the time goes by, we do not have enough time to complete the research of this part. We only have password hash.

Testing Toolkits We used

CodeIgniter Unit Testing

Because of using the CodeIgniter as our PHP framework for our web application, therefore, naturally we use its built-in test kit. Although it is not a complete and mature kit for testing, it still provides a simple way to determine and assess whether that the coding works as we expected.

Selenium

It is a software testing framework for web applications. It does not require to learn a test scripting language. It provides a record and playback tool for authoring the tests. Different languages can be produced by Selenium, including some of the most popular language, such as C#, Java, PHP, Python and etc.

Exploratory Testing

Exploratory testing is a continuous to improve the testing method and order as the testing goes on. However, due to the dependencies and some incomplete parts, the exploratory testing does not go as smooth and well as we planned.

Inventory

Tests done in Inventory use CodeIgniter Unit Testing Class.

Test Type of Items

This testing is a unit testing that test which type of data is the output

```
$test = $data['items']; //test if list of items is in an array
$test_name = "item list datatype check";
echo $this->unit->run($test, 'is_array', $test_name);
```

The output result should be an array.

Test Name	item list datatype check
Test Datatype	Array
Expected Datatype	Array
Result	Passed
File Name	C:\Users\cheungtin\Documents\GitHub\bizwebsys\application\controllers\inventory.php
Line Number	140
Notes	

Above is the output of test. It is passed.

Test Number of Items

This testing is a unit testing that test the output number of items is correct.

```
$test = sizeof($data['items']); //test if number of items in list is correct, at time of test, there are 5 items
$test_name = "item list length";
echo $this->unit->run($test, 5, $test_name);
```

The output result should be 5.

Test Name	item list length
Test Datatype	Integer
Expected Datatype	Integer
Result	Passed
File Name	C:\Users\cheungtin\Documents\GitHub\bizwebsys\application\controllers\inventory.php
Line Number	145
Notes	

Above is the output of test. It is passed.

Automating

The test is done by the CodeIgniter Unit Testing, therefore, it is easily to test again automatically by enabling the unit testing again in the coding.

Contacts

Tests done in Inventory use CodeIgniter Unit Testing Class.

Test Retrieval Type of Contacts

This testing is a unit testing that test which type of data is the retrieval datatype

```

$test = $this->contacts_model->get_all_contact(); //test for contact retrieval datatype
$test_name = "check contact array";
echo $this->unit->run($test, 'is_array', $test_name);

```

The output result should be an array.

Test Name	check contact array
Test Datatype	Array
Expected Datatype	Array
Result	Passed
File Name	C:\Users\cheungtin\Documents\GitHub\bizwebsys\application\controllers\contacts.php
Line Number	38
Notes	

Above is the output of test. It is passed.

Test Number of Contacts

This testing is a unit testing that test the output number of items is correct.

```

$test = $data['contacts']['count']; //test for contact number variable used for layout , at time of testing, we have 2 contacts
$test_name = "number of contacts";
echo $this->unit->run($test, 2, $test_name);

```

The output result should be 2.

Test Name	number of contacts
Test Datatype	Integer
Expected Datatype	Integer
Result	Passed
File Name	C:\Users\cheungtin\Documents\GitHub\bizwebsys\application\controllers\contacts.php
Line Number	42
Notes	

Above is the output of test. It is passed.

Automating

The test is done by the CodeIgniter Unit Testing, therefore, it is easily to test again automatically by enabling the unit testing again in the coding.

POS

Tests done in POS use CodeIgniter Unit Testing Class.

Test new order

This testing is a unit testing that tests the process of passing data to model and inserting a new order record in SalesOrder table, and then controller use the ref number to get the newly inserted order_id from database.

```

//unit test
$test = $this->_test_new_order();

$expected_result = 'is_int';

$test_name = 'test new order';

echo $this->unit->run($test, $expected_result, $test_name);

private function _test_new_order()
{
    //create new order in salesorder table
    $date = date("Y_m_d");
    $payment_method = "Cash";
    $ref = "POS".date("d_m_Y_H_i_s");
    $employee_id = $this->session->userdata('employee_id');
    $this->pos_model->set_order($ref,$date,$payment_method,$employee_id);
    $order_id = $this->pos_model->get_orderid($ref);
    return intval($order_id);
}

```

The output result should be the order_id which is int type.

Test Name	test new order
Test Datatype	Integer
Expected Datatype	int
Result	Passed
File Name	C:\wamp\www\bizwebsys\application\controllers\pos.php
Line Number	612
Notes	

Above is the output of test. It is passed.

Automating

The test is done by the CodeIgniter Unit Testing, therefore, it is easily to test again automatically by enabling the unit testing again in the coding.

Sales and Purchases

Tests done in Inventory use CodeIgniter Unit Testing Class.

Test Retrieval Type of Sales

This testing is a unit testing that test which type of data is the retrieval datatype

The output result should be an array.

```

//test for salesorders retrieval datatype
$test = $this->sales_model->get_orders();
$test_name = "check salesorder array";
echo $this->unit->run($test, 'is_array', $test_name);

```

Test Name	check salesorder array
Test Datatype	Array
Expected Datatype	Array
Result	Passed
File Name	C:\wamp\www\bizwebsys\application\controllers\sales.php
Line Number	44
Notes	

Above is the output of test. It is passed.

Automating

The test is done by the CodeIgniter Unit Testing, therefore, it is easily to test again automatically by enabling the unit testing again in the coding.

Employee

Tests done in Inventory use CodeIgniter Unit Testing Class.

Test Retrieval Type of Employee

This testing is a unit testing that test which type of data is the retrieval datatype

The output result should be an array.

```
$test = $this->employee_model->get_all_employee(); //test for employee retrieval datatype
$test_name = "check employee array";
echo $this->unit->run($test, 'is_array', $test_name);
```

Test Name	check employee array
Test Datatype	Array
Expected Datatype	Array
Result	Passed
File Name	C:\Users\cheungtin\Documents\GitHub\bizwebsys\application\controllers\employee.php
Line Number	36
Notes	

Above is the output of test. It is passed.

Test Number of Employee

This testing is a unit testing that test the output number of contacts is correct.

The output result should be 7.

```
$test = $data['employees']['count']; //test for contact number variable used for layout , at time of testing, we have 7 contacts
$test_name = "number of employee";
echo $this->unit->run($test, 7, $test_name);
```

Test Name	number of employee
Test Datatype	Integer
Expected Datatype	Integer
Result	Passed
File Name	C:\Users\cheungtin\Documents\GitHub\bizwebsys\application\controllers\employee.php
Line Number	40
Notes	

Above is the output of test. It is passed.

Automating

The test is done by the CodeIgniter Unit Testing, therefore, it is easily to test again automatically by enabling the unit testing again in the coding.

Social Networks

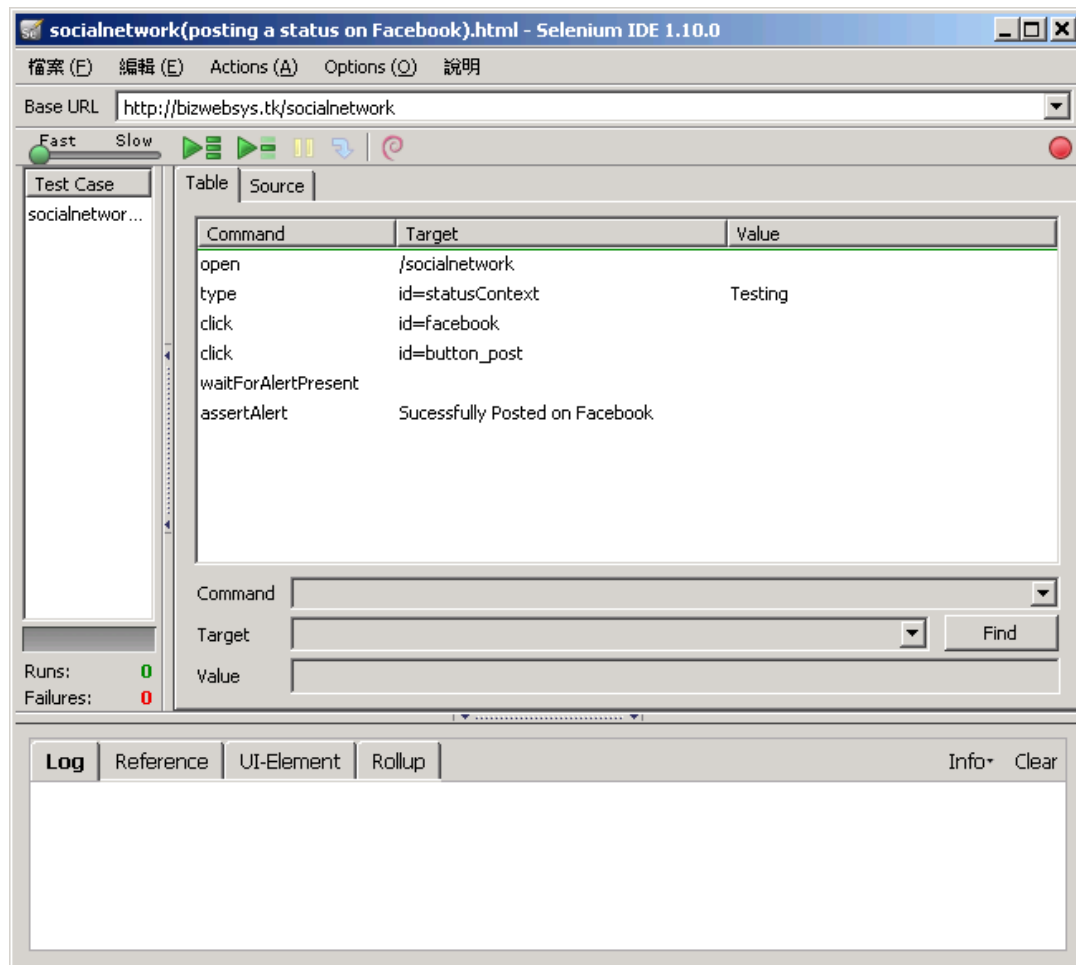
All of the tests in this part, are done by Selenium.

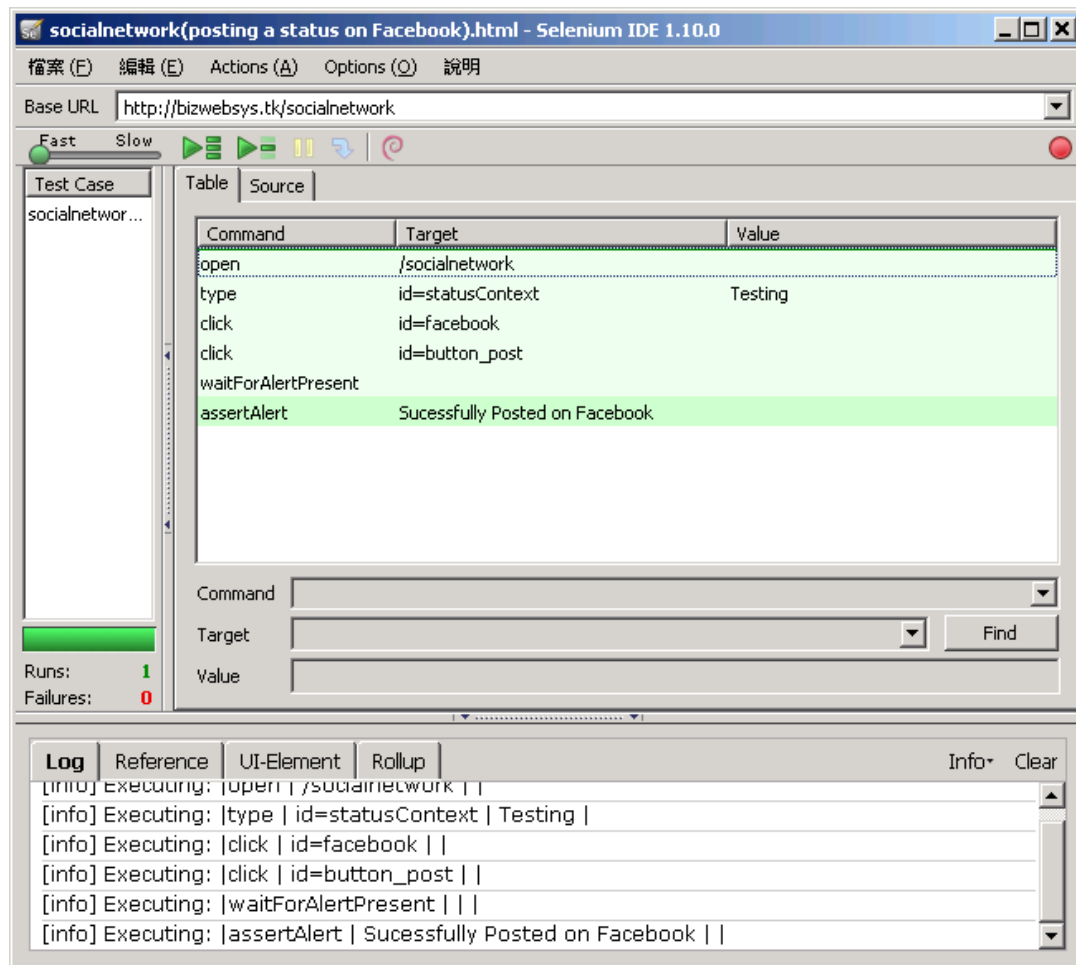
For this 'Connection to the Public', the results are all expected to be displayed clearly by a pop-up alert box.

Testing for Posting a Status onto Facebook

This testing is a functional testing that making sure that the application can post a status on to the social network, Facebook.

Here is the test table on the Selenium for this test



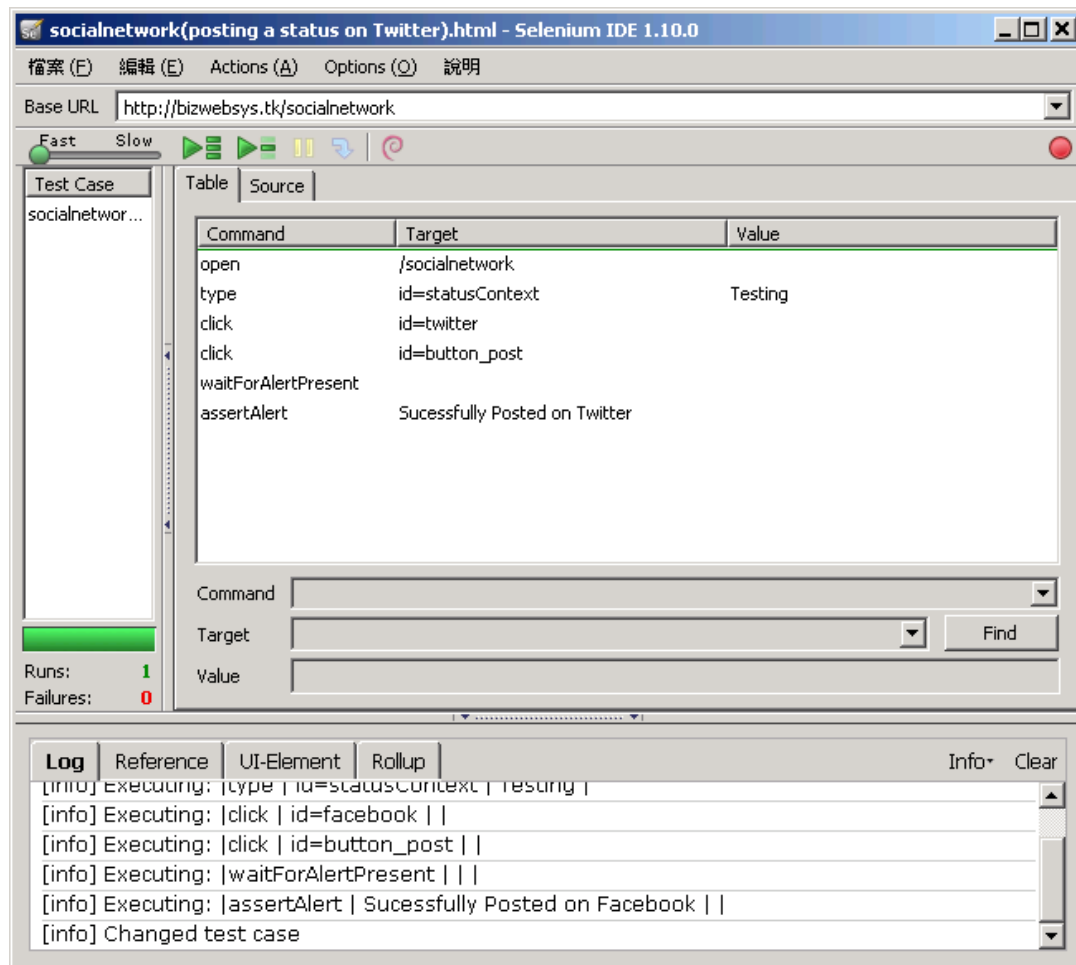


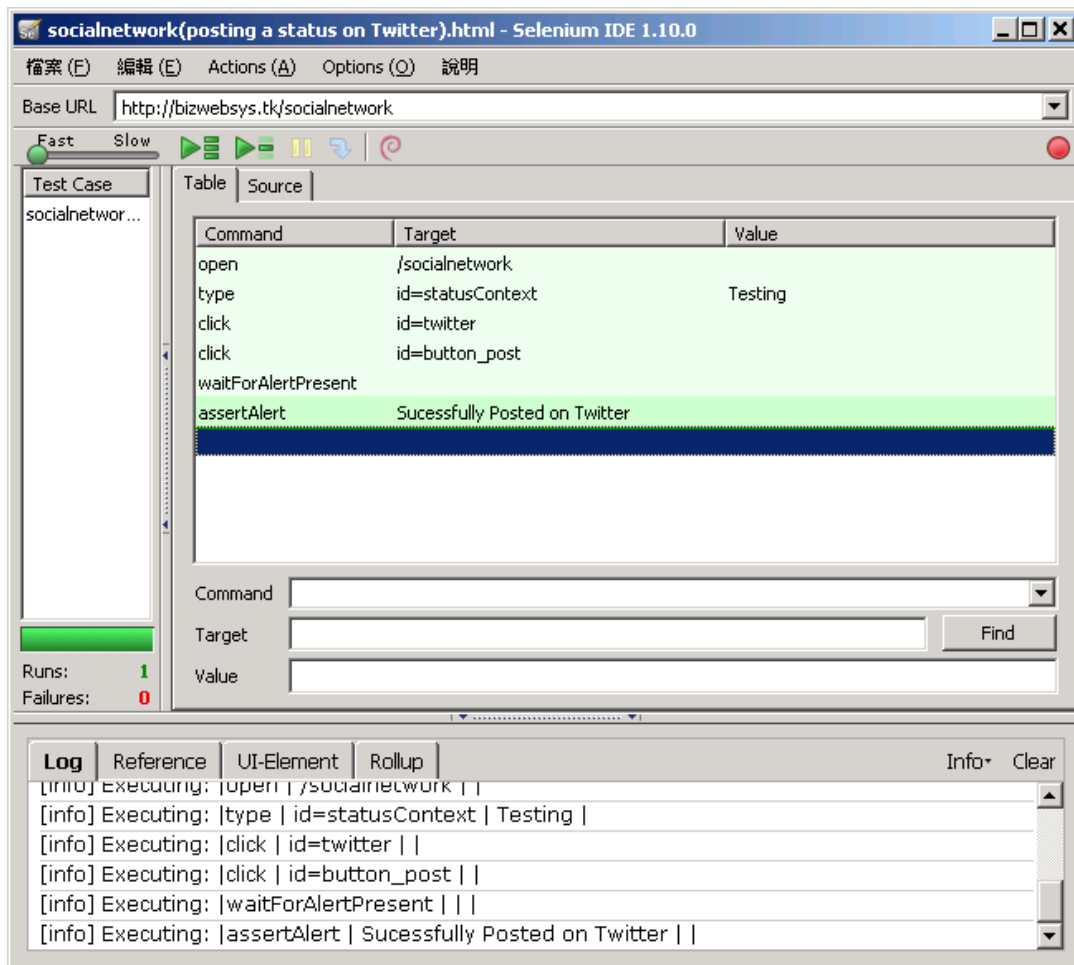
Therefore, the test is passed as it does what I expected.

Testing for Posting a Status onto Twitter

This testing is a functional testing that making sure that the application can post a status on to the social network, Twitter.

Here is the test table on the Selenium for this test





Therefore, the test is passed as it does what I expected.

Testing for Posting a Status onto Both Facebook and Twitter

This testing is a functional testing that making sure that the application can post a status on to the social networks, both Facebook and Twitter.

Here is the test table on the Selenium for this test

socialnetwork(posting a status on Both).html - Selenium IDE 1.10.0

檔案 (F) 編輯 (E) Actions (A) Options (O) 說明

Base URL

Fast Slow

Test Case

socialnetwor...

Table Source

Command	Target	Value
open	/socialnetwork	
type	id=statusContext	Testing 2
click	id=facebook	
click	id=twitter	
click	id=button_post	
waitForAlertPresent		
assertAlert	Sucessfully Posted on Facebook	
waitForAlertPresent		
assertAlert	Sucessfully Posted on Twitter	

Command

Target Find

Value

Runs: 1

Failures: 0

Log Reference UI-Element Rollup Info+ Clear

[info] Executing: |click | id=twitter | |

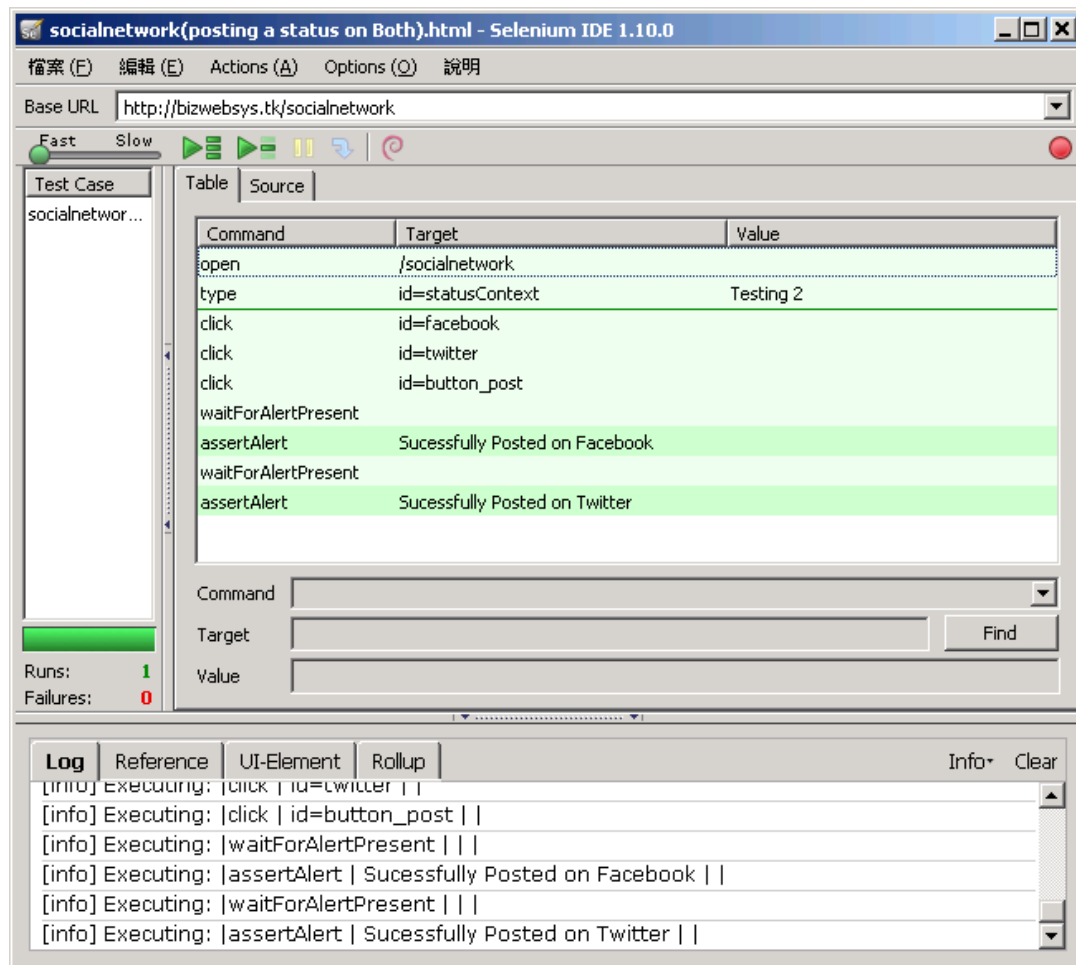
[info] Executing: |click | id=button_post | |

[info] Executing: |waitForAlertPresent | | |

[info] Executing: |assertAlert | Sucessfully Posted on Twitter | |

[info] Changed test case

[info] Changed test case

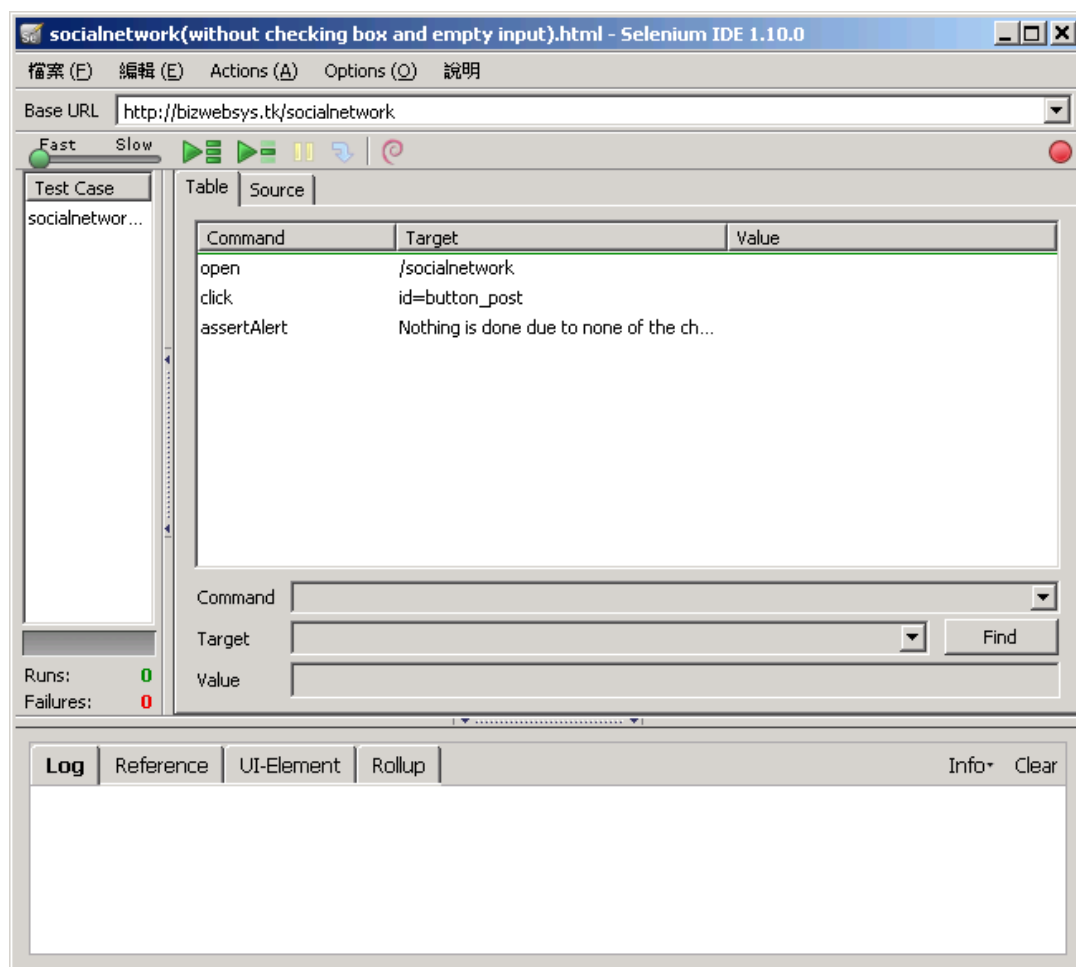


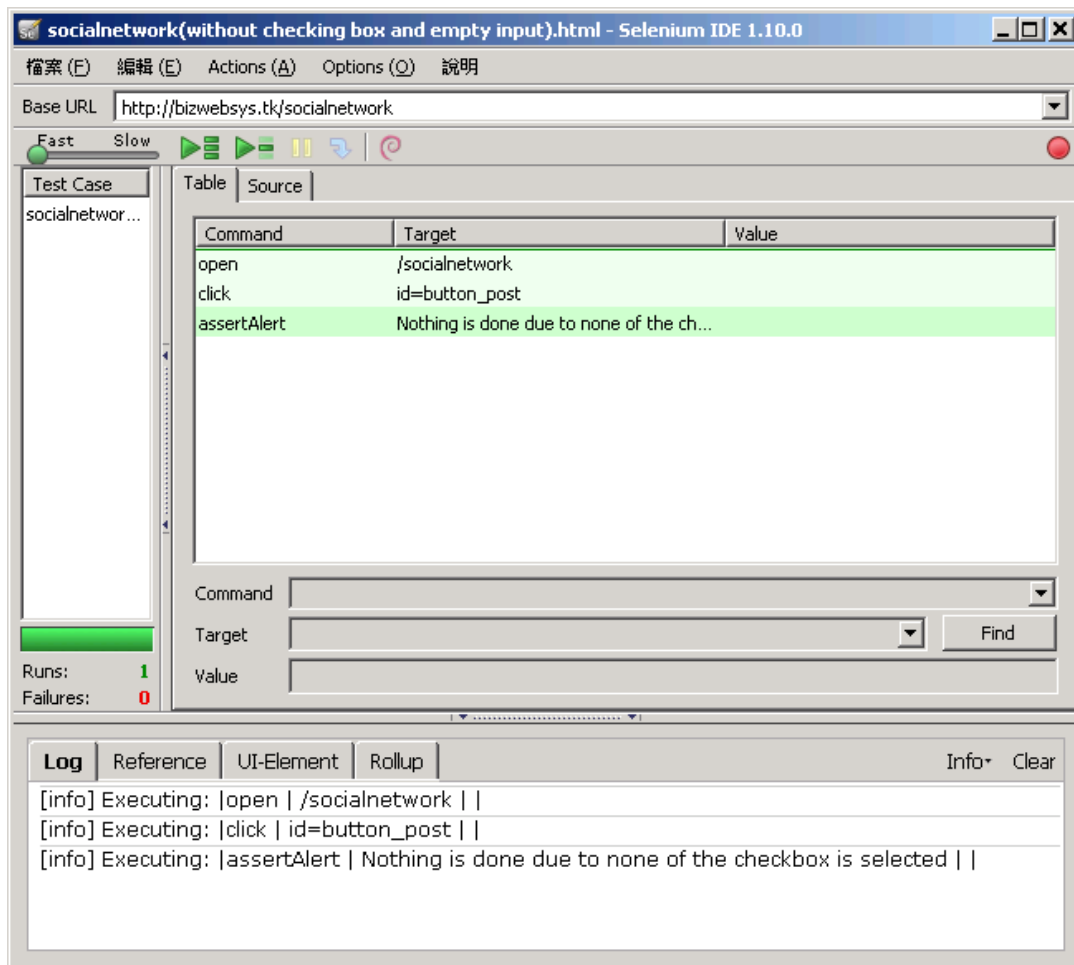
Therefore, the test is passed as it does what I expected.

Testing for Error Message for not Checking Any Checkboxes

This testing is a functional testing that making sure that the application preform a checking to alert a warning when none of the check boxes is checked.

Here is the test table on the Selenium for this test





Therefore, the test is passed as it does what I expected.

Automating

For each of the test, I have created a test script by Selenium. Therefore, for future testing, user only have to install a Selenium plug-in for the browser. Then, from the Selenium it can be easily run through all of the above test. The actual scripts will be at the appendix.

Review

By the end of project, most of the core functionalities are implemented and work as expected, including HTML5 app panel, mail system and private cloud file storage.

DELIVERED	
Name	Result
Inventory	Fully implemented and tested. Work as required.
Contacts	Fully implemented. Work as required.
POS	Fully implemented and tested. Work as required.
Sales/Purchases	Partial implemented.
Employee	Fully implemented. Work as required.
Mail	Fully implemented. Work as required.
OwnCloud	Fully implemented. Work as required.
Connecting to the Public	Fully implemented and tested. Work as required.

Due to time constraint, there are yet functionalities not delivered.

UNDELIVERED	
Name	Comment
generic yet customizable app panel	<p>The original idea is to make a customizable app panel where users can add new app or delete existing app. This was implemented and demonstrated in the COMP2013 prototype. But all the apps are hyperlinks to other online apps.</p> <p>In COMP2014, we did interview with two small businesses and found out some core functionalities they need in common. The team hence focused on developing apps to achieve these functionalities. But total number of apps are eight which is not enough for adding and deleting customization.</p> <p>Yet it is still possible to add in another customizable app panel which contains hyperlinks of other online apps.</p>
Online store	Due to time constraint, this has been scraped half way in development. However, given fully implemented POS module, which is an in-store selling terminal, it just needs to integrate online payment to function as online store.

Design evaluation

Cohesion and Decoupling: MVC

The system is developed using CodeIgniter PHP framework, which is based on the Model-View-Controller development pattern. The Model represents data structures. Model classes handle operations with database. The View is the information presented to user. In our system, a view is a webpage or webpage fragment like header. The Controller serves as an intermediary between the Model and the View. In our system design, each Controller handles an app, and each Controller associates with its own Model and several Views. This design pattern allows decoupling of modules,

each module only focus on a specific purpose, which is also reusable and easy to maintain. Besides, it also complies with the separation of concerns design principle.

Appropriate division into components and sub-systems

The system has been divided into three sub-systems: main system, mail system and cloud storage system. The three system all retrieve LDAP data for user authentication. Hyperlinks in main system to the other sub-systems link the three systems together. A failure in one system will not bring down the whole system.

In the main system, functionalities are divided into nine different apps. Each takes care of a specific business aspect. All ERP apps share one database, allowing data communication between apps.

This structure also allows extensibility for more apps and subsystems add in future.

Robustness and stability

To achieve robustness, all input fields have validation either using front-end JavaScript or back-end php or both. There are enough alert and warnings help user recover from errors. Some data get sanitized before inserting to database. Hence invalid and unexpected input will not break the operation. As long as the hosting server is in operation, the system will be up running.

Scalability

The system adapts well to increasing number of users and data. In the system, users are stored in LDAP server. If company grows fast and become supersize, it can start replicating more LDAP servers to balance the load. If there is huge amount of increasing data and files, then more specific servers can be setup to hold main system, MySQL database, mail system and cloud storage system separately. In this way, traffic can be shared among several servers instead of one.

Usability

The system interface is self-explanatory and simple to use. A normal sales person with a basic experience of using computer would be able to use it without training. The whole system interface is consistent since every page is rendered from the same skeleton view. User would not find much difference in terms of layout in apps.

Portability

The system is developed in a self-contained VM, it can be replicated and setup on other servers without much effort. It has minimal dependency on 3rd parties. Most of the code is written by our own developers. There are no vendor proprietary APIs.

Standards

The system implementation has a good respect to industrial standards in terms of Interoperability (between subsystems and apps), Standards compliant and Industry conventions. For example, the naming convention of php code strictly follows CodeIgniter naming convention. The front end code all follows HTML5 and CSS3 standards.

Aesthetic and minimalist design

A bootstrap CSS framework along with a jquery-ui-bootstrap theme is used to create a professional and nice look of the system. Following our goal to provide an uncluttered and easy to use user interface, the

interfaces are kept simple and concise but in the same time they offer all of the information the user needs to see at a certain time.

Things to be improved

Single database design

Despite the core functionalities achieved, there are notable drawbacks of the system. The critical one is single database design. The whole system uses only one database, which limits it to single company use. This also result in a tedious deploy process. On a second thought, if we have adopted multiple database design, which create a new database for each newly registered company online, it would not require deploy on own server at all. In that way, it will be a pure web service available for all companies.

Security

All user passwords are encrypted using sha. Clear text password never stored in session, cookie or send to email. The reset password mechanism enforce user to use the latest reset link within a specific period of time, invalid or expired link will not work. These provide a basic security level, but are certainly not enough for a secure business use. More security mechanisms shall be added in future to form a secure industrial standard system.

SSO

Although users can use same account to login all subsystems, it is more preferable to have single sign-on, so that users do not need to perform login each time.

Future Development

Mobile apps

Mobile apps on Android and IOS platforms can be developed with RESTful-ize JSON feeds.

Online Shopping

An online store auto generated from inventory can be developed. Some payment methods like paypal, credit card online payment can be integrated into it for secure transferring.

Calendar

A calendar can be integrated with ERP dates, so that it can function as a due reminder, shipping delivery lookup table. Direct edits will be through CalDav.

Newsletter

Integrate Mailchimp API with our mail system. Business users can send promotion newsletters to registered users. Newsletter content is auto-generated from items on offer. Newsletter will also be automatically published on WordPress webpage.

SEO

So far, the system only has a hyperlink to Google Webmaster to function as a SEO tool. In the future, we should develop our own SEO tool which enables search engine optimization of special deals on auto-generated WordPress webpages.

Customizable app panel

In future, we can add in more apps like office and project management tools. Users can choose apps from 'App Center' and add into their workspace. Also, users can delete apps from their workspace.

Project management evaluation

Management	Rate	Comment
Planning	B	Milestones were set at the beginning of the project, providing goals and main tasks for each phase of project development. But the goals set are proved to be too ambitious. Neither of the milestones was met on time. Tasks are assigned to team members via DO.com or through Facebook group effectively.
Scoping	B	With time constraint, decisions were made to scale project down and down. Some essential part like online store was scrapped off. But the remaining functionalities were still enough to meet the basic requirements.
Governance	B	Team leaders have a fairly good control over team members' progress. Submission of work was usually lagging, but most of the time, acceptable. However, with two joining leaders, sometimes a decision could not be made efficiently, which result in undetermined problem.
Quality control	A	Leader checked on code pushed onto repo. Using github issues panel, bugs and issues were raised to developers. So far, there are eighteen issues raised. Two are still open, and 16 closed. Among closed issues, twelve are fixed, four are wontfix.
Status reporting	A	Weekly report were gathered and submitted to department every week. Project blog was regularly updated. Communication with professor and TA was frequent. When there were any problems that cannot be solved within group, it would be reported to professors.

It has been learnt that project planning should be feasible, not being too ambitious. Otherwise, with the frustration of missing first two goals, the team will get slack and continue missing other goals. It has also been learnt that project management procedures should be defined up front, which including how the team will manage issues, scope change, quality, communication, and so on. It is important for the whole team to understand how the project will be managed. Besides, keeping track of finished and unfinished tasks, reviewing work plans regularly are also very important for project running.

Therefore, in future projects, we will carefully plan the workflow beforehand, making sure it is feasible. A project management procedure will be drew up and made clear to every team member at the beginning of project. At the meantime, keep the whole team progress under control. Do regular reviewing on project plans. Resolve issues as quickly as possible, the project manager should manage open issues diligently to ensure that they are being resolved.

Conclusion

The objective of the project is to create a thin web service solution for small business user to help improve their businesses. An HTML5 app panel has been developed which includes ERP apps, mail system, file storage, social media and SEO app. With the ERP apps taking care of essential business management including inventory, contacts, employee, sales and purchases, an in-store selling terminal POS is also provided. Two subsystems, mail and cloud storage, which are connected to LDAP server,

retrieve employees as users and use same authentication with main system. Social media and SEO apps help companies improve their publicity on the web.

Despite achieving the essential functionalities a small business would require, we could extend the system with more services to be a commercial level business tool kit. With greatly expanded services and more integrated subsystems, the system has a potential to be a capable product with more coherent experience.

System Manual

The system can be deployed on company's own server (or commercial server). The deploy process involves registering domain name, setup WampServer (windows), Apache + php +MySQL (Linux) and LDAP server setup.

Get system files

Clone a copy from <https://github.com/horaceli/BizWebSys> and name it "bizwebsys".

Domain name and hosting

There are many websites providing domain name registration service. User can just go to one of the websites and buy a domain name for company.

It is a cost effective way for small business user to host their website on a dedicated hosting company. For a reliable, reputable hosting service for business use, the cost is £9-10 per month or more. This will be on a web server shared with maybe 50 other websites. If users want to have their own web server, then the cost starts at over £60 per month. Domain name registration and hosting can usually be set up through the same company.

For windows server

Setup WampServer

WampServer is a Windows web development environment. It allows user to create web applications with Apache2, PHP and a MySQL database. Alongside, PhpMyAdmin allows user to manage easily their database.

1. Go to WampServer website: <http://www.wampserver.com/en/>
2. Double click on the downloaded file and just follow the instructions. It is like a normal software installation wizard. The WampServer package is delivered with the latest releases of Apache, MySQL and PHP.
3. User can test the WampServer setup by clicking on the "localhost" link in the WampSever menu or open internet browser and go to the URL : <http://localhost>

Install system

1. The "www" directory will be automatically created on WampServer setup (usually c:\wamp\www). Create a subdirectory called "bizwebsys" in "www" and put cloned PHP files "bizwebsys" inside.
2. Find httpd.conf from wamp menu, search for "DocumentRoot" and edit it as below:

```
DocumentRoot "C:\wamp\www\bizwebsys"  
<Directory "C:\wamp\www\bizwebsys">
```
3. To test the installation, open internet browser and go to the URL : <http://localhost>. It now should display the main page of bizwebsys.

Setup LDAP server

There is a free OpenLDAP package available for download here:

<http://www.userbooster.de/en/download/openldap-for-windows.aspx>

User can install OpenLDAP following this article: <http://www.userbooster.de/en/support/feature-articles/openldap-for-windows-installation.aspx>

For Linux server

Apache installation

1. Open up the Terminal (Applications > Accessories > Terminal).
2. Copy/Paste or type the following line of code into Terminal and then press enter:

```
sudo apt-get install apache2
```

3. The Terminal will then ask you for your password, type it and then press enter.
4. To test apache, open up any web browser and then enter the following into the web address: <http://localhost/>. You should see a folder entitled apache2-default/. Open it and you will see a message saying "It works!".

PHP installation

1. Open up the Terminal (Applications > Accessories > Terminal).
2. Copy/Paste or type the following line into Terminal and press enter:

```
sudo apt-get install php5 libapache2-mod-php5
```
3. In order for PHP to work and be compatible with Apache we must restart Apache. Type the following code in Terminal to do this:

```
sudo /etc/init.d/apache2 restart
```

Test PHP

1. In the terminal copy/paste or type the following line:

```
sudo gedit /var/www/testphp.php
```

This will open up a file called testphp.php.
2. Copy/Paste this line into the phptest file:

```
<?php phpinfo(); ?>
```
3. Save and close the file.
4. Now open a web browser and type the following into the web address:
<http://localhost/testphp.php> . It will show you the page that has all information about your php.

MySQL installation

1. Open up the amazing Terminal and then copy/paste or type this line:

```
sudo apt-get install mysql-server
```

2. Type the following into Terminal:

```
mysql -u root
```

Following that copy/paste or type this line:

```
mysql> SET PASSWORD FOR 'root'@'localhost' = PASSWORD('yourpassword');
```

3. We are now going to install a program called phpMyAdmin which is an easy tool to edit your databases. Copy/paste or type the following line into Terminal:

```
sudo apt-get install libapache2-mod-auth-mysql php5-mysql phpmyadmin
```

Get PHP to work with MySQL

1. open a file entitled php.ini. To open it type the following:

```
gksudo gedit /etc/php5/apache2/php.ini
```

Change this line:

```
;extension=mysql.so
```

To look like this:

```
extension=mysql.so
```

2. restart Apache by type code below:

```
sudo /etc/init.d/apache2 restart
```

Install system

1. Find the "www" directory under Apache directory. Create a subdirectory called "bizwebsys" in "www" and put cloned PHP files "bizwebsys" inside.
2. Find httpd.conf under Apache directory, search for "DocumentRoot" and edit it as below:

```
DocumentRoot "{directory to www}\www\bizwebsys"  
<Directory "{directory to www}\www\bizwebsys">
```
3. To test the installation, open internet browser and go to the URL : <http://localhost>. It now should display the main page of bizwebsys.

Setup LDAP server

For Ubuntu users, please follow installation steps here:

<https://help.ubuntu.com/12.10/serverguide/openldap-server.html>

Setup Mail Servers

Download the config files from the Github repository and replace the appropriate directories under /etc/

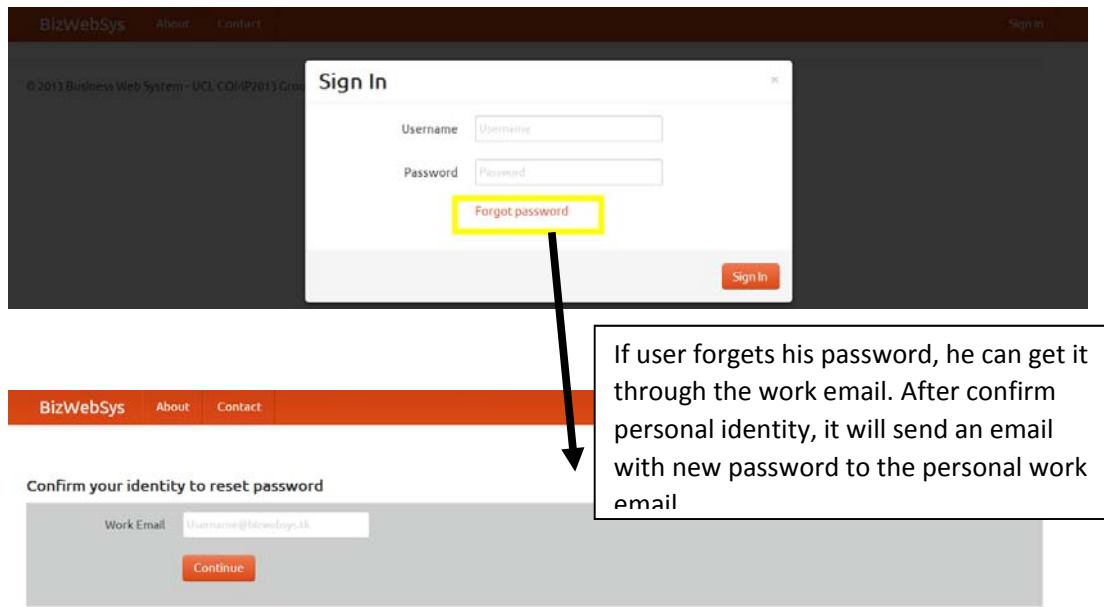
Other Subsystems

Download the appropriate packages from the repository, extract, and edit the configuration files as recommended by the applicable readme file.

User Manual

Login page

After launch to <http://bizwebsys.tk/>, click Sign in button which on the right hand side. It will appear a Sign in pop-up. The initial employees' username and password are given by owned company administrator.



Workspace panel

After login successfully, it will display the workspace panel. There are three parts in the panel, ERP, Apps and personal setting. ERP is for user to control the business, and Apps is for personal purpose use.



ERP

ERP is divided into five sections, Inventory, Contacts, POS, Sales/Purchases and Employee. Inventory is used to control items, Contacts for manage customers and suppliers, POS is a sales terminal,

Sales/Purchases for manage sales and purchases invoices for the business, Employee is for human management.



Inventory

Inventory home screen

It shows the all items details in the inventory, including SKU, name, category, stock, cost price, net price(the price not including taxes).

BizWebSys Workspace ww.liang

Inventory Q Search

Create Print Delete

	SKU	Name	Category	Stock	Cost Price	Net Price
<input type="checkbox"/>	S_B_TISSUE	Sainsbury's Basics Tissue	Health	7	4.00	2.00
<input type="checkbox"/>	S_Butter	Butterlicious	Food	977	5.00	6.00
<input type="checkbox"/>		Guitar		100		150.00
<input type="checkbox"/>		dfgdf	Baby & Toddler			

© 2013 Business Web System - UCL COMP2013 Group 10

Select the items want to manipulate to print or delete

Create a new item in inventory, print and delete items

Each item line is clickable, it can check item detail information


Create an new item in inventory

To fill the information of an item(name, category, supplier, cost price, net price, VAT rate, discount rate, stock, stockROP(the item stock cannot under specific number), GTIN, SKU and description).

BizWebSys Workspace www.lang

Item / New

Save Discard



Item Name

Category

Supplier

Cost Price

Net Price

VAT Rate

Discount Rate

Description

Stock

Stock ROP

GTIN

SKU

To upload a company's logo

Item name is required


Display an item

This view shows information of an item with corresponding id and name, and data in all fields cannot be changed.

BizWebSys Workspace www.lang

Item / 1 Sainsbury's Basics Tissue

Back **Stock up** Edit Print Delete



Item Name :

Category

Supplier

Cost Price

Net Price

VAT Rate

Discount Rate

Description

Stock

Stock ROP

GTIN

SKU

Change the number of

Stock Up

Stock to add :

Stock Up

Cancel

Edit item

BizWebSys Workspace ww.lang

Edit / Sainsbury's Basics Tissue

Save Cancel

Item Name: Sainsbury's Basics Tissue		Category: Please Choose :	
Supplier: Jude	Stock: 7	Stock ROP: 1	GTIN: 01179059
Cost Price: 4.00		SKU: S_B_TISSUE	
Net Price: 2.00			
VAT Rate: 0.0250			
Discount Rate: 1.000			
Description: Quality Tissue			

© 2013 Business Web System - UCL COMP2013 Group 10

Contacts

Contacts home panel

This shows the business contacts including customers and suppliers of the business.

BizWebSys Workspace ww.lang

Contacts Search

Create

 Jude	 katy perry
------------------------------------------------------------------------------------------	------------------------------------------------------------------------------------------------

© 2013 Business Web System - UCL COMP2013 Group 10

It is clickable, to view contact or supplier

Display detail

It includes image, first name, surname, common name, home address, city/state, zip code, country, work, mobile fax, email, organization and postal address. Each field in this view is unable to edit.

BizWebSys Workspace www.liang

Contact / 3 Jude

[Back](#) [Edit](#) [Print](#) [Delete](#)

No Image Available

First Name: Jude Surname: Law
Common Name: Jude

Personal Information Postal Details

Home Address: Work: Mobile: Fax: E-mail: Organization:
City/State: Zip Code: Country:

© 2013 Business Web System - UCL COMP2013 Group 10

Edit detail

BizWebSys Workspace www.liang

Edit / Jude

[Save](#) [Cancel](#)

No Image Available

First Name: Jude Surname: Law
Common Name: Jude

Personal Information Postal Details

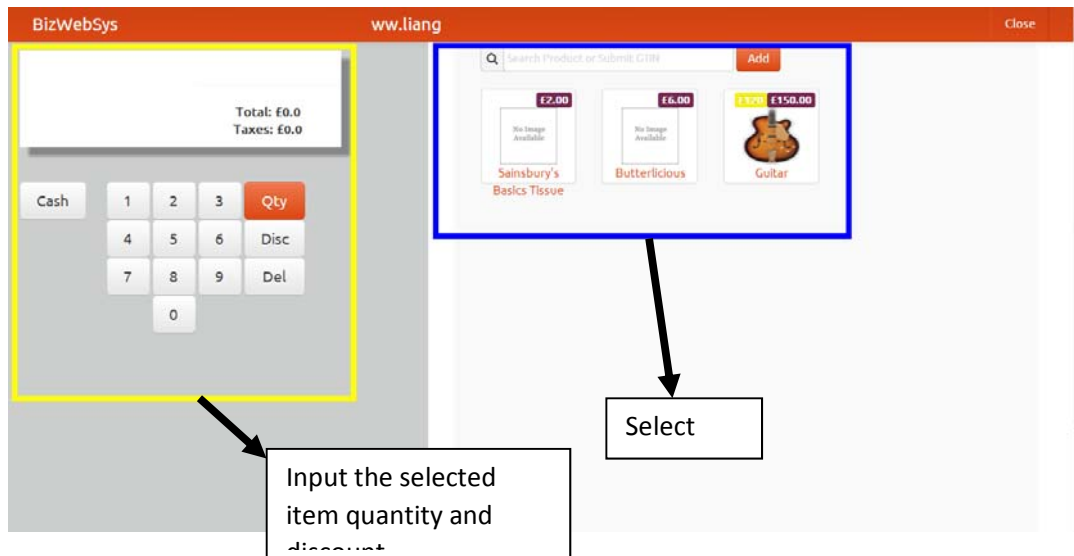
Home Address: Work: Mobile: Fax: E-mail: Organization:
Street name 1: Street name 2: City/State: Postal Co: Country:

© 2013 Business Web System - UCL COMP2013 Group 10

POS

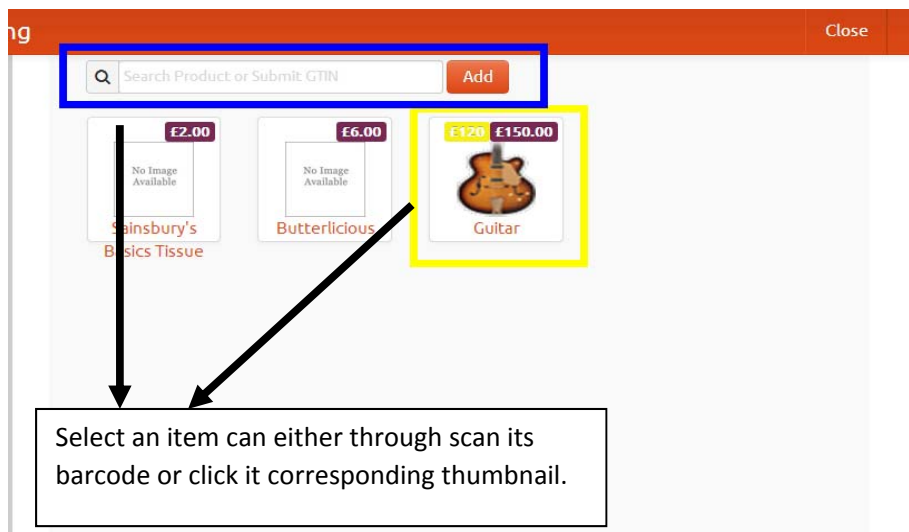
Pos home panel

POS is a sales terminal used during store selling. It is designed for touch screen. It can be divided into two parts, right hand side is to select items and right hand side is to make an order, input the quantity and discount for the selected item.



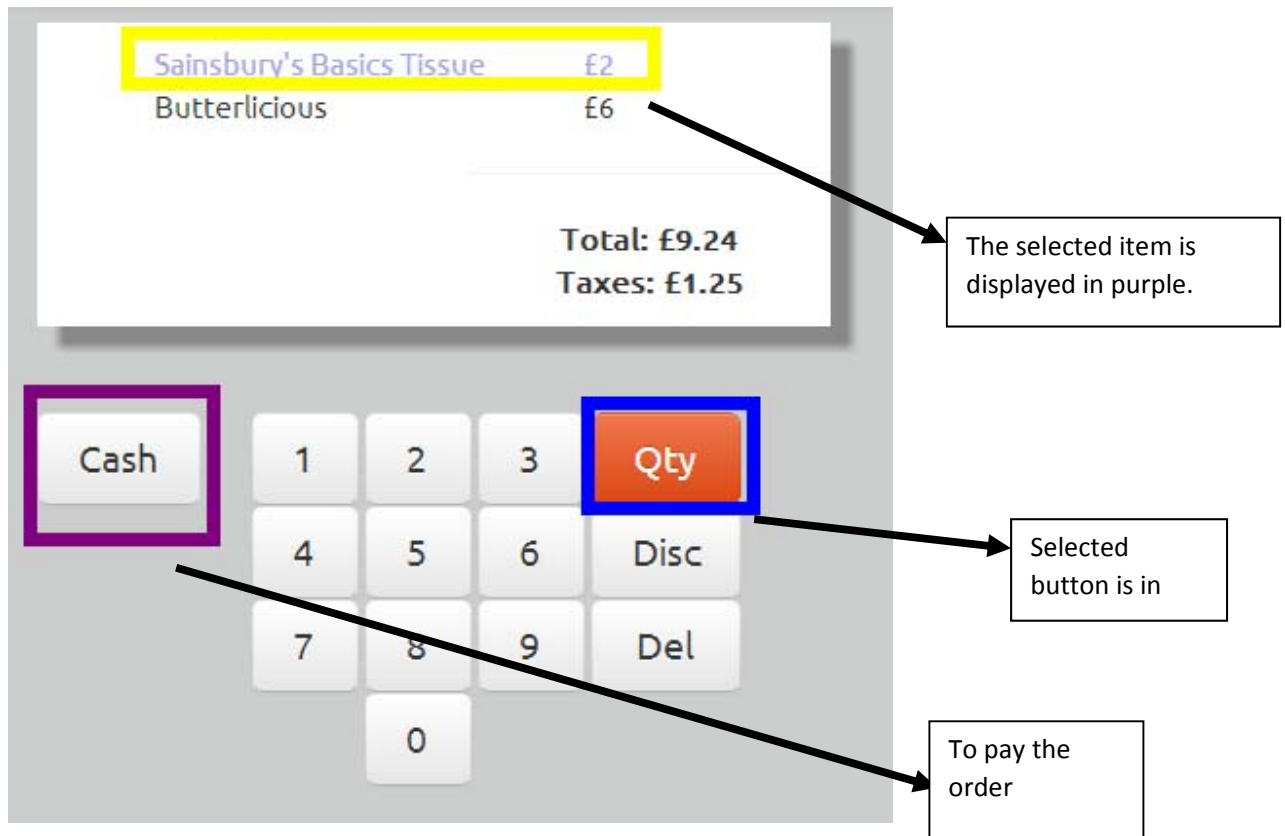
Select items

There are two ways to select an item either through scan or type GTIN, which is convenient for use purpose. Or select by thumbnails, which can clearly see the price and discount.



Make a order

The selected item in right hand side will appear in this view simultaneously. User can input the quantity and discount for the item. User can select multiple items in the same order. The delete button can work for both quantity and discount. NOTICE: discount is default start from 0. E.g. discount: 0.9, press Disc button first, then press 9.



Make a payment

After confirm the order, it appear the payment view to make a payment. If the Cash amount is less than the total amount, the validate button cannot be click.

Payment

Total:	£9.03
Cash (Pound):	<input type="text" value="7"/>
Paid:	7
Remaining:	2.02
Change:	0

Back Validate

Show receipt

After make a payment, it will appear the receipt showing the payment details. This receipt can be print out.

Receipt

08/05/2013 00:01:03 Order 12

Sainsbury's Basics Tissue £1.79
quantity: x3
discount: x0.3

Butterlicious £6

Subtotal: £7.8

Tax: £1.25

Total: £9.05

Cash: £10

Change: £0.96

Print

Next Order

Sales and Purchases

Sales panel

It shows the all sales invoice in the database. Dispatched button is designed to dispatch items convenient when the order was already paid.

BizWebSys

Workspace

wu.liang

SALES

Invoices

PURCHASES

Supplier Payment

Supplier Invoices

Create

	Customer	Invoice Date	Internal Reference	Salesperson	Payment Deadline	Total	Paid	Dispatched
<input type="checkbox"/>	Jude	2013-04-03	POS01_04_2013_12_00_37	EeRen	2013-05-31		2013-05-24	2013-04-16
<input type="checkbox"/>	Jude		POS02_04_2013_10_23_39	Li Jing			2013-04-02	2013-04-16
<input type="checkbox"/>	Jude	2013-04-15	POS15_04_2013_16_06_25	Weiwei			Outstanding	2013-04-16
<input type="checkbox"/>	Jude	2013-04-15	POS15_04_2013_16_11_26	Timothy			Outstanding	2013-04-16
<input type="checkbox"/>	Jude	2013-04-16	POS16_04_2013_12_31_16	Li Jing			2013-04-16	<div>Dispatch</div>
<input type="checkbox"/>	katy perry	2013-04-24	POS24_04_2013_03_53_47	Horace Li			2013-04-24	<div>Dispatch</div>
<input type="checkbox"/>	katy perry	2013-05-02	POS02_05_2013_14_19_01	Li Jing			2013-05-02	<div>Dispatch</div>
<input type="checkbox"/>		2013-05-03	POS03_05_2013_11_22_47	Li Jing			2013-05-03	<div>Dispatch</div>
<input type="checkbox"/>		2013-05-03	POS03_05_2013_11_25_26	Li Jing			2013-05-03	<div>Dispatch</div>
<input type="checkbox"/>		2013-05-08	POS08_05_2013_00_00_57	Weiwei			2013-05-08	<div>Dispatch</div>
<input type="checkbox"/>		2013-05-08	POS08_05_2013_00_01_01	Weiwei			2013-05-08	<div>Dispatch</div>
<input type="checkbox"/>		2013-05-08	POS08_05_2013_00_01_01	Weiwei			2013-05-08	<div>Dispatch</div>

Create new invoice

Make new sales invoice line for customer.

Invoices / New

Save Discard

Order

Customer: Jude Date: 05/24/2013

Invoice Lines

Product	Quantity	Unit Tax	Unit Price	Untaxed Amount
Butterlicious	5	1.20	6.00	30.00

Add an item

Days until Payment Due: Untaxed Amount: 0.00
Taxes: 0.00
Additional Information: Deduction: 0.00

Total: 0.00

Display invoice

This displays the sales invoice details. If the invoice was not been paid, it can be pay in this time by click a Register Payment button. Once the invoice paid, this button will not appear in this view.

Customer Invoice / 1

Edit Print Delete

Invoice 1

Customer: Jude Invoiced Date: 2013-04-03
Paid: 2013-05-24 by Debit Card

Invoice Lines

Product	Quantity	VAT	Net Price	Net Total
Sainsbury's Basics Tissue	200	40	2.00	400

Days until Payment Due: 2013-05-31 Subtotal: 400
Tax: 40
Additional Information: remark1 Deduction: -5.50

Grand Total: 434.5

Purchases panel

Purchases panel is similar to sales panel. The most different part is that it can make a new payment directly, don't need to make an invoice forehead. Notice: purchases panel is currently cannot use.

Employee

Employee is similar to Contact.

Edit employee

For employees, it can only display employee details. Employees do not have edit view. For the administrator, they can edit the employees' information.

BizWebSys Workspace

Edit / eeren.tan

Save Cancel

First name :
Ee Ren

Surname :
Tan

Common name :
EeRen

Username :
eeren.tan

Password : (Leave it blank if you do not wish to reset)

Contact Information

Email : eeren.tan@bizwebsys.tk

Home phone : 072349019209

Mobile : 09238948293

Administrator can reset the employees'

Email cannot be change

Apps

App panel includes four apps, Bizsysmail, Owncloud, webmaster tool and connecting to public. Bizsysmail is a work mail. Owncloud is personal cloud service. Wenmaster tool is used to analyze and manage BizWebSys. Connecting to public is post message to the public via BizWebSys.

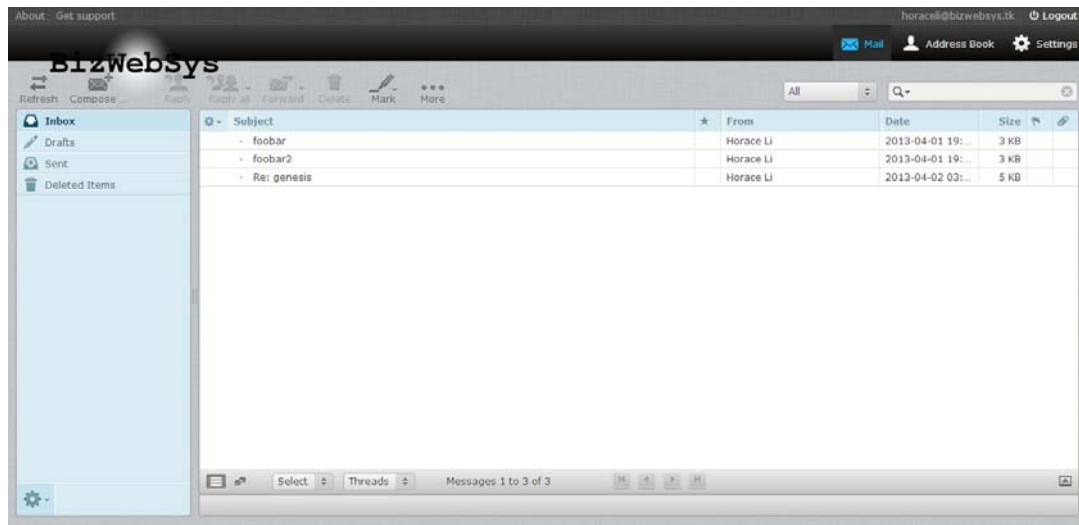
Apps



Mail

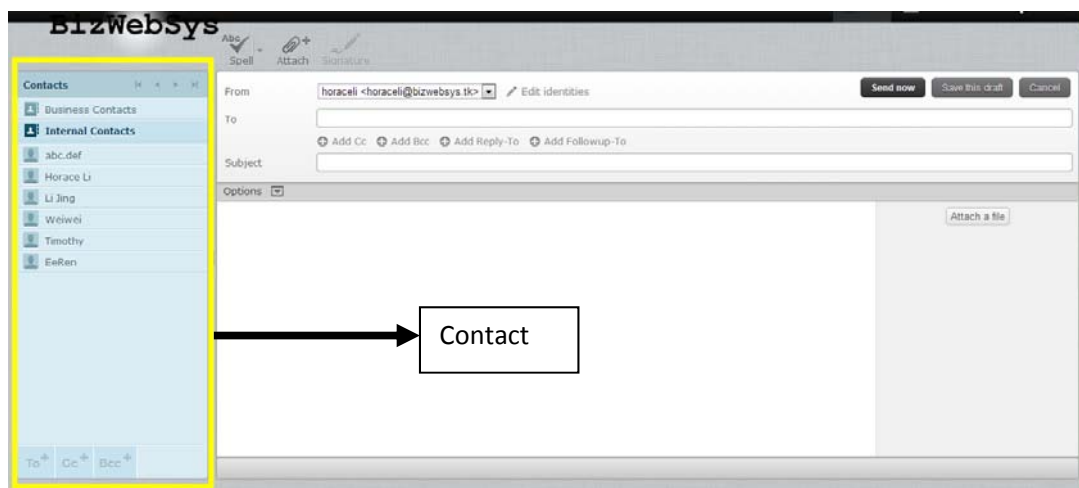
Use BizWebSys user name and password to login the BizWebSys mail. The email service provides a way for inter-company communicate and communicate to outer emails, like Hotmail and Gmail.



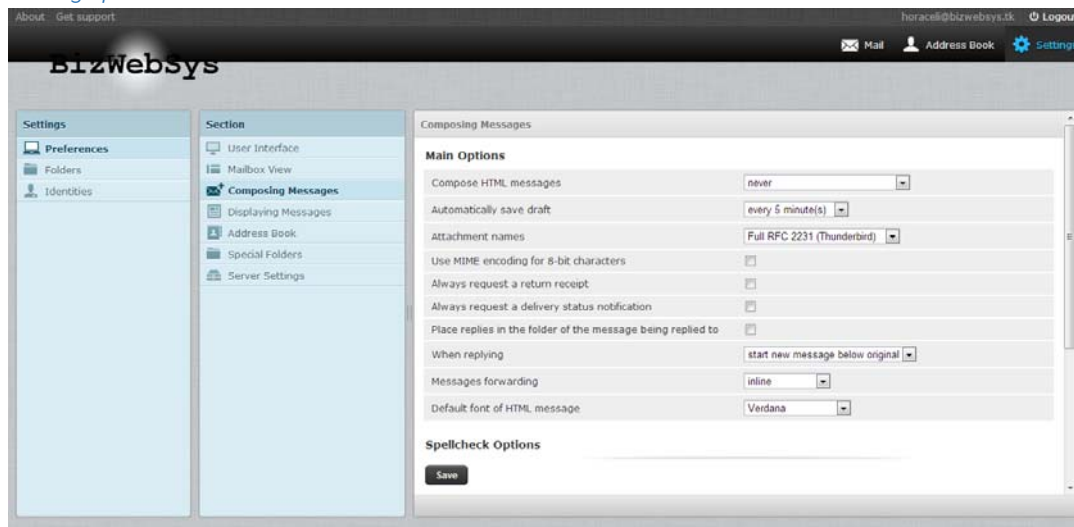


Contacts

All employee contacts display in internal contact.



Settings panel



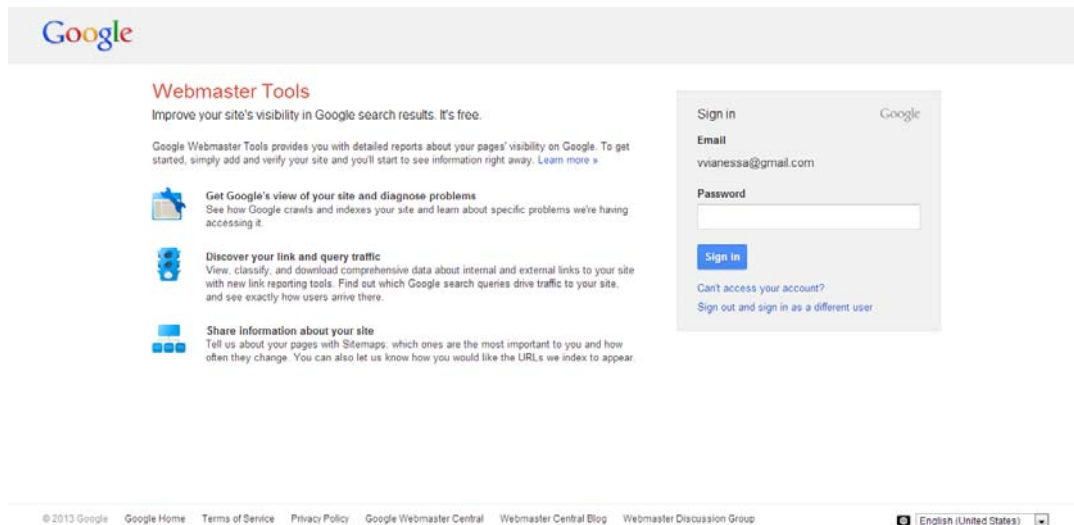
Owncloud

Owncloud is for employees to store files, have online calendar, photo storage and task management. It need to login using BizWebSys account.



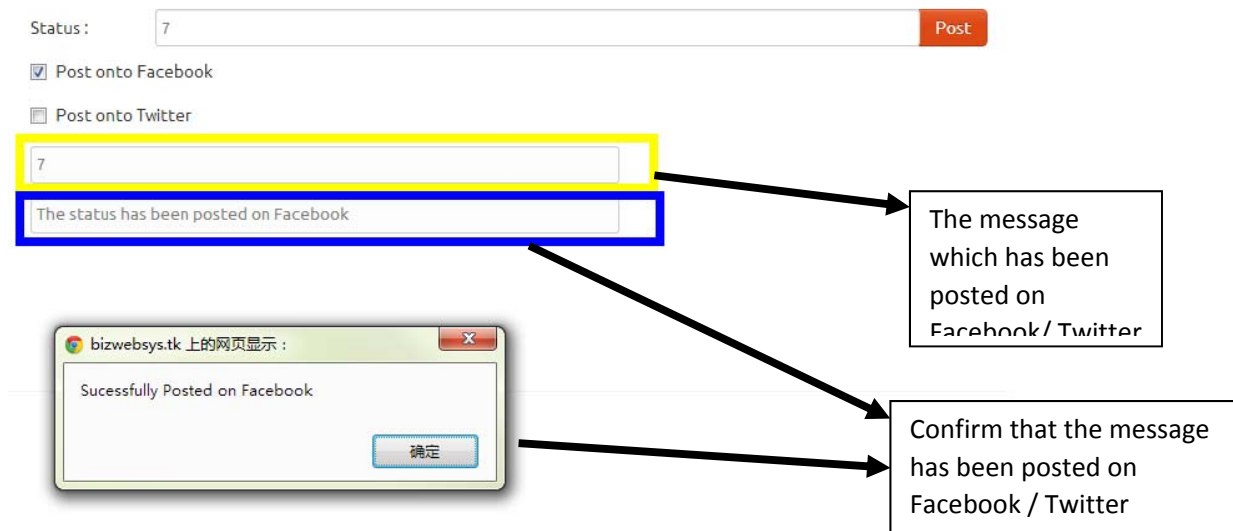
Webmaster tools

It is a hyperlink to Google Webmaster tools. It needs users own Google account to login.



Connecting to the public

This is a way to post states via BizWebSys. It can post on Facebook and Twitter. For Facebook, Notice: first time need to login to the Facebook account. For Twitter, Notice: need to login to Twitter account forehead.



Status :

☒ Post onto Facebook
☒ Post onto Twitter

Post same message on both Facebook and

bizwebsys.tk 上的网页显示 :

Sucessfully Posted on Twitter

☐ 禁止此页再显示对话框。

确定

Settings

Change password

User can change their own password.

Change password

password

New password

Confirm password

Company Settings

The setting panel only displays in administrator account. This setting used to set up the basic information for the company.


BizWebSys

Workspace

Settings

horaceli

Company Info



Change Logo

Company Name :

Green Leaf

Address :

Gower Street, London

Street name 2 :

City :

State :

Zip :

Belize

Phone :

098765456789

Vat Code :

GB111111

Save

© 2013 Business Web System - UCL COM-IP2013 Group 10