|  |
| --- |
| Software Architecture Document |
| Vintage Vogue: ITP Semester 2 2011 |
| John Agbulos, Norman Taminaya, Sang Uk Kim, Cameron Ly |

Table of Contents

[Introduction 3](#_Toc309622975)

[Purpose 4](#_Toc309622976)

[Scope 5](#_Toc309622977)

[Architectural Representation 6](#_Toc309622978)

[Architecture goals and constraints 6](#_Toc309622979)

[Technology selection 7](#_Toc309622980)

[Use case 8](#_Toc309622981)

[Use case description 8](#_Toc309622982)

[Logical view 12](#_Toc309622983)

[Sequence diagrams 12](#_Toc309622984)

[Package diagram 15](#_Toc309622985)

[Development view 16](#_Toc309622986)

[Development diagram 16](#_Toc309622987)

[Component diagram 17](#_Toc309622988)

[Physical view 18](#_Toc309622989)

[Deployment diagram 18](#_Toc309622990)

[Data view 19](#_Toc309622991)

[Data Dictionary 19](#_Toc309622992)

[Table: User 19](#_Toc309622993)

[Table: Set 19](#_Toc309622994)

[Table: Goods 19](#_Toc309622995)

[E-R Diagram 20](#_Toc309622996)

# Introduction

The following document is a Software architecture Document(SAD) which describes the architecture and design of the Vintage Vogue website.

The notable features of the SAD will contain:

* A definition of the scope of the project
* An architectural representation of the software deliverable, described in 4+1 architectural view model
* Architecture goals and constrains
* Technological selection

The purpose of the SAD initially is for the Vintage Vogue team to conceptualise and design the proposed software solution in different views. Primarily the SAD is beneficial for programmers and analysts, however due to the differing views of the 4+1 architectural view model, the document can also be of use to non-technical stakeholders.

# Purpose

The purpose of the SAD is to provide a comprehensive architectural overview of the system, using a number of different architectural views to depict the same system in different ways. It is intended to capture and convey the significant architectural decisions which have been made on the project.

The SAD methodology chosen is 4+1. 4+1 describes four views:

* Logical
* Implementation/Physical
* Development
* Data

These views provide different views using a commonality, the use case. The use case is an objective interpretation of the scope of the project while the views provide a specific perspective of said scope.

The software architecture chosen for the Vintage Vogue project is the content management system Joomla. Joomla was suggested to the group by the client. Additionally it is also a model view controller. An MVC, similar to 3 tier architecture, separates into business logic, input logic and UI logic, allowing independent development and testing of each component.

|  |  |  |
| --- | --- | --- |
| View | Document | Description |
| Logical | Sequence diagram | Provides a sequence of events describing how each function is handled throughout different components and entities |
|  | Package diagram | Displays the packages involved in the project and their dependencies |
| Physical | Deployment diagram | Provides a model of the physical components of the final product of the software |
| Development | Development diagram | Describes the overall component requirements of the software product |
|  | Component diagram | Displays how components communicate and transfer functionality and data to each other |
| Data | E-R diagram | A visual description of the contents of the database |
|  | Data dictionary | A description of each data value in the database |
| Use case | Use Case Diagram | Describes actions which are crucial to the operation of the system |
|  | Detailed use case description | A detailed description of the use cases in the diagram |

# Scope

The scope of the Vintage Vogue Project was initially developed along with the project charter and monitored through the 9 areas of project management document. The following scope definitions are taken from said documents.

|  |  |
| --- | --- |
|  | Requirements |
| 1 | Backup feature |
| 2 | User profiles |
| 3 | Increase/decrease stock |
| 4 | Publish/unpublish sold/unsold stock |
| 5 | Create new stock |
| 6 | Edit stock details |
| 7 | Disable profile |
| 8 | Create new profile(Register) |
| 9 | Editing profile |
| 10 | Call to action button |
| 11 | Paypal sandbox access |
| 12 | Provide a platform in which to demonstrate the web site |

|  |  |
| --- | --- |
|  | Boundaries |
| 1 | All payments is handled by paypal |
| 2 | No connectivity with web cameras |
| 3 | Web security is handled by Joomla! CMS and no further development will be made by the team |
| 4 | Four members are maintained throughout the entire project |
| 5 | Unreasonable changes to project scope may alter the development quality and schedule of the final product |
| 6 | The project is to be given to the client no later than 25th November |

# Architectural Representation

The document presents the software architecture for the Vintage Vogue Project using a 4+1 methodology. Each view gives a different perspective of the system for a different area of focus, represented by a series of UML diagrams; enabling stakeholders to view those sections which are most relevant to them. The following outlines the views which comprise the SAD, their focus and intended audience.

|  |  |  |
| --- | --- | --- |
| View | Audience | Description |
| Logical | Designers, software architect,  End users | Describes the functionality of the project and their effect to end-users and relationships between other system components |
| Development | Programmers, software managers | Sometimes referred to as implementation view, this view is to do with the technical design of the software, its components, and layers in the Vintage Vogue Project. |
| Physical | Future developers, Client, web administrators, systems engineer | Sometimes referred to as deployment view. The physical view gives a broad topological view of how the system components will physically connect and interact when the project comes to an end and is deployed. It can also serve as a general outline if another project team is to continue with or further develop the project |
| Data | Database Programmers, Administrators | Describes the method which the data for the system is logically stored. |
| Use case | All stakeholders | Use cases contain set of specific scenarios which describe functionality defined in the scope statement. It shows all architecturally significant requirements of the system |

# Architecture goals and constraints

Beside the client suggesting Joomla as a development platform for the Vintage Vogue project there are other goals and constraints to use the Joomla Content Management System. The one advantage of Joomla package is that it comes with a handler for both front-end and back-end.

The well developed back-end of Joomla is an advantage for future maintenance and development of the website. One of our goals is to create components which serve both front and back end, with the exception of administrator/super user rights.

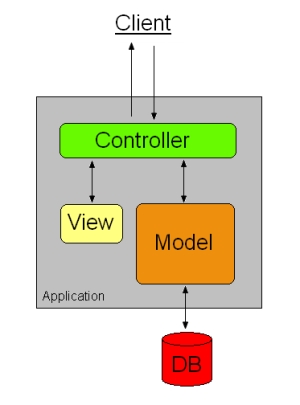
An MVC is similar to 3-tier architecture where the functionality of the system is broken up into parts which serve a particular purpose. Whilst 3 tier has presentation, business and database layers, the Joomla model view controller(MVC) contains controller, view and model. Each component represents a specific function and thusly each function can be maintained independently from each other.

The constraints that the group has imposed on Joomla is the default architectural constraints that Joomla can handle. For instance data structures and capacity has limits within mySQL. Another instance is Joomla has difficulty transferring string from components and our group has to work around the problem. Similarly we expect limitations within the Joomla architecture. Also it has been agreed that security will not be developed further aside from what Joomla and PayPal provides.

# Technology selection

The content management system chosen is Joomla whilst the back-end handling is operated by myphpAdmin and mySQL. Web forms are to be designed in HTML with the functionality provided through php code.

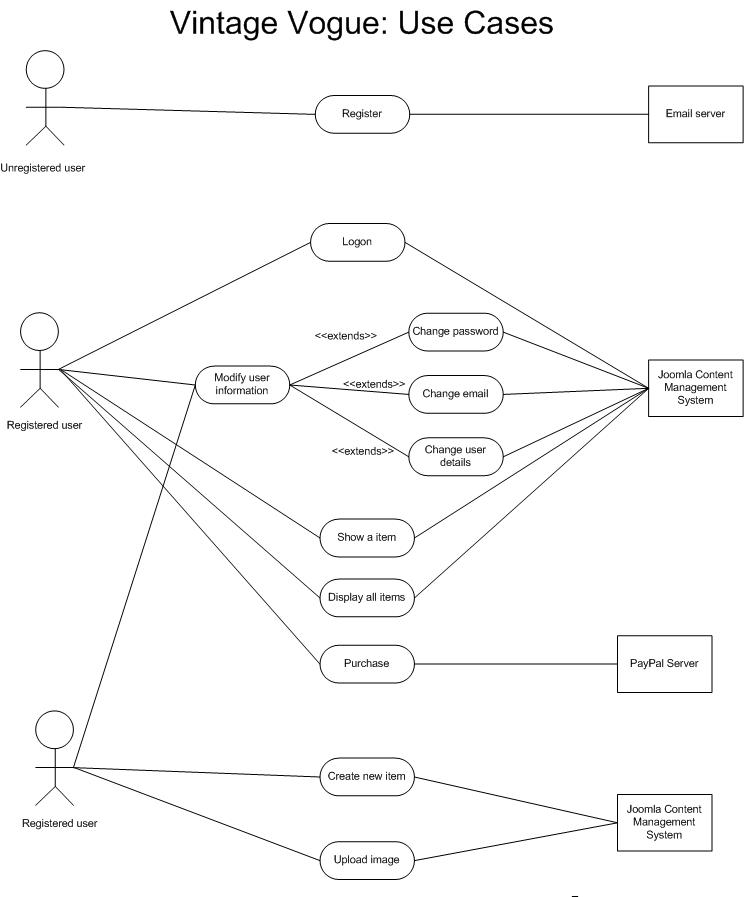
More specifically Joomla is a Content Management System based on the existing architecture of an MVC. Developers can independently create extensions or components which serve a single(or multiple) purpose allowing for easy extensibility and modification.

 <http://en.wikipedia.org/wiki/File:MVC_Diagram_3.jpg>

The diagram above shows the generic architecture for an MVC. Joomla naming conventions are different however with a J prefix i.e JController, JView, JModel.

The controller is responsible for interpreting commands from the user, which is then passed on to the model. The model comprehends that command and executes it appropriately. Finally the executed function is then passed into the view for the user to see.

# Use case



## Use case description

|  |  |
| --- | --- |
| **Use Case Name** | Register |
| **Stakeholders** | Unregistered user, Joomla content management system, administrator, super user |
| **Summary** | Registers a member of the Vintage Vogue team on the website |
| **Basic course of events** | 1. The User enters the website 2. The user clicks the register link 3. A registration form appears which the user must fill which includes name, contact details, email, username and password 4. The registration notification is sent to the super user or administrator of the system 5. The administrator or super user approves the registration 6. An email is sent to the user to confirm the email is correct 7. The link sends the user back to the website where the email is confirmed |
| **Alternative paths** | 1. In Step 5 , if the user is disapproved by the super user or administrator and email is sent notifying the failure of registration |
| **Preconditions** | * The user must have access to his/her email * The email address must be a valid email |
| **Postconditions** | * The user must validate the email which has been input in the registration form |

|  |  |
| --- | --- |
| **Use Case Name** | Logon |
| **Stakeholders** | Registered user, Joomla content management system |
| **Summary** | Allows a registered user to log on the website |
| **Basic course of events** | 1. The user enters the website 2. The user fills out the logon details (password and username) 3. The user is directed to its user page |
| **Alternative paths** | 1. In step 2, if the user inputs the wrong password or username the user will be notified to try again |
| **Preconditions** | The user must have a registered account on the website |
| **Postconditions** |  |

|  |  |
| --- | --- |
| **Use Case Name** | Upload image |
| **Stakeholders** | Administrator, super user, Joomla content management system |
| **Summary** | Allows the administrator or super user to upload an image for an item |
| **Basic course of events** | 1. The link is chosen to prompt the upload file function 2. The user selects which image is to be uploaded 3. Joomla CMS checks if the upload was an image 4. The image is copied into the component data where images reside |
| **Alternative paths** |  |
| **Preconditions** | The file must be an image |
| **Postconditions** |  |

|  |  |
| --- | --- |
| **Use Case Name** | Modify user information |
| **Stakeholders** | Registered user, Joomla content management system, Email server |
| **Summary** | Once a user has registered his/her account and has logged on, he/she can change its details |
| **Basic course of events** | 1. The user selects which field to change 2. The user inputs a new value into the selected field |
| **Alternative paths** | 1. If the user wishes to change email address, a new verification email will be sent to the new email to validate it |
| **Preconditions** | The user must have a registered account and is logged on |
| **Postconditions** | The user must verify the new email |

|  |  |
| --- | --- |
| **Use Case Name** | Display all items |
| **Stakeholders** | Registered user, unregistered user, super user, administrator, Joomla content management system |
| **Summary** | Retrieves a list of all the items for sale on Vintage Vogue |
| **Basic course of events** | 1. The user enters the catalogue page 2. Joomla accesses mySQL database and retrieves the all sets on sale 3. The data of the sets is loaded onto the webpage |
| **Alternative paths** |  |
| **Preconditions** | The mySQL database is operational, 1 or more sets exists |
| **Postconditions** |  |

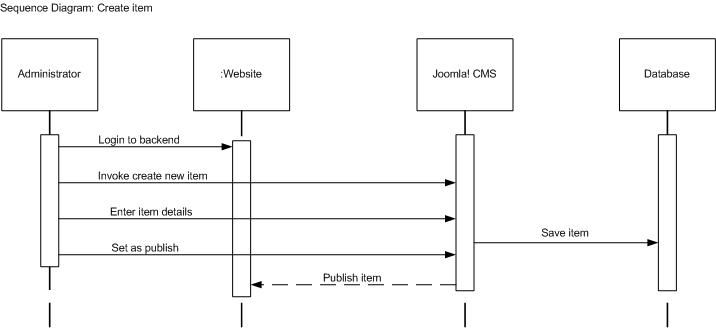
|  |  |
| --- | --- |
| **Use Case Name** | Create item |
| **Stakeholders** | Registered user, super user, Joomla content management system |
| **Summary** | Allows a registered user to create a new item |
| **Basic course of events** | 1. The user enters the website 2. The user logs on 3. The user is forwarded to its account page 4. The uploads picture and enters details about its set |
| **Alternative paths** | 1. If the user chooses to sell a single item then it is still stored in the database as a set, however it appears as a single item |
| **Preconditions** | The user must have a registered account on the website, the user must be logged on |
| **Postconditions** | The picture and details are saved on the database |

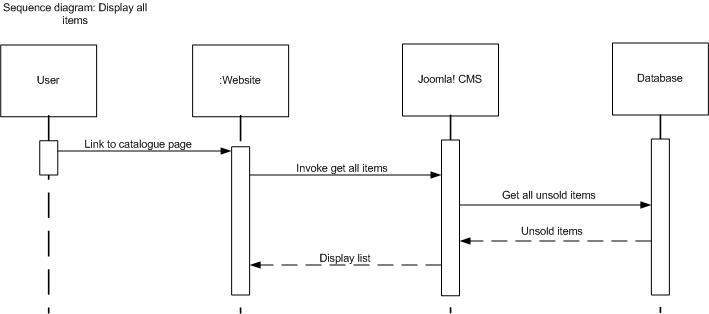
|  |  |
| --- | --- |
| **Use Case Name** | Show a set |
| **Stakeholders** | Registered user, unregistered user, super user, administrator, Joomla content management system |
| **Summary** | Specifically views details of a single item |
| **Basic course of events** | 1. The user enters the website 2. Use case “Display all sets” is activated 3. The user selects a single set to view 4. The database retrieves the details of the specified set 5. The information is uploaded onto the website |
| **Alternative paths** |  |
| **Preconditions** | A set must be selected |
| **Postconditions** |  |

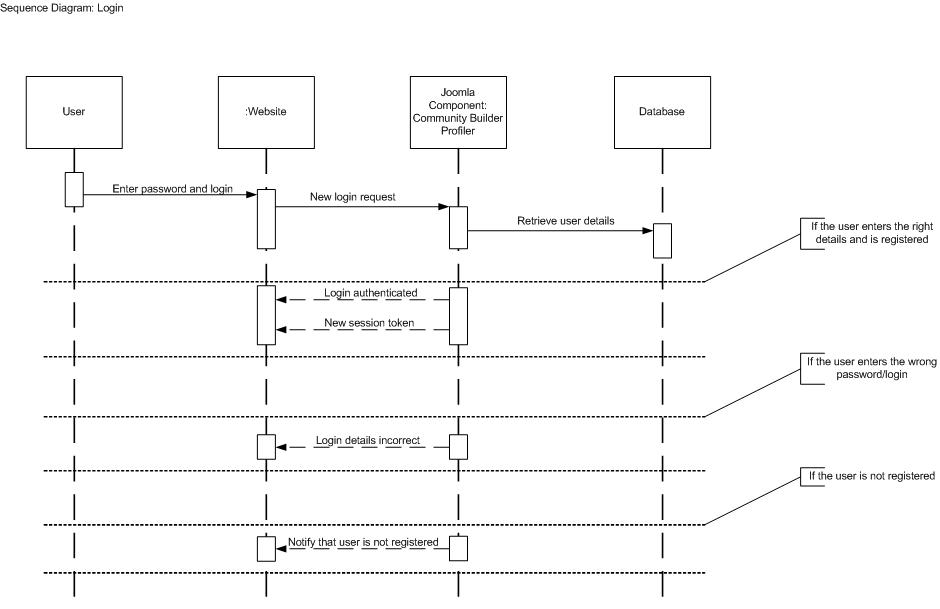
|  |  |
| --- | --- |
| **Use Case Name** | Purchase |
| **Stakeholders** | Registered user, unregistered user, PayPal server, Joomla content management system |
| **Summary** | Once a set has been selected a user can purchase said item. The payment is conducted in Paypal |
| **Basic course of events** | 1. The user selects the set which he/she wishes to purchase 2. The user is required to fill a form which inputs payment details 3. Once the payment details has been filled the payment process is redirected to PayPal where it is verified 4. Once the payment has been made, the branch which has the set is notified of the purchase and the super user/administrator is emailed with a notification of purchase 5. The database is updated to say that the set has been sold |
| **Alternative paths** | 1. In step 3, if the payment is unsuccessful the user is asked to try inputting the details again |
| **Preconditions** | The set must be on sale, the set is currently on sale |
| **Postconditions** | The user’s payment details must be verified |

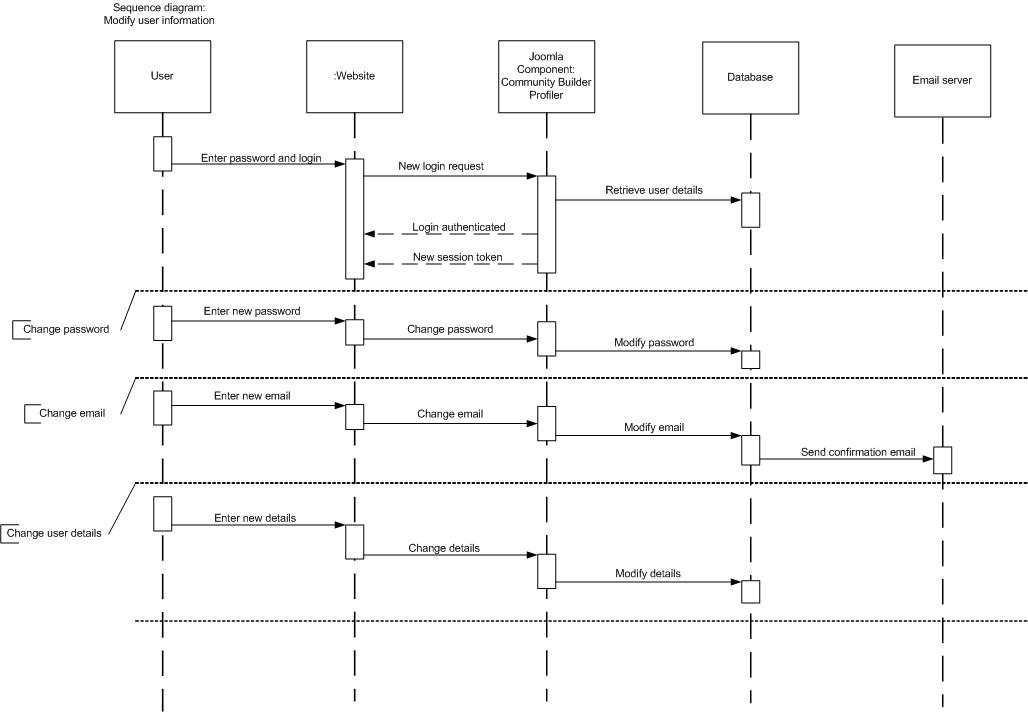
# Logical view

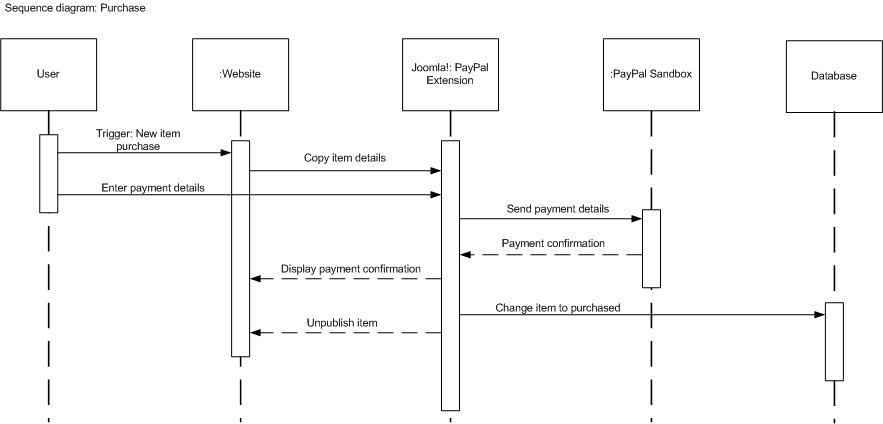
## Sequence diagrams

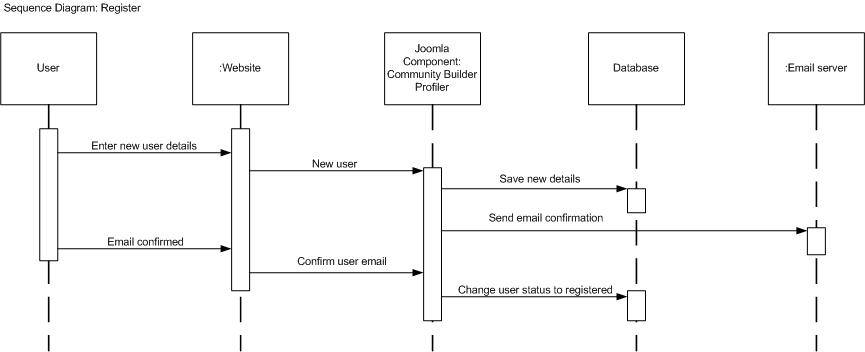


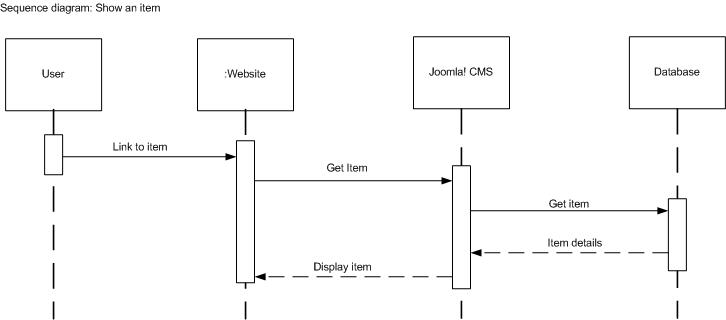


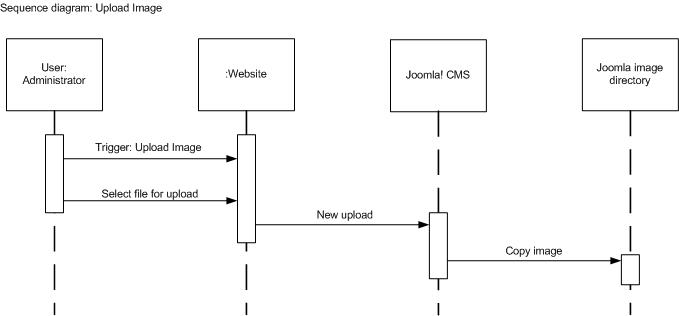








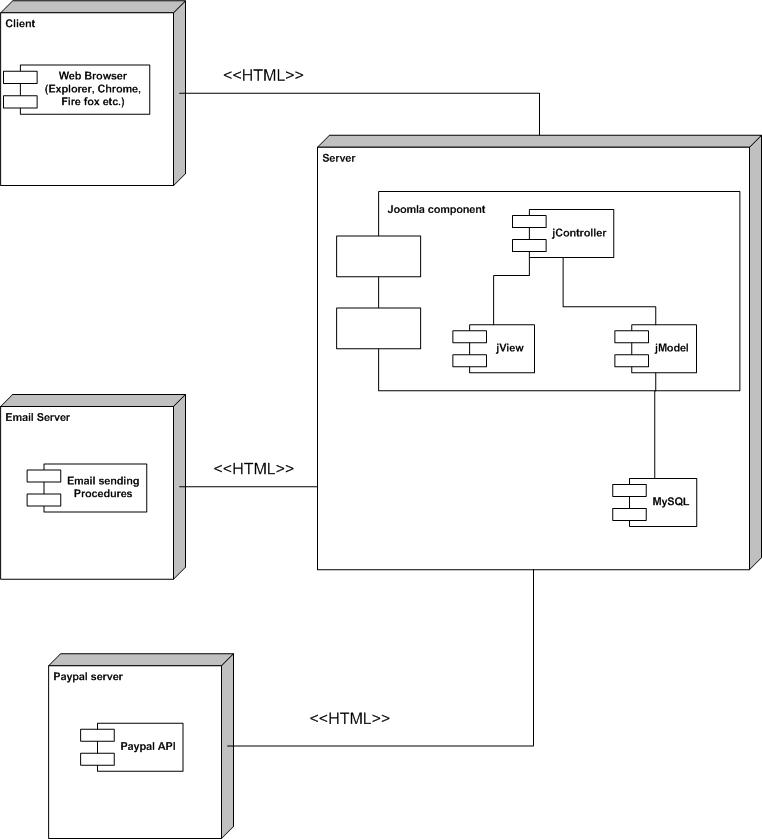




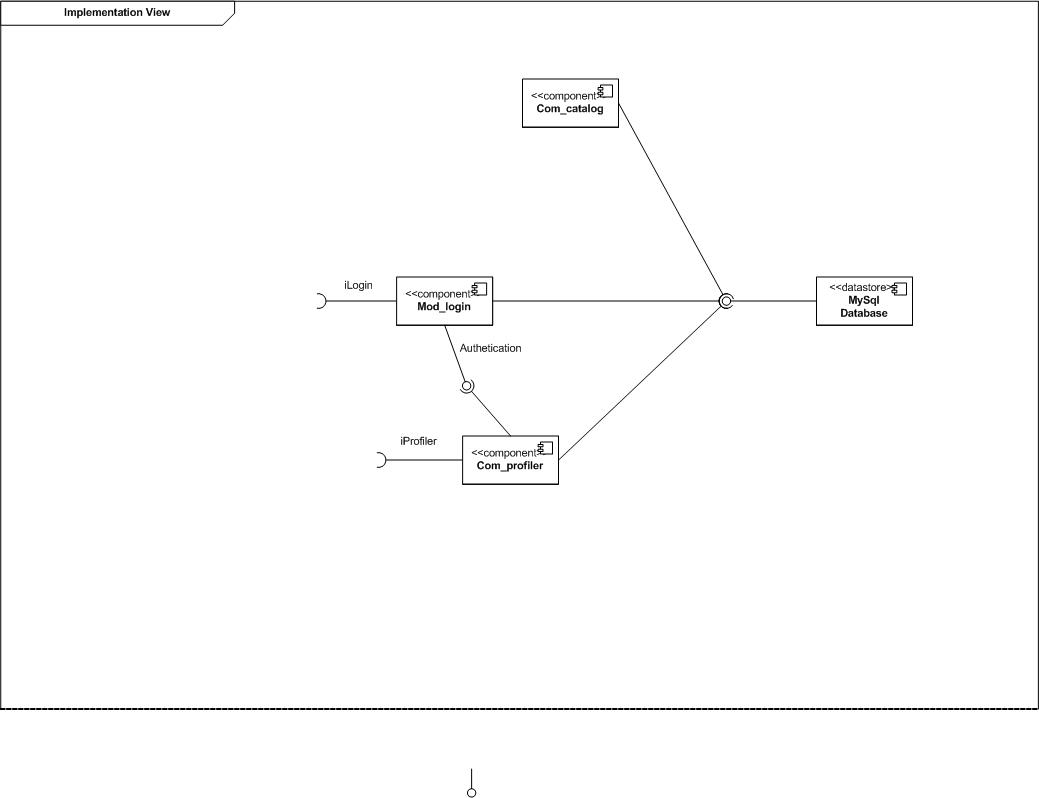
## Package diagram

# ppppdiagram.jpgDevelopment view

## Development diagram

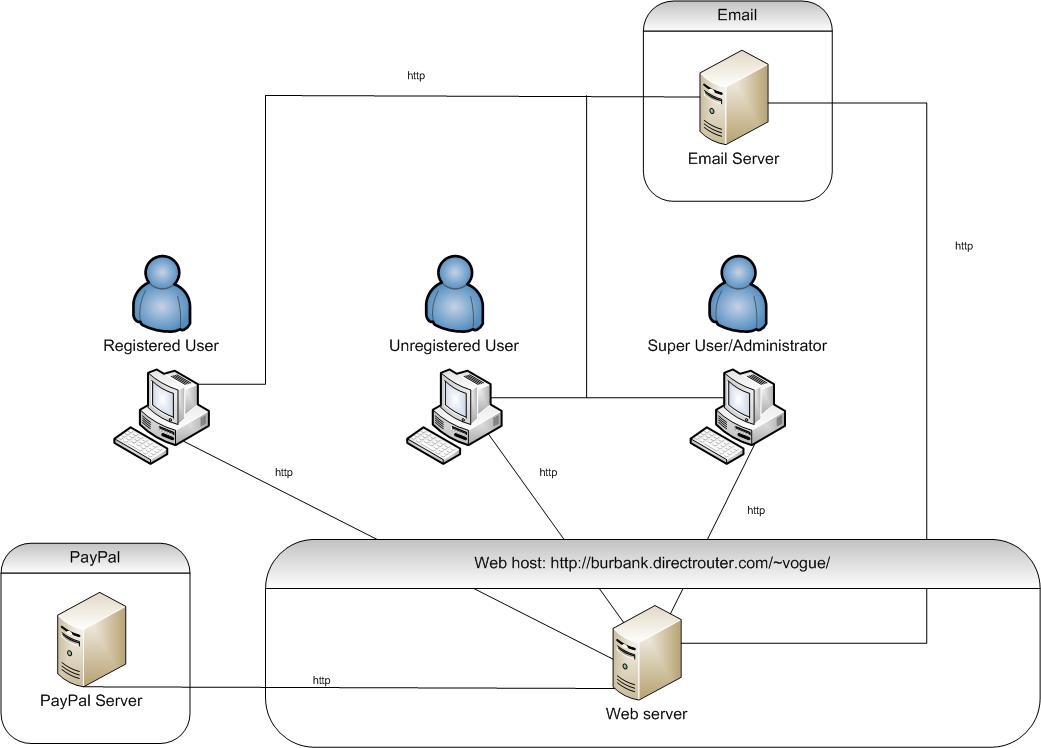


## Component diagram



# Physical view

## Deployment diagram



# Data view

## Data Dictionary

### Table: User

|  |  |  |
| --- | --- | --- |
| Field name | Field type | Description |
| **username** | Varchar | The users chosen username for login purposes |
| **Firstname** | varchar | The user’s first name |
| **Lastname** | Varchar | The user’s last or family name |
| **Password** | Varchar | The user’s account password |
| **Email** | Varchar | The user’s email address |
| **Group** | Varchar | The group which the user belongs to |
| **registerDate** | Datetime | The date which the user registered |
| **lastVisit** | Datetime | The last time the user logged on to the website |
| **enableStatus** | Bool | The status of the user e.g if they are disabled or enabled |
| **Confirmed** | Bool | The status of the user if they are a confirmed user or pending for approval in terms of content that the user has put up like profile picture |
| **approved** | Bool | A Boolean value to check if the user has been approved or pending for approval |
| **id** | Int | A unique id used for database purposes, it assures each username account is unique. Not seen by the user |

## Table: Set

|  |  |  |
| --- | --- | --- |
| Field name | Field type | Description |
| **st\_id** | int | The unique identifier for a set, not seen by the user |
| **St\_name** | varchar | The name of the set |
| **St\_price** | int | The cost in AUS$ for the set |
| **St\_status** | varchar | The current status of the advertised item(sold or unsold) |
| **st\_notes** | varchar | Notes about the set |
| **St\_alias** | longtext | Alias of the set |
| **Published** | int | Determines if the product is published |

### Table: Goods

|  |  |  |
| --- | --- | --- |
| Field name | Field type | Description |
| **Goods\_id** | Int | A unique identifier for an item, not seen by the user |
| **Goods\_image** | Varchar | An image stored as a varchar within the database |
| **Goods\_name** | Varchar | The name of the good |
| **goods\_price** | int | The cost in AUS$ for the good |
| **goods\_status** | Varchar | The current sale status of the good(sold or unsold) |
| **Goods\_category** | varchar | The category which the good belongs to(hat, shirt, pants, etc) |
| **goods\_size** | Varchar | The size of the good(If a shirt M XL etc, if shoes 11 12 etc) |
| **Goods\_colour** | Varchar | The dominant colour of the goods |
| **Goods\_notes** | Varchar | The particular notes for the goods |
| **Goods\_alias** | Varchar | Alias of the goods |
| **Published** | int | Determines if the product is published |

## E-R Diagram

