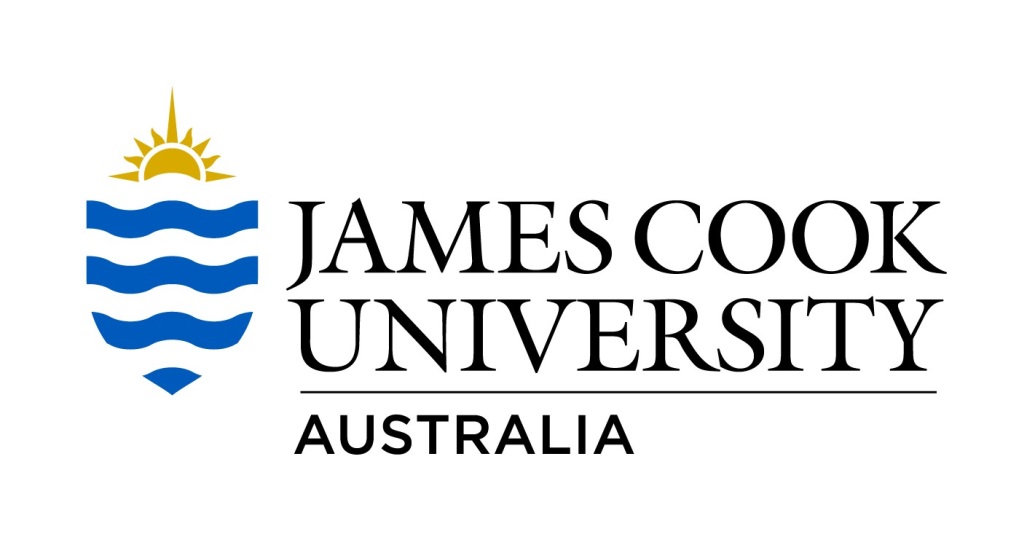
**SIM UNIVERSITY**

**SCHOOL OF SCIENCE AND TECHNOLOGY**



**CP5046 ICT PROJECT 1**

**Collectibles Auctions System on EBay Marketplace (CASE)   
(version 1.0)**

Team NAME : TEAM AWESOME  
PROJECT TEAM :

Karimullah Bin Syed Ahmed (12749720)   
Vimal Adithann (12741594)   
Aswath Baskaran (12686276)

**Jan 2013**

# Abstract

Ebay.com has been an industry leader over the last decade unbeatable in the ecommerce auctions industry and succeeding in brining unique products to the masses.

Over the past few years, EBay has been working on its own application programming interface to enable independent developers to drive their businesses using eBay’s inventory and infrastructure as a backbone. With the EBay API, developers have an easy and clear way of creating stable and complete business solutions fast and at low cost.

The Collectibles Auctions System on EBay marketplace (CASE) aims to capitalize on the functionalities and features provided by EBay to provide a risk free and hassle free environment to do transactions in. Users of the system will be able to buy items, sell items, browse for deals, get recommendations for items and leave feedback on items that they have bought.

The software development life cycle, database design, prototyping and project management skills have been fused together for this project. Each skill set will be used as a guideline during the implementation process. The entire planning and design process has been documented into this report and can serve as a guideline to manage other projects on developing integrated software systems.

Table of Contents

[Abstract 2](#_Toc344334911)

[1 Introduction 5](#_Toc344334912)

[1.1 Objectives 5](#_Toc344334913)

[1.2 Background 6](#_Toc344334914)

[1.3 User Background 7](#_Toc344334915)

[1.4 System Requirements and Objectives 7](#_Toc344334916)

[2 Proposed Method and Approach 9](#_Toc344334917)

[2.1 Identifying Software Development Lifecycle Approach 9](#_Toc344334918)

[2.2 Identifying Database Lifecycle 11](#_Toc344334919)

[2.3 Gathering User Requirements 13](#_Toc344334920)

[2.4 Identifying Hardware and Software Requirements 14](#_Toc344334921)

[2.5 Literature Review 15](#_Toc344334922)

[2.5.1 Software Design 15](#_Toc344334923)

[2.5.2 Software Implementation 15](#_Toc344334924)

[2.5.3 Database Design and Implementation 16](#_Toc344334925)

[2.5.4 EBay Application Development 16](#_Toc344334926)

[2.5.5 Contemporary Products 17](#_Toc344334927)

[3 Requirement Analysis 18](#_Toc344334928)

[3.1 Understanding EBay 18](#_Toc344334929)

[3.1.1 Selecting EBay for integration 18](#_Toc344334930)

[3.1.2 Current Trends 18](#_Toc344334931)

[3.1.3 EBay SDK 18](#_Toc344334932)

[3.1.4 EBay Sandbox 20](#_Toc344334933)

[3.1.5 EBay API 20](#_Toc344334934)

[3.1.6 Constraints 21](#_Toc344334935)

[3.2 Identifying System Users and Tasks 22](#_Toc344334936)

[3.3 Functional Requirements 23](#_Toc344334937)

[3.3.1 Use Case 23](#_Toc344334938)

[3.3.2 Use Case Description 24](#_Toc344334939)

[3.4 Non-Functional Requirements Description 26](#_Toc344334940)

[3.5 Implementation Requirements Description 26](#_Toc344334941)

[3.6 Interface Requirements Description 26](#_Toc344334942)

[3.7 Features Out of Scope 26](#_Toc344334943)

[4 System Design 27](#_Toc344334944)

[4.1 Identifying Entity, Boundary and Control Objects 27](#_Toc344334945)

[4.2 Identifying Associations and Aggregations 36](#_Toc344334946)

[4.3 Mapping Use Cases to Objects with Sequence Diagrams 37](#_Toc344334947)

[4.4 Identifying Attributes for Classes 47](#_Toc344334948)

[4.5 User Interface Design 50](#_Toc344334949)

[5 Database Design 54](#_Toc344334950)

[5.1 Entity Relationship Diagram 54](#_Toc344334951)

[5.2 Entity Type and Description 55](#_Toc344334952)

[5.3 Relationship Description 59](#_Toc344334953)

[5.4 Assumptions and Constraints 60](#_Toc344334954)

[5.5 Functional Dependencies and Normalization 60](#_Toc344334955)

[5.6 Relational Headings 62](#_Toc344334956)

[6 Prototyping 63](#_Toc344334957)

[7 Conclusion 63](#_Toc344334958)

[7.1 Summary 63](#_Toc344334959)

[7.2 Further Developments 63](#_Toc344334960)

[7.3 Challenges Encountered 63](#_Toc344334961)

[8 References 64](#_Toc344334962)

# 

# Introduction

The detailed objectives analysis of this is as follows.

## Objectives

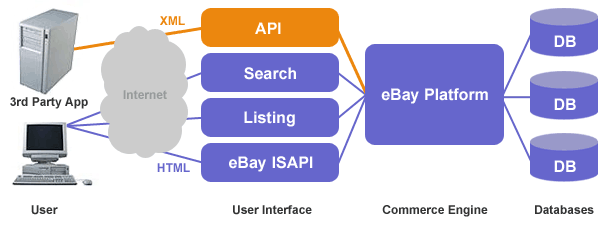
* To solve real world problems by facilitating solutions based on real users’ business requirements.
* To understand and practice software system integration of web elements, software design and database design.
* To build software and database from scratch based on the user requirements.
* Enhancement of non-technical skills such as report writing, time and project management and effective communication.
* To experience how to solve problems as the surface during the various phases of the project life cycle.

## Background

Ebay.com has been has been generating revenue from its auctions, offering a wide range of products worldwide.

The EBay developers program was founded in 2000, and is currently responsible for 25% of EBay listings to date. What the developer program does is allow third party applications to tap into EBay’s database and generate their own revenue.

EBay has released the application programmable interface (API) that can be used to communicate directly with the eBay database in XML, SOAP or HTTP format. By using the API, third party application can provide a custom interface, functionality and specialized operations not otherwise afforded by the original eBay interface. The API is not dependent on the eBay user interface, so it allows for a developer to create stable and customized features to suit the market’s business needs.



**Fig 1.1 EBay API operations cycle – courtesy www.ebay.com**

The API takes on the responsibilities of communication with the EBay databases and handles all transactions. Fig 1.1 illustrates the operations cycle of the EBay API. EBay generates unique tokens for each application to identify itself to the EBay platform. This allows for auditing and makes transactions secure.

The API affords us the opportunity to cater to the local market where people are always looking for the cheapest way to buy and the best way to get the best return for selling an item. The following are some of problems that users have with EBay.

P01. Local users are more comfortable buying and selling through a local company to minimize risks.

P02. Users want to buy items from reliable buyers. EBay’s search does not allow users to sort by seller ratings and it can be tedious for users to have to click on each item to know a sellers ratings.

## User Background

Mr. Zulfadli Mohd Amin holds a full time job as an operations executive at the Land Transport Authority. He is also an avid toy collector and works as a part time agent for other enthusiast. Over the past 2 years, his list of customers has grown and he is unable to cater to all the demands of his customers.

As such the user would like to work with Team Awesome to develop a web based shopping service that would allow his users to buy and sell their toys and other collectibles with minimum effort from him.

The team held extensive discussions with the user to fully understand his requirements and have translated them into system objectives that will be explained in the following sub section.

## System Requirements and Objectives

|  |
| --- |
| System Object S01 |
| The system needs to allow users to buy products online without having the need to interact with EBay. No EBay account should be required to use the system buy items. |

|  |
| --- |
| System Object S02 |
| The system needs to allow users to sell products online without having the need to interact with EBay. No EBay account should be required to sell items. |

|  |
| --- |
| System Object S03 |
| The system needs have a search functionality that allows users to search for products by categories. |

|  |
| --- |
| System Object S04 |
| Search functionality should be able to perform searches for items available only within Singapore. |

|  |
| --- |
| System Object S05 |
| The system has to allow a buyer to view the product details before making the purchase. |

|  |
| --- |
| System Object S06 |
| The system needs to be able to notify users via email with appropriate information once a purchase has been made or an item has been put up for sale. |

|  |
| --- |
| System Object S07 |
| The system needs to allow sellers to set their own buying and bidding prices. |

|  |
| --- |
| System Object S08 |
| The system needs to be able to allow buyers to feedback on the transaction and products. |

|  |
| --- |
| System Object S09 |
| The system needs to be able to recommend items to buyers based on search category that they have selected |

|  |
| --- |
| System Object S10 |
| The system needs to be able recommend the top searches that other are using to enhance their purchasing options based on the selected search category. |

|  |
| --- |
| System Object S11 |
| Users should have a summary view of all their transactions e.g. (products sold, products bought, feedbacks made). |

# Proposed Method and Approach

## Identifying Software Development Lifecycle Approach

Various software development process models are available and each has been well documented with advantages and shortcomings.

The model used in this project is the V-Model which is illustrated on Figure 2.1 below.

System Requirement & Requirement Elicitation

Requirements Analysis

Design

Implementation

Unit & Integration Testing

System & Integration testing

Operation & Maintenance

Acceptance Testing

**Validate Requirements**

**Verify Design**

**Figure 2.1 the V-Model**

**(Referencing Page 630 of OOSE: Using UML, Patterns and Java [1])**

The V-model uses a well-balanced method in which each phase can be implemented using the documentation of the phase before it. It also saves time by allowing phases to be done concurrently. The V-model has several phases.

The Verification Phases are on the Left hand side of the V, the implementation Phase is at the bottom of the V and the Validation Phases are on the Right hand side of the V. The left and right side of the V are also linked at several stages implying that when problems are discovered at the validation stages, the verification stages can be reworked. This opens up aspects of the SDLC like iterations and re-workings that are hidden in other models like the water fall model.

The waterfall model was not adopted as iterations and rework aspects are hidden and thus deemed impractical in the real world.

The spiral model is known for its complexity and needs a learning curve to see its real potential. Time is also a factor as steps in spiral model need to be further expanded.

**System Requirement Analysis and Elicitation**

* Understanding the purpose of system
* Understanding the background
* Identifying Hardware and Software Requirements
* Literature Review

**Requirement Analysis Phase**

* Understanding Interaction with Ebay
* Gathering User Requirements
* Identify Functional Requirements with Use Cases
* Identify Non-Functional and Implementation Requirements

**Design Phase**

* Transform Use cases from previous phase into classes
* Identify Associations between classes
* Identify Attributes and Methods for classes
* Create sequence diagrams to show interaction between classes

**Prototyping**

* Use of Rapid Application Development for prototyping
* Evaluation by user

**Implementation Phase (ICT 2)**

* Transform Design model into source code

**Testing Phase (ICT 2)**

* Test classes
* Test User Interface
* Test class interaction with Database

## Identifying Database Lifecycle

Conceptual Design

Logical Design

Physical Design

Planning

Requirement Analysis

Implementation

Testing

Database Design

DBMS Selection

**Fig. 2.2 Stages of the Database Development Lifecycle**

**(Referencing Page 272 of Database Systems: A Practical Approach to Design [2])**

The database development lifecycle comprises of the following steps:

**Planning Phase**

* Understanding the project background
* System Requirements and Project Objectives

**System Definition Phase**

* Identifying Users
* Identifying system boundaries

**Requirement Analysis Phase**

* Gather User requirements
* Literature Review

**Design Phase**

* Identify DBMS to use
* Identify attributes, data type
* Identify relationship description
* Create an Entity Relationship Diagram
* Normalization

**Prototyping Phase**

* Creating Tables
* Populating Tables

## Gathering User Requirements

An important part of a software development project is the requirements gathering portion. It clearly highlights what is to be achieved. My strategies for effectively gathering user requirements are:

Brainstorming  
Brainstorming can be an effective way to generate lots of ideas on a specific issue by thinking from the user’s perspective. These ideas were then prioritized in the level of importance and implemented accordingly.

Case Study  
By studying similar applications, we can gather information on how feasible the proposed project is, what solution is being used by that system and the good and bad of the system. We can use this knowledge to refine our project goals and implementation targets.

Interviews  
Interviews with the user to understand the functional and non-functional requirements that would be acceptable to the user in the final product. The requirements would include system, environmental and user interface.

## Identifying Hardware and Software Requirements

The hardware and software requirements are identified. The costing in respect to the resource needed is also included. Please refer to the Resource Plan at Appendix A1.

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Resource** | **Cost** | **Resource Source** |
| Development | | |  |
| 1 | Visual Studio.Net 2010 Express Edition for development | Free | http://www.microsoft.com/express/downloads/ |
| 2 | MySQL Database Server for development | Free | http://www.mysql.com/ |
| 3 | Microsoft .Net Framework 4.0 | Free | http://www.microsoft.com/downloads/details.aspx?FamilyId=333325FD-AE52-4E35-B531-508D977D32A6&displaylang=en |
| 4 | EBay SDK for .Net / Windows | Free | http://developer.ebay.com/developercenter/windows/sdk/ |
| 5 | Development machine | Free | Student Asset |
| Production (ICT 2) | | |  |
| 5 | Webhosting with MYSQL Database | $10 per mth | http://www.godaddy.com/Hosting/web-hosting.aspx?ci=9009&isc=goazsg300c |

## Literature Review

A literature review is to keep a reader updated on the knowledge that is current to the specific field the report is addressing. This section discusses the collection of books, websites and articles used in the research and design of ICT 1.

A simple summary of the texts that we have based my research of the report are as follows.

### Software Design

The Book, **Object-Oriented Software Engineering, using UML, Patterns and Java TM**, has allowed me to understand the more granular tasks involved in the steps of a software development lifecycle. The book explains tasks involved from Requirements Elicitation to Analysis to Design to Implementation thru to testing and maintenance.

**Chapter 4: Requirements Elicitation** (Page 122-150)   
Explains gathering user requirements and understanding the background and environment of a project before any form of study is started. It explains gathering of Functional and Non-functional requirements and implementation requirements.

**Chapter 5: Analysis** (Page176-188)   
Explains various steps of converting use cases (functional model) into an analysis model and subsequently into a design model. It also includes the identification of entities, their associations, aggregations, attributes and methods.

**Chapter 6: System Design** (Page 228-254)   
Explains how to make classes and their subsystems can be made more robust and to adhere to proper design standards and goals. It also discusses coupling and cohesiveness.

### Software Implementation

The Book, **Object-Oriented Software Engineering, using UML, Patterns and Java TM**, has allowed me to understand the more granular tasks involved in the steps of a software development lifecycle. The book explains tasks involved from mapping class and sequence diagrams to source code to testing classes and the interaction between classes.

**Chapter 10: Mapping Models to Code** (Page394-422)   
Discusses how to map the design model into source code. The book uses the Java language as a baseline, but the object oriented concept can be used for any Object Oriented language which in our case is VB.Net.

**Chapter 11: Testing** (Page 443-472)   
Explains the testing concepts. It explains Unit testing which is based on the classes themselves and subsequently the system testing which covers the interaction of classes inside the system as a whole.

### Database Design and Implementation

The book **Database Systems: A Practical Approach to Design, Implementation, and Management**, gives a clear and concise view of founding an effective database and also delves into the implementation and management issues that may consequently arise.

**Chapter 3: The Relational Model** (Page 70 - 86)

This chapter explains the concepts of relational modeling and illustrates with suitable diagrams and explanations, the relationships, and referential integrity rules.

**Chapter 9: Database Planning, Design, and Administration** (Page 269 – 301)

This is an overview of the major stages involved in a database development lifecycle and goes further to elaborate the main activities in each stage. Please refer to chapter 2.2 of this report of the overview of a Database Development Lifecycle.

### EBay Application Development

The book, **eBay Application Development,** explains in great detail what EBay offers to developers and how the EBay API can be used in a third party application using VB.Net

**Chapter 3: Introduction to EBay SDK** (Page 40-69)   
This chapter introduces the EBay SDK in simple terms and its capabilities in general.

**Chapter 5: Managing Items with EBay SDK** (Page 103 – 121)  
This chapter explains how items are listed in EBay and how the SDK can be used to buy and List items.

**Chapter 8: Using the EBay API** (Page 172-192)   
Delves into the API and the calls and references that can be used within a software.

**Chapter 9: Using the EBay API within a Web Site** (Page 192 – 278)  
Explains how to interact with EBay and provides a lot of examples with syntax and scenarios.

### Contemporary Products

There are contemporary products available online which we can base our design model upon when building the solution.

**Half.com**This site uses EBay’s platform to run their own e-commerce business. It sells any EBay product that is available for buying immediately but most items are only available to customers in the United States.

Some features that are available in my system that is not found in Half.com are as follows:

* Ability to allow users to list their items
* Searching only the highly rated sellers
* Available for use by people outside the US, mainly Singapore

# Requirement Analysis

## Understanding EBay

### Selecting EBay for integration

The main reason to integrate with EBay is very simple: the potential of having millions of customers. As long as the application is involved in an industry of selling products or services, numbers is the key to generate revenue and leveraging on EBay makes sense.

### Current Trends

There are many third party websites and applications which do not integrate with EBay but rather use a process called screen scraping where they programmatically parse the HTML that makes up a page in EBAY and then using the result to drive their applications.

This method needs users to have their own EBay Ids as the application is merely a output and not making any calls natively. Screen Scraping is also not allowed by EBay as it allows users to perform bid sniping and gain an advantage at the last few seconds of an item bid’s closing. Also, from a technical perspective, every time EBay changes a portion of their web UI, the application driven by shell scrapping will fail. So this approach will not be adopted for this project.

To allow developers a safe and legal way of accessing eBay, the company has release its own API. This will be discussed in detail in the following section.

### EBay SDK

The EBay SDK provides a high level interface to the features of the EBay API. This allows a developer to concentrate of building service rich and customized solutions rather than worrying about the underlying complexities of interfacing with EBay.

EBay SDK is available for Visual Studio.Net and allows a developer to write in any language that complies with the .Net framework. This project makes use of VB.Net for all coding needs.

Under the EBay SDK lies its heart, the EBay API. It is different from traditional APIs like JAVA Swing, where it is an XML driven application. It comprises of a series of predefined XML tags and attributes. Developers use these tags to send and collect information from EBay.

The advantages of this would be that it is ideal for Web based solutions as the only requirement for the EBay API is Https and XML support. The downside however is that if API calls are made without care, the XML based API could deliver thousands of results which are normally not needed and hence slowing down the system.

The following fig 3.1 illustrates the relationship between the eBay SDK, API and EBay platforms in detail.

Conventional  
EBay Users

My Application

INTERNET

EBay SDK

EBay ISAPI

EBay API  
Handler

EBay Platform

EBay Database

**Fig. 3.1 the Relationship between the EBay SDK, API and EBay platform**

**(Referencing Page 8 of EBay Application Development)**

### EBay Sandbox

EBay provides developers with a virtual environment to test the developed application before publishing the application to the production environment. As such, EBay addresses its test environment as Sandbox and its real-time environment as production. For the purpose of this report, we shall use the same terminology.

Access to EBay sandbox is granted to registered developers. A set of unique Sandbox keys are generated. Any effort to access EBay Sandbox must be made through these keys. Sandbox Keys will not work in production environment and vice versa.

### EBay API

The EBay API has several different sets of interfaces that can be used to suit different purposes of an application. The following is a brief introduction of the 3 prominent APIs being offered. Each API comes with its own reference calls.

Searching on EBay  
**Finding API**: Search for items based on user’s specific criteria.

Selling on EBay  
**Trading API**: Secure, authenticated access to private eBay data. List or buy eBay items.

Buying on EBay  
**Shopping API**: Search for and browse eBay items fast! Best API for widgets

The Finding API is a newly introduced API and is still undergoing bug fixes. It is also not compatible with EBay Singapore. Another drawback is that the Finding API does not work in EBay sandbox environment and is intended for applications already in the production environment. For practical purposes, we shall only use the Finding and Trading API.

#### Trading API

The eBay Trading API offers secure, authenticated access to private eBay data. The eBay Trading API can be used to for buying items listed and for putting up items for bid.

**Formats:** XML, SOAP  
**Protocols:** HTTPS POST

**EBay Trading APIs support the following popular actions:**

* Submit a listing to eBay.
* Purchase items from EBay

#### The Shopping / Finding API

The Shopping API is optimized for response size, speed and usability. It is primarily used as a searching tool and offers many customizations to streamline the users search criteria.

**Formats**: XML, SOAP, Name Value, JSON   
**Protocols**: HTTP GET (REST) and POST

**EBay Shopping APIs currently support the following actions:**

Item Search

* Simple keyword and SellerID based search for items on eBay
* Advanced search for items on eBay via keyword, ProductID, SellerID and several search filters.
* Search for products on eBay via keyword or ProductID.

Item Data

* Simplified buyer specific view of item data
* Retrieve changing item information towards the end of an auction
* Retrieve item shipping details
* Retrieve multiple item data in a single request.

User Reputation

1. Retrieve eBay user profile and feedback information.

EBay Buying History

1. Retrieve popular and related search keywords by category or keyword.
2. Retrieve popular items by category

### Constraints

As of the time of the writing of this report, EBay does not allow any buying of items through the APIs in the production environment. Such operations can only be done in the sandbox environment.

For an application to get approval from EBay to perform buying operations in the production environment, the application has to be successfully published and working in the sandbox environment. EBay will then put the application through an approval process. This is done to ensure that third party applications do not include bid sniping mechanisms or other form of unethical or malicious mechanisms.

Once an application is approved, with new keys given by EBay, testing can begin in production, but this testing is chargeable as every operation is considered an actual transaction.

So for the scope of this report and the prototype developed, all design, implementation and demonstrations will be based on the sandbox environment.

## Identifying System Users and Tasks

|  |  |  |
| --- | --- | --- |
| **No.** | **User** | **Tasks** |
| 1 | User | * Buy Items * Perform Search for Items * Leave feedback for bought items * View List of transactions * Put Item for Sale * View Detail of an Item |
| 2 | Administrator | * Delete Listed Items * Send messages to users * Delete user account |

## Functional Requirements

### Use Case

**E-Commerce Auctions System for EBay Marketplace**

includes

**Customer**

includes

includes

**Administrator**

**Figure 3.3: Use Case Diagram of the Functional Requirements**

**(Referencing Page 31 of Object Oriented Software Engineering, Using UML, Patterns and Java )**

### Use Case Description

|  |  |  |
| --- | --- | --- |
| **Actor** | **Use Case** | **Process Description** |
| Customer | View Item Details | 1. User clicks on a search result item. 2. System will show the following info:    1. Item’s Name    2. Item’s Location    3. Item’s Cost and Shipping    4. Item’s Description    5. Seller information    6. Option to Purchase Item |
| Customer | Buy Item | 1. User clicks on the “Buy” button for an item 2. System allows user the following choices:    1. Edit Shipping information    2. Proceed to buy. 3. User completes transaction by proceeding to buy. |
| Customer | Make Payment | 1. User Clicks on Buy Item to complete payment 2. System Allows user to enter payment details 3. If payment is approved, item is bought or else, reason for failure is displayed. |
| Customer | Search for Items | 1. User Chooses and category and enters a search criteria. 2. System processes search and displays all the results to user. |
| Customer | Fetch EBay Categories | 1. User clicks on search item or sell an item. 2. System retrieves the categories currently on EBay and presents to user. |
| Customer | Put Item for Sale | 1. User Logs in and Clicks on Sell Item.    1. User Selects categories to sell item under    2. User provides item name and description    3. User provides selling price    4. User provides condition of item    5. User provides shipping cost    6. User provides item picture 2. System processes the order with EBay and notifies user of outcome.    1. Items put for sale successfully are also confirmed via email to user. |
| Customer | View User Transaction History | 1. User Logs in 2. System displays the following information    1. Total Items sold and their information    2. Total items bought and their information    3. Total bought items to leave feedback for |
| Customer | Rate Bought Items | 1. User Clicks on an item to leave feedback for 2. System allows user to place a feedback for that item selected. |
| Administrator | View User Transaction History | 1. Administrator logs in and enters userid of a user 2. System displays the following information    1. Total Items sold and their information    2. Total Items listed and pending sale.    3. Total items bought and their information   Total bought items to leave feedback for |
| Administrator | Delete User Account | 1. Administrator logs in and enters user-id of a user 2. System allows administrator to delete the account of the user.    1. User account is permanently deleted after a conformation |
| Administrator | Send Payment to Seller | 1. Administrator Receives payment for a successfully sold item 2. Administrator sends payment minus fees to user who sold item. |

## Non-Functional Requirements Description

Requirements that are not related to the functional behaviour of the system are as follows:

* The Web Server and Database server must be up 24/7.
* The response time of the database must not take more than 7 seconds.
* The response time from EBay must be less than 10 seconds.
* A transaction once completed cannot be duplicated.
* There must be data integrity and no data redundancy in the generated results.

## Implementation Requirements Description

The constraints imposed by the implementation of the system are as follows:

* For implementation, web server must be windows and .Net compatible
* Database server must support MySql format.

## Interface Requirements Description

The constraints imposed by external systems are as follows:

* System performance will be affected by any problem at EBay sites.
* System may have inconsistency issue due to bugs found in the EBay API.
* Currency conversion rates will be based on EBay’s conversion rates.

## Features Out of Scope

* The payment feature will not be included in the design or the implementation in ICT1 and ICT2 after discussion with the user.
* Payment will require integration with payment merchants like Paypal and the user is not willing to pay for the administrations charges or the SDK.

# System Design

## Identifying Entity, Boundary and Control Objects

This section aims to use the use cases derived from the functional requirements and identify the entity, control and boundary objects from them.

**Fig 4.1 - Use Case Diagram depicting a customer viewing an Item.**leshaseOrderssociations and Aggregations the seller who is a registered user int he nd displays it to the user. and boundary ob

**ViewSelectedItem**

**Customer**

includes

**Table 4.1 – Objects for the ViewSelectedItem use case**

|  |  |  |
| --- | --- | --- |
| **Object Name** | **Object Type** | **Description** |
| Customer | Entity | The user of the system who views Items. A Customer view items for understanding the Item’s information. Customers are identified uniquely by their email address as usernames. |
| Item | Entity | An Item that a customer can view. Item consists of information like item cost, location, description and condition. |
| ViewButton | Boundary | The link or button used by a customer to view an item/ initiate the ViewSelectedItem usecase. |
| ViewItemForm | Boundary | This form is used for output to display information about a item selected for viewing. |
| ViewItemControl | Control | This object is created when an Item is selected for viewing. This control queries EBay for the Item’s details and displays to user. |

**Fig 4.2 - Use Case Diagram depicting a customer viewing an Item.**

**BuyItem**

**Customer**

includes

**Table 4.2 – Objects for the BuyItem use case**

|  |  |  |
| --- | --- | --- |
| **Object Name** | **Object Type** | **Description** |
| Customer | Entity | The user of the system who buys items. After viewing an Item, user can select payment mode and make payment for item. |
| Payment | Entity | Payment information that is derived from Item and submitted to Payment Merchant. |
| PurchaseOrder | Entity | The purchase information is generated into an order. |
| BuyButton | Boundary | The button to initiate this use case |
| MakePaymentControl | Control | This object interfaces with Payment Merchant and deducts the payment accordingly. |
| BuyItemControl | Control | This object is created when an Item is triggered to be bought. This control tells EBay that the Item has been bought and saves transaction history to database. |

**Fig 4.3 - Use Case Diagram depicting a customer searching for items.**

**SearchForItems**

**Customer**

includes

**Table 4.3 – Objects for the SearchForItems use case**

|  |  |  |
| --- | --- | --- |
| **Object Name** | **Object Type** | **Description** |
| Customer | Entity | The user of the system who searches for items by providing a search term and selecting a Ebay Category to search under |
| ebayCategory | Entity | The categories that EBay allocates items under. |
| SearchButton | Boundary | The button to initiate this use case. |
| SearchForm | Bounday | The input form for user to specify the search criteria. |
| FetchCategoryControl | Control | This object interfaces with EBay Databases to get all categories available for searching. |
| SearchItemControl | Control | This object is created when a search is to be made. The category and search term and supplied to EBay and search results are displayed to user. |

**SellItem**

**Fig 4.4 - Use Case Diagram depicting a putting up an item for sale**

**Customer**

**Table 4.4 – Objects for the SellItems use case**

|  |  |  |
| --- | --- | --- |
| **Object Name** | **Object Type** | **Description** |
| Customer | Entity | The user of the system who puts an item for sale after specifying all the required item details. |
| ebayCategory | Entity | The categories that EBay allocates items to be sold under. All items sold and bought must belong to a EBay category |
| SalesOrder | Entity | The information of the item listed in Ebay is generated into a Sales Order. |
| SellButton | Boundary | The button to initiate this use case. |
| SellItemForm | Boundary | This form is for input by user specifying the item to be sold. |
| FetchCategoryControl | Control | This object interfaces with EBay Databases to get all categories available for searching. |
| SellItemControl | Control | This object is created when a sale is to be made. The category and item details and supplied to EBay and a listing at EBay is attemped. The outcome, success or failure is displayed to the user. |

**Fig 4.5 - Use Case Diagram a user viewing all his/her transaction history**

**ViewUserTransaction**

**Customer**

**Table 4.5 – Objects for the ViewUserTransaction use case**

|  |  |  |
| --- | --- | --- |
| **Object Name** | **Object Type** | **Description** |
| Customer | Entity | The user of the system who has made transaction in the system. |
| SalesOrder | Entity | An order object of an item that is being sold or sold |
| PurchaseOrder | Entity | An order object of an item that has been purchased by a customer. |
| Feedback | Entity | Feedbacks for items the user has bought or feedbacks the user has to make. |
| ViewButton | Boundary | The button to initiate this use case. |
| TrasactionForm | Boundary | This is the form to output all the transactions for the user |
| TransactionControl | Control | This object queries the database for all items that have been bought by the user and displays to the user.  This object queries that database for all items the user has put up for listing on EBay or successfully sold and displays to the user.  This object queries for any pending feedbacks the user has to make, or past feedbacks the user has made and displays it to the user. |

**Fig 4.6 - Use Case Diagram depicting a customer leaving a feedback for a purchased item.**

**RateBoughtItem**

**Customer**

**Table 4.6 – Objects for the RateBoughtItem use case**

|  |  |  |
| --- | --- | --- |
| **Object Name** | **Object Type** | **Description** |
| Customer | Entity | The user of the system who has made transaction in the system. |
| Item | Entity | An Item that a fits the customer has bought and needs to leave a feedback for. |
| Feedback | Entity | Feedbacks for items the user has bought. |
| RateButton | Boundary | The button to initiate this use case. |
| FeedbackFrom | Boundary | The feedback form that the user would submit with details about an item bought. |
| FeedbackControl | Control | This object queries the database for all items that are pending feedbacks. |

**Fig 4.7 - Use Case Diagram an administrator viewing all his/her transaction history for a particular user**

**ViewUserTransaction**

**Administrator**

**Table 4.7 – Objects for the ViewUserTransaction use case**

|  |  |  |
| --- | --- | --- |
| **Object Name** | **Object Type** | **Description** |
| Administrator | Entity | The user of the system who has made transaction in the system. |
| SalesOrder | Entity | An order object of an item that is being sold or sold |
| PurchaseOrder |  | An order object of an item that has been purchased by a customer. |
| Feedback | Entity | Feedbacks for items the user has bought or feedbacks the user has to make. |
| ViewButton | Boundary | The button to initiate this use case. |
| ViewTransactionForm | Boundary | The form where the transaction history of a user is output for the administrator to view. |
| TransactionControl | Control | This object queries the database for all items that have been bought by the user and displays to the user.  This object queries that database for all items the user has put up for listing on EBay or successfully sold and displays to the user.  This object queries for any pending feedbacks the user has to make, or past feedbacks the user has made and displays it to the user. |

**Fig 4.8 - Use Case Diagram an administrator deleting a registered user’s account**

**DeleteUserAccount**

**Administrator**

**Table 4.8 – Objects for the DeleteUserAccount use case**

|  |  |  |
| --- | --- | --- |
| **Object Name** | **Object Type** | **Description** |
| Administrator | Entity | The user of the system who has made transaction in the system. |
| DeleteButton | Boundary | The button to initiate this use case. |
| DeleteAccountForm | Boundary | Input form for administrator to specify which user account to delete. Also, the result of the operation is output here. |
| DeleteAccountControl | Control | This object queries the database for all registered account and displays to user. |

**Administrator**

**SendPayemntToSeller**

**Fig 4.9 - Use Case Diagram an administrator deleting a registered user’s account**

includes

**Table 4.9 – Objects for theSendPaymentToSeller use case**

|  |  |  |
| --- | --- | --- |
| **Object Name** | **Object Type** | **Description** |
| Administrator | Entity | The user of the system who has made transaction in the system. |
| SalesOrder | Entity | An order object of an item that is being sold or sold |
| Payment | Entity | Payment information that belongs to the item that has been sold. |
| SendPaymentButton | Boundary | The button to initiate this use case. |
| SendPaymentForm |  | Output to inform administrator that payment has been debited to seller’s account. |
| SendPaymentControl | Control | This object sends the payment approved to the seller who is a registered user in the system. |

## Identifying Associations and Aggregations

**PurchaseOrder**

**Customer**

**Item**

**SalesOrder**

**PurchasePayment**

**SalesPayment**

**Feedback**

**1**

**\***

Submits

**1**

**\***

Sells

**1**

**1**

processes

**1**

**1**

auctions

deducts

deposits

Rates

**Class Diagram for** **E-Commerce Auctions System for EBay Marketplace**

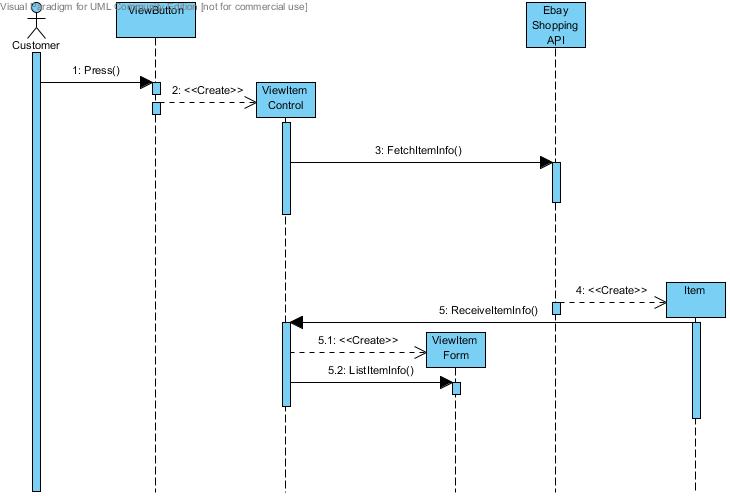
**Figure 4.11: Class Diagram of the Entities Indentified from Use Cases**

**(Referencing Page 190-193 of Object Oriented Software Engineering, Using UML, Patterns and Java )**

## Mapping Use Cases to Objects with Sequence Diagrams

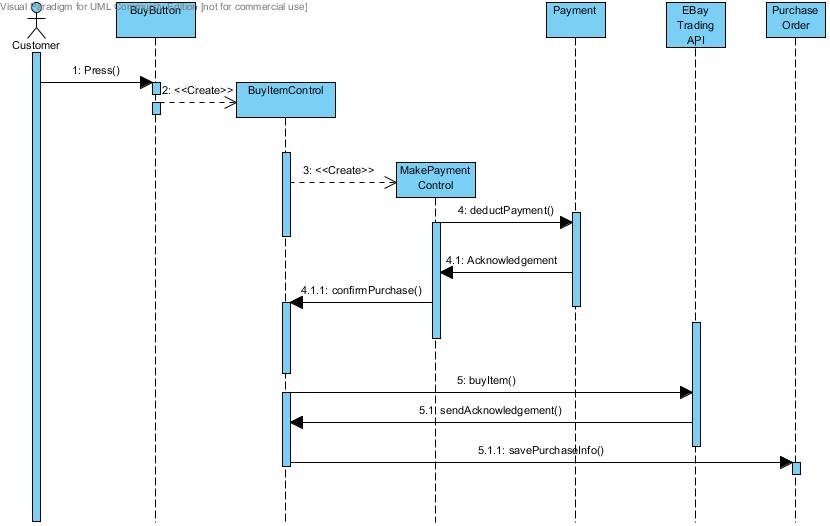
This Section aims to illustrate how each use case and objects identified in Section 4.1 behave with each other. Sequence diagrams are used to display these interactions.

Use Case: ViewSelectedItem (Please refer to Fig. 4.1)   
Objects: (Please refer to Table 4.1)

****

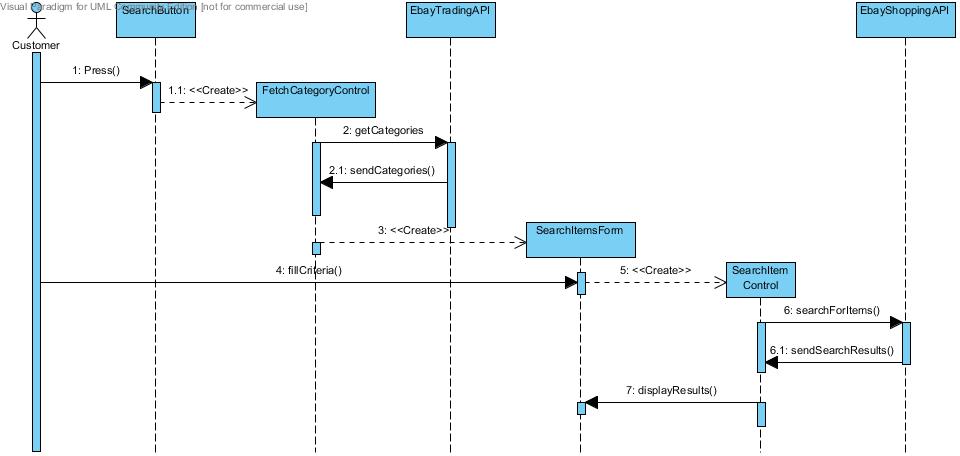
**Fig 4.12 – Interaction of objects in the ViewSelectedItem use case.**

Use Case: BuyItem (Please refer to Fig. 4.2)   
Objects: (Please refer to Table 4.2)

****

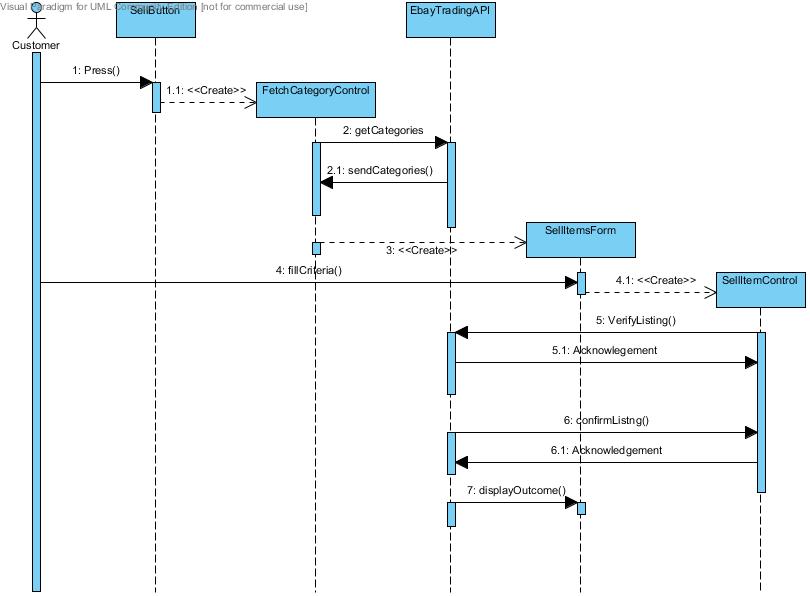
**Fig 4.13 – Interaction of objects in the BuyItem use case.**

Use Case: SearchForItems (Please refer to Fig. 4.3)   
Objects: (Please refer to Table 4.3)

****

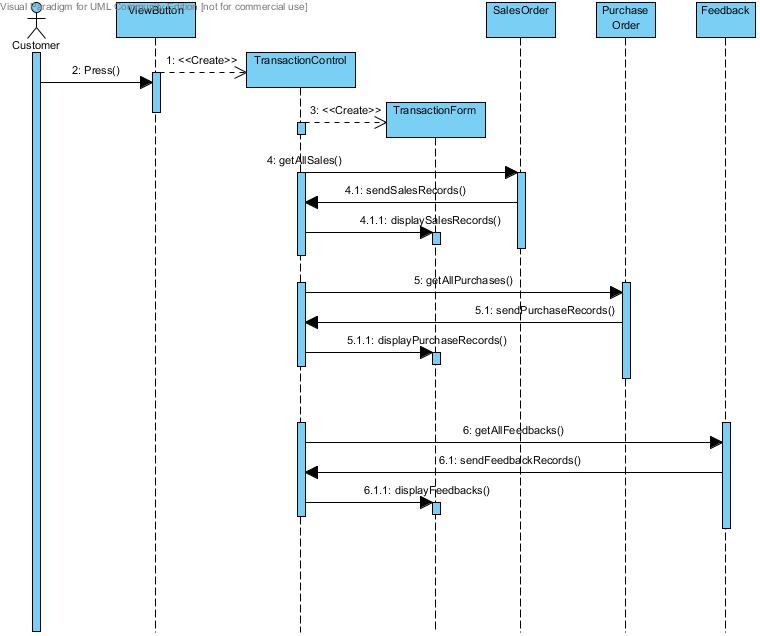
**Fig 4.14 – Interaction of objects in the SearchForItems use case.**

Use Case: SellItem (Please refer to Fig. 4.4)   
Objects: (Please refer to Table 4.4)

****

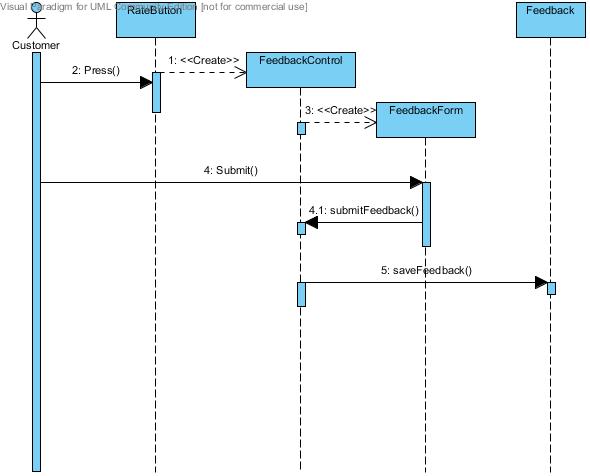
**Fig 4.15 – Interaction of objects in the SellItem use case.**

Use Case: ViewUserTransactions (Please refer to Fig. 4.5)   
Objects: (Please refer to Table 4.5)

****

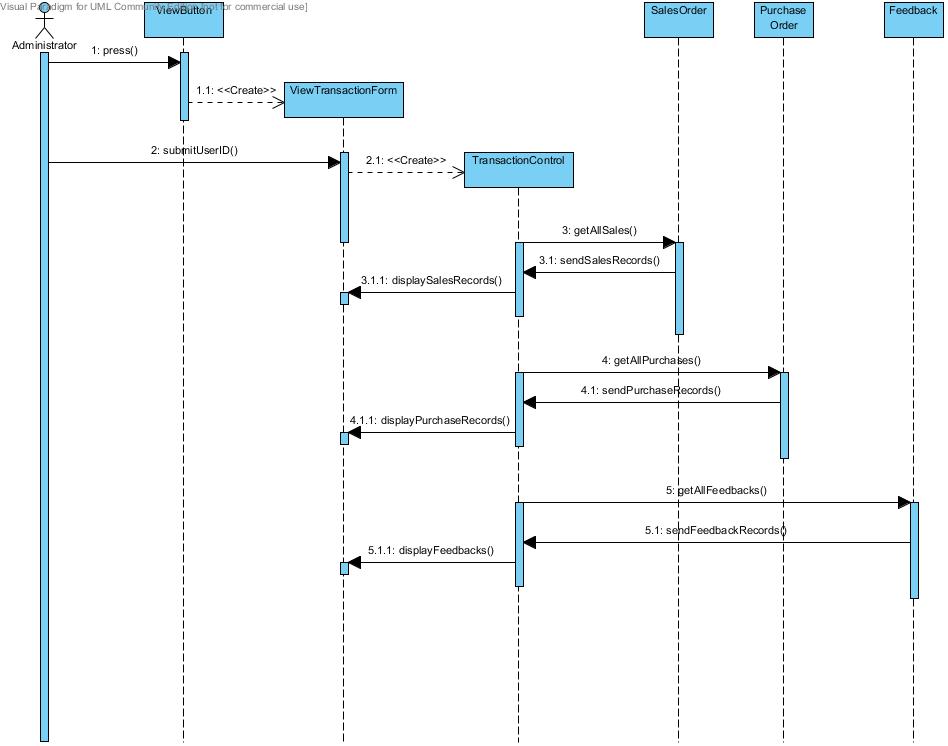
**Fig 4.16 – Interaction of objects in the ViewUserTransactions use case.**

Use Case: RateBoughtItem (Please refer to Fig. 4.6)   
Objects: (Please refer to Table 4.6)

****

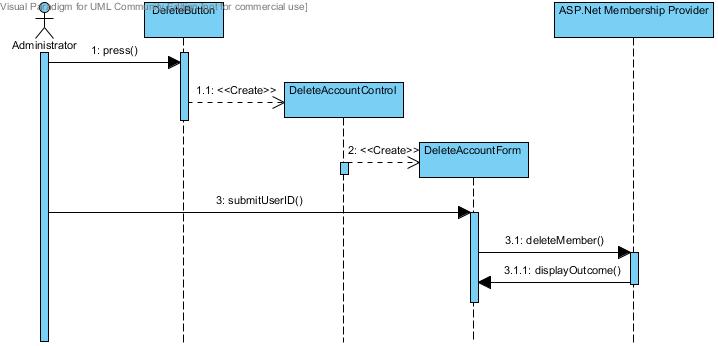
**Fig 4.17 – Interaction of objects in the RateBoughtItem use case.**

Use Case: ViewUserTransactions (Please refer to Fig. 4.7)   
Objects: (Please refer to Table 4.7)

****

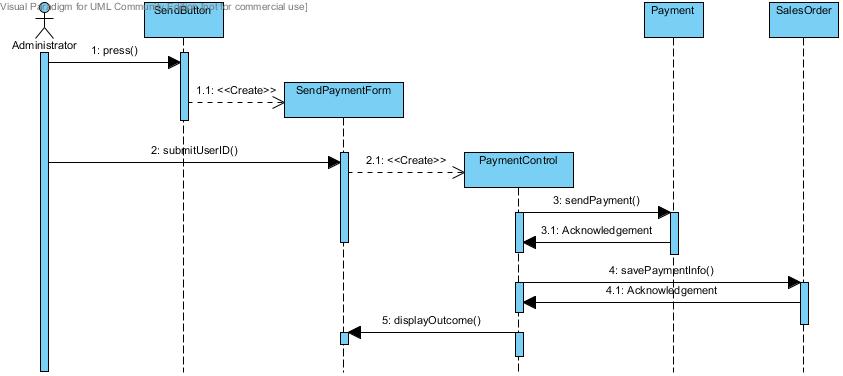
**Fig 4.18 – Interaction of objects in the ViewUserTransactions use case**

Use Case: DeleteUserAccount (Please refer to Fig. 4.8)   
Objects: (Please refer to Table 4.8)

****

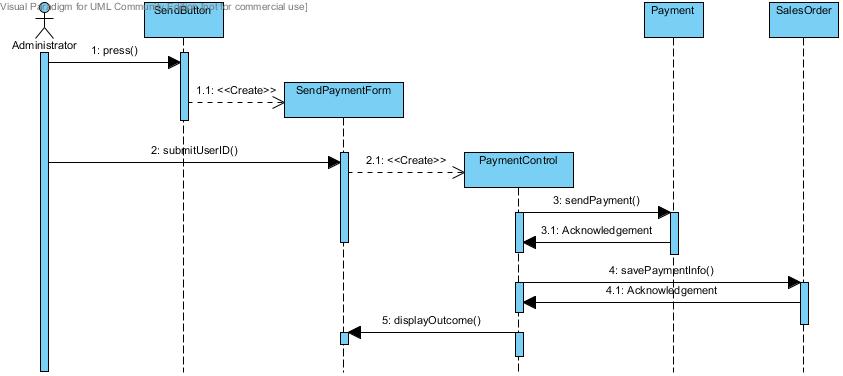
**Fig 4.19 – Interaction of objects in the DeleteUserAccount use case**

Use Case: SendPaymentToSeller (Please refer to Fig. 4.9)   
Objects: (Please refer to Table 4.9)

****

**Fig 4.20 – Interaction of objects in the SendPaymentToSeller use case**

Use Case: SendMessageToUsers (Please refer to Fig. 4.10)   
Objects: (Please refer to Table 4.10)

****

**Fig 4.20 – Interaction of objects in the SendPaymentToSeller use case**

## Identifying Attributes for Classes

Attributes are the properties of individual objects. For this section, only attributes that are relevant to the implementation of the system will be discussed. As such, EBay API class and their attributes and ASP.net Membership Provider classes and their attributes will not be discussed.

**Table 4.10 – Attributes Definitions of Identified Classes**

|  |  |  |  |
| --- | --- | --- | --- |
| **Entity Object** | **Attributes** | **Type** | **Definition** |
| Customer | firstName | String | The first name of the customer |
| lastName | String | The last name of the customer |
| ID | integer | The unique user id used by the system to identify the user. |
| gender | char | implied |
| dob | date | The date of birth of user. |
| address | String | Address of user used for delivering purchases. Also is the billing address |
| postal | numeric | Implied |
| city | String | Implied |
| country | String | Implied |
| tel | numeric | Implied |
| email | String | Implied. Also the login id used to authenticate user. |
| **Entity Object** | **Attributes** | **Type** | **Definition** |
| Item | itemID | numeric | 10 Digit item id that is used by EBay |
| itemTitle | String | The Item’s Description as used in EBay. |
| itemSubTitle | String | The Item’s secondary description as used in EBay. |
| itemURL | String | The URL of item’s image as used by EBay |
| itemCost | double | The immediate purchase price of an item from EBay. |
| itemShippingCost | double | The shipping cost to buyer’s location from EBay |
| itemDesc | String | Text information describing the item. |
| itemLocation | String | The location where the item is located at. |
| itemCondition | String | The item’s condition. |
| itemSpecifics | String | Specialised information based on the item’s category. E.g. DVDs will have region codes, while clothes will be label type. |
| **Entity Object** | **Attributes** | **Type** | **Definition** |
| Feedback | feedbackDate | Date | The date a feedback for done. |
| feedbackDesc | String | A description of the overall experience for an item. |
| feedbackType | integer | The type of feedback done. Positive, Negative or Neutral. |
| costFeeback | String | The satisfaction level for the cost of a purchase. |
| qualityFeedback | String | The satisfaction level for the quality of a purchase in respect to the description made by seller. |
| shippingFeedback | String | The satisfaction level for the shipping and delivery of a purchase. |
| **Entity Object** | **Attributes** | **Type** | **Definition** |
| SalesOrder | orderID | integer | The ID that uniquely identifies this sales order. |
| orderDate | Date | The date the order was generated |
| orderStatus | String | The current status of the order. |
| itemID | numeric | The item ID of the item that was processed by this order. |
| paymentReceived | Boolean | The payment for the item once it has been sold in EBay has been received from buyer. |
| paymentToSeller | Boolean | The payment for the item to Seller of this item in the system. |
| **Entity Object** | **Attributes** | **Type** | **Definition** |
| PurchaseOrder | orderID | Date | The ID that uniquely identifies this sales order. |
| orderDate | String | The date the order was generated |
| orderStatus | integer | The current status of the order. |
| itemID | numeric | The item ID of the item that was processed by this order. |
| **Entity Object** | **Attributes** | **Type** | **Definition** |
| Payment | amount | double | The amount used in the transaction |
| itemID | numeric | The item ID of the item that was processed by this order. |
| paymentAmt | double | The payment for the item to be deposited to the account of the Seller of this item in the system. |

## User Interface Design

Since this project is dealing with e-commerce on the web which has the potential to attract users from all walks of life, the graphical user interface has to be simple, precise and attractive. This section aims to highlight the prototype GUI that has been developed for this very purpose. We have based our user interface design upon the User Interface Design Heuristics by Jacok Nielson at (http://www.useit.com/papers/heuristic/heuristic\_list.html)

The following aims to highlight how some of the major principals of UI design have been incorporated into the design.

|  |  |
| --- | --- |
|  |  |

**Fig 4.21 – Displaying the visibility of system status**

**Clear Visibility of System Status**  
System should always inform the user of whatever is happening with appropriate feedback messages.

|  |
| --- |
|  |

**Fig 4.22 – Display of information in simple terms**

**Effective Use of Communication**  
The system should speak the user’s language. That would mean using simple and concise language and avoiding jargons.

|  |
| --- |
|  |

**Fig 4.22 – A registration page with errors**

**Aesthetic and minimalist design**  
Dialogues should not contain information which is irrelevant or rarely needed. System should only present information that is relevant to the present situation

|  |
| --- |
|  |

**Fig 4.22 – Illustration of reviewing an Item before Buying an Item**

**User Control and Freedom**  
The user should be given the freedom to have control of his/her actions. System should always present information to a user and ask for confirmation before committing the transaction.

# Database Design

Crow’s foot notation has been used to draw the ER diagram. It provides easy understanding and clearly identifies relationships between entities.

## Entity Relationship Diagram

**Fig 5.1 – ERD of eACE systen**

**EBAY\_CHILD\_CAT**

feedbackDesc  
itemCost  
itemQuality  
itemShipping  
positiveExperience  
positiveExperiencePic  
dateRated

itemId

**ORDER\_FEEDBACK**

has

title  
subtitle  
itemDesc  
picURL  
listingPrice  
buyItNowPrice  
feesDeductible  
startTime  
endTime  
orderStatus  
salePrice

itemId

**CUST\_SELLING\_ORDER**

Email  
Comment  
Password  
IsApproved  
LastActivityDate  
LastLoginDate  
LastPasswordChangedDate  
CreationDate  
IsLockedOut

userId

**MY\_ASPNET\_MEMBERSHIP**

subscribes

auctions

title  
itemCost  
ShippingCost  
totalCost  
dateOrdered  
timeOrdered

itemId

**CUST\_PURCHASE\_ORDER**

fName  
lName  
gender  
dob  
address  
postal  
city  
country  
contact

customerId

**CUSTOMER**

generates

categoryName

has

categoryName

categoryId

**EBAY\_PARENT\_CAT**

version

versionId

**EBAY\_CATEGORY\_VER**

localValue

custID  
itemID

**SHOPPINGCART**

categoryId

## Entity Type and Description

**Table 5.1 – Attributes and description of Entities identified**

|  |  |  |  |
| --- | --- | --- | --- |
| Entity Name : **CUSTOMER** Description : Contains customer details | | | |
| **Attribute** | **Data Type** | **Description** | **Example** |
| customerId | Int(11) | Unique Customer identification ID created by ASP membership provider | 1, 235,1000 |
| fName | Varchar(18) | The first name of customer | John |
| lName | Varchar(18) | The last name of customer | White |
| gender | Char(1) | The gender of customer | M, F |
| dob | date | The date of birth of customer | 14-09-1983 |
| address | Varchar(50) | The billing and residential address of customer | Blk 152, #09-89, Bedok |
| postal | Decimel(8) | The Postal code | 234567 |
| city | Varchar(25) | City residing in | Singapore |
| country | Char(5) | The country | Singapore |
| contact | Decimel(12) | Contact number of customer | 98765342 |
| Entity Name: **CUST\_PURCHASE\_ORDER** Description: Contains the details of a sales transaction that the customer has made and paid for. | | | |
| **Attribute** | **Data Type** | **Description** | **Example** |
| itemId | Varchar(15) | The EBAY item identifier | Z25061445018 |
| title | Varchar(255) | The Item’s title | Brand New Iphone |
| itemCost | Double(10,2) | The Buy it Now Price of Item | 23.89 |
| shippingCost | Double(10,2) | The Shipping cost to customer’s address | 10.00 |
| totalCost | Double(10,2) | Shipping + Item Cost | 33.89 |
| dateOrdered | Date | Date of the transaction | 2009-12-27 |
| timeOrdered | time | Time of the transaction | 01:16:31 |
| Entity Name: **CUST\_SELLING\_ORDER** Description: Contains the details of an Item the customer wants to sell on EBay | | | |
| **Attribute** | **Data Type** | **Description** | **Example** |
| itemId | Varchar(15) | The EBAY item identifier | Z25061445018 |
| title | Varchar(255) | The Item’s title | Brand New Iphone |
| subtitle | Varchar(255) | The Item’s subtitle | 32gb black, brand new |
| itemDesc | text | A brief introduction about product and other information. | - |
| picURL | Varchat(151) | The URL of the item for putting up on eBay’s gallery. | http://www.jcu.edu.au/pic.gif |
| buyItNowPrice | Double(10,2) | The Buy it Now Price of Item | 23.89 |
| listingPrice | Double(10,2) | The starting list price for item. | 23.89 |
| salePrice | Double(10,2) | The price the item was sold for | 23.89 |
| feesDeductible | Double(10,2) | The administrative charges for listing this item. | 5.00 |
| orderStatus | Varchar(15) | The status of the item listed. | Completed, Pending |
| startTime | Date | The Date the auction is to start | 2009-12-27 |
| endTime | Date | The date the auction will expire | 2009-12-29 |
| Entity Name: **EBAY\_CATEGORY\_VER** Description: The current index of EBay’s categories. A change in this value indicated that ebay has updated its category indexing which happens every 6-8 months. | | | |
| **Attribute** | **Data Type** | **Description** | **Example** |
| id | Int(1) | Unique identifier | 1 |
| version | Int(5) | The index version for EBAY’s categories | 639 |
| Entity Name: **EBAY\_PARENT\_CAT** Description: Contains the categories details that eBay uses to catalog all items. | | | |
| **Attribute** | **Data Type** | **Description** | **Example** |
| categoryId | Varchar(8) | The unique category index supplied by EBay | 10542 |
| categoryName | Varchar(125) | The category name that is used to catalog items. | Real Estate, Coins, Toys |
| Entity Name: **EBAY\_CHILD\_CAT** Description: Contains the Leaf categories that EBay includes under the main parent categories. | | | |
| **Attribute** | **Data Type** | **Description** | **Example** |
| categoryId | Varchar(8) | The unique category index supplied by EBay | 10542 |
| categoryName | Varchar(125) | The category name that is used to catalog items. | Real Estate, Coins, Toys |
| Entity Name: **ORDER\_FEEDBACK** Description: Contains the feedback left by the customer for an item or transaction that he/she has completed successfully. | | | |
| **Attribute** | **Data Type** | **Description** | **Example** |
| itemId | Varchar(15) | The EBAY item identifier | Z25061445018 |
| feedbackDesc | Varchar(255) | User’s overall written feedback for transaction | Nothing Great About this Book, Just the normal thing u can find in the internet |
| itemQuality | Varchar(20) | The Item’s quality feedback | Highly Satisfied |
| itemCost | Varchar(20) | The Item’s Cost feedback | Highly Satisfied |
| itemShipping | Varchar(20) | The Shipping if Item feedback | Disappointing |
| positiveExperience | Varchar(20) | Rate Positive, Negative or Neutral | Neutral |
| positiveExperiencePic | Varchar(20) | The correspondinig picture for the experience | neutral.gif |
| dateRated | Date | Date the feedback was completed | 2009-12-27 |
| Entity Name: **MY\_ASPNET\_MEMBERSHIP** Description: ASP.Net table handled by .Net membership provider. A user’s login credentials are stored here. | | | |
| **Attribute** | **Data Type** | **Description** | **Example** |
| userId | Int(11) | Unique Customer identification | 1, 1000 |
| Email | Varchar(128) | User’s email address. Must be unique. Used as a username to login | xyz@my.jcu.edu.au |
| Comment | Varchar(255) | Administrator remarks on account should account be locked or disabled. | - |
| Password | Varchar(128) | User’s password to login to system. Stored encrypted, cannot be reversed. | 9hqYAWKluMMFEmKj64zbter2CG8= |
| IsApproved | Tinyint(1) | Once account has been activated by user, account is approved. Non approved accounts cannot be used. | 1,0 |
| LastActivityDate | date | The last date/time that user used system. | 2010-02-02 23:56:02 |
| LastLoginDate | date | The last date/time that user used system. | 2010-02-02 23:25:07 |
| LastPasswordChangedDate | date | The last date/time that user changed password. | 2009-12-27 02:08:34 |
| CreationDate | date | The date the account was created. | 2009-12-27 01:13:52 |
| IsLockedOut | Tinyint(1) | Account can be locked out if password is wrong continuously for 10 times. | 1,0 |
| Entity Name: **SHOPPINGCART** Description: Contains the item information in a users shopping cart | | | |
| **Attribute** | **Data Type** | **Description** | **Example** |
| custID | Int(11) | Unique Customer identification | 1, 1000 |
| itemId | Varchar(15) | The EBAY item identifier | Z25061445018 |
| localValue | Int(5) | The id of the eBay site the item belongs | 0, 216 |

## Relationship Description

|  |  |
| --- | --- |
| **CUSTOMER** | **CUST\_SELLING\_ORDER** |
| Mandatory | Optional |

A CUSTOMER auctions zero, one or more CUSTOMER\_SELLING\_ORDER

A CUSTOMER\_SELLING\_ORDER belongs to one CUSTOMER

|  |  |
| --- | --- |
| **CUSTOMER** | **CUST\_PURCHASE\_ORDER** |
| Mandatory | Optional |

A CUSTOMER generates zero, one or more CUST\_PURCHASE\_ORDER

A CUST\_PURCHASE\_ORDER belongs to one CUSTOMER

|  |  |
| --- | --- |
| **CUSTOMER** | **MY\_ASPNET\_MEMBERSHIP** |
| Mandatory | MANDATORY |

A CUSTOMER subscribes to one MY\_ASPNET\_MEMBERSHIP

A MY\_ASPNET\_MEMBERSHIP belongs to one CUSTOMER

|  |  |
| --- | --- |
| **CUST\_PURCHASE\_ORDER** | **ORDER\_FEEDBACK** |
| Mandatory | Optional |

A CUST\_PURCHASE\_ORDER has zero or one ORDER\_FEEDBACK

An ORDER\_FEEDBACK belongs to one CUST\_PURCHASE\_ORDER

|  |  |
| --- | --- |
| **EBAY\_PARENT\_CAT** | **EBAY\_CHILD\_CAT** |
| Mandatory | Optional |

An EBAY\_PARENT\_CAT has zero, one or more EBAY\_CHILD\_CAT

An EBAY\_CHILD\_CAT belongs to one EBAY\_PARENT\_CAT

## Assumptions and Constraints

The following are the assumptions and constraints identified for the database design.

Constraints

C01. Password cannot be mailed to the user after the initial registration email. User has to reset password.

C02. Each item the customer wishes to sell is listed on EBay for a minimum of 7 Days.

C03. Shopping Cart is cleared after each logout.

Assumption

A01. Each unique email address is considered as a unique user account, even if all other information is similar.

## Functional Dependencies and Normalization

**CUSTOMER**

customerId -> fname

customerId -> lname

customerId -> gender

customerId -> dob

customerId -> address

customerId -> postal

customerId -> city

customerId -> country

customerId -> contact

**CUST\_SELLING\_ORDER**  
itemId -> title  
itemId -> subtitle  
itemId -> itemDesc  
itemId -> buyItNowPrice  
itemId -> listingPrice  
itemId -> salePrice  
itemId -> feesDeductible  
itemId -> orderStatus  
itemId -> startTime  
itemId -> endTime  
itemId -> customerId

**CUST\_PURCHASE\_ORDER**  
itemId -> title  
itemId -> itemCost  
itemId -> shippingCost  
itemId -> totalCost  
itemId -> dateOrdered  
itemId -> timeOrdered  
itemId -> customerId

**ORDER\_FEEDBACK**  
itemId -> feedbackDesc  
itemId -> itemQuality  
itemId -> itemCost  
itemId -> itemShipping  
itemId -> positiveExperience  
itemId -> positiveExperiencePic  
itemId -> dateRated

**EBAY\_CAT\_VER**id -> ver

**EBAY\_PARENT\_CAT**categoryId -> categoryName

**EBAY\_CHILD\_CAT**categoryId -> categoryName  
categoryId -> parentCat

**SHOPPINGCART**itemID,custID -> localValue

**MY\_ASPNET\_MEMBERSHIP**

userId -> Email

userId -> Comment

userId -> Password  
userId -> IsApproved  
userId -> LastActivityDate  
userId -> LastLoginDate  
userId -> LastPasswordChangedDate  
userId -> CreationDate  
userId -> IsLockedOut

## Relational Headings

The primary keys and foreign keys are highlighted by underline and italic respectively.

* **CUSTOMER** (*customerId*, fName, lName, gender, dob, address, postal, city, country, contact)
* **CUST\_PURCHASE\_ORDER** (itemId, title, itemCost, shippingCost, totalCost, dateOrdered, timeOrdered, *customerId*)
* **CUST\_SELLING\_ORDER** (itemId, title, subtitle, itemDesc, picURL, buyItNowPrice, listingPrice, salePrice, feesDeductible, orderStatus, startTime, endTime, *customerId*)
* **EBAY\_CATEGORY\_VER** (id,ver)
* **EBAY\_PARENT\_CAT**(categoryId, categoryName)
* **EBAY\_CHILD\_CAT**(categoryId, categoryName, *parentCat*)
* **SHOPPINGCART**(custID, itemID, localValue)
* **ORDER\_FEEDBACK** (*itemId*, feedbackDesc, itemQuality, itemCost, itemShipping, positiveExperience, positiveExperiencePic, dateRated)
* **MY\_ASPNET\_MEMBERSHIP** (*userId*, Email, Comment, Password, IsApproved, LastActivityDate, LastLoginDate, LastPasswordChangedDate, CreationDate, IsLockedOut)

# Prototyping

The prototype will be based on the ASP.NET technology and will be written in VB.Net to facilitate rapid application development that would allow us to complete this project in 12 weeks. The prototype will evaluated with the user and refinements will be made during the full implementation process in ICT 2.

The prototype will also be demonstrated to the supervising and fellow project teams. Prototype will allow user registration, search for items on eBay (Dynamic) and buying items off the eBay sandbox environment which we will simulate.

# Conclusion

## Summary

An average of 200 hours of research and work has been spent on this project. We have strived to follow all the best practices in analysis, planning, design and implementation. We have attempted to document all steps and procedures as part of this report.

## Further Developments

ICT 2 that is coming in the next semester will be a continuation of project and we will be working on building the complete solution, developing test cases and performing testing and bug fixes.

## Challenges Encountered

The learning process to use eBay API with the .NET framework has been steep as the documentation off eBay has not been extensive. But due to the development of the prototype it would be easier to develop the solution in the coming semester.

EBay is also depreciating certain classes of the Shopping API and this we have used the Finding API in conjunction with the Shopping API to future proof the solution.

# References

1. *Ebay.com. 1995-2012.* Main page*. [Online] (Updated 19th Dec 2012) Available at:* [*http://www.ebay.com*](http://www.ebay.com) *[Accessed 19th Dec 2012].*
2. *Ebay.com. 1995-2012.* About Us – EBay Developers Program. *[Online] (Updated Dec 2012) Available at:* [*http://developer.ebay.com/businessbenefits/aboutus/*](http://developer.ebay.com/businessbenefits/aboutus/) *[Accessed 21st Dec 2012].*
3. *Bernd Bruegge and Allen H.Dutoit, 5th October 2003, Object Oriented Software Engineering, Using UML, Patterns and Java – Second Edition, Pearson International Edition*
4. *Thomas M.Connolly and Carolyn E.Begg, 1st August 2001, Database Systems: A Practical Approach to Design, Implementation, and Management – Third Edition, Addison Wesley*
5. *Directriz, 2009. Requirements Gathering Techniques. [Online] Directriz. Available at:*[*http://epf.eclipse.org/wikis/openupsp/openup\_basic/guidances/guidelines/req\_gathering\_techniques,\_OnoNQNSAEdmLhZ9H5Plxyw.html*](http://epf.eclipse.org/wikis/openupsp/openup_basic/guidances/guidelines/req_gathering_techniques,_OnoNQNSAEdmLhZ9H5Plxyw.html) *[Accessed 5th Dec 2012].*
6. Language and Learning Online, *2007.* *Writing Literature Reviews. [Online]* Literature Review Guide*, Gail Craswell, ANU. Available at:* [*http://www.monash.edu.au/lls/llonline/writing/general/lit-reviews/index.xml*](http://www.monash.edu.au/lls/llonline/writing/general/lit-reviews/index.xml) *[Accessed 15th Dec 2012].*
7. *Ray Rischpater, 12th March 2004 , eBay Application Development – Second Edition, Apress Publishing*
8. *Widerman, M., 2008. The Project Life Cycle*. *[Online] Max Widerman grp. Available at:*[*http://www.maxwideman.com/papers/life\_cycle/life\_cycle.pdf*](http://www.maxwideman.com/papers/life_cycle/life_cycle.pdf) *[Accessed 21st Oct 2012].*
9. *IITL, 12th August 2007, IT Project Management Fundamentals – First Edition, IITL*
10. *Half.com. 1995-2010. Main Page. [Online] (Updated Nov 2012) Available at:* [*http://www.half.com*](http://www.half.com) *[Accessed 23rd Nov 2012].*
11. *Ebay.com. 1995-2010. EBay Sandbox – EBay Developers Program. [Online] (Updated Dec 2012) Available at:* [*http://developer.ebay.com/support/tools/sandbox/*](http://developer.ebay.com/support/tools/sandbox/) *[Accessed 27th Dec 2012].*
12. *Jacok Nielson. 2005. User Interface Design Heuristics. [Online] Useit Grp. Available at:* [*http://www.useit.com/papers/heuristic/heuristic\_list.html*](http://www.useit.com/papers/heuristic/heuristic_list.html) *[Accessed 4th Dec 2012].*
13. *Ebay.com. 1995-2010. GetNotificationsUsage – Call Reference – EBay Trading API. [Online] (Dec 2012) Available at:* [*http://developer.ebay.com/DevZone/XML/docs/Reference/eBay/GetNotificationsUsge.html*](http://developer.ebay.com/DevZone/XML/docs/Reference/eBay/GetNotificationsUsge.html) *[Accessed 27th Nov 2010].*

THE END