Final report
developing an online market place
20th of June 2008

Technical University of Delft, EWI faculty
Computer Science, BSc-project, IN3405

Authors
Frank Groeneveld, 1262998
Martijn Reijerse, 1263137

Contractors
O&S CICAT / TU-Delft - Ir. E.M.A. Geers
University Computing Centre / University of Dar Es Salaam - A. Nkelame

Supervisor TU-Delft
Ir. B. R. Sodoyer

Version 4, awaiting feedback
Preface

Martijn Reijerse and Frank Groeneveld have done their Bsc-project at the University Computing Centre in Dar Es Salaam, Tanzania. This internship was part of their bachelor Computer Science at the Technical University of Delft, faculty EWI. During this internship they had to build an online market place that can be used by the students of the University of Dar Es Salaam.
Table of contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summary</td>
<td>4</td>
</tr>
<tr>
<td>Problem and analysis</td>
<td>5</td>
</tr>
<tr>
<td>Design</td>
<td>7</td>
</tr>
<tr>
<td>Implementation</td>
<td>9</td>
</tr>
<tr>
<td>Conclusions</td>
<td>10</td>
</tr>
<tr>
<td>Advises</td>
<td>11</td>
</tr>
<tr>
<td>Evaluation</td>
<td>12</td>
</tr>
<tr>
<td>Literature</td>
<td>14</td>
</tr>
<tr>
<td>Appendixes</td>
<td>15</td>
</tr>
<tr>
<td>Appendix A: Requirements document</td>
<td>15</td>
</tr>
<tr>
<td>Appendix B: Architectural design</td>
<td>50</td>
</tr>
<tr>
<td>Appendix C: Technical design</td>
<td>54</td>
</tr>
</tbody>
</table>
Summary

This document describes what problems the developers of the online marketplace have encountered and how they solved it. Furthermore it describes design decisions that were made, details about the implementation and some concluding remarks. In the last few chapters the developers give some concluding advises as well as an evaluation of the internship they did. The original requirements document, architectural design document and technical design document are added as appendices.
Problem and analysis

For students in Tanzania it is hard to buy appropriate study materials. Study books are even harder to get. There was no easily accessible place for students to sell or buy any secondhand study materials like books or computers. This project was started for this reason. The goal of this project is to create an online trading place where students can trade their study materials, so that it will be easier for students to buy study materials. It will also be easier to sell used study materials, which will increase the availability of study materials.

There are a number of problems regarding the design of this online marketplace. One problem is the internet speed in Tanzania. Therefore a solution has to be implemented to reduce loading times, so that everyone is able to view the website.

Another problem is the payment methods in Tanzania. It is very hard for a Tanzanian to get a credit card. Therefore it is a bad idea to implement an Ebay based payment method, because only then a small percentage of the users will be able to actually buy items on this marketplace.

For buyers it is important that they pay an appropriate price for the items they want to buy. For sellers it is important to receive a good price as well. Therefore a function is needed to test the market for an appropriate value for the items.

The content that will be placed on this marketplace should be regulated somehow, because this internet website is accessible for every student and it is almost impossible to identify a user.

Users should be able to store items in a wish list, because this allows the user to inspect the item at another time, without searching for it again.

All the functions described above will work only if there are enough users that will make use of marketplace. Therefore the university will have to advertise for this new marketplace. Currently the university has about 18.000 students. The first version of the online marketplace will be hosted locally somewhere on the university and will be targeted at roughly half of all students, meaning 9.000 students. The students can access it from anywhere on the university. Locally hosting also means there is enough bandwidth available, because the wireless networks of the university are 11Mbps or faster and the wired networks are 100Mbps and some are even 1 Gbps. Furthermore the costs of hosting the site will not be very high if it is hosted locally, because local traffic is a lot cheaper.
The marketplace is entirely built on the idea of being free for everybody. This means all the costs will need to be earned back by using advertisements. Every advertiser has a budget, that is automatically lowered when somebody views one of their advertisements. Furthermore, the budget can be lowered for every day an advertisement is on the website. At first, the website will not have any advertisers. After the number of users has grown to at least 1,000 students, it will be easier to get advertisers. The University Computing Centre hopes this kind of funding will be enough to keep the website profitable.
Design

The online marketplace was developed using an iterative and incremental development model. This means a basic design of the system was created first after acquiring the requirements. During the implementation a few prototypes and concepts were created. These prototypes and concepts were discussed with the project supervisor (Mr. A. Nkelame) and some software developers of the UCC. If any problems arose, the design was adjusted and the implementation was altered. The big advantage of this model was, that problems could be solved in an early stage, instead of at the end of the development process.

The marketplace will be used for many years. In these years it has to be modified according to the needs of the users. Extending marketplace is only possible if this project is easy to modify and maintain.

A local team of engineers has to modify and maintain the marketplace, because the current developers will leave the country when the initial version is finished. The project has to be very well documented, otherwise the local team of engineers will not be able to maintain the website.

The online marketplace is built using the model-view-controller design pattern to make the design very clean. Accessing the database is done in a separate subsystem (the models), creating the html output is done in a separate subsystem (the view) and everything is combined using a separate subsystem (the controller). This will make the system very easy to modify and maintain.

As described above not everybody in Tanzania has access to a fast internet connection. Therefore a lightweight version of marketplace has to be available for people with slow connection speeds. A text-based version will be available to reduce loading times to a minimum.

No payment system will be implemented in the initial version, it is better idea to adopt the Marktplaats.nl payment method. Marktplaats.nl allows buyers and sellers to communicate with each other. The buyer and seller can then meet each other and exchange the product and the money.

The website should allow sellers to easily find out what price he could get for an item. The buyer should also be able to see the appropriate price for an item. This will be done by giving the users the opportunity to place a bid on an item. A new bid has to be higher than the previous bids.
There are two options to consider to control the content that will be placed on the website. One option is to create a team of moderators that will inspect every new item for inappropriate content on the marketplace, but this is a very expensive option to implement. A better option is to give every user the possibility to mark an item as inappropriate. These items will be reported to the moderators who can easily check if this item is indeed inappropriate. This option requires a smaller number of moderators.

To serve the visitors with a function to store their favorite items every user can register and create their own wish list. This wish list will contain all the stored items with their highest bids.

Another option to advertise for this new marketplace is to let the users tell their friends about this website. This will be done by a function called ‘tell a friend’. A user can select an item and send the link of the page to a friend. This friend will receive an email that contains information about the item and a link to it.

Items will be removed when the sellers or an administrator removes the item. In both cases the item is totally removed. There will be no information left in the database about the removed item. Items that are older than one year will automatically be removed from the database.

Users that are registered cannot remove themselves. Only an administrator is able to remove the user. In this case the user is totally removed from the system.

Unregistered users that are selling items will be saved in the user database.

All information about users and user actions has to be handled in a safe way. This information cannot be sold to a third-party company and can only be used for marketplace purposes.

An IP-address will be stored by all the actions a user can make. This is done for controlling purposes.
Implementation

The complete system will be implemented in PHP5 with MySQL as database back end. This language was chosen because the University Computing Centre already has knowledge about PHP, version 4 in particular, so they know how to modify and update the system after its deployment.

The marketplace is built using Zend Framework, an open source development framework written in PHP5. This framework allows the developers to create a website using the model-view-controller design principle in an easy way. It provides the programmer with an easy way to access MySQL databases and build things like user authentication and authorization. The developers chose this framework, because it's very well tested and support by a lot of big companies [1], including IBM, Google and Microsoft.

The Computing Centre didn't have much knowledge about PHP5. So the developers had to give a workshop about it. They explained the most significant new features and gave a short introduction to Object Oriented programming.

The work that had to be done to create this project was done by two people, Martijn Reijerse and Frank Groeneveld. Martijn did most of the work that had anything to do with the graphical representation of the website and Frank did most of the work that had anything to do with the database. The rest of the code was written by both of them, it was the “glue” between those two subsystems. When looking at the model-view-controller design pattern, you could say Martijn made most of the views and Frank made most of the Models. Together they created the controllers.

The developers organized a feedback session every week. In this meeting the developers showed the current status of the project and let the client try concept versions. These meetings were very useful, as they allowed the developers to adapt the system in a very early stage.

During the development of the project the developers thoroughly tested everything on their own machines. However, because the UCC couldn't prepare their server in time it could not be tested by more people so test plan wasn't written.
Conclusions

Looking at the result of the project, the new marketplace, we can be satisfied with the result. This project is a solution for the long existence problem for students in Tanzania. For them it is hard to get study materials. Our project will help these students to sell and buy their materials.

We are also satisfied because we developed a full-working version. The first thought was to make a prototype of the marketplace, but after a few weeks we agreed to make a full-working marketplace. Now this goal has been reached and a full-working version has been made.

Another requirement was that this project should be easy to maintain. We think that the project is now very easy to maintain, because of the well structured design, the provided design documents and the user and administrator manuals. The structure of this project is very clear, we have made an detailed design of the marketplace and there is a lot of online documentation about the Zend Framework. Furthermore some maintenance has been automated: old items are removed automatically and advertisers receive an email when they run out of credits.

Before we leave the UCC we will hand in all our design documents, all the source code and the user/administrator manual. Furthermore we hope to stay in touch with the UCC and help them out with any problems that might arise.

Of course the success of the new marketplace doesn't depend only on our efforts. Our work has to be continued by Tanzanian people, but we think that we made this job as easy as possible and we see a bright future for the marketplace.
Advises

The main goal of the new Marketplace is to serve buyers and sellers as good as possible. To guarantee that their trading experience is as best as possible a few things needs to be taken care of.

First of all it is important that the items that are on the website are real items, not some fake items from somebody that wanted to be funny. There is a function for registered user to notify administrators of inappropriate behavior on the website, but these function won't work if there is no administrator that looks at this notifications. Therefore a small team of administrators has to put together to control the content that will be placed on the website. They have to take care of the notifications that were made by the users. These administrators can only do their job properly when there are guidelines and rules. These guidelines and rules have to describe about what is appropriate and what is not. Actually, the guidelines and rules have to be ready before the website goes live.

To maintain the structure of the new marketplace the categories have to cover all different kinds of items accordingly. If the category 'other' has more than 20 items that share the same category that doesn't exist, a new category has to be added. This way the items are easier to find on the marketplace. Another rule has to be developed to control the content of the other categories. If a category contains to many items (for example more than 500), this category has to be split into two new categories to maintain the structure of the marketplace.

Buyers and sellers will only put their items online when the website is easy accessible at any time. This means that there has to be a team that is responsible for the maintenance to the marketplace-server. They will make sure that whenever a problem arises that this problem will be solves as quick as possible. They are also responsible to upgrade the PHP-version and MySQL-version when newer versions become available. They are also responsible to install a extra webserver when there are to many visitors on the website to be handled by one webserver.

It is important that the extension of the marketplace is going simultaneously with the number of visitors of the website. For example if there are to many empty categories the website will look unused. On the other hand if the categories contain to many items it will be hard to find the needed items. Therefore the number of visitors, items and categories has to be examined very well. If changes are needed, these changes have to be applied carefully because visitors don't like abrupt changes in a product that they loved the way it was.
Evaluation

Before we started with this project we made a strict planning of this project. Our planning was as follows:

- Week 1: Gathering information and concept of requirements document
- Week 2: Requirements documents, architectural and technical design documents
- Week 3: Test plan
- Week 4 to 9: Implementation of the project
- Week 10 to 11: Testing
- Week 12: Finish project

As we can say now, at the end of the project, we followed the planning until week 9 quite well. After week 2 all the design documents were ready. We replaced the test plan with the implementation and put the thorough testing at the end of our project. At the end of week 9 the implementation was done. During the implementation phase, we already tested most of the functions. Until this point we were on schedule. Unfortunately we had to wait with the real testing, especially testing of all the emails that are automatically sent, because of some problems with the web server that will eventually host the project. This web server will run the website so that it would be accessible for every UDSM students. Until the moment of writing this document we are still waiting for the server to become ready to run the project, so no real testing is done so far. We do not expect that the server will become ready very soon, therefore the real testing phase has to be done by the staff of the UCC.

If we look back at our project we can conclude that the planning worked out quite well. This is due to the fact that we were able to work on our own and that we did not depend on other people. We could work on our own tempo and we didn’t have to wait for other people. But in the final phase of our project the planning failed. This is due to the fact that we depended on other people in that phase.

We learned a lot of things when we worked on this project. Not only did we learn new technical things about Zend Framework, CSS and PHP, but we also learned a lot of things about the differences between cultures of the world. Working for a company in Tanzania is different than working for a company in The Netherlands. The automatization of Tanzania is on a different level than that in The Netherlands. Most of the software that is developed in Tanzania is used to reduce the amount of paperwork, while in The Netherlands software is often used for more advanced problems. Besides the differences in the type of software there is also a difference in the speed of working. In Tanzania the people work longer but produce less work. Another aspect that confirms the difference between these two country is the fact the in The Netherlands schedules are followed as strictly as possible, while in
Tanzania there is no such thing as a strict planning. Even when there is a planning this planning will be adjusted easily when needed.

There are a few employees at the UCC that do make their deadlines and produce much work in a small amount of time. We think the UCC really dependents on these employees.

We also learned a new way of web programming with the Zend Framework. Before we started with this project we already had a lot of experience with developing websites, because we run our own web development company, but we never made a real website with such a framework. With this framework a lot of work has already been done by the Zend company. So we only had to implement application specific things. We think it is really a benefit to use the Zend Framework, because to our opinion it is useless to invent the wheel every time when we need to make a new website.
Literature

Appendixes

Appendix A: Requirements document

Introduction

When developing an application every detail of it has to be specified beforehand. This document does exactly that; it specifies all the features and requirements of the online trading website that is going to be developed.

Current situation

Currently a few websites exist to sell and buy items. The aspects of the different websites will be described in this chapter.

**UCC Bazaar**

The problem with the UCC Bazaar trading website is that it is only accessible for users that have access to the intranet of the UCC. This place is called ‘bazaar’ and isn't used by many people.

When placing an item on this website, the user has to specify a few specification of his or her item:

1. Name*
2. Description*
3. Photo
4. Price*
5. Contact Person*
6. Contact Email*
7. Mobile Phone

Items marked with a * are required to fill in.

There are no categories to order the items. When an user opens the UCC Bazaar he or she will see all the items available.

**Virtual Bazaar**

Another website that is available for trading items in Tanzania is [www.virtualbazaar.co.tz](http://www.virtualbazaar.co.tz). Only companies can put their items on this website, therefore this website isn’t accessible for everyone.
Proposed solution

Overview
The proposed solution is a website which can be accessed by anybody. It’s possible to browse through the website and see what everybody is selling and contact the sellers or place a bid on a product. Furthermore every user can add items to sell. All this can be done without signing up, although signing up offers more features and allows you to add items faster. Selling an item is done by filling in some forms with information about the product and choosing a category in which the product has to be displayed. Advertisements will be shown when browsing the website. The advertisements will be adopted to the current page that is being viewed. For example, when viewing a page with study books, the advertisements can be about courses on the UDSM.

Functional requirements
Actors that will use this system:
• Visitors
• Advertisers
• Administrators

All functional requirements are listed with a code behind it. This code start with a V for visitors, with an R for advertisers and with an A for administrators.

Visitors of the website
Visitors are the persons who will sell or buy items on the website. Visitor are able to do the following:

• Search for items Vs01
  Visitor can search items with the following specifications:
  1. Title
  2. Description
  3. Price range
  4. Keywords matching any specification
  Besides these specifications a visitor can also search on specific specifications when a category is selected. For example, when the selected category is ‘Books' the visitor is able to search on ISBN, author, publisher etc.
  For an overview of all specifications see appendix A.

• Browse items Vb01
  Visitor can browse items by specifying categories.
• **View item details Vv01**
When opening an item all specifications will be shown of this item. Also the bids for this item will be shown.
Detailed information about a seller will also be shown when an item is viewed. This information includes: name, region, rating.

• **Print an item Vp01**
When viewing an item there is an option to print this item properly.

• **Place a bid Vp02**
When viewing an item there is a possibility to place a bid on this item. This bid has to be higher than the last placed bid. The visitor has to specify his name and email when placing a bid, unless he is currently logged in.

• **Remove a bid Vr01**
When the user has placed a bid, he should also be able to remove it. How this could be done, depends on whether he is a registered user or not.

• **Send link to a friend Vs02**
On the search result page and on the item detail page a “send to friend” link should be displayed, which allows you to send the current link to a friend by email.

• **Register Vr02**
Visitors are able to register to create their own place on marketplace. After registering the visitor is able to create a wish list.
The following specifications are needed to register:
1. First name
2. Surname
3. Email
This email address has to be confirmed before the registering process is complete.
The following specifications are optional:
1. Telephone number
2. Address
3. City
During the registering process of the visitor a time stamp of the creation date will be saved. The IP-address of the user will also be saved.

• **Login Vl01**
Visitors can view their wish list or view items that are placed by their own only after logging in to the website. Registering is required for logging in.

- **Add item to wish list Va01**
  When viewing an item there is an option to add this item to the visitors wish list. Registering is required for this feature. If the visitor is already logged in, he or she can add this item directly to the wish list. If the visitor is not logged in he or she has to log in first.

- **Add item Va02**
  Visitors can put their items on the website. Registering is not required. When putting an item online the following information needs to be provided:
  1. First name
  2. Surname
  3. Email
  4. Title
  5. Description
  6. Price
  7. Other specifications of the item depend on the category
  If the visitor is already logged in, specifications 1 to 3 are not relevant and therefore these specifications are filled in automatically.
  During the saving process of the item a time stamp will be made of the creation date. This will be added to the item.
  The visitor will receive an email after the saving process of the item. These email contains three important links:
  1. A link to view the item on the website
  2. A link to edit the item
  3. A link to remove the item
  If the user wasn’t already registered an additional link will be provided. This link will confirm his email address and creates a personal page for the visitor. This process is the same as the registering process described above.
  Even when the user is a registered user this email will be sent, because it gives him the ability to edit the item with just one click.

- **View personal page Vp03**
  A visitor can view all his own items. The visitor needs to be registered and logged in to access this page. On this page the visitor will see all his items.

- **Remove item Vr01**
  It is possible to remove an item in two ways:
1. Access the link provided in the email that was send when the item was added.
2. Access the link provided on the personal page of the visitor. This option is only possible when the visitor is a registered visitor.

A confirmation will be asked before the item is really removed from the website. A reason for removal will be asked when removing the item from the website. If the visitor removes the item because he or she sold it, he will be asked to provide the email address of the buyer. An email will be send to the buyer to ask him or she to rate the seller. A cookie will be set at the sellers browser to make it harder to rate himself.

- **Edit item Ve01**
  It is possible to edit an item in two ways:
  1. Access the link provided in the email that was send when the item was added.
  2. Access the link provided on the personal page of the visitor. This option is only possible when the visitor is a registered visitor.

After either one of the two options the user will access the editing page of the item. On this page it is possible to edit all specifications of the item. After editing the item the IP-address information will be updated. Though the creation date of the item will not be updated.

- **Contact seller Vc01**
  When the visitor is viewing the details of a specific item, it is possible to send an email to the seller. This is done by accessing a link on the this page. The visitor will only see a form to type his message. Details about the email address of the seller will not be revealed.

- **Rate seller Vr03**
  As mentioned above the email address of the buyer will be asked when removing an item. This email contains two links:
  1. A link to rate the seller negative. This means the selling process wasn’t handled in a positive way.
  2. A link to rate the seller positive. This means the selling process was handled in a positive way.

By providing two links in the email it is possible to rate the seller very fast. This way it is easier for buyers to rate the seller.

- **View advertisement Vv02**
On every page a number of advertisements will be shown depending on the contents of that page. A visitor can click on the advertisement to go directly to the webpage of the advertiser.

- **List advertisers VI02**
  Visitors can view a list of all advertisers. A visitor can also see all advertisers that belong are advertising in a specific category.

- **View advertiser details Vv03**
  A visitor can view the details of an advertiser by selecting a advertiser out of the list of advertisers. The following information will be shown:
  1. Name
  2. Address
  3. Email
  4. Webpage
  5. Telephone

- **View site map Vv04**
  A site map will be shown when choosing this option.

- **View help Vv05**
  A manual for the website will be shown when choosing this option.

- **Contact administrator Vc02**
  A form will be shown when choosing this option. A visitor can contact the administrator by filling in this form. No email address information of the administrator will be revealed.

- **Flag inappropriate Vf01**
  If an item contains inappropriate language or pictures a visitor can flag this item. This item will reported to the administrator. This item will also be marked on the items list as possibly inappropriate.

- **List statistics VI03**
  Every user should be able to see how much visitors the website has.

- **Switch language Vs01**
  It should be easy to extend the system with multiple language support.

These requirements have a few dependencies, listed in table 1.
Advertisers
Advertisers are the companies or persons who will advertise on this website.Advertisers are able to do all the actions that visitors are able to do. Additional actions that are available for advertisers:

- **Add advertisement Ra01**
  Every advertiser has a number credits to put advertisement on the website. The advertiser can add advertisement until his credit is reduced to 0. The following information needs to be provided:
  
  1.  Title
  2.  Line 1
  3.  Line 2
  4.  Keywords
  5.  (Optional) link

- **Edit advertisement Re01**
  An advertiser is able to edit his own advertisement. Every detail of the advertisement can be edited on this page.

- **Remove advertisement Rr01**
  An advertiser is able to remove his advertisement. The amount of credits is reduced slower when an advertisement is removed.

- **List own advertisements Rl01**
  A advertiser can view a list of own advertisements after logging into the website. The login information will be provided by the administrator. Statistics about every advertisement will also be provided on this page.

These requirements have a few dependencies, listed in table 2.

<table>
<thead>
<tr>
<th>Requirement (code)</th>
<th>Depends on (code)</th>
</tr>
</thead>
<tbody>
<tr>
<td>VI01</td>
<td>Vr02</td>
</tr>
<tr>
<td>Vp03</td>
<td>VI01</td>
</tr>
<tr>
<td>Va01</td>
<td>VI01</td>
</tr>
</tbody>
</table>

*Table 1. Requirements dependencies for visitors.*
Administrators
Administrators are the person who will control the website. They are responsible for keeping the website running.
Administrators are able to do the following:

- **Add category Ac01**
  An administrator is able to add a category to the website. The administrator needs to provide the extra specifications for this category

- **Edit category Ae01**
  An administrator is able to edit a category. No specifications can be removed. Specifications can only be added.

- **List inappropriate Al01**
  An administrator is able to view all inappropriate marked items. Then the administrator can remove or edit these items. After editing these items, they are no longer marked as inappropriate.

- **Add administrator Aa01**
  An administrator should be able to add other administrators.

- **Remove administrator Ar01**
  An administrator should be able to remove other administrators.

- **Remove item Ar02**
  An administrator can remove an item if necessary.

- **Edit item Ae02**
  An administrator can edit an item if necessary.

- **Add advertiser Aa02**
  An administrator is able to add an advertiser. The following information needs to be provided:
  1. Name
  2. Email
3. Total amount credits
4. Credit cost for one advertisement per day

- **Remove advertiser Ar03**
  An administrator is able to remove an advertiser if necessary.

- **Edit advertiser Ae03**
  An administrator is able to remove an advertiser if necessary.

- **List registered users Al02**
  An administrator is able to view all users.

- **Search registered users As01**
  An administrator is able to search for a user. The administrator needs to provide a keyword. A user will be added to the result list if this keyword matches any specification of the user.

- **View details registered user Av01**
  An administrator is able to select a user from the list of users and view the details of this user.

- **Edit registered user Ae04**
  An administrator is able to edit any specification of a user.

- **Remove registered user Ar04**
  An administrator is able to remove any user.

- **Email registered user Ae05**
  Any user can be emailed by the administrator.

These requirements have a few dependencies, listed in table 3.

<table>
<thead>
<tr>
<th>Requirement (code)</th>
<th>Depends on (code)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A*</td>
<td>Vl01</td>
</tr>
</tbody>
</table>

*Table 3. Requirements dependencies for administrators.*

**Non-functional requirements**

*User interface and usability*
This website has to be as simple and easy to use as possible. Every novice user should be able to buy or sell an item using this website.

**Documentation**
A clear and structured document about how to maintain this system has to be written. Furthermore an extensive document has to be written in which most functions and classes have to be described.

**Hardware and software requirements**
To host the website, a computer is needed with PHP 5, MySQL 5 and Apache installed on it. Visitors of the website need a computer with internet connection and a web browser.

**Performance characteristics**
The website should also work well on slow internet connections, because not everybody has access to a reliable and fast internet connection. It might even be necessary to create another version of the website without pictures and photographs.

**Security**
The website has to be very secure and it should not be possible to use it for other purposes.
There should be different right levels for administrators. For example, not every administrator can add or remove other administrators.

Users of the website can create their own marketplace. They have to register and login to be able to open their own marketplace. Advertisers need to login as well to be able to add/remove advertisements. Administrators need to login as well to be able to do maintenance on the website.

This login procedure has to very secure and safe from SQL-injections. Passwords needs to be stored as encrypted passwords.

**Policies**
Administrators have to be trustable persons. They have to act right when judging about inappropriate language, when adding administrators and when controlling all details about registered users. It is illegal for an administrator to distribute user details to third parties. It is also illegal for administrators to remove registered users, unless this users are acting inappropriate.
A policy has to be made to judge whether behavior of a user is inappropriate. This policy will be made in cooperation with the UCC.
Personal information like email addresses and passwords will be stored until the users himself asks an administrator to remove it or until an administrator decided the users should be removed because of inappropriate language or violating other rules.
Usecase models

Usecase models describe for every actor what the possible actions are he or she can take.

Visitor

Figure 1: Use case model Visitor
Use case: Search for items Vs01

Related use cases: View item details

Actors: Visitors, advertises, administrators

Entry condition: User is on the front page.

Steps:

User action: System reaction:

1. User fills in keywords and clicks search
   2. System searches with the keywords provided.
   3. System shows list with results to user.

Exit condition: List of results is shown.

Use case: Place a bid Vp02

Related use cases: View item details

Actors: Visitors, advertises, administrators

Entry condition: If the user has an account, he’s logged in.

Steps:

User action: System reaction:

1a. (Not registered) User fills in a bidding price, email address and clicks add.
   2a. System sends a confirmation email to the provided email address.

1b. (Registered) User fills in a bidding price and clicks add.
   3a. User clicks on confirmation link.
   4. System adds bid to the item and sends email to seller.

Exit condition: Bid is added to the item in the database.

Use case: Browse items Vb01

Related use cases: View item details

Actors: Visitors, advertises, administrators

Entry condition: User is on the front page.

Steps:

System reaction: 2. System shows all items in that category, divided over different pages if needed.

Exit condition: Results are shown.

Use case: Send link to a friend Vs02
Related use cases: View item details, search
Actors: Visitors, advertises, administrators
Entry condition: If the user has an account, he’s logged in.

Steps:

User action: 1a. (Not registered) User fills in an email address of a friend, a message, his own name and clicks send.

System reaction: 1b. (Registered) User fills in an email address of a friend, a message and clicks send.

2. System sends the link to the provided email address with the provided message and name.

Exit condition: Link is send to provided email address.

Use case: Remove bid Vr01
Related use cases: Place bid
Actors: Visitors, advertises, administrators
Entry condition: If the user has an account, he’s logged in and viewing an item detail page.

Steps:

User action: 1a. (Not registered) User clicks remove bid in the place bid confirmation email.

System reaction: 1b. (Registered) User clicks remove bid alongside his bid on the item details page.
User action: 2. System removes bid and sends email to seller.

Exit condition: Bid is removed from the item.

Use case: Add item Va02
Related use cases:
Actors: Visitors, advertises, administrators
Entry condition: If the user has an account, he’s logged in.
User is viewing the front page.

Steps:

User action: System reaction:
1a. (Not registered) User clicks add item on front page.
2. System shows add item page
1b. (Registered) User clicks add item on personal page or front page.
3. User selects category
4. System displays fields required for provided category.
5. User fills in fields.
5a. (Not registered user) System sends confirmation email.
6a. (Not registered) User clicks confirmation link in email.
7. System saves item.

Exit condition: Item is added to the database.

Use case: Edit item Ve01
Related use cases: Add item
Actors: Visitors, advertises, administrators
Entry condition: If the user has an account, he’s logged in and viewing his personal page. User has added items.

Steps:
User action:  
1a. (Not registered) User clicks edit item in the add item confirmation email.  
1b. (Registered) User clicks edit item alongside the item on his personal page.  

System reaction:  
2. System shows edit item form.  
3. User edits fields and clicks save.  
4. System saves the item.  

Exit condition:  
Updated item is saved in the database.

Use case:  
Remove item Vr01  
Related use cases:  
Add item  
Actors:  
Visitors, advertises, administrators  
Entry condition:  
If the user has an account, he’s logged in.  
Steps:  

User action:  
1a. (Not registered) User clicks remove item in the add item confirmation email.  
1a. (Registered) user clicks remove item alongside the item on his personal page.  

System reaction:  
2. System shows form asking why the user wants to remove the item.  
3. User clicks ‘sold it’ and fills in email address of the buyer.  
4. System sends an email with a rate positive and a rate negative link to buyer.  

Exit condition:  
Item is removed from the database.

Use case:  
List statistics VI03  
Related use cases:  
Actors:  
Visitors, advertises, administrators  
Entry condition:  
User is on the website.  
Steps:

System reaction: 2. System shows statistics of the website.

Exit condition: Statistics are shown.

Use case: Contact seller Vc01
Related use cases: View item details
Actors: Visitors, advertises, administrators
Entry condition: If the user has an account, he’s logged in. User is viewing an item details page.

Steps:

User action:

1. User clicks on contact seller on the view item details page.

System reaction: 2. System shows a contact form.

2a. (Not registered) User fills in message, email address and clicks send.

2b. (Registered) User fills in message and clicks send.

3a. System sends confirmation email to users email address.

4a. User clicks confirmation link.

5. System sends email to seller.

Exit condition: Email is send to provided email address.

Use case: View help Vv05
Related use cases:
Actors: Visitors, advertises, administrators
Entry condition: User is on the website.

Steps:

User action:

1. User clicks help.

System reaction: 2. System shows help.

Exit condition: Help is shown.
Use case: Flag inappropriate Vf01
Related use cases: View item details
Actors: Visitors, advertises, administrators
Entry condition: User is on an item details page.
Steps:

User action: System reaction:
1. User clicks flag inappropriate 2. System marks item as possibly inappropriate.

Exit condition: Item is marked inappropriate.

Use case: View site map Vv04
Related use cases:
Actors: Visitors, advertises, administrators
Entry condition: User is on the website.
Steps:

User action: System reaction:
1. User clicks on site map. 2. System shows site map.

Exit condition: Site map is shown.

Use case: List advertisers Vl02
Related use cases:
Actors: Visitors, advertises, administrators
Entry condition: User is on the website.
Steps:

User action: System reaction:
1. User clicks on list advertisers. 2. System shows all advertisers.

Exit condition: List of advertisers is shown.

Use case: Rate seller Vr03
Related use cases:
Actors: Visitors, advertises, administrators
Entry condition: User is a buyer of an item.
Steps:

**User action:**

1. User clicks either on rate negative or rate positive link in sold email.

2. System saves vote.

**Exit condition:**

Seller has been rated by the buyer.

**Use case:**

Print an item **Vp01**

**Related use cases:**

View item details

**Actors:**

Visitors, advertises, administrators

**Entry condition:**

User is on an item details page.

Steps:

**User action:**

1. User clicks print item.


**Exit condition:**

The item has been printed.

**Use case:**

Contact administrator **Vc02**

**Related use cases:**

View item details

**Actors:**

Visitors, advertises, administrators

**Entry condition:**

If the user has an account, he’s logged in.

Steps:

**User action:**

1. User clicks contact admin.

2. System shows contact form.

3. User fills in form and clicks send.


5. System shows confirmation message.

**Exit condition:**

Message has been send to an administrator.

**Use case:**

View item details **Vv01**

**Related use cases:**

Search

**Actors:**

Visitors, advertises, administrators

**Entry condition:**

User is browsing a list of items.
Steps:

User action: 
1. User clicks on an item.

System reaction: 
2. System shows page with item details and all bids (if any)

Exit condition: 
Item details are shown.

Use case: View advertisement Vv02
Related use cases: Search, browse items
Actors: Visitors, advertises, administrators
Entry condition: User is browsing items.

Steps:

User action: 
1. User searches or browses a category.

System reaction: 
2. System shows page with items including relevant advertisements

Exit condition: Advertisements are shown.

Use case: View advertiser details Vv03
Related use cases: List advertisers
Actors: Visitors, advertises, administrators
Entry condition: User is browsing list of advertisers.

Steps:

User action: 
1. User clicks on an advertiser.

System reaction: 
2. System shows a page with details of the advertiser.

Exit condition: Details of advertiser are shown.

Use case: Register Vr02
Related use cases: 
Actors: Visitors, advertises, administrators
Entry condition: User is on the front page.

Steps:
**User action:**
1. User fills in email address, password, check ‘new user’ and clicks login.

**System reaction:**
2. System sends a confirmation email.
3. User clicks link in confirmation email.
4. System shows form asking for more details, like first name and last name.
5. User fills in form and clicks save.
6. System saves user.

**Exit condition:**
A new user account has been added to the database

**Use case:**
Login VI01

**Related use cases:**
Register

**Actors:**
Visitors, advertises, administrators

**Entry condition:**
User is on the front page.

**Steps:**

**User action:**
1. User fills in email address, password and clicks login.

**System reaction:**
2a. (Success) System shows personal page.
2b. (Failure) System shows form again.

**Exit condition:**
User is logged in.

---

**Use case:**
Add to wish list Va01

**Related use cases:**
Login, view item details

**Actors:**
Visitors, advertises, administrators

**Entry condition:**
User is a registered user and user is logged in.

**Steps:**

**User action:**
1. User clicks add to wish list on view item details page.

**System reaction:**
2. System adds the item to wish list of currently logged in user.

**Exit condition:**
Item is added to wish list of user.
Use case:   View personal page Vp03
Related use cases:   Login
Actors:   Visitors, advertises, administrators
Entry condition:   User is a registered user and user is logged in.
Steps:

User action:  System reaction:

1. User clicks on personal page link.

2. System shows personal page, including wish list, user items and user details.

Exit condition:   Personal page is shown.
Advertiser

Figure 2: Use case model Advertiser

Use case: List advertisements R101
Related use cases: Login, personal page
Actors: Advertises, administrators
Entry condition: User is logged in.
Steps:

User action: System reaction:
1. User logs into the system. 2. System shows personal page with a list of advertisements and details of all advertisements.

Exit condition: Advertisements are shown.

Use case: Add advertisement Ra01
Related use cases: Login, personal page
Actors: Advertises, administrators
Entry condition: User is logged in and has credits.
Steps:

User action: System reaction:

1. (Registered) User clicks on add advertisement on personal page. 2. System shows form to add advertisement.

3. User fills in the form and clicks add

4. System saves advertisement.

5. System shows personal page.

Exit condition: Advertisement is added to the database.

Use case: Edit advertisement Re01
Related use cases: Login, personal page
Actors: Advertises, administrators
Entry condition: User is logged in and has added an advertisement.

Steps:

User action: System reaction:

1. (Registered) User clicks on ‘edit advertisement’ on personal page.

2. System show form with values of selected advertisements.

3. User edit the values and clicks save.

4. System updates the advertisement

5. System shows personal page

Exit condition: Updated advertisements is saved in the database.

Use case: Remove advertisement Rr01
Related use cases: Login, personal page
Actors: Advertises, administrators
Entry condition: User is logged in and has added an advertisement.
Steps:
User action:  System reaction:
1. (Registered) user selects
   advertisements and clicks on ‘remove
   advertisement’

2. System asks for confirmation

3. User selects ‘yes’ to remove the
   advertisement

4. System deletes advertisement

5. System shows personal page

Exit condition: Advertisement is removed from the database.
Figure 3: Use case model Administrator
Use case: List registered users Al02
Related use cases: Login
Actors: Administrators
Entry condition: User is logged in.
Steps:
User action: System reaction:
1. User clicks list registered users link. 2. System shows list with all registered users.

Exit condition: List of registered users is shown.

Use case: View details registered user Av01
Related use cases: Login, list registered users
Actors: Administrators
Entry condition: User is logged in and browsing registered users list.
Steps:
User action: System reaction:
1. User clicks on a registered user in the list of registered users. 2. System shows page with details about the registered user.

Exit condition: Details of a registered user are shown.

Use case: List inappropriates Al01
Related use cases: Login
Actors: Administrators
Entry condition: User is logged in.
Steps:
User action: System reaction:
1. User clicks list inappropriate link. 2. System shows page with all possibly inappropriate items.

3. User can edit or remove items.
Exit condition: Inappropriate items are shown.
Use case: Remove registered user Ar04
Related use cases: Login, list registered users
Actors: Administrators
Entry condition: User is logged in and viewing list of registered users.
Steps:

User action: System reaction:
1. User clicks remove registered user. 2. System asks confirmation.

Exit condition: User is removed from the database.

Use case: Email registered user Ae05
Related use cases: Login, list registered users
Actors: Administrators
Entry condition: User is logged in and viewing list of registered users.
Steps:

User action: System reaction:
1. User clicks on email user in list of all registered users. 2. System shows email form.
3. User fills in form and clicks send. 4. System sends the email message.

Exit condition: Email message is send to the registered user.

Use case: Add category Ac01
Related use cases: Login
Actors: Administrators
Entry condition: User is logged in and viewing personal page.
Steps:

User action: System reaction:
1. User clicks on add category. 2. System shows category form.
3. User fills in form and clicks add.
User action: System reaction:

Exit condition: Category is added to the database.

Use case: Edit category Ae01
Related use cases: Login
Actors: Administrators
Entry condition: User is logged in and viewing list of categories.
Steps:

User action: System reaction:

1. User clicks on edit category.

2. System shows category form.

3. User fills in form and clicks edit.

4. System edits category.

Exit condition: Updated category is saved in the database.

Use case: Add administrator Aa01
Related use cases: Login, list registered users
Actors: Administrators
Entry condition: User is logged in and viewing personal page.
Steps:

User action: System reaction:

1. User clicks on add administrator in the registered users list.

2. System shows administrator form.

3. User fills in form, assigns rights and clicks add.

4. System saves administrator with provided rights.

Exit condition: New administrator is added to the database.

Use case: Remove administrator Ar01
Related use cases: Login, list registered users
Actors: Administrators
Entry condition: User is logged in and viewing list of administrators.
Steps:

User action: System reaction:
1. User clicks on remove administrator in the administrator list. 2. System asks for confirmation.

Exit condition: Administrator is removed from the database.

Use case: Search registered users As01
Related use cases: Login
Actors: Administrators
Entry condition: User is logged in and viewing his personal page.

Steps:

User action: System reaction:
1. User clicks search user. 2. System shows search for registered user page.
3. User fills in a name to search for and clicks search. 4. System shows list with users matching this name.

Exit condition: List of matching users is shown.

Use case: Remove registered user Ar04
Related use cases: Login, Search registered users, List registered users
Actors: Administrators
Entry condition: User is logged in and viewing a list of registered users.

Steps:

User action: System reaction:
1. User clicks remove behind a user. 2. System asks for a confirmation and the reason why this user should be removed.
3. User fills in form and confirms.
User action:  System reaction:
4. System removes user and sends this user an email with the message filled in earlier.

Exit condition:  Registered user is remove from the database.

Use case:  Edit registered user Ae04
Related use cases:  Login, Search registered users, List registered users
Actors:  Administrators
Entry condition:  User is logged in and viewing a list of registered users.
Steps:

User action:  System reaction:
1. On the users list, the user clicks edit behind the name of a user.
2. System shows a form with user details.
3. User edits the form and clicks save.
4. System saves the edited user.

Exit condition:  Updated user is saved in the database.

Use case:  Edit item Ae02
Related use cases:  Login, Search for item, List inappropriate
Actors:  Administrators
Entry condition:  User is logged in and viewing a list of items.
Steps:

User action:  System reaction:
1. On the item list, the user clicks behind an item on edit.
2. System shows a form with item details.
3. User edits the form and clicks save.
4. System saves the edited item.

Exit condition:  Updated item is saved to the database.

Use case:  Add advertiser Aa02
Related use cases:  Login
Actors:  Administrators
Entry condition: User is logged in and viewing his personal page.

Steps:

User action: System reaction:
1. The user clicks add advertiser. 2. System shows a form with advertiser details.
3. User fills in the form and clicks save.
4. System saves the new advertiser.

Exit condition: The new advertiser has been added to the database.

Use case: Edit advertiser Ae03
Related use cases: Login, List advertisers
Actors: Administrators

Entry condition: User is logged in and viewing a list of advertisers.

Steps:

User action: System reaction:
1. On the advertisers list the user clicks edit behind the name of an advertiser.
2. System shows a form with advertiser details.
3. User edits the form and clicks save.
4. System saves the edited advertiser.

Exit condition: Updated advertiser is save in the database.

Use case: Remove advertiser Ar03
Related use cases: Login, List advertisers
Actors: Administrators

Entry condition: User is logged in and viewing list of advertisers.

Steps:

User action: System reaction:
1. On the advertisers list the user clicks remove behind the name of an advertiser.
2. System asks for confirmation and a reason for deletion.
User action: 3. User fills in the form and confirms.

System reaction: 4. System removes advertiser and sends an email with the message provided earlier to the advertiser.

Exit condition: Advertiser is removed from the database.
Flowcharts

These flowcharts show all the different screens possible.

Visitor

Figure 4: Flowchart Visitor
Advertiser
An advertiser can do everything a normal visitor can. This flowchart shows it’s screens.

Administrator
An administrator can do everything an advertiser can do and more.
Appendix B: Architectural design

Introduction

This architectural design document describes the global design of the online marketplace that’s going to be implemented. It describes the various subsystems and how these subsystems are going to work together.

Subsystem decomposition

This system will be implemented by using the model-view-controller design pattern. This pattern is displayed in figure 1.

![Model-view-controller design pattern](image.png)

The model-view-controller (MVC) design pattern is used to isolate business logic from user interface considerations. Applying this design to an application makes it easier to maintain, because the interface can be changed without the underlying business logic or vice versa. Globally the components of an MVC can best be described as:

**Model**  The domain-specific representation of the data the application operates on.
**View**  
Used to render the model in a human readable from.

**Controller**  
Connects both the model and view. The controller also deals with all input that the user sends to the application and uses it to manipulate the model, after which the view can be updated.

Applying this design pattern to the system that is going to be developed will give something like figure 2.

![MVC Design Pattern Diagram](image)

*Figure 2. MVC design pattern applied to the system.*
In the eventual application the controller will receive all HTTP requests. The controller than
decides what has to happen, for example do a search for items. The controller construct
the right model and asks it for the data. The model connects to the database, queries it for
the right records and returns the results in a useful form. The controller then constructs a
view and hands it the data from the model. After the view is rendered, it is send to the
client’s internet browser.

Framework

The online market place has to be developed using PHP and MySQL. Because the MVC
design pattern is a widely used principle, especially in the web development world, a lot of
so-called frameworks exist. These frameworks provide thorough tested components which
can be used to create an MVC application. The online market place is going to be built
using the Zend Framework.

The Zend Framework (ZF) is an open source framework implemented in PHP 5 and
licensed under the New BSD License. Zend Framework is developed by a company called
Zend Technologies and their partners, including IBM, Google, Microsoft and StrikeIron.
Code contributions to Zend Framework are subject to rigorous code, documentation and
test standards. All code must meet ZF’s coding standards and unit tests must reach 80%
code coverage before the corresponding code may be moved to the release branch.¹

Database

All data is going to be stored in a database. ZF support multiple database systems,
including MySQL, which the online trading website will be using.
The database design is described in figure 3. The red table (books) is used as an
example, for every category such a table is going to be constructed, so that each category
can have specific specifications.


Version 4, awaiting feedback
Figure 3. Database design.
Appendix C: Technical design

Class Diagrams

Below the class diagram of the total system will be given. No information about the classes will be shown in this diagram, because of the dimensions of all the classes. Further on more detailed diagrams are given. The total system will be decomposed into two parts: the models and the controllers.

Total System

![Class Diagram of the Total System](image)

Figure 1. Total class diagram.
Controllers

Below the class diagram for all the controllers will be given.

Figure 2. Controller class diagram.
Figure 3. Model class diagram.
Graphical User Interface
Below the graphical user interface is shown.

Figure 4. Front page of the marketplace.
Figure 5. Search results screen.
Figure 6. Item detail screen.
Figure 7. Personal page of visitor screen.
**Figure 8. Personal page of advertiser screen.**
Figure 9. Personal page of administrator screen.
Optimizing

In the beginning of the project, the site will be launched internally within the UDSM. This means that the internet connection speed to this website is fast enough to support a common graphical website. Although the bandwidth is limited, so the size of used images has to be reduced to a minimum.

In a later stage this website will be used nationally. The internet connection speed in Tanzania is very low, so this website has to be optimized for this connection speed.