SOFTWARE REQUIREMENTS SPECIFICATION

for

Mar.kt

Version 1.3 approved

Prepared by
Aayush Jain
Aditi Rao
Spandana Balumuri

CodeCacophony

21 February 2019

Contents

Revision History

Name	Date	Reason For Changes	Version
Mar.kt	17 Jan 2019	Base SRS	1.0
Mar.kt	21 Feb 2019	Addition of features	1.1
Mar.kt	6 Mar 2019	Addition of diagrams	1.2
Mar.kt	23 Mar 2019	Updates in additional requirements	1.3

1 Introduction

1.1 Purpose

Mar.kt (currently in version 1.3) is a one-stop marketplace geared towards college students for sale of second hand items, or free distribution of electronic learning aids. This document covers the requirements for the entire application up till its proposed version 2.0 release.

1.2 Document Conventions

The descriptions of requirements in this document are ordered in increasing order of complexity of the functionality they seek to provide. Thus, they are essentially ordered according to the proposed sequence of integration and testing at any point of time.

1.3 Intended Audience and Reading Suggestions

The SRS is intended for users (students), developers, project managers, testers and document writers. The SRS starts with an introduction which is meant for all audiences, then it gives an overall description of the product, its functions, operating environment etc., which are intended for users as well as developers. The user documentation part is meant for the document writers. Next come the external interface requirements, other requirements which are useful for developers, testers. The non-functional requirements are intended for both developers and project managers. Next comes the system requirements are useful for testers as well as developers.

1.4 Project Scope

Mar.kt is envisioned to be a one-stop marketplace especially geared towards college students for sale of second hand items, or free distribution of electronic learning aids. Due to the difficulties in transportation and the unique requirements of colleges (in terms of books, notes etc.), Mar.kt functions within the college environment. It fills the gap in the market in terms of determining pricing through bids, ensuring that both sellers and buyers are getting the best deals. Allowing for seller review functionality for personal items like notes allows students to make informed decisions about their purchases, as well as increasing the seller's credibility (thus influencing future sales).

1.5 References

- Heroku: https://devcenter.heroku.com/
- Django: https://docs.djangoproject.com/en/2.1/
- Python: https://docs.python.org/3/

2 Overall Description

2.1 Product Perspective

Online shopping is the process whereby consumers directly buy goods, services etc. from a seller interactively in real-time without an intermediary service over the internet. An E-Commerce portal serves as a virtual marketplace, which acts as an interface between buyers and sellers, for the sale of goods and services. Here we create an E-Commerce portal, a web application which acts as an interface between students to buy and sell goods.

2.2 Product Functions

- A basic UI where a customer is provided with an online cart and also access the product catalogue
- A UI where a vendor is provided with an online inventory management system to list products for sale.
- A feedback system where customers can rate the vendors and write a review on the products and on the transaction with the vendor.

2.3 User Classes and Characteristics

There are two classes :

Customer : A customer can

- view the product catalogue
- bid on products
- add/delete products from the online cart
- write and read reviews for the products.

Vendor : A vendor can

- add and delete the products for sale
- choose a bid from the list of bids for a particular product.

Note: A Customer can also be a Vendor, so the login details are stored in the same 'User' table.

2.4 Operating Environment

The app will run on any modern web browser, for example Google Chrome, Internet Explorer, Mozilla Firefox, Safari and will be hosted on Heroku.

2.5 Design and Implementation Constraints

The users should have any one of the modern web browsers mentioned above with a secure working internet connection. Since we use are using Python 3.x the recommended system requirements are -

- Processors: Intel® Core™ i5 processor 4300M at 2.60 GHz or 2.59 GHz (1 socket, 2 cores, 2 threads per core), 8 GB of DRAM.
- Disk space: 2 to 3 GB.
- Operating systems: Windows® 10, macOS, and Linux

We are using Django 3.7.x in Mar.kt. For a production environment, Django follows the WSGI spec, PEP 3333, which means it can run on a variety of server platforms. We are using SQLite as the database which is supported by Django.

2.6 User Documentation

- Heroku: https://devcenter.heroku.com/
- Django: https://docs.djangoproject.com/en/2.1/
- Python: https://docs.python.org/3/

2.7 Assumptions and Dependencies

Issues faced during development:

- Python is slower than other modern programming languages like Java and C++. However, the developers can create a custom runtime, and use it instead of the default runtime of the programming language.
- Python does not support block comments. Hence, the programmers have to assess the quality of the code either by writing comments for each line of code or removing a specific section of code at the time of execution
- Django is too monolithic - a single-tiered software application in which the user interface and data access code are combined into a single program from a single platform, and knowledge of full system is required to work.

- Components get deployed together in Django. You have to manually scale your application on high traffic in Heroku.
- You can't login to your server via SSH.

Assumptions in Mar.kt:

- No quality assurance provided for the products.
- The users should be students and will only be able to access the web application only through a *".edu"* mail-id.
- The seller's information is not verified and the seller is given a choice in type of contact information (mobile number of e-mail address).
- All further communication after the seller and buyer have been matched have to be done outside the application.

3 External Interface Requirements

3.1 User Interfaces

User interfaces are designed to mimic existent E-Commerce sites while the design will be in line with modern social media apps (appealing to the younger generation). The following themes and design standards will be followed (in brief):

Overall Design Concept: Minimal, product focus, in line with traditional systems bordering on social media interface design. Screen Layouts: In line with traditional E-Commerce Applications in terms of form and functionality (e.g.: Ebay, Flipkart) Color Schemes: use of minimalist material pastel shades to give the impression of quality, as well as high technological capability Font Specification : sans-serif font faces Standard functionality : Login and Logout to be provided on every page. Option to switch from buyer to seller interfaces in a single button click (present near logout) Feedback systems: High priority feedback : Popup confirmation alerts. Low Priority Feedback: Loading icons, color changes etc.

User interfaces will be designed in HTML5 with CSS 3.0 for styling and JavaScript integration for interactivity and feedback systems. Golden Rules of design shall be incorporated within the design with minimal trade-offs (expected).

3.2 Hardware Interfaces

- The hardware interfaces used are - monitor, keyboard and mouse.
- The monitor is used to display the different pages of the web application and to show the changes being made, to the user.
- A mouse is used to navigate from one page to the other by clicking on buttons and links.
- We use a keyboard while filling forms, for example the details of product, login page, registration page etc...
- We can also use the keyboard (arrow keys) to navigate through the application.

3.3 Software Interfaces

We are using Python 3.x for the back-end, since it has great support for various libraries and APIs. All the business logic will be written in Python using a Django 2.7.x framework. For the front-end we use HTML, CSS and JavaScript. The versions used are

HTML5 and CSS 3.0. To store the user, product, bids and mail data we use SQLite as the database which is supported by Django. Initially we have a login/registration page where we use the user table and the user id is passed through all the pages. Later we use the product table and bids table to display the product, category pages and bids (of a particular product by a vendor) respectively.

3.4 Communications Interfaces

We plan to host Mar.kt on Heroku, a cloud platform as a service. That means we do not have to worry about infrastructure and can just focus on your application. After a customer bids on a certain product, the bid is visible in list of bids to the vendor. Now the vendor can choose one of the bids, then that particular customer is notified via mail about the acceptance. Also the customer has to send a pre-written acknowledgment mail to the vendor to proceed with the transaction. The Heroku platform automatically routes HTTP requests sent to your app's hostname(s) to your web dynos. The entry point for all applications on the Common Runtime stack is the herokuapp.com domain which offers a direct routing path to your web dynos.

4 System Features

The functional requirements are organized based on the user classes (customer, vendor) -

4.1 Customer Class

4.1.1 Description and Priority

The customer class essentially represents a buyer. The attributes of the class are the attributes in the User table, because a user can be both a customer as well as a vendor. This class in fact is of very high priority, because without the customer class the transactions cannot occur.

4.1.2 Stimulus/Response Sequences

A customer can:

- login/register in Mar.kt - the details are stored in the User table in the database. (6)
- browse through the product catalogue - products are classified into different categories and clicking on product displays the complete details of the product. (9)
- bid on products - the product is automatically added into the cart and also into the vendor's bids list for the product. (9)
- add/delete/change bid on products in the online cart - the addition/deletion/change in bid price is automatically reflected in vendor's bids list for the product. (4)
- write and read reviews for the products - after a transaction, the customer can write reviews on the product and rate it. (4)

4.1.3 Functional Requirements

R1. the customer should be provided with a login/registration form.

R2. the web application should have categories and product pages to display the details of the products.

R3. the customer should be provided with an online cart.

R4. the customer should be provided with a feedback system/form.

R5. the customer should be provided with a mailbox.

4.2 Vendor Class

4.2.1 Description and Priority

The vendor class essentially represents a seller. The attributes of the class are the attributes in the User table, because a user can be both a customer as well as a vendor. This class in fact is of very high priority, because without the vendor class there wouldn't be any products for sale.

4.2.2 Stimulus/Response Sequences

A vendor can:

- login/register in Mar.kt - the details are stored in the User table in the database. (9)
- add and delete the products for sale - this is reflected in the categories of products, also in a customer's cart in case the product is deleted. (9)
- choose a bid from the list of bids for a particular product - the customer corresponding to the chosen bid is notified via mail, who later sends an acknowledgement mail if he/she wants to proceed with the transaction. (6)

4.2.3 Functional Requirements

R1. the vendor should be provided with a login/registration form.

R2. the vendor should be provided with an inventory to add/delete products.

R3. the customer should be provided with a mailbox.

5 Other Nonfunctional Requirements

5.1 Performance Requirements

The product shall be based on web and has to be run from a web server. The product shall take initial load time depending on internet connection strength which also depends on the media from which the product is run. The performance shall depend upon hardware components of the client/customer.

5.2 Safety Requirements

Mar.kt isn't responsible for the damage of products. It is only an interface between customers and vendors, the product quality is not assured by Mar.kt. Although it provides reviews by customers on products which they have bought, thus giving an intuition of the product and also about the seller.

5.3 Security Requirements

Since, application is intended for the authentication users only, so anonymous person should not be able to access and operate over the user data. Also the web application is intended for the exchange of products within the students only. In order to respect the privacy of the seller, the seller's contact information is disclosed to the customer only when the seller sends a confirmation email.

5.4 Software Quality Attributes

The web application is developed using the latest technologies which are quite fast and convenient for the students. Also Mar.kt is made to be supported on different types of systems. It is available 24/7 and reliable. It is made to be robust by using secure frameworks and test on maximum possible cases. It is very useful interface for students as they can get items quite quickly and at cheaper prices. The software technologies used are very easy to maintain as they are supported with a strong documentation.

5.5 Business Rules

The app grants access to two kinds of individuals, the buyer and the seller. Any given account can function as both a buyer and seller, however to become a seller, the account

must fulfill certain criteria. The buyer can bid on items marked for bidding or may buy items at a fixed price, as set by the seller. The seller has the ability to upload items to the app to sell them at a marked price or as a bid, being able to set a minimum bid amount. Additionally, the users must be able to communicate with each other using the provided messaging tool, and share files through the "Downloads" section of the app freely.

6 Other Requirements

The database used will be SQLite. The database will consist of Products, Users, Mails and Bids entities. Database entities can be noted in Appendix B (Class Diagram). Development of this project occurs in a system running the hardware requirements laid out in Section 2.5. Otherwise, the system must meet at least the minimum requirements of:

- Processors: Intel Atom® processor or Intel® Core™ i3 processor
- Disk space: 1 GB
- Operating systems: Windows* 7 or later, macOS, and Linux

All documentation is prepared in LaTeXand Markdown is the only other acceptable format (primarily for writing short summaries). The SRS is based off of the format specified by IEEE.
The UI depends on external libraries like FontAwesome, jQuery whose versions are yet to be added to this SRS.

6.1 Appendix A: Glossary

1. *Django: Python-based free and open-source web framework, which follows the model-view-template architectural pattern*
2. *Heroku: Cloud platform as a service supporting several programming languages including Python*
3. *SQLite: Relational database management system contained in a C programming library that is embedded into the end program rather than a client-server database engine*
4. *WSGI: Web Server Gateway Interface is a simple calling convention for web servers to forward requests to web applications or frameworks written in Python*
5. *PEP: Python Enhancement Proposal is the primary mechanism for proposing major new features, collecting community input on issues and documenting Python design decisions.*

6.2 Appendix B: Analysis Models

All analysis models have been designed in LucidChart using UML diagramming guidelines.

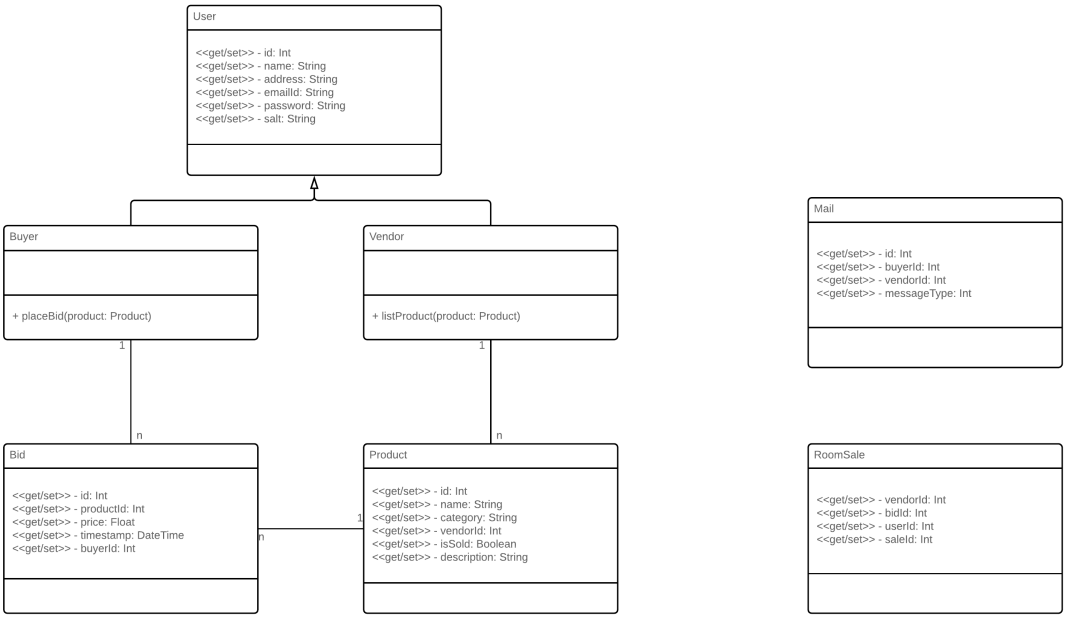

Figure 6.1: Class Diagram

6.3 Appendix C: To Be Determined List

1. *FontAwesome version*
2. *jQuery version*
3. *Database tables*

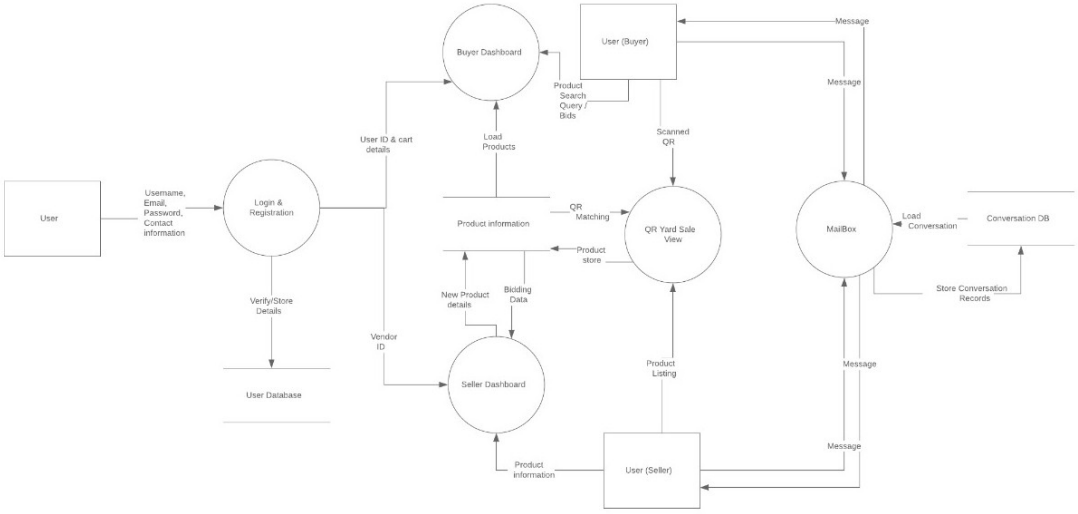

Figure 6.2: Data Flow Diagram

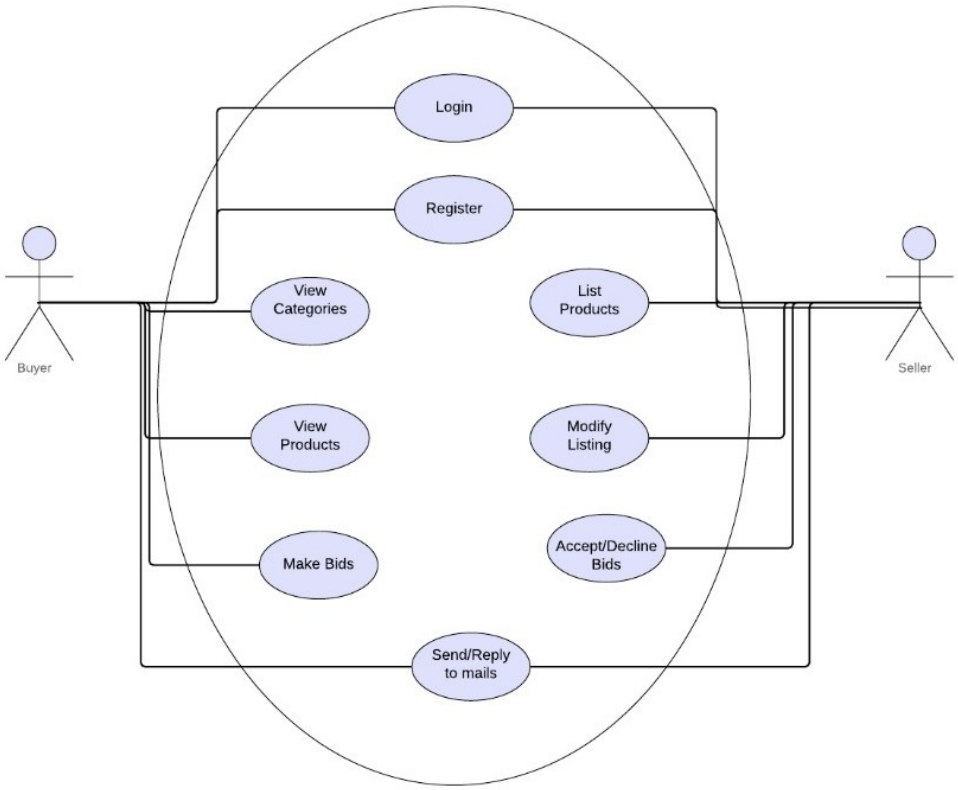

Figure 6.3: Use Case Diagram

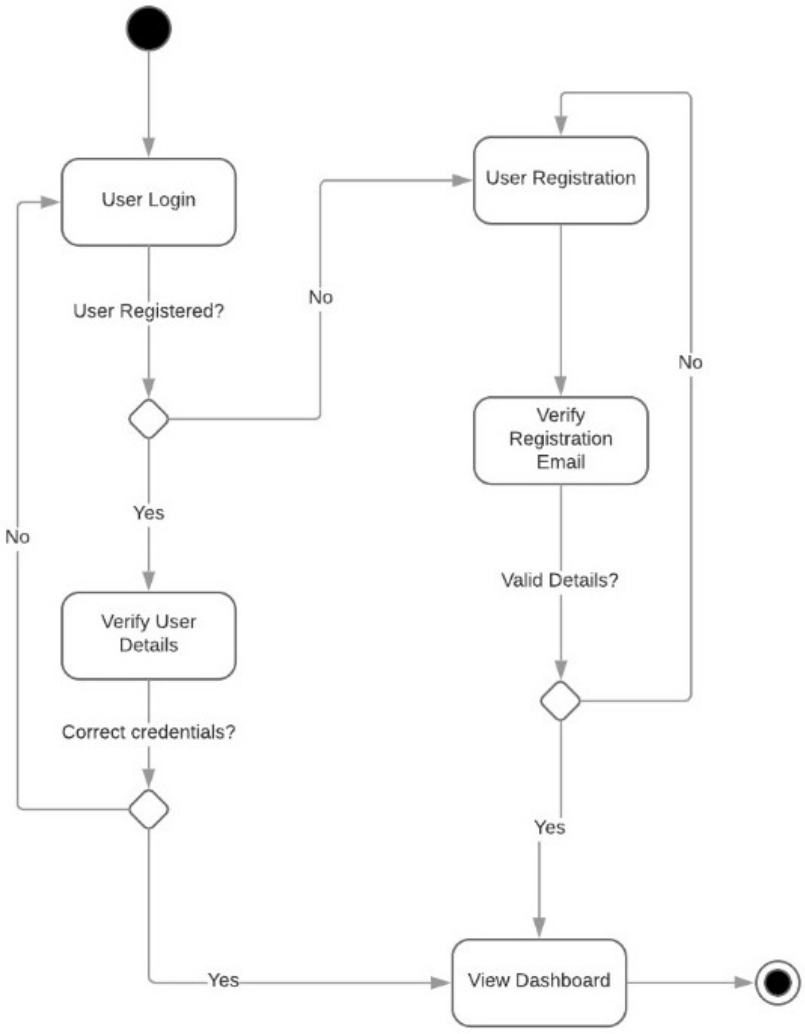

Figure 6.4: Login Activity Diagram

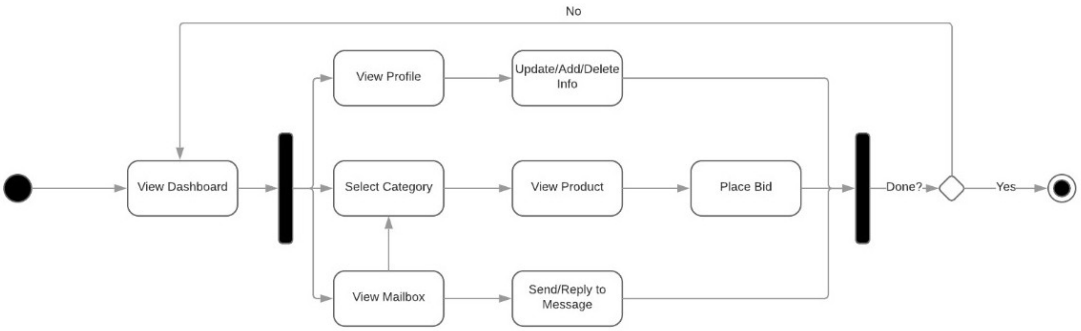

Figure 6.5: Buyer Activity Diagram

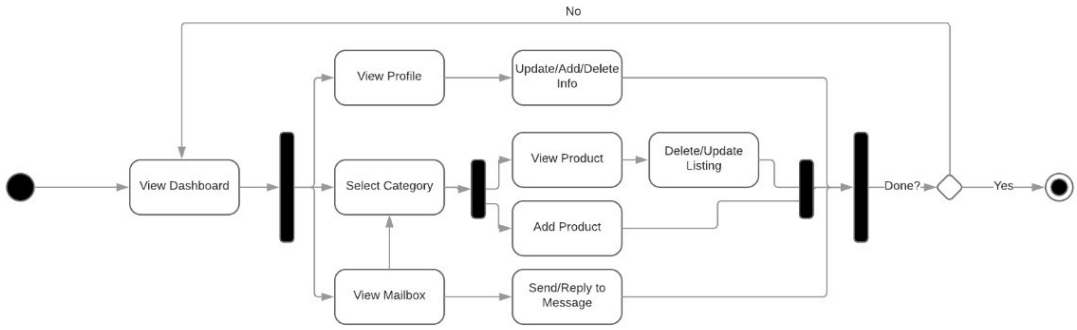

Figure 6.6: Seller Activity Diagram