[1. Introduction: 1](#_Toc408733305)

[1.1. Problems with traditional commerce: 1](#_Toc408733306)

[1.2. Objectives: 2](#_Toc408733307)

[1.3. Methodology: 2](#_Toc408733308)

[1.4. Tools 2](#_Toc408733309)

[2. Theoretical frame 3](#_Toc408733310)

[2.1. Electronic mobile Commerce: 3](#_Toc408733311)

[2.2. Shopping cart: 3](#_Toc408733312)

[2.3. Advantages of M-Commerce: 4](#_Toc408733313)

[2.4. Present Challenges Facing E-commerce and M-Commerce: 5](#_Toc408733314)

[2.5. Proposed project: 5](#_Toc408733315)

[2.6. Agile software development: 5](#_Toc408733316)

[2.6.1. Agile principles: 6](#_Toc408733317)

[2.6.2. Agile methods: 6](#_Toc408733318)

[2.6.3. Agile practices: 7](#_Toc408733319)

[2.7. Extreme programming (XP): 8](#_Toc408733320)

[2.7.1. XP Activities: 8](#_Toc408733321)

[2.7.2. XP Practices: 10](#_Toc408733322)

[2.8. Component-based software engineering (CBSE): 11](#_Toc408733323)

[2.9. Enterprise JavaBeans (EJB): 11](#_Toc408733324)

[2.10. Java Platform, Enterprise Edition (Java EE): 12](#_Toc408733325)

[2.11. Web service: 12](#_Toc408733326)

[2.12. Representational state transfer (REST): 12](#_Toc408733327)

[2.13. Object databases: 13](#_Toc408733328)

[2.13.1. ObjectDB: 13](#_Toc408733329)

[2.13.2. Java Persistence API (JPA): 13](#_Toc408733330)

[3. Analysis 14](#_Toc408733331)

[3.1. Functional requirement: 14](#_Toc408733332)

[3.2. Non-Functional requirement: 15](#_Toc408733333)

[3.3. Conceptual model: 17](#_Toc408733334)

[3.4. User stories: 19](#_Toc408733335)

[3.4.1. Story mapping: 20](#_Toc408733336)

[3.5. UML: 21](#_Toc408733337)

[3.5.1. Component diagram: 21](#_Toc408733338)

[3.5.2. Package diagram: 22](#_Toc408733339)

[3.5.3. Class diagram 23](#_Toc408733340)

[3.5.4. Use Case Diagram 26](#_Toc408733341)

[3.6. Data base design 38](#_Toc408733342)

[4. Implementation 39](#_Toc408733343)

[4.1. User experience (UX) 39](#_Toc408733344)

[4.2. User interface (UI) 39](#_Toc408733345)

[4.2.1. Graphical user interface (GUI) 40](#_Toc408733346)

[4.3. Implementation of (Kites administration panel – JORY) 41](#_Toc408733347)

[4.4. Implementation of (Kites client application-YASMIN) 44](#_Toc408733348)

[5. Conclusion 51](#_Toc408733349)

[5.1. Conclusion 51](#_Toc408733350)

[5.2. Recommendations 52](#_Toc408733351)

[5.3. References 53](#_Toc408733352)

Abbreviations

|  |  |
| --- | --- |
|  |  |
| TLS | Transport Layer Security |
| SSL | Secure Sockets Layer |
| AES | Advanced Encryption Standard |
| RSA | (Rivest, Shamir, Adleman) Cryptosystem |
| JPA | Java Persistence API |
| EJB | Enterprise Java Beans |
| REST | Representational State Transfer |
| JAX-RS | Java API for RESTful Web Services |
| JAXB | Java Architecture for XML Binding |
| SHA-256 | Secure Hash Algorithm (256 bit Digest size) |
| JSON | JavaScript Object Notation |
| XML | Extensible Markup Language |
| OODBMS | Object Oriented Data Base Management System |
| UI | User Interface |
| UX | User Experience |
| MVC | Model View Controller |
| FXML | JavaFX XML |
| XP | Extreme programming |
| CBSE | Component Based Software Engineering |
| IoT | Internet of Things |

1. Introduction:

In online marketing, a shopping cart is a piece of e-commerce or m-commerce software on a web server that allows visitors to an online store to select items for eventual purchase, The software allows online shopping customers to accumulate a list of items for purchase, described metaphorically as “placing items in the shopping cart” or “add to cart”. Upon checkout, the software typically calculates a total for the order, including shipping and handling charges and the associated taxes, as applicable.

Mobile Commerce (M-Commerce) primary goal is to simplify online shopping and make it quick and secured. Effective software helps in processing the order placement much quicker, safer, and easier.

According to BI Intelligence in January 2013, 29% of mobile users have now made a purchase with their phones. Walmart estimated that 40% of all visits to their internet shopping site in December 2012 were from a mobile device[[1]](#footnote-1).

According to technology research firm, Gartner, mobile commerce will see a 44 percent increase globally, reaching $235 billion this year. These increases will only continue, forecasting a $721 billion market with 450 million mobile commerce users by 2017. This figure includes transactions such as bill payments, money transfers and consumer payments[[2]](#footnote-2).

This project aims to create complete M-Commerce shopping cart software consisting of the following components:

* Android application (Shopping cart client application used by consumers).
* Desktop application (Shopping cart Administration panel used by owners).
* Web service (To deliver content to the android application).
  1. Problems with traditional commerce:
* No way of delivering electronic commerce capabilities directly into the consumer’s hand.
* Individuals are involved in all stages of business transactions.
* Manual processing of traditional business transactions.
* Much time is wasted by individuals.
* Time and place restriction.
  1. Objectives:
* The delivery of electronic commerce capabilities directly into the consumer’s hand, anywhere, via wireless technology.
* Automated processing of business transactions.
* Minimal individual involvement in business transactions.
* Provide merchandising trends - such as mobile catalogs and coupons
  1. Methodology:

Agile software development, extreme programming (XP) approach is used throughout the development.

* 1. Tools
* Eclipse as an Integrated Development Environment (IDE).
* Enterprise Architect as a UML modeling tool.
* Wireframe Sketcher as a wireframe designing tool.

1. Theoretical frame
   1. Electronic mobile Commerce:

M-commerce or mobile e-commerce is a term for any type of business, or commercial transaction, which involves the transfer of information across the Internet. It covers a range of different types of businesses, from consumer based retail stores, through auction or music sites, to business exchanges trading goods and services between corporations. It is currently one of the most important aspects of the Internet to emerge.

Unlike traditional commerce that is carried out physically with effort of a person to go & get products, M-commerce has made it easier for human to reduce physical work and to save time. M-Commerce which was started no so long ago has taken a great leap in the world of technology, but the fact that has hindered the growth of m-commerce and e-commerce in general is security. Security is the challenge facing m-commerce today & there is still a lot of advancement made in the field of security.

The main advantage of m-commerce over traditional commerce is the user can browse online shops, compare prices and order merchandise from anywhere and at any time.

* 1. Shopping cart:

A shopping cart is a piece of software that acts as an online store's catalog and ordering process. Typically, a shopping cart is the interface between a company's online store and its deeper infrastructure. Contrary to popular belief among merchants, an online shopping cart does not process credit card payments. Instead, shopping cart software serves three other main purposes. First, it allows merchants to setup and manage online sales processes such as adding products, inventory management, order fulfillment and customer data collection. Second, it acts as the ordering interface for the customer allowing him/her to add/remove products (items) from the shopping cart and checkout. And lastly, it communicates the payment information to the payment gateway which encrypts the data and sends it to the banks for authorization if applicable. Shopping carts come in two forms: (1) Self-Hosted Software – This type of shopping cart is a dedicated piece of software that is hosted on the retailer’s own server. There are several varieties of licensed carts that range from open source free carts to proprietary packages that can cost thousands of dollars. (2) 3rd Party Hosted Software – Also known as an Application Service Provider (ASP), this type is shopping cart is hosted on the ASP’s secured servers and integrated with the retailers website using code snippets, JavaScript and/or API connections . Merchants who choose to host their own shopping carts are also responsible for securing their server and software against data theft which is a costly ongoing task. Generally, self-hosted shopping carts are only advisable for mid-to-large online retailers that have someone dedicated to maintaining the security of the online store[[3]](#footnote-3).

* 1. Advantages of M-Commerce:

M-commerce obviously has numerous advantages over traditional commerce, the number one advantage being convenience of use. The user can browse online stores, compare prices and order merchandise sitting at home or anywhere they like. According to Forrester, 37 percent of online consumers use customer service more from Web retailers than traditional retailers because of its potential for ease of use and quick response times. "Once excellent proactive customer service is in place, merchants must build their brands around the promise of a satisfying experience," said Forrester's Christopher Kelley. "This means not only advertising a call center but also bragging about speedy response times and knowledgeable service reps". Dell, for example has adopted the same approach to selling their products both online and offline. This however, has proven beneficial for the company, since it helped them cut down on warehousing costs. One way in which the company has encouraged online ordering is by offering rebates on the products that are bought online. Other companies should follow Dell's example if they want to succeed in the online world.

* 1. Present Challenges Facing E-commerce and M-Commerce:

Speaking of obstacles, there are a lot of them that need to be uprooted before m-commerce can compete with traditional commerce. The biggest obstacle in the course of advancement of m-commerce is that the consumer's senses are limited to seeing and hearing the product. The second largest problem that e-commerce has been facing over the past few years is that of security. Traditional buyers and sellers are still paranoid about conducting business online. According to Hal Loevy, vice president of Global Marketing and Partnerships for SGSonSITE, "Despite all the noise about e-commerce, which is significant, companies still have to keep their old business practices: Can I trust who I am buying from? Who am I doing business with? What is their trading history? Am I obeying the law? Will I receive the goods as specified on screen and who do I approach if I have a problem?” According to emarketer.com, "70% of US consumers are concerned about online security; this discourages consumers from using credit cards to shop online (Payment One)".

* 1. Proposed project:

Kites M-commerce shopping cart, is a software Developed using state of the art technologies, based on various component, utilizing advanced programming concepts and techniques such as (RESTful web services, JPA, EJB, …) built on top of Java EE, Java SE and android platforms. Extreme programming is used throughout the development of the project.

* 1. Agile software development:

Agile software development is a group of software development methods in which requirements and solutions evolve through collaboration between self-organizing, cross-functional teams. It promotes adaptive planning, evolutionary development, early delivery, continuous improvement and encourages rapid and flexible response to change. It is a conceptual framework that focuses on delivering working software with the minimum amount of work[[4]](#footnote-4).

* + 1. Agile principles:

The Agile Manifesto is based on twelve principles:

* Customer satisfaction by rapid delivery of useful software
* Welcome changing requirements, even late in development
* Working software is delivered frequently (weeks rather than months)
* Close, daily cooperation between business people and developers
* Projects are built around motivated individuals, who should be trusted
* Face-to-face conversation is the best form of communication (co-location)
* Working software is the principal measure of progress
* Sustainable development, able to maintain a constant pace
* Continuous attention to technical excellence and good design
* Simplicity the art of maximizing the amount of work not done is essential
* Self-organizing teams
* Regular adaptation to changing circumstances
  + 1. Agile methods:

Well-known agile software development methods and/or process frameworks include:

* Adaptive Software Development (ASD)
* Agile Modeling
* Agile Unified Process (AUP)
* Crystal Methods (Crystal Clear)
* Disciplined Agile Delivery
* Dynamic Systems Development Method (DSDM)
* Extreme Programming (XP)
* Feature Driven Development (FDD)
* Lean software development
* Kanban (development)
* Scrum
* Scrum-ban
  + 1. Agile practices:

Agile development is supported by a bundle of concrete practices suggested by the agile methods, covering areas like requirements, design, modeling, coding, testing, project management, process, quality, etc. Some notable agile practices include:

* Acceptance test-driven development (ATDD)
* Agile Modeling
* Backlogs (Product and Sprint)
* Behavior-driven development (BDD)
* Cross-functional team
* Continuous integration (CI)
* Domain-driven design (DDD)
* Information radiators (Scrum board, Kanban board, Task board, Burndown chart)
* Iterative and incremental development (IID)
* Pair programming
* Planning poker
* Refactoring
* Scrum meetings (Sprint planning, Daily scrum, Sprint review and retrospective)
* Test-driven development (TDD)
* Agile testing
* Time boxing
* Use case
* User story
* Story-driven modeling
* Velocity tracking
  1. Extreme programming (XP):

Extreme programming (XP) is a software development methodology which is intended to improve software quality and responsiveness to changing customer requirements. As a type of agile software development, it advocates frequent "releases" in short development cycles, which is intended to improve productivity and introduce checkpoints at which new customer requirements can be adopted[[5]](#footnote-5).

Other elements of extreme programming include: programming in pairs or doing extensive code review, unit testing of all code, avoiding programming of features until they are actually needed, a flat management structure, simplicity and clarity in code, expecting changes in the customer's requirements as time passes and the problem is better understood, and frequent communication with the customer and among programmers. The methodology takes its name from the idea that the beneficial elements of traditional software engineering practices are taken to "extreme" levels. As an example, Code reviews are considered a beneficial practice; taken to the extreme, code can be reviewed continuously, i.e. the practice of Pair programming[[6]](#footnote-6).

XP attempts to reduce the cost of changes in requirements by having multiple short development cycles, rather than a long one. In this doctrine, changes are a natural, inescapable and desirable aspect of software-development projects, and should be planned for, instead of attempting to define a stable set of requirements.

Extreme programming also introduces a number of basic values, principles and practices on top of the agile programming framework.

* + 1. XP Activities:

XP describes four basic activities that are performed within the software development process: coding, testing, listening, and designing. Each of those activities is described below.

* Coding

The advocates of XP argue that the only truly important product of the system development process is code – software instructions that a computer can interpret. Without code, there is no working product.

Coding can also be used to figure out the most suitable solution. Coding can also help to communicate thoughts about programming problems. A programmer dealing with a complex programming problem, or finding it hard to explain the solution to fellow programmers, might code it in a simplified manner and use the code to demonstrate what he or she means. Code, say the proponents of this position, is always clear and concise and cannot be interpreted in more than one way. Other programmers can give feedback on this code by also coding their thoughts.

* Testing

Extreme programming's approach is that if a little testing can eliminate a few flaws, a lot of testing can eliminate many more flaws.

Unit tests determine whether a given feature works as intended. A programmer writes as many automated tests as they can think of that might "break" the code; if all tests run successfully, then the coding is complete. Every piece of code that is written is tested before moving on to the next feature.

Acceptance tests verify that the requirements as understood by the programmers satisfy the customer's actual requirements.

System-wide integration testing was encouraged, initially, as a daily end-of-day activity, for early detection of incompatible interfaces, to reconnect before the separate sections diverged widely from coherent functionality. However, system-wide integration testing has been reduced, to weekly, or less often, depending on the stability of the overall interfaces in the system.

* Listening

Programmers must listen to what the customers need the system to do, what "business logic" is needed. They must understand these needs well enough to give the customer feedback about the technical aspects of how the problem might be solved, or cannot be solved. Communication between the customer and programmer is further addressed in the planning game.

* Designing

From the point of view of simplicity, of course one could say that system development doesn't need more than coding, testing and listening. If those activities are performed well, the result should always be a system that works. In practice, this will not work. One can come a long way without designing but at a given time one will get stuck. The system becomes too complex and the dependencies within the system cease to be clear. One can avoid this by creating a design structure that organizes the logic in the system. Good design will avoid lots of dependencies within a system; this means that changing one part of the system will not affect other parts of the system.

* + 1. XP Practices:

Extreme programming has been described as having 12 practices, grouped into four areas:

* Fine-scale feedback
  + Pair programming
  + Planning game
  + Test-driven development
  + Whole team
* Continuous process
  + Continuous integration
  + Refactoring or design improvement
  + Small releases
* Shared understanding
  + Coding standards
  + Collective code ownership
  + Simple design
  + System metaphor
* Programmer welfare
  + Sustainable pace
  1. Component-based software engineering (CBSE):

CBSE also known as component-based development (CBD) is a branch of software engineering that emphasizes the separation of concerns in respect of the wide-ranging functionality available throughout a given software system. It is a reuse-based approach to defining, implementing and composing loosely coupled independent components into systems. This practice aims to bring about an equally wide-ranging degree of benefits in both the short-term and the long-term for the software itself and for organizations that sponsor such software[[7]](#footnote-7).

A computer running several software components is often called an application server. This combination of application servers and software components is usually called distributed computing.

Enterprise JavaBeans, Java EE and RESTful Web services are some technologies that are built on the concept of CBSE which are used in this project.

* 1. Enterprise JavaBeans (EJB):

EJB is a managed, server-side component architecture for modular construction of enterprise applications that encapsulates the business logic of an application.

The EJB specification details how an application server provides the following responsibilities

* Transaction processing.
* Integration with the persistence services offered by the Java Persistence API (JPA).
* Concurrency control.
* Event-driven programming using Java Message Service and Java EE Connector Architecture.
* Asynchronous method invocation.
* Job scheduling.
* Naming and directory services (JNDI).
* Interprocess Communication using RMI-IIOP and Web services.
* Security (JCE and JAAS).
* Deployment of software components in an application server.
  1. Java Platform, Enterprise Edition (Java EE):

Java EE is Oracle's enterprise Java computing platform. The platform provides an API and runtime environment for developing and running enterprise software, including network and web services, and other large-scale, multi-tiered, scalable, reliable, and secure network applications. Java EE extends the Java Platform, Standard Edition (Java SE) providing an API for object-relational mapping, distributed and multi-tier architectures, and web services. The platform incorporates a design based largely on modular components running on an application server. Software for Java EE is primarily developed in the Java programming language[[8]](#footnote-8).

* 1. Web service:

A Web service is a method of communication between two electronic devices over a network. It is a software function provided at a network address over the web with the service always on.

The W3C defines a Web service generally as:

a software system designed to support interoperable machine-to-machine interaction over a network.

* 1. Representational state transfer (REST):

REST is an architectural style consisting of a coordinated set of architectural constraints applied to components, connectors, and data elements, within a distributed hypermedia system. REST ignores the details of component implementation and protocol syntax in order to focus on the roles of components, the constraints upon their interaction with other components, and their interpretation of significant data elements.

The REST architectural style is also applied to the development of web services as an alternative to other distributed-computing specifications such as SOAP. One can characterize web services as "RESTful" if they conform to the constraints described in the architectural constraints section.

HTTP based RESTful web service APIs are defined with these aspects

* Base URI, such as (http://example.com/resources/).
* An Internet media type for the data. This is often JSON but can be any other valid Internet media type (e.g. XML, Atom, microformats, images, etc.).
* Standard HTTP methods (e.g., GET, PUT, POST, or DELETE).
* Hypertext links to reference state.
* Hypertext links to reference related resources.
  1. Object databases:

An object database (also object-oriented database management system (OODBMS)) is a database management system in which information is represented in the form of objects as used in object-oriented programming. Object databases are different from relational databases which are table-oriented. Object-relational databases are a hybrid of both approaches.

* + 1. ObjectDB:

ObjectDB is an object database for Java. That support two standard Java APIs - (JPA and JDO).

* + 1. Java Persistence API (JPA):

The JPA is a Java programming language application programming interface specification that describes the management of relational data in applications using Java Platform, Standard Edition and Java Platform, Enterprise Edition.

1. Analysis
   1. Functional requirement:

* Security

All password stored within the database are hashed using SHA-256, thus having the same long no matter how long is the password the customer or the user has provided.

All transactions data between the web service and the client application are highly encrypted using advanced cryptographic techniques and algorithms (AES-256 with RSA-2048).

However the web service does not use TLS/SSL for secure communication between it and the client application.

* Documents formatting

Messages produced by the web service are formatted in JSON and XML.

Internationalization is supported using Properties and XML files for different languages and different locales.

* Concurrent users support

Supports for multiple concurrent users and operations are mandatory.

* UI

FXML and XML files are used to separate business logic code from UI and reuse some components whenever needed.

* OODBMS

Any persistent information is stored in an ObjectDB database. And JPA is used for data accessing to improve performance, portability and scalability.

* Interface for regular customers

Regular customers are allowed to do the following things

* 1. Browse products in catalogs by categories, and also search for a specific product by name.
  2. Add products to a virtual shopping cart. They can also edit its content or drop products they have already added to the cart.
  3. To do the above things customers do not need to login or register. If customers login they will be able to:
  4. View all previous orders and current state of orders.
  5. When customers proceed to checkout they are requested to login to an existing account or register.
  6. After customers proceed to checkout a confirmation dialog is shown so they can again review their shopping cart, edit it and then purchase the products.

Administrative interface

The system provides an administrative interface for site administrators. Administrators are provided with the following functionality

* 1. Add, edit or delete categories. (When a category is deleted all the products for that category will be removed from the database).
  2. Add, edit or delete products inside categories.
  3. Browse all orders and change their status.
  4. Backup and restore the database.
  5. Non-Functional requirement:
* Model View Controller(MVC)

Different functionality is separated. This project uses a Model, View and Controller design pattern.

The model encapsulates functionality and data of all entities and data models, Consists of a number of Dispatcher classes.

The view retrieves data from the model and presents it, Consists of FXML and XML layouts for both client and administration application respectively.

The controller receives input and translates it into requests to the model or the view based on certain application logic.

* System up time

System must be available 99.99% of time during business hours.

* Technology used
  + The project is implemented in Java and uses various APIs: JAX-RS RESTful web services, JPA persistence layer, JavaFX rich client interfaces and EJB for business logic.
  + Oracle’s glassfish open source server as the Applications server.
  + The system persist data to an OODBMS database (objectDB database).
  + Eclipse platform is used throughout the development and implementation.
* Documentation

The system should be well documented. And contains detailed instructions on how to use it.

* Localization & Internationalization

The system should be internationalized by supporting localization using standard methods.

* Recovery

The system must provide methods for backing-up and restoring data

* 1. Conceptual model:

Customer

Payment

Order

Shopping cart

Product (Item)

Catalog

3

1

2

5

5

4

4

3

* Customer:

A customer (sometimes known as a client, buyer, or purchaser) is the recipient of a good, service, product, or idea, obtained from a seller, vendor, or supplier for a monetary or other valuable consideration[[9]](#footnote-9). Customers are generally categorized into two types:

* + An intermediate customer or trade customer (more informally: "the trade") who is a dealer that purchases goods for re-sale.
  + An ultimate customer who does not in turn re-sell the things bought but either passes them to the consumer or actually is the consumer.
* Catalog:

Catalog or catalogue is a publication containing a list of general merchandise from a company. Companies who publish and operate mail order catalogues are referred to as cataloguers within the industry.

* Product:

In economics and commerce, products belong to a broader category of goods.

In economics, a good is a material that satisfies human wants and provides utility.

In marketing, a product is anything that can be offered to a market that might satisfy a want or need.\* In retailing, products are called merchandise.

A product can be classified as tangible or intangible. A tangible product is a physical object that can be perceived by touch such as a building, vehicle, gadget, or clothing. An intangible product is a product that can only be perceived indirectly such as an insurance policy.

* Shopping cart:

A shopping cart is a piece of software that acts as an online store's catalog and ordering process. It allows visitors to an online store to select items for eventual purchase.

* Order:

In business or commerce, an order is a stated intention, either spoken or written, to engage in a commercial transaction for specific products or services. From a buyer's point of view it expresses the intention to buy and is called a purchase order. From a seller's point of view it expresses the intention to sell and is referred to as a sales order. When the purchase order of the buyer and the sales order of the seller agree, the orders become a contract between the buyer and seller.

* Payment:

A payment is the transfer of an item of value from one party (such as a person or company) to another in exchange for the provision of goods, services or both, or to fulfill a legal obligation.

There are two types of payment methods; exchanging and provisioning. Exchanging is to change coin, money and banknote in terms of the price. Provisioning is to transfer money from one account to another. In this method, a third party must be involved. Credit card, debit card, Cheque (Check in US), money transfers, and recurring cash or ACH (Automated Clearing House) disbursements are all electronic payments methods. Electronic payments technologies are magnetic stripe card, smartcard, contactless card and mobile handset. Mobile handset based payments are called mobile payments.

* 1. User stories:

Stories represent self-contained, individual elements of the project. They tend to correspond to individual features.

Stories are customer-centric, describing the results in terms of business results. They’re not implementation details, nor are they full requirements specifications[[10]](#footnote-10).

* 1. As a customer,  
     I want to be able to browse products organized in categories  
     so that I can find my needs quickly.
  2. As a customer,  
     I want to be able to see product details (name, image, price) before I add it to my shopping cart.
  3. As a customer,  
     I want to be able to see and manipulate products (items) inside my shopping cart  
     so that I have more flexible purchase process.
  4. As a customer,  
     I want to be able to create an account using my desired username and password  
     so that I have more freedom over my account.
  5. As a customer,  
     I want to be able to search for a particular product by name,  
     so that I can find what I want quickly.
  6. As a customer,  
     I want to be able to browse product offline,  
     so that I can spare some extra time.
  7. As a customer,  
     I want to be able to change my password online.
  8. As a shop owner,  
     I want to be able to manage(add, remove, edit) product and organize them in categories   
     so that I can expand my business.
  9. As a shop owner,  
     I want to be able to check new orders  
     so that I can process them.
  10. As a shop owner,  
      I want to be able to back-up my data and restore it in a fast, efficient and reliable ways  
      so that I can have the minimum down time.
  11. As a shop owner,  
      I want to see detailed statistics about my business.
  12. As a shop owner,  
      I want to limit access to my application for authorized users only  
      so that I can further protect my business.
      1. Story mapping:

Is a practice intended to provide a more structured approach to release planning; story mapping consists of ordering user stories along two independent dimensions. The "map" arranges user activities along the horizontal axis in rough order of priority. Down the vertical axis, it represents increasing sophistication of the implementation.\*

Given a story map so arranged, the first horizontal row represents a "walking skeleton", a barebones but usable version of the product. Working through successive rows fleshes out the product with additional functionality[[11]](#footnote-11).

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Releases |  | Time | | |  | Priority |
| 1 | Story (1) | Story (8) | Story (9) | H |
| 2 | Story (2) | Story (10) | Story (12) | M |
|  |  |  |
| 3 | Story (3) | Story (5) | Story (4) | L |
|  |  |  |
|  |  |  |

* 1. UML:

The Unified Modeling Language (UML) is a general-purpose modeling language in the field of software engineering, which is designed to provide a standard way to visualize the design of a system.

* + 1. Component diagram:

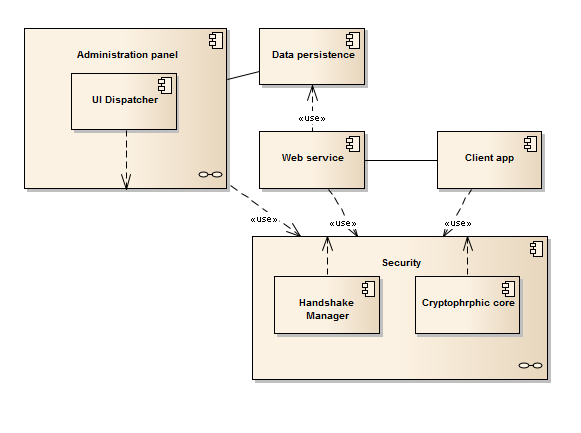
Component diagram depicts how components are wired together to form larger components and or software systems. They are used to illustrate the structure of arbitrarily complex systems.

Components are wired together by using an assembly connector to connect the required interface of one component with the provided interface of another component. This illustrates the service consumer - service provider relationship between the two components.

An assembly connector is a "connector between two components that defines that one component provides the services that another component requires. An assembly connector is a connector that is defined from a required interface or port to a provided interface or port”.

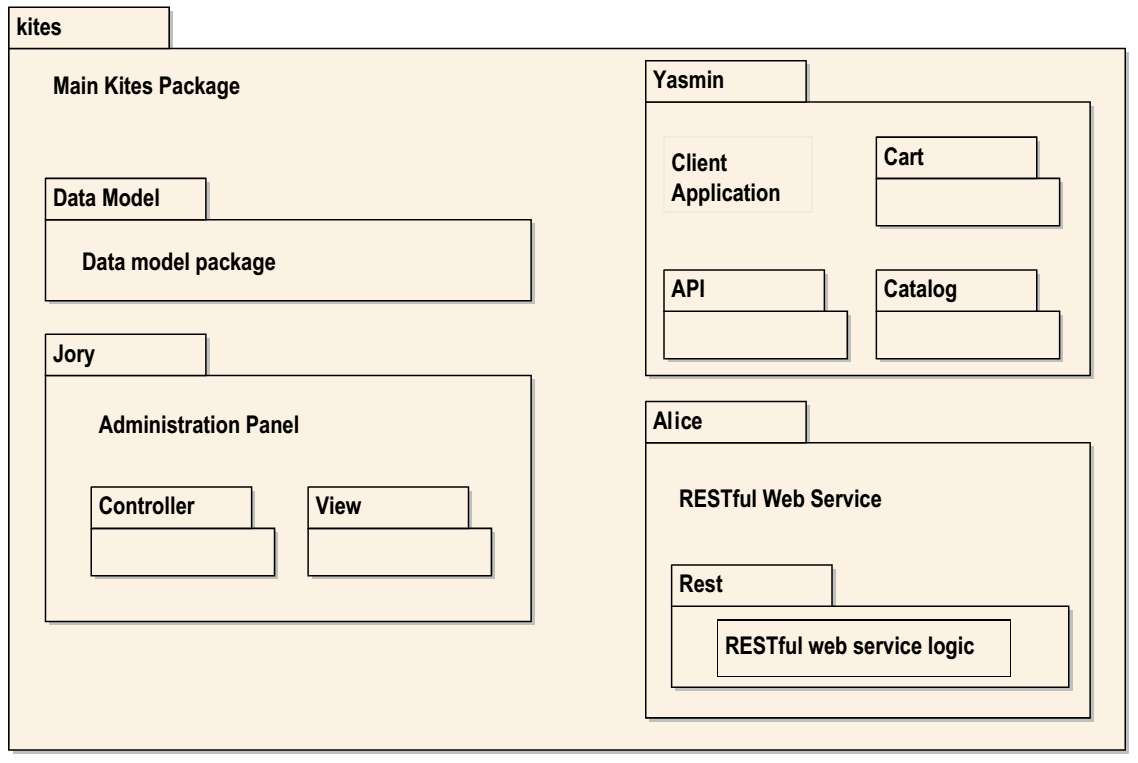
When using a component diagram to show the internal structure of a component, the provided and required interfaces of the encompassing component can delegate to the corresponding interfaces of the contained components.

A delegation connector is a "connector that links the external contract of a component (as specified by its ports) to the internal realization of that behavior by the component’s parts.



* + 1. Package diagram:

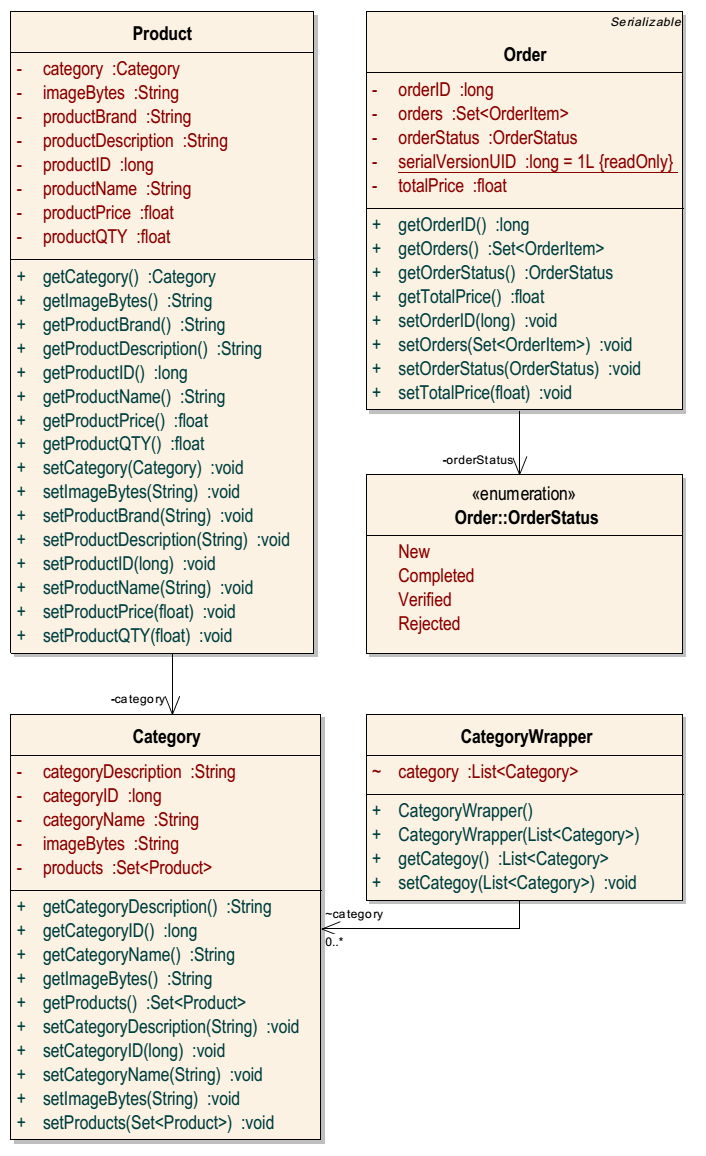
Package diagram depicts the dependencies between the packages that make up a model.

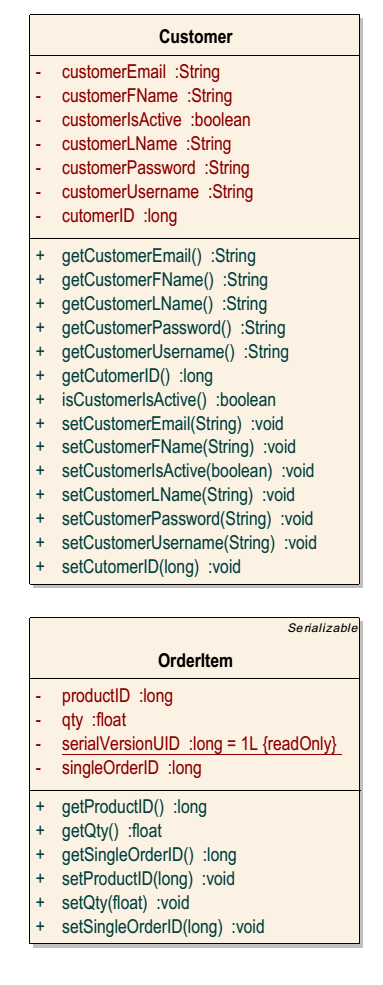


* + 1. Class diagram

Class diagram describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among objects.

The class diagram is the main building block of object oriented modeling. It is used both for general conceptual modeling of the systematics of the application, and for detailed modeling translating the models into programming code. Class diagrams can also be used for data modeling. The classes in a class diagram represent both the main objects, interactions in the application and the classes to be programmed.

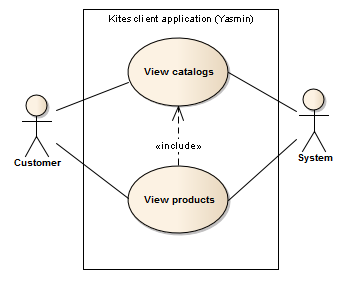
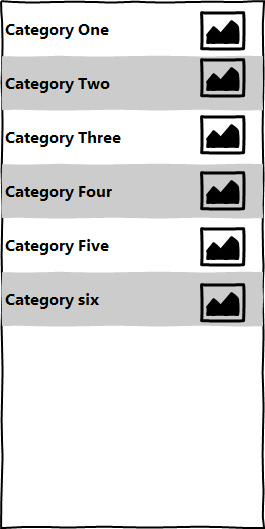
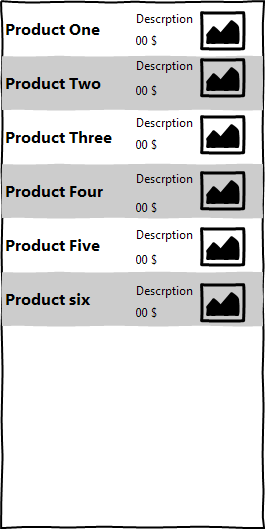
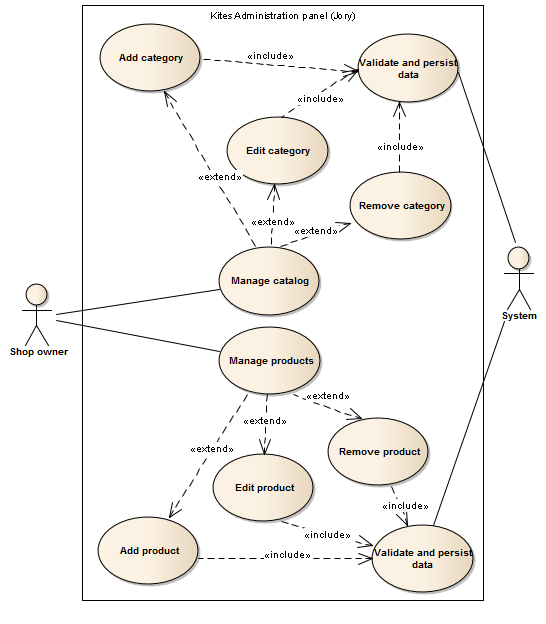
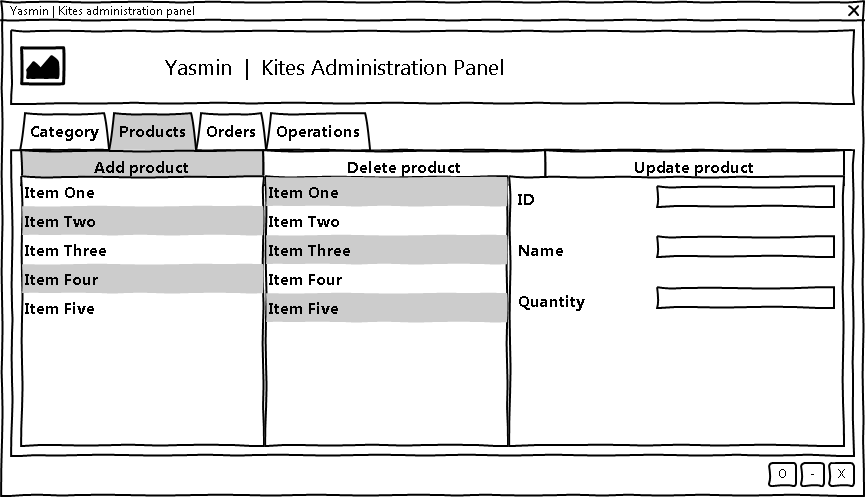
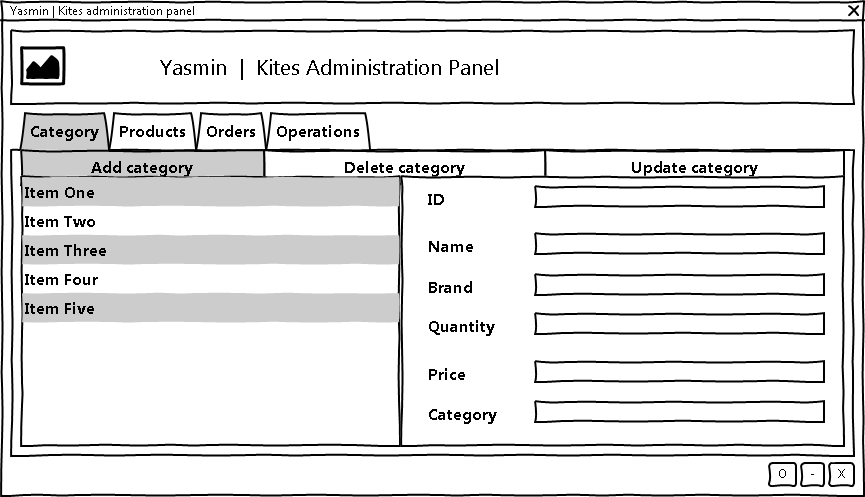
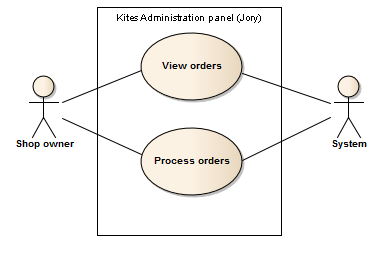
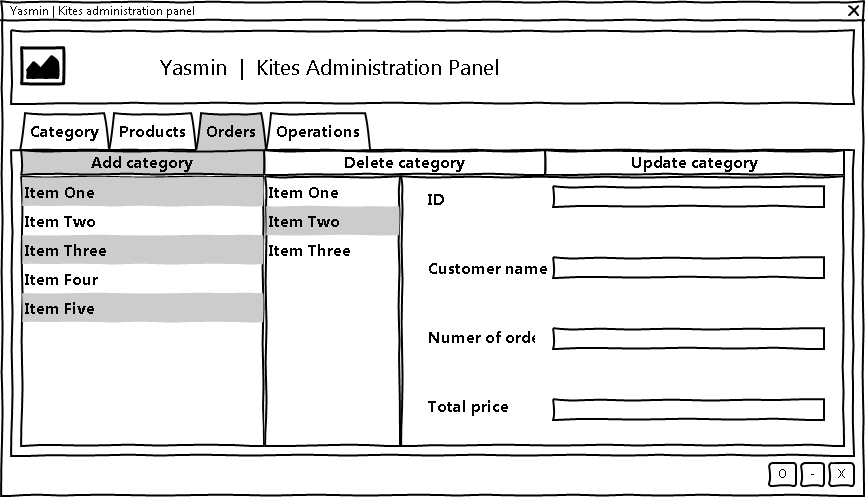
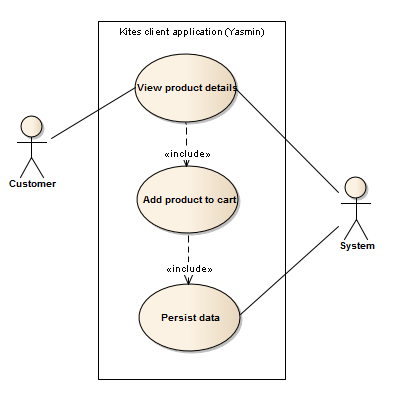
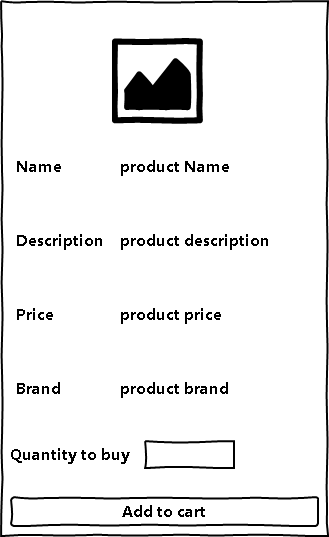
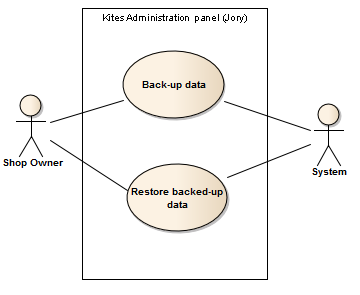
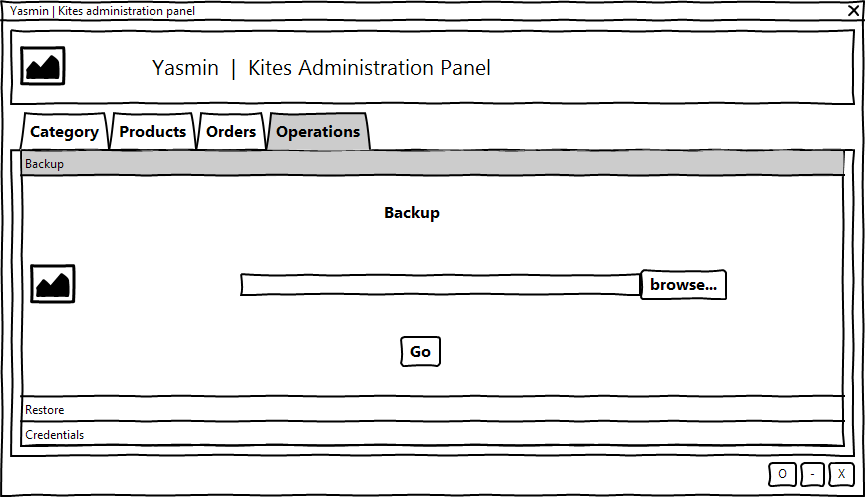
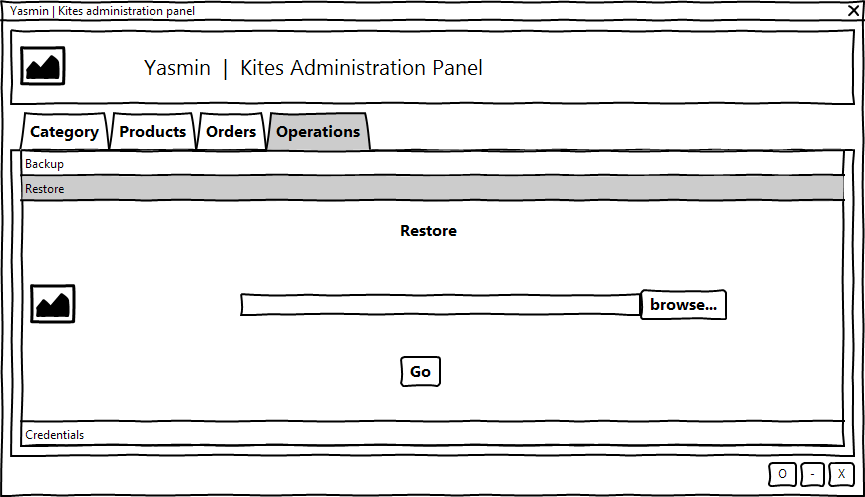
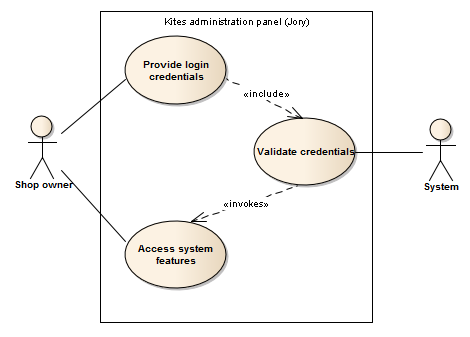
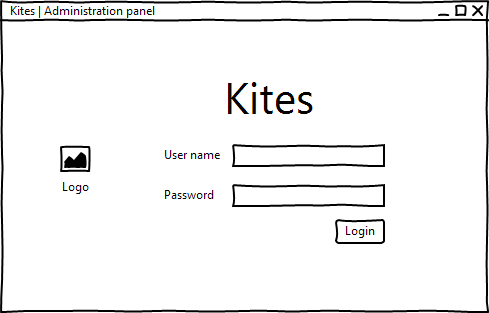
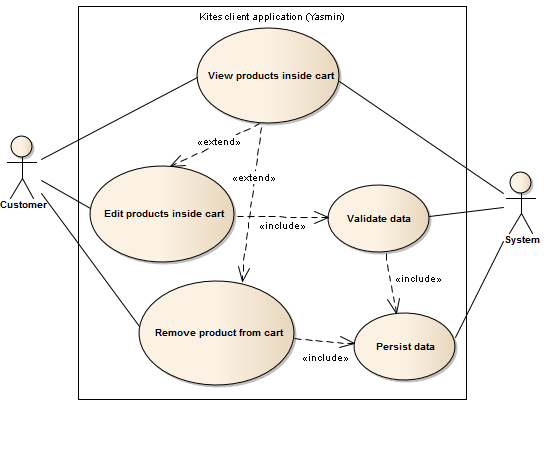
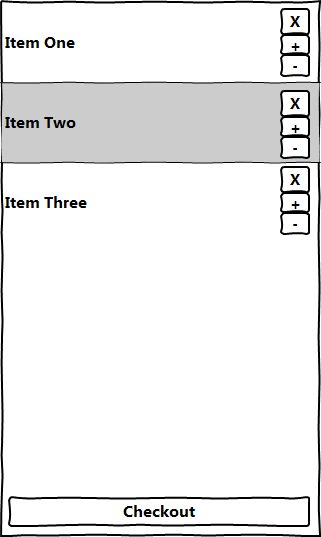
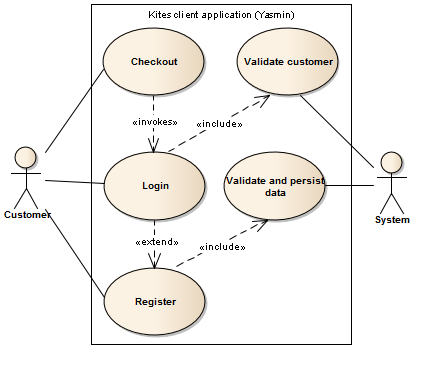
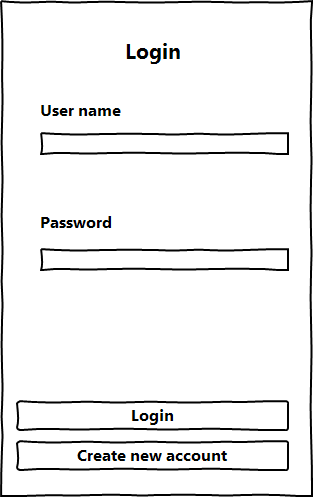
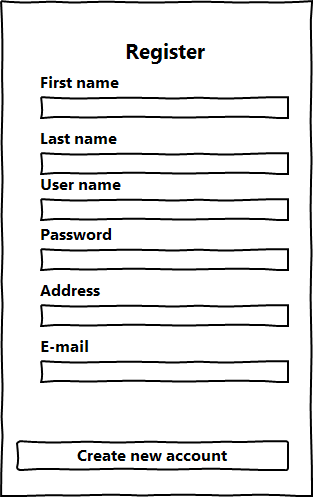
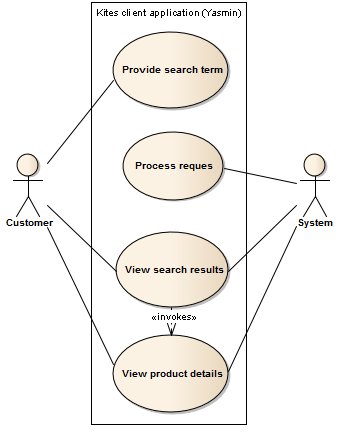
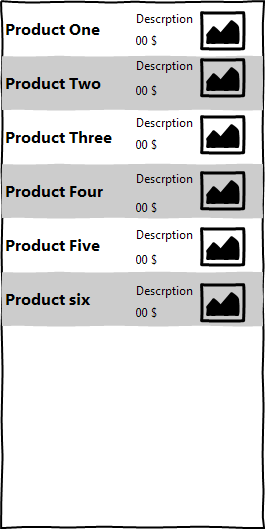




* + 1. Use Case Diagram

Use case diagram at its simplest is a representation of a user's interaction with the system and depicting the specifications of a use case. A use case diagram can portray the different types of users of a system and the case.

Following agile development we’ll describe each story with its own diagram according to the story map.

* Story (1)  
  The customer should be able to browse a digital catalog through the ‘Kites client application(YASMIN)’ containing categories organized in a list with products inside each category also organized in a list, and have freedom over navigating back and forth between the two lists.  
    
  Use case diagram  
  
* Conceptual wireframe  
    
    
  
* Story (8)  
  The shop owner should be able to manage(add, edit, remove) categories and products organized in lists with a dedicated button for each operation and a set of widgets to view the details of the currently selected item (category or product).
* Use case diagram  
  
* Conceptual wireframe  
    
  
* Story (9)  
  The shop owner should be able to see new orders with their details (products forming the order) in a list that is automatically gets updated to show new orders once they are made.
* Use case diagram  
    
  Conceptual wireframe  
  
* Story (2)  
  Upon selecting a product from the catalog the customer should be able to see all product information and have the freedom to add it to his/her cart or go back to the catalog.
* Use case diagram  
    
  Conceptual wireframe  
  
* Story (10)  
  Shop owner should have a screen where he/she can perform ‘Back-up’ or ‘Restore’ for his/her data specifying the directory to saving and retrieving data.  
    
  Use case diagram  
    
  Conceptual wireframe  
    
  
* Story (12)  
  Shop owner should only access ‘Kites administration panel (JORY)’ after providing credentials (username and password).  
    
  Use case diagram  
    
  Conceptual wireframe  
  
* Story (3)  
  Customer should be able to see the content of his/her cart at any time and be able to manipulate(edit, remove) items (products) inside it. And have the freedom to proceed to checkout or continue shopping.  
     
  Use case diagram  
    
  Conceptual wireframe  
  
* Story (4)  
  Upon checkout the customer is asked to provide his/her credentials in order to complete the checkout process, and have a way to create a new account if necessary.  
    
  Use case diagram  
    
  Conceptual wireframe  
    
  
* Story (5)  
  The customer should be able to search for a products by name and then choose one for detailed information about it.  
    
  Use case diagram  
    
  Conceptual wireframe  
  
  1. Data base design

|  |  |  |
| --- | --- | --- |
| Kites.model.User | | |
| Id | HashedPassword | Userame |
| Long | String | String |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Kites.model.Customer | | | | | | | |
| Id | isActive | Address | Email | Fname | Lname | Username | HashedPassword |
| long | boolean | String | String | String | String | String | String |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Kites.model.Category | | | | |
| Id | Name | Description | ImageBytes | Products |
| Long | String | String | String | Set<Product> |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Kites.model.Product | | | | | | | |
| Id | Name | Description | ImageBytes | Brand | Quantity | Price | Category |
| Long | String | String | String | String | Float | Float | Category |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Kites.model.Order | | | | | | |
| Id | CustomerID | TotalPrice | Address | OrderStatus | OrderType | Orders |
| Long | Long | Float | String | Enum | Enum | List<OrderItem> |

|  |  |  |
| --- | --- | --- |
| Kites.model.OrderItem | | |
| Id | ProductID | Quantity |
| Long | Long | Float |

1. Implementation

In computer science, an implementation is a realization of a technical specification or algorithm as a program, software component, or other computer system through computer programming and deployment[[12]](#footnote-12)

* 1. User experience (UX)

User Experience involves a person's behaviors, attitudes, and emotions about using a particular product, system or service.

User experience includes the practical, experiential, affective, meaningful and valuable aspects of human–computer interaction and product ownership. Additionally, it includes a person’s perceptions of system aspects such as utility, ease of use and efficiency. User experience may be considered subjective in nature to the degree that it is about individual perception and thought with respect to the system. User experience is dynamic as it is constantly modified over time due to changing usage circumstances and changes to individual systems as well as the wider usage context in which they can be found.

Many factors can influence a user's experience with a system. To address the variety, factors influencing user experience have been classified into three main categories: user's state and previous experience, system properties, and the usage context (situation). Studying typical users, contexts, interactions and resulting emotions help in designing the system[[13]](#footnote-13).

* 1. User interface (UI)

The user interface is the space where interactions between humans and machines occur. The goal of this interaction is to allow effective operation and control of the machine from the human end, whilst the machine simultaneously feeds back information that aids the operators decision making process[[14]](#footnote-14).

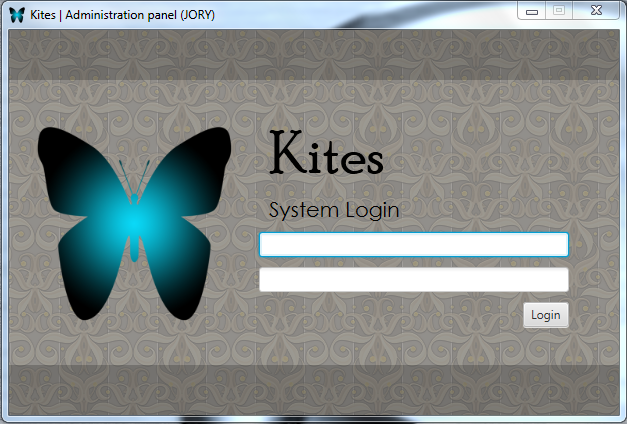
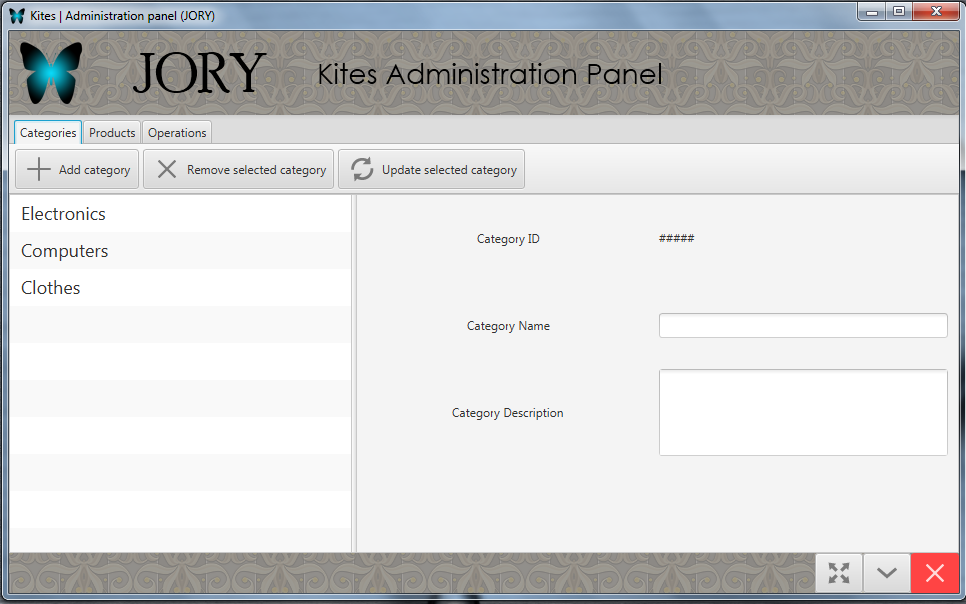
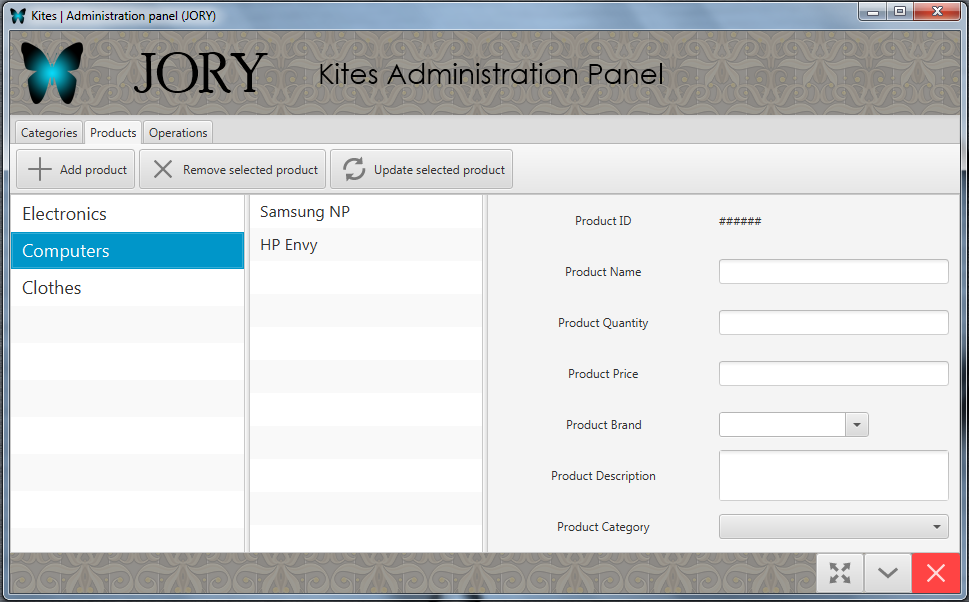
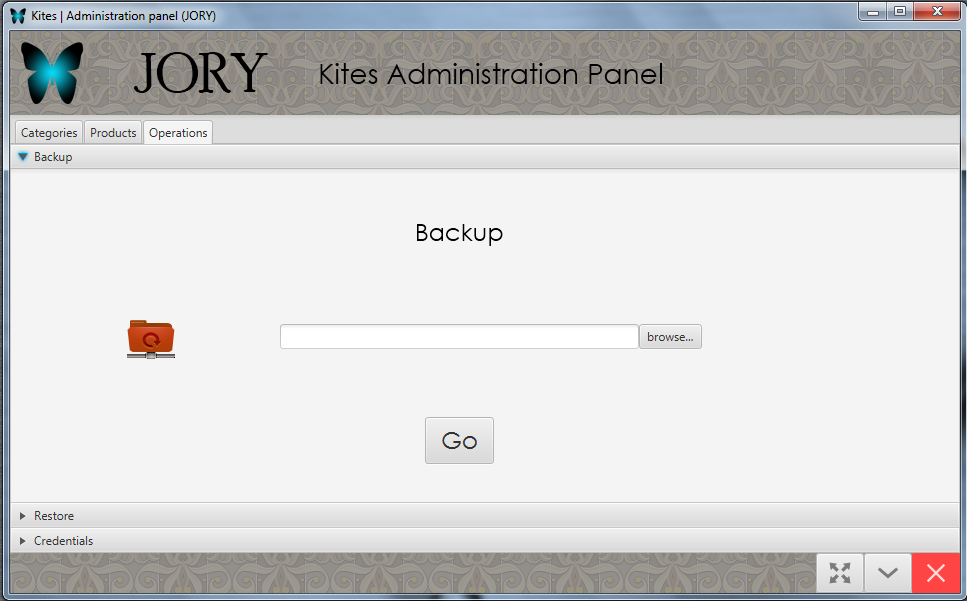
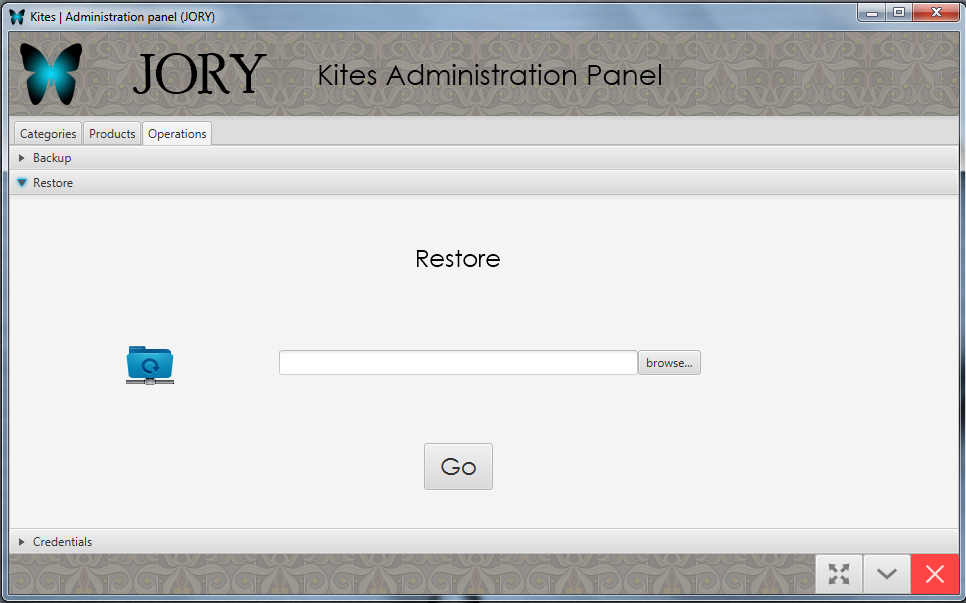
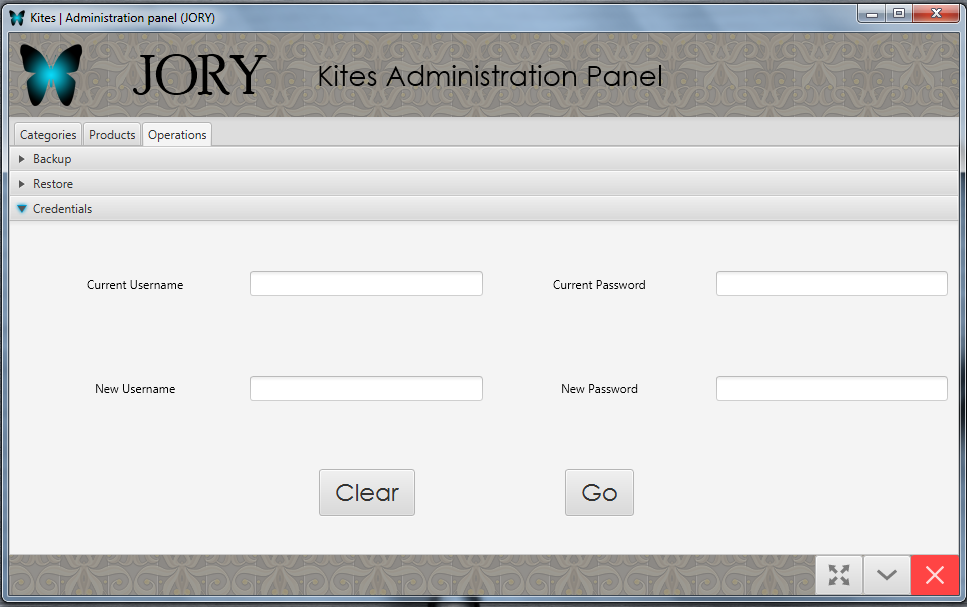
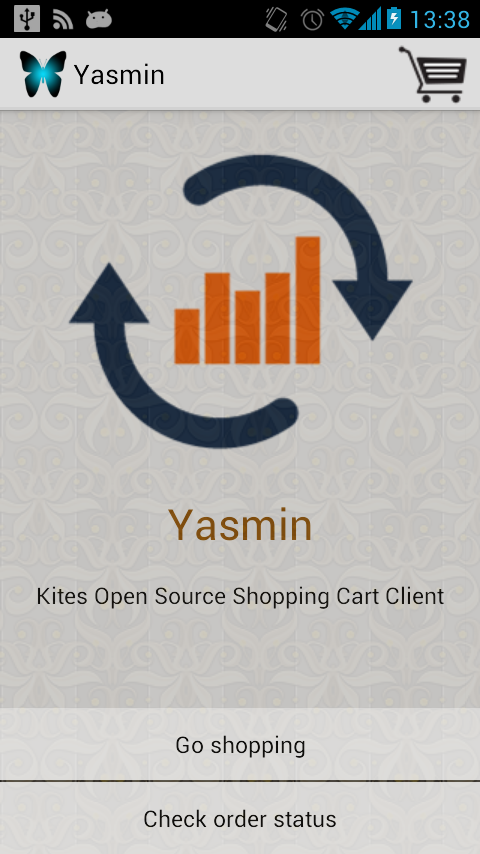
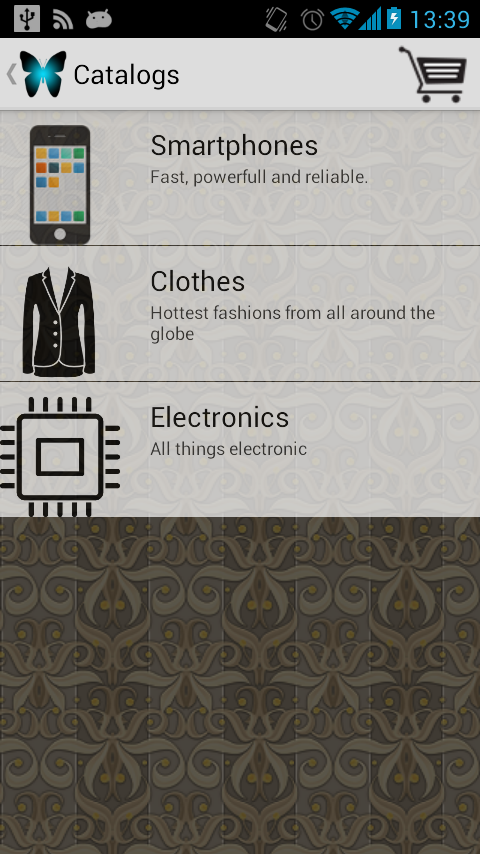
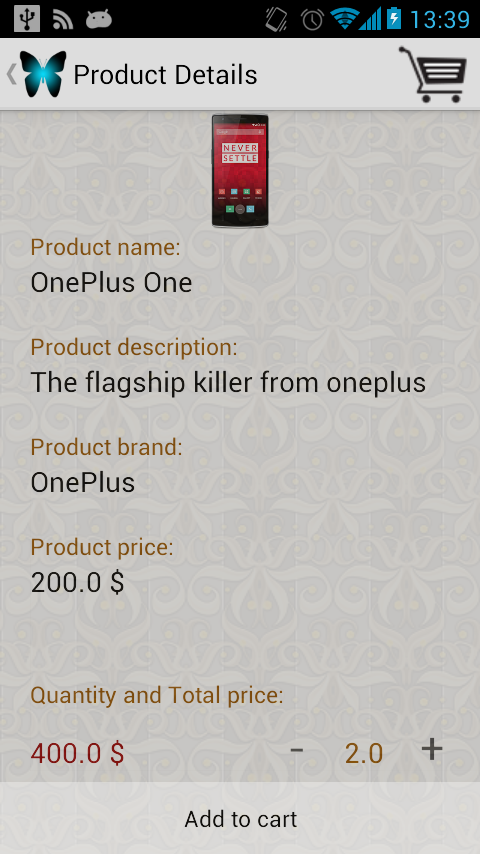
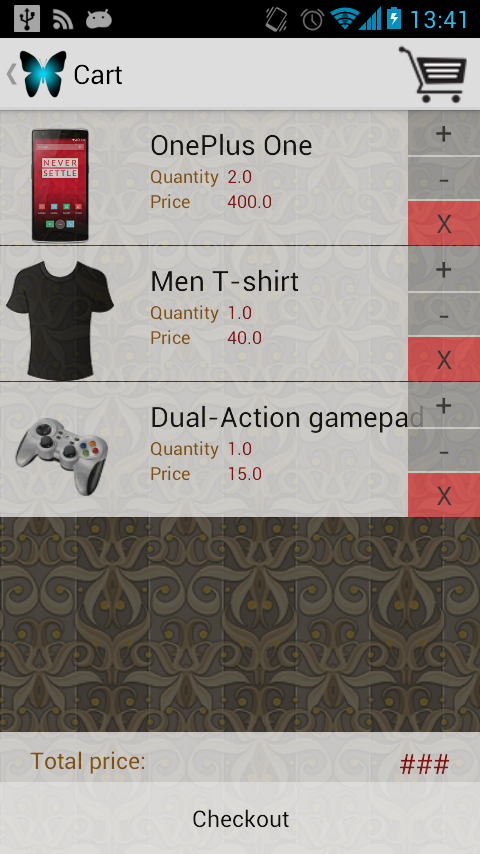
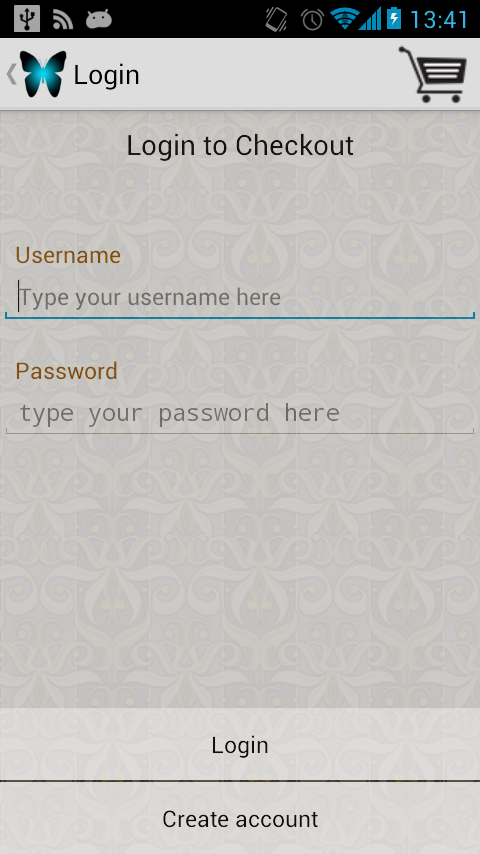
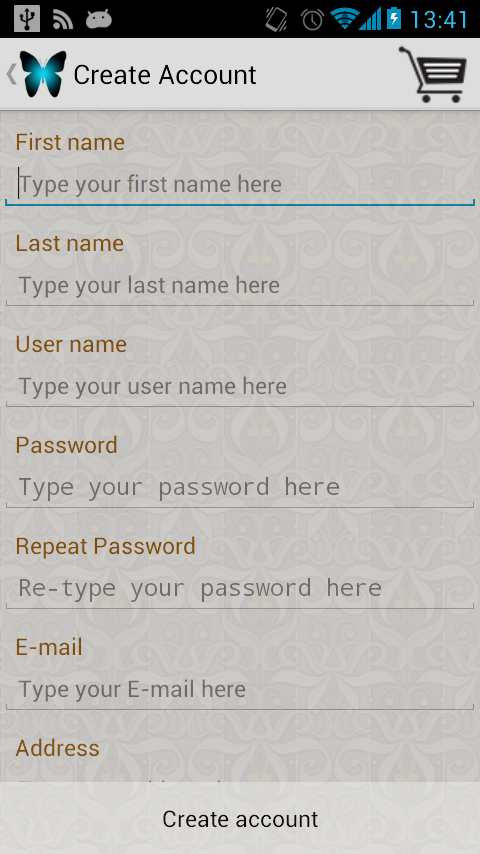
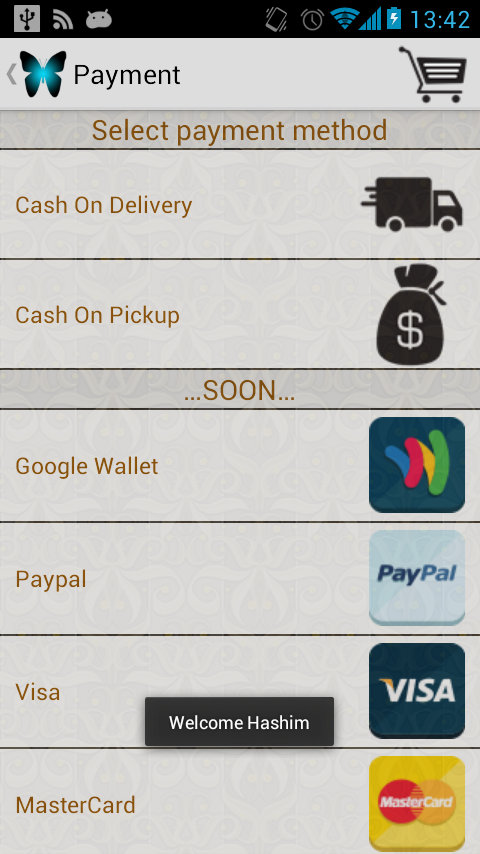
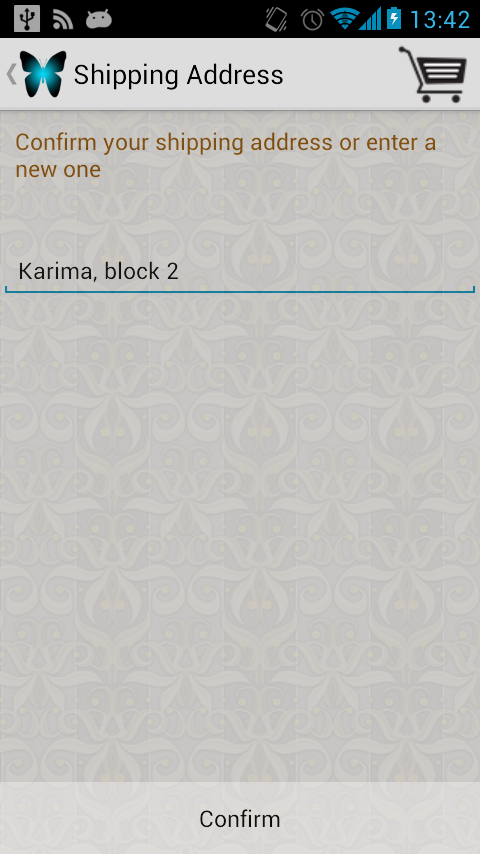
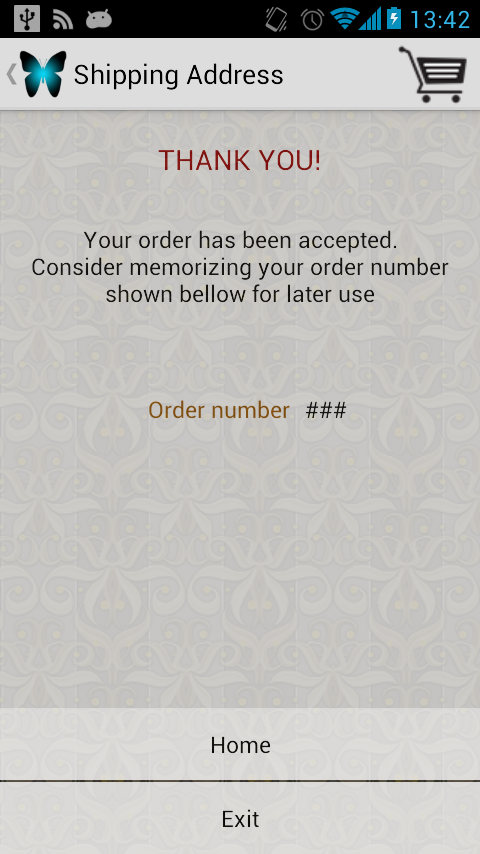
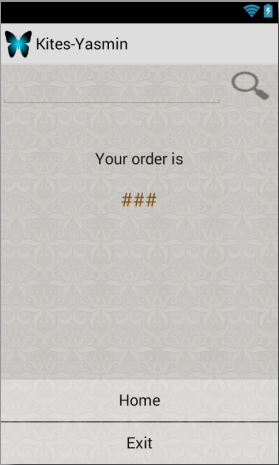
All great interfaces share eight qualities or characteristics:

* Clarity: The interface avoids ambiguity by making everything clear through language, flow, hierarchy and metaphors for visual elements.
* Concision: It’s easy to make the interface clear by over­-clarifying and labeling everything, but this leads to interface bloat, where there is just too much stuff on the screen at the same time. If too many things are on the screen, finding what you’re looking for is difficult, and so the interface becomes tedious to use. The real challenge in making a great interface is to make it concise and clear at the same time.
* Familiarity: Even if someone uses an interface for the first time, certain elements can still be familiar. Real-life metaphors can be used to communicate meaning.
* Responsiveness: A good interface should not feel sluggish. This means that the interface should provide good feedback to the user about what’s happening and whether the user’s input is being successfully processed.
* Consistency: Keeping your interface consistent across your application is important because it allows users to recognize usage patterns.
* Aesthetics: While you don’t need to make an interface attractive for it to do its job, making something look good will make the time your users spend using your application more enjoyable; and happier users can only be a good thing.
* Efficiency: Time is money, and a great interface should make the user more productive through shortcuts and good design.
* Forgiveness: A good interface should not punish users for their mistakes but should instead provide the means to remedy them.
  + 1. Graphical user interface (GUI)

A graphical user interface is a type of interface that allows users to interact with electronic devices through graphical icons and visual indicators such as secondary notation.

Designing the visual composition and temporal behavior of a GUI is an important part of software application programming in the area of human-computer interaction. Its goal is to enhance the efficiency and ease of use for the underlying logical design of a stored program, a design discipline known as usability. Methods of user-centered design are used to ensure that the visual language introduced in the design is well tailored to the tasks.

Kites open source shopping cart was implemented with a good visual design and a great experience both involving heavy interaction with system users during development, which helped defining and shaping the user interface and the overall user experience.

* 1. Implementation of (Kites administration panel – JORY)
* Login screen  
  
* Catalog management screen  
  
* Products management screen  
  
* System operations (Back-up, Restore, Change credentials)  
    
    
  
  1. Implementation of (Kites client application-YASMIN)
* Welcome screen  
  
* Catalog view screen  
  
* Products view screen  
  
* Single product detail screen  
  
* Cart screen  
  
* Login screen  
  
* Registration screen  
  
* Payment method selection screen  
  
* Address confirmation screen  
  
* Order result screen  
  
* Order status checking screen  
  

1. Conclusion
   1. Conclusion

Mobile commerce shopping cart, another step integrating technology in traditional daily activities by delivering electronic commerce capabilities directly into the consumer’s hand, anywhere, anytime and automating the process of business transactions, utilizing already existing technologies and devices such as smartphones computers which make the process require minimum involvement of both shop owners and customers, thus saving time and effort.

The system (Kites open source shopping cart) was made of three main components; JORY the administration panel for shop owners, YASMIN for customers and ALICE the web service that connects to previous components.

All components were made following best practices, design patterns and cutting edge technologies. With standard specifications, ease of use and security in mind.

The use of technology also helps open the way for new generation of connected software and devices we hope that kites as software helps shaping the future evolutions such as Semantic Web and Internet of Things (IoT), providing Meta data and open API.

* 1. Recommendations
* Improve Kites administration panel (JORY) to access data persistence through Kites RESTful web service (ALICE) for better performance and scalability.
* Add support for multiple physical store branches.
* Add support for different Mobile platforms for Kites client application (YASMIN).
* Add support for TLS/SSL to all Kites components for better security.
* Add a web portal component for shop owners and customers and integrate it with existing components.
* Add support for offline catalog browsing.
  1. References
* Books
  + Kotler, P., Armstrong, G., Brown, L., and Adam, S. (2006) Marketing, 7th Ed. Pearson Education Australia/Prentice Hall.
  + Bannock, Graham et al. (1997). Dictionary of Economics, Penguin Books.
  + Juran, Joseph M. (1988). Quality Control Handbook (4th ed.)
  + International Organization for Standardization (2009). Ergonomics of human system interaction - Part 210: Human-centered design for interactive systems (formerly known as 13407). ISO FDIS 9241-210:2009.
  + Beck, Kent; et al. (2001). "Manifesto for Agile Software Development". Agile Alliance. Retrieved 14 June 2010
  + Griffin, Ben; Baston, Laurel. "Interfaces" (Presentation). p. 5. Retrieved 7 June 2014.
  + "Design Patterns and Refactoring", University of Pennsylvania, 2003, webpage: UPenn-Lectures-design-patterns.
  + The National Implementation Research Network.
  + Foukalas et al "Protocol Reconfiguration Using Component-Based Design"
  + Gosling, James; Joy, Bill; Steele, Guy; Bracha, Gilad; Buckley, Alex (2014). The Java® Language Specification (Java SE 8 ed.)
* Websites
  + "Mobile commerce", http://www.iamwire.com/2013/10/10-reasons-mobile-commerce-india-bigger-online-commerce, accesstime[4/22/2014 09:48]
  + "E-commerce", http://www.creativeworld9.com/2011/03/abstract-on-e-commerce.html, access time [8/20/2014 10:17]
  + "Extreme programming", ootips.org, accesstime[8/21/2014 21:35]
  + "Extreme Programming", http://agilealliance.org/resources/accesstime[12/14/2014 18:25]

1. “Why Mobile commerce is set to explode. http://www.businessinsider.com/why-mobile-commerce-is-set-to-explode-2013-5 [↑](#footnote-ref-1)
2. “US Mobile commerce” http://www.insidemobileapps.com/2013/07/05/us-mobile-commerce-to-become-235-million-market-in-2013-slow-to-adopt-mobile-consumer-payments [↑](#footnote-ref-2)
3. "Shopping cart", http://usatoday30.usatoday.com/yourlife/health/2011-03-02-grocerycarts\_N.htm [↑](#footnote-ref-3)
4. Agile Alliance. [↑](#footnote-ref-4)
5. Agile alliance. [↑](#footnote-ref-5)
6. "Design Patterns and Refactoring", University of Pennsylvania [↑](#footnote-ref-6)
7. Foukalas et al "Protocol Reconfiguration Using Component-Based Design" [↑](#footnote-ref-7)
8. The Java Language Specification [↑](#footnote-ref-8)
9. Kotler, P., Armstrong, G., Brown, L., and Adam, S. (2006) Marketing, 7th Ed. Pearson Education Australia/Prentice Hall. [↑](#footnote-ref-9)
10. "Agile allience [↑](#footnote-ref-10)
11. Agile alliance [↑](#footnote-ref-11)
12. The National Implementation Research Network. [↑](#footnote-ref-12)
13. International Organization for Standardization [↑](#footnote-ref-13)
14. Griffin, Ben; Baston, Laurel. "Interfaces" (Presentation). [↑](#footnote-ref-14)