Developing Innoventory, the Inventory and Ecommerce Application

1st Broday Walker   
*McCoy College of Science, Mathematics & Engineering*  
*Midwestern State University*Wichita Falls, USA  
brodaywalker@gmail.com  
  
Ben Diekhoff  
*McCoy College of Science, Mathematics & Engineering*  
*Midwestern State University*Wichita Falls, USA  
ben.diekhoff@gmail.com  
  
Matthew Stanley  
*McCoy College of Science, Mathematics & Engineering*  
*Midwestern State University*Wichita Falls, USA  
stanleycmatt@gmail.com  
 Corbin Matamoros  
*McCoy College of Science, Mathematics & Engineering*  
*Midwestern State University*Wichita Falls, USA  
corbinmatamoros@yahoo.com

*Abstract*— With the widespread adoption of the internet, online shopping has exploded in growth thanks to its convenience, competitive prices, and wide variety of goods for sale. Consumers are rapidly moving away from traditional brick-and-mortal stores to desktop computers and mobile devices to do their online shopping. The intuitiveness and simplicity of an online desktop or mobile application that can be used to purchase goods and services from anywhere in the world holds an undeniable appeal to consumers. Retailers and service providers must adapt to the changing landscape of ecommerce business in order to stay competitive. Online marketplaces must be easy to access with friendly interfaces that facilitate purchasing. The sooner a business offers a mobile-focused alternative to its desktop service, the more it stands to gain in market share.

This research focuses on the process of engineering Innoventory, a software solution that brings both sellers and consumers together. Innoventory features a desktop application that allows a business to manage its inventory and users to purchase goods. Innoventory also features a sleek, intuitive, mobile application that allows users to do their online shopping on the go.

In this paper, we will discuss why we chose this project, the software we utilized in the development process, and our requirements, design, and implementation activities. It is our goal that our research will provide insight into the process of engineering and developing an ecommerce application using agile methodologies.

Keywords— software engineering, online shopping, ecommerce, mobile commerce, inventory, mobile app, application

# Introduction

With the advent of the ubiquitous smartphone, more people than ever before are shopping online using mobile devices, such as smartphones, laptops, and tablets. The amount of dollars generated via mobile commerce, the subset of e-commerce sales from mobile devices, has steadily risen since as far back as 2015. By 2024, mobile commerce is expected to account for nearly half of all US ecommerce volume. With over $418 billion dollars being predicted to come from mobile phones alone, the writing’s on the wall for businesses: appeal to mobile users or get left behind, losing both customers and revenue [7].

Innoventory is our concept application developed to create a system that unites shoppers and businesses within a single ecosystem, while assisting businesses with their everyday inventory management. It consists of a mobile application and a desktop application. We developed Innoventory with agile methodologies, particularly Scrum. As such, our requirements specification, design, and implementation were interleaved and the requirements changed often.

The mobile application is aimed at smartphone users who want to buy goods from online sellers. It allows customers to create accounts, search for items and buy products from a business. The main focus of the mobile application is to provide an uncomplicated and intuitive user experience. It features a simple layout and transitions to give the customer the feeling of browsing a premium online marketplace.

The desktop application can be used by customers or businesses. It offers the same purchasing functionality as the mobile application, but includes tools that businesses can use for inventory management. A business can designate employee users that have the capability to manage inventory through the desktop GUI. The extra functionality includes changing an item’s availability and modifying their prices.

Our target users are small businesses, entrepreneurs, and consumers. Unlike large businesses, small businesses may not have inventory management software. These small businesses need an inventory management system that is affordable and less complex. Integrating our software into their business comes at a relatively low cost compared to Amazon or another retail giant replacing their entire system. Small businesses do not need overly complicated inventory management software where even a simple spreadsheet would suffice.

In the discussion of the Innoventory desktop and mobile applications, this paper will be divided into six sections: project context and description, requirements specification, design and implementation, validation and testing, results and discussion, and finally, a conclusion.

# Project Context and Description

This section briefly discusses the context of our project and the supportive software we reused in its development.

## Project Context

We chose to develop Innoventory as a concept application because we knew it would be a complex but achievable project, possessing novel topics such as mobile application development and GUI design. The idea of an application that shared functionality between a desktop and mobile version was intriguing. Finally, it built upon knowledge from our previous classes, offering a capstone project for our college experience.

In line with an Agile methodology, we split our application development into iterations containing the following parts: requirements specification, design and implementation, and validation and testing. The latter three we intended to perform concurrently, with changes to any requirements made if necessary. With one-and-a-half months to complete the project, we established seven-day sprints and weekly meetings to outline the deliverables for the next sprint. Lastly, we split our group into two subgroups for each to focus on either the desktop or mobile application.

We focused on making a simply-designed user interface with the basic features found in an online shopping application, such as the ability to log in after creating an account. We drew our inspiration from the likes of Amazon and eBay, which both have account-creation support, product search functionality, a shopping cart to which the user can add products, and the ability to purchase items in the user's shopping cart.

To improve their user’s searching experience, Amazon’s A9 search algorithm has optimizations that account for misspellings and variations of search terms [1]. A9’s ranking system also relies on product popularity to sort products returned from a customer’s query [2]. More so, A9 can differentiate between a product name and a product type (e.g. the term “casual dress”). The Best Match algorithm, eBay’s search feature responsible for ordering search results, judges listings by their popularity and closeness to the buyer’s search terms [6]. Best Match also protects the buyer from low-quality products by ranking search results by seller track record (those with good reputations will rank higher). Amazon shopping cart allows users to change the quantity of an item therein or remove it [3]. eBay offers users the ability to buy and sell by creating an account with a unique username [4] [5].

## Utilized Software

One of the most important aspects of the mobile application’s development was selecting the language and libraries with which to build it. We ultimately selected Python as our main language and the Kivy framework to handle the GUI aspect of our mobile application. For the desktop application, Python was also utilized in conjunction with PyQt for the GUI.

### Python: We selected Python for a few reasons. First of all, each member of our group had previous experience with Python. Some of our prior experience was in areas related to what we’d need to work on with Innoventory, such as database management. Python also has excellent official and unofficial documentation, and is used widely enough that online resources, such as tutorials or forums, are plentiful and easy to find. Finally, Python offers many libraries and frameworks, some of which can be used to build a GUI. We had little experience building GUIs, so the numerous options allowed us to pick and choose the software we’d use.

The primary benefit of using two GUI frameworks that share an underlying language in Python is the ability to create one shared interface that can be used to interact with the backend of the application. Both the mobile and desktop applications reuse the same components from the interface layer that sits between the frontend GUIs and the backend MongoDB document store. This feature allowed for development to be simplified as both applications utilized the same login system, search/query algorithm, shopping cart update system, and checkout functionality.

### Kivy: The frontend mobile GUI was developed using the Kivy python framework. Kivy is a framework that allows for development of GUI apps that can run on Windows, OS X, Linux, Android, and iOS. We chose to use Kivy to develop the mobile application primarily to make it easier to port the application to Android and iOS. In many cases, the application needs no additional effort to deploy on a new platform. The Kivy website also showcased multiple apps that ensured it would work for our purposes, which increased confidence in the framework.

Another benefit of the Kivy framework is in its own optional language called KVlang, which allows developers to code components of their GUI in a separate Kivy file. For example, a screen with several elements can be declared in the .kv file and referenced from python code. This allows for more readable code, since combining the GUI and application logic can quickly lead to dense, hard to read code. The .kv file also allows for the reuse of screen layouts, buttons, and widgets, thus reducing the amount of code on the Python side of the mobile application. In light of this, we used a Kivy file to hold as much of the GUI code as possible while including as much of the programming logic in the python code.

The Kivy framework supports the reuse of software in other ways as well: each widget is a class that can be extended to provide more complex custom GUI functionality. For example, several screens in the Kivy implementation of the Innoventory mobile application extend the Screen class, the BoxLayout class, and the GridLayout class to show dynamically-generated lists of items returned by search results. This functionality is extended to several other classes to facilitate the display of product information throughout the Innoventory application.

### PyQt: PyQt is the Python driver for the UI framework, Qt, and we used it to develop the desktop application. This package came with an application that allows users to design a window using labels, buttons, and other objects visually, called Qt Designer. The Qt Designer allowed for rapid deployment of changes to the GUI throughout the development process.

### Once a window has been designed, the .ui file is able to be converted to Python code and directly integrated into the Innoventory application. The PyQt implementation of the Innoventory Desktop application provides a sleek interface with all the functionality of the mobile application as well as some additional features for businesses to manage their inventory.

### MongoDB: MongoDB is a NoSQL document store, where documents resemble Python dictionaries. MongoDB is flexible, allowing for a reduction in effort when creating a database schema. This benefit allows for the Innoventory application to store and process complex product data that mimicks real-world data. The simulation data stored in the MongoDB backend was generated using a free online resource called Mockaroo. In total, 5,000 complex, realistic pieces of product data were generated and stored in the product database. The flexibility of the document-store, NoSQL design as well as our familiarity with the layout and supportive tools lead us to select MongoDB for our database solution.

# Requirements Specification

In this section, we state and discuss select requirements that were satisfied in the Innoventory application, the system architecture, and implementation. Some of the original requirements were not satisfied and the development of Innoventory evolved to accomplish as much as possible in the given timeframe. Both the mobile and desktop applications provide the basic functionality one would expect from an online e-commerce shopping interface.

## Customer Requirements

### Login: Innoventory must only allow users with valid accounts to access the store. Customers without accounts must be prompted to create one.

### Item Search and Discovery: Users can search for items in the store. Searches are performed by typing an item name into the search bar. Depending on whether there are results for a search, the application will either display the results one after the other vertically or a popup window explaining there are no results for the search will appear.

### Item Selection: From the search results page, a user may open an item’s description page by selecting the item. The description page must display an item’s, name, price, availability, and the quantity in stock. Lastly, the page must offer the customer the ability to add the selected item to their cart.

### Shopping Cart: From the search and item description pages, the customer must select a button to display items in their shopping cart. The shopping cart page must display an item’s name, its quantity, and the price for that quantity of item. If more than one item is in the shopping cart, information for each must be displayed vertically one after the other. A grand total for all items purchased must be displayed towards the bottom of the page.

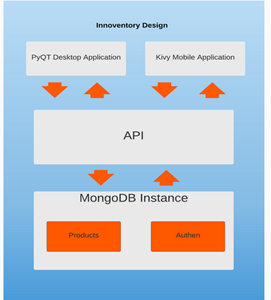
### Item Purchase: From the shopping cart page, the user can tap a button to checkout. Upon selecting checkout, the application should prompt them to confirm their purchase. The purchasing prompt shall submit the order if the customer reaffirms their checkout decision.

## Employee Requirements

### Login/Logoff: Employees must only be permitted to access the application through the desktop.

### Store Modification: The duties of an employee include product addition to and removal from the store as well as product quantity modification. These duties must be easily performed through the desktop application only.

# Design and Implementation

This section discusses the completion of requirements from section III. It is divided into four parts for discussion: system architecture and the mobile and desktop application.

*Figure 1 – Innoventory System Architecture*

## System Architecture

When designing the system architecture for the Innoventory applications, the team utilized the technique of enough design up front. This technique allowed for rapid development of the application while also providing the flexibility to add components and functionality to the system during each iteration. To support this, Innoventory was designed using three discrete layers.

Innoventory’s layers, as seen in Figure 1 – consist of the front end user interfaces, the API, and the database. By using the GUI, the users can navigate between pages, search for items, and/or add items to their cart. At no point is the user exposed to the backend. Instead, all requests for information or potential data manipulation access are processed by the interface layer. If any information or data is sent from the backend and returned to the user, this is also accomplished by the API.

The API provides a barrier between the user and the database. All interactions with the database, such as adding users, removing users, and authentication, are routed through the API. This object-oriented approach greatly simplifies our design, and also prevents our component code from being inundated with database calls and queries. An added benefit of separating the Innoventory application into discrete tiers is security-related. The interface layer will not allow customer users read or write privileges to the authentication database nor will it allow regular customer users to modify product availability, price, or quantity. In this way, we are able to strictly control the access to the sensitive backend database.

The API class has all the database connection strings and the basic functions for connecting to and querying the database. The Login class is used when a user tries to access the Innoventory application by logging in. The UserManager class has functions related to user account handling, such as adding a new user when they create an account. The Employee class provides functionality for users who can modify product quantity and availability. Lastly, the ShoppingCart class allows access into the Shopping Cart collection in the database, and lets customers modify items in their shopping cart.

Finally, the database receives information from and sends replies to the API when appropriate. For example, any search terms entered by the user from the front end GUI are received by the API and passed to the backend after the search term is validated. This separation allows for the interface layer to check and ensure all information entered into any front end GUI text boxes does not contain malicious code or symbols that are not legal. All product information displayed to the user, such as search results and prices, as well as the authentication service when a user logs in is handled by the database.

## Desktop and Mobile Application

The desktop and mobile applications operate in conjunction with the Innoventory API.

### User login and authentication

Upon starting up the Innoventory application, the user is greeted with a login page asking for the user’s credentials (username and password). If the user does not currently have an account, they are prompted to create an account by selecting on the “Create Account” button.

When selected, the “Create Account” button will redirect the user to a new window where they may create an account by entering a username and password. Once these credentials have been entered, the user may create their account by selecting a “Create Account” button present on this window. As an added security feature, a user cannot create an account with a username that is already being used. If there are no issues with account creation, the user’s credentials are stored into Innoventory’s database for future authentication. Once an account has successfully been created, the user may return to the login page through a “back” button where they must login to access the store.

### Item search and discovery

Upon logging in, the user is transferred to the search page which has a menu bar, a search bar, and a confirmation button. The user may type the name of an item into the search bar and select the confirmation button to perform the search. Behind the scenes, the search algorithm first searches all the store products for a direct match with the user’s search string. Afterwards, the search string is split by whitespaces (if the string was multi-worded) and a second search is performed with each word.

If there are search results, the application switches to the search results page with all of them visible. The most relevant search results are placed at the top of the page in descending order. Each item is interactable, and selecting one of them will take the user to its description page. On the other hand, if the search algorithm returns no results, the application stays on the search page and simply displays a message saying no results were found.

### Item selection and purchase

Innoventory contains features that show the user exactly what they are purchasing, and ensures they are absolutely certain about their purchase before completing a transaction. These features are implemented with an item description page, as well as a shopping cart and checkout page.

While searching for items, if a user discovers any item they have an interest in purchasing, they may observe this item’s description page by selecting it in the search results. This description page displays various information on an item such as its name, description, price, availability, and quantity currently available in the store. After reviewing this information, if the user still maintains an interest in purchasing the item, they may add it to their cart through an “Add to cart” button. By default, only one of those items can be added to the cart at this time. Upon selecting “Add to cart”, the user is notified that the item has been successfully added to their cart.

Once the user has finished searching for items and adding them to their cart, they may checkout their items by navigating to the shopping cart page. This page will display the items that are currently in the user’s cart, the quantity of each, their individual price, and the cart’s total price. The user can select on an item and either remove it from their cart or change the quantity they want to buy. Once satisfied, they select the “Check-out” button to complete their purchase. As a final confirmation, the user is asked if they are sure. This should ensure the user only purchases items they truly want.

### Employee-related functionality

As stated in the requirements, the employee may only perform their duties from the desktop application. Within the database, we’ve added a field to user accounts that identifies them as either a customer or an employee. When a user logs in from the desktop application, if they are acknowledged as an employee, the item description page updates to include “Change Quantity” and “Change Availability” buttons for the employee to modify an item’s quantity and availability. The “Change Quantity” button will be tied to a box in which the employee can type the new quantity. The “Change Availability” button does not delete or add the item from the product database; it toggles the item’s “availability” field between “true” and “false”. No new items can be added to the product database through the Innoventory application.

### Displaying dynamics sets in the mobile application

The most challenging aspect of the mobile application implementation was displaying search results and product information on a specific screen. A search for the letter “a” will return more results than a single screen can contain. To resolve this issue, we used RecycleView. RecycleView allows the application to display large, dynamic data sets in a scrollable list. This allowed the mobile application to deliver all search results in a single page instead of having them spread out over multiple pages, which would have increased the complexity of development. Additionally, RecycleView allowed us to display results for any item in a single product screen, again reducing the development complexity.

# Validation and Testing

This section discusses the desktop and mobile testing stage. We will discuss how we tested each implementation, as well as the API.

The primary method of testing for both applicaitons was whitebox testing, where tests were developed based on our knowledge of the source code. For both applications as well as the interface layer, there are automated tests that confirm the previously stated requirements are met. These tests include adding a new user, adding a new shopping cart for a new user, changing a user password, searching for an item, and deleting items from the users shopping cart, among other tests. To track the tests and their applicability to requirements, a testing matrix was used and included in the final project. This matrix shows information such as the requirement tested, the designer of the test, the result of the test, and whether a defect was identified.

In addition to the automated tests, manual testing was used on the front end GUI layer for both the mobile and desktop applications. This method was identified as the most reasonable approach to testing the GUIs, as automated GUI testing was beyond the scope of our project. The manual tests were mainly based on white box testing but also included destructive testing in an attempt to exploit or crash the application in an unexpected way. This destructive testing allowed for an examination of the desktop and mobile application’s robustness to unexpected user behavior or input.

## API

The API tests were organized according to the classes within it and the specific requirement(s) those classes were used to meet. For each tested requirement, a set of tests were conducted and run with the API Python file as the main file. In other words, running api.py executes all tests contained in the API’s main function. We inspected each class’ methods and ran tests that would cover as many possible uses of each method as possible. While no form of testing can test every possibility, our test suite covered as much of the functionality provided by the API as practicable. Each time a new feature or functionality was added to the interface layer, a test was added to the main function.

## Desktop and Mobile

The Desktop and Mobile tests were performed after a feature had been added to either implementation. We manually tested the GUI functionality using both white-box and black-box testing. White-box testing was conducted after a feature was added while black-box testing involved human users who were unfamiliar with the application. Each developer utilized friends or family members as testers of the application. This allowed us to observe user behavior from a unique standpoint and allowed us to identify improvements that could be made to facilitate a better experience for the user.

# Results and Discussion

The Innoventory application was completed over a period of nine weeks and was submitted along with documentation that consists of a user manual, a test case matrix, UML diagrams, and this report.

We modified the original requirements document as the application was developed since it was apparent we couldn’t complete all initial requirements within the nine weeks. For example, we initially wanted to create an automated load test involving the entire system simulating thousands of users interacting with the application simultaneously. This would have shown the capability (or incapability) of the application and the system architecture to scale to handle a large number of requests.

In addition to the lack of a large-scale user simulation, we were unable to develop a report generator, which would have printed a list of products in stock at any given time as well as trend data displaying product performance by individual type, product type, or price range. This functionality was identified as a valuable feature but not one necessary to producing the minimum viable product. Despite this, our product does provide shopping functionality for customers on both the mobile and desktop applications as well as some inventory management functionality for business employees on the desktop application.

If time had permitted, we could have improved the application’s overall quality by adding images to products, selecting a more reputable, albeit more complicated, framework for the mobile application such as Flutter, and developing only for one platform. Adding images would increase the usability of the applications as well as provide quicker selections for search results. Utilizing a more current mobile development framework would have benefitted the project by providing a more aesthetically-pleasing GUI. Our mobile application is simple to use, but it lacks a certain amount of flair that most customer users have become accustomed to recently.

After extensive work with Kivy on the mobile development, we would have preferred a different mobile framework, as Kivy’s lack of documentation and pervasive compatibility issues frustrated our productivity. Proper investigation into Kivy documentation and product examples would have discouraged our use of it, as it is not supported by a major company and development frequently lags behind its counterparts in the mobile framework space. Component compatibility is very important, and although we saved a lot of time by reusing software provided by Kivy, we did not set aside enough time in the design and implementation stages to investigate the documentation of selected components to ensure compatibility.

Ultimately, the most important lesson learned in the development of Innoventory was in the area of effort and time management. In hindsight, the project would have developed faster and more functionality would have bene delivered if we had focused on one platform (such as Android devices or mobile devices in general). Developing for two different platforms required our four-man team to split their effort into learning the foundations of two frameworks, which resulted in a disproportionate amount of effort expended for the functionality provided. Redirecting the efforts of the second team of two programmers would have allowed us to add more functionality such as a Manager role, who would have been able to add products to stock or generate stock reports.

# Conclusion

Developing the Innoventory e-commerce application was an incredible learning experience. This group project involved learning new frameworks as well as tying together multiple layers of a system architecture, which most of the development team had not had experience with before. Ultimately, the process of specifying the requirements for the application, designing and implementing the software, and testing and delivering the product reinforced the need to study and understand software engineering principles. Without the guidance of proper software design principles, complex application development is difficult at best, and nearly impossible at its worst.

##### References

1. B. Bowman, “Amazon SEO: How To Rank Products A9 Search Algorithm Info (2019),” The BigCommerce Blog, 24-Jan-2020. [Online]. Available: https://www.bigcommerce.com/blog/amazon-seo-strategy/#say-hello-to-a9-amazons-ranking-algorithm. [Accessed: 30-Apr-2020].
2. D. Sorokina and E. Cantu-Paz, “Amazon Search,” Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval - SIGIR 16, 2016.
3. “About the Shopping Cart,” Amazon, 2011. [Online]. Available: https://www.amazon.com/gp/help/customer/display.html?nodeId=201889540. [Accessed: 30-Apr-2020].
4. “Signing up for an eBay account,” eBay. [Online]. Available: https://www.ebay.com/help/account/signing-ebay-account/signing-ebay-account?id=4191. [Accessed: 30-Apr-2020].
5. “Changing your username,” eBay. [Online]. Available: https://www.ebay.com/help/account/changing-account-settings/changing-username?id=4198. [Accessed: 30-Apr-2020].
6. “Optimizing your listings for Best Match,” eBay. [Online]. Available: https://www.ebay.com/help/selling/listings/listing-tips/optimising-listings-best-match?id=4166. [Accessed: 30-Apr-2020].
7. A. Meola, “Rise of M-Commerce: Mobile Ecommerce Shopping Stats & Trends in 2020,” Business Insider, 17-Dec-2019. [Online]. Available: https://www.businessinsider.com/mobile-commerce-shopping-trends-stats. [Accessed: 29-Apr-2020]