Sarah Edris
03/17/13
6.170

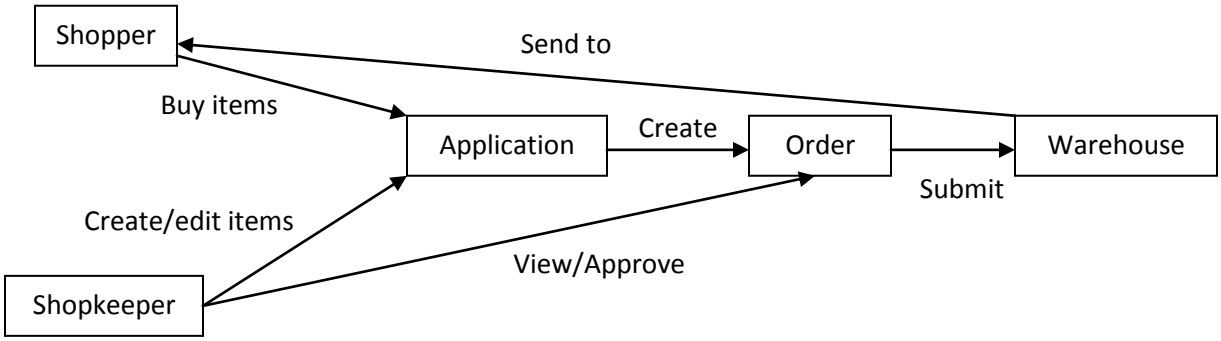
Project 2.3: Shopping Cart

Overview

Purpose and Goals

This system is a shopping cart application, with separate shopper and shopkeeper interfaces. It allows for the shopper to add items to a cart, save items and carts, and submit an order, and for the shopkeeper to do all of the same, as well as edit items and view recent orders. The main purpose is for it to be a simple and easy to use system, in comparison to sites such as Amazon and Ebay, with only the very necessities of a shopping application. Those larger applications often have more features, like having multiple stores and departments, which while useful, are sometimes not needed. A key motivation in creating this application is the desire to become more familiar with Ruby on Rails, authorization, authentication, sessions, and cookies.

Context Diagram



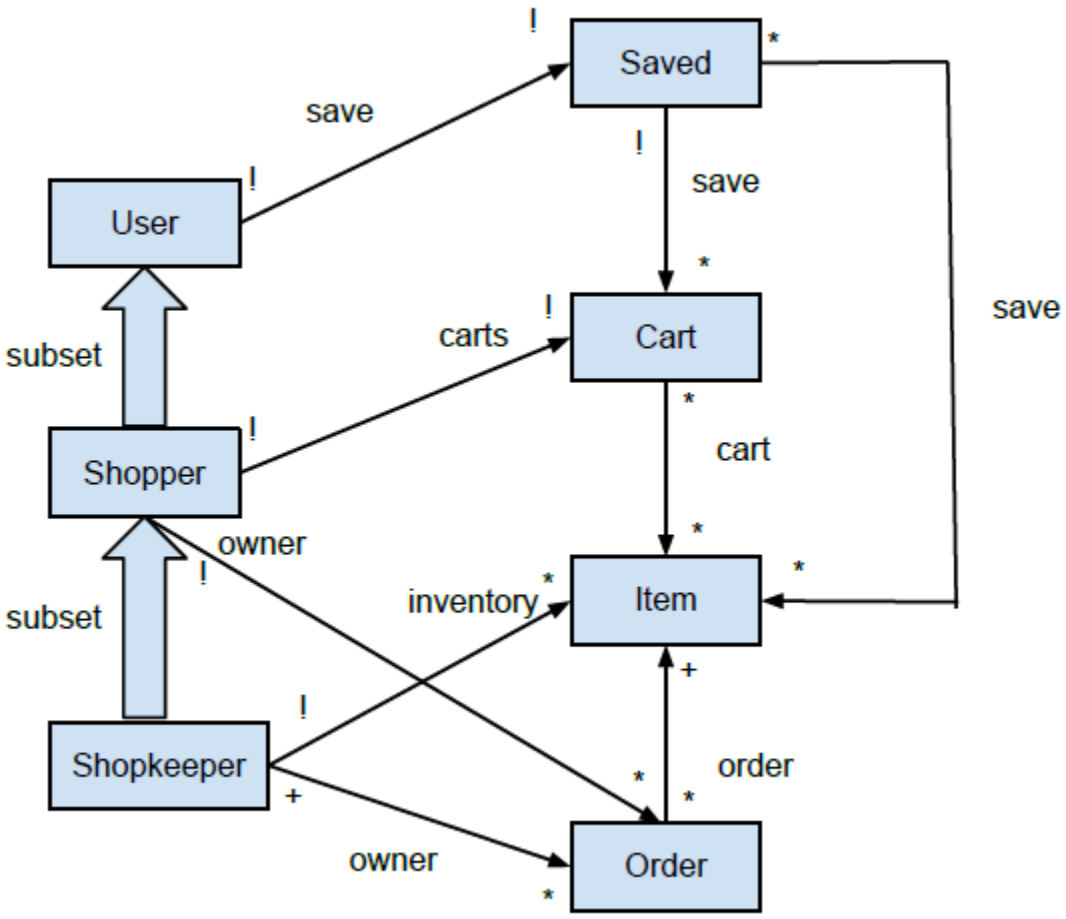
Concepts

Key Concepts

The key concepts are:

- A **user** is a person registered on the site
- A **shopper** is a user of the application that can purchase items and save items/carts
- A **shopkeeper** is a user of the application that can sell items and edit their information, as well as anything a shopper can do
- An **item** is any good that is sold through the application. An item is ‘unlimited’ in quantity, and the user adds one item to his cart at a time.
- A **cart** is a collection of items a shopper might buy
- An **order** is a collection of items that a shopper has decided to buy
- A **saved cart** is a collection of items a user had in their cart and saved for purchase later. A **saved item** is an item that the user has selected to save for purchase later.

Object Model



Behavior

Feature Descriptions

The following features are provided:

- **Signing up/Logging in:** The currently logged in user is saved using a cookie. All users must log in to access the site.
- **Shoppers putting/removing items from a cart:** A shopper can add and remove items from his cart. An item in a cart represents a single item. Shopping cart information is saved on the server, and not through a session.
- **Shoppers placing their order:** A shopper can place an order from the items in his cart.
- **Shopkeepers creating/editing/deleting items:** Shopkeepers can create, edit, and delete their items.
- **Shoppers and Shopkeepers viewing orders:** Shoppers can view their recent orders, and Shopkeepers can view their past orders and orders placed on their items.

- **Saving items and carts:** Items and carts can be saved for purchase later. Saved items can be added to the current cart, and saved carts can be directly checked out.

Security Concerns

User account protection is one concern, so all users that with an account must have a username and password, which is hashed on the server side. There are potential security threats with this though:

- **User not logging out**: While keeping the user's logged in state throughout browser closing and openings is convenient for the user, it could be dangerous if someone else were to use their computer while they were still logged in and bought items on their account. Because of this, a 'log out' button will be displayed prominently.
- **Users accessing site without registering**: There is a small added layer of security, as users must be logged into the site to interact with and view it.
- **Knowledge of users:** A list of all shopkeepers is provided so that users may access their items; however, a list of all users (including shoppers), is not provided anywhere, so shoppers have a level of anonymity.

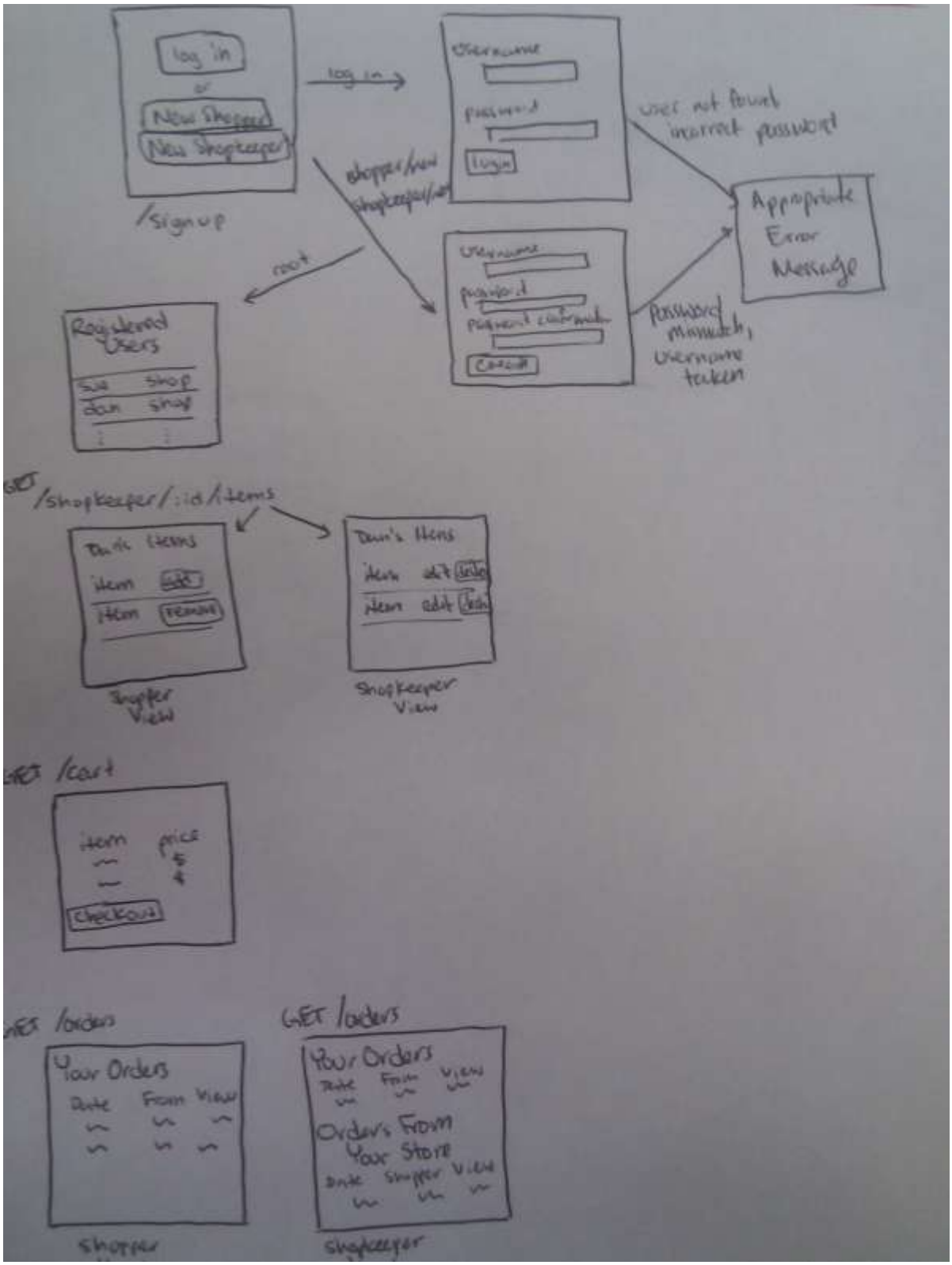
Other more complicated forms of hijacking or attacking will be ignored in this application because of its small scale. Further security measures may be added in future versions.

User Interface

A user initially coming to the site with no account must create an account, choosing between being a shopper, and being a shopkeeper. Attempting to go to a page in the site without being logged in will redirect the user to the login page. Once logging in, they are taken to a page listing all shopkeepers, and can select to visit a shopkeeper's store. Once at a shopkeeper's store, they may place items in their cart, remove items from their cart, save items to purchase later, or remove items from being purchased later. Items can also be removed through the user's cart page, and items/carts saved for later can be removed through the user's saved page. Once a user would like to place an order, they must go to their cart and select 'Checkout'. Once an order is placed, the user may continue shopping and adding items to the now empty cart, or view past orders on the orders page.

The shopkeeper has all of the same functionality as a user, but also owns items to sell. When going to their items page, they see added functionality to add, edit, and delete items. Items represent an unlimited quantity of the product.

Users may also logout through the logout link in the top right hand corner.



Challenges

Design Challenges:

The challenges in designing this include:

- **Relationship with Orders**
  How should orders be related to other models?
  Options:
  1. Owned by shoppers
  2. Owned by shopkeepers
  3. Owned by carts

  Each shopper can only have once cart, however, the relationship with orders is slightly trickier. Orders have ties to shoppers, shopkeepers, carts, and Items. The first and third options allow for easier searching through orders by shoppers, but not shopkeepers, while the second option does the opposite. Orders inherently come from the items from carts though, which made me choose the third option, while having each order keep a reference to shopkeepers that own its items, so orders can be searched through easily by shopkeeper. A shopper's orders can then be accessed by looking at the orders from their cart, or saved carts.

- **Carts and Shops**
  Should carts and shops be used? Or should Users just have items?
  Options:
  1. Items belong directly to shopper and/or shopkeeper.
  2. Intermediate layer of cart or shop between shopper/items and shopkeeper/items

  I originally planned on having carts as the intermediate object between shoppers and items, and shops as the intermediate object between shopkeepers and items. The shop and cart provide symmetric views of similar concepts, allowing for either to be later extended, making the second option the better one. However, for this project, shopkeepers are not needed to have multiple shops, although shoppers must be able to save carts, making the idea of a cart necessary. Because of this, I chose to use carts but not shops in my design.

- **Logging in**
  Should users have to log in to start using the site?
  Options:
  1. Login for access
  2. Can add items to cart, must login to order

  I originally planned on implementing a system to allow for unregistered users to add items to their cart and only log in once registered. This is more user friendly than the implementation I went with; all users must be logged in to access the site and add items to carts, as well as order. This was easier to implement, and because the sign up process is so simple (with only a username and password), I felt that this could be justified. If the process was longer, or more difficult, I would then switch to the second option.

- **Shoppers, Shopkeepers**
  What is a shopper? A shopkeeper?
  1. Shoppers are subset of users. Shopkeepers subset of Shoppers.
  2. Shoppers and Shopkeepers are both users with separate functionality

  The second option is easier to implement, as shoppers have shopping functionality (buy items/save for later), and shopkeepers have shopkeeper functionality (add/edit/delete store

items). However, the first option is more user friendly, and allows shopkeepers to have all of the functionality of shoppers instead of creating two accounts if they wish to sell and purchase items.

- **Saving Items and Carts**
  How to save items and carts for later?
  1. Add ‘saved’ flag to items in user’s cart.
  2. Add ‘saved’ model which contains items and carts.

  The first option might be easier to implement as it requires changing less. Items can all be added to a user’s cart, and shown in the ‘cart’ page or ‘saved items’ page depending on the flag. However, this design limits what can be saved to only items, meaning saving a cart would cause the grouped items to disassociate into saved items. This could possibly be fixed by allowing a user to have many carts, and marking extra carts as saved.

  I chose the second option because it clearly separates saved items and carts, and can be extended later if wanted, if perhaps a user wants to save other objects as well, like stores.