Sarah Edris
03/12/13
6.170

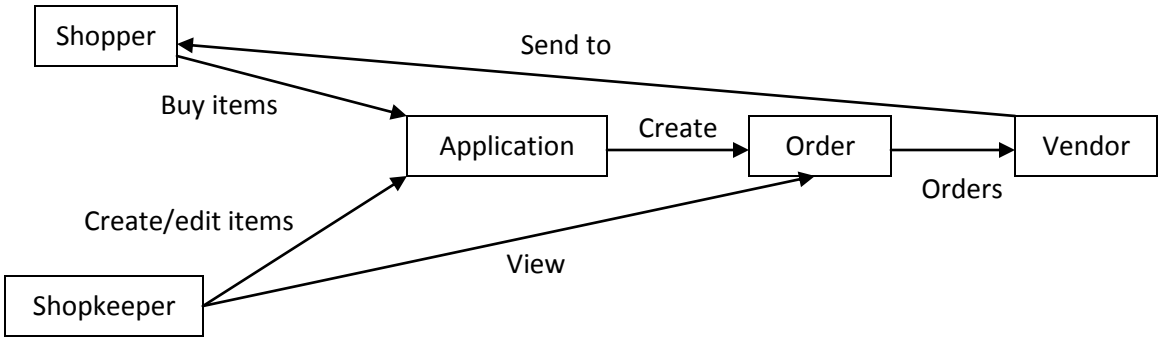
Project 2.2: Shopping Cart

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 and checkout, and for the shopkeeper to edit items and view recent orders. This is meant to be a simple and easy to use system, with only the necessities of a shopping application. It is designed to be easy to extend, specifically in a later part of this project. It is also a way to become more familiar with Ruby on Rails, authorization, authentication, sessions, and cookies.

Context Diagram



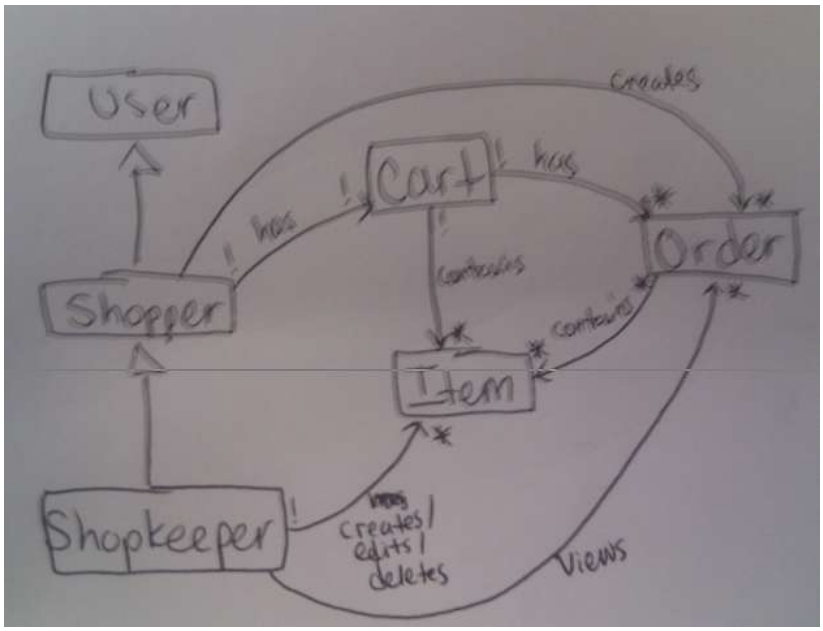
Concepts

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
- A **shopkeeper** is a user of the application that can sell items and edit their information, as well as purchase items
- An **item** is any good that is sold through the application
- 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

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. 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 shops:** Shopkeepers can own more than one shop to sell items from.
- **Shopkeepers editing item prices:** A shopkeeper can edit item prices. A shopkeeper must be a registered user.
- **Shoppers and Shopkeepers viewing orders:** Shoppers can view their recent orders, and Shopkeepers can view their past orders and orders placed on their items.

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.

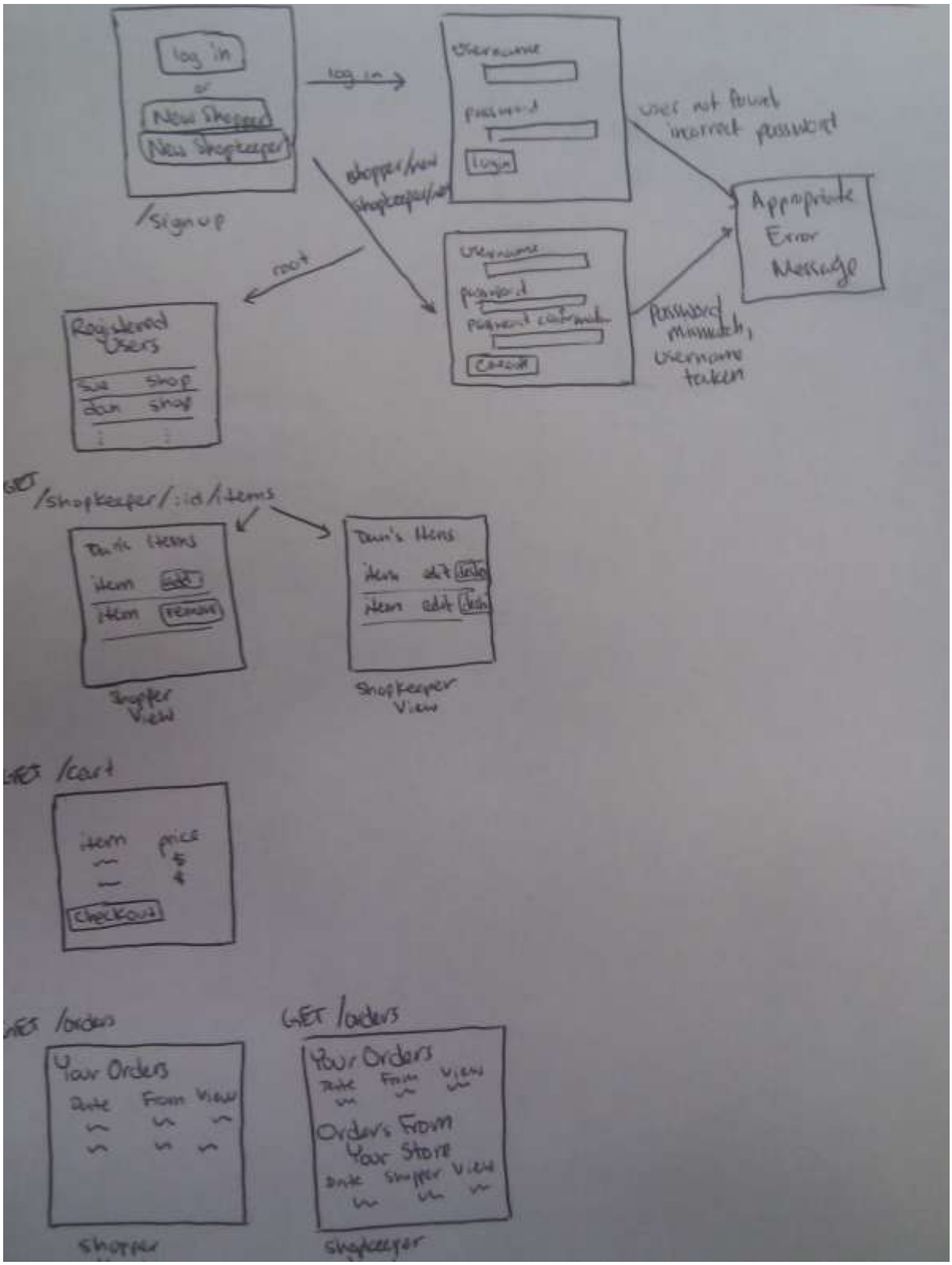
Other more complicated forms of hijacking or attacking will be ignored in this application because of its small scale. Further security measures may be added in future versions.

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. They are then taken to a page displaying all users, and can select to visit a shopkeeper's store. Once at a shopkeeper's store, they may place items in their cart, or remove items. Items can also be removed through the user's cart 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.

All users are displayed on the main page. The currently logged in user can delete their account through this page, by selecting 'destroy' next to their username. Users can only delete their own account.


log in
or
New Shopper
New Shopkeeper
/signup
log in
Username
password
login
user not found
incorrect password
shopper/new shopkeeper/
root
Registered Users
Sue shop
Dan shop
Username
password
Password confirmation
Create
Appropriate Error Message
Password mismatch, username taken
/shopkeeper/:id/items
Dan's Items
item edit remove
item edit remove
Shopper View
Dan's Items
item edit delete
item edit delete
Shopkeeper View
/cart
item price
$
$
checkout
/orders
Your Orders
Date From View
GET /orders
Your Orders
Date From View
Orders From Your Store
Date Shopper View
Shopper
Shopkeeper

Challenges

Design Challenges:

The challenges in designing this include:

- **Relationship with Orders:** Each shopper can only have once cart, however, the relationship between carts and orders is slightly trickier. Orders have ties to shoppers, shopkeepers, carts, and Items. I chose to have each cart have many orders, and each order to have a reference to the cart, and thus the shopper that made it. Orders also contain items, which can be used to determine the shopkeepers associated with them. This allows for accessing of orders through shoppers and shopkeepers, which is more user friendly as both types of users can view orders they care about. A less confusing way to do this would have been to simply associate orders with a shopper or multiple shopkeepers, however this would require looping through all orders to find the associated shopper of shopkeepers, which is not effective.
- **Carts and Shops:** 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. However, for this project, shopkeepers are not needed to have multiple shops. The shopper's cart is used to create orders, thus making the cart more necessary in this case than shops, which have not been implemented.
- **Concept of shoppers, shopkeepers, and unregistered users:** 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 first system.
- **Authentication:** All parts of the site are only meant to be accessible to registered users, which meant that for every action, the verification of a user must be made. However, some actions, such as creating a user, do not require this verification.
- **Authorization:** There are many parts of the system that are specific to just shopkeepers or the current user. For example, deletion of a user, and adding/editing/deleting items from a store. To make this occur, there are many gems that can be used. However, as an exercise in learning more about Ruby on Rails, and because this is meant to be a small application, this verification was done without other sources, and instead done by conditionally displaying links and redirecting in actions that users are not meant to access.