CS 360 Perfect Purchase Final Report

David Waters
University of Idaho
Moscow, Idaho, USA
elde1366@vandals.uidaho.edu

Rodney McCoy
University of Idaho
Moscow, Idaho, USA
rbmj2001@outlook.com

Hunter Leppek
University of Idaho
Moscow, Idaho, USA
lepp8728@vandals.uidaho.edu

ABSTRACT

The Perfect Purchase project is a service-matching commerce website which enables transactions between customers and vendors. The project operates on a database that contains the needs of the customers as well as the products and services offered by the vendors. The project has different interfaces available depending on the type of user account, which protects the confidentiality of both the customer and the vendor. Customers can choose to anonymously make a wishlist in which they specify their needs. The project will then search through the offered bundles, products, and services to find the option that best fits the customer's wishlist. The site also aims to find relevant results in this search in order to protect the customer from overpaying for a product or service that they did not need. Vendors can choose to offer products, services, and bundles that have binding contracts attached to them on the site. These offers will be shown to customers if they meet the needs specified by the customer's wishlist. The Perfect Purchase project is limited in ensuring the integrity of transactions that occur on the site and has no means of validating customers or vendors. However, future improvements to the project could involve some form of authentication of vendors so that customers can be more assured that the products and services offered on the site are legitimate.

CCS CONCEPTS

• **Information systems** → **Resource Description Framework (RDF)**; **Question answering**; *Graph-based database models*; *Web Ontology Language (OWL)*; • **Computing methodologies** → **Natural language processing**; **Knowledge representation and reasoning**.

KEYWORDS

Administrator, Web developer, Django, SQLite, Mockaroo, Customer, Vendor, Perfect Purchase project

ACM Reference Format:
David Waters, Rodney McCoy, and Hunter Leppek. 2018. CS 360 Perfect Purchase Final Report. In *CIKM '22: 31st ACM International Conference on Information and Knowledge Management*. ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/1122445.1122456

CIKM '22, Oct 17–22, 2022, Atlanta, Georgia, USA

ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

"Effective website design plays a critical role in attracting and maintaining customers' interest and in influencing their purchase behavior" [2]. Most websites, including web-based projects such as the Perfect Purchase project, are comprised of two main parts: the front end and back end. The front end is the interface that the users interact with to traverse the website. These interfaces should be simple to navigate, as well as appealing to look it so that users have a comfortable experience while on the website. The website itself should be quickly responsive to user input, as well as be able to handle any requests the user has at a given time. In the case of the Perfect Purchase website, the front end allows users to create entries in the SQL database, as well as search the database for products and services. The actions that the user can take on the front end are dictated by the type of user. That is, only vendors will be able to offer products, services, and bundles whereas customers can only specify their needs. The back end of websites is where the information is contained and manipulated. Notably, the databases of the website are kept in the back end and are only accessible via the front end for normal users. Web developers are the people who create both the front end and back end of the website and tailor it to meet their specific needs. Special users called administrators are responsible for maintaining the website. For example, administrators keep web page contents up to date, repair broken links, and implement security measures such as firewalls to protect the integrity of the website. Additionally, administrators keep back-ups of database information in the event that data integrity is somehow compromised on the back end of the website. Both the web developer and administrator are a type of "superuser," who has the ability to modify both the front and back end of the website. In the case of the Perfect Purchase project, we were tasked with creating a website with both a front and back end that would enable transactions between customers and vendors. This project was initially intended to be hosted on a university server, but the specifications of the project were changed so that the hosting was done locally. Since the project is locally hosted, there was little need for a dedicated administrator role. As such, every member in our development group for the project played the role of web developer, as well as the role of administrator.

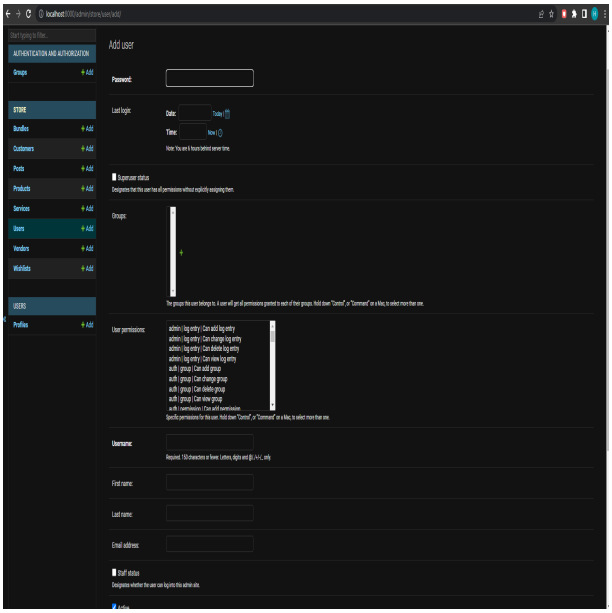

Figure 1: Administrative action.

1.1 Project Proposal

In the Perfect Purchase project, our group has created a service-matching commerce website for customers and vendors. The customer will anonymously specify their background and needs in the form of a wishlist. The vendors will offer products, services, and bundles (with associated binding contract) on the website to potential customers. The website will then search through the offered products, services, and bundles to find the option that best fits the customer's wishlist, while also protecting the customer from overpaying for something that they do not need. The wishlist of the customers can be in one of three forms: the first form being an exact match. This wishlist form specifies that the customer wants a particular product or service, such as a specific type of cellular device from a specific company. The next form of wishlist is a closest match, in which the customer will specify criteria for their need across multiple categories. These criteria could be labeled in order of importance. For example, a critera-based wishlist may specify that it wants to see all bundles that have televisions under $300[1st priority] that can be delivered in three days[2nd priority], and 50 available channels for two months [3rd priority]. The last form of wishlist is a requirement-based best match. This form matches the product with the need, and forgoes accounting for features. Essentially, this form of wishlist finds the essential elements offered that will satisfy the need of the customer.

2 DESIGN AND ARCHITECTURE

The Perfect Purchase project is a commerce website that matches a customer's needs (provided as a wishlist in one of three forms) to a product, service, or bundle provided by a vendor. The schema used by this project's databases are focused on the specifications of the wishlist, held in the wishlist table.

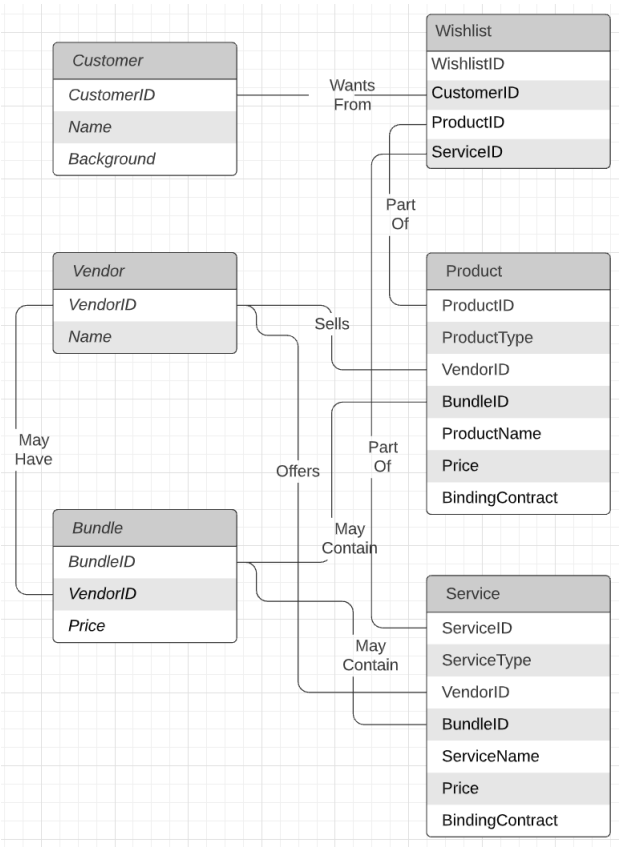

Figure 2: ER Diagram

2.1 Customer

The customers table holds information relevant to the customer, such as the customer ID number, the customer's name, and their background. The information in the customer table is accessible only to administrators and web developers, who are under policy to keep personal information confidential. Furthermore, vendors are not able to access this information. Consequently, the Perfect Purchase project does not compromise the privacy of its users. Customers are able to create wishlists to specify their needs. These wishlists will be used by the back end of the project to find products, services, and bundles that match the needs outlined in the wishlist. Note that the customer cannot directly access the products, services, and bundles offered by the vendors. This is simply a convenience for the customer since they do not have to do any searching themselves, rather they only need to voice their needs in a wishlist and the project website will take care of the rest.

2.2 Vendor

The vendor table holds the ID and name of the vendor. Vendors are able to create offers for products, services, and bundles on the project website. The details for each respective item or service offered are kept in the respective tables in the database. In the current state of the project, there is no way to link vendor accounts together, so a company would likely have multiple separate accounts under the same vendor name.

2.3 Product, Service, and Bundle

These three tables hold the relevant information defined by the vendor, such as the name, price, and contract of the particular product or service. Bundles are a way for a vendor to group products and services together and offer them at a usually discounted price. Currently, bundles do not have their own binding contracts, rather the contracts are a part of the included products and services.

2.4 Wishlist

The wishlist table holds the ID of the customer as well as the matched product and service IDs. The customer is able to indirectly query information about the products and services offered through the website via wishlists. This is done by the customer creating a wishlist and detailing their need in one of three forms: exact match, closest match, or requirement-based best match . Once a wishlist is created, the website will find products, services, and bundles that the wishlist defines as acceptable and return them to the wishlist. The wishlist will then return the information back to the customer, who can browse through the options and choose to accept the binding contract of a particular product or service.

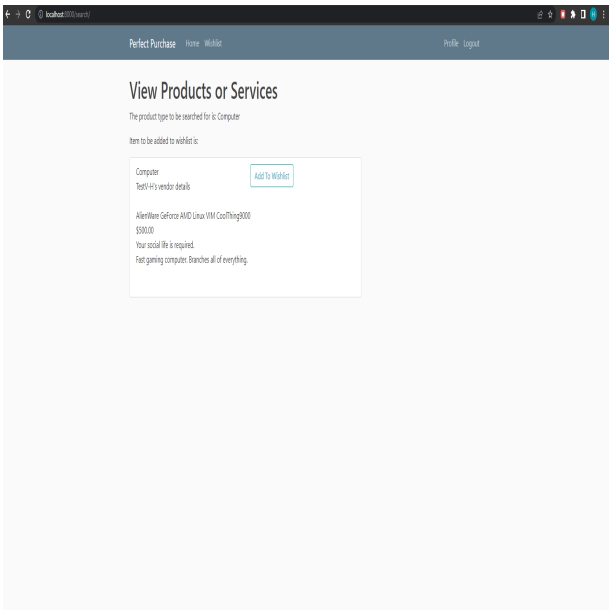

Figure 3: An interface implementation of the wishlist.

3 FUNCTIONALITIES

The project website has three interfaces available: one for customers, one for vendors, and one for superusers.

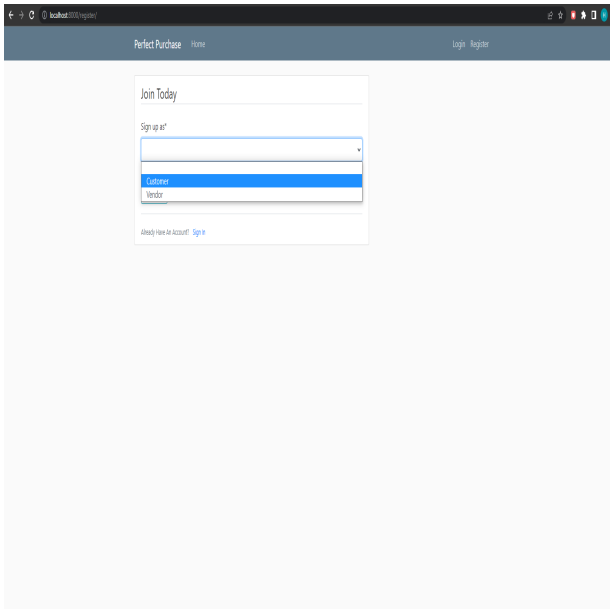

Figure 4: Creating an account.

3.1 Customer Interface

The customer interface is available to users with a customer account. The information related to the customer account is kept confidential and is only able to be modified by the customer who created the account or a superuser. From the customer interface, a customer is able to create a wishlist that will allow them to indirectly interact with vendors to find a product, service, or bundle that suits the need detailed in the wishlist. Wishlists will be able to read information in the product, service, and bundle tables and report listings that match the needs to the customer.

3.2 Vendor Interface

The vendor interface is available to users with a vendor account. Similar to a customer account, the information relevant to the vendor account is to be kept hidden from other customers and vendors, and can only be viewed by a superuser or the account from which the information originated. From the vendor interface, vendors can offer products, services, and bundles indirectly to the customers on the website by creating entries in the product, service, and bundle table. From this interface, vendors are also able to update and delete existing rows in the product, service, and bundle tables if their vendor ID matches the one stored in a particular entry.

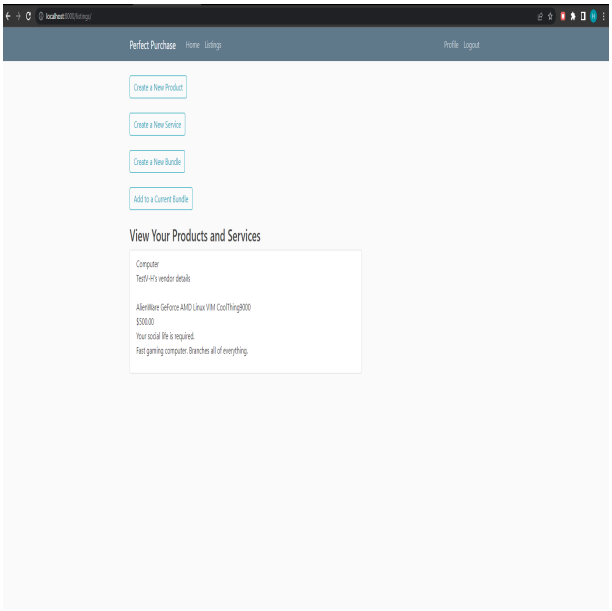

Figure 5: Vendor Interface.

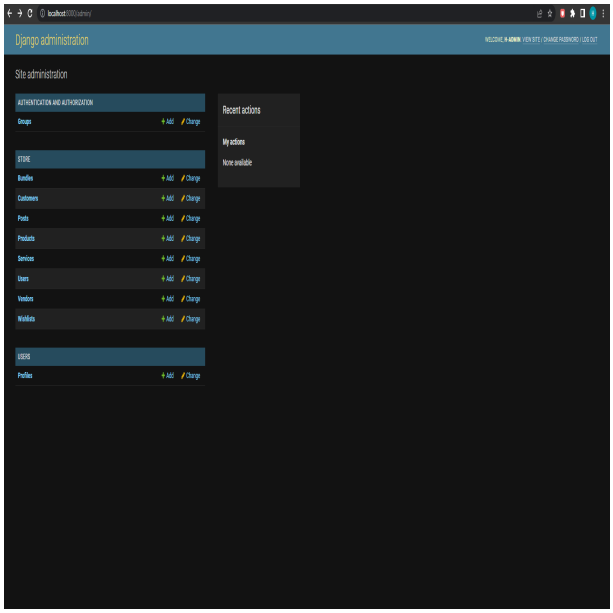

Figure 6: Superuser Interface.

3.3 Superuser Interface

Superusers are users such as administrators or web developers that need the ability to modify the website on both the front end and back end. A superuser is neither a customer nor a vendor, but is able to access and change both customer and vendor views. Superusers are able to directly see all information available in the database and create, remove, update, or delete rows as necessary. As such, only privileged users are able to see the superuser interface. Access to this interface requires a superuser account, which is only available to those who designed the website. In the case of our Perfect Purchase project, the web developer and administrator roles are carried out by all members. As such, only one superuser interface is needed. This foundation of this interface was automatically provided by Django, so our development team needed only to change a few minor details to fully complete this interface.

4 DESIGN CHOICES AND PLATFORM

The Perfect Purchase project was developed using Django, a high-level Python web framework, and SQLite, an in-process library that implements a transactional SQL database engine. Both of these platforms are free to use and have a plethora of documentation available online. The text editor used to develop this project was Visual Studio Code, and the code was distributed via GitHub (see section 8). Lastly, the databases used in this project were populated using Mockaroo.

4.1 Django

Projects in Django can easily be created by running the command: "django-admin startproject name". This command creates a directory with files that implement a foundation for a website project. Namely, the "manage.py" file allows web developers to run commands such as "runserver", which starts the website. Actions such as modifying the content and function of the website are done throughout multiple other files that the web developer will likely create in order to mold the website to fit the desired function.

4.2 SQLite

SQLite is an extension that can be easily added to a Django project with Visual Studio Code's extension feature. Once the feature is installed, an executable for SQLite (downloaded externally) can be dropped into the project. This executeable can be invoked within a terminal in the project to perform a number of database-related operations such as importing tables from .csv files.

4.3 Mockaroo

To populate our database with randomized test data, we used Mockaroo: a free random data generator. Mockaroo is able to generate tables with random fields of specific types, which allows our project to imitate information that may be stored in the database were the

project publicly deployed. While Mockaroo provides a wide range of random data of certain types, it does not cover every possible category of data. However, Mockaroo allows you to import lists so that it can populate fields with random values from the imported lists. Once a user is done creating their data, they can download it from Mockaroo in common data formats, such as .csv format. These formats are able to be imported into Django projects using software such as SQLite.

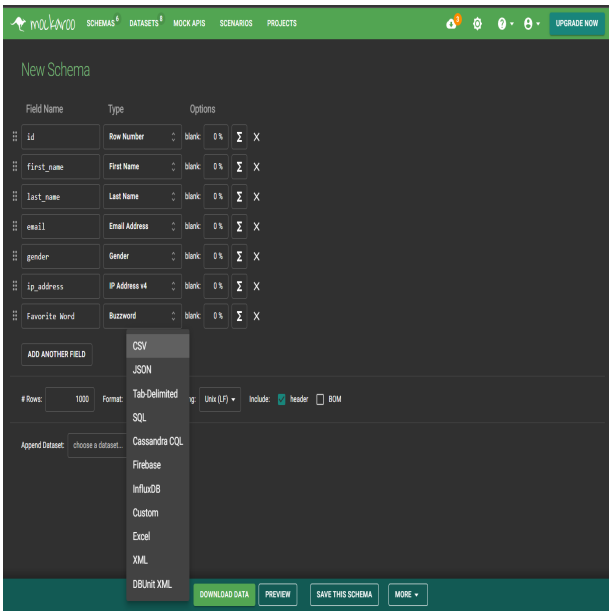

Figure 7: Using Mockaroo to generate random data

5 CURRENT LIMITATIONS

As stated before, the Perfect Purchase project can only run locally at this current time. We created the project with this point in mind, so were it to be deployed publicly, there would be a few limitations that would warrant addressing.

5.1 Locally Hosted

Having this project locally hosted means that every customer and vendor would need access to the localhost to use the site, which is not feasible. To fix this limitation, the project could be publicly deployed in a number of ways. For example, if the university had servers available for use, the project could be hosted on that. Another method would be to use one of the many web hosting services, such as A2 hosting or InMotion hosting [1], to publicly deploy the project to the world.

5.2 Security

The Django web framework provides a plethora of security options, such as mandating strong passwords and protection against common attacks such as XSS and CSRF. However, the security of the entire platform is dependent on how our project uses other tools to function. With every included tool, the attack surface of our project increases. Thus, it is important for us as developers and administrators to stay vigilant in upkeeping the security of the project. Security is an ongoing battle for pretty much anything web-based, so there is no easy way to solve this limitation other than always being active and mindful of threats against the project and the information it holds.

5.3 Maintenance

As discussed before, the local hosting of the project allowed our group to forgo having a dedicated administrator role, and instead, every group member played the role of web developer and administrator during the development of this project. However, this would likely not be feasible were the project to be publicly deployed since some aspects of the project would inevitably break and cause the development of any future features to halt. To fix this, our group could either expand to include more members that would fill the administrator role, or since the bulk of the development would be over with the launch of the website, one member could relinquish their role as web developer and instead be the dedicated administrator.

6 FUTURE IMPROVEMENTS

6.1 Bundle Binding Contracts

Currently, bundles by themselves do not have binding contracts, rather, the products and services included within the bundles have their own binding contracts. In this case, the customer would have to agree to all of the contracts in order to accept the bundle. However, this is not ideal because there may be overlap between the contracts. Furthermore, this structure allows for less flexibility for vendors offering the bundles since they cannot explicitly specify the terms for accepting the bundle. A better system would be to let bundles have their own binding contracts which would overwrite the individual contracts of the constituent products or services. This would allow vendors to offer more specialized deals to customers on the project website.

6.2 Linking Vendor Accounts

Currently, there is no way for vendor accounts to be linked in some way. This poses an issue because large companies that offer a variety of services may have multiple accounts on the website. Not only does this make it more difficult for our administrators to keep track of each vendor, but customers may be suspicious of many different accounts claiming to be a part of the same company. Lastly, if there were multiple accounts for the same vendor, the vendor may offer the same product multiple times across many accounts, since the vendor has no way of checking what is already posted on a different account. This would create unnecessary bloat on both the front end and back end of the project, and is thus undesirable. A solution would be to have an additional field in the vendor table, perhaps a Company ID, that would link accounts together. This linking would not allow access to the resources of all accounts under that company, but it would let every account see what is listed by the company already. Additionally, this would make it easier for the website to perform its function. Lastly, this linking fix would minimize excess bloating on both the front and back end that may arise due to duplicate listings and typos.

7 CONCLUSION

Throughout the development of this web application, our group has tailored the Perfect Purchase project to match the needs of a customer to a product, service, or bundle offered by a vendor. Moreover, the project is relatively secure and can be easily modified to be deployed in a public environment. A new customer could easily create an account on the website and begin crafting a wishlist that will find deals offered by vendors to suit their needs. Vendors can create offers for products, services, and bundles that will be indirectly presented to customers via the customer's wishlist. Although there are a few shortcomings of the project that would need to be fixed before complete deployment to the public, the Perfect Purchase project possesses the requisite functions to enable transactions between customers in need and vendors willing to sell.

8 SOURCE CODE

The source code for this project can be found at

https://github.com/RodneyMcCoy/store-website-database

Note that this repository will be private until the project is graded and turned in. Request access if need be.

REFERENCES

[1] Zachary McAuliffe, David Gewirtz, and Rayome Alison DeNisco. 2022. Best web hosting providers: A2hosting, HostGator and more. https://www.cnet.com/tech/services-and-software/best-web-hosting/

[2] Jaeki Song and Fatemeh Zahedi. 2001. Web design in e-commerce: a theory and empirical analysis. *ICIS 2001 Proceedings* 1, 1 (2001), 24.