# Team J

CSC 322: Software Engineering Group Project
Software Requirements Specification For
Donut PCs

**Front End:**
Shehab Mohsen
Brandon Lam

**Back End:**
Jed Rendo Magracia
Justin Jacob

# Revision History

| Date | Version | Description | Author |
|---|---|---|---|
| 3/23/2023 | 1.0 | Initial Draft | Jed Rendo Magracia<br>Justin Jacob<br>Shehab Mohsen<br>Brandon Lam |
| 3/25/2023 | 1.0 | Adding information to section 2 and 3 | Justin Jacob |
| 3/25/2023 | 1.0 | Adding details to 3.1 and 1.3 | Justin Jacob |
| 3/27/2023 | 1.0 | Added details to 2.2 | Jed Rendo Magracia |
| 3/27/2023 | 1.0 | Added ER diagram and Use Case diagram model | Brandon Lam<br>Jed Rendo Magracia<br>Shehab Mohsen |
| 3/27/2023 | 1.0 | Added User interface prototypes | Brandon Lam |
| 4/06/2023 | 2.0 | Adding team memos and GitHub repository | Justin Jacob |
| 4/10/2023 | 2.0 | Migrated report to LaTeX | Jed Rendo Magracia |
| 4/19/2023 | 2.0 | Updated group memos | Jed Rendo Magracia |
| 4/20/2023 | 2.0 | Updated pseudocode and inserted ER diagram | Jed Rendo Magracia<br>Justin Jacob |
| 4/21/2023 | 2.0 | Use case diagram. | Shehab Mohsen |
| 4/23/2023 | 2.0 | Worked on more wire frames, and added UI examples onto the report. | Shehab Mohsen |
| 4/24/2023 | 2.0 | Inserted full website collaboration class diagram. Added to use-case survey the types of cases it is (normal vs exceptional). | Justin Jacob |
| 4/25/2023 | 2.0 | Inserted petri-nets and sequence diagrams for login, employee, customer | Brandon Lam |

# Table of Contents

## List of Figures

# 1 Introduction

## 1.1 Purpose

The purpose of this Software Requirements Specification document is to describe the behavior of the subsystem which is a PC purchasing website. Functionalities and requirements for the separate components of the system, interaction with the end user, and specific methods of integration will all be discussed in this report.

## 1.2 Scope

DP is a web application focused on PC parts and creation where we've developed a system so that customers can build and purchase computers and computer parts. The main set of actors involved are visitors, customers, employees, and owners. Visitors will simply be able to view the website and products, they cannot do anything the customers can do. Visitors to the website will have the option to sign up for further privileges. PC builds will be custom and pre-built. The website will allow the customers to purchase, like, comment, create their own PCs, and have a financial system. Employees will be able to set up all this content, moderate it, and suggest builds. Owners are able to manage employees and also do anything that an employee can do as they are superior to the employee class. Custom builds will focus on compatibility while pre-built builds will focus on user reviews.

## 1.3 Definitions, Acronyms, and Abbreviation

- **Python** - Python is an interpreted, object-oriented, high-level programming language with dynamic semantics (do not need to declare variable types)
- **HTML** - HyperText Markup Language
- **CSS** - Cascading Style Sheets; is a style sheet language used for describing the presentation of a document in a markup language such as HTML
- **JavaScript** - JavaScript is a scripting language that enables you to create dynamically updating content, control multimedia, animate images, and pretty much everything else
- **TypeScript** - A superset of the scripting language JavaScript. Allows for better error handling, editor support, and readable code by adding another layer of syntax
- **Sass** - A stylesheet language that's compiled to CSS. It allows for usage of variables, nested rules, mixins, functions, etc. Helps keep stylesheets well-organized and makes it easy to share design within and across projects
- **Django** - Django is a Python-based web framework that follows the model–template–views architectural pattern
- **React** - React is a free and open-source front-end JavaScript library for building user interfaces based on components

- **SQL** - Structured Query Language. A Programming Language which is used to manage databases and perform various manipulations with data
- **API** - Application Programming Interface. In the context of APIs, the word Application refers to any software with a distinct function. Interface can be thought of as a contract of service between two applications
- **Visitor** - An unregistered user who can only view the website but not use any of the functionalities. They will be prompted to sign-up if they try to do any of the customer-marked actions
- **Customer** - Is a registered user who is capable of using all the functionalities of the website such as purchasing anything, creating a pc, depositing money, making comments, making ratings, etc.
- **Employee** - Is an administrator capable of managing customers, products, and the rating systems
- **Owner** - Is an administrator capable of managing customers, products, employees, the rating systems, and the overall direction of the website

### 1.4 References

Each of the bullet points are hyperlinks to their documentation.

- Python Documentation
- JavaScript Documentation
- TypeScript Documentation
- Sass Documentation
- Django Documentation
- ReactJS Documentation
- PostgreSQL Documentation
- Spec Sheet: Provided by Professor Jie Wei

### 1.5 Overview

In the following sections, we will discuss how the PC and parts purchasing website will work. We will go over how the customers can interact with the website and how the employees can manage the website. We will overview the functionality of the database, hosting servers, and the API developed for the frontend. Furthermore, the report will showcase the requirements needed for certain conditions and features for the customer/employees/visitors/parts/PCs.

# 2 Overall Description

## 2.1 Use-Case Model Survey

- User Types
  - Owners
    - Can override employee decisions on customer statuses.
    - Can receive criticisms and compliments from customers.
    - Can manage the employment status of the employees.
  - Employees
    - They can set up certain kinds of pre-built builds for certain groups of people.
    - Employees can rate and manage products and users based on the feedback received from their other team members and customers.
    - Employees can receive 3 warnings to be demoted.
    - Employees can receive 2 demotions to be fired.
    - Employees with 3 compliments will be promoted.
    - Can approve or reject new customer applications.
  - Customers
    - They can sign in and buy any number of prebuilt PCs or parts.
    - They are able to browse the store.
    - They can deposit money into their account.
    - They can build custom PC rigs.
    - They can rate any product from 1 to 5 stars.
    - Customers that receive 3 warnings will be kicked off the system.
    - Customers that receive 3 compliments will receive a 20% discount on their next purchase.
  - Visitors
    - They are unregistered users that can view the website.
    - They can register to get the perks of being a customer if wanted.
- Use Cases
  - Home Page
    - This will display general information about what we sell, featured production, and navigation to other pages.
    - This page will be accessible to all users (customers and visitors).
  - Product Pages
    (This will include information on the product)
    - Name
    - Price

    - Specs
    - The ability to add the product to cart
    - Products it is paired with
    - Navigation to other pages
  - Checkout Page
    - This will display all of the products added to the cart.
    - This page will allow a user to use the money in their account to purchase desired items.
  - Build-Your-Own PC
    (Here a user will be able to design their own custom PC)
    - Will use compatibility checking to make sure that the PC will work.
    - They will be able to add the PC to their cart.
  - Login Page
    - This page will allow existing users to log into their accounts and access the website with all the user functionalities.
    - This page will also navigate visitors to the sign-up page.
  - Signup Page
    - Here a visitor will be able to sign up for an account to access the wider features of the website.
  - Employee Management Page
    - Here employees will be able to manage products.
    - The page will also allow managing of customers.
    - Both of the above will be based on personal auditing and comments and complaints left by the user.
  - Owner Page
    - Here the owner will be able to manage their employees.
    - This page will allow managing of customers as well.
    - They will be able to see stats about their business.
    - They will be able to see inventory.

## 2.2 Assumptions and Dependencies

- Django REST Framework will work as intended
  - The backend framework will be able to correctly handle the API requests that the frontend sends.
  - It will be able to handle multiple calls and handle calls that require proper authentication.
  - The framework will be able to connect and query to the database properly and retrieve the correct information needed for the API request.

- The database is correctly designed
  - No repetitions of tables or user information.
  - Data can be correctly inserted into the tables, and properly displayed from the tables.
  - Foreign keys are set up correctly to work between different tables in the database.
  - The tables within the database are correctly designed, without any corrupted data and it will handle deletions of data properly.
- The application will be web-based
  - Users will be able to navigate through the UI from various devices such as phones and computers.
  - The website will have accessibility features, such as working well with screen readers and color contrast.
  - Users will be able to design their own PCs and make sure that the parts chosen are compatible.

# 3 Specific Requirements

## 3.1 Use-Case Reports

| **Features** | **Who Can Do It?** | **Which Page?** |
|---|---|---|
| Sign Up (normal) | Visitors | Signup Page |
| Login (normal) | Customers, Employees, Owners | Login Page |
| View Products (normal) | Visitors, Customers, Employees, Owners | Product Page |
| View Pre-built PCs (normal) | Visitors, Customers, Employees, Owners | Product Page |
| Build Custom PC (normal) | Visitors, Customers | Product Page |
| Purchase Products (normal) | Visitors, Customers, Employees | Product Page |
| Make Comments/Criticisms (normal) | Customers | Product Page |
| Make Ratings (normal) | Customers | Product Page |
| Ban/Give Customer Discount(normal) | Employees, Owners | Employee Managing Page<br>Owner Managing Page |
| Fire/Demote/Promote Employees (normal) | Owners | Owner Managing Page |
| Review Applications (normal) | Employees, Owners | Owner Managing Page<br>Employee Managing Page |
| Add/Remove Items (normal) | Employees, Owner | Owner Managing Page<br>Employee Managing Page |
| Maintenance (exceptional) | Employees, Owner | Django Maintenance Page |

## 3.2 Supplementary Requirements

Our PC purchasing website has some software and hardware requirements that are free and some non-free. Some of these requirements will include a stable and reliable internet connection, cloud server space, and a hosting domain. For the use of this website, a customer must have a personal computer that uses a React-supported browser and up-to-date updates with Python and JavaScript.

# 4 Group Meeting Memo

## 4.1 Meeting Notes

1. 3/30/23
    - All members werre present
    - Discussed API routes and web-scraping
    - Discussed special features (might use ChartJS) or make a chatbot. Gambling for PC Builds + Parts
    - For frontend: Prioritize customer features/pages then move on to staff/admin pages. What remains to be done on the wireframe for now is register/login pages, order history page, cart page, comments component, suggest build page, checkout page.
    - For backend, work on implementing the Django API with all authentication done. Set up a PostgreSQL database and populate with initial sample data. Run created scripts for downloading product images.
2. 4/5/23
    - Shehab worked on login/register forms + styles
    - Jed made an initial backend commit which was pushed
3. 4/6/2023
    - Shehab worked on login/register form logic.
    - Jed completed authentication endpoints
    - Justin fixed up ER diagrams
4. 4/7/2023
    - Shehab worked on auth context (connected register, login, logout functions to backend)
    - Jed made the computer parts database model
5. 4/9/2023
    - Shehab made the conditionally rendered profile menu button.
    - Jed finished the remainder of database models (computer parts, part comments)
    - Justin worked on web scraping data for the database by writing Python scripts based on Selenium
    - Brandon setup development environment and approved user wireframes (from both employee and owner pov)
    - Jed and Justin met today to discuss web scraping and database design.
    - Shehab and Brandon discussed sprint plans for the client
    - Jed shared status updates on the backend and web scraping

6. 4/17/2023
    - Jed and Justin discussing logistics for web scraping
    - Discussed how the webscraped data will be formatted and how it will be preprocessed when presented
7. 4/25/2023
    - Brandon, Jed, Justin, and Shehab did a final quality check of the report before submitting it.

### 4.2 Team Concerns

1. None so far. All work is being delegated well. The team is on track because of the Notion page we have set up and our frequent and honest communication and meetings.
2. Our only concern lies with the speed at which we can work. Due to other responsibilities, we are finding it difficult to give full focus to all aspects of this project at the moment. We will work on coordinating our time better to allow for more progress on the project.
3. Now that we are a bulk of the way done with the project, our previous concerns are mitigated. All we want to do now is continue at the pace we are going at. No other concerns.

# 5 Design Diagrams

## 5.1 ER Diagram

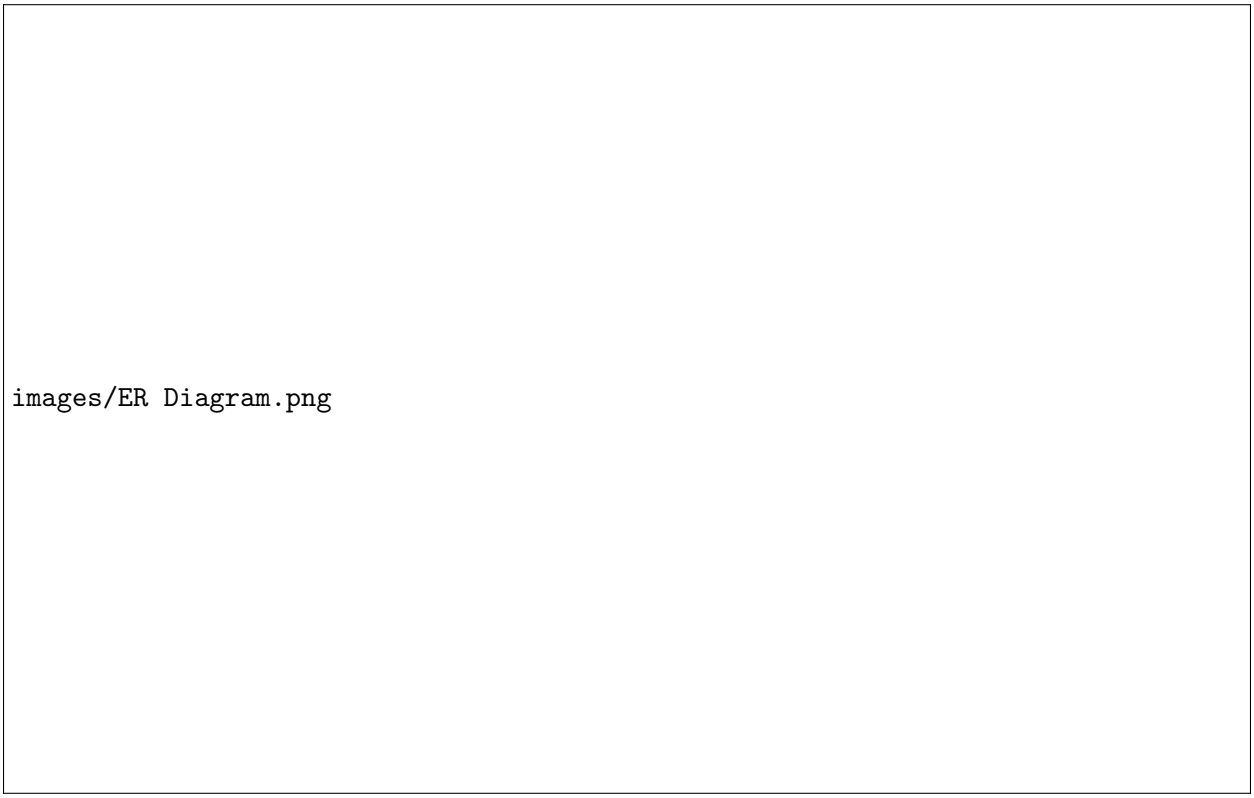


Figure 1: ER Diagram

## 5.2 Use Case Diagram


Figure 2: Use Case Diagram

### 5.3 Collaboration Class Diagram


Figure 3: Collaboration Class Diagram

## 5.4 Sequence Class Diagrams


Figure 4: Login Sequence Class Diagram

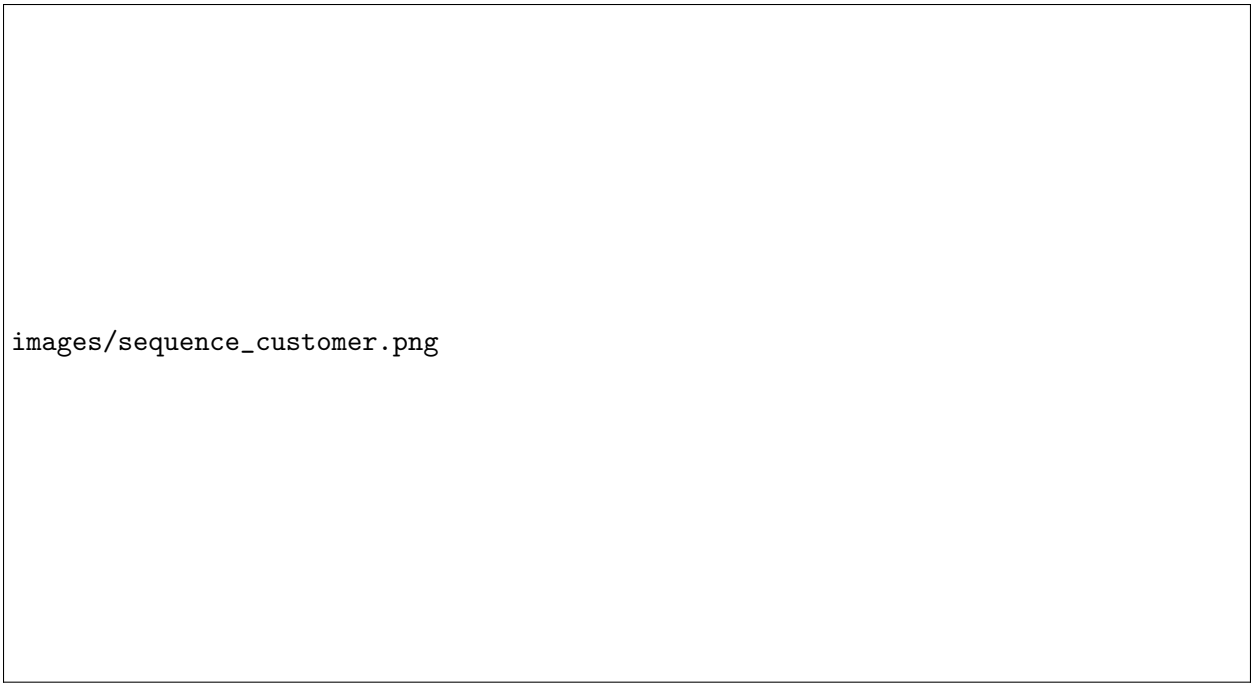


Figure 5: Customer Sequence Class Diagram

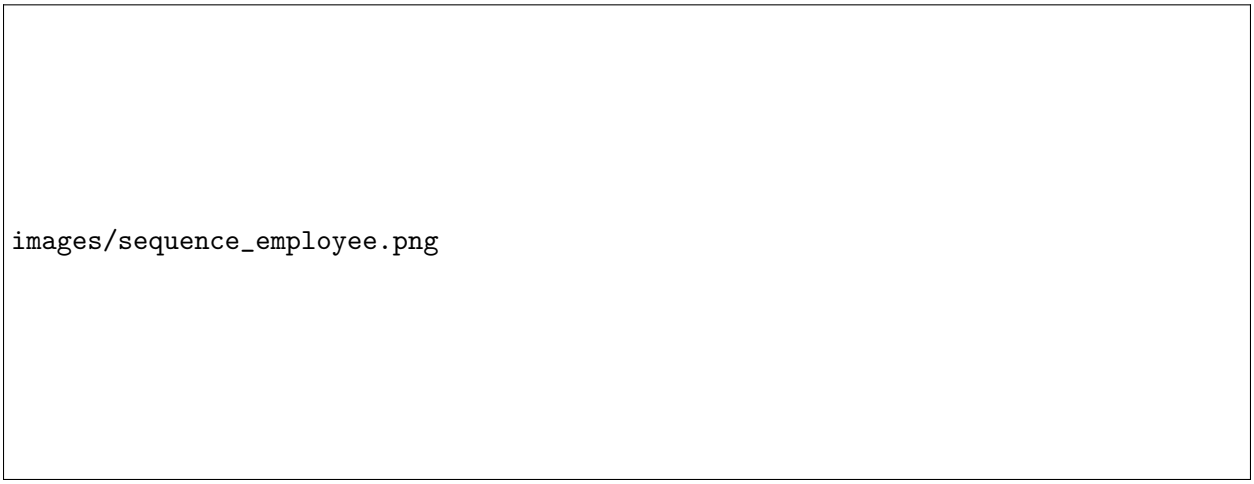


Figure 6: Employee Sequence Class Diagram

## 5.5 Petri Nets


Figure 7: Login Petri Net Diagram

images/petri-net_customer.png

Figure 8: Customer Petri Net Diagram

images/petri-net_employee.png

Figure 9: Employee Petri Net Diagram

# 6 Detailed Design

## 6.1 Authentication Endpoints

### 6.1.1 Register User

```
@AllowAnyPerms
def register(request):
    data = request.data

    username = data['username']
    email = data['email']
    first_name = data['first_name']
    last_name = data['last_name']
    password = data['password']
    re_password = data['re_password']

    user_type = data['user_type']

    if user_type is neither "customer" or "employee":
        return HTTP_400

    if password != re_password:
        return HTTP_400

    if username already exists:
        return HTTP_400

    if email already exists:
        return HTTP_400

    if user_type == "customer":
        Users.create_customer(username, email, first_name, last_name,
            password, re_password)
    else:
        Usdres.create_employee(username, email, first_name, last_name,
            password, re_password)

    return
```

### 6.1.2 Retrieve Current User

```
def retrieve_me(request):
    user = request.user
    return Users.get(user=user)
```

#### 6.1.3 Login User

```
def login_user(request):
    data = request.data
    username = data['username']
    password = data['password']

    if username doesn't exist:
        return HTTP_404

    if passwords don't match:
        return HTTP_401

    if user is inactive:
        return HTTP_401

    return new refresh and access JWT Token
```

#### 6.1.4 Logout User

```
def logout(request):
    refresh_token = request.data['refresh']
    token = RefreshToken(token)
    token.blacklist()
```

### 6.2 User Endpoints

#### 6.2.1 Retrieve User List

```
def retrieve_users(request):
    user = request.user

    if user is not an owner:
        return HTTP_401

    user_list = Users.filter(is_superuser=False)
    return user_list
```

#### 6.2.2 Retriever Customer List

```
def retriever_customers(request):
    user = request.user
    activated = request.query_params.get('activated')

    if user is not an owner or an employee:
        return HTTP_401
```

```
    if activated parameter doesn't exist:
        customer_list = Users.filter(is_customer=True)
        return customer_list

    if activated == "true":
        customer_list = Users.filter(is_customer=True, is_active=True)
        return customer_list

    if activated == "false":
        customer_list = Users.filter(is_customer=True, is_active=False)
        return customer_list
```

#### 6.2.3 Retriever Customer Detail

```
def retriever_customer_detail(request, id):
    user = request.user

    if user is not owner or an employee:
        return HTTP_401

    customer = Users.get(id=id)

    if customer doesn't exist:
        return HTTP_404
    if customer is employee:
        return HTTP_400

    return customer
```

#### 6.2.4 Retrieve Blacklisted Users

```
def retrieve_blacklist(request):
    user = request.user

    if user is not an owner:
        return HTTP_401

    blacklist = Users.get(blacklisted=True)
    return blacklist
```

#### 6.2.5 Blacklist User

```
def blacklist_user(request, id):
    user = request.user
```

```
        if user is not an owner:
            return HTTP_401

        user_to_blacklist = Users.get(id=id)

        if user_to_blacklist doesn't exist:
            return HTTP_404

        if user_to_blacklist is an owner:
            return HTTP_400

        if user_to_blacklisrt is blacklisted:
            return HTTP_400

        user_to_blacklist = Users.set(user_to_blacklist, blacklisted=True)
        return
```

#### 6.2.6 Activate a User

```
    def activate_user(request, id):
        user = request.user

        user_to_activate = Users.get(id=id)
        if user_to_activate doesn't exist:
            return HTTP_404

        if user_to_activate is active:
            return HTTP_400

        if user is not an owner or an employee:
            return HTTP_401

        user_to_activate = Users.set(user_to_activate, is_active=True)
        return
```

#### 6.2.7 Add Balance

```
    def add_balance(request):
        user = request.user
        card_number = request.data['card_number']
        balance = request.data['balance']

        if balance < 0:
            return HTTP_400
```

```
        if not card_valid(card_number):
            return HTTP_400

        user.balance += balance
        return
```

## 6.3 Product Endpoints

### 6.3.1 Get All Products

```
    def retrieve_products(request):
        return Products.all()
```

### 6.3.2 Retriever Product Detail

```
    def retrieve_product_detail(request, id):
        return Products.get(id=id)
```

### 6.3.3 Add Product to Shopping Cart

```
    def add_product_to_shopping_cart(request, id):
        user = request.user

        if user is not a customer:
            return HTTP_401

        product = Product.get(id=id)
        if product doesn't exist:
            return HTTP_404

        user.shopping_cart.add(product)
        return
```

### 6.3.4 Remove Product from Shopping Cart

```
    def remove_product_from_shopping_cart(request, id)
        user = request.user

        if user is not a customer:
            return HTTP_401

        product = Product.get(id=id)
        if product doesn't exist:
            return HTTP_404
```

```
        if product not in user.shoppping_cart:
            return HTTP_404

        user.shopping_cart.remove(product)
        return
```

## 6.4 Custom Build Endpoints

### 6.4.1 Verify Custom Build

```
    def verify_custom_build(request):
        items = request.data

        if items.total_power_output > items.psu.wattage:
            return False

        if items.cpu not in items.motherboard.chipset:
            return False

        if length(items.ram) > items.motherboard.ram_slots:
            return False

        if items.motherboard.form_factor not in
            items.case.supported_motherboards:
            return False

        return True
```

### 6.4.2 Add Custom Build to System

```
    def add_custom_build(request):
        user = request.user
        items = request.data

        if items are not compatible:
            return HTTP_400

        CustomBuild.create(build_maker=user, items=items)
        return
```

### 6.4.3 Add Rating to Custom Build

```
    def add_rating_custom_build(request, custom_build_id):
        user = request.user
        rating = request.data['rating']
```

```
        custom_build = Products.get(id=custom_build_id)
        if custom_build.ratings has 5 >
```

## 6.5 Order Endpoints

### 6.5.1 Submit Order

```
    def submit_order(request):
        user = request.user
        products = user.shopping_cart.items()
        total_price = user.shopping_cart.total_price

        address = request.data['address']

        if total_price > user.balance:
            if user is not Anonymous:
                    user.warnings += 1

                    if user.warnings > 3:
                        user.blacklisted = True

                return HTTP_400

        Orders.create(customer=user, address=address,
            items=products, total_price=total_price)
        return
```

### 6.5.2 Retrieve User Orders

```
    def retrieve_orders(request):
        user = request.user

        if user is not a customer:
            return HTTP_401

        orders = Orders.get(user=user)
        return orders
```

## 6.6 Comment Endpoints

### 6.6.1 Add Comment to Product

```
    def add_comment(request, product_id):
        user = request.user
```

```
if no user:
    user = "Anonymous"

product = Products.get(id=product_id)

if product doesn't exist:
    return HTTP_404

comment = request.data['comment']

if comment.contains_profanity():
    if user is not Anonymous:
        user.warnings += 1

        if user.warnings > 3:
            user.blacklisted = True

    return HTTP_400

products.add_comment(comment, user)
return
```

# 7 User Interface Examples

## 7.1 Home


Figure 10: Home

## 7.2 Computer Parts

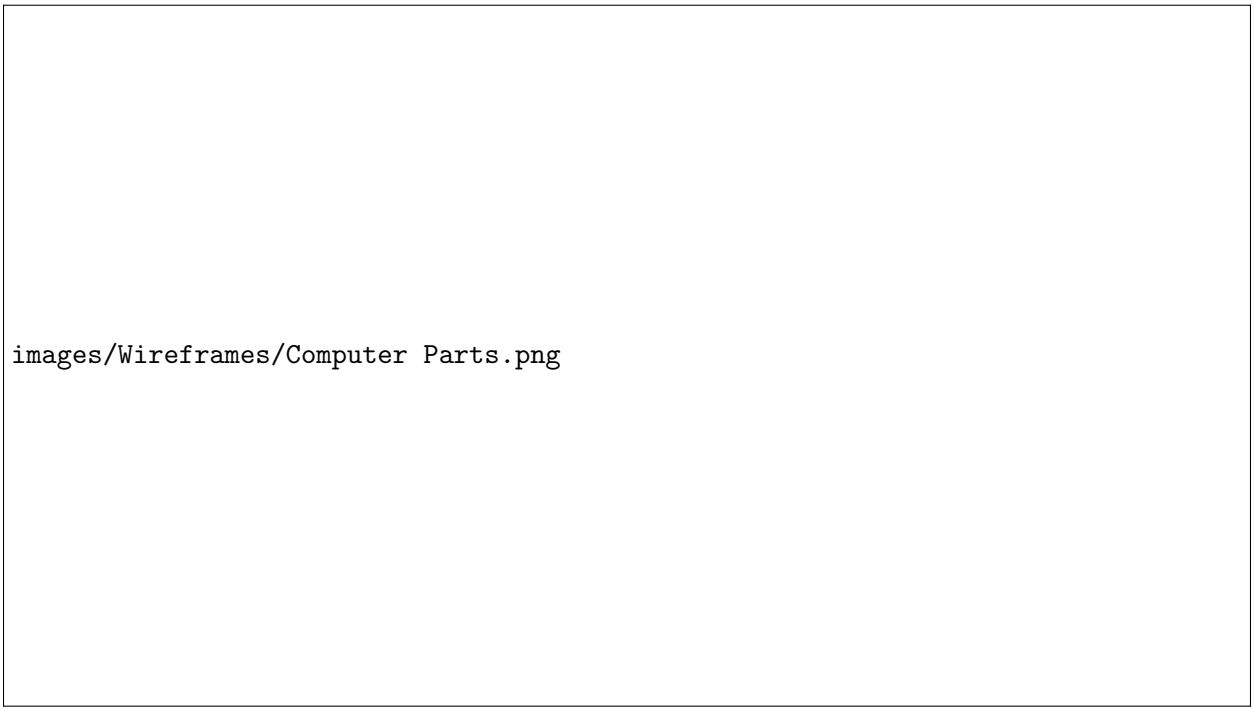


Figure 11: Computer Parts

### 7.3 Featured Builds

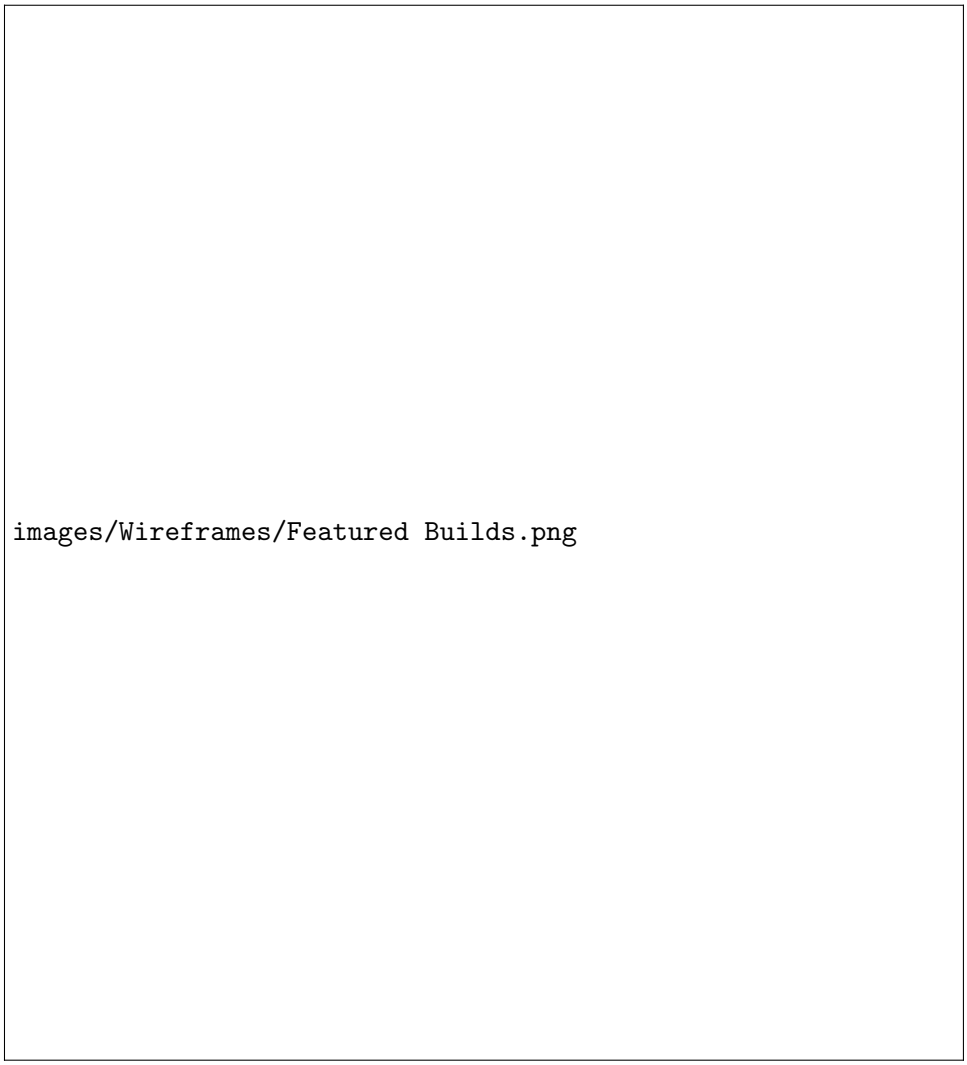


Figure 12: Featured Builds

### 7.4 Sign In

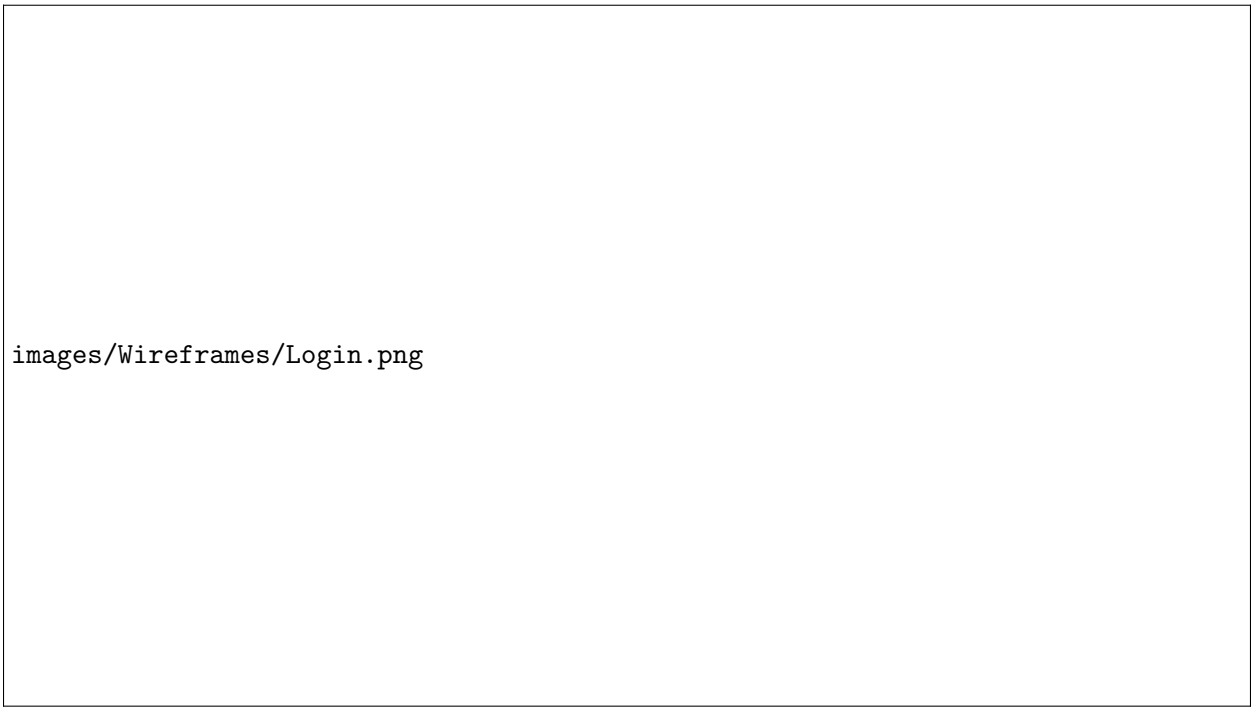


Figure 13: Login Form

### 7.5 Sign Up


Figure 14: Sign Up Form

### 7.6 Approve Account Requests (Employee POV)

images/Wireframes/Account Requests.png

Figure 15: Account Requests Form

### 7.7 Customize Computer Parts List (Customer POV)

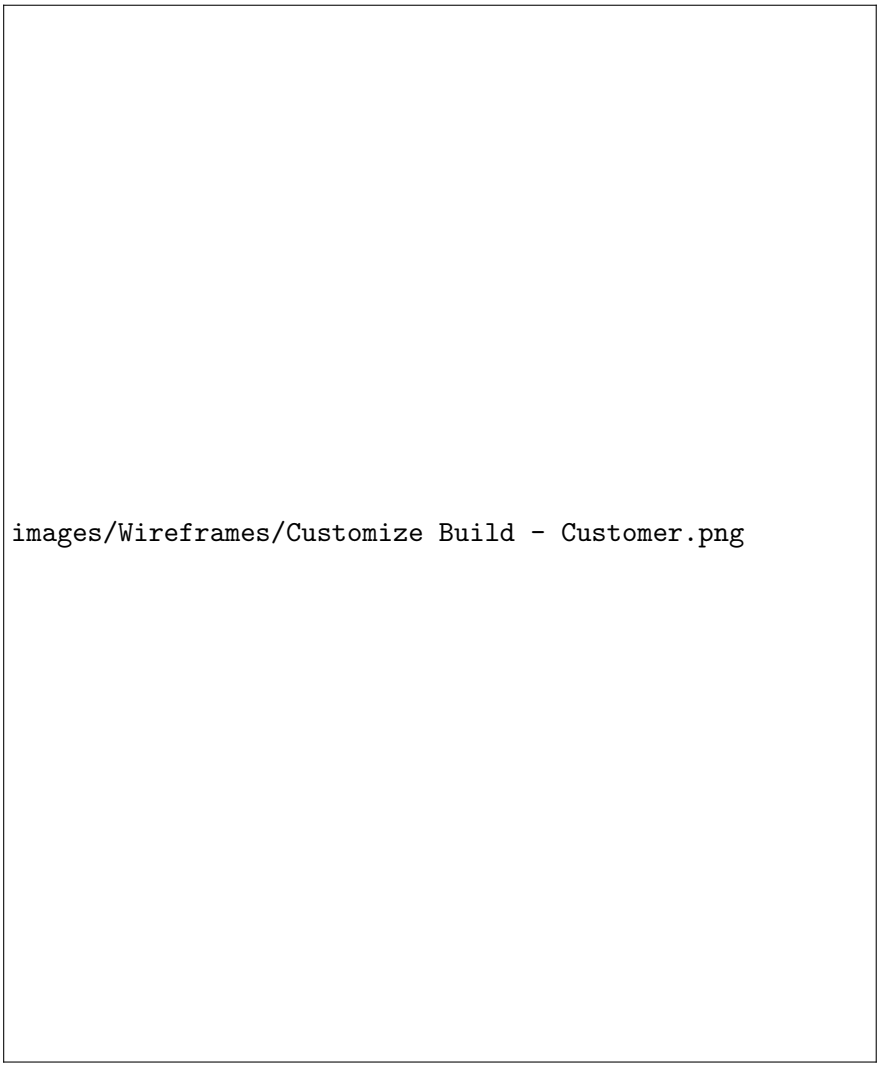


Figure 16: Customize Parts List (Customer)

### 7.8 Customize Computer Parts List (Employee POV)

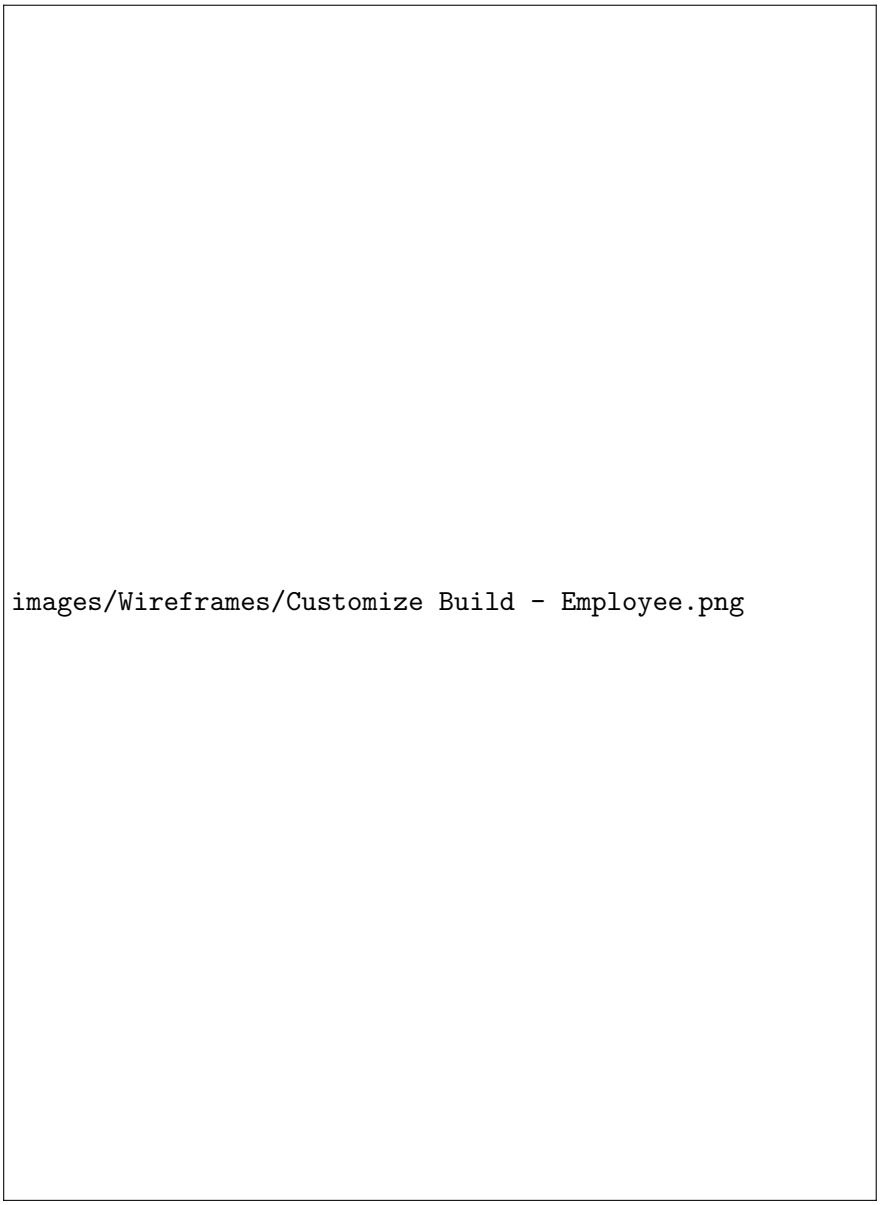


Figure 17: Customize Parts List (Employee)

### 7.9 Shopping Cart


Figure 18: Shopping Cart

# 8 Supporting Information

## 8.1 Links

- **GitHub Repository** - $https://github.com/brandonlam4237/CSC322_Project$