Software Design Specification

Software Engineering

The Interior House

An Online Marketplace for Interior Design Products

1. Vibha Puthran (01FB16ECS441)
2. Vishal S Rao (01FB16ECS450)
3. Vishnu V Singh (01FB16ECS451)
4. Aditya Lokesh (01FB16ECS461)
5. Ayushi S Mehta (01FB16ECS465)
6. Parul Tebriwal (01FB16ECS475)
7. Sanjana U (01FB16ECS478)

Document Owner

Name: Vishnu V Singh

SRN: 01FB16ECS451

Project Name: Interior Marketplace

Email: vishnuvijaysingh@gmail.com

Name: Ayushi S Mehta

SRN: 01FB16ECS465

Project Name: Interior Marketplace

Email: mehtayushi@gmail.com

Index

Document Outline

Document Description

Introduction

This document is the design report for a web-based Interior Marketplace, The Interior House.

This is mainly about the ‘how to do’ and also will help provide an insight into the whole system design and implementation of the online marketplace. This software has the following three main components:

1. Implement the different types of user – Administrator, Customer, and Vendors.
2. Implement a shopping cart for checkout and payment procedures.
3. Management of products and orders.

This design document mainly consists of Class Design, Internal Data Structures, Architectural design, and Testing. The main purposes of this design document are listed below:

1. A precise understanding of the requirements and constraints related to the programming language, and User Interface.
2. System decomposition into manageable units or modules
3. An abstraction of the system implementation with the help of classes
4. Provide a basic outline of the User Interface of the online marketplace.

This report is the result of the design phase. The marketplace will be implemented using Javascript, Python (Flask) as the programming language. MongoDB database will be used to store vital user, product and order information.

System Overview

Module	Description
Customer Login Form	The starting page asks for login credentials and then they are directed to the appropriate web page.
Browse and Search	Customers can browse the products the vendors have listed and can also search by category, price, name and price.
Cart	Customers can add/remove items in their cart and also apply any coupon code if applicable.
Payment Form	Checks out to a third party payment form and once the payment is successful, the database is updated.
Chat Portal	The vendor and the customer can interact using the chat portal to address any concerns.

Design Considerations

Assumptions and Dependencies

- Lucidchart and Draw.io is used to build the use case diagram.

General Constraints

- Must be coded efficiently enough to run well on provided server hardware.
- Client-side code and/or web pages must be able to run efficiently on low-end hardware.
- The database will be created and maintained in a way that makes it reasonable and of manageable size.
- Must write secure code and the system should be capable to handle any security threats.

Goals and Guidelines

- Follow the KISS principle.
    - The eight requirements that identify for a good design that are well structured, simple, efficient, adequate, flexible, practical, implementable and standardized are the guidelines to create this design.
- Working, looking, or "feeling" like an existing online application.
- Emphasis on Speed vs Memory Use.

The goal of this project is to deliver the product completed on time. Use all the recommended models in the design document during coding.

Development Methods

The design method of the system is an important start. The used design method should help the designer to produce a system that is structured in a consistent way. The use of a design method both helps with defining the chosen architectural form and also establishes a set of common standards, criteria and goals for use by the team.

The Data-Flow diagram can be one of the design diagrams used in our project. The DFD is mainly used for describing a very problem-oriented view of the workings of a system. It provides a description based on modelling the flow of information around a network of operational elements, with each element making use of or modifying the information flowing into that element.

We would like to apply more viewpoints to be able to define the system better for the developers. However we will concentrate on the constructional and the behavioral viewpoints. We will define class diagrams which is a core concept of the object model that is centered upon the relationships that involve classes and any objects that are created from these. Identification of candidates for classes is one of the primary activities in object-oriented practices.

Architectural Strategies

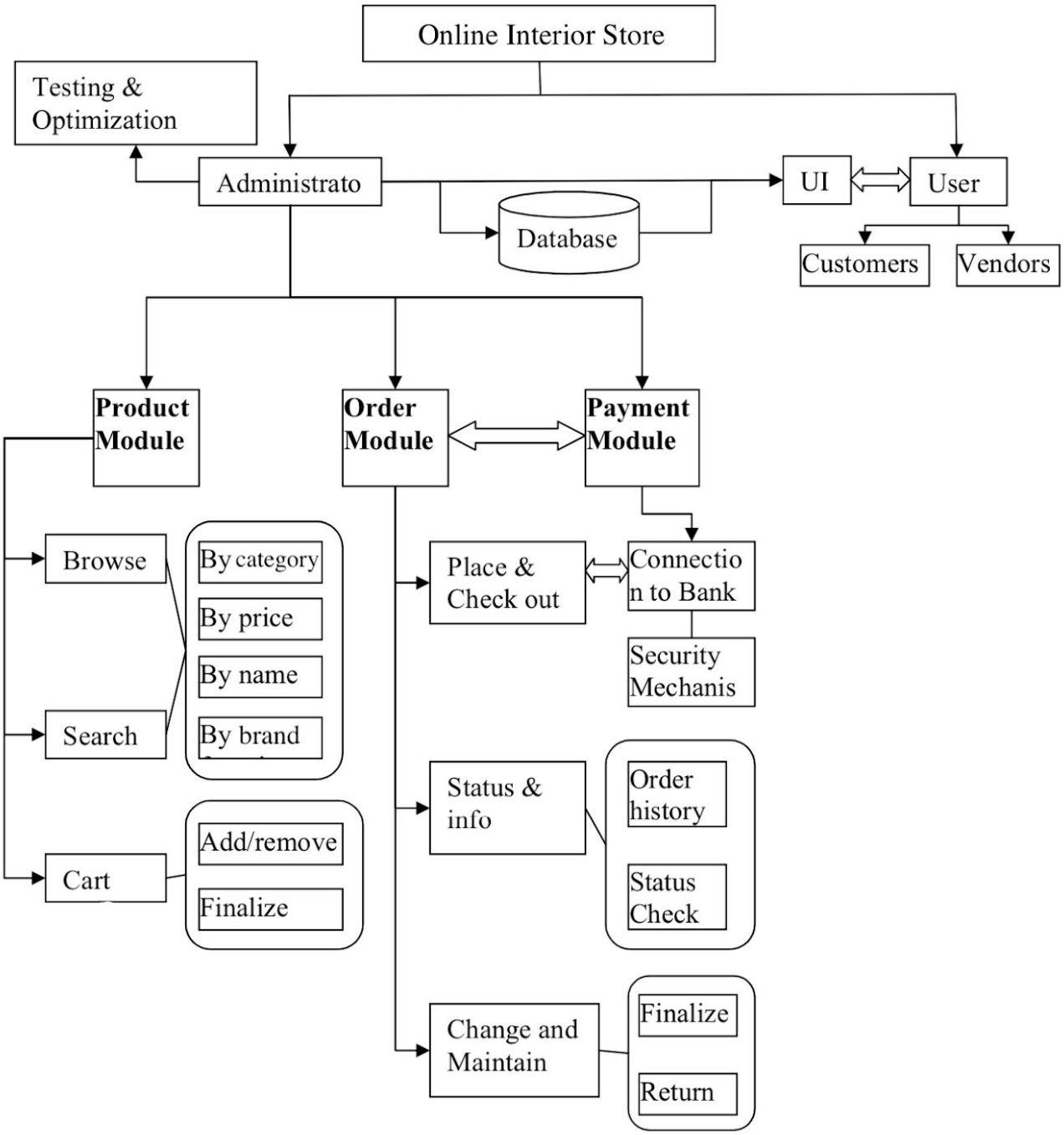
Architecture is the set of decisions that must be made at the enterprise level before specific applications are designed and built in order to provide conceptual integrity and sanity across the systems.

Architecture not only divides the system, but it also divides the roles and responsibilities of those who work with the system into separate organizational concerns and disciplines that are conceptually tractable and can be effectively managed. In our project, behavioral and functional viewpoints will be used to design it. Additionally, constructional viewpoint strategy can also be used.

- Use of a particular type of product (programming language, database, library, etc. ...)
    - There will be a database involved in this system. MongoDB will be required.
    - Python(Flask) will be used to build the system.
- Reuse of existing software components to implement various parts/features of the system
    - For additional features re-use of the forms is possible.
- Future plans for extending or enhancing the software.
    - This interior marketplace is the basic prototype. More additional features can be added if needed.
- Error Detection and Recovery
    - Error detection and recovery will be done. To be able to separate error-handling code from the regular code, we will add exception errors in the code.

System Architecture

This part mainly focuses on the overall system design architecture, which describes the internal necessary requirements and structures during the design process for the system designers.



Except for UI, testing, and optimization, the interior marketplace is divided into 3 major modules as below. Each of them implements a primary utility for either vendors or customers or both.

- Product Module
    - Some of its sub-modules (or functions) are exclusive for customers. This module requires designers to implement browse and search function for our web visitors and is supposed to be as friendly as possible and as reliable as possible (i.e. fast, barely break down and recovery). Browse/search should be by different categories. Then cart is necessary for users' convenience and they should be able to modify any selected items in the cart list.
- Order Module
    - It contains three major roles, check-out, which connects customers, vendors, banks, and administrators. Order information/status check, and order maintenance, which allows users to act (cancel/return, etc.) on their placed orders to some extent.
- Payment Module
    - Provides payment methods (i.e. various bank cards or other commercial tools) and provides a security mechanism

All the modules above should be able to connect to the MongoDB database.

Policies and Tactics

- Programming Methodology
    - All variables and functions will be named using camel casing notation.
    - The code will follow OOPs concepts.
    - Code will not be repeated multiple times, i.e, functions will be used appropriately. Also, functions will not be created unnecessarily.

- Languages Used
    - Frontend - HTML, CSS, Javascript.
    - Backend - Python.

- Database Used
    - MongoDB.

- Design Patterns
    - We are using AJAX concepts for better UX.
    - Periodic Refresh using AJAX.
    - AJAX will be implemented using XMLHttpRequest.
    - Communication between client and server will be using REST APIs.

- Testing
    - We will have unit tests for each REST API.
    - We will follow the Test Driven Development methodology from XP for implementing a thoroughly tested code.

- Maintainability
    - We will split the web application into microservices so that maintaining and updating a particular feature of the application is made easier.

Testing

Unit Testing

- Log in
    - Identification and Password properly initiated, encrypted, and validated
    - Checking for uppercase, lowercase, numbers, special character in ID and password
    - Either of ID and password not blank.
    - Checking for weak passwords

- Start shopping
    - Correctly validated before starting shopping
    - Not possible to check out products which is already in progress
    - Displaying list of all available products
    - Checking that the newest version products are placed on the first page.
    - Checking for the exact number of products stocks
    - Checking for available coupon information
    - Checking for visibility about the previous search history

- Payment
    - Verify username and card information for security
    - Certification for expiration month and year
    - Certification about CVV number
    - Checking for signature

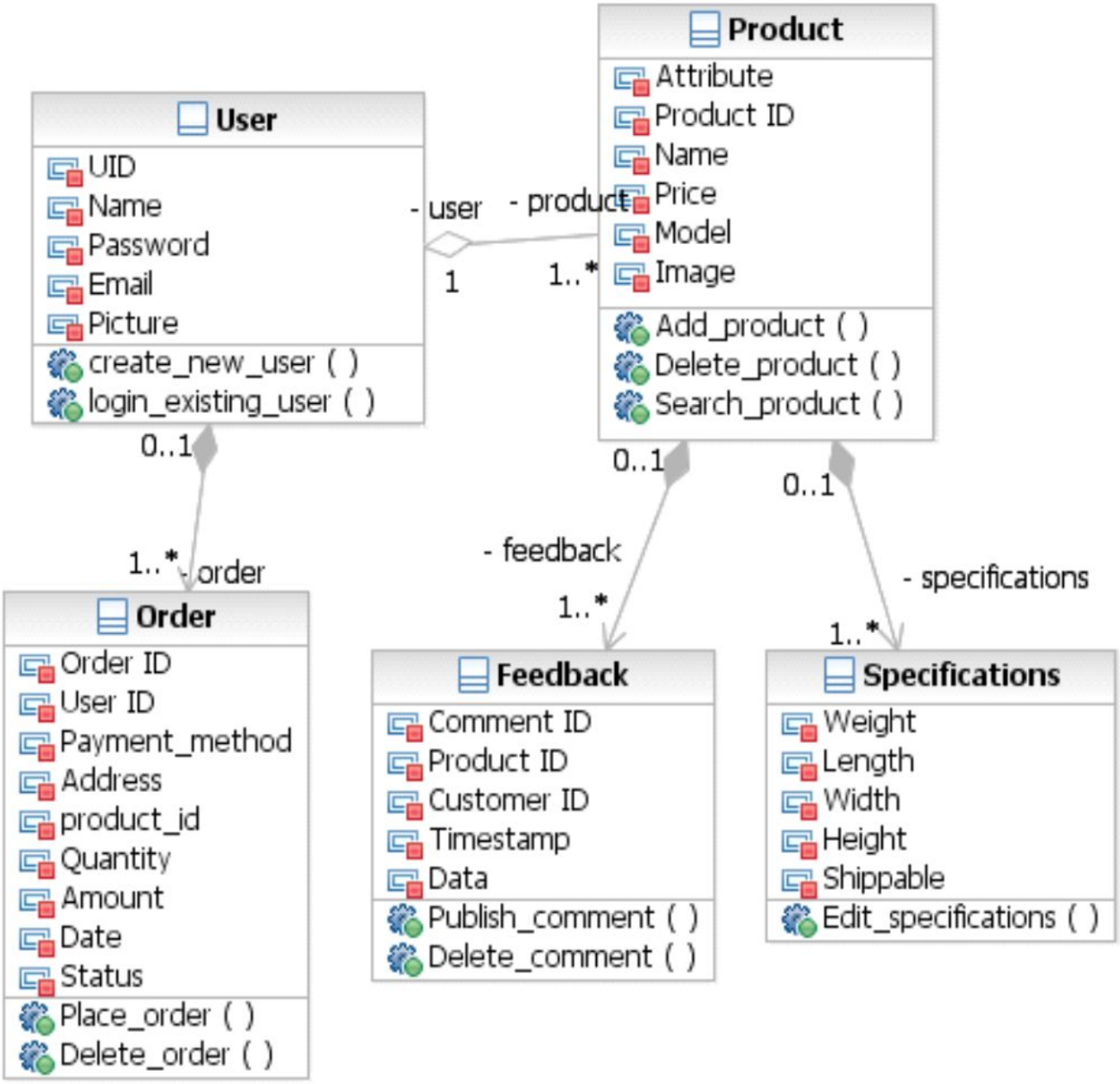
- Shipping
    - Checking for available address information
    - Checking for private information and updated address
    - Checking if shipping charges are added
    - Checking the time for shipment
    - Check for the comment when the consumer will not at home

- Logout
    - Search history saved properly after logout
    - Checking out information saved in the database
    - Customer redirected to the login screen

Detailed System Design

Classification

The class diagram for our system is given below -



Definition

- User Class
    - The user class is used to store the details of all the users. It is used to create a new user or help an existing user login.
    - There are two types of users: admin and a normal user. Every user can place an order or search for a product. The admin can add and delete products as well.

- Order Class
    - The order class contains the details of the placed order. This class is used to place or delete an order. The order class is notified when an order is placed or deleted by a user.

- Product Class
    - The product class contains the details of the products. This class will be notified when the user searches for a product or the admin adds or deletes a product.

- Feedback Class
    - The feedback class contains the details of the comments posted on the various products. This class is responsible for posting comments on products and deleting them by users.

- Specifications Class
    - The specifications class contains the various specifications of the products. This class is used to edit the specifications of the products.

Composition

- User Class
    - **UID**: this is the unique identifier of each user
    - **Name**: the name of the user
    - **Password**: the password of the user
    - **Email**: the email id of the user
    - **Picture**: the picture uploaded by the user

- Order Class
  - **Order ID**: the order ID associated with that particular order
  - **User ID**: the ID of the user who placed the order
  - **Payment method**: the mode of payment
  - **Address**: the shipping address of the user
  - **Product_id**: the ID of the product that was ordered
  - **Quantity**: the number of items that were ordered
  - **Amount**: the cost of the order
  - **Date**: the date the order was placed
  - **Status**: the current status of the order, whether the order was shipped or still in the processing stage

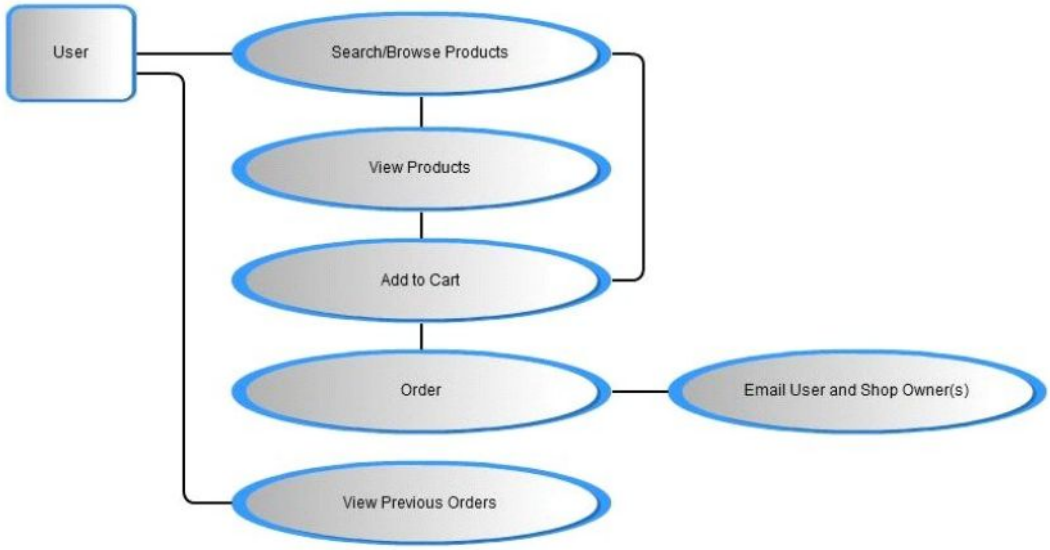
- Product Class
  - **Attribute**: the description of the product
  - **Product ID**: the unique ID of the product
  - **Name**: the name of the product
  - **Price**: the price of the product
  - **Model**: the model number of the product
  - **Image**: a picture of the product

- Feedback Class
  - **Comment ID**: the ID of the comment published
  - **Product ID**: the ID of the product on which the comment is made
  - **Customer ID**: the ID of the customer who made the comment
  - **Timestamp**: the time when the comment was made
  - **Data**: the string that was posted as a comment

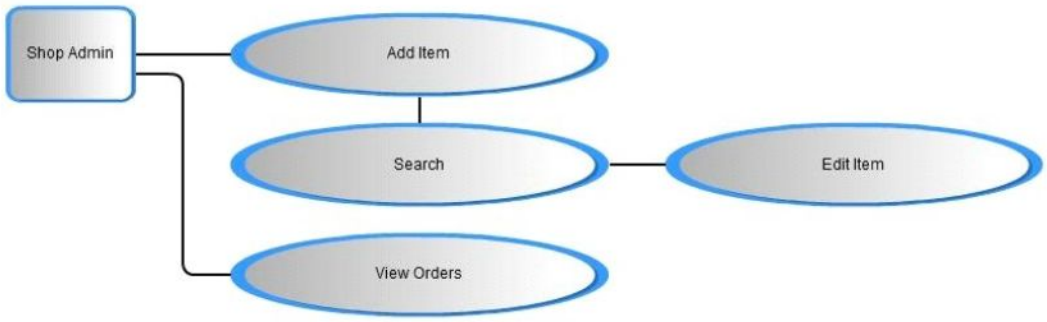
- Specifications
  - **Weight**: the weight of the product
  - **Length**: the length of the product
  - **Width**: the width of the product
  - **Height**: the height of the product
  - **Shippable**: whether the product is shippable or not

DFD

- Users



- Shop Admins



- Administrators

Resources

This is a server, client tool. Most of the functions will be running from the server and managed through the server. All files and information will be saved in JSON format on the server.

Processing

Handling of exceptional conditions should be done in each module. All the scenarios that can cause errors need to be handled and not cause applications to crash. Error detection and recovery will be done. To be able to separate error-handling code from the regular code, we will add exception errors in the code. For example, use the following to print the stack trace.

```
catch (Exception e) {

//Additional messages can be added by the developer if needed.
    ...
}
```

Glossary

- DFD: Data flow Diagram
- XP: Xtreme Programming