Database Application 2014

Introduction

The system which is going to be implemented is a simple online shopping cart for a flight company. Customers can use this system to order tax-free goods before they take specific flights, and then buy these products on the airplanes. In addition to tax-free shopping, customers could also be able to make some special requests through this system (e.g. reserve flight meals). Typically customers should ask the flight company to deliver the ordered products to their specific seats (but I guess it is also possible to book something only with the customer's name). Tax-free products are divided into several categories (e.g. electronics, watches, skin-care and makeup products) and displayed on the system.

The system should support the following functions:

- Login: Customers and Company personals could log into this system with their user accounts and passwords
- Making order: customers who have logged in could book tax-free goods from this system, and choose the time and flight to buy those goods
- Changing and cancelling orders: customers could change or cancel their orders some days before flights take off
- Order display: customers could view all their orders in a displaying page, and company staffs could choose to view all the orders of a specific flight
- Notifications: there will be notifications about the status of customers' order (e.g. orders have been confirmed by the flight company)
- Products information: personels of the flight company could change product information and remove some products from the system

This system will run on/with a common HTTP server (e.g. Apache, Nginx), and all the data will be stored in a PostgreSQL database. The server-side programs will be written in PHP, and some client-side JavaScript files might be used.

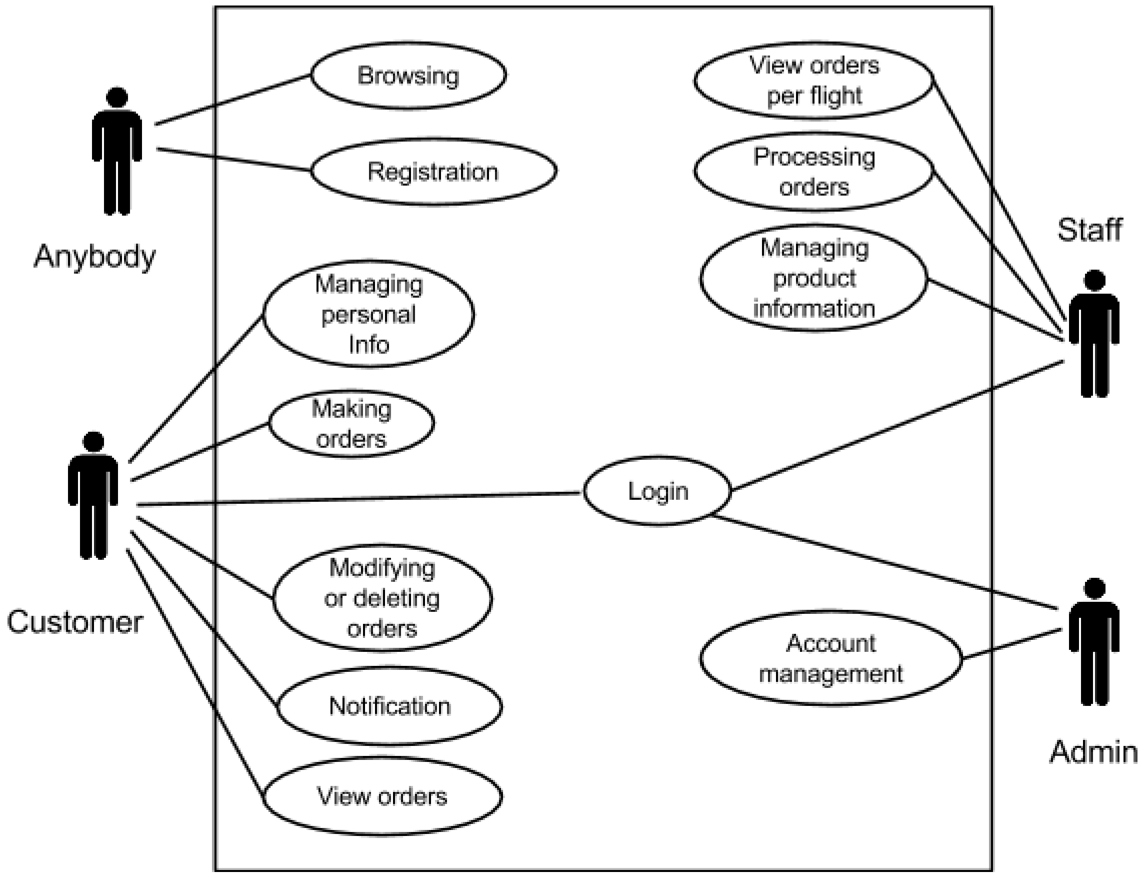
Overview of the system

User Groups

- Anybody: anyone who reads the web page
- Customer: customer is a registered user who could make orders
- Flight company staff: people who are responsible for processing orders and updating product information

- System administrator (optional for this system): super administrator to the system, could delete and add users/staffs

User Case Diagram



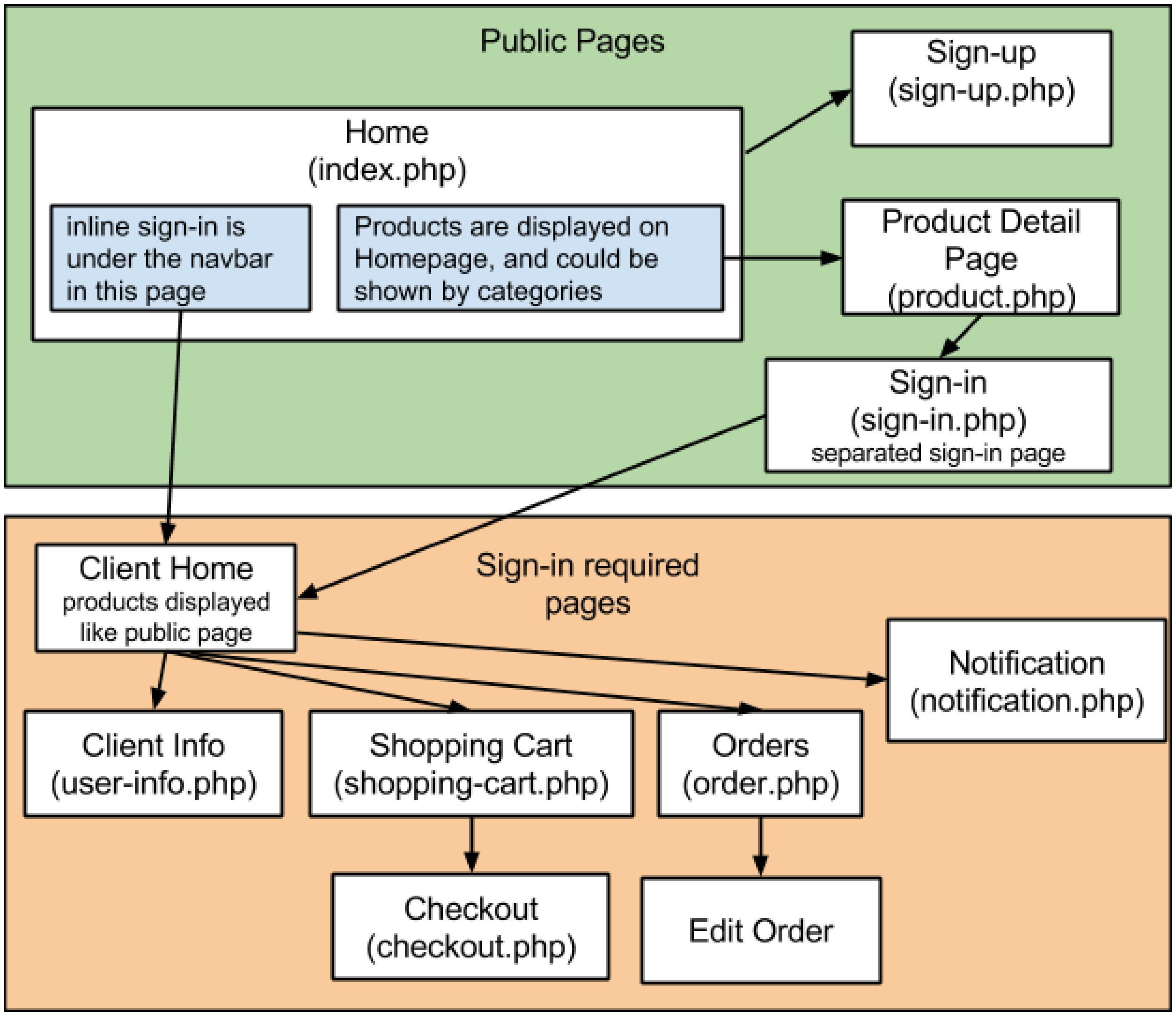
User Case Descriptions

- User cases for Anybody:
  - Browsing: anybody could read company’s general information pages and product pages. They could see all the description of tax-free goods
  - Registration: by registering, the user could become a customer, and then some other services will be available to the user. Anybody could become a customer after completing a registration form via the web page
- User cases for Customer:
  - Login: registered customers could log into the system with their usernames and passwords, and then the following customer user cases will be available for them
  - Managing personal information: customers could store and edit their personal information (e.g. name, contact information) in the system

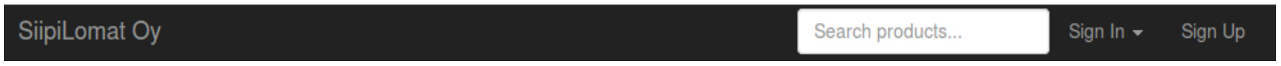
    - Making orders: choose tax-free goods for a specific flight, add them in one shopping cart, and submit the shopping cart as one order. Customers may need to give more detailed information (e.g. seat number, flight number) when they make orders
    - Modifying or deleting orders: customers could edit existing orders, or even delete them some days before the take-off time of the specified flight.
    - View orders: view existing orders made by this customer
    - View and delete notifications: when a order is confirmed by the flight company, the system will send the customer a notification message. The message will be stored in a message box under the customer's account, and the user could view and delete them
- User cases for Staff:
    - Login: log into this system with staff username and password, and then the following staff user cases will be available
    - Processing orders: when a new order is submitted, the staff could see it and confirm it. Order confirmation means that the company has finished processing phase, and been ready to deliver the goods to the customer. Then the system will send a notification to the user
    - Managing product information: company staff could add, edit or delete some products
    - View orders per flight: staff could choose to view all the orders for a specific flight
- User cases for Administrator:
    - Login
    - Account management: add or delete customers and/or staffs

Outline of the User Interface

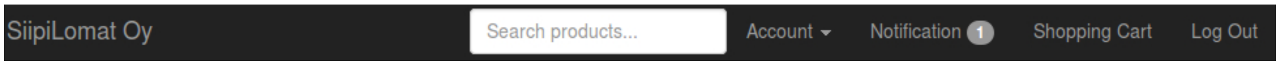
The products are displayed in the home page, and users could choose to see a particular category of goods on this page. Some basic navigation links are added to this home page (e.g. sign-in, sign-up). The home page is public for everyone, but only when users sign in, they could add products to their shopping carts and manage their orders. The “Client Home” refers to the home page for users who have logged in, and it is almost the same to the public home page, but some new navigation elements would show up (e.g. shopping cart, order management, user info management). The “Product Detail Page” could also be used both for public and for users who have logged in. The following picture shows some basic components of the user interface.


Public Pages
Home (index.php)
inline sign-in is under the navbar in this page
Products are displayed on Homepage, and could be shown by categories
Sign-up (sign-up.php)
Product Detail Page (product.php)
Sign-in (sign-in.php) separated sign-in page
Sign-in required pages
Client Home products displayed like public page
Notification (notification.php)
Client Info (user-info.php)
Shopping Cart (shopping-cart.php)
Orders (order.php)
Checkout (checkout.php)
Edit Order

For example, the following pictures show the basic navbar for Home and Client Home.



Navbar for Home without login



Navbar for Client Home (after login)

The System Data

System information content

User Info

Each user should have one profile like this

Attribute	**Data Type**	**Description**
Email	String, max 64 characters	user’s email address, this is also used as the name of user account for sign up
Password	String, max 64 characters	sign up password
Real Name	String, max 64 characters	User’s real name, e.g. Albert Einstein
Type	String, max 10 characters	customer, staff or other

Product

Attribute	**Data Type**	**Description**
Name	String, max 128 characters	product name, e.g. “Luke Skywalker Bird Plush Toy”
Picture	Text	image url address
Brand	String, max 64 characters	brand name of the product, e.g. “Fazar”
Category	String, max 64 characters	the category that the product belongs to, e.g. “Food”, “Fun”
Price	Decimal	price in euro
Description	Text	short description of the product

Item in Shopping Cart

The items in user’s shopping cart. These items have not been submitted as one order yet.

Attribute	**Data Type**	**Description**
User	depend on the type of user ID	whose shopping cart this item belongs to
Product	depend on the type of product ID	product id
Quantity	Decimal	numbers of the product in the shopping cart

Order

Attribute	Data Type	Description
User	depend on the type of user ID	who made this order
Flight No	String, max 10 characters	flight identical number
Flight Date	Date	flight date, e.g. 2014-10-21
Flight Seat	String, max 10 characters	seat number on the flight, could be left as NULL
Date	Date	the date when the order is submitted to the system
Request	Text	special requests for the order
Status	String, max 20 characters	Processing, or Confirmed

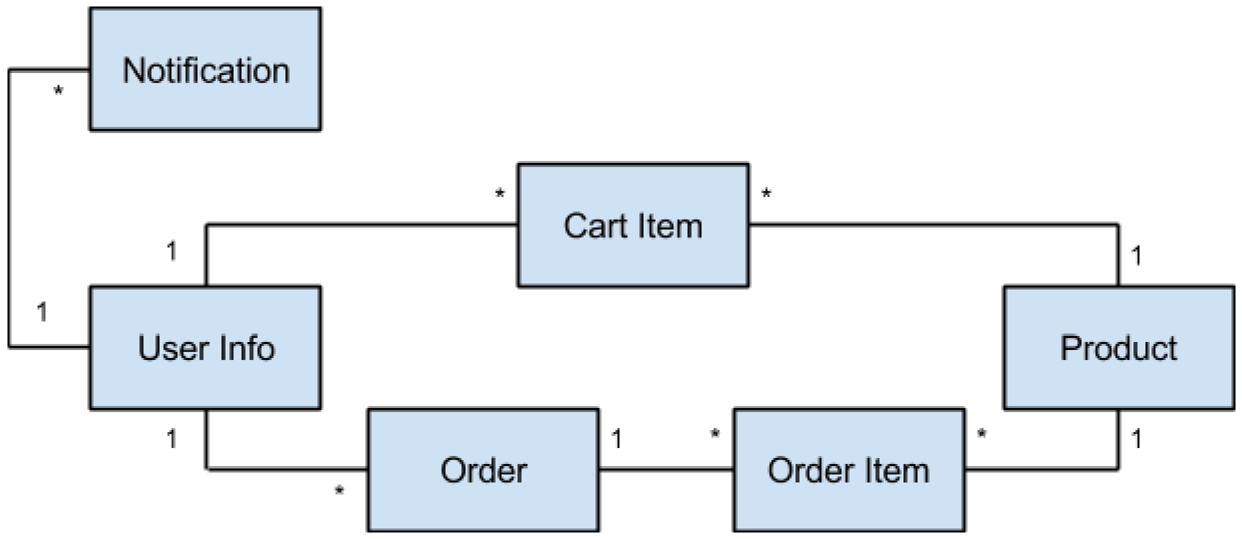
Order Item

Attribute	Data Type	Description
Order	depend on the type of order ID	which order this item belongs to
Product	depend on the type of product ID	product id
Quantity	Decimal	number of the product in the corresponding order

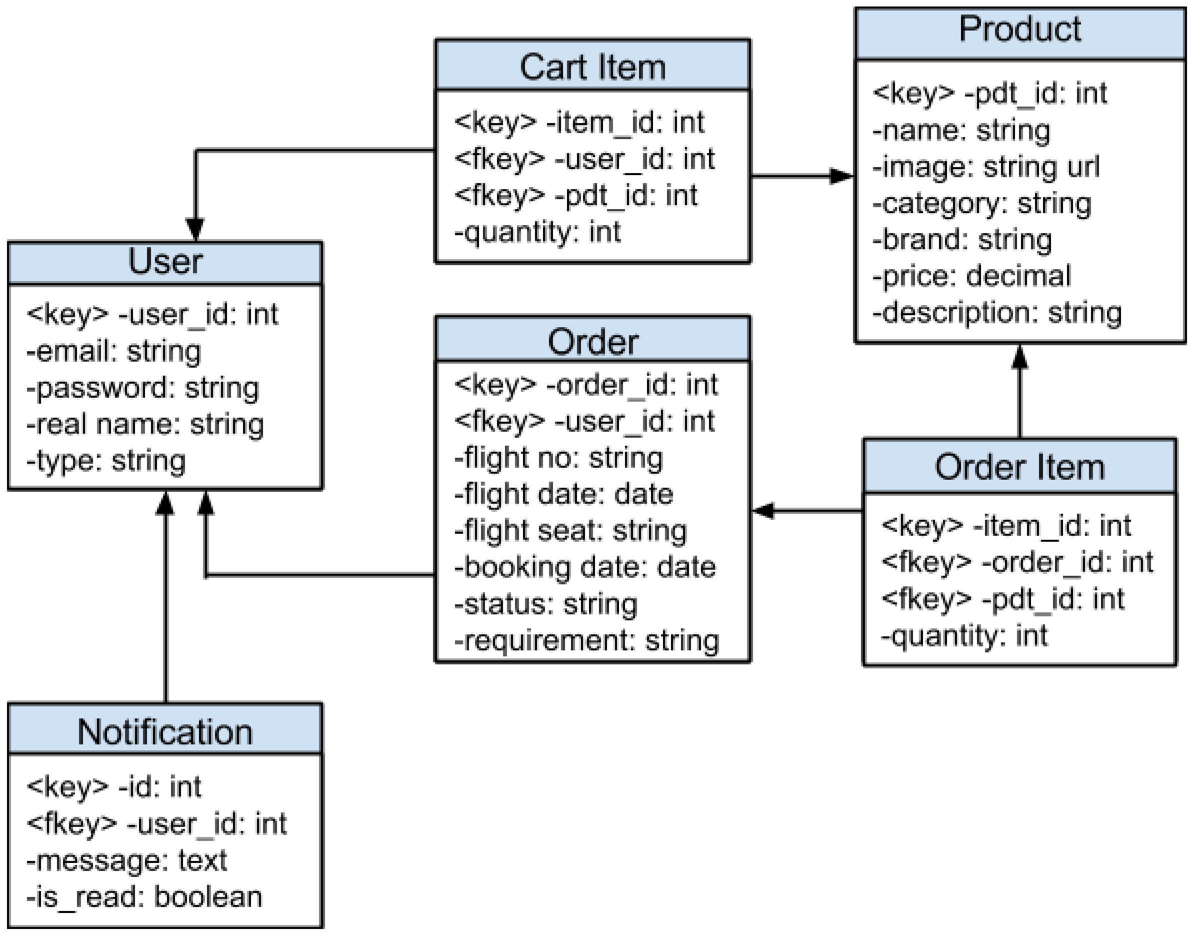
Notification

Attribute	Data Type	Description
User	depend on the type of User ID	which user this notification message belongs to
Message	String (Text)	Notification message
Has_been_read	Boolean	Whether the message is read by user

Concept Relationships



Relational Database Scheme



The general structure of the system

The whole application follows MVC model, and the basic file structure is like this:

```
database application/
|-js/
|-css/
|-sql/
|-views/
|  |-navbar.php
|  +-......
|-lib/
|  |-models/
|  |  |-order.php
|  |  +-......
|  |-db.php
| |-view_components.php
| +-......
|-index.php
+-......
```

All the database table related classes and functions are put in folder lib/models/, and all the HTML view templates are located in folder views/ (some generic view functions are put in lib/view_components.php which also based on those view templates). System controllers are separated with their functions and put in the top level of the directory. The homepage entry is index.php. All of the controllers are named with their main functionalities. For example, orders.php handles all the stuff related to processing orders (add, edit, delete, and confirm), and product_management.php is responsible to manage products in the system (add, edit, and delete).

Some common utilization PHP functions are put in folder lib/ (including models classes). For example, database connection function (create PDO database connection, and return this connection for future use) is in lib/db.php. Other useful files, like CSS classes are JavaScript functions are also organized into specific folders on the top level.

Installation

To deploy this application to a new environment, you should at least have a working HTTP server (e.g. Apache2, Nginx, or other staff you like) and a PostgreSQL server. Then just follow these steps to finish the deployment:

- copy the whole application folder from GitHub to your server, give all the directories at least ‘x’ right (enterable), and all the other php files with ‘readable’ right to your HTTP server.
- run sql script to create required tables
  ```
  cd sql
  cat create-tables.sql | psql
  ```
- Perhaps you would also like to specify some system parameters (now only the PostgreSQL connection parameter is supported). Open the file /lib/config.php, edit variable $PDO_PGSQL. For example:
  ```
  $PDO_PGSQL = ‘pgsql:host=localhost;port=5432;’
               . ‘dbname=test;user=abc;password=123’;
  ```

  If you are using our department’s “Users” server, there should be no need to change the default value.

Now the application should be working :)

Instructions for Startup/Use

Now the application is running on 'Users' server, and the visiting URL is

http://yfliu.users.cs.helsinki.fi/database-app/.

For ordinary customers, there should be no special requirement for using the application. The shopping cart and order system are similar to common online shopping sites (e.g. Amazon), and easy to use. A testing account for customer is customer@test.com, and the password is 123456.

Sign-up for staff user is not open to public. Currently the only available staff account is staff@test.com, and the password is 123456.

Testing and Further Development Ideas

Right now all the key user cases are tested manually (with self-designed functional testing cases), and all of them are working well. However, this application is not verified by using any automatic and TDD (test-driven development) PHP testing framework (like PHPUnit), nor tested by input fuzzing, which could be considered as insufficient from the point of modern agile development. Those topic could be explored in my further study.

In addition, the user case for managing customers' information is not implemented yet. Since it is not a key point for this simple shopping cart system, and not required in this course topic, I select to drop this functionality this time. Also, some other models (e.g. shopping cart, orders, products) have already been implemented with full CRUD operation set, so I guess it is not very necessary to repeat the similar stuff again to customer's information.

Learning Experience

This is my first time to develop a real Web application with its own database. Because of the simplicity of PHP programming, all the stuff goes quite well. PHP is easy to learn and use, and the programming process is also straight-forward. Thanks to the course, I have a better understanding on MVC architecture, and become more familiar with PostgreSQL.

Since this application is not very complex, I just use PHP without any framework. However, this might be a bit challenging for a large-scale Web application. If possible, I will try to explore some

Web development frameworks later (like Rails for Ruby and Django for Python). In addition, automatic testing is still missing for this project.

Database Creation File

The table creation file is put in the ‘sql’ folder of my GitHub repository :)