**Domain Analysis**

**1 Domain Model**

**1.1 Concept Definitions**

**Table 6.1** Concept Definitions

|  |  |  |
| --- | --- | --- |
| **Responsibility** | **Type** | **Concept** |
| R1: Coordinate activities between users, administrators, users and administrators. | D | Controller |
| R2: Save the user's account records in a favorite folder and rate them | K | Customer Profile |
| R3: Display various operational options on the user interface. | D | Interface |
| R4: Prompt the customer to select a product for payment. | D | Controller |
| R5: Store the user-determined order in the database | K | Communicator |
| R6: Wait for the administrator or system to complete the order confirmation | K | Order Queue |
| R7: Manage interactions with the database | K | Connection |
| R8:Record the user's daily browsing history | D | Connection |
| R9:Prevent customers from using unwanted choices | D | Status |
| R10: Allow time to upload related changes to the customer | D | Status |
| R11: Input to receive payment type and amount | D | Payment System |
| R12: Display user's favorite products in the user interface for easy access | D | Interface |
| R13: The system automatically deletes unused items. | D | System control |

**1.2 Association Definitions**

**1.2** Association Definitions

|  |  |  |
| --- | --- | --- |
| **Concept Pair** | **Association Description** | **Association Name** |
| User profile ⇔ Interface | Show user options | Show |
| Interface ⇔ Connection | Get user data from the database | User Action |
| Communicator ⇔ Order Queue | Send the user's order to the system background | system |
| Interface ⇔ Controller | Allow various systems to interact with customers | User Action |
| Communicator ⇔ Connection | Add or modify data in the database | Update |
| Customer Profile ⇔Connection | Get customer related information from the database | Query |
| Controller ⇔ Payment System | Customer completes the final payment process | payment system |

**1.3 Attribute Definitions**

1.3 Attribute Definitions

|  |  |  |
| --- | --- | --- |
| **Concept** | **Attribute** | **Description** |
| User Info | account Username | User's related user name, if no system assigns user to apply for username |
| Account Password | User-designed account name and password |
| Customer's favorite goods | Includes user-provided suggestions |
| Interface | Current user interface | Current feature only applies to current account |
| Controller | Show | Show users the current purchaseable items |
| Confirm Order | After the customer's confirmation, the user confirms his order through payment. |
| Payment Made | Finally, the user pays through the online wallet. |
| Analytic Calc | User order record statistics | Record the user's purchase order record in the user's purchase history for the user to view |
| Order Status | Order Status | Whether the order has been reviewed and whether the steps to review the order have been executed |
| Payment System | Payment Made | The user's payment system, including the network wallet payment used by the user, etc. |

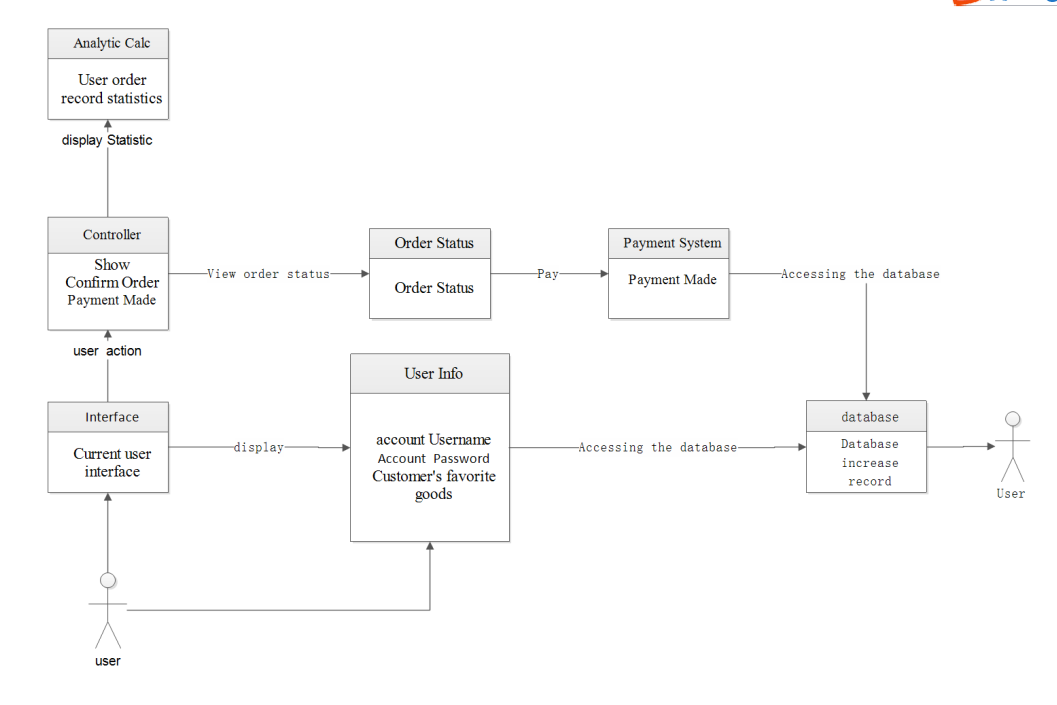
**1.4 Traceability Matrix**

**1.4 Traceability Matrix**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Use Case** | **PW** | **Domain Concepts** | | | | | |
| User Info | Interface | Controller | Analytic Calc | Order Status | Payment System |
| UC-1 | 4 | × | × | × | × |  |  |
| UC-2 | 6 | × | × | × | × |  |  |
| UC-3 | 6 | × | × | × |  |  |  |
| UC-4 | 9 | × | × | × |  |  |  |
| UC-5 | 8 |  | × | × | × | × |  |
| UC-6 | 5 |  | × | × |  |  |  |
| UC-7 | 6 | × | × | × | × |  |  |
| UC-8 | 5 | × | × | × |  | × | × |
| UC-9 | 7 | × | × | × |  | × | × |
| UC-10 | 4 | × | × |  |  | × | × |
| UC-11 | 5 | × | × |  | × |  |  |
| UC-12 | 6 | × | × | × | × | × |  |
| Max PW | | 9 | 9 | 9 | 8 | 8 | 7 |
| Total PW | | 58 | 71 | 62 | 40 | 30 | 16 |

**1.5 Domain Model Diagram**

1.5 Domain Model Diagram



**6.2 System Operation Contracts**

6.2 System Operation Contracts

|  |  |
| --- | --- |
| Name: | Registration |
| Responsibilities: | Users can register their own account password on the website and can log in successfully. |
| Use Case: | UC -1 |
| Exception: | None |
| Preconditions: | User opens the website registration page |
| Postcondition: | nothing worth mentioning |

|  |  |
| --- | --- |
| Name: | Login |
| Responsibilities: | Login |
| Use Case: | UC -2 |
| Exception: | None |
| Preconditions: | The password account is successfully registered. |
| Postcondition: | nothing worth mentioning |

|  |  |
| --- | --- |
| Name: | Manage personal information |
| Responsibilities: | Improve or modify the user's personal information, including name, contact information, shipping |
| Use Case: | UC -3 |
| Exception: | None |
| Preconditions: | The user has successfully logged in and is ready to maintain personal information |
| Postcondition: | Successfully connected to the system database for read/write operations |

|  |  |
| --- | --- |
| Name: | Publish goods |
| Responsibilities: | Want to rent their own products on this platform |
| Use Case: | UC -4 |
| Exception: | None |
| Preconditions: | Support from the system, goods can be reviewed in the background and can be stored in the database |
| Postcondition: | Provide product information feedback. |

|  |  |
| --- | --- |
| Name: | Delete/Modify product |
| Responsibilities: | Want to delete his own published product |
| Use Case: | UC -5 |
| Exception: | None |
| Preconditions: | This feature is available in the system |
| Postcondition: | Provide information feedback. |

|  |  |
| --- | --- |
| Name: | Search and view goods |
| Responsibilities: | Open the website click on the |
| Use Case: | UC -6 |
| Exception: | None |
| Preconditions: | Users need to open the site and successfully log on to their account, open the phone storage rights. |
| Postcondition: | Access to the background database of the website |

|  |  |
| --- | --- |
| Name: | Filter goods according to specific conditions |
| Responsibilities: | Open the website click on the search box and enter search keywords |
| Use Case: | UC -7 |
| Exception: | None |
| Preconditions: | Users need to open the site and successfully log in to their account, open the search box and set the search criteria, and turn on phone storage permissions. |
| Postcondition: | Access the database and search by search restrictions |

|  |  |
| --- | --- |
| Name: | Shopping cart checkout |
| Responsibilities: | Check out the goods in the shopping cart |
| Use Case: | UC -8 |
| Exception: | None |
| Preconditions: | The relevant purchase information of the product has been completed before being added to the shopping cart. |
| Postcondition: | Run related background settlement system |

|  |  |
| --- | --- |
| Name: | Full purchase of goods |
| Responsibilities: | Purchase second-hand goods for permanent use |
| Use Case: | UC -9 |
| Exception: | None |
| Preconditions: | Let the lessee agree to pay the full purchase of the goods |
| Postcondition: | Have enough balance in the wallet |

|  |  |
| --- | --- |
| Name: | payment procedure |
| Responsibilities: | Confirm the price of the item purchased by the user and pay |
| Use Case: | UC -10 |
| Exception: | None |
| Preconditions: | The user has added the item he wants to purchase to the shopping cart and is ready to check out |
| Postcondition: | The system detects the price, quantity and other information of the product selected by the user and calculates the price. The database checks the inventory information and prepares for modification. |

|  |  |
| --- | --- |
| Name: | comments |
| Responsibilities: | Can satisfy already bought the customer to express the opinion, may provide the reference value for other customers |
| Use Case: | UC -11 |
| Exception: | None |
| Preconditions: | The user must pay successfully first |
| Postcondition: | Display successfully in the comments page |

|  |  |
| --- | --- |
| Name: | View order status |
| Responsibilities: | Users can know the status of goods in real time |
| Use Case: | UC -12 |
| Exception: | None |
| Preconditions: | The user must log in successfully |
| Postcondition: | The user browses the order information successfully |

**6.3 Mathematical Model**

6.3 Mathematical Model

**Table specification algorithm**

Algorithm that allows the customer's various processes

code:

@Controller

@RequestMapping(value = "/goods")

public class GoodsController {

@Autowired

private GoodsService goodsService;

@Autowired

private ImageService imageService;

@Autowired

private CatelogService catelogService;

@Autowired

private UserService userService;

@Resource

private PurseService purseService;

/\*\*

\* The home page displays the goods, and each item is queried 6 pieces. According to the latest sort of keys, the key names are catelogGoods1, catelogGoods2....

\*

\* @return

\* @throws Exception

\*/

@RequestMapping(value = "/homeGoods")

public ModelAndView homeGoods() throws Exception {

ModelAndView modelAndView = new ModelAndView();

// Number of product types

int catelogSize = 7;

// Display the number of items for each category

int goodsSize = 6;

List<Goods> goodsList = null;

List<GoodsExtend> goodsAndImage = null;

/\* Get the latest release list \*/

goodsList = goodsService.getGoodsOrderByDate(goodsSize);

goodsAndImage = new ArrayList<GoodsExtend>();

for (int j = 0; j < goodsList.size(); j++) {

// Encapsulate user information and image information into the GoodsExtend class and pass it to the foreground

GoodsExtend goodsExtend = new GoodsExtend();

Goods goods = goodsList.get(j);

List<Image> images = imageService.getImagesByGoodsPrimaryKey(goods.getId());

goodsExtend.setGoods(goods);

goodsExtend.setImages(images);

goodsAndImage.add(j, goodsExtend);

}

String key0 = "catelog" + "Goods";

modelAndView.addObject(key0, goodsAndImage);

/\* Get other list item information \*/

for (int i = 1; i <= catelogSize; i++) {

goodsList = goodsService.getGoodsByCatelogOrderByDate(i, goodsSize);

goodsAndImage = new ArrayList<GoodsExtend>();

for (int j = 0; j < goodsList.size(); j++) {

// Encapsulate user information and image information into the GoodsExtend class and pass it to the foreground

GoodsExtend goodsExtend = new GoodsExtend();

Goods goods = goodsList.get(j);

List<Image> images = imageService.getImagesByGoodsPrimaryKey(goods.getId());

goodsExtend.setGoods(goods);

goodsExtend.setImages(images);

goodsAndImage.add(j, goodsExtend);

}

String key = "catelog" + "Goods" + i;

modelAndView.addObject(key, goodsAndImage);

}

modelAndView.setViewName("goods/homeGoods");

return modelAndView;

}

/\*\*

\* Search for goods

\*

\* @param str //Ajax pass value

\* @return

\* @throws Exception

\*/

@RequestMapping(value = "/search")

public ModelAndView searchGoods(@RequestParam(value = "str", required = false) String str) throws Exception {

List<Goods> goodsList = goodsService.searchGoods(str, str);

List<GoodsExtend> goodsExtendList = new ArrayList<GoodsExtend>();

for (int i = 0; i < goodsList.size(); i++) {

GoodsExtend goodsExtend = new GoodsExtend();

Goods goods = goodsList.get(i);

List<Image> imageList = imageService.getImagesByGoodsPrimaryKey(goods.getId());

goodsExtend.setGoods(goods);

goodsExtend.setImages(imageList);

goodsExtendList.add(i, goodsExtend);

}

ModelAndView modelAndView = new ModelAndView();

modelAndView.addObject("goodsExtendList", goodsExtendList);

modelAndView.addObject("search", str);

modelAndView.setViewName("/goods/searchGoods");

return modelAndView;

}

/\*\*

\* Query this product

\*

\* @param id

\* Require this parameter to be empty

\* @return

\* @throws Exception

\*/

@RequestMapping(value = "/catelog")

public ModelAndView homeGoods(HttpServletRequest request, @RequestParam(value = "str", required = false) String str)

throws Exception {

ModelAndView modelAndView = new ModelAndView();

// Display the number of items for each category

int goodsSize = 12;

List<Goods> goodsList = null;

List<GoodsExtend> goodsAndImage = null;

/\* Get the latest release list \*/

goodsList = goodsService.getGoodsByStr(goodsSize, str, str);

goodsAndImage = new ArrayList<GoodsExtend>();

for (int j = 0; j < goodsList.size(); j++) {

// Encapsulate user information and image information into the GoodsExtend class and pass it to the foreground

GoodsExtend goodsExtend = new GoodsExtend();

Goods goods = goodsList.get(j);

List<Image> images = imageService.getImagesByGoodsPrimaryKey(goods.getId());

goodsExtend.setGoods(goods);

goodsExtend.setImages(images);

goodsAndImage.add(j, goodsExtend);

}

modelAndView.addObject("goodsExtendList", goodsAndImage);

modelAndView.addObject("search", str);

modelAndView.setViewName("/goods/catelogGoods");

return modelAndView;

}

/\*\*

\* Query this product

\*

\* @param id

\* Require this parameter to be empty

\* @return

\* @throws Exception

\*/

@RequestMapping(value = "/catelog/{id}")

public ModelAndView catelogGoods(HttpServletRequest request, @PathVariable("id") Integer id,

@RequestParam(value = "str", required = false) String str) throws Exception {

List<Goods> goodsList = goodsService.getGoodsByCatelog(id, str, str);

Catelog catelog = catelogService.selectByPrimaryKey(id);

List<GoodsExtend> goodsExtendList = new ArrayList<GoodsExtend>();

for (int i = 0; i < goodsList.size(); i++) {

GoodsExtend goodsExtend = new GoodsExtend();

Goods goods = goodsList.get(i);

List<Image> imageList = imageService.getImagesByGoodsPrimaryKey(goods.getId());

goodsExtend.setGoods(goods);

goodsExtend.setImages(imageList);

goodsExtendList.add(i, goodsExtend);

}

ModelAndView modelAndView = new ModelAndView();

modelAndView.addObject("goodsExtendList", goodsExtendList);

modelAndView.addObject("catelog", catelog);

modelAndView.addObject("search", str);

modelAndView.setViewName("/goods/catelogGoods");

return modelAndView;

}

/\*\*

\* Query the product details based on the product id

\*

\* @param id

\* @return

\* @throws Exception

\*/

@RequestMapping(value = "/goodsId/{id}")

public ModelAndView getGoodsById(HttpServletRequest request, @PathVariable("id") Integer id,

@RequestParam(value = "str", required = false) String str) throws Exception {

Goods goods = goodsService.getGoodsByPrimaryKey(id);

User seller = userService.selectByPrimaryKey(goods.getUserId());

Catelog catelog = catelogService.selectByPrimaryKey(goods.getCatelogId());

GoodsExtend goodsExtend = new GoodsExtend();

List<Image> imageList = imageService.getImagesByGoodsPrimaryKey(id);

CommentExtend CommentExtend=goodsService.selectCommentsByGoodsId(id);

goodsExtend.setGoods(goods);

goodsExtend.setImages(imageList);

ModelAndView modelAndView = new ModelAndView();

modelAndView.addObject("CommentExtend",CommentExtend);

modelAndView.addObject("goodsExtend", goodsExtend);

modelAndView.addObject("seller", seller);

modelAndView.addObject("search", str);

modelAndView.addObject("catelog", catelog);

modelAndView.setViewName("/goods/detailGoods");

return modelAndView;

}

/\*\*

\* Post a comment

\* @return

\*/

@RequestMapping(value = "/addComments",method=RequestMethod.POST)

public void deleteFocus(HttpServletRequest request,Comments comments) {

User cur\_user = (User)request.getSession().getAttribute("cur\_user");

comments.setUser(cur\_user);

SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd hh:mm:ss");

Date createAt =new Date();

comments.setCreateAt(sdf.format(createAt));

goodsService.addComments(comments);

}

/\*\*

\* Edit product information

\*

\* @return

\* @throws Exception

\*/

@RequestMapping(value = "/editGoods/{id}")

public ModelAndView editGoods(HttpServletRequest request,@PathVariable("id") Integer id) throws Exception {

User cur\_user = (User) request.getSession().getAttribute("cur\_user");

Goods goods = goodsService.getGoodsByPrimaryKey(id);

List<Image> imageList = imageService.getImagesByGoodsPrimaryKey(id);

GoodsExtend goodsExtend = new GoodsExtend();

goodsExtend.setGoods(goods);

goodsExtend.setImages(imageList);

ModelAndView modelAndView = new ModelAndView();

Integer userId = cur\_user.getId();

Purse myPurse = purseService.getPurseByUserId(userId);

modelAndView.addObject("myPurse", myPurse);

// Add product information to model

modelAndView.addObject("goodsExtend", goodsExtend);

modelAndView.setViewName("/goods/editGoods");

return modelAndView;

}

/\*\*

\* Submit product change information

\*

\* @return

\* @throws Exception

\*/

@RequestMapping(value = "/editGoodsSubmit")

public String editGoodsSubmit(HttpServletRequest request, Goods goods) throws Exception {

User cur\_user = (User) request.getSession().getAttribute("cur\_user");

goods.setUserId(cur\_user.getId());

String polish\_time = DateUtil.getNowDay();

goods.setPolishTime(polish\_time);

goods.setStatus(1);

goodsService.updateGoodsByPrimaryKeyWithBLOBs(goods.getId(), goods);

return "redirect:/user/allGoods";

}

/\*\*

\* Merchandise

\*

\* @return

\* @throws Exception

\*/

@RequestMapping(value = "/offGoods")

public ModelAndView offGoods() throws Exception {

return null;

}

/\*\*

\*User deleted item

\*

\* @return

\* @throws Exception

\*/

@RequestMapping(value = "/deleteGoods/{id}")

public String deleteGoods(HttpServletRequest request, @PathVariable("id") Integer id) throws Exception {

Goods goods = goodsService.getGoodsByPrimaryKey(id);

// After deleting the item, the number-1 of the catlog, the goods\_num-1, image of the user table are deleted, and the value of the session is updated.

User cur\_user = (User) request.getSession().getAttribute("cur\_user");

goods.setUserId(cur\_user.getId());

int number = cur\_user.getGoodsNum();

Integer calelog\_id = goods.getCatelogId();

Catelog catelog = catelogService.selectByPrimaryKey(calelog\_id);

catelogService.updateCatelogNum(calelog\_id, catelog.getNumber() - 1);

userService.updateGoodsNum(cur\_user.getId(), number - 1);

cur\_user.setGoodsNum(number - 1);

request.getSession().setAttribute("cur\_user", cur\_user);// Modify session value

//imageService.deleteImagesByGoodsPrimaryKey(id);

goodsService.deleteGoodsByPrimaryKey(id);

return "redirect:/user/allGoods";

}

/\*\*

\* Publish goods

\*

\* @return

\* @throws Exception

\*/

@RequestMapping(value = "/publishGoods")

public ModelAndView publishGoods(HttpServletRequest request) {

// Can verify that the user is logged in

User cur\_user = (User) request.getSession().getAttribute("cur\_user");

// if (cur\_user == null) {

// return "/goods/homeGoods";

// } else {

Integer userId = cur\_user.getId();

Purse myPurse = purseService.getPurseByUserId(userId);

ModelAndView mv = new ModelAndView();

mv.addObject("myPurse", myPurse);

mv.setViewName("/goods/pubGoods");

return mv;

}

/\*\*

\*Submit published product information

\*

\* @return

\* @throws Exception

\*/

@RequestMapping(value = "/publishGoodsSubmit")

public String publishGoodsSubmit(HttpServletRequest request, Image ima, Goods goods, MultipartFile image)

throws Exception {

// Query the current user cur\_user object, easy to use id

User cur\_user = (User) request.getSession().getAttribute("cur\_user");

goods.setUserId(cur\_user.getId());

goodsService.addGood(goods, 10);// Insert items in the goods table

// Returns the id of the inserted item

int goodsId = goods.getId();

ima.setGoodsId(goodsId);

imageService.insert(ima);//Insert product image in image table

// After the product is released, the number of the catlog is +1, the goods\_num+1 of the user table, and the value of the session is updated.

int number = cur\_user.getGoodsNum();

Integer calelog\_id = goods.getCatelogId();

Catelog catelog = catelogService.selectByPrimaryKey(calelog\_id);

catelogService.updateCatelogNum(calelog\_id, catelog.getNumber() + 1);

userService.updateGoodsNum(cur\_user.getId(), number + 1);

cur\_user.setGoodsNum(number + 1);

request.getSession().setAttribute("cur\_user", cur\_user);// Modify session value

return "redirect:/user/allGoods";

}

/\*\*

\* Upload item

\*

\* @param session

\* @param myfile

\* @return

\* @throws IllegalStateException

\* @throws IOException

\*/

@ResponseBody

@RequestMapping(value = "/uploadFile")

public Map<String, Object> uploadFile(HttpSession session, MultipartFile myfile)

throws IllegalStateException, IOException {

// Original name

String oldFileName = myfile.getOriginalFilename(); // Get the original name of the uploaded file

// Store the physical path of the picture

String file\_path = session.getServletContext().getRealPath("upload");

// System.out.println("file\_path:"+file\_path);

// upload image

if (myfile != null && oldFileName != null && oldFileName.length() > 0) {

// New image name

String newFileName = UUID.randomUUID() + oldFileName.substring(oldFileName.lastIndexOf("."));

// New picture

File newFile = new File(file\_path + "/" + newFileName);

// Write data in memory to disk

myfile.transferTo(newFile);

// Return the new image name to the front end

Map<String, Object> map = new HashMap<String, Object>();

map.put("success",);

map.put("imgUrl", newFileName);

return map;

} else {

Map<String, Object> map = new HashMap<String, Object>();

map.put("error", "Image is illegal");

return map;

}

}

/\*\*

\* Query the product details based on the product id

\*

\* @param id

\* @return

\* @throws Exception

\*/

@RequestMapping(value = "/buyId/{id}")

public ModelAndView getGoodsdetailById(HttpServletRequest request, @PathVariable("id") Integer id)

throws Exception {

Goods goods = goodsService.getGoodsByPrimaryKey(id);

GoodsExtend goodsExtend = new GoodsExtend();

List<Image> imageList = imageService.getImagesByGoodsPrimaryKey(id);

goodsExtend.setGoods(goods);

goodsExtend.setImages(imageList);

User cur\_user = (User)request.getSession().getAttribute("cur\_user");

Integer userId = cur\_user.getId();

Purse myPurse=purseService.getPurseByUserId(userId);

ModelAndView modelAndView = new ModelAndView();

modelAndView.addObject("goodsExtend", goodsExtend);

modelAndView.addObject("myPurse",myPurse);

modelAndView.setViewName("/user/pay");

return modelAndView;

}

}