|  |
| --- |
|  |
| **DropCop** |
|  |
| **Software Design Document** |
| **Version 2.0**  Document Number: SDD-002 |
|  |

|  |
| --- |
|  |

Project Team Number: B12

Pavandip Singh

Mohammed Alam

Justin Opraseuth

**REVIEW AND APPROVALS**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Role** | **Date** | **Signature** |
| Fred Strauss | Approval |  | On File |
| Pavandip Singh | Author and Reviewer | April 28th, 2015 | On File |
| Mohammed Alam | Author and Reviewer | April 28th, 2015 | On File |
| Justin Opraseuth | Author and Reviewer | April 28th, 2015 | On File |

**REVISION LEVEL**

|  |  |  |
| --- | --- | --- |
| **Date** | **Revision Number** | **Purpose** |
| March 22, 2015 | Version 1.0 | Initial Release |
| April 28, 2015 | Version 2.0 | Eliminate inconsistencies |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

**Table of Contents**

**1. INTRODUCTION ....................................................................................................... 6**

1.1 PURPOSE......................................................................................................... 6 1.2 SCOPE.......................................................................................................................... 6

1.3 IDENTIFICATION .......................................................................................................6

1.4 DOCUMENT SUMMARY.......................................................................................... 6

1.5 SYSTEM OVERVIEW................................................................................................ 7

1.6 DOCUMENT OVERVIEW.......................................................................................... 7

**2. REFERENCE DOCUMENTS .................................................................................... 7**

**3. SYSTEM WIDE DESIGN DECISIONS..................................................................... 7**

3.1 SOFTWARE COMPONENT ARCHITECTURAL DESIGN......................... 8

3.2 SOFTWARE ARCHITECTURE GENERAL DESCRIPTION ...................... 8 3.3 SOFTWARE ITEM COMPONENTS…………………….............................. 8

3.4 COMPONENT INTERFACE IDENTIFICATION......................................... 9

3.5 SOFTWARE COMPONENT CONCEPT OF EXECUTION…......................9

**4. SOFTWARE ITEM DETAILED DESIGN ............................................................. 11**

4.1 STRUCTURE................................................................................................. 11

4.1.1 Software Unit Detailed Design........................................................ 11

4.2 STATIC RELATIONSHIP OF SOFTWARE UNIT …................................. 13

4.2.1 Run-time Object Instances............................................................... 13

4.3 BEHAVIOR.................................................................................................... 13 4.3.1 Interaction Diagrams…………….................................................... 13

4.3.2 Collaboration Diagrams................................................................... 14

4.3.3 Activity Diagrams …………………............................................... 15

4.4 CONCEPT OF EXECUTION........................................................................ 15

4.5 INTERFACE DESIGN .................................................................................. 16

4.5.1 Interface Identification and Diagrams.............................................. 16

4.5.2 Unique identifier of Interface .......................................................... 17

**5. IMPLEMENTATION ARCHITECTURE (NOT REQUIRED)…………............ 18**

5.1 ALL ACTIVE AND PASSIVE CLASSES ASSIGNED TO COMPONENTS……………………………………………………………........ 17

5.2 DIAGRAMS OF PHYSICAL PACKAGING OF LOGICAL COMPONENTS ............................................................................................................................... 17

**6. DEPLOYMENT ARCHITECTURE…………….................................................... 18**

6.1 PHYSICAL DEPLOYMENT ARCHITECTURE DIAGRAM…................. 18

**7. DICTIONARIES………………................................................................................. 19**

**8. SOFTWARE ITEM COMPUTER RESOURCE UTILIZATION…………….... 21**

**9. REQUIREMENTS TRACEABILITY…………...................................................... 22**

9.1 SOFTWARE COMPONENT-LEVEL REQUIREMENTS TRACEABILITY …………………………………………………................................................... .22

**10. SYSTEM DESIGN TESTING.................................................................................. 22**

**11. RATIONALE……………………………………………….................................... 23**

**12. NOTES…………………........................................................................................... 23**

**13. APPENDICES……………………........................................................................... 23**

13.1 DICTIONARIES…………………………………………………………... 23

13.2 UML DIAGRAMS ....................................................................................... 23

13.3 SCHEDULE TRACKING………………………….................................... 23

13.4 DEFECT TRACKING……………………….............................................. 25

13.5 GANTT CHART………………………………………………………….27-28

**1. INTRODUCTION**

This section introduces Software Design Description (SDD) of the DropCop system to its developers, SQA team, and project managers.

**1.1 Purpose**

The general purpose of this document is to define the contents of the DropCop’s application design. For its very own purpose, this document will be provided to the SQA (Software Quality Assurance group), along with the development team and client. SQA will use this document to examine and test the product through its life-cycle based on its given functionalities/requirements. The Client will then use this and other documents to make sure that the specified requirements align with their demands. The development team using the Software Requirements Specification (SRS) and SDD will create the actual product.

**1.2 Scope**

The scope of this document, the System Design Description, is to describe the contents of the DropCop application infrastructure. This includes some general definitions of application building along with the components that go along with building the application. This Document will outline a clear path to the application’s architecture and Detailed Design. Diagrams will be used in this document to give its readers a vivid image to show the interactions and relationships between each component of the system. While that is laid out, scripts and designs of other sub-systems will be generally shown to discuss the foundation of each component and its relationship to the application as a whole. The scope of the entire document will ensure that the information being given to its reader is easy to understand.

**1.3 Identification**

DropCop, Version 2.0, April 28th 2015

**1.4 Document Summary**

The main purpose of this document will be to help the reader grasp how the base level system works. With that, it will help show the various standpoints and details regarding the design of the system. By doing that we are also painting the image of the applications workability. In this document, the aim is generally to highlight the core components of the system architecture and its infrastructure as a whole. This document then can be used for creating such a product and to test its functionality.

**1.5 System Overview**

The overview of the system includes the application being managed well on its own as it will be developed as a sub-part of the project

**1.6 Document Overview**

Format and content of this Document:

Title Page (formatted the same as standard cover)

Review/Approval Signatures

Table of Revisions (revision number, date, purpose)

Approval page

Preface (information the reader should be familiar with)

Table of Contents( which includes the following)

(System Wide Design Decisions

Software Item Detailed Design

Deployment Architecture

Dictionaries Requirements

Traceability System

Design testing

List of Figures)

( Object Oriented: UML – Use Case diagrams, Class and Object diagrams, Interaction diagrams, Key Event diagrams, Sequence Diagrams, Dictionaries, etc.) will be included in the Detailed Design Section

**2. REFERENCE DOCUMENTS**

DropCop, Team B12, SRS, Version 1.0

DropCop, Team B12, SPMP, Version 1.0

DropCop, Team B12, SAS, Version 1.0

DropCop, Team B12, RAS, Version 1.0

DropCop, Team B12, SPMP, Version 2.0

**3. SYSTEM WIDE DESIGN DECISION**

**3.1 Software Component Architectural Design**

Login Page

User Account

1) Product/Item info

2) Buy Item/checkout page

Database connector

HTTP Request Handler

Database

User Layer

Application Layer

Database Management Layer

Registration Page

HomePage

1)Search Items

2) Post Item for sale

**3.2 Software Architectural General Description**

The architecture is composed of three components/layers. The three layers are the User Layer, Application Layer, and the Data Management Layer. With the User Layer we make it possible to have a secure login in page which then using the application layer to connect each user to his own unique account. Simultaneously, the database layer is querying the right/ proper data to the account. The database unit includes all information regarding the user account, which items are in the cart, and page information (item information).

**3.3 Software Item Components**

User Interface Components

|  |  |
| --- | --- |
| **Component** | **Description** |
| Login | Allows for the user to log in |
| Registration | Allows user to create account |
| Search Items | Allows user to search for a desired item |
| Manage Account | Allows user to update/modify account |
| Account info | Allows user to view their account information |
| Post for Selling Item | Allow user to post an item they are selling |

Application Layer Components

|  |  |
| --- | --- |
| **Component** | **Description** |
| Availability of Items | Verify validation of item |
| User Authentication | Verify validity of login info |
| Item Checkout | Allow user to buy product |

Database Components

|  |  |
| --- | --- |
| **Component** | **Description** |
| Login Information Table | Table for all Login info |
| Item Information Table | Table for all available items to buy |

**3.4 Component Interface Identification**

There are multiple unique component connectors/interfaces. The communication between the Application layer and the User layer will be done through the use of HTTP requests. This helps us get a communication ready between the client and the server. We will also use a C++ database connector such as libpqxx. This will allow the server to communicate with the PostgreSQL Database. The interface connectors are used in a specific way where the User interface connects with the application interface and application interface also connects to the database interface.

**3.5 Software Component Concept of Execution**

User Interface enters execution state when:

* When a request is made to open the GUI
* UI is receiving data from application server
* When a request is sent by the user/client by clicking a button to go to a different page such as (see cart, manage account)

Application Server enters execution State when:

* A User requests to see items for sale
* A User requests to see his cart
* A new account is created or deleted
* User Authentication
* Item Validation
* Cart Checkout

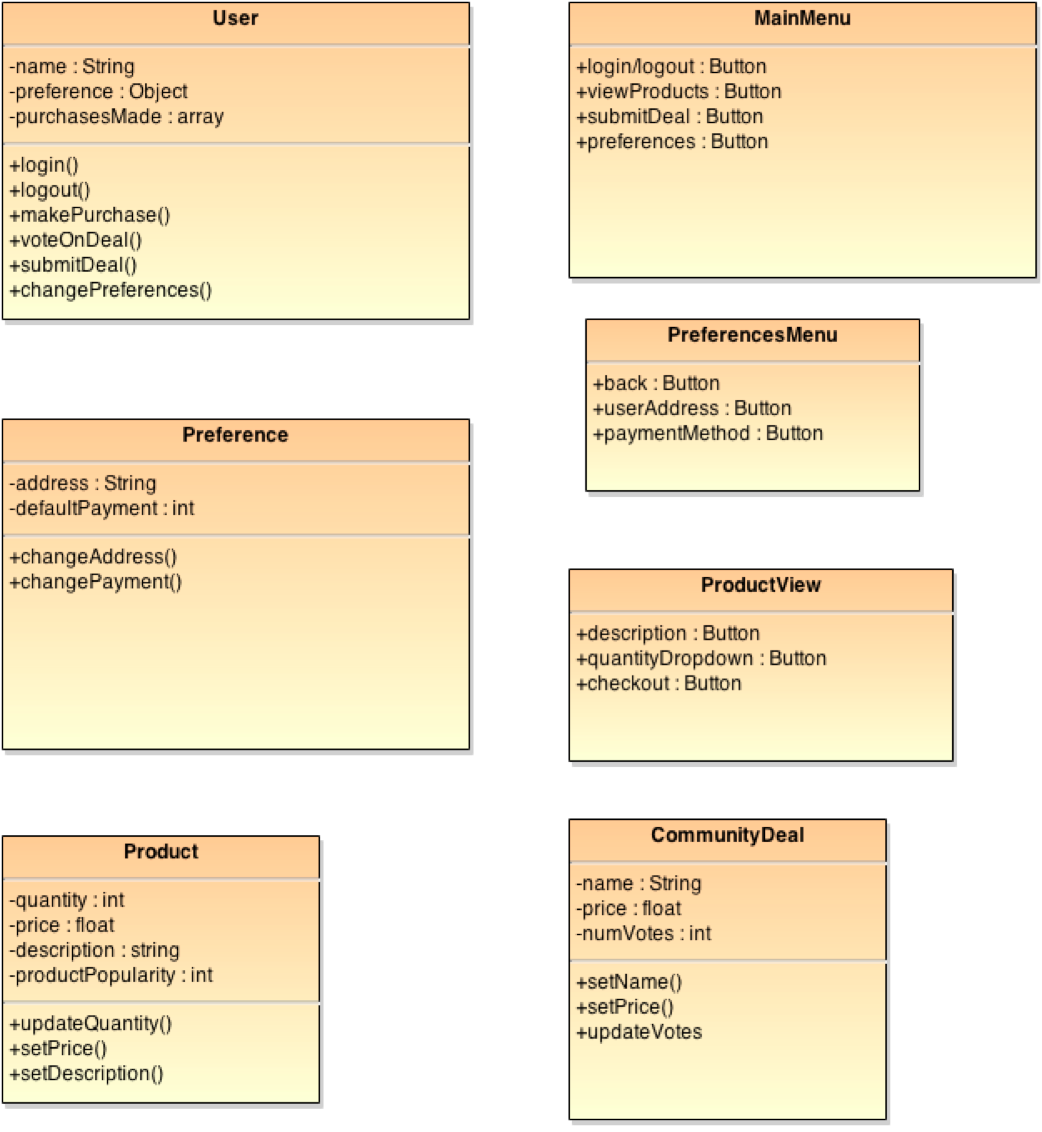
Database enters execution state when: (mostly same stuff as Application Server because all information of the Application Server comes from the database, so whenever Application server is in execution state so is the database)

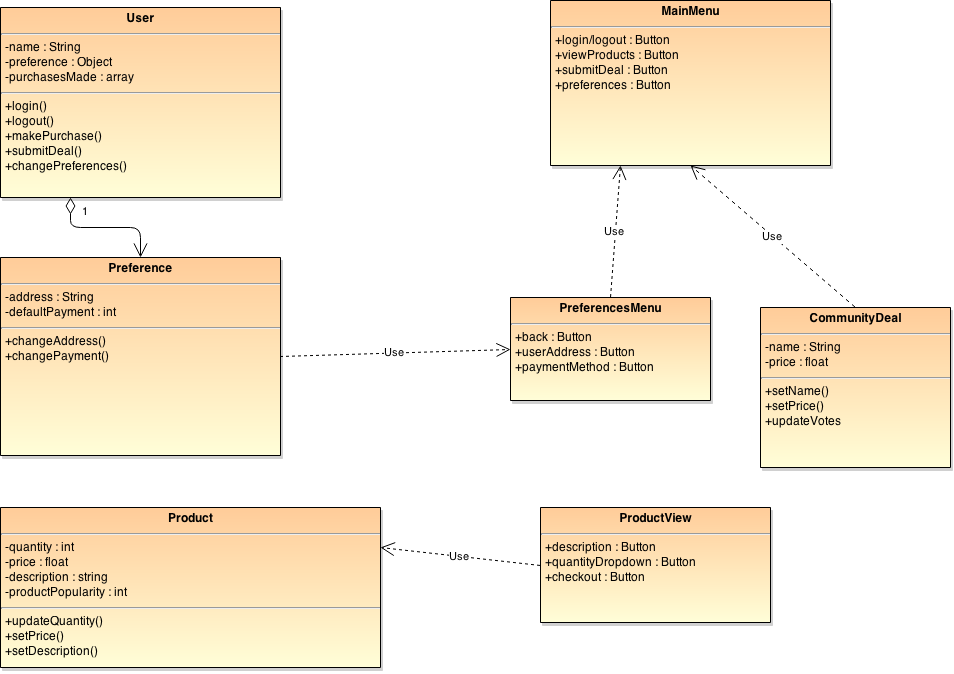
* When an item for sale is posted
* The app server requests to create or delete a login account

**4. SOFTWARE ITEM DETAILED DESIGN**

**4.1 Structure**

**4.1.1 Software Unit Detailed Design**





**4.2 Static Relationship of Software Unit**

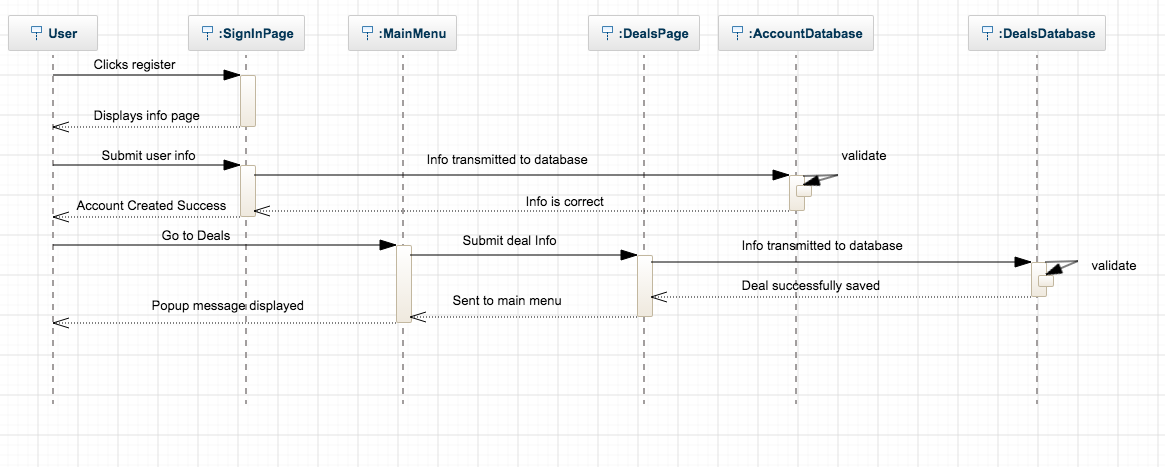
**4.2.1 Run-time Object Instances**

Refer to Section 4.3

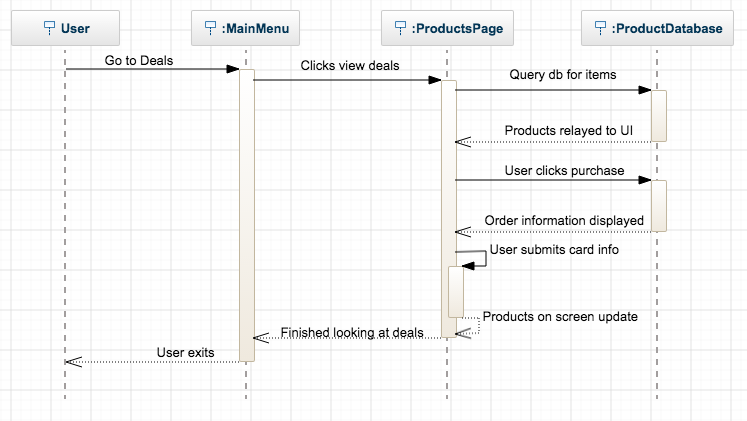
**4.3 Behavior**

**4.3.1 Interaction Diagrams**

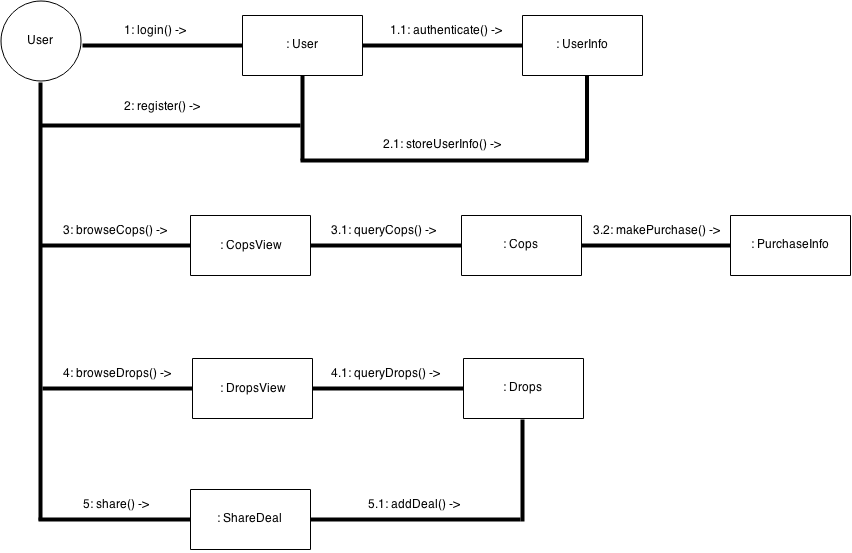
User Registers and Submits deal (Drop)



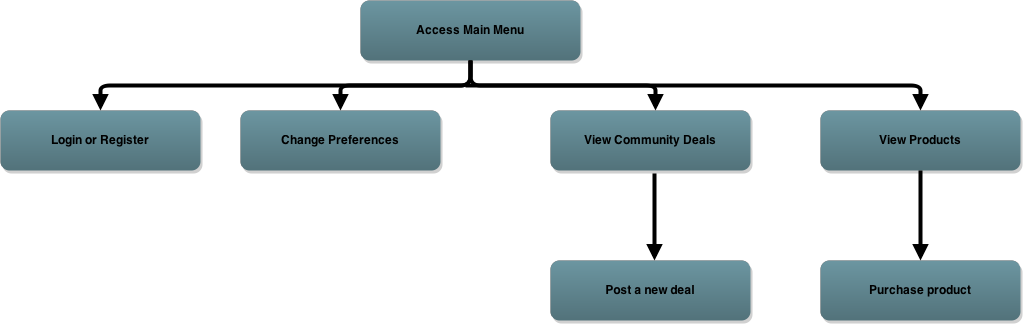
User Purchases a Product (Cop)



**4.3.2 Collaboration Diagrams**



**4.3.3 Activity Diagrams**



**4.4 Concept of Execution**

A client-side application will serve as the medium for all interactions. The user will be able to register for a new account requiring a name, email address, and password. The application server then updates the corresponding tables within the database. The application server authenticates a user’s login as well.

The application allows a user to both browse and purchase Cops if they wish, and also submit drops to the database. If the user chooses to browse Cops, by default the application will show the most recent products available. By scrolling through the Cops, the user may click on an item and make a purchase. The application will send all card information to the payment processor. All purchase details will be stored in the database as well. A user is also able to browse the user-submitted Drops using the application.

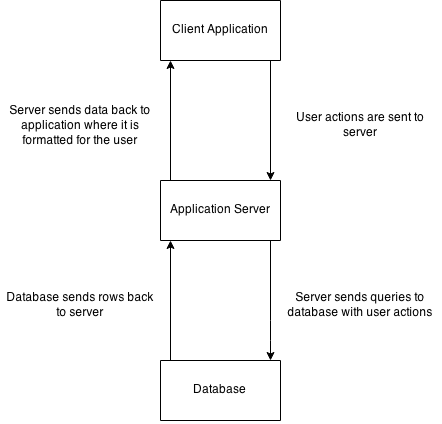
A user can share drops by submitting the information about the deal and a link to purchase the item. Drops are stored in a separate database from Cops. The application layer will transmit all data to the database.

**4.5 Interface Design**

The DropCop application sends user actions as requests to the application server. SQL queries are crafted as necessary and are sent to the database. Only simple queries are expected to be made: to add rows for product information, pull up information sorted by date added, and search rows pertaining to a user’s orders. The application server then transfer this information to the application where it is formatted and displayed to the user.

In the case of a purchase, transaction details are sent to the payment processor and the results of the charge are transmitted to the server. At this point, order details are written within the database.

**4.5.1 Interface Identification and Diagrams**



**4.5.2 Unique identifier of Interface**

An interface between the client-side application and the application server manages user actions and data that will be displayed on-screen. An interface between the application server and the database queries for products and also orders that may have been placed. The Application Server is the core element of DropCop as it handles communication both to the client and the database.

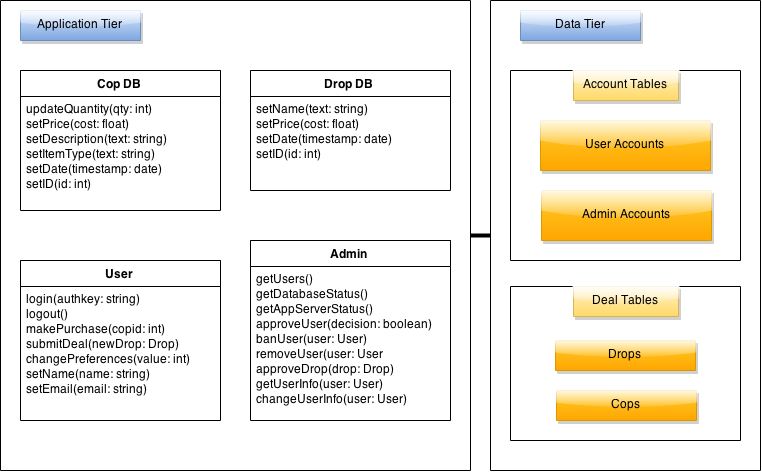
**5. IMPLEMENTATION ARCHITECTURE (NOT REQUIRED)**

**5.1 All Active and Passive Classes assigned to Component**

**5.2 Diagrams of Physical Packaging of Logical Component**

**6. DEPLOYMENT ARCHITECTURE**

**6.1 Physical Deployment Architectural Design**



**7. DICTIONARIES**

|  |  |  |
| --- | --- | --- |
| **Class** | User |  |
| **Parent** | None |  |
|  | **Name** | **Description** |
| **Methods** | login | Log into the DropCop server |
|  | logout | Logs user out from application |
|  | makePurchase | Handles checkout process for user |
|  | submitDeal | Adds a deal to become available for community voting |
|  | **Name** | **Description** |
| **Attributes** | name | Full name |
|  | preferences | Object holds user data |
|  | purchasesMade | Shows past orders |

|  |  |  |
| --- | --- | --- |
| **Class** | Product |  |
| **Parent** | None |  |
|  | **Name** | **Description** |
| **Methods** | updateQuantity | Change quantity after purchase or if stock needs to be updated |
|  | setPrice | Change price of the item |
|  | setDescription | Change item description shown on product page |
|  | **Name** | **Description** |
| **Attributes** | quantity | How many units are available |
|  | price | Current price of item |
|  | description | Description of product on site |
|  | productPopularity | Current sales rank on the site |

|  |  |  |
| --- | --- | --- |
| **Class** | CommunityDeal |  |
| **Parent** | None |  |
|  | **Name** | **Description** |
| **Methods** | setName | Updates name of deal |
|  | setPrice | Updates the user submitted price |
|  | **Name** | **Description** |
| **Attributes** | name | Name of deal |
|  | price | Price of deal found |

|  |  |  |
| --- | --- | --- |
| **Class** | MainMenu |  |
| **Parent** | None |  |
|  | **Name** | **Description** |
| **Methods** |  |  |
|  | **Name** | **Description** |
| **Attributes** | login/logout | Button allows user to login/logout |
|  | viewProducts | Button shows list of available products |
|  | submitDeal | Button for pop-up interface so user can enter deal |
|  | preferences | Button to access user preferences |

|  |  |  |
| --- | --- | --- |
| **Class** | ProductView |  |
| **Parent** | None |  |
|  | **Name** | **Description** |
| **Methods** |  |  |
|  | **Name** | **Description** |
| **Attributes** | description | Shows product name |
|  | quantityDropdown | Changes amount of product one wants to buy |
|  | checkout | Button starts checkout process pop up window |

**8. SOFTWARE ITEM COMPUTER RESOURCE UTILIZATION**

The system does not require a large overhead. A simple host server will be needed that can serve multiple clients reliable and store data on a drive. Storage requirements are a minimum because we only store users login information and the product data in databases so 50GB is more than enough space to hold all data. The processor should be capable of handle multiple threads and there should be enough ram to accommodate many users trying to login and buy products at the same time. A system with a quad core processor and 8GB of RAM should be more than sufficient to handle all the tasks and leave room for increased traffic. The internet connection is probably key in the success of the server. Without a sufficient network connection, users will not be able to connect to the server or the server could not host multiple clients and respond as expected. As such, a connection with at least 25 Mbps up and down would be enough to ensure no one has to wait for data to or from the server. Additionally, the user must have apply pay set up on their phone or else they will not be able to buy products from the system, but can still use links to third party dealers.

**9. REQUIREMENTS TRACEABILITY**

**9.1 Software Component-Level Requirements Traceability**

1. This section maintains information about the origins of requirements and their traceability both forward and backward. The following is a description of the key requirements and dependencies of DropCop. This section will be used as a reference in case any requirements need to be modified.
2. The software is dependent on the mobile operating systems in use. As of now, the project will be maintained for both iOS and Android. These two variants of the application must be kept up to date with key OS upgrades. DropCop should support any new features that an OS upgrade will bring to the table.
3. The internet connection of mobile devices has a measurable effect on a user’s experience with the application. The application works best with a 4G or WiFi connection. The application’s data transmissions should be kept as small as possible to ensure a good experience with a wide range of devices.
4. As stated previously, Google Wallet and Apple Pay will be the two major payment providers within the application. The cost of these two networks will be monitored and it may be necessary to switch to a lower cost payment processor if market conditions change.
5. The DropCop servers will receive memory dumps from devices whenever the application crashes. The events leading up to application crashes will be recorded in a document. Users may also submit feedback through the application.

**10. SYSTEM DESIGN TESTING**

The DropCop system has many facets. In order to properly test the functionality of the mobile apps, they must be tested under several circumstances. First, the general case must be tested in which a user on either iOS or Android will browse through and purchase a deal. This will be the most trivial case seeing as to how it is the purpose of the application. There should be no errors within the basic UI of the application. The layout should look the same no matter what the device’s screen resolution is. Also, the robustness/stability of the application server must be tested. Multiple devices will be emulated and the server performance under heavy load will be measured. This will simulate when an extraordinary deal is posted on the site. All transactions should be handled without payment issues. Another scenario will be to make sure that simultaneous orders are handled properly. If an item is about to run out of stock, only one customer should be able to purchase the last item.

**11. RATIONALE**

The application was developed in order to make it possible for a large audience of people to get great deals on items that they want to purchase. By using the economic principle of volume discounts, a large amount of money will be saved and will be kept in consumers’ pockets. It would be great if DropCop were able to convince more consumers to use online shopping. Discounting items further than e-commerce sites such as Amazon and eBay may be the impetus that some consumers may need to purchase items online. The ease of use of the application and also the security provided with the application will draw consumers to purchase items using the complex network of vendors.

The community deals of the application will be the social media aspect of the application. People will be able to communicate with each other by see what trends are in the current market. With this data, only the best deals will be provided. The vendors that we work with will also be able to see how their products compare to other options.

The team creators would like it for DropCop to become a household name as large as Amazon or Wal-Mart. DropCop’s network of vendors will be able to penetrate the current online shopping market!

**12. NOTES**

**13. APPENDICES**

**13.1 Dictionaries**

Included within the body of the document. Refer to section 7.

**13.2 UML diagrams, if not included in the body of the document**

Included within the body of the document. Refer to section 4.1.1.

**13.3 Schedule Tracking**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Artifact or**  **Deliverable** | **Who (individual**  **or Team)** | **Estimated** | **Actual** | **Difference** |
| RAS | Pavandip Singh | 1 hours | 1 ½ hours | ½ hours |
| RAS | Justin Opraseuth | 2 hours | 1 ½ hours | -½ hours |
| RAS | Mohammed Alam | 1 ½ hours | 1 ½ hours | 0 hours |
|  | Total | 4 ½ | 4 ½ | 0 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Artifact or**  **Deliverable** | **Who (individual**  **or Team)** | **Estimated** | **Actual** | **Difference** |
| SDD initial,final | Pavandip Singh | 3 hours | 3 ½ hours | -½ hours |
| SDD initial, final | Justin Opraseuth | 3 ½ hours | 3 ½ hours | 0 hours |
| SDD initial,final | Mohammed Alam | 3 hours | 3 hours | 0 hours |
|  | Total | 9 ½ | 9 ½ | -½ |

**Cumulative**

|  |  |  |  |
| --- | --- | --- | --- |
| **Who (individual**  **or Team)** | **Estimated** | **Actual** | **Difference** |
| Pavandip Singh | 4 hours | 4 hours | 0 hours |
| Justin Opraseuth | 5 ½ hours | 5 hours | -½ hours |
| Mohammed Alam | 4 ½ hours | 4 ½ hours | 0 hours |
| Total | 14 | 13 ½ | -½ |

**13.4 Defect Tracking**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Artifact or**  **Deliverable** | **Who (individual**  **or Team)** | **Estimated** | **Actual** | **Difference** |
| RAS | Pavandip Singh | 9 | 14 | 5 |
| RAS | Justin Opraseuth | 8 | 10 | 2 |
| RAS | Mohammed Alam | 10 | 11 | 1 |
|  | Total | 27 | 35 | 8 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Artifact or**  **Deliverable** | **Who (individual**  **or Team)** | **Estimated** | **Actual** | **Difference** |
| SDD initial, final | Pavandip Singh | 9 | 14 | 5 |
| SDD initial,final | Justin Opraseuth | 16 | 14 | -2 |
| SDD initial, final | Mohammed Alam | 11 | 13 | 2 |
|  | Total | 36 | 41 | 5 |

**Cumulative**

|  |  |  |  |
| --- | --- | --- | --- |
| **Who (individual**  **or Team)** | **Estimated** | **Actual** | **Difference** |
| Pavandip Singh | 18 | 28 | 10 |
| Justin Opraseuth | 24 | 24 | 0 |
| Mohammed Alam | 21 | 24 | 3 |
| Total | 63 | 76 | 13 |

**13.5 Gantt Chart**

