Electrical & Computer ENGINEERING

CARNEGIE MELLON UNIVERSITY

C-MART

Web Page Descriptions

Andrew TURNER, Andrew FOX, John PAYNE, Hyong KIM

August 14, 2012

Contents

1 Home Page **2**

2 Login Page **3**

3 Register Page **3**

4 View Item Page **4**

5 My Account Page **5**

6 Update User Details Page **5**

7 Sell Item Page **6**

8 Upload Images Page **7**

9 Confirm Sell Item Page **7**

10 Browse Page **7**

11 Buy Item Page **8**

12 Confirm Bid Page **9**

13 Confirm Buy Page **9**

14 View User Details Page **9**

15 Logout Page **9**

16 Search Page **10**

17 Bid History Page **10**

18 Ask Question Page **11**

19 Answer Question Page **11**

20 Leave Comment Page **12**

21 Confirm Comment Page **13**

At the top of every page there is a header bar with a set of 5 links. The first three links go to the Home Page, the Browse Page, and the Sell Page. The last two links go to the Login Page and the Register page is the user is not logged in, or to the My Account Page and Logout Page if the user is logged in. A search bar to search for items is displayed beside these 5 links. These links should be considered for each page in addition to the specific page details.

For all forms the entry fields are "text" format unless otherwise indicated.

1 Home Page

This page is loaded from the HomeServlet. If the user is not logged in the page displays a simple welcome message. If the user is logged in there is a set of 3 recommended items on the page. These items are determined from an AJAX request from getRecommendationItem.js to the HomeServlet after the page has loaded. The response is an XML document containing the id, name, thumbnailURL, and recommendation page number of three recommended items. There are links next to the recommended items for a previous and next page of recommended items. These links send out another AJAX request from getRecommendationItem.js for more item recommendations. On the HTML5 page the AJAX requests come from getRecommendationItemHTML5.js to handle the slightly different syntax of the page structure.

HTML5
The following is written assuming that the Home Page is the first page the user visits. If that is not true the following will happen on the first page the user visits. If the populated field in the user's browser's local storage is not set to 1 then an AJAX request to the GetBulkDataServlet is made from database.js. A C-MART SQLite database is created in the user's browser with the following tables:

- addresses
- bids
- categories
- comments
- itemcache
- items
- purchases
- questions
- sellers
- states
- users

The response from the GetBulkDataServlet populates the categories table with the id, parentId, and name of every category as well as a time stamp of when the data was loaded. The states table is loaded with the id, shortName, and longName of the US states. Finally the Local Storage is updated so that the populated key has a value of 1.

2 Login Page

The login page consists of a login form with entry fields for a username and password. Upon successful completion of the login the user is automatically directed to the My Account page.

HTML4
The login form data is submitted as a POST request to the LoginServlet. The username and password are checked in the database. If there is a match the user's My Account page is returned. If there is an error the Login Page is returned with the original data remaining in the form and the Error table updated to reflect the errors.

HTML5
The login form data is submitted as a GET request to the LoginServlet. The username and password are checked in the database. A JSON object is returned indicating if the login was successful. If the login was not successful the Errors table is updated from the login.js JavaScript file. If the login was successful the My Account page is returned and the user's authToken and userID are stored in the user's local storage.

3 Register Page

The register page consists of a registration form with the following fields:

- Username
- Password
- Confirm Password
- E-Mail address
- Confirm E-Mail address
- First name
- Last name
- Street
- Town
- Zip Code
- State (select menu)

Upon successful completion of the registration the user is automatically directed to the My Account page.

HTML4
The login form data is submitted as a POST request to the RegisterUserServlet. The fields are checked to confirm that passwords and e-mail addresses match and that there is no existing user already registered with the same username or password. The zipcode and state are checked to see if they are valid entries. All fields in the form must be filled out. If there is an error the Register Page is returned with the original data remaining in the form and the Error table updated to reflect the errors. A successful registration redirects the user to the My Account Page.

HTML5
The login form data is submitted as a GET request to the RegisterUserServlet. The fields are checked to

confirm that passwords and e-mail addresses match and that there is no existing user already registered with the same username or password. The zipcode and state are checked to see if they are valid entries. However before submission HTML5 is used to make sure the e-mail addresses and passwords match and that all required fields are filled out. This prevents the user from submitting the form if one of these errors is to occur. A JSON object is returned from the server indicating a successful registration. If the registration is unsuccessful the Errors table is updated with the returned errors from the login.js JavaScript file. If the registration is successful the My Account page is returned and the user's authToken and userID are stored in the user's local storage.

4 View Item Page

The View Item Page is loaded from the ViewItemServlet. The left side of the page is a table containing details on the item. This includes the seller (and a link to the seller's View User Page), the Seller's rating, the date the bidding ends, the number of bids (and a link to the bidding history), and the current price. The header of the table is the name of the item and above the table is a thumbnail image of the item. Below the table is the desciption of the item and links to Leave a Comment and Ask a Question about the item. Below the links is a table of questions from other users and the answers from the seller of the item. Below the questions is a slideshow of the images that were uploaded for the item. On the right side of the page is bidding information. If a reserve price exists there is a line of text indicating if the reserve has or has not been met. Below this is a table where the user can bid on the item. The fields in the bid form are:

- Quantity
- Bid
- Maximum Bid
- userID (HTML4 only, hidden)
- authToken (HTML4 only, hidden)
- itemID (hidden)
- useHTML5 (HTML5 only, hidden)

Below the table if the item is available for Buy Now there is a button indicating to do so next to the price of the item. The inputs to this form are:

- userID (HTML4 only, hidden)
- authToken (HTML4 only, hidden)
- itemID (HTML4 only, hidden)
- quantity (HTML4 only, hidden)

There are also periodic AJAX requests from itemPriceUpdate.js (HTML4) or database.js (HTML5) every 5 seconds to the ViewItemServlet to update the bid price of the item. The response is the text of the new item price and it is updated wherever shown on the page. For the HTML5 version if the price is changed this is also updated in the items SQLite database table. There are differences in how this page is created and how the forms are submitted for the HTML4 and HTML5 versions which will now be detailed:

HTML4
The bid and buy now forms are submitted as POST requests to the ViewItemServlet and BuyItemServlet respectively. For the bid form, the inputs are checked to make sure the user is authenticated and the bid is valid for the item. If there us are errors the View Item Page is returned with the Errors table reflecting the

errors. For the bid and buy now forms it is possible that the item has already been bought when the item is submitted. In this case the View Item page is returned. If the bid is successful the response is redirected to the Confirm Bid Page. If the buy now form submission is successful the user is sent to the Buy Item Page.

HTML5
The bid and buy now forms are submitted as GET requests to the ViewItemServlet and BuyItemServlet respectively. They are checked for the same errors as for the HTML4 pages and direct to the same pages after. However since the forms are submitted as AJAX requests the entire page does not need to be remade in the event of an error. The response from the AJAX request is a JSON object from which the Errors table is updated. The response also contains the item and the seller in case there is an update since the last visit it can be updated in the SQLite database. A successful bid's details are entered in the SQLite bid database.

When the initial request to the page is made what is returned is viewitem.html which is the shell of the item page on which the item data will be populated. If the item is in the SQLite database and is less than 5 minutes old then the item data is loaded from the SQLite items table and rendered into the page. If these conditions are not met an AJAX request is made to the server to return the item data. Upon the return of this request the data is rendered onto the page and the entry in the SQLite items table is updated.

5 My Account Page

The My Account Page contains a link at the top to update the user's details and then five tables containing the user's interactions with different items. The five tables are Currently Bidding Items, Items Previously Bid On (items that have been sold to other users), Purchases, Items Currently Selling, and Items Previously Sold. Every item has a link to the item page.

HTML4
The entire page is rendered by the MyAccountServlet on the server by selecting the appropriate items from the database.

HTML5
The initial request is sent to the MyAccountServlet and the return is myaccount.html which is a shell for the My Account Page on which the item data needs to be rendered at the client side. A subsequent AJAX request is issued to the MyAccountServlet which contains the time stamp from myAccountTS in the Local Storage which was the last time the My Account Page was accessed. The items returned are those that were updated since the last time stamp. These items are loaded into the SQLite items database and rendered onto the page. Any item that does not require an update is obtained from the SQLite items database and rendered onto the page. the myAccountTS value in the Local Storage is updated with the time stamp of the new response. The bids and purchases tables are similarly updated with the item data.

6 Update User Details Page

This page is loaded from the UpdateUserDetailsServlet after pressing the Update details link on the My Account Page. The page contains a form on which the user can update their personal details. The inputs to the form are as follows:

- Password
- Confirm password
- E-mail
- Confirm E-mail

- First Name
- Last Name
- AddressIsDefault (hidden)
- addressID (hidden)
- Street
- City
- Zip code
- State (select menu)

The page is initially populated with the user's data from the database.

HTML4
The form is submitted as a POST request to the UpdateUserDetailsServlet. The passwords and e-mail addresses are checked to confirm a match and the other inputs are checked to make sure they are valid. If there is an error the page is returned with the input entries in the form and the Errors table updated accordingly. A successful update redirects the user to the My Account Page.

HTML5
The form is submitted as an AJAX GET request to the UpdateUserDetailsServlet. The inputs are checked and if there is an error the JSON response from the AJAX request contains the errors to be updated into the Errors table. HTML5 is used to indicate that ever field is required, that the e-mails and passwords match and are of proper format, and that the zip code matches the proper pattern. The states select menu is populated from the SQLite states table. A successful form submission redirects the user to the My Account Page.

7 Sell Item Page

The Sell Item Page consists of a Sell Item form with the following fields:

- Item Name
- Description
- Start Price
- Reserve Price
- Buy Now Price
- Quantity (number type for HTML5)
- Category (select menu)
- End Date (jQuery Calendar, prevents user from entering past date)

Successful submission of this form returns the Upload Images Page.

The sell form data is submitted as a POST (HTML4) or GET (HTML5) request to the SellItemServlet. The servlet validates that the Item Name has been entered and there are no conflicts between the three input prices. If the reserve price is set to zero there is no reserve price. If the Buy Now Price is set to zero the item is for sale as an auction only.

HTML4
If there is an error as a result of the form inputs the Sell Item Page is returned with the inputs remaining in the fields and the Errors table updated to reflect the respective errors. If the inputs are valid the item is put up for sale and the Upload Images Page is returned so the user can associate images with the item for sale.

HTML5
HTML5 is used to make sure that the Item Name is required, the price inputs are in valid formats and required, and the Quantity and End Dates are required so that the form does not need to be submitted if any of these conditions are violated. If an error is returned from the server the form inputs remain and a JSON object is returned indicating that the sale didn't work and containing the errors to update to the Errors table. If the form inputs are valid the item is put up for sale and the Upload Images Page is returned so the user can associate images with the item for sale.

8 Upload Images Page

This page is loaded as a redirect after an item has successfully been put up for sale. The page contains a form with the following inputs:

- Image1 (file)
- Image2 (file)
- Image3 (file)
- Image4 (file)
- userID (hidden)
- authToken (hidden)
- itemID (hidden)
- useHTML5 (HTML5 only)

This form allows the user to upload up to four images to the item for sale. The form is a POST request to the SellItemImagesServlet for both the HTML4 and HTML5 versions. Upon submitting the form the userID, authToken, and itemID are checked to make sure the user is authenticated and owns the item. The images are then uploaded, compressed, and associated with the item. The first image serves as the thumbnail image for the item. If no images are uploaded, a default "No Image" image is used for the item. Upon successful completion of the form, the user is redirected to the Confirm Sell Item Page.

9 Confirm Sell Item Page

This page is loaded as a redirect after the Upload Images Page. This is a simple page containing only the links in the header bar and a text message telling the user that the item has been successfully put up for sale. This is a static page for the HTML5 version.

10 Browse Page

The Browse Page is created by a request to the BrowseCategoryServlet. On the right side of the page is a table of items in the category clicked (from all items if no category was chosen). Each row in the table is an item with a thumbnail image, the name of the item which is also a link to the item page, the current bid price of the item, and the end date of the auction. In the header of the table there are links to sort the items by Bid Price or End Date. At the bottom of the table there are links to the Previous and Next pages which will return a page with more items to display from the BrowseCategoryServlet. The left side of the page is a list of categories. Clicking a link to one of the categories initiates a request to the BrowseCategoryServlet and the returned page consists of items in that category and the category list on the left side is updated to child categories of the one chosen. How this page is generated and rendered though is very different for the HTML4 and HTML5 pages.

HTML4
For the HTML4 page the entire page is rendered at the server. Items are selected from the database and the appropriate information is added to the Browse Page.

HTML5
The original return from the Browse Page request is the browse.html page. This is just a shell page which will be subsequently populated with the item data. The item data can be populated by a number of different methods. After the browse.html page is returned an AJAX request is sent to the BrowseCategoryServlet to return a JSON request with the item data. This request contains the hasItems field which is a CSV list of the 250 most recently loaded items into the browser's SQLite items database that are less than 5 minutes old. The response from the servlet is a JSON object containing a list of itemIDs which is the order that items are to appear in the rendered table. If any of those items were not on the hasItems list that was sent in the request then the JSON response also contains the item data which is added to the SQLite items database as well as a timestamp of when the data was loaded. These items are rendered onto the page from the JSON response. If the item in the ordered list is not returned in the JSON response then the data is added to the page from the SQLite database entry on the item from the items table. For every item that is returned in the JSON response, an object on the seller of the item is also returned which is entered into the sellers SQLite table. The list of categories on the left side of the page is drawn from the categories SQLite table by getting all child categories of the category currently selected.

11 Buy Item Page

The Buy Item Page contains a form to enter payment information for the item that was desired to be purchased on the previous View Item page. The item is not finally bought until this form is submitted. The following are the fields on the form:

- Quantity
- Street
- Town
- Zip Code
- State (select menu)
- Name on Credit Card
- CVV2 code
- Credit Card Expiration Date
- userID (HTML4 only, hidden)
- authToken (HTML4 only, hidden)
- itemID (hidden)

- addressID (hidden)
- accountID (HTML4 only, hidden)

The form is submitted to the BuyItemServlet. The form is initially populated when loaded with the address information of the user, however this can be changed on the page.

HTML4
The form is submitted as a POST request. When the form is submitted, the form data is checked to make sure the user is valid and the item is still for sale. All fields on the form must be filled out with valid inputs, including the Luhn algorithm on the credit card number. If there is an error on the form, the same Buy Item Page is returned with the inputs the same as submitted with the Errors table reflecting the errors. If the inputs are valid the item is purchased for the user and the user is directed to the Confirm Buy Page.

HTML5
The form is submitted as an AJAX GET request. The same errors mentioned for the HTML4 part are checked on the Servlet. If there is an error the JSON response to the AJAX request contains the error which is updated to the Errors table. If the form data is valid and the item is bought the user is directed to the Confirm Buy Page. The States menu on the form is populated form the SQLite states table.

12 Confirm Bid Page

This page is loaded as a redirect after the View Item Page when a successful bid is made. This is a simple page containing only the links in the header bar and a text message telling the user that the bid has been successfully made. This is a static page for the HTML5 version.

13 Confirm Buy Page

This page is loaded as a redirect after the Buy Item Page when an item has been successfully purchased. This is a simple page containing only the links in the header bar and a text message telling the user that the item has been successfully purchased. This is a static page for the HTML5 version.

14 View User Details Page

The View User Details Page is accessed from a View Item Page when pressing on the Seller's name. The page contains a table with information about the seller of the item. For the HTML4 page the information is obtained from the database at the server, then rendered and sent back to the user as the response. For the HTML5 version, the response to the user is the viewuser.html page which is a shell of the View User Details Page. The data to populate the table for the seller information is obtained from the SQLite sellers table.

15 Logout Page

The Logout Page is linked to from the header bar and the request is submitted to the LogoutServlet. The page displays a message indicating if the user was logged out or if there was an error. Logging out invalidates the user's authToken at the server.

HTML5
When the user logs out from the HTML5 version of C-MART, all the local SQLite tables are cleared. Also the username, userID, and authToken fields are cleared from the Local Storage.

16 Search Page

The Search Page is generated by a request to the SearchServlet. The search form is available in the header on every page. The fields in the search form are as follows:

- userID (hidden)
- authToken (hidden)
- Search Term

The structure of the Search Page is similar to the Browse Page, except there are no categories displayed, only a table of the items as a result of the search. The items can be sorted by Bid Price and End Date in the header of the table and there are links to the Previous and Next Page of results underneath the table.

HTML4
After the search query is submitted, the database is searched for matching items. The appropriate item details are rendered into the Search Page at the server and the page is sent back to the user as the response.

HTML5
The response from the initial page request is search.html, which is a shell of the Search Page on which the item data will be rendered by the client's browser. A subsequent AJAX request is sent to the SearchServlet with the search term and the hasItems field as described in the Browse Page Section. The response is similar to that on the Browse Page. A JSON object is returned specifying the order that the items are to appear in the results table on the Search Page. Any item which is not in the SQLite items database or has an expired timestamp is returned in the JSON response as well as a JSON object detailing the seller of the item. The SQLite databases are populated with this data and the page is rendered with the data from the JSON response. Any item for which the SQLite entry is still valid is rendered from the SQLite items table entry for that item.

17 Bid History Page

The Bid History page is accessed from an item page and is generated by the BidHistoryServlet. It contains a link at the top to return to the item page followed by the name of the item. The total number of bids are shown and if there are more than zero bids on the item a table is shown displaying all the bids. The data in the bids table is the bidder's disguised name (5 characters, with the first and last replaced by the bidder's username's first and last characters, with the middle three characters replaced by asterisks), the amount of the bid, and the time the bid was made. This page is rendered differently for the HTML4 and HTML5 architectures.

HTML4
For the HTML4 page the entire page is rendered at the server. The bids are selected from the database according to the itemID and the appropriate information is inserted into the Bid History Page.

HTML5
The initial return for the Bid History Page request is just the bidhistory.html page. This is just a shell of the Bid History Page which will be subsequently populated with the item and bids data. After the bidhistory.html page is returned an AJAX request is made to the BidHistoryServlet for the bids data. The response is a JSON object containing all the bids on the item. This bids data is used to render the page on the client side by populating the bid history table. The itemID from the initial URL request is used to locate the item in the SQLite items database and populate the page with the item name and link back to the View Item Page.

18 Ask Question Page

The Ask Question Page is accessed from an item page. It is where a user can ask a question to the seller of an item. The Ask Question Page contains a table with the item's name and description and form where the question can be asked. The form contains the following inputs:

- Question
- userID (hidden, HTML4 only)
- authToken (hidden, HTML4 only)
- itemID (hidden, HTML4 only)

The question is submitted in the following manner:

HTML4
The form is submitted as a POST request to the AskQuestionServlet. The user and item information are checked to see if they are valid, and the question is checked to make sure it is not null. If there is an error the Ask Question Page is returned. If the form inputs are valid then the user is redirected to the View Item Page for the item the user was asking the question.

HTML5
The form is submitted as a GET request to the AskQuestionServlet. The userID, authToken, and itemID are added as queries to the request by calling them from Local Storage and the url of the request to get to the Ask Question Page. The inputs are checked at the server for validity the same as HTML4 and HTML5 is used to make sure that the question field is not null. The response is a JSON object indicating if the question was successfully entered into the database. If the question was not successful, the Errors table is updated with the respective errors. If the question was successfully asked the JSON object also contains data on the question. This question data is inserted into the SQLite questions database. The user is then redirected to the View Item Page for the item the user was asking the question.

19 Answer Question Page

The Answer Question page is accessed from an item page. This is where the seller of an item can answer a question that was proposed by potential bidders/buyers of the item for sale. The Answer Question Page contains a table with the item's name as the header and the question being asked in the first row of the table. The third row contains a form with a text box where the answer can be input. The form contains the following inputs:

- Answer
- questionID (hidden)
- userID (hidden, HTML4 only)
- authToken (hidden, HTML4 only)
- itemID (hidden, HTML4 only)

The answer is submitted in the following manner:

HTML4
The form is submitted as a POST request to the AnswerQuestionServlet. The user, item, and questionID information are checked for validity and the answer is checked to make sure it is not null. If there is an error

the Answer Question Page is returned. If the form inputs are valid then the user is redirected to the View Item Page for the item the user was answering the question.

HTML5
The form is submitted as a GET request to the AnswerQuestionServlet. The userID, authToken, and itemID are added as queries to the request by calling them from Local Storage and the url of the request to get to the Answer Question Page. The inputs are checked at the server for validity the same as HTML4, and HTML5 is used to make sure that the answer field is not null. The response is a JSON object indicating if the answer was successfully entered into the database. If the answer was not successful, the Errors table is updated with the respective errors. If the answer was successful, the JSON object also contains data on the answer. This answer data is inserted into the SQLite questions database (the answers contain the same fields as the questions). The user is then redirected to the View Item page for the item the user was answering the question.

20 Leave Comment Page

The Leave Comment page is accessed from an item page. This is where users may comment on the item and leave a rating. The Leave Comment Page contains a table with the item's name as the header, followed by the seller and a link to the sellers page. The description of the item is the next row which is followed by a form with a text box where the comment can be input. The next row of the table contains a select menu where the user can input a rating from 0 to 5. The form contains the following inputs:

- Comment
- Rating
- userID (hidden, HTML4 only)
- authToken (hidden, HTML4 only)
- itemID (hidden, HTML4 only)

The comment is submitted in the following manner:

HTML4
The form is submitted as a POST request to the CommentItemServlet. The user and item data is checked for validity at the server and the comment and ratings are checked to make sure they are not null and meed the proper data input type. If there is an error the Leave Comment page is returned with the Errors table appropriately updated. If the form inputs are valid and the comment is successfully entered into the database then the user is redirected to the Confirm Comment Page.

HTML5
The form is submitted as a GET request to the CommentItemServlet. The userID, authToken, and itemID are added as queries to the request URL by calling them from the Local Storage and the URL of the request to the Leave Comment Page. The inputs are checked at the server for validity in the same manner as HTML4, and HTML5 is used to make sure that the comment field is not null. The response is a JSON object indicating if the comment was successfully entered into the database. If the comment was not successful, the Errors table is updated with the errors from the JSON response. If the comment entry was successful, the JSON object contains the comment data and this is inserted into the SQLite comments database. The user is then redirected to the Confirm Comment Page.

21 Confirm Comment Page

This page is loaded as a redirect after the Leave Comment Page when a comment has been successfully left on an item. This is a simple page containing only the links in the header bar and a text message telling the user that the comment has been successfully left. This is a static page for the HTML5 version.