Code Inspection Report

UMBC Market

Client
Abhay Kashyap

Team 4
Cory Ferrier
Sam Leung
Seth Jenkins
Wesley Chiou
Zachary Robinson

11/30/2016

UMBC Market
Code Inspection Report

Table of Contents

	Page
1. Introduction	2
1.1 Purpose of This Document	2
1.2 References	2
1.3 Coding and Commenting Conventions	2
1.4 Defect Checklist	2
2. Code Inspection Process	5
2.1 Description	5
2.2 Impressions of the Process	5
2.3 Inspection Meetings	6
3. Modules Inspected	6
4. Defects	9
Appendix A – Peer Review Sign-off	12
Appendix B – Document Contributions	14

1. **Introduction**

1.1 Purpose of This Document

This Code Inspection Report will officially define coding conventions for code generated in the production of the UMBC Market application. It will additionally provide record of the inspection process as conducted by the members of the UMBC Market production team. This document is intended to be read and referenced by all active and future UMBC Market production team members and the client.

1.2 References

- Meteor Code Style Guidelines: https://guide.meteor.com/code-style.html
- UMBC Market System Requirements Specifications
- UMBC Market User Interface Document
- UMBC Market System Design Document

1.3 Coding and Commenting Conventions

The coding conventions used in the production of the UMBC Market application were adopted from the Meteor Developers Code Style conventions. These conventions were adopted to ease the first time use, for some team members, of the Meteor framework and because they offer a high level of code readability and adaptability. In addition to these conventions, the file naming scheme follows a simple format, associating each .html file with a .js file of the same name, where both are used in conjunction to generate a page of the application. For example, a profile page would be generated using ‘profile.html’ and ‘profile.js.’

Comments on code were generated where an explanation of a function or methodology was necessary, at the discretion of the code author.

1.4 Defect Checklist

The following is a list of potential defects looked for during formal and informal code inspections. The defect categories include general bugs, coding style and oversights.

Defect	**Category**
Logic errors	General bugs

Misuse of “==” vs “===”	Coding style
Unclear comments	Coding style
Dead code	Oversights
Incorrectly formatted comments	Coding style
Unnecessary logical behavior	Oversights/general bugs
Security flaws	Oversights
Bad code styling	Coding style
Optimizations	Oversights
Syntax errors	General bugs
Duplicated code	Oversights
Unnecessary files	Oversights
Commented out (but not removed) code	Oversights
Logical oversight (different from logical errors)	Oversights
Unused function and variable declarations	Oversights

2. **Code Inspection Process**

2.1 Description

The members of the UMBC Market production team conducted code inspection by two different processes, one informal, and the other formal. The informal inspection process was performed individually by each team member before committing any modifications to the Git repository. This is to say, before any team member committed files or modifications to the repository, they proofread the code that they wrote or modified to the best of their abilities, checking for error and for consistency with coding convention.

The second and formal code inspection process was conducted collectively. The project implementation leader first individually inspected each file of the application, compiling notes. The implementation leader then met with members of the team who participated in writing code, all together, and again went through each file line by line inspecting the code. This time, though, at each relevant point, the implementation leader discussed with team members the previously compiled notes. Modifications were made to code when a consensus was met between participating team members that there was a defect in the code. This process was conducted once, after all system requirements had been implemented.

2.2 Impressions of the Process

Both inspection processes undertaken were effective. The informal inspections conducted individually by team members kept code clean, readable, and modifiable by all team members. The formal code inspection was then less cumbersome because of how clean and manageable the code had been kept throughout the entire implementation process.

The formal inspection process certainly has its merits as well. Through this process we were able to identify larger pieces of code that, while sticking to convention, could be optimized or implemented in a more efficient or succinct way. It was also a way for team members to discuss strategy, methodology, and ideas openly between one another.

When conducted in this two part process, the individual team members retain a large amount of responsibility for their own code, which is desirable. The individual and informal inspections then allow for a more manageable formal inspection. The implementation leader in this case isn't micromanaging the process of each author, but instead acts as more of a proof-reader or editor who can facilitate discussion between group members about methodology and reasoning behind implementation decisions.

The best modular unit of the application is the listings unit, comprised of the following files: api/listings.js, ui/listings.js, and listings.html. These three files are some of the oldest of the application and have gone through significant modification. Because

of our informal inspection process, a large number of modifications results in a large amount of inspection, and so this code is nearly perfect.

The worst modular unit of the application is the error detection unit, comprised of just one file: errorDisplay.js. It is in the worst condition for the same reason that the listings unit is in the best condition: the level of informal inspection. This unit is the most recently added unit to the application, and so has gone through the least amount of informal inspection.

2.3 Inspection Meetings

Formal Code Inspection Meeting

Date: 11/30/2016
Location: UMBC Commons
Start time: 12:00pm
End time: 12:45pm
Participants: Seth Jenkins, Zachary Robinson, Cory Ferrier, Sam Leung
Moderator: Seth Jenkins
Scribe: Seth Jenkins
Authors: Zachary Robinson, Cory Ferrier, Sam Leung
Units Covered: Entire Application

3. **Modules Inspected**

The following section identifies each module of the UMBC Market application, along with its implemented functionality and relation to the design expectations.

Module	Functionality	Design/Architecture
client/main.js	Sets up client with routing	N/A
api/conversations.js	Defines methods and database communication for implementation of user to user conversations	Fairly consistent with the 'messageChains' referenced in the SDD. Primary differences are conversations instead consist of the ID of the listing the message is about, the ID of the first participant, and the ID of the second participant instead of arrays. Messages for each conversation are found by doing a MongoDB filter for messages with parameters matching those

		3 id's. This saves on space as a tradeoff to a minor penalty in time.
api/listings.js	Defines methods and database communication for implementation of product and service listings	Instead of multiple images, there are is currently only 1 image per listing. This is due to an unforeseen performance penalty incurred by storing images in MongoDB. Better solutions would involve storing images on a 3rd party platform, but that would cost money regardless of the service chosen.
api/messages.js	Defines methods and database communication for implementation of user messages	No changes from documents
api/profiles.js	Defines methods and database communication for implementation of user profiles	Listings is no longer an array here, it follows the same pattern as conversations and simply shows listing with a mongoDB filter for that user applied. This saves on space at a minor time penalty. The same holds true for messages by that user.
accounts-ui-config.js	File for setting up built-in Meteor account functionality	Meteor required bootstrap file. Not included in Documentation.
routes.js	Defines all potential routes within application	Logic is consistent with Figure 3.1 in UID
ui/ApplicationLayout.html	Template for basic page layout used universally within application, including separately defined header template	Sorting, Tags, and Help/Faq are currently missing. Sorting potentially to be added later. Since search bar searches descriptions as well as title they are redundant, and Help/FAQ was deemed unnecessary for the simplicity of the application.
ui/conversation.html	Template for conversation UI	Missing from UID. N/A
ui/conversation.js	Handles back end	See comments for

	functionality, including database communication and routing, for conversation unit	“conversation.html”
ui/edit_listing.html	Template for UI for editing listings	Missing from UID, but same as Figure 2.2(Create Offer Page)
ui/edit_listing.js	Handles back end functionality for editing listings	See comments for “edit_listing.html”. Consistent with SDD figure 2.1 and 2.2.
ui/header.js	Handles back end functionality for universal application header	See comments for ‘ApplicationLayout.html’
ui/listing.html	Template for single listing UI	Consistent with Figure 2.0 in UID. Missing tags and only has a single image. See comments for “api/listings.js” and “editlisting/js” for reasoning.
ui/listing.js	Handles back end functionality for single listing page	See comments for “listing.html”. Consistent with SDD figure 2.1 and 2.2.
ui/listings.html	Template for multiple listings (list of listings) UI	Consistent with UID Figures 1.0 and 1.0. Tags are missing due to reasons discussed above.
ui/listings.js	Handles back end functionality for pages consisting of multiple listings, including home, search results, and user-specific listings.	See comments for “listings.html”. Consistent with SDD figure 2.1 and 2.2.
ui/message.html	Template for message creation UI, specifically when sending a message straight from a listing page	Consistent with UID FIgure 2.1. Subject line is not included because the subject is simply based on the name of the listing (per clients request)
ui/message.js	Handles back end functionality for message creation page	See comments for “message.html”. Consistent with SDD figure 2.1 and 2.2.
ui/messages.html	Template for UI showing	Consistent with UID Figure 2.3.

	all messages of a particular user-user conversation	Send new Message button is missing because messages are tied exclusively to listings by clients request.
ui/messages.js	Handles back end functionality for messages.html	See comments for “messages.html”. Consistent with SDD figure 2.1 and 2.2.
ui/new_listing.html	Template for UI for creation of a new listing	Consistent with UID Figure 2.2. Tags are missing as are additional pictures, for reasons stated above.
ui/new_listing.js	Handles back end functionality for creation of a new listing	See comments for “new_listing.html”. Consistent with SDD figure 2.1 and 2.2.
ui/profile.html	Template for profile page UI	Based on Figure 3.0 in UID. Very different because of various changes in application to reduce amount of information on each page. Now contains basic information on the user and a button to view listings by that user. Account management such as changing password are done through the button at the top of the header.
ui/profile.js	Handles back end functionality for displaying user profiles	See comments for “profile.html”. Consistent with SDD figure 2.1 and 2.2.
server/main.js	Starts server and defines some user account creation and use functionality	Meteor required bootstrap file. Not included in Documentation.
main.css	Defines commonly used UI specifications	N/A

4. **Defects**

The following section contains record of the specific defects found during the formal code inspection, including the module location, category, and description of each defect:

Module name	Conversations
Category	Unnecessary logic/optimization: Other
Description	Removed “!” operator and replaced > with <=

Module name	Listings
Category	Bad Code Styling: Commenting
Description	Old TODO’s never removed

Module name	Messages
Category	Old commented out code: Commenting
Description	Old code that was commented out not removed

Module name	Messages
Category	Unused declaration: Other
Description	Unused variable “id”

Module name	Header
Category	Unnecessary declaration/optimization: Other
Description	Variable target created and used exactly once.

Module name	Main server
Category	Bad code styling: Commenting
Description	Old TODO’s not removed

Module name	Main server
Category	Bad code styling: Commenting
Description	Commented out code never removed

Module name	new_listing
Category	Bad code styling: Code conventions
Description	No spaces at several points between operators

Module name	Header
Category	Dead code: Correctness
Description	Dead code: breaks placed after returns inside switch statements

Module name	Listings
Category	Optimizations: Other
Description	Unnecessary if(condition) that returned true or false. Just return (condition)

Module name	Message
Category	Correctness
Description	No handler for unknown error

Appendix A - Team Review Sign-off

Each team member has reviewed this document and approves of its content and format. Any minor points of contention are addressed in the comments below.

Team

Name(print): Cory Ferrier Date: 10/31/2016

Signature: Cory Ferrier

Comments:

_____

_____

_____

Name(print): Zachary Robinson Date: 11/30/2016

Signature: Zachary Robinson

Comments:

_____

_____

_____

Name(print): Sam Leung Date: 11/30/2016

Signature: Sam Leung

Comments:

_____

_____

_____

Name(print): _____ Date: _____

Signature: _____

Comments:

_____

_____

_____

Name(print): _____ Date: _____

Signature: _____

Comments:

_____

_____

_____

Appendix B - Document Contributions

Zachary Robinson contributed significantly to the drafting and editing of this document, specifically including drafting sections 1.1, 1.2, 1.3, 2.1, 2.2, and 2.3, and contributing heavily to sections 1.4 and 3.

Cory filled in the third column (comparing SDD/UID/SRS to Actual) In section 3.

Seth Jenkins managed the code inspection process and took detailed notes on missing pieces.

Sam Leung reviewed the document and made small edits as needed.