College of Saint Benedict & Saint John's University
Computer Science Department

GABeS

Phase 4

Team Potatoes

Grant Boyer, Kyle Olson, Tom Husen

[This page has been intentionally left blank]

Table of Contents

Accessing the System

URL: http://tomcat.csbsju.edu/csci331/tehusen/gabes/index.html

Sample Admin:
Username: tehusen
Password: p@$$word

Sample Regular User:
Username: irahal
Password: puppies123

Phase Summary

The way our team approached this phase was through both division of labor yet simultaneously maintaining close collaboration throughout. We began similar to how we began the third phase by creating a spreadsheet (Appendix A) of everything that needed to get done, who was going to be responsible for what, and how things would work together.

Working our way through the system description document, as well as our own documentation we created a list of all functionalities we would need to complete to make the best system possible. Using a draft format, we all laid claim to tasks we wanted to take on, and also tried to align tasks that would be similar (i.e. Admin Reports) to one person. This promoted consistency and saved us from having to re-invent the wheel. For a more detailed task decomposition please see the *Task Decomposition* page included in this report on page 12.

Grant focused primarily on the feedback aspects of the system (both viewing and leaving of feedback) in addition to the reports for admin users and listing my items up for auction. Using aggregate functions and advanced SQL statements, the database is queried for information about sales, commissions, and feedback for all sales processed through our system. Averages are calculated, records are sorted into groups, and all of this information is presented in a very useful and efficient manner. Additionally, when leaving feedback for a transaction, a check is included to prevent feedback from being left multiple times. Once feedback is left for a transaction rather than a *Leave Feedback* button appearing a thank you message lets the user know they have already left feedback. Once feedback is left for someone, they can now view that information from their *My GABeS* menu.

Kyle focused mostly on item and bidding facets of the system. Key functionalities such as show item info or bid on an item were all part of Kyle's contribution to our final product. Additional aspects including show list of bidders, show items bid on, and show items won were all also included in Kyle's work for this phase. Using sophisticated integrity checks, Kyle's bidding system prevents any bids being submitted that would be considered invalid. His pages also have detailed instructions so users can quickly and easily understand how the bidding works to ensure they have a successful bidding experience. His item reports also contain a wealth of information and provide quick access to relevant actions such as bidding on an item or getting even more information.

Tom worked mostly on user-side functionalities as well as 'outlier' functionalities. Essential operations such as login or add new item were completed by Tom in addition to updating a user profile, adding a new user, and search. One of the unique and standout features of our site is the excellent user interface. During the login functionality, the credentials of the user are established and follow them throughout their visit to our system. The system is smart enough to know when a user is an admin and will include the additional *Admin Options* only when appropriate. The search is also unique in that you can search using any combination of the fields, you are not limited to only certain combinations. In addition to the search function included under the *Shop* menu, Tom also developed a quick search bar for the homepage.

Within this search bar you are able to search by keyword and get a list of results that allow you to submit a bid very fast.

Reflection

In the fourth and final phase of our database systems project we were able to accomplish so much more than we ever anticipated when we began this project more than three months ago. From the beginning steps of trying to understand what this 'GABeS' system was all about, to designing a fully functional and beautiful website of our own we have learned and accomplished so much.

In our final submission, we have completed all of the base requirements as well as several additional ones.

One of the few things we omitted was the use of two distinct login pages. We approached this project from the view of a potential client or end user who would end up using this website on a regular basis. What we decided was it is very important for an admin to be able to accomplish things a standard user cannot, but it is also important that an admin is able to interact with the system in the same ways a standard user can as well. With this in mind what we did was incorporate the admin functionalities into a menu drop-down from our main menu bar. A variable within the *User* JavaBean establishes whether the currently logged in user is an admin or not, and only displays accordingly. By omitting distinct login pages, we were able to make our system more efficient by eliminating redundant pages and simultaneously significantly improving the end users experience when interacting with the site – both as an admin or as a standard user.

In addition to the extra GABeS functionalities we added, which will be discussed later in this report, we also added several elements to our website to improve the quality of our final product. One such feature is related to leaving feedback for an item that you have won. Once you win an auction, the newly won item will move from *Items I've Bid On* to *Item's I've Bought*. When visiting the second page there is a button for you to add feedback about the transaction. One thing our group added to this page was that once a user has left feedback about a transaction, rather than seeing a button to leave feedback, they will see a message thanking them for already leaving feedback about this transaction. This ensures no conflicts within the database, as well as allows the user to quickly and easily see which transactions they have and have not left feedback for.

In the interest of creating the best user experience possible, we knew it was crucial to include functionalities such as *Leave Feedback*, *Add New User*, *and Update Item Info*. These are three important things to have in a system such as ours for several reasons. First, they allow the system to be easily managed, updated, and maintained. Without operations like these, manual inserts into the database would be required on many circumstances which leads to an

increased likelihood of redundancy and poor data integrity. Additionally, as a user these functionalities are very basic when trying to use a bidding service such as ours. Without having these abilities, the scope of what you are able to accomplish is severely diminished.

We also added a sophisticated searching protocol. Rather than having to generate multiple search methods based on various combinations of inputs, our search takes into consideration any information you give it and uses it to narrow results. The only exception is when an *Item ID* is entered that is the only field considered – since there will exist one and only one item with that ID. By only having one method that is extremely flexible we cut down on code redundancy as well as ensure a complete user experience free of any glitches that could result from an unforeseen search combination. Another part of our search is a *Quick Search* bar on the homepage. This allows you to do a very fast search by keyword only.

Overall, we completed all of the requirements and then some. The only drops we had from the original requirements document were in the sake of user experience and took absolutely no functionality away from the system. We added an enhanced experience for nearly every operation and ended up with a system that is both functional as well as highly intuitive to use. As an entire group we were extremely proud of ourselves and all we have accomplished. After sinking countless hours into laying out the system and database, creating that database, and linking to an excellent web experience our understanding of the concepts encompassed herein are far beyond our loftiest goals.

It is our opinion that not only have we met the requirements to receive an A but we have in fact surpassed them. With our very stable and intuitive web interface, included additional features, normalization analysis, advanced SQL queries and procedure usage, and security and integrity checks built into the front end, our GABeS system is unmatched. For the reasons encompassed within this report and as demonstrated with our website we feel we deserve an A on this phase of our project.

Argument

As discussed previously in this report and demonstrated in our live demo, all 12 required functionalities are included and work flawlessly. We also included 4 additional requirements as they relate to the system.

The first additional requirement is 3 GABeS features from the description document, but not incorporated within the phase 4 requirements. These functionalities include: *Add Feedback, Add New User,* and *Edit Item Info*. Add Feedback is linked to from the *Items I Won* page which is populated once you win an auction. From this page, you are able to leave feedback including delivery score (1-10), quality score (1-10), and overall score (1-10). You are also able to leave comments that the seller can then see and use to improve their selling. A unique feature to our site is the once you submit feedback for an item you are no longer presented with the option to leave feedback. Rather you see a thank you message for your previous feedback. The Add New User functionality works as you would expect. Only available to admins, this form takes all the necessary inputs (username, password, email, etc.) and then adds this user to the database. It is a pretty straightforward functionality. Finally, the Edit Item Info functionality works how you would expect as well. A user can edit their posted items so long as their auction time has not ended. You're only able to edit the name, category, description, and end date for these items, the rest of the attributes are unchangeable.

Security checks and error handling are also a big part of what sets our system apart from the rest. As discussed earlier, using a function within our *User* bean, we are constantly able to tell whether the current user is an admin or not. This is relevant when displaying the menu bar as the *Admin Options* dropdown will only appear to those users whom it applies. An additional security feature we have built into the system is if a non-admin user attempts to access any of these admin only features using a direct link path (i.e. GABES/User/SalesSummary.jsp) entry to the address bar, our security checking will recognize the lack of credentials and re-direct the request to the welcome page. Another aspect of error handling included is within the *List New Item* and *Search* functionalities. One of the most delicate entries into our database happens to be dates due to the formatting required to use TIMESTAMP functionalities in the SQL server. To combat this sensitivity, there are checks built into these pages to determine whether the date entered meets the requirements. If so, it proceeds with the request. If not, it redirects to an error page informing the user of the problem and then allows them to retry their operation. As discussed previously in this report, there is also validity checking in relation to bidding which prevents invalid bids – a key aspect to ensure is correct as it is the main function of the entire system.

Also included in this report is the normalization analysis for all tables within our database and an explanation relating to these normalized tables. The complete normalization analysis can be found on the page 9.

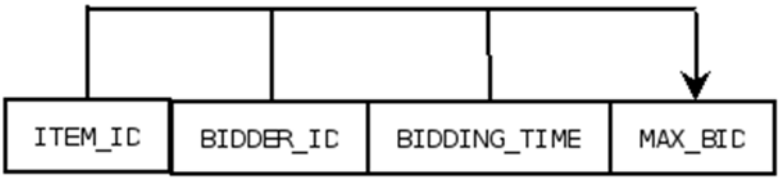
As clearly relevant when interacting with our site, there is a sophisticated web interface front-end to our system that makes the user experience pleasant, efficient, and productive. All of the

formatting throughout the site is accomplished through CSS (potatoes.css) which provides a seamless experience for the user. Included on our login page as well as the homepage to the system there is also graphics to make the users visit more enjoyable (Turn your volume on and click one). Many of the functionalities that we use to make our website as good as it is would be impossible without a deep understanding of the abilities of HTML, CSS, Java, and how they can all work together to make something greater than the sum of the parts. We are fully confident that the end user of our product will be completely satisfied with their experience visiting our gorgeous site.

To a lesser extent than the other requirements, we should also note that we included elements of other options as well. We used Oracle Sequences to generate Item IDs and User IDs to ensure no repetition and continued availability of new IDs. We also used a fair amount of stored procedures, functions, and triggers within our system. One of the more creative uses of these is updating time. Our original plan was to use an Oracle Event to check every 15 seconds or so and fire a procedure to update the database. When trying to implement it however, we ran into permissions issues requiring us to take a new approach. The way our system handles the time checking is by using a procedure (named CHECK_TIME) which will update all required info when the current timestamp is past the end date. Though it could be considered kind of a work around, we call this procedure every time our site accesses information relating to the item, feedback, or bids tables. By checking whether the end date has passed yet, we are able to essentially maintain up to date records, though not truly up to date.

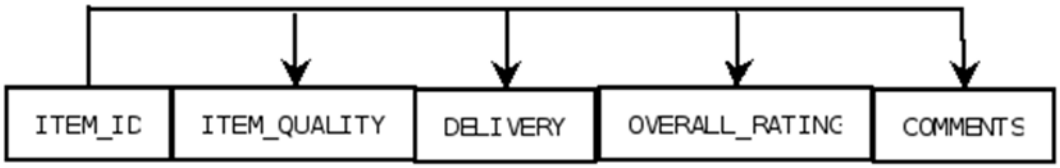
Normalization Analysis

The **BIDS** relation is currently in BCNF. The relation is in 1NF because it has a candidate key (*ITEM_ID*) that spans the entire relation. It is also a minimal key because no subset of this key can span the entire set. It is in 2NF because there are no partial dependencies on the key. Every attribute depends on the entire key. It is in 3NF because there are no transitive dependencies. Every attribute derives its value directly from the key. This also makes it in BCNF because attributes are only dependent on the key.



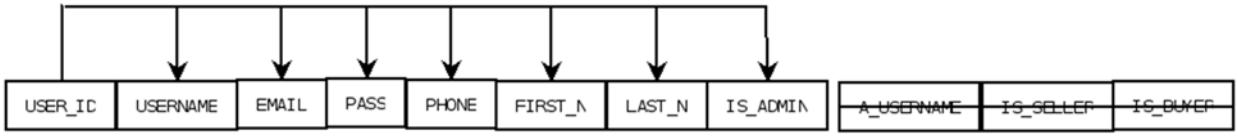
BIDS Relation

The **FEEDBACK** relation is also currently in BCNF. It's in 1NF because it has a candidate key (*ITEM_ID*) that spans the entire relation. We know this key is minimal because it is a single item key that derives a value from nothing. The relation is also in 2NF because there are no partial dependencies. Since this key is only a single item we know that no attribute can depend on only part of this key. It is in 3NF because there are no transitive dependencies. Every attribute derives its value directly from the key which also means it is in BCNF as well.



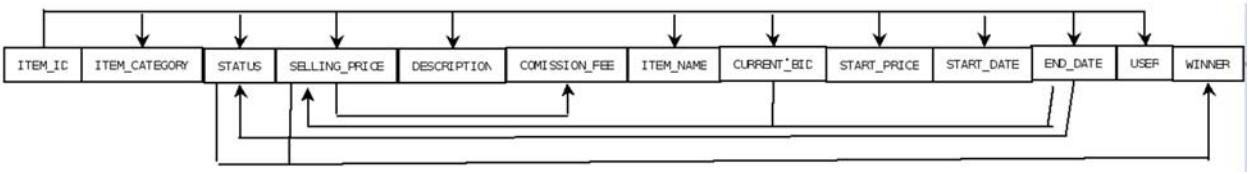
FEEDBACK Relation

The **USER** relation is also in BCNF. During the process of normalization, we also realized the attributes: *A_USERNAME, IS_SELLER, and IS_BUYER* were not relevant to our implementation so we could drop them completely. This relation is in 1NF because it has a candidate key (*USER_ID*) that spans the entire relation. We also know this key is minimal because no other attribute determines anything else. It is in 2NF because there are no partial dependencies on the key. It is also in 3NF because there are no transitive dependencies meaning every attribute derives its value directly from the key. This also means it is in BCNF as well.



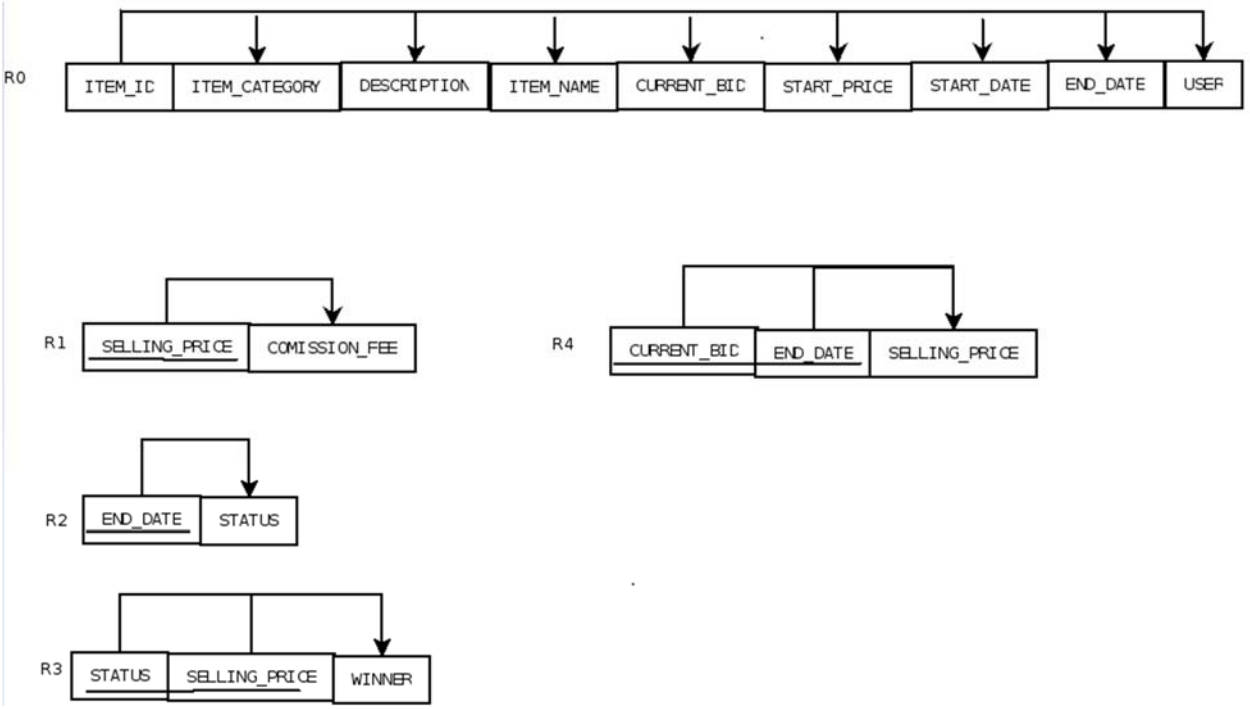
USER Relation

The **ITEM** relation is currently in 2NF. It is in 1NF because it has a candidate key (*ITEM_ID*) that spans the entire relation. We also know it's minimal because nothing determines the value of it. It is also in 2NF because there are no partial dependencies – every attribute depends on the entire key.



ITEM – 2NF

To make this relation we have to eliminate the transitive dependencies that are creating all the lines on the lower part of the relation. We will create 4 new relations (as shown below) which will eliminate these transitive dependencies. At the same time, these new relations also make the relation in BCNF because each attribute depends only on its candidate key.



ITEM - BCNF

Issues Faced

During this 4th phase of our project, we faced several issues, but thankfully none proved to be too much to overcome. As with any development issues included methods or files not behaving as expected, a problem being more complex than it originally appears, or SQL information not always being what you expected impacted us throughout the phase but in the end nothing caused too much of a headache.

One of the more frustrating issues we were faced with during this phase popped up about a week into development – once we had about three-quarters of the functionalities working properly. During phase 3 we had created an SQL file that created all the stored procedures, functions, views, etc. in one simple script. Due to some funky things occurring with our records after so much trial and error we decided to run this file again and get a clean start. What we did not realize however is that during the course of developing the first chunk of functionalities we had also changed some SQL procedures, triggers, etc. in the database but never in our creation SQL file. This lead to some of the functionalities that had been working, to break and leave us utterly perplexed for a while. Upon discovering the issue, it did take a little bit to right the wrongs we had done a week earlier but eventually got everything straightened out.

Considering the scope of this project and this phase in particular, we did quite well when it comes to major roadblocks. Small things here or there slowed us down but we did an excellent job working as a team to progress through these obstacles and figure out a good solution.

Task Decomposition

Grant:

- Feedback JavaBean
- Admin Sales Summary
- Admin Commission Report
- View My Feedback
- Add New Feedback
- List My Items

Kyle:

- Bids JavaBean
- Show Item's Info
- List of Items User Bid on
- List of Items User Won (Bought)
- List of all Items for Sale
- Edit Item's Info
- Show List of Bidders for an Item
- Bid on an Item

Tom:

- User and Item JavaBean
- Add New User
- Add New Item
- Update Profile
- Login/Logout
- Search

All Team Members:

- Populated beans with methods pertaining to our respective functionalities
- Normalized the tables from our database
- Implemented an extremely user-friendly UI using in depth CSS/HTML
- Tested and tested and tested every functionality to ensure completion

Team Potatoes Minutes
December 1-3, 2016

Meetings began at 9:00 pm.

In Attendance:

- Grant Boyer
- Kyle Olson
- Thomas Husen

Grant:

- Started creating page shells for JSP/HTML
- Started work on JavaBeans

Kyle:

- Started creating page shells for JSP/HTML
- Started work on JavaBeans

Tom:

- Created Login stuff
- Started creating page shells for JSP/HTML
- Started work on JavaBeans

All:

- Based off of requirements and past phases created a spreadsheet to divide up work between team members
-

Meetings adjourned at 12:00 am.

Team Potatoes Minutes
December 4-6, 2016

Meetings began at 8:00 pm.

In Attendance:

- Grant Boyer
- Kyle Olson
- Thomas Husen

All:

- Continued working on tasks assigned in spreadsheet
- Finished assigned JavaBeans
- Worked in tandem to solve any problems that we experienced either individually or as a team
- Developed how our website will look/function/etc.
- Began testing functionalities as they became operational

Meetings adjourned at 12:00 am.

Team Potatoes Minutes
December 7, 2016

Meeting began at 9:00 pm.

In Attendance:

- Grant Boyer
- Kyle Olson
- Thomas Husen

Grant:

- Finishing touches on leave new feedback functionality
  - Added restriction about reviewing a transaction multiple times
- Began generating information for the phase report

Kyle:

- Continue work on editing an item's info.
- Put finishing touches on bidding functionality

Tom:

- Continue work on search functionality
- Compile various versions and various files into our master repository

All:

- Continued testing the website, trying out many different scenarios to improve reliability and consistency throughout the entire user experience
- Assisted each other with any blockers they experienced with their assigned tasks
- Continued implementing CSS styling throughout the site

Meeting adjourned at 12:00 am.

Team Potatoes Minutes
December 12, 2016

Meeting began at 8:30 pm.

In Attendance:

- Grant Boyer
- Kyle Olson
- Thomas Husen

Grant:

- Continuous testing of the system.

Kyle:

- Continuous testing of the system.
- Finish error catching/security protocols

Tom:

- Continuous testing of the system.
- Finish Search and Quick Search

All:

- Assisted each other with any blockers experienced
- Performed normalization analysis for our database

Meeting adjourned at 2:30 am.

Appendix A:

[Link to Full Spreadsheet]

	Progress	Assignment	Similar To...
Login (Both)	done	Tom	
View Sales Summary (Admin)	done	Grant	
View Commission Report (Admin)	done	Grant	
Update Profile (Both)	done	Tom	-
List My Items (Both)	done	Kyle	
Show Item Info (Both)	Done	Kyle	
List of Items User Bought		Kyle	
Show List of Bidders (Both)	Done	Kyle	
Add New Item (Both)	done	Tom	Add User
Search (Both)	*create query to get the info, then use Kyle's item list to display results. Ins	Tom	
Show Item Info (Both) - Same as above	done	Kyle	
List of All Currently Up for sale	done - Has its own item info page jsp file		
Bid on an Item (Both)	done	Kyle	Add User
Show Items Bid on (Both)	done - fix to display current winner, not final winner(maybe use a flag indic	Kyle	
View my Feedback (Both)	done	Grant	
Add User (Admin)	done	Tom	-
Leave Feedback (Both)	done	Grant	Add User
Edit Item (Admin) - Only certain fields	Almost done - item session bean issue - wont save	Kyle	Update Profile
Normalization of Tables		Group	
Make bad ass with CSS/JavaScript	done? make more bad ass?	Group	
Java Beans			
User Bean	done	Tom	
Item Bean	done	Tom	
Feedback Bean	done	Grant	
Bids Bean	done	Kyle	
GitHub: https://github.com/tomhusen/Database_Project			