Memorandum

To: Alka Harriger, CNIT 25500 Professor

From: The openLibrary team:  
Tai Gunter  
Ethan Madden

Date: April 28, 2013

Subject: Team openLibrary‘s CNIT 25500 Team Project Summary Evaluation

Enclosures: APPENDIX A: Table of Individual Contributions   
APPENDIX B: Testing Instructions   
APPENDIX C: List of Team files   
APPENDIX D: Final Usability Tests with results and analyses  
APPENDIX E: Client Evaluation/Feedback   
APPENDIX F: Peer Evaluation/Feedback1   
APPENDIX G: Team's Time Logs

Related: *To be sent under SEPARATE cover:* Individual evaluations from each team member

This report summarizes our experiences in developing an operational C# desktop application for **Chaz Brown**. It also provides information on our individual contributions to various aspects of the project. In addition to the technical aspects of developing and implementing this project, we also learned the following general, valuable lessons:

* Learning and implementing new features into C#
* Working with a client to meet his needs
* Working with a partner to do a large amount of work in a short period of time

# Use of Feedback from Each Source

Getting feedback from several different people was absolutely critical when developing this application. When programming, it is easy to get caught up in your own features and forget that not everyone is as familiar with the program as you are. Furthermore, not everyone using the program will have as much computer skill as you do. Having users test the application really helped us know what we were doing well and what we could improve on. Not all features that we thought were easy to use and intuitive were viewed the same way by our user base. Also, having the peer evaluations was a big help. Getting feedback from other programmers is useful too, because they might have ideas for different features that we simply didn’t think of. Finally, instructor and TA feedback is helpful because they’ve done this before and know what to look for. Getting feedback from them helped us to improve our application even further.

|  |  |
| --- | --- |
| SOURCE | FEEDBACK SUMMARY AND IMPACT ON PROJECT |
| Client | Told us what he was looking for, allowed us to expand into a more business-centered application |
| Users (usability tests) | Gave us feedback from users that are not as familiar with computers; this helped  us to develop an application that worked well and was easy to use, even if the  user isn’t familiar with the program. |
| Class (peer evaluations) | Gave us good insight to what other programmers and developers thought of the application, they also had good ideas from improvements. |
| Instructor and TA (milestone feedback,  office hours,  presentation feedback) | The feedback we received helped us to know that we were doing everything as  expected and that we were on track compared to others in the class. |
| Team | Only having two members lessened this effect, but it was nice to have someone  else to talk to when we ran into a problem. If one person developed a new  feature, it was likely that the other had a way to make it better. |

# Discussion of Positive Lessons Learned

Throughout the development of this application, we each learned several valuable lessons. So many of the features in our program weren’t discussed in class, so we had to learn how to code and implement these features. On several occasions, we had to reference websites and documentation to learn how to implement a feature or write a particular section of code. Most of the time, the code we were reading wasn’t doing exactly what we needed, so we had to take that code and modify it to work with our program. Reading the code of others isn’t an easy task, and it was something that both of us got good at throughout the development of the program.

Working with a client was also a new experience for us. In the past, we’ve developed applications and written programs, but they were just for ourselves or for someone else who had already put forth very specific guidelines. In this case, we had to constantly be in contact with the client to make sure the features we were implementing would work as he expected them to. We also were able to surprise the client with a few new features he wasn’t planning on, which the client liked. This experience taught us to always think of who you’re programming for first, because they are the ones that will evaluate the application in the end.

Finally, this project taught us both a lot about working on a very tight schedule. There was a lot to get done in this project, and over the approximate six week timeframe, we had to write a lot of code. Since we were expected to double the requirements for a four-person team, with only half the members, we had an approximate workload of twenty-four weeks to complete in six weeks. This meant that between other classes, projects, tests, and exams, we were always improving the program. There were many late nights and a lot of coding without the presence of the other person. This taught us the importance of time management and planning ahead in the creation of the application. Having only two people in the group ended up being a benefit; there were less disagreements and the could be written faster.

# Discussion of Suggested Improvements

After development of the application had completed, we looked back on what we would have liked to accomplish had we been given more time to code. The thing that we both noticed was lacking from our project was the ability to fine people based on how overdue their book was. Currently, our program is set up to give overdue notices to people who have overdue books, but this is the only way that overdue books are pointed out.

The application could very easily implement a feature that told the user how much a given customer owed in fees, but we chose not to include that feature because we didn’t have the time to *fully* implement it. In our opinion, fully implementing this feature would mean that we would have to add an entire sales system to the application. There would be some situations where customers would simply pay the fines and be on their way, but other situations that would involve the book needing replaced. We are pulling the cost of the book from Amazon, so that wouldn’t be hard to implement, but the sales system would be complicated enough (different payment methods, tax, etc.) that we simply didn’t have time to implement it. As a group, we felt that choosing not to implement a feature at all was a better decision than implementing the feature but not doing a very good job.

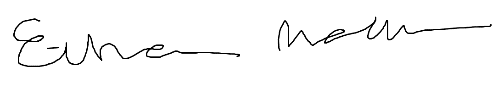
# Discussion of Future Implementation

Our project is fully featured at this time and could be used by the client for his intended purposes. As a home system, it’s fully ready to go. It meets all requests of the client and does not lack any features necessary for home use. Since it is an open-source project, our client as well as other people that would like to use the application are free to use it and modify it.

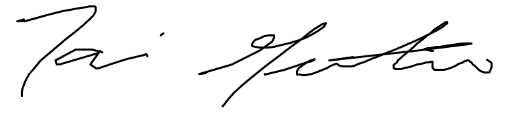
This also means that a corporate library could, in theory, download and use our application. While it has a lot of features, we do not feel that it is ready for use in a corporate environment. There are a few features that would still need to be added (security, fining system), before we as a team would feel comfortable releasing this for recommended use in a corporate environment.

# Conclusion

In conclusion, this project was a unique experience for both of us that forced us to think of our project from a number of directions that we were not previously accustomed to. We saw our project from the client's and users' perspectives, from the perspective of our classmates, and still got the input from our TA and instructor that we are more used to. We also learned a lot about time management and teamwork, mostly due to the complexity of the project. This was the first project that either of us had been allowed to design from the ground up. This caused us to invest ourselves much more heavily into the project than one that was just assigned to us, because it was personal and interesting.



Ethan Madden



Tai Gunter

APPENDIX A: Individual Contributions

**Individual work contribution percentages**

|  |  |  |  |
| --- | --- | --- | --- |
| **Team Member** | **Total** | **Tai Gunter** | **Ethan Madden** |
| **Milestone reports and related documentation** | 100% | 40% | 60% |
| **Development of specific methods/classes (listed below) (See note 1 at the bottom of this table)**  **frmAbout.cs**  **frmAddBook.cs**  **frmAddCustomer.cs**  **frmAddEmployee.cs**  **frmAddGame.cs**  **frmAddMovie.cs**  **frmAddMusic.cs**  **frmAmazonKeys.cs**  **frmBugReport.cs**  **frmClockIn.cs**  **frmClockOut.cs**  **frmCurrentlyClocked.cs**  **frmEditBook.cs**  **frmEditCustomers.cs**  **frmEditEmployee.cs**  **frmEditGames.cs**  **frmEditMovies.cs**  **frmEditMusic.cs**  **frmFindMedia.cs**  **frmHomeScreen.cs**  **frmOverdueItems.cs**  **frmPrint.cs**  **frmViewBooks.cs**  **frmViewCustomers.cs**  **frmViewEmployees.cs**  **frmViewGames.cs**  **frmViewMovies.cs**  **frmViewMusic.cs**  **bookLookup.cs**  **databaseHandler.cs**  **otherLookup.cs**  **settings.cs**  **SignedRequestHelper.cs**  **TrackLookup.cs** | 100%  100%  100%  100%  100%  100%  100%  100%  100%  100%  100%  100%  100%  100%  100%  100%  100%  100%  100%  100%  100%  100%  100%  100%  100%  100%  100%  100%  100%  100%  100%  100%  100%  100%  100% | 49.558%  100%  60%  80%  80%  60%  60%  60%  75%  100%  75%  75%  75%  0%  0%  0%  0%  0%  0%  0%  80%  90%  100%  75%  75%  75%  75%  75%  60%  0%  65%  0%  10%  0%  5% | 50.441%  0%  40%  20%  20%  40%  40%  40%  25%  0%  25%  25%  25%  100%  100%  100%  100%  100%  100%  100%  20%  10%  0%  25%  25%  25%  25%  25%  40%  100%  35%  100%  90%  100%  95% |
| **Usability Testing** | 100% | 35% | 65% |
| **Presentation** | 100% | 50% | 50% |
| **Cool Feature 1: DataGridView Implementation** | 100% | 80% | 20% |
| **Cool Feature 2: Amazon API Book Lookup** | 100% | 20% | 80% |
| **Cool Feature 3: Amazon API Game Lookup** | 100% | 20% | 80% |
| **Cool Feature 4: Amazon API Movie Lookup** | 100% | 30% | 70% |
| **Cool Feature 5: Amazon API Music Lookup** | 100% | 30% | 70% |
| **Cool Feature 6: Music Preview** | 100% | 85% | 15% |
| **Cool Feature 7: Instant Search** | 100% | 50% | 50% |
| **Cool Feature 8: Email Notifications** | 100% | 30% | 70% |
| **Cool Feature 9: Receipt Printing** | 100% | 100% | 0% |
| **Cool Feature 10: Auto-Updating Login Menu** | 100% | 100% | 0% |
| **Cool Feature 11: Right Click Functionality** | 100% | 5% | 95% |
| **Total Contribution** | 100% | 49.166% | 50.833% |
| **Total hours on project-related tasks** | 165.5 | 81.25 | 84.25 |
| **Letter grade based on contribution** | A | A | A |

Note 1: A count using the Visual Studio Object Browser revealed about **340** methods in our program. Writing a percentage breakdown for each one of these methods would be extremely time consuming and not reveal any valuable information. Instead, we have listed a breakdown for each form or class file in the project, bringing the number down to a much more reasonable **34.**

APPENDIX B: Testing Instructions

| **Evaluation Criteria** | **Testing** | **Developed By** |
| --- | --- | --- |
| **Access Instructions** | | |
| * Identify all passwords and special links required to use the application and all of its features. If access for any section is via databases/text files, please indicate the section and where to get the login from. | When running the program, use the .exe file  found at  **\openLibrary 2.0\openLibrary 2.0\bin\x86\**  **debug\openLibrary 2.0**  If debugging the program, Visual Studio  must be set to debug in x86 mode:  C:\Users\taiiiiiiiiiiiiii\Documents\GitHub\library\Pics for report\21.PNG | n/a |
| **Navigation Structure** | When on the home screen, there are options in  menus as well as through buttons.  On other screens, most options are accessible  through buttons or right-clicking on a record  (in view screens) | Tai Gunter  Ethan Madden |
| **Application design and layout** | The application strives for a consistent interface. Everything that a user might need is accessible either through buttons, menu items, or right-clicking (where applicable) | Tai Gunter  Ethan Madden |
| **Usage of tools and technologies taught in class** | | |
| * Using databases to lookup records. | Refer to **Testing Instruction D** | Tai Gunter  Ethan Madden |
| * Using databases to add records. | **Add > Books**  or  **Add > Movies** or  **Add > Music** or  **Add > Games** | Tai Gunter  Ethan Madden |
| * Using databases to delete records. | **View > Books** or  **View > Movies** or  **View > Music** or  **View > Games**  …then…  Right Click a record > **Delete** | Tai Gunter  Ethan Madden |
| * Using databases to update records. | **View > Books** or  **View > Movies** or  **View > Music** or  **View > Games**  …then…  Right Click a record > **Edit** | Tai Gunter  Ethan Madden |
| * Using text files to read data. | Refer to **Testing Instruction B** | Tai Gunter  Ethan Madden |
| * Using text files to write data. | Refer to **Testing Instruction C** | Tai Gunter  Ethan Madden |
| * Additional features not demonstrated in class | **Cool Feature 1**: DataGridView: Implemented to improve the simplicity of viewing data; using a DataGridView means that the data can be  consistently and easily viewed, unlike a list box,  which can become poorly formatted very easily  when working with strings of variable length. this  was implemented using the documentation of C#;  all code was written from scratch and debugged as  such. It was relatively challenging to figure out in  the beginning but works great for this application  since it loads directly from the database and  requires minimal formatting so it was crucial to  understand. Without it, the application would be  much more cluttered and it would be harder to  read a lot of the data.  **Cool Features 2-5:** Amazon API Lookup: This feature was added because in a library  environment, it would be highly tedious to  manually enter all the information when adding  books, music, movies, and games to the library’s  database. Using the Amazon API lookup, we form  a query based on the item’s ISBN or UPC code  that the user scans. After authenticating and  processing our request, Amazon returns the information requested. This information is then populated in the text boxes, ready to be added to  the database. This was very challenging to  implement and took a significant amount of time.  **Cool Feature 6:** Music Preview: This was added  so that customers could preview the music on an  album before checking it out. This is helpful if a  customer wants to make sure that a specific song  is on an album before checking it out. It was challenging to implement and buggy once we got  it working. Eventually, we worked out all of these bugs. | Tai Gunter  Ethan Madden |
| * **Additional features not demonstrated in class** | **Cool Feature 7:** Instant Search: This feature is  pretty simple; it uses textChanged events to  trigger the searches instead of buttons. This  speeds everything up and makes it easier on the  user because they can find items quicker without  having to manually click buttons.  **Cool Feature 8:** Email Reporting: If a customer  has an overdue item, we’ve integrated a feature to  send them an email about it. We used some code  on the internet for this, but ended up having to  re-write most of the code. It is helpful because in  a real library environment, the manager will want  to let customers know that they have overdue  items. Email is a great way to do this.  **Cool Feature 9:** Receipt Printing: After seeing  several other groups implement receipt printing,  we decided to do the same. We were able to  quickly adapt some code found on the internet to  meet our needs. When an employee ends a transaction, they have the option to print a receipt.  If they choose yes, one will be printed.  **Cool Feature 10:** Auto-Updating Login Menu:  This feature is entirely designed for the corporate  library system. It integrates a time clock into the  application. To use any part of the application,  an employee must be logged in. This is so that all  checkouts can be recorded with an employee ID,  for accountability purposes. However, in order to  be logged in, the employee must also be clocked  in. Being clocked in means that your hours  worked are being logged in the database. When  an employee comes to work in the morning, he  clocks in, and will remain clocked in until clocked out, even if the application gets closed. Now that  he is clocked in, the menu item for **Log in** has  been updated with his name; he can now log in  and use the application. This system prevents unauthorized users from using the application,  and also means that only employees who are “on  the clock” can use the application.  **Cool Feature 11:** Right-Click Functionality:  This feature allows users to right click a record in  the “view” forms and choose to either edit or  delete them. We wanted to make the interface  simple and intuitive. Having separate buttons that  did these things would clutter the interface. Right  clicking is a simple design that all users are  familiar with. Upon choosing “edit” from the right  click menu, a new form appears with text boxes  to edit the information. If “delete” is chosen, the  user is asked to confirm, then the record is deleted | Tai Gunter  Ethan Madden |
| **Student-Client communication** | Please refer to client documentation (Appendix E) | Tai Gunter  Ethan Madden |
| **Project Documentation** | Please refer to time logs (Appendix F) | Tai Gunter  Ethan Madden |

1. Testing instructions for demonstrating uniform application
   * 1. Open the application.
     2. Follow the instructions on following page to clock in and log in to the database.
     3. Scan a customer card or choose Find User to select a user without their card.
     4. Click Check Out
     5. Scan an item or choose Find Item to select an item without having it physically with you.
     6. Click Submit.
     7. Choose another item to check out or click Complete.
     8. Click End Transaction, then click Yes to print a receipt

These is the main feature and the one that will probably be the most used throughout our application. Of course, there are many other features, many of which you’ll find in the upcoming “Cool Features” section. As you use these features, you’ll see a lot more of the work we put into the application. Features not covered include checking in items and renewing items; this can simply be done after scanning a customer ID and clicking the appropriate button. To check in, click Check In and scan an item, then click Submit, and Complete. To renew, click on an item in the list box that is already checked out, then click renew.

1. Testing instructions for demonstrating reading from text files and exception handling.
   * 1. Open the application.
     2. Follow the instructions on the following page to clock in and log in to the database
     3. Select File
     4. Select Administrative
     5. Select API Keys
     6. On this screen, the API keys used throughout the application are read from a text file and loaded into the text boxes. If desired, these could be changed. The idea is that an actual library could make their own Amazon Associates account and use their own keys. For the purposes of testing this application, leave the keys as they are. Changing them will mean that none of the features that rely on Amazon will work.
     7. In addition to this screen, the API keys are read from file every time a request to Amazon is made. There are numerous examples of this throughout the program; including adding media and music preview.
2. Testing instructions for demonstrating writing to text files and exception handling.
   * 1. Similar to Testing Instructions: Part B, if the keys were changed on the API keys form, they would be written to the file and used throughout the rest of the program. Again, it is not recommended that you change these keys unless you have an Amazon Associates account and have been given a set of access keys to use the Product Advertising API.
3. Testing instructions for demonstrating database usage for lookup and exception handling.
   * 1. Open the application
     2. Follow the instructions on the following page to clock in and log in to the database.
     3. Select View
     4. Select any of the options presented in the list
     5. The following DataGridView is populated using a SELECT statement that references the Access Database.
4. Testing instructions for demonstrating database usage for add and exception handling.
   * 1. Follow the instructions for Cool Feature 2, 3, 4, or 5
     2. After the text boxes have been populated, click Add to add them to the database using an insert statement.
5. Testing instructions for demonstrating database usage for delete and exception handling.
   * 1. Follow instructions for Cool Feature 11
6. Testing instructions for demonstrating database usage for update and exception handling.
   * 1. Follow instructions for Cool Feature 11
7. Testing instructions for demonstrating collection of user information with appropriate validation.
   * 1. Open the application.
     2. Follow the instructions on the following page to clock in and log in to the database.
     3. Select Add
     4. Select Employee or Customer
     5. Fill out the information and add the user to the database.
     6. Close the form.

* Instructions to clock in and log in to the database (Note: For testing, we’ve left Ethan Madden clocked in as an employee. You can simply go to **File > Login > Ethan Madden** and not worry about these steps.)
  + Open the application (Figure 1)
  + Open the database associated with the application
    - Select **File**
    - Select **Open…** (Figure 2)
    - Choose the database **library.mdb**
  + Clock in a user to the system
    - Select **Time Clock**
      * Select **Clock In** (Figure 3)
    - Scan or manually enter the Employee ID number of an Employee in the system (Figure 4)
      * Note: The Employee ID entered must be an Employee within the database. A card for Tai Gunter (not currently clocked in) has been provided.
  + Log in an Employee
    - Select **File** 
      * Select **Log In**
        + Select the name of an Employee currently clocked in (Figure 5)

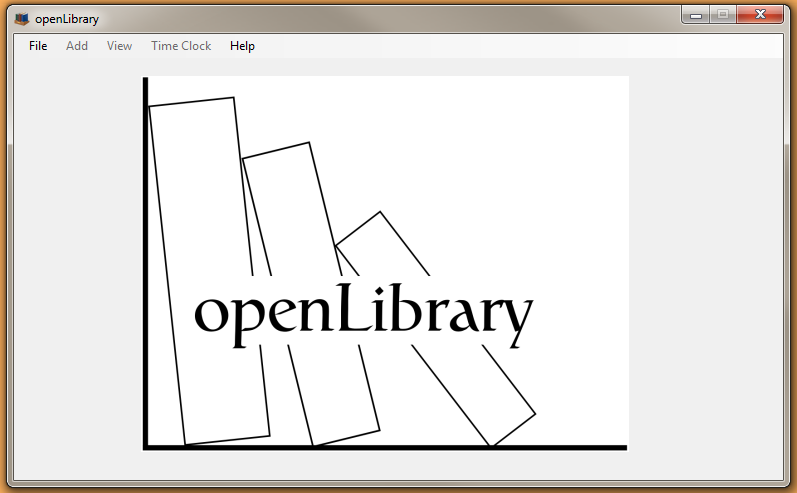
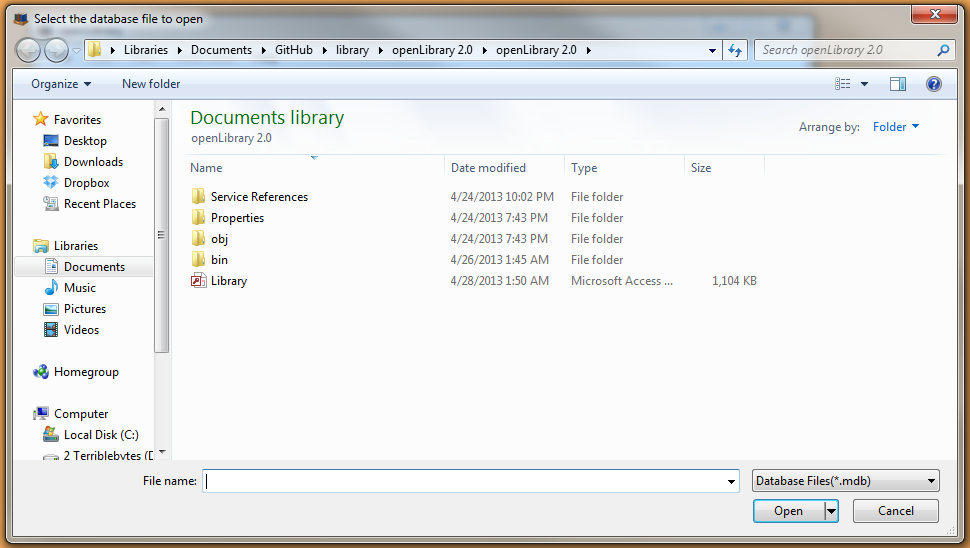
 

Figure 1 Figure 2

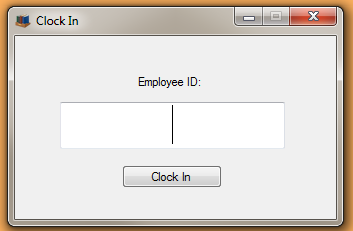
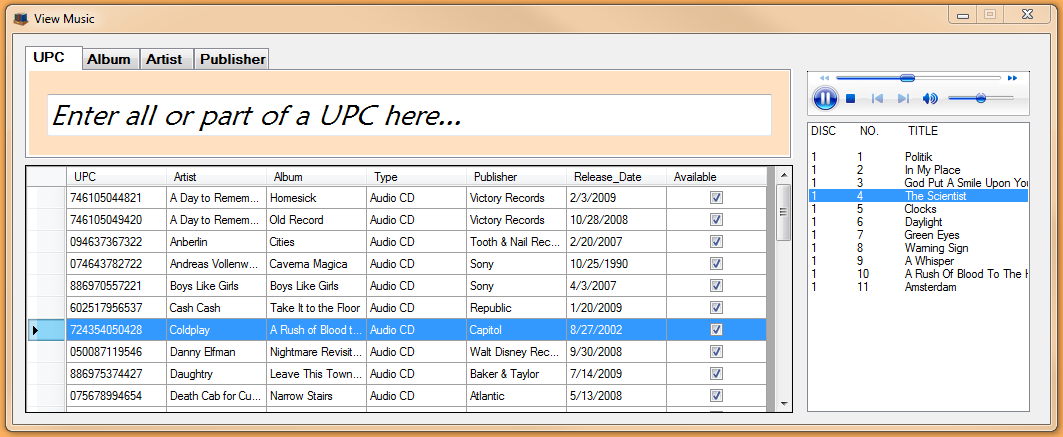
 

Figure 3 Figure 4



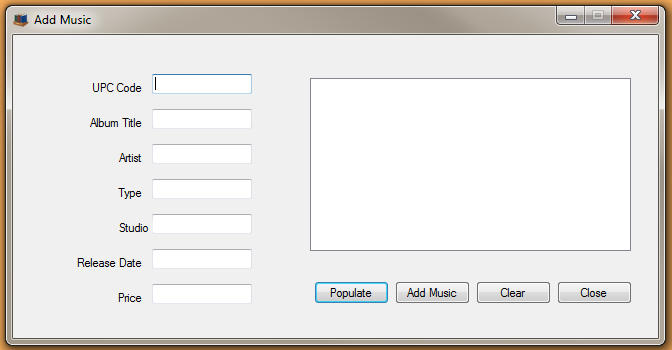
Figure 5

* ***“Cool Feature” # 1:*** DataGridView: Used to view the database’s data in a simple, organized manner
  + Open the application
  + Follow the instructions to clock in and log in to the database
  + Select **View**
    - Select any of the following
      * **Books**
      * **Movies**
      * **Music**
      * **Games**
      * **Customers**
      * **Employees**
  + Upon making a selection, a form will appear with the relevant data loaded into a DataGridView. (Figure 6)
  + Close the form



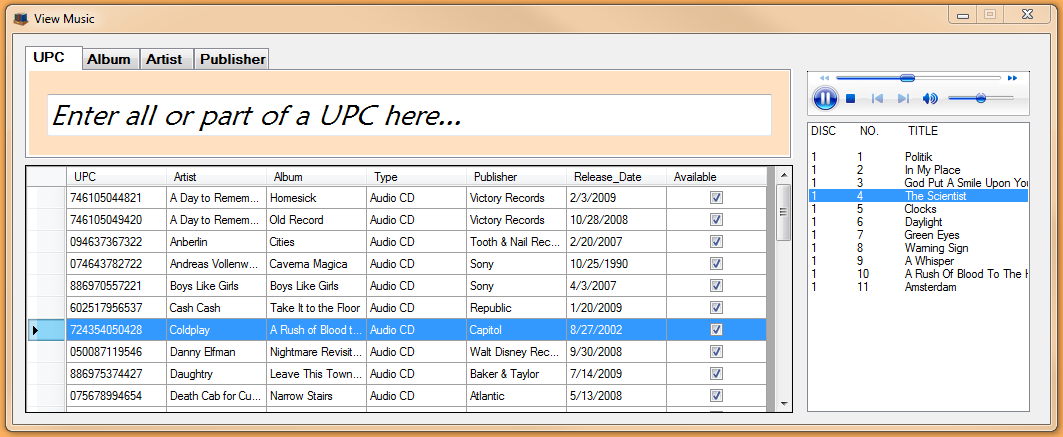
*Figure 6*

* ***“Cool Feature” # 2:*** Amazon API Lookup for books: To automatically populate fields to reduce the amount of work required of the user
  + Open the application
  + Follow the instructions to clock in and log in to the database
  + Select **Add**
    - Select **Book**
  + In the default field “IBSN”, scan or manually enter an ISBN for virtually any book
  + The rest of the fields will auto-populate using data from the Amazon database
  + Close the form
* ***“Cool Feature” # 3:*** Amazon API Lookup for Games:
  + Open the application
  + Follow the instructions to clock in and log in to the database
  + Select **Add**
    - Select **Game**
  + In the default field “UPC”, scan or manually enter a UPC code for virtually any video game (XBOX 360, PS3, Wii, Nintendo DS, PC, etc.)
  + The rest of the fields will auto-populate using data from the Amazon database
  + Close the form
* ***“Cool Feature” # 4:*** Amazon API Lookup for Movies:
  + Open the application
  + Follow the instructions to clock in and log into the database
  + Select **Add**
    - Select **Movie**
  + In the default field “UPC” scan or manually enter a UPC code for virtually any movie
  + The rest of the fields will auto-populate using data from the Amazon database
  + In addition to the fields normally populated, the main actors will also be populated into the list box on the form
  + Close the form
* ***“Cool Feature” # 5:*** Amazon API lookup for Music:
  + Open the application
  + Follow instructions to clock in and log in to the database
  + Select **Add**
    - Select **Music**
  + In the default field “UPC”, scan or manually enter a UPC code for virtually any audio CD. (Figure 7)
  + The rest of the fields will auto-populate using data from the Amazon database
  + In addition to the fields normally populated, the tracks on the audio CD will be auto-populated into the list box on the form
    - The tracks are separated by disc number and track number
  + Close the form



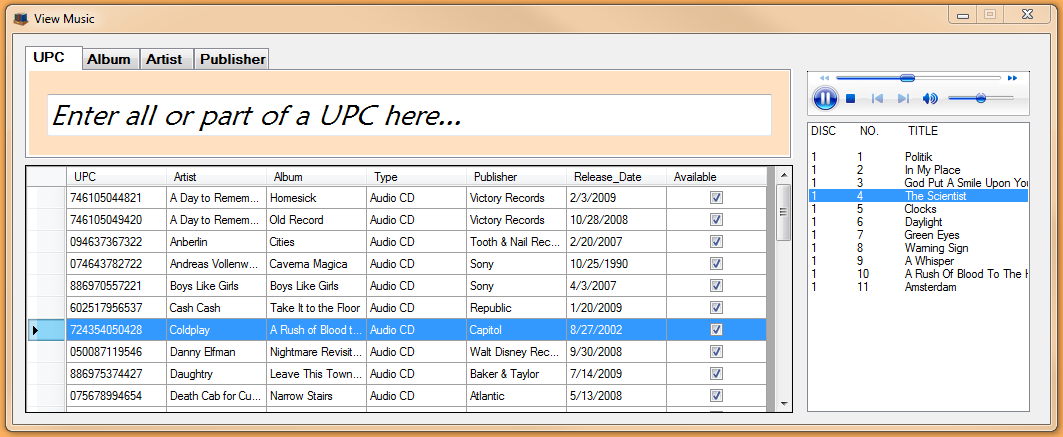
*Figure 7*

* ***“Cool Feature” # 6:*** Music Preview
  + Open the application
  + Follow instructions to clock in and log in to the database
  + Select **View**
    - Select **Music**
  + Select any audio CD from the DataGridView
  + The audio tracks on the corresponding CD will auto-populate into the listbox
  + Click any audio track within the listbox (Figure 8)
  + An audio preview will be downloaded and begin to play
    - Choosing another audio track will stop the current track and begin a new preview
  + Close the form



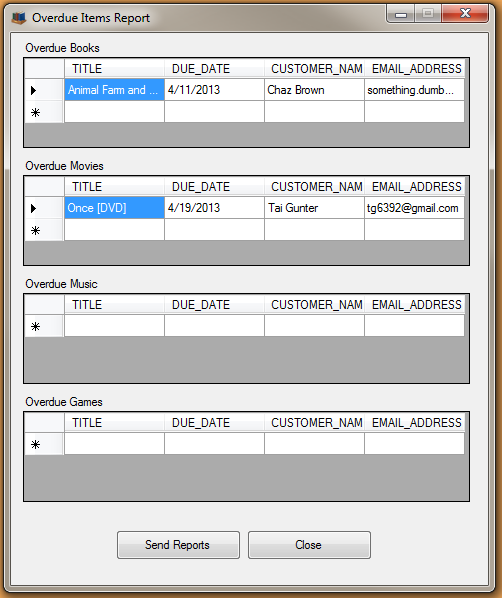
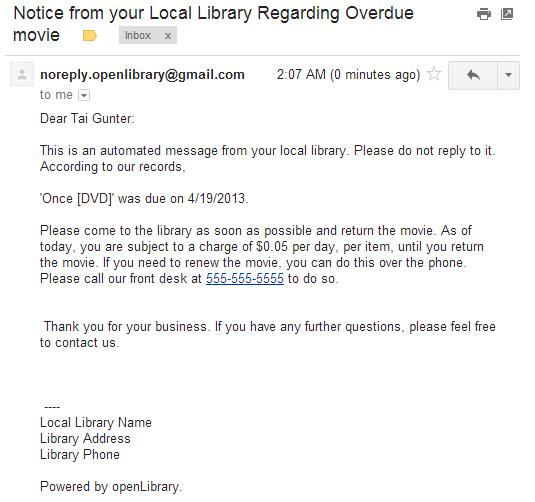
*Figure 8*

* ***“Cool Feature” # 7:*** Instant Search
  + Open the application
  + Follow instructions to clock in and log in to the database
  + Select **View**
    - Select any one of the following
      * **Books**
      * **Movies**
      * **Music**
      * **Games**
  + In the search box at the top of the screen, choose a tab to search by
  + As the search query is typed, the DataGridView will automatically be updated (Figure 9)
  + Changing the tab or otherwise leaving the search box will reset the search
  + Close the form



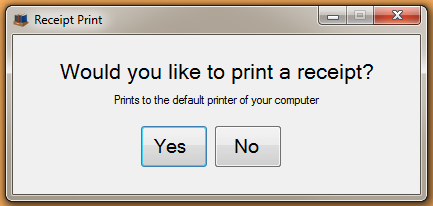
*Figure 9*

* ***“Cool Feature” # 8:*** Email Notifications
  + Open the application
  + Follow instructions to clock in and log in to the database
  + Select **File**
    - Select **Administrative**
      * Select **Overdue Report**
  + All overdue items within the database will be shown in the respective DataGridViews of the form. (Figure 10)
  + Click **Send Report**
  + This will send an email to all customers with an overdue book, notifying them of the issue.
    - “Alka Harriger” is a customer in the database with an overdue book. An email has been sent. (Similar to Figure 11)
  + Close the form

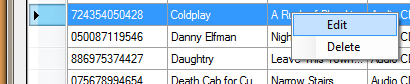
*Figure 10 Figure 11*

* ***“Cool Feature” # 9:*** Receipt Printing
  + Open the application
  + Follow instructions to clock in and log in to the database
  + Scan or manually enter a customer ID number in the text box on the form (A customer card for Alka Harriger has been provided)
  + Click **End Transaction**
    - When asked if you’d like to print a receipt, click **Yes.** A receipt will be printed to the default printer.
    - Note: The lack of “print preview”, “page setup” or “choose printer” dialogs is that in a true library environment, it is extremely unlikely that a user would choose a printer other than the default printer already selected, because the computer being used would most likely be a dedicated checkout/check in machine. Furthermore, print preview and page setup functionalities are unnecessary because unlike a word processing document, there (intentionally) isn’t any way for the user to edit what gets printed. A print preview option would be worthless because the user can’t change what’s being printed. Furthermore, any relevant settings in “page setup” have already been changed.



*Figure 12*

* ***“Cool Feature” # 10:*** Auto-updating login menu
  + Open the application.
  + Select **File**
    - Select **Open**
      * Choose **library.mdb**
  + Select **Time Clock**
    - Select **Clock in**
      * Scan or manually enter an Employee ID
  + Select **File**
    - Select **Log in**
      * The user that was just clocked in is now available to log in to the system.
      * If you return to the **Clock in** screen and clock in another employee, their name will show up in the log in screen.
* . ***“Cool Feature” # 11:*** Right Click functionality
  + Open the application.
  + Follow instructions to clock in and log in to the database
  + Select **View**
    - Select any one of the following
      * **Books**
      * **Movies**
      * **Music**
      * **Games**
  + Right click on any record in the DataGridView
    - This brings up a menu to either **Edit** or **Delete** records. (Figure 12)
  + Close the form

**

*Figure 13*

APPENDIX C: List of Team files (all non-graphic files)

Note: This does not include prototype programs that were not used in the final submission. These programs were used to test features; most of their code was implemented into the final solution.

├── booklibrary.ico

├── openLibrary 2.0

│   ├── AccDB.Designer.cs

│   ├── AccDB.xsc

│   ├── AccDB.xsd

│   ├── AccDB.xss

│   ├── ACCESSDB.Designer.cs

│   ├── ACCESSDB.xsc

│   ├── ACCESSDB.xsd

│   ├── ACCESSDB.xss

│   ├── App.config

│   ├── bin

│   │   ├── Debug

│   │   │   ├── AxInterop.WMPLib.dll

│   │   │   ├── Interop.WMPLib.dll

│   │   │   ├── Library.mdb

│   │   │   ├── openLibrary 2.0.exe

│   │   │   ├── openLibrary 2.0.exe.config

│   │   │   ├── openLibrary 2.0.pdb

│   │   │   ├── openLibrary 2.0.vshost.exe

│   │   │   ├── openLibrary 2.0.vshost.exe.config

│   │   │   ├── openLibrary 2.0.vshost.exe.manifest

│   │   │   ├── settings.txt

│   │   │   ├── temp2MP3.mp3

│   │   │   └── tempMP3.mp3

│   │   └── x86

│   │   ├── Debug

│   │   │   ├── AlbumArt\_{7DB5EBCC-60C1-4F74-8E8F-736292FB0223}\_Large.jpg

│   │   │   ├── AlbumArt\_{7DB5EBCC-60C1-4F74-8E8F-736292FB0223}\_Small.jpg

│   │   │   ├── AlbumArtSmall.jpg

│   │   │   ├── AxInterop.WMPLib.dll

│   │   │   ├── desktop.ini

│   │   │   ├── Folder.jpg

│   │   │   ├── Interop.WMPLib.dll

│   │   │   ├── Library.mdb

│   │   │   ├── openLibrary 2.0.exe

│   │   │   ├── openLibrary 2.0.exe.config

│   │   │   ├── openLibrary 2.0.pdb

│   │   │   ├── openLibrary 2.0.vshost.exe

│   │   │   ├── openLibrary 2.0.vshost.exe.config

│   │   │   ├── openLibrary 2.0.vshost.exe.manifest

│   │   │   ├── settings.txt

│   │   │   ├── temp2MP3.mp3

│   │   │   └── tempMP3.mp3

│   │   └── DebugtempMP3

│   ├── bookLookup.cs

│   ├── curved\_arrow.png

│   ├── curved\_arrow.svg

│   ├── databaseHandler.cs

│   ├── frmAbout.cs

│   ├── frmAbout.Designer.cs

│   ├── frmAbout.resx

│   ├── frmAddBook.cs

│   ├── frmAddBook.Designer.cs

│   ├── frmAddBook.resx

│   ├── frmAddCustomer.cs

│   ├── frmAddCustomer.Designer.cs

│   ├── frmAddCustomer.resx

│   ├── frmAddEmployee.cs

│   ├── frmAddEmployee.Designer.cs

│   ├── frmAddEmployee.resx

│   ├── frmAddGame.cs

│   ├── frmAddGame.Designer.cs

│   ├── frmAddGame.resx

│   ├── frmAddMovie.cs

│   ├── frmAddMovie.Designer.cs

│   ├── frmAddMovie.resx

│   ├── frmAddMusic.cs

│   ├── frmAddMusic.Designer.cs

│   ├── frmAddMusic.resx

│   ├── frmAmazonKeys.cs

│   ├── frmAmazonKeys.Designer.cs

│   ├── frmAmazonKeys.resx

│   ├── frmBugReport.cs

│   ├── frmBugReport.Designer.cs

│   ├── frmBugReport.resx

│   ├── frmClockIn.cs

│   ├── frmClockIn.Designer.cs

│   ├── frmClockIn.resx

│   ├── frmClockOut.cs

│   ├── frmClockOut.Designer.cs

│   ├── frmClockOut.resx

│   ├── frmCurrentlyClocked.cs

│   ├── frmCurrentlyClocked.Designer.cs

│   ├── frmCurrentlyClocked.resx

│   ├── frmEditBook.cs

│   ├── frmEditBook.Designer.cs

│   ├── frmEditBook.resx

│   ├── frmEditCustomers.cs

│   ├── frmEditCustomers.Designer.cs

│   ├── frmEditCustomers.resx

│   ├── frmEditEmployee.cs

│   ├── frmEditEmployee.Designer.cs

│   ├── frmEditEmployee.resx

│   ├── frmEditGames.cs

│   ├── frmEditGames.Designer.cs

│   ├── frmEditGames.resx

│   ├── frmEditMovies.cs

│   ├── frmEditMovies.Designer.cs

│   ├── frmEditMovies.resx

│   ├── frmEditMusic.cs

│   ├── frmEditMusic.Designer.cs

│   ├── frmEditMusic.resx

│   ├── frmFindMedia.cs

│   ├── frmFindMedia.Designer.cs

│   ├── frmFindMedia.resx

│   ├── frmHelp.cs

│   ├── frmHelp.Designer.cs

│   ├── frmHelp.resx

│   ├── frmHomeScreen.cs

│   ├── frmHomeScreen.Designer.cs

│   ├── frmHomeScreen.resx

│   ├── frmOverdueItems.cs

│   ├── frmOverdueItems.Designer.cs

│   ├── frmOverdueItems.resx

│   ├── frmPrint.cs

│   ├── frmPrint.designer.cs

│   ├── frmPrint.resx

│   ├── frmViewBooks.cs

│   ├── frmViewBooks.Designer.cs

│   ├── frmViewBooks.resx

│   ├── frmViewCustomers.cs

│   ├── frmViewCustomers.Designer.cs

│   ├── frmViewCustomers.resx

│   ├── frmViewEmployees.cs

│   ├── frmViewEmployees.Designer.cs

│   ├── frmViewEmployees.resx

│   ├── frmViewGames.cs

│   ├── frmViewGames.Designer.cs

│   ├── frmViewGames.resx

│   ├── frmViewMovies.cs

│   ├── frmViewMovies.Designer.cs

│   ├── frmViewMovies.resx

│   ├── frmViewMusic.cs

│   ├── frmViewMusic.Designer.cs

│   ├── frmViewMusic.resx

│   ├── Library.mdb

│   ├── obj

│   │   ├── Debug

│   │   │   ├── AxInterop.WMPLib.dll

│   │   │   ├── DesignTimeResolveAssemblyReferences.cache

│   │   │   ├── DesignTimeResolveAssemblyReferencesInput.cache

│   │   │   ├── Interop.WMPLib.dll

│   │   │   ├── openLibrary 2.0.csproj.FileListAbsolute.txt

│   │   │   ├── openLibrary 2.0.csproj.GenerateResource.Cache

│   │   │   ├── openLibrary 2.0.csprojResolveAssemblyReference.cache

│   │   │   ├── openLibrary 2.0.csproj.ResolveComReference.cache

│   │   │   ├── openLibrary 2.0.exe

│   │   │   ├── openLibrary\_2.\_0.frmAbout.resources

│   │   │   ├── openLibrary\_2.\_0.frmAddBook.resources

│   │   │   ├── openLibrary\_2.\_0.frmAddCustomer.resources

│   │   │   ├── openLibrary\_2.\_0.frmAddEmployee.resources

│   │   │   ├── openLibrary\_2.\_0.frmAddGame.resources

│   │   │   ├── openLibrary\_2.\_0.frmAddMovie.resources

│   │   │   ├── openLibrary\_2.\_0.frmAddMusic.resources

│   │   │   ├── openLibrary\_2.\_0.frmAmazonKeys.resources

│   │   │   ├── openLibrary\_2.\_0.frmBugReport.resources

│   │   │   ├── openLibrary\_2.\_0.frmClockIn.resources

│   │   │   ├── openLibrary\_2.\_0.frmClockOut.resources

│   │   │   ├── openLibrary\_2.\_0.frmCurrentlyClocked.resources

│   │   │   ├── openLibrary\_2.\_0.frmEditBook.resources

│   │   │   ├── openLibrary\_2.\_0.frmEditGames.resources

│   │   │   ├── openLibrary\_2.\_0.frmEditMovies.resources

│   │   │   ├── openLibrary\_2.\_0.frmEditMusic.resources

│   │   │   ├── openLibrary\_2.\_0.frmHelp.resources

│   │   │   ├── openLibrary\_2.\_0.frmHomeScreen.resources

│   │   │   ├── openLibrary\_2.\_0.frmOverdueItems.resources

│   │   │   ├── openLibrary\_2.\_0.frmPrint.resources

│   │   │   ├── openLibrary\_2.\_0.frmViewBooks.resources

│   │   │   ├── openLibrary\_2.\_0.frmViewCustomers.resources

│   │   │   ├── openLibrary\_2.\_0.frmViewEmployees.resources

│   │   │   ├── openLibrary\_2.\_0.frmViewGames.resources

│   │   │   ├── openLibrary\_2.\_0.frmViewMovies.resources

│   │   │   ├── openLibrary\_2.\_0.frmViewMusic.resources

│   │   │   ├── openLibrary 2.0.pdb

│   │   │   ├── openLibrary\_2.\_0.Properties.Resources.resources

│   │   │   └── TempPE

│   │   │   ├── AccDB.Designer.cs.dll

│   │   │   └── Properties.Resources.Designer.cs.dll

│   │   └── x86

│   │   └── Debug

│   │   ├── AxInterop.WMPLib.dll

│   │   ├── DesignTimeResolveAssemblyReferences.cache

│   │   ├── DesignTimeResolveAssemblyReferencesInput.cache

│   │   ├── GenerateResource.read.1.tlog

│   │   ├── GenerateResource.write.1.tlog

│   │   ├── Interop.WMPLib.dll

│   │   ├── openLibrary 2.0.csproj.FileListAbsolute.txt

│   │   ├── openLibrary 2.0.csproj.GenerateResource.Cache

│   │   ├── openLibrary 2.0.csprojResolveAssemblyReference.cache

│   │   ├── openLibrary 2.0.csproj.ResolveComReference.cache

│   │   ├── openLibrary 2.0.exe

│   │   ├── openLibrary\_2.\_0.frmAbout.resources

│   │   ├── openLibrary\_2.\_0.frmAddBook.resources

│   │   ├── openLibrary\_2.\_0.frmAddCustomer.resources

│   │   ├── openLibrary\_2.\_0.frmAddEmployee.resources

│   │   ├── openLibrary\_2.\_0.frmAddGame.resources

│   │   ├── openLibrary\_2.\_0.frmAddMovie.resources

│   │   ├── openLibrary\_2.\_0.frmAddMusic.resources

│   │   ├── openLibrary\_2.\_0.frmAmazonKeys.resources

│   │   ├── openLibrary\_2.\_0.frmBugReport.resources

│   │   ├── openLibrary\_2.\_0.frmClockIn.resources

│   │   ├── openLibrary\_2.\_0.frmClockOut.resources

│   │   ├── openLibrary\_2.\_0.frmCurrentlyClocked.resources

│   │   ├── openLibrary\_2.\_0.frmEditBook.resources

│   │   ├── openLibrary\_2.\_0.frmEditCustomers.resources

│   │   ├── openLibrary\_2.\_0.frmEditEmployee.resources

│   │   ├── openLibrary\_2.\_0.frmEditGames.resources

│   │   ├── openLibrary\_2.\_0.frmEditMovies.resources

│   │   ├── openLibrary\_2.\_0.frmEditMusic.resources

│   │   ├── openLibrary\_2.\_0.frmFindMedia.resources

│   │   ├── openLibrary\_2.\_0.frmHelp.resources

│   │   ├── openLibrary\_2.\_0.frmHomeScreen.resources

│   │   ├── openLibrary\_2.\_0.frmOverdueItems.resources

│   │   ├── openLibrary\_2.\_0.frmPrint.resources

│   │   ├── openLibrary\_2.\_0.frmViewBooks.resources

│   │   ├── openLibrary\_2.\_0.frmViewCustomers.resources

│   │   ├── openLibrary\_2.\_0.frmViewEmployees.resources

│   │   ├── openLibrary\_2.\_0.frmViewGames.resources

│   │   ├── openLibrary\_2.\_0.frmViewMovies.resources

│   │   ├── openLibrary\_2.\_0.frmViewMusic.resources

│   │   ├── openLibrary 2.0.pdb

│   │   ├── openLibrary\_2.\_0.Properties.Resources.resources

│   │   └── TempPE

│   │   ├── AccDB.Designer.cs.dll

│   │   └── Properties.Resources.Designer.cs.dll

│   ├── openLibrary 2.0.csproj

│   ├── openLibrary 2.0.csproj.user

│   ├── otherLookup.cs

│   ├── Program.cs

│   ├── Properties

│   │   ├── AssemblyInfo.cs

│   │   ├── Resources.Designer.cs

│   │   ├── Resources.resx

│   │   ├── Settings.Designer.cs

│   │   └── Settings.settings

│   ├── settings.cs

│   ├── SignedRequestHelper.cs

│   └── TrackLookup.cs

├── openLibrary 2.0.sln

├── openLibrary 2.0.suo

└── openLibrary 2.0.v11.suo

12 directories, 230 files

APPENDIX D: Final Usability Tests with results and analyses

We thought that the easiest way to test the usability of openLibrary would be to simply launch the application and let our users do the rest. This would allow us to see what features our users were most interested in and excited about, as well as find the points in our application that were less intuitive for the users. Fortunately our application, though complicated, does not require much in the way of technical skill to operate. Because no one working on this project had contacts in an actual library, all of our usability testing was focused toward users looking to manage a personal library, instead of an enterprise-grade library.

USER 1

This user was our client. During this usability test the client didn't run into issues with the application and was impressed with the stability despite the complexity. He pointed out a few changes that would make the user interface a bit more intuitive, but we were unable to implement the ideas due to both time and complexity.

USER 2

This user was our client's roommate. She was a little bit lost in all of the initial steps required (opening the database, clocking in, and logging in), but thought that the rest of the application was very intuitive and smooth. She really liked the music sample feature, but stumbled across a bug where the program would fail silently if Amazon failed to find a preview for the selected track.

USER 3

This user had a particularly large personal library consisting primarily of CDs, including many CDs by smaller unsigned bands. He was extremely impressed with how many CDs were immediately recognized when they were scanned in and the ability of the sample player to play a short clip from an extremely wide variety of artists and albums. He also pointed out a few buttons that didn't seem to do anything when clicked. These buttons have since been removed.

USER 4

Our fourth user works with the client. He thoroughly enjoyed the application, going so far as to say "Everything was tight." He had no further comments, but made it clear that he was thoroughly impressed.

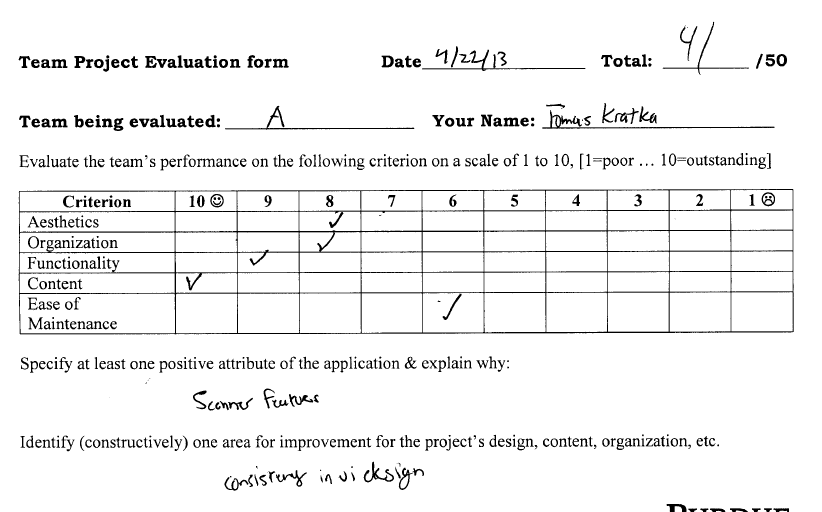
USER 5

Our final user tested the application fully intent to break it. It was surprised to discover that our inputs were filtered, preventing him from executing arbitrary SQL code on our database. He tried to force our application to throw an unhandled error, but was unable to do so by checking out books that were already out, checking in books that were already in, and deleting clocked in employees.

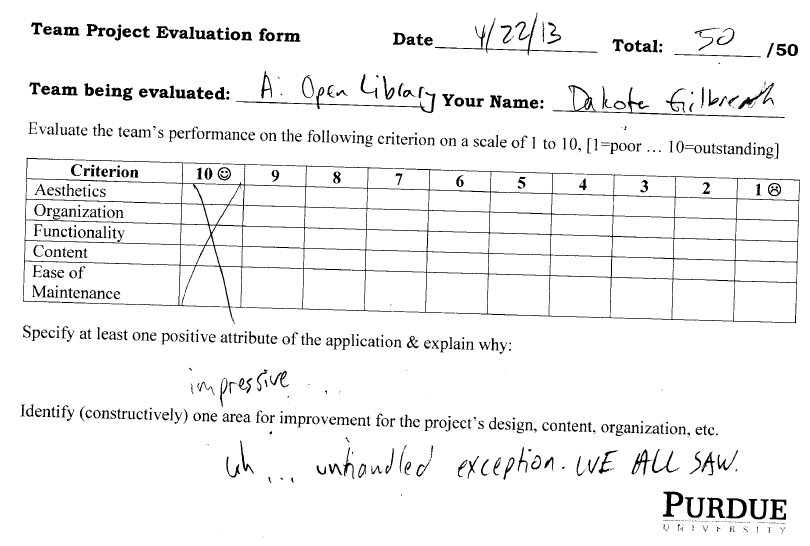
APPENDIX E: Client Evaluation/Feedback

APPENDIX F: Peer Evaluations / Feedback

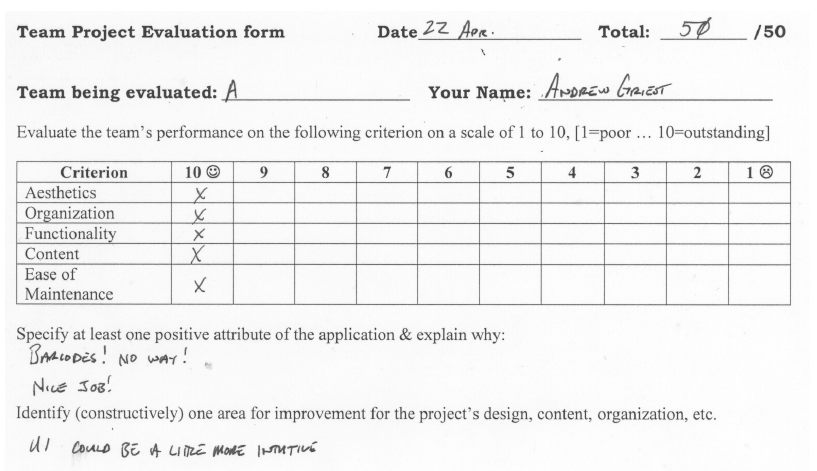
We gained valuable feedback from the peer evaluations. Two of the evaluations (Figure 14, 16) thought that we could have used a better UI design. After the presentation, we cleaned up the interface and made sure things were consistent across all the windows. The evaluation in figure 15 called us out for our unhandled exception that popped up during the presentation. This was caused by overloading the download services by requesting too many mp3’s before they could all be processed. We fixed this error through some careful debugging. Finally, figure 17 shows an evaluation that mentioned feature functionality. Although we didn’t show any unfinished features, there were some hidden in our program, and we got those working before turning in the final project. Getting this feedback from our peers was definitely a help to our presentation. Overall, our feedback scores were very good: we received 2279 out of the total possible of 2300 points, a percentage of 99.1%.

**

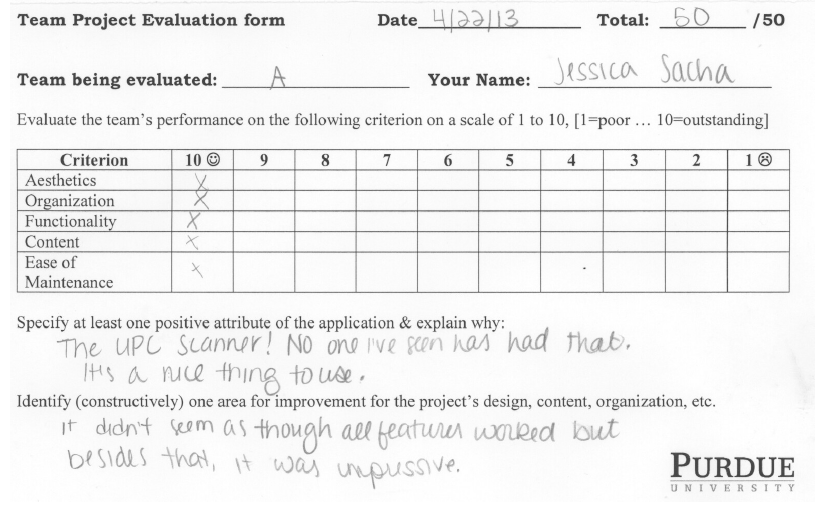
*Figure 14*

**

*Figure 15*

**

*Figure 16*

**

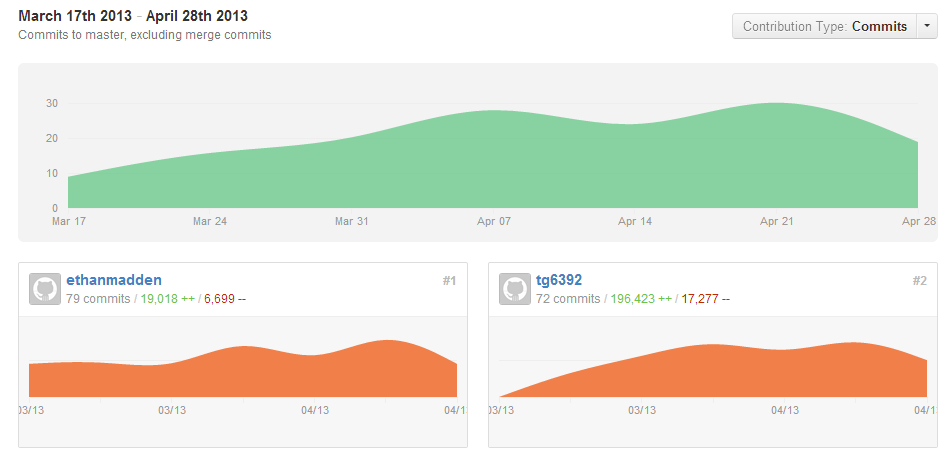
*Figure 17*

APPENDIX G: Time Logs

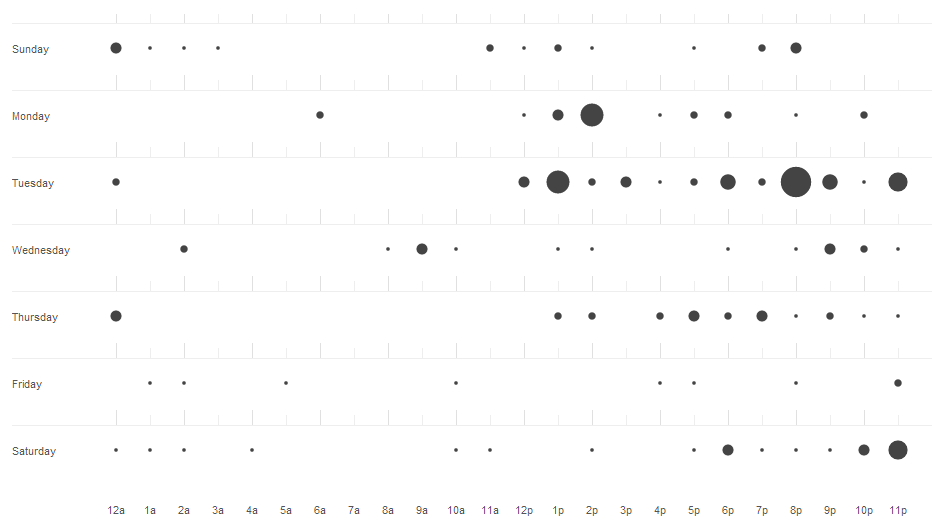
|  |  |  |  |
| --- | --- | --- | --- |
| **USER** | **DATE** | **DESCRIPTION** | **HOURS** |
|  |  |  |  |
| Ethan Madden | 3-18 | Created Github account, repository, and working directory | 1 |
| Ethan Madden | 3-19 | Created database model in Oracle SQL developer | 3 |
| Ethan Madden | 3-21 | Created Access database  Created **library.mdb** | 2 |
| Ethan Madden | 3-21 | Created relationships in Access Database | 1 |
| Ethan Madden | 3-26 | Fixed issues in BOOK table of database  Began working on API pull for books | 2 |
| Ethan Madden | 3-26 | Continued work on database and Amazon API pulls | 1 |
| Tai Gunter | 3-26 | Added C# project to working directory: openLibrary 1.0  Created **frmMain.cs** | 1 |
| Tai Gunter | 3-26 | Added database connections, began work on adding books  Created **databaseHandler.cs** | 2 |
| Ethan Madden | 3-26 | Amazon API successfully pulls data about books | 4 |
| Tai Gunter | 3-26 | Began working on adding books to the database  Created **frmAddBook** | 2 |
| Ethan Madden | 3-26 | Simplified and increased efficiency of Amazon API lookup  Created **lookup.cs** | 2 |
| Ethan Madden | 3-27 | Added functionality to pull more things from Amazon API | .5 |
| Ethan Madden | 3-27 | Added error handling if the ISBN was of an incorrect format | .5 |
| Tai Gunter | 4-1 | Created new working directory for the project: openLibrary 2.0 | .25 |
| Tai Gunter | 4-1 | Continued work to add books to the database | 1.5 |
| Tai Gunter | 4-2 | Added handling to fix apostrophes in the textboxes | 1 |
| Ethan Madden | 4-2 | Added functionality to pull CD information from Amazon API  Created **otherLookup.cs** | 3 |
| Tai Gunter | 4-2 | Added functionality to view the database contents from within the app using DataGridViews  Created **frmViewBook.cs** | 3.5 |
| Ethan Madden | 4-2 | Increased functionality of CD lookup through Amazon API | 1.5 |
| Ethan Madden | 4-3 | Added functionality to look up movies and games through Amazon API | 4 |
| Ethan Madden | 4-3 | Improved functionality of movie and game lookup from Amazon API | 2 |
| Tai Gunter | 4-4 | Attempted to fix issues with DataGridView not refreshing when a new book is added to the database | 3 |
| Tai Gunter | 4-9 | Fixed issue where DataGridView was not refreshing after a new book was added. Added functionality to view games, customers, and employees. Added forms to add employees, customers, music, games, and movies.  Created **frmAddCustomer.cs,**  **frmAddEmployee.cs,**  **frmAddGame.cs,**  **frmAddMovie.cs,**  **frmAddMusic.cs,**  **frmViewGames.cs,**  **frmViewCustomers.cs,**  **frmViewEmployees.cs** | 6 |
| Ethan Madden | 4-10 | Moved functionality to add items into a new menu, as per our client’s request | 1 |
| Ethan Madden | 4-10 | Made changes to the database by adding tables for new media and changing the required tables | 1.5 |
| Ethan Madden | 4-10 | Added functionality to add customers and employees to the database | 2 |
| Tai Gunter | 4-11 | Added instant book search | 2.5 |
| Tai Gunter | 4-11 | Added music preview functionality | 3.5 |
| Ethan Madden | 4-12 | Modified Amazon lookup to pull track information for CDs | 3 |
| Ethan Madden | 4-12 | Added track numbers to the tracks being pulled from Amazon | 2 |
| Ethan Madden | 4-13 | Added support for CDs that have multiple discs | .75 |
| Ethan Madden | 4-13 | Added track information to the main project (from prototype project) | 1 |
| Tai Gunter | 4-13 | Added insert statement to add tracks to the database when scanned | 2.5 |
| Ethan Madden | 4-13 | Modified Amazon lookup to pull star actors for movies scanned into the database | 3 |
| Tai Gunter | 4-13 | Working on getting the tracks to show up when an album is clicked in the DataGridView | 2 |
| Ethan Madden | 4-14 | Actors are now added to the database when a movie is scanned and added | 2.5 |
| Ethan Madden | 4-15 | Fixed relationships within the database now that actors and tracks are being added | .5 |
| Tai Gunter | 4-15 | Tracks now successfully show up in a listbox when an album is clicked in the DataGridView | 2 |
| Ethan Madden | 4-15 | Made a new Amazon lookup call to find the ASID of a track when it is clicked in the listbox, will be used with music preview | 2 |
| Tai Gunter | 4-15 | Music preview is now successfully integrated and functional | 3 |
| Tai Gunter | 4-16 | Added functionality to clock in, clock out, log in, log out, and created checkout interface  Created **frmLogin.cs,**  **frmLogout.cs,**  **frmCurrentlyClocked.cs** | 4 |
| Tai Gunter | 4-17 | Added functionality to checkout and clocking in / clocking out. Time clock information is now stored in the database | 2 |
| Tai Gunter | 4-18 | Worked on checkout and storing this information in the database | 1.5 |
| Tai Gunter | 4-19 | Worked on checkout some more; | 1 |
| Tai Gunter | 4-19 | Ability to renew books added | 1.5 |
| Tai Gunter | 4-20 | Checkout is now fully functional | 2 |
| Ethan Madden | 4-20 | Setup created to make email notifications  Created **mail.cs** | 2 |
| Tai Gunter | 4-21 | Integrating email into the main app (from prototype)  Created **frmOverdueItems.cs** | 2 |
| Ethan Madden | 4-21 | Created prototype for settings file, used to store Amazon API key information  Created **settings.txt** | 2 |
| Ethan Madden | 4-21 | Integrated settings file so that the Amazon API lookup uses the information | 1 |
| Tai Gunter | 4-21 | Began to redo checkout to improve functionality and make it more organized | 1 |
| Tai Gunter | 4-21 | Checkout fully redone, writes to new tables in the database, does not record in the media tables anymore | 2 |
| Tai Gunter | 4-22 | Checkout bugs fixed, Check in works (redesigned), Renew works (redesigned), made changes to time clock setup, overdue items report works completely, settings.txt fully integrated with GUI, email notifications fully integrated, added ability to look up a customer if without their ID number, added music search, added game search, fixed bugs in addMovie, actors are now being added correctly to the database  Created **frmAmazonKeys.cs** | 9 |
| Tai Gunter | 4-22 | Created PowerPoint for presentation  Created **openLibrary.pptx** | 1 |
| Ethan Madden | 4-23 | Fixed a bug in music preview where the program would crash if three songs were selected within 30 seconds of each other | 2 |
| Ethan Madden | 4-23 | Added search functionality for movies | 2 |
| Ethan Madden | 4-24 | Added functionality to delete things from the database through DataGridView via right click | 3.5 |
| Ethan Madden | 4-24 | Began work on edit functionality through DataGridView via right click  Created **frmEditBook.cs** | 2 |
| Tai Gunter | 4-24 | Added functionality to print receipts  Created **frmPrint.cs** | 4 |
| Ethan Madden | 4-25 | Completed edit functionality for books | 2 |
| Ethan Madden | 4-25 | Added edit functionality for music  Created **frmEditMusic.cs** | 1 |
| Ethan Madden | 4-25 | Fixed settings file to no longer ask for email address, also will create the file if it does not exist (bugfix) | 1 |
| Ethan Madden | 4-25 | Added edit functionality for movies  Created **frmEditMovies.cs** | 1 |
| Ethan Madden | 4-25 | Added edit functionality for games  Created **frmEditGames.cs** | 1 |
| Tai Gunter | 4-26 | Added “about” page  Added bug reporting feature  Created **frmAbout.cs**  Created **frmBugReport.cs** | 3 |
| Ethan Madden | 4-26 | Added validation to make sure that only employees that are in the database can clock in. | 1 |
| Ethan Madden | 4-26 | Added availability Boolean to media items | 2 |
| Ethan Madden | 4-26 | View media menus now show things in a better order and show if the item is available to be checked out. | 2 |
| Tai Gunter | 4-27 | Work on report | 4 |
| Ethan Madden | 4-27 | Added validation to make sure items must be checked in before they can be checked out | 1 |
| Tai Gunter | 4-27 | Changed available column to mark new items as available when they are added to the database  Small bugfix in viewMusic | 1 |
| Tai Gunter | 4-27 | You can now find an item in the database if you don’t know what the UPC/ISBN is.  Created **frmFindMedia.cs** | 2.5 |
| Tai Gunter | 4-27 | Implemented searching for customers and employees by name or card code | 1.5 |
| Ethan Madden | 4-27 | Fixed bugs with music preview, checkin, and checkout | 2 |
| Ethan Madden | 4-28 | Added Edit Employee Functionality | 2 |
| Tai Gunter | 4-28 | Added icons to all forms, disabled maximize buttons | .5 |
| Ethan Madden | 4-28 | Edit Customer code | 2 |
| Ethan Madden | 4-28 | Increased efficiency of viewMedia forms | 2 |
| Ethan Madden | 4-28 | Added more error handling to music preview | 1 |
| Ethan Madden | 4-28 | Work on report | 2 |
| Tai Gunter | 4-28 | Work on report | 6 |

On the following page, figure 18 shows our GitHub commits over time. It shows the most productive points throughout the six week project for the team as a whole and each contributor individually.

Figure 19 shows the most common time for GitHub commits. The graph shows that the most common time for work was Tuesday (our lab day). Usually, most of the commits happened in the afternoon or evening, with some early morning commits as well.



*Figure 18*



*Figure 19*