Personal Archival Companion (P.A.C)

Project Team: Cody Cothern, Quincy Hinson, Jacob Oleson, Andrew Menezes

Table of Contents

**1. Project Definition (**100 - 200 words**)** – *Group responsibility*

* Why (it is needed)
* What (is the goal of the project)
* How (how will it be achieved)

**2. Project Requirements** – *Group responsibility*

* Functional
* Usability
  + User interface
  + Performance
* System
  + Hardware
  + Software
  + Database
* Security

**3. Project Specification** – *Group responsibility*

* Focus / Domain / Area
* Libraries / Frameworks / Development Environment
* Platform (Mobile, Desktop, Gaming, Etc)
* Genre (Game, Application, etc)

**4. System – Design Perspective** – *Group responsibility*

* Identify subsystems – design point of view
  + Illustrate with class, use-case, UML, sequence ..... diagrams
  + Design choices (Optional)
* Sub-System Communication (Diagram and Description)
  + Controls
  + I/O
  + DataFlow
* Entity Relationship Model (E-R Model)
  + Example - <https://en.wikipedia.org/wiki/Entity%E2%80%93relationship_model>
* Overall operation - System Model
  + Simplified Sub-system to System interaction

**5. System – Analysis Perspective** – *Group responsibility*

* Identify subsystems – analysis point of view
* System (Tables and Description)
  + Data analysis
    - Data dictionary (Table - Name, Data Type, Description)
  + Process models
* Algorithm Analysis
  + Big - O analysis of overall System and Sub-Systems

**6. Project Scrum Report -** *Group Responsibility*

* Product Backlog (Table / Diagram)
* Sprint Backlog (Table / Diagram)
* Burndown Chart

**7. Subsystems**

**7.1 Subsystem 1** – Name 1 - *Individual responsibility*

* Initial design and model
  + Illustrate with class, use-case, UML, sequence ..... diagrams
  + Design choices
* Data dictionary
* If refined (changed over the course of project)
  + Reason for refinement (Pro versus Con)
  + Changes from initial model
  + Refined model analysis
  + Refined design (Diagram and Description)
* Scrum Backlog (Product and Sprint - Link to Section 6)
* Coding
  + Approach (Functional, OOP)
  + Language
* User training
  + Training / User manual (needed for final report)
* Testing

**7.2 Subsystem 2** – Name 2 - *Individual responsibility*

* Initial design and model
  + Illustrate with class, use-case, UML, sequence ..... diagrams
  + Design choices
* Data dictionary
* If refined (changed over the course of project)
  + Reason for refinement (Pro versus Con)
  + Changes from initial model
  + Refined model analysis
  + Refined design (Diagram and Description)
* Scrum Backlog (Product and Sprint - Link to Section 6)
* Coding
  + Approach (Functional, OOP)
  + Language
* User training
  + Training / User manual (needed for final report)
* Testing

**7.3 Subsystem 3** – Name 3 - *Individual responsibility*

* Initial design and model
  + Illustrate with class, use-case, UML, sequence ..... diagrams
  + Design choices
* Data dictionary
* If refined (changed over the course of project)
  + Reason for refinement (Pro versus Con)
  + Changes from initial model
  + Refined model analysis
  + Refined design (Diagram and Description)
* Scrum Backlog (Product and Sprint - Link to Section 6)
* Coding
  + Approach (Functional, OOP)
  + Language
* User training
  + Training / User manual (needed for final report)
* Testing

**7.4 Subsystem 4** – Name 4 - *Individual responsibility*

* Initial design and model
  + Illustrate with class, use-case, UML, sequence ..... diagrams
  + Design choices
* Data dictionary
* If refined (changed over the course of project)
  + Reason for refinement (Pro versus Con)
  + Changes from initial model
  + Refined model analysis
  + Refined design (Diagram and Description)
* Scrum Backlog (Product and Sprint - Link to Section 6)
* Coding
  + Approach (Functional, OOP)
  + Language
* User training
  + Training / User manual (needed for final report)
* Testing

**8. Complete System** – *Group responsibility*

* Final software/hardware product
* Source code and user manual – screenshots as needed - Technical report
  + Github Link
* Evaluation by client and instructor
* Team Member Descriptions

1. Project Definition

People are always on the go; they go to and come from work, travel for business and pleasure. To entertain themselves during these typically boring periods of travel, people tend to read, either physical books or ebooks, or, if their hands are busy, listen to audiobooks or podcasts. A problem arises when one has to many books and files to easily keep track of.

Now it is true that many applications exist to help make handling and organising all of this entertainment easier, but most tend to focus on only one type. Most of these tend to be clunky, slow, not user-friendly, or lack ease of life features. No one wants to have to worry about having dozens of applications to handle a few tasks, especially if those applications are not easy and intuitive to use. Our goal with this project is to take all of these problems, bind them together, and solve them with one easy-to-use application. The method to solving this will be two-fold.

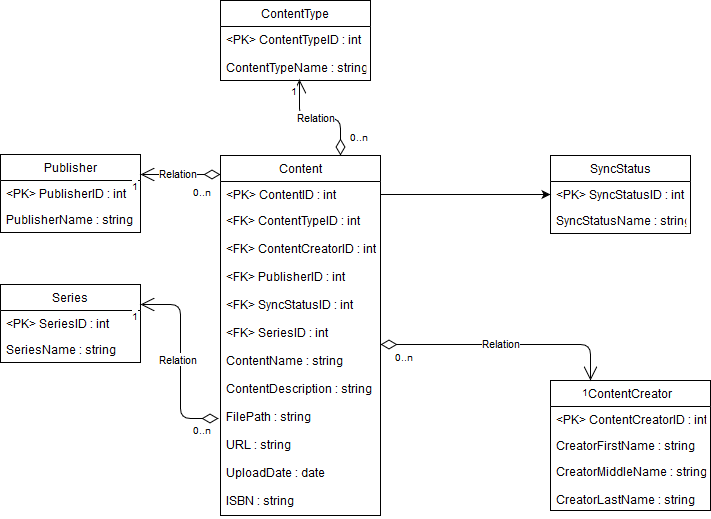
First, we will create a desktop application that acts as the repository and hub for all of the user’s ebook, audiobooks, and podcast files, as well as keeping track of the user’s physical books, both owned and desired. Secondly, we will create a mobile application that will sync with the desktop version and allow the user to access their aforementioned files wherever they happen to be. The mobile app will also include a way to scan the barcode of physical books to automatically look up the books for addition to the user’s personal archive.

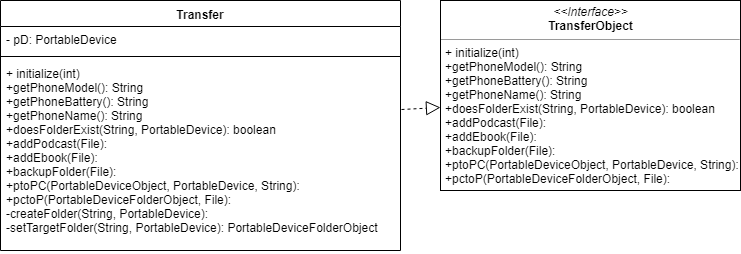
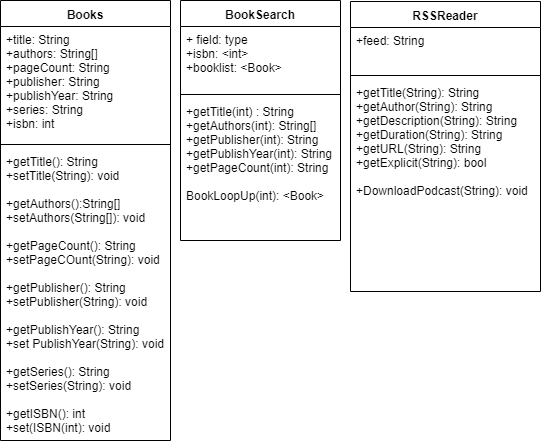
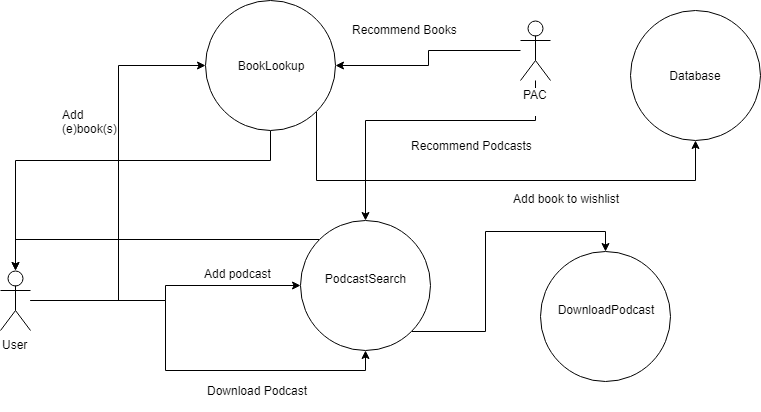
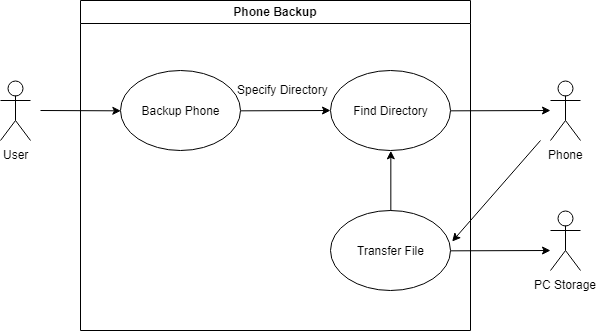
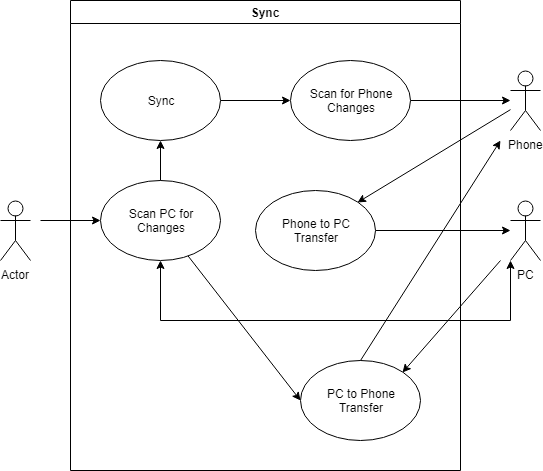
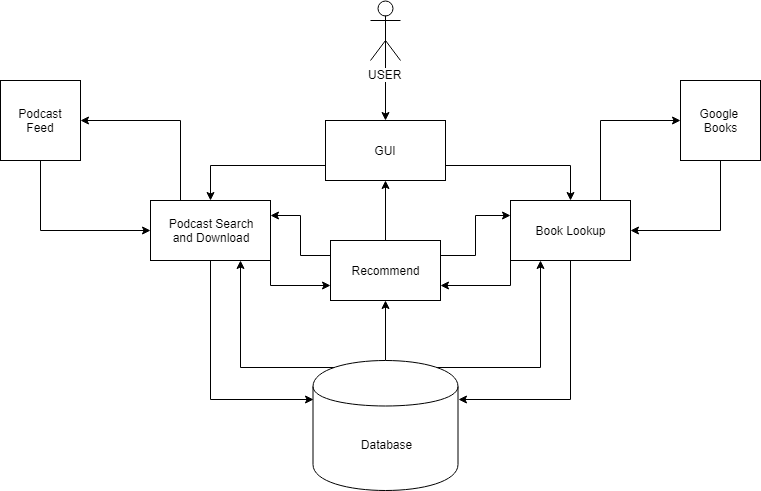
1. Project Requirements
   1. Functionality
      1. Catalog of user’s ebooks, audiobooks and podcasts.
         1. User can manually enter a book’s information to catalog it
            1. User can organise books by various categories

ex.) genre, author, length, etc

* + - 1. User can scan a physical book’s barcode with their mobile phone to add that book to their collection
      2. User can search the internet for a book to catalog it.
    1. User can add/remove books from their wishlist
    2. User can use the app to manage podcasts and organise them based on genre
    3. User will be given different recommendations based on the books/podcasts/etc already in their library
    4. File transfer between desktop version and mobile version.
       1. User will be able to specify which ebook/podcasts will be added to their mobile phone and how many will be maintained at all times.
          1. ex.) keeping five episodes of a podcast on the phone at all times, so when one is watched the next in the series is added.
          2. ex.) when the current ebook in a series is finished, it will be removed from the phone and the next one will be added.
    5. Application will use the newest information from either the desktop app or mobile app to ensure that the information is up to date across devices
  1. Usability
     1. User Interface
        1. Simple interface that is easy to understand
        2. Files containing users’ contents will be appropriately labeled and accessible
        3. Basic settings will be customizable and accessible to the user.
     2. Performance
        1. Application will be responsive and utilize a SQL Database to load and manage their personal libraries
        2. In the event of errors or shut downs the user will be notified with an adequate explanation of why the error occurred
  2. System
     1. Hardware
        1. A PC running the Windows OS
        2. A mobile android device
     2. Software
        1. The Java runtime environment for Windows
  3. Security
     1. Prevent against SQL Injections to protect users’ accounts and their library of media

1. Project Specification
   1. Focus - Users with a large collection of physical books, ebooks, audiobooks, or podcasts
   2. Libraries
      1. JSON
         1. Used to read book information after search
      2. Java ROME
         1. Used to read in rss feed and get podcast information and download podcasts
   3. Platform
      1. Windows PC
      2. Android mobile
   4. Genre
      1. Archival
      2. Organization
      3. Entertainment(Reading and Audio)
2. System - Design Perspective
   1. Identify subsystems – design point of view
      1. Database
         1. Table Diagram



* + 1. GUI
    2. Inter-Device Communication
       1. UML Diagram
          1. 
    3. Data and Information Input
       1. UML Diagram (may be updated later)
          1. 
       2. Use-Case Diagram
          1. 
          2. 
          3. 
  1. Sub-System Communication (Diagram and Description)
     1. I/O
        1. Data will be input by the user which will be fed through to either the podcast or book search algorithms. After the desired information is gathered it will be given to the database for storage. From there it can be displayed as audio/visual output to the user in the form of an audio-player for podcasts and audiobooks or a text display for ebooks. The total collection can also be retrieved from the database and be displayed to the user for accessing or organising.
     2. Controls
        1. Application queries the database for user requested content (Specific books/audiobooks/podcasts to be added or viewed).
        2. Database searches for requested information and passes it to the application to be displayed in the GUI.
     3. Dataflow
        1. 
  2. Entity Relationship Model (E-R Model)
     1. Delayed due to formatting issues

1. System - Analysis Perspective
   1. Identify Subsystems -
      1. Database
         1. We will utilize a SQLExpress database embedded into the application to store important information about the user’s content.
         2. The database is composed of a main content table which contains information relevant to each individual piece of media. It has a many to one relationship with the other tables which include information about a publisher, author, series, and sync status. The reason those tables exist is to reduce the possibility of repeating data.
         3. DB structure may be subject to change and additional attributes may be added or removed from the tables as we progress through the project and optimize certain features.
      2. GUI
      3. Inter-Device Communication
      4. Data and Information Input
         1. Books and eBooks will be searched via their isbn numbers that can be input manually or eventually via barcode scan.
         2. Podcasts will be found using their RSS feed URL and be downloaded and stored as required.
         3. Recommendations will be found using the above search methods, but using data gathered from the database (ex. Genre, author, etc.).
   2. System (Diagram and Description)
      1. Data Analysis
         1. Data Dictionary
            1. Genre - String - What category of content the data is (fantasy, sci-fi, etc.).
            2. ContentCreator - String - Author of the book, or speaker in the podcast.
            3. ISBN - String - unique identifier on every book
            4. PageCount - Integer - How many pages a book contains
            5. Publisher - String - The company that published the work
            6. PublishYear - String - The year that the work was published
            7. Title - String - The title of the work
            8. SubTitle - String - The subtitle of the work, may not apply to all entries
            9. Description - String - A short description of the content of a podcast
            10. Duration - String(Potentially change to Time/Int - the length of a podcast or audiobook
            11. URL - String - link to access or download a podcast
            12. Explicit - Boolean - True/False value of whether or not a podcast contains explicit content
            13. Synced - Boolean - If content from desktop is synced to mobile device.
   3. Algorithm Analysis
      1. BookLookup Algorithm
         1. Assuming adequate internet speed, this algorithm has a complexity of O(n) where n is the number of books the user is searching for.
         2. This algorithm simply reads in a list of isbns and uses the Google Books api to find the desired information and then creates a book object using this information
      2. PodcastSearch and Download Algorithms
         1. Again assuming adequate internet speed, this algorithm should have a complexity of O(n).
         2. Downloading depends on how many podcasts must be downloaded along with the size of each file and the internet speed available.
      3. Phone Backup algorithm
         1. To backup files on a phone, an array list of the file/folder locations that must be backed up must be passed to the backup method. If no array is being passed, then the entire phone will be backed up. In the best case scenario only one file is specified, and it's on the root of the phone making O(n).To find files and folders with the library im using folders are looped through to find files. Unfortunately most normal people will have several folders with folders in them. This means that for most cases the speed would be O(n2) since it loops to find specific files which themselves could be in a folder within a folder.
      4. SQL Algorithms
         1. Inserting new entries into the database is O(1).
         2. Selecting entries from the database takes approximately O(n) with n being the number of entries in a table to sort through.
2. Project Scrum Report
3. Subsystems

\*emphasize mobile and desktop communication

\*more functionality

\*emphasize organising and categorising media

\*work on recommendations based on current library of books

\*potentially add recommendations to wishlist