CS3354 Software Engineering
Final Project Deliverable 2

# Group 9
# Bookkeeper

*Group 9 Members:*
Luca Dejesu,  Miranda Flemming, Shreya Gazawada, Andy Shao,
Vaibhav Sharma, Danh Vo, Jonathan Wachholz

1. [5 POINTS] Well described delegation of tasks, i.e. who did what in the project. Now that your project is complete, you are required to submit the delegation of tasks from beginning of the project until the end. Please make sure to fairly distribute tasks in the team and remember that in the end of the semester, each member of a team will receive the same grade. See grading policy below for more detail. If no/poor contribution by a member, please specify clearly so that we can grade each student fairly.

Deliverable 1

1. Addressing Feedback (Vaibhav)
2. Software Process ModelSetting Up GitHub Repository (Danh, Shreya, Miranda)
    a. adding all collaborators (Danh)
    b. creating README.md (Shreya)
    c. creating a project scope file (Miranda)
3. Delegation of Tasks (Everyone)
4. Software Process model (Luca)
5. Software functional requirements (Vaibhav, Miranda)
6. A Use Case Diagram. (Jonathan)
7. Sequence Diagrams
    a. Search books by text query (Miranda)
    b. View Book (Shreya)
    c. Managing Books
        i. Load books from system into database (Luca)
        ii. Delete books (Luca)
    d. Manage Categories
        i. add (Shreya)
        ii. delete (Shreya)
8. Class Diagram. (Danh, Andy)
9. Choosing and Applying one appropriate Architectural Design for our project. (Vaibhav, Shreya, Andy)

Deliverable 2

4.1 Project Scheduling (Shreya)
4.2 Cost, Effort and Pricing Estimation (Luca)
4.3 Estimated cost of hardware products (such as servers, etc.) (Vaibhav)
4.4 Estimated cost of software products (Miranda)
4.5 Estimated cost of personnel (Andy)
5. Unit Test (Johnathan, Danh)
6. Comparison of your work with similar designs (Miranda)
7. Conclusion (Vaibhav)
8. References  (everyone)
9. Presentation Slides (everyone - each person made the slides for their corresponding part in the report)
11. GitHub (Danh)

2. [5 POINTS] Everything required and already submitted in Final Project Deliverable

    1. Please specify this part as "Project Deliverable 1 content".

**Included in the zip file**

IMPORTANT NOTE: The following items will all need to be calculated / worked on based on the project you are designing. As an example, if a team of 6 students in CS3354 class is working on the development of a hospital information system, this group will prepare the project scheduling, cost, effort and pricing estimation calculations based on the hospital information system design, NOT based on their 6 people team. Think of the analogy to the "Inception" movie: What you will be working on is the dream in a dream, i.e. the dream in the second level, NOT in the first level.

3. [5 POINTS] Choose only one of the following two options and specify clearly which is your choice. This will help us post a live presentation schedule. Please understand that we wouldn't know which group will present live, and which group will prerecord instead before the due date of final project deliverable2. So, thank you very much for your patience. We will post a presentation schedule once we have each group's submission.

- Option 1: Present live during scheduled class time via BBC (Blackboard collaborate). Each member of the group has to be present and participate in this option. No need to prerecord your presentations if you choose option 1 as your live presentations will be recorded. Still, you should include your (non-recorded) presentation slides into your project deliverable2 submission bundle.
- **Option 2: Prerecord your captioned presentations. Each member of the group has to talk and participate in this option as well. Save your captioned recording at a URL. Then provide this URL (where you host your prerecorded presentation) exactly here, inside your Final Project Deliverable2 report so that we post it for students to access. Remember: Captioned recordings are UTD requirement. Make sure each group member talks in that recorded presentation. DO NOT SEND TO US your presentation recordings, as your captioned pre-recording file will be too large. Only provide its URL here. Still, you should include your (non-recorded) presentation slides into your project deliverable2 submission bundle.**

**We will present using Option 2**

4. [35 POINTS] Project Scheduling, Cost, Effort and Pricing Estimation, Project duration and staffing: Include a detailed study of project scheduling, cost and pricing estimation for your project. Please include the following for scheduling and estimation studies:

**Assuming there are 5 people on the team for Question 4**

4.1. [5 POINTS] Project Scheduling. Make an estimation on the schedule of your project. Please provide start date, end date by giving justifications about your estimation. Also provide the details for:

- Whether weekends will be counted in your schedule or not : No
- What is the number of working hours per day for the project: 8 hours **(Shreya)**

## Project Scheduling

Agile methodology will be used to develop this software application. So, the tasks will be divided and completed in one week sprints. 5 people will work on this application in total. There will be three teams: two teams of 2 will handle development and 1 person will work on testing. This will allow for concurrent development, reducing development time. This is only an estimation, while developing the application the team may need to go back and spend more time on a certain view and/or function. Sprint 3 does not have much development. It will test the application more thoroughly and deploy the code. The sprints are focused on building certain features, and then moving on to the next set of features. Weekends will not be counted in the schedule because working 8 hours a day Monday through Friday (inclusive) for a month will be sufficient time to finish the project according to the estimation.

**Start Date: Nov 9**
**End Date: Nov 27**

*Bookkeeper Schedule*

**Sprint 1**

Design Home Screen View : Nov 9 -10
Design Book Management View : Nov 9 -10
Create Home Screen View: Nov 10 -11
Create Book Management View: Nov 10 -11
Develop Loading Books Functionality : Nov 11 - 12
Develop Delete Books Function : Nov 11 -12
Testing Home Screen Functions : Nov 12 - 13

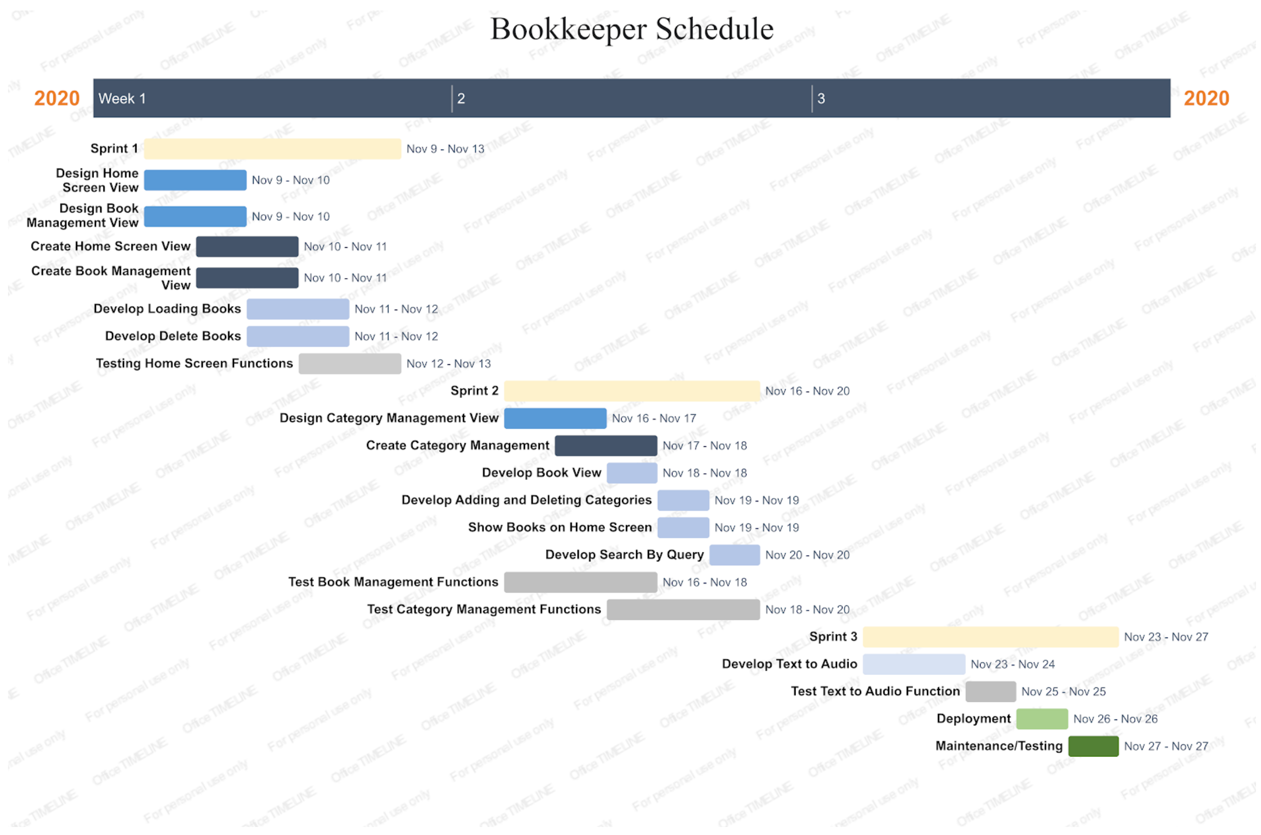
**Sprint 2**

Design Category Management View: Nov 16 -17
Create Category Management: View Nov 17 -18
Develop Book View Functionality: Nov 18
Develop Adding and Deleting Categories Functionality: Nov 19
Show Books on Home Screen: Nov 19
Develop Search By Query Functionality: Nov 20
Test Book Management Functions: Nov 16 -18
Test Category Management Functions: Nov 18 -20

**Sprint 3**

Develop Text to Audio Functionality : Nov 23 - 24
Test Text to Audio Function: Nov 24 - 25
Deployment: Nov 26
Maintenance: **Nov 27 -**

Gantt Chart:



**4.2. [15 POINTS] Cost, Effort and Pricing Estimation. Describe in detail which method you use to calculate the estimated cost and in turn the price for your project. Please choose one of the two alternative cost modeling techniques and apply that only:**
Our group decided that the functional point algorithmic method of cost modeling will suit the bookkeeper application best.

This method will allow for a simpler and more reasonable cost and effort estimation, because our application is not composed of a lot of pre-existing applications. If it were, then application composition would likely be a better model, but it does not.

Some of the parameters we came up with were home screen management, management page, user database, book view interface, and load/delete book functions.

Before diving into the analysis, we should consider some factors affecting cost modeling and whether or not they impact the development of Bookkeeper.

The first external factor we need to consider is how we will organize the book databases and select the ones to interface with. We have elected to use other online databases for book purchasing and downloads, as it will spare us the complexity of developing and maintaining a separate online database of our own.

While this solves our issue of online data management, it presents a new one, and that is choosing the right databases to access and interfacing with those who own them. A lot of books are available on Google, Amazon, and a variety of other sources, so choosing the ones to integrate and how to organize them will impact the total effort.

Another external factor affecting effort that we need to consider is how to delegate the project among our group members. Our group will consist of 5 of us, and we all have relatively similar areas of expertise: we are all third year computer science or software engineers at UT Dallas. So, when it comes to delegating, it could be difficult to have the right person on the right task as we have similar skill sets. This could incur extra time spent on the project if one person is not able to complete their work as effectively.

One final external factor we should consider in regards to effort approximation is how much time we will spend to ensure the program is easy to update and develop going forward. The initial release of the bookkeeper software will not be the final version, and the updates will be fluid as users request new functionality. We have to consider spending extra time to make sure the code is development friendly, as other e-reader applications have needed frequent updates shortly after initial release. One example is the application 'EBook reader' available on the app store. It was released in May of 2011, and within the first three months it required 5 updates due to user reported bugs and similar **[1].**

Now, we can proceed to estimate the effort required for the Bookkeeper application by functional point analysis:

**Step 1**
Derive the function category counts.
As mentioned above, we identified our formal parameters first and foremost. In total, we find eight: *Home screen, book management interface, user library interface, current book view, find book functionality, delete book functionality, user account database, and external database online from which to purchase books.*

Here is how we categorized them by the five functional point categories:

| Category | Count | Examples |
|---|---|---|
| **User Inputs** | 2 | Home screen page, Management Page |
| **User Outputs** | 2 | User library, book view |
| **User Queries** | 2 | Find book, delete book |
| **Data Files** | 2 | Account database, external database |
| **External Interfaces** | 1 | External database |

**Step 2**

We estimated the complexity for each of these five categories, and calculated the gross functional points (GFP).

Here is a brief explanation as to why we decided on the complexities that we did:

For all user-related categories, namely input, output, and queries, we determined these to be of **simple** complexity. The main reason is the fact that the core difficulty in these resides in graphical user interface, and the functionality should be easy to implement in code.
We chose **average** complexity for the remaining two categories, data files and external interfaces, because we have to integrate our application with some pre-existing online book databases. This is the main composition part of the application, and the fact that there must be some decided upon organization makes it a bit more complex than simply programming the functionality.

Here is the calculation for the **GFP, gross function points:**

| | Function Category | Count | Complexity | | | Count x Complexity |
|---|---|---|---|---|---|---|
| | | | Simple | Average | Complex | |
| 1 | Number of user input | 2 | 3 | 4 | 6 | 6 |
| 2 | Number of user output | 2 | 4 | 5 | 7 | 8 |
| 3 | Number of user queries | 2 | 3 | 4 | 6 | 6 |
| 4 | Number of data files and relational tables | 2 | 7 | 10 | 15 | 20 |
| 5 | Number of external interfaces | 1 | 5 | 7 | 10 | 7 |

**Step 3**
Compute the Processing Complexity Adjustment, **PCA:**

Firstly, we responded to each of the 14 complexity questions:

**(1) Does the system require reliable backup and recovery? 4**

Data loss, through not having the books backed up reliably, will reflect negatively on our software, so it is important for Bookkeeper to have reliable backup and solid recovery options.

**(2) Are data communications required? 0**

In the future, communications could be integrated through a review system or similar, but currently that is not in the plan for starting out so it is relatively unimportant.

**(3) Are there distributed processing functions? 0**

There are none that we could identify.

**(4) Is performance critical? 2**

Performance will mainly relate to storage, as user interaction is only occasional. We assess it to be 2 on the scale, as it is important for reliable storage and loading of books, but not heavy on user interaction.

**(5) Will the system run in an existing, heavily utilized operational environment?1**

It mainly will rely on the internal functionality, except for when transferring books from existing online databases. We decided that this is a relatively simple task, so we scored it a 1.

**(6) Does the system require online data entry? 0**

We initially contemplated having an online Bookkeeper specific database, but we figured it was unnecessary as we could interface with existing databases. This is unimportant as it does not require online data entry now.

**(7) Does the online data entry require the input transaction to be built over multiple screens or operations?0**

For loading books from online, we will use our own management interface. Since it will only require this one screen, we rank this as 0.

**(8) Are the master files updated online? 0**

We decided to rely on existing databases rather than managing our own.

**(9) Are the inputs, outputs, files, or inquiries complex? 1**

There are a small number of inputs, outputs, and queries, but they are all simple.

**(10) Is the internal processing complex? 1**

The only slightly complex processing component is organizing and implementing functionality with the online databases. The rest is simple programming that can be language independent.

**(11) Is the code designed to be reusable? 3**

Since we are using the model-view-control design pattern, we need to pay good attention to code reusability. The MVC pattern emphasises it for simple interaction between the 3 main components of model, view, and controller.

**(12) Are conversion and installation included in the design? 3**

We score this as a 3, because we will need to include conversion and installation of the software. However, it is not a 4 or 5 because the conversion and installation should be relatively simple. This is because we are basing the program on reusable components, and the conversion would relate to simple software updates.

**(13) Is the system designed for multiple installations in different organizations? 1**

Bookkeeper should be executable on tablets, phones, and laptops. However, the core difference lies in simple graphical user interface features, which are a relatively simple issue.

**(14) Is the application designed to facilitate change and ease of use by the user? 5**

This is the most important part of the Bookkeeper software. We need it to be user friendly, as there are a lot of e-readers, and user dissatisfaction could ruin the chance of success. It is also planned that the software will evolve primarily based on user response (the reviews will dictate the updates).

**Calculation of PCA:**

**Total GFP: 2*3 + 2*4 +2*3 +2*10+ 1*7 = 47**

**Total PC: 4 + 2 + 1 + 1 + 1 + 3 + 3+ 1 + 5 = 21\**

**PCA = 0.65 + 0.01(sum of all question responses)**

**PCA: 0.65 + 0.01(4+2+1+1+1+3+3+1+5) = 0.65 + 0.01(21)= 0.86**

**Step 4**
Calculate the functional points:

(Functional Points = Gross Functional Points times Processing Complexity Adjustment)
**FP = GFP x PCA = 47 x 0.86 = 40.42**

**Step 5**
Estimate the effort in weeks.

*Effort = Functional Points/ Productivity of the Group.*

First, we need to estimate what sort of productivity our group can achieve per week.
We can estimate it based on a few factors: years experience with programming, programming language knowledge, and domain knowledge.
Being that our group consists of third year computer science or software engineer majors at UT Dallas, we can determine there are likely 2 years of experience programming as well as at least 2 languages learned (UT Dallas specifically teaching Java and C++ the first two years). Lastly, we all have experience

with e-readers based on group discussion. This gives each of us junior level experience with programming, knowledge of two or more languages per person, and decent exposure to the domain.
As a reference, we used the example regarding the shipping software in the slides for chapter 23. That example specifies that each developer on the team produces 60 functional points per week. Given that our team consists of mainly students, we cannot expect that many per week. We can realistically say the team could do about a fourth of that work per week, so we will choose **15 functional points per week.**

**So, effort is E = 40.42/15 = 2.695, we can round up to 3 person-weeks.**

There are five people working on the project, based on our assumption, so this will give us a
**project duration (D) = 3 person-weeks/5 people = 0.6 weeks, rounding up this gives us an estimated duration of 1 week or so.**

4.3. [5 POINTS] Estimated cost of hardware products (such as servers, etc.)

The team must accommodate developing the application for the most popular operating systems on both mobile and desktop devices. For this purpose, Apple Macbook Air devices were chosen for development, as they are able to run both Windows and macOS, and also support development environments for Android and iOS. Apple laptops would cut costs by providing a single device that could be used to develop on multiple platforms, so Android, iPhone, Windows, and macOS development could be conducted without additional computing hardware. Laptops also have integrated webcams, microphones, and good portability, which would allow flexibility in terms of work environment and greater ease in arranging team meetings.

As for mobile development devices, the iPhone SE was chosen, which is currently retailing for $399, since it is cost-efficient, incorporates all of the important features from the iPhone line while only lacking a few advanced features which are irrelevant to the application such as 3D touch and AI acceleration. As for Android, the Pixel 4a was chosen, currently retailing for $349, as a development device since it provides a pure Android platform, unchanged by any manufacturer specific modifications. Unlike with iOS and iPhone, where Apple has full control over the market, major phone manufacturers are able to tweak the implementation of the Android OS, providing a flavor of it customized to their specifications. Thus, in order to ensure compatibility across the widest range of devices, it is important for the team to have access to devices which provide a 'stock' experience, not dependent on any features or software that would not be universally available across Android phones.

Finally, since it is anticipated that the team could largely be working remotely, and to protect against the risk of unanticipated data loss, it was decided that a cloud storage solution was necessary. We chose Google One due to its competitive pricing and reliability. After analyzing the available alternatives, we felt that it offered the best combination of ease of use, reliability, and pricing. For our team of five, Google One offers a 2TB plan at $9.99 monthly which would provide adequate storage space for each developer.

| | |
|---|---|
| $399 * 5 | iPhone SE |
| $929 * 5 | Macbook Air |
| $349 * 5 | Pixel 4a |
| $9.99 * 5 | Google Drive 4TB 1 month |

Total Hardware Cost: **$8444.94**

4.4. [5 POINTS] Estimated cost of software products (such as licensed software, etc.)

**IDE, OS [Miranda]**

The cost of software products for this project will likely be a one time fee $124 and recurring fees of $99 per year and $4 per month. The main costs associated with our project's development will come from licensing and registration to various app stores. Ideally, Bookkeeper will be a cross-platform application so the development team will need to make our product available on multiple application stores like Apple's app store, the google play store, and microsoft app store.

In order to distribute Bookkeeper on the app store and make our software accessible to apple/iOS devices, we will need to enroll in the Apple Developer program. Membership in this program will cost about $99 per year.

To distribute Bookkeeper on the Google Play and Microsoft stores we will have to pay one time fees of $25 and $99 dollars respectively for company membership. Both of these memberships will be necessary to make Bookkeeper accessible to devices that use Android and Windows.

Bookkeeper will be developed using private github repositories, at a cost of $4 per month. The development team will program Bookkeeper using free IDEs such as Eclipse, NetBeans, Atom, Microsoft Visual Studios, etc.

4.5. [5 POINTS] Estimated cost of personnel (number of people to code the end product, training cost after installation) Andy Shao

The cost of personnel is 3-person weeks and we assume that the average salary of our developers is $30 per hour. If we assume that one working week consists of 40 hours, then the cost of personnel is 40 * $30 or $1200. After installation, we assume a total of 10 hours of maintenance per year. The cost of personnel for maintaining the software would be 10 * $30 or $300 per year.

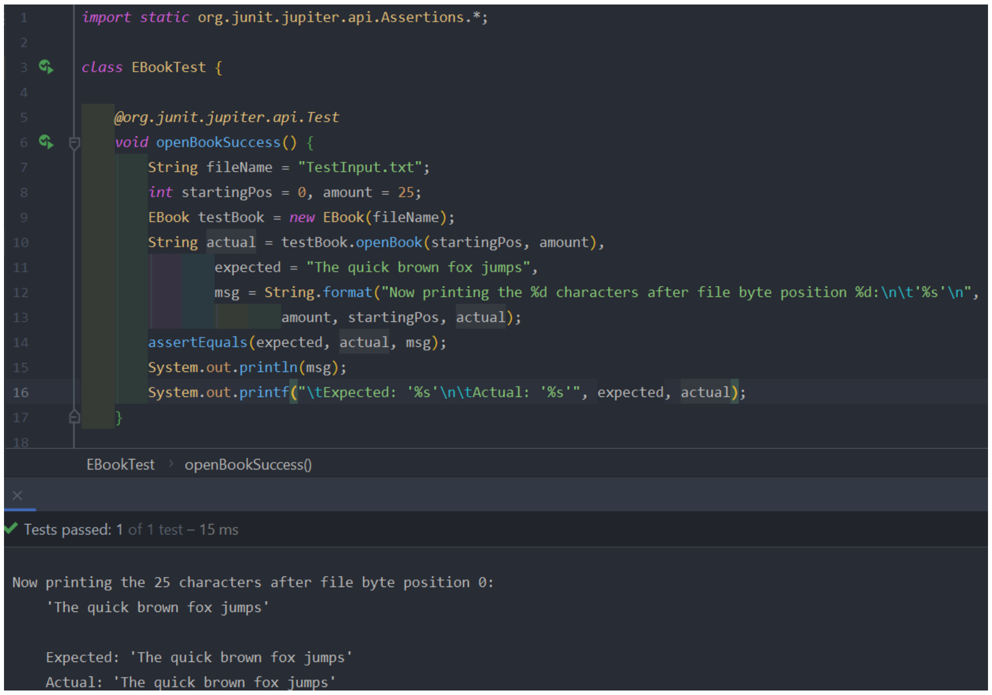
**Total Cost = $1200 x 5 people x 3 weeks = $18,000 + $300 for maintenance = $18,300**

5. [10 POINTS] A test plan for your software: Describe the test plan for testing minimum one unit of your software. As an evidence, write a code for one unit (a method for example) of your software in a programming language of your choice, then use an automated testing tool (such as JUnit for a Java unit) to test your unit and present results. Clearly define what test case(s) are provided for testing purposes and what results are obtained (Ch 8). Include your test code as additional document in your zip file submitted.
**(Jonathan, Danh)**
**Test plan:**

The unit selected to test is the book reading unit. The Bookkeeper software will have a book reading unit to read a text file into a book object. The unit will take a text file as an input. We have to show that the book reading unit can successfully interpret the text file input to a book object by outputting the first 25 characters retrieved from the original text file.

To test the unit, we will be using Junit testing technique. The idea is to use assertEquals method to verify the identicality between the first 25 characters of the text file and the unit's output.

**Successful case:**
If the text file has the first 25 characters = “The quick brown fox jumps” and we assert if the unit’s output is = “The quick brown fox jumps”.
Demo code:



```
import static org.junit.jupiter.api.Assertions.*;

class EBookTest {

    @org.junit.jupiter.api.Test
    void openBookSuccess() {
        String fileName = "TestInput.txt";
        int startingPos = 0, amount = 25;
        EBook testBook = new EBook(fileName);
        String actual = testBook.openBook(startingPos, amount),
                expected = "The quick brown fox jumps",
                msg = String.format("Now printing the %d characters after file byte position %d:\n\t'%s'\n",
                        amount, startingPos, actual);
        assertEquals(expected, actual, msg);
        System.out.println(msg);
        System.out.printf("\tExpected: '%s'\n\tActual: '%s'", expected, actual);
    }
```

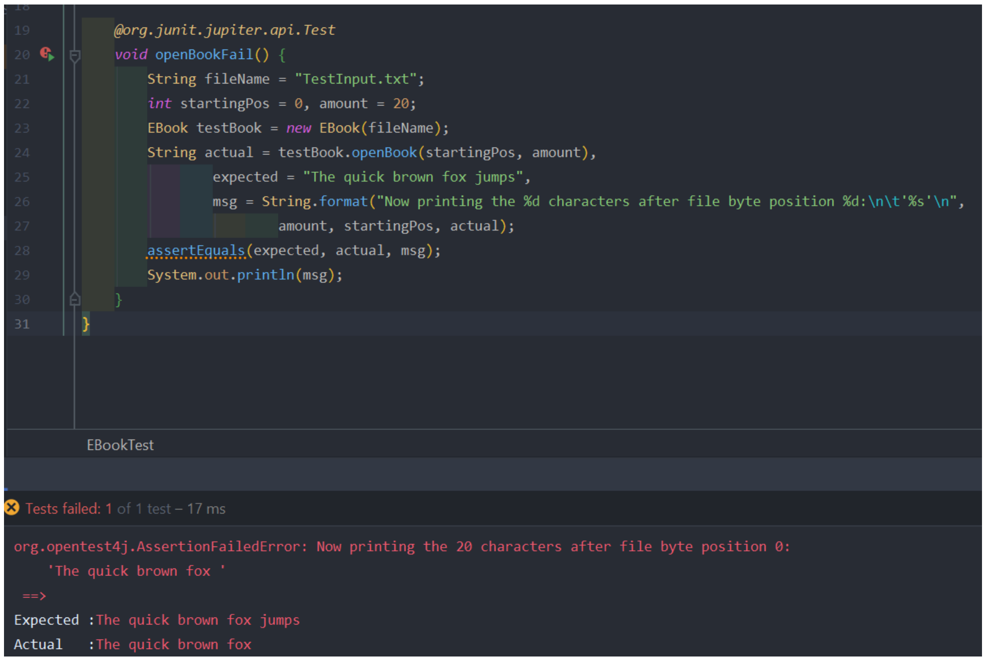
EBookTest › openBookSuccess()

Tests passed: 1 of 1 test – 15 ms

```
Now printing the 25 characters after file byte position 0:
    'The quick brown fox jumps'

    Expected: 'The quick brown fox jumps'
    Actual: 'The quick brown fox jumps'
```

**Failure case:**
IF the text file has the first 25 character = “The quick brown fox jumps” and we assert if the unit’s output is = “The quick brown fox”.
Demo code:



```
    @org.junit.jupiter.api.Test
    void openBookFail() {
        String fileName = "TestInput.txt";
        int startingPos = 0, amount = 20;
        EBook testBook = new EBook(fileName);
        String actual = testBook.openBook(startingPos, amount),
                expected = "The quick brown fox jumps",
                msg = String.format("Now printing the %d characters after file byte position %d:\n\t'%s'\n",
                        amount, startingPos, actual);
        assertEquals(expected, actual, msg);
        System.out.println(msg);
    }
}
```

EBookTest

Tests failed: 1 of 1 test – 17 ms

```
org.opentest4j.AssertionFailedError: Now printing the 20 characters after file byte position 0:
    'The quick brown fox '
 ==>
Expected :The quick brown fox jumps
Actual   :The quick brown fox
```

6. [10 POINTS] Comparison of your work with similar designs. This step requires a thorough search in the field of your project domain. Please cite any references you make. [Miranda]

The first e-book, a digital copy of the Declaration of Independence, was published in 1971 by the newly formed Gutenberg Project. Since their birth, ebooks have grown in popularity and made reading more convenient and accessible than ever. Our ebook management app, Bookkeeper, aims to preserve this momentum and give consumers another way to store, read, and organize books on their electronics. Other applications in the e-reader market include software like Calibre, Icecream, and Adobe Digital.

Calibre is a free, cross-platform e-book reader and manager for desktop computers and laptops. Like bookkeeper, Calibre allows the user to edit their libraries, search the contents of their e-books, and more. Calibre also supports other impressive features such as file type conversions, news bulletins from the user's favorite source(s), and customization of the user's reading mode.

The icecream ebook reader is another program that serves a similar purpose to our bookkeeper application. Like bookkeeper, Icecream manages the user's digital library, allows the user to search the contents of their books, organizes the user's e-books into custom categories, and supports multiple views to display the user's books. Icecream goes beyond bookkeeper's design and supports a myriad of other useful tools and functions, like bookmarks, notes and annotations, and translations using google translate.

Adobe Digital is an e-book reader and manager that allows users to access books available from libraries and digital book vendors. Like Bookkeeper, the application supports text searches, book reading/viewing, and library organization. Unlike Bookkeeper, Adobe Digital supports downloads directly from third-party sources like local libraries and ebook stores.

7. [10 POINTS] Conclusion - Please make an evaluation of your work, describe any changes that you needed to make (if any), if things have deviated from what you had originally planned for and try to give justification for such changes.

Bookkeeper, the application we are developing, has its strengths primarily in the areas of ensuring a great core experience and in pushing the envelope in terms of accessibility. Reading on computing devices is a universal, uncomplicated activity. While researching established e-reader applications, we were impressed by the wide array of features present on the most popular applications, but we also discovered a niche where we felt Bookkeeper would be able to provide value.

In particular, from our study of other e-readers, our own reading experience and preferences, and our observations of the rapid improvement in natural language processing technology, we sought to create an application that provides the customization for preferences that people naturally develop throughout a lifetime of reading while not overwhelming the user with toolbar upon toolbar of options to pick through. We identified that our foremost development goal should be maintaining a solid core experience, such that the user does not have to think about the UI beyond initial configuration and customization of a few settings such as default file path, text color, scrolling style, and page color. Building upon this goal, we also prioritized development on the major operating systems that people use on their phones, desktops, and tablets today. We felt that beyond creating an intuitive application, another key niche our application could fill would be availability across platforms. During our research, we found that even the most popular e-readers were generally unavailable on other platforms. Some were restricted only to iOS, others to Android or to Windows, and still others were available on iOS and Android but not on the desktop. We addressed this by developing Bookkeeper with support for desktop and phone operating systems as a priority from the beginning. We felt that this was a particularly crucial feature to have in order to simplify the number of programs an end-user has to learn across different platforms, and to make it easier for a user to guarantee that all their titles would be stored in the same state across all of their devices, when we implement online book buying and synchronization features in later updates.

Beyond providing a simplified, unified user experience, the other major niche that Bookkeeper fills is in providing greater accessibility to written material for anyone without access to audiobooks, including people that enjoy listening to rather than reading books, multi-taskers, and the sight-disabled. We found that through the great strides natural language processing software and mobile computer hardware has made, the time was ripe for implementing text-to-voice transcription, which is a feature that we found that other e-readers lack. With our implementation, users are no longer required to have an audiobook copy of a title for a hands-free reading experience.

Moving from the idea phase to hashing out the actual implementation, we did make substantial changes to what we had originally envisioned. We initially wanted to include Internet features, such as the ability to browse and buy from online bookstores, from the start, but we felt that with the ease of obtaining reading material from a variety of sources, that it would not be a feature that would make our application stand out, and thus we pushed the feature back to a later update until we have fine-tuned the core aspects. Furthermore, we had originally planned for a login system so that multiple users could read their own books they had saved under their own account, but after more reflection we scrapped the feature, because we felt that it introduced unnecessary complexity and again did not strengthen the core experience. The target audience for this application would have one or more unshared personal computing devices, smartphones would be unlikely to be shared, and we felt that even for multi-user computers, handling multiple users would be best accomplished through the operating system. Through this, we realized that due to the lack of sensitive data or personally identifying information used by our application should shift our priorities more to focusing on convenience features, such as ensuring that user data would be durable, through a book backup feature we added

Through our thorough analysis and research, we were surprised by the general feasibility of the application we had chosen to develop, even for a student group. Specifically, we found we could deploy our application relatively quickly, due to the limited amount and simple nature of user interactions, small number of interfaces, and lack of complex operations as outlined in our functional point analysis.

We feel that this deep-dive into methodologically planning and building our own project was particularly instructive in the importance of scope management, determining the order of feature development, utilizing agile techniques, and testing. By compartmentalizing features according to the MVC architecture and systematically analyzing the complexity of features, importance, and time needed for implementation, we were able to pare less important features such as online bookstore access and a login system from our  application, add an important feature for data backup, and ultimately create a more competitive application optimized to distinguish itself in a crowded field by focusing on accessibility and simplicity.

8. [5 POINTS] References: Please include properly cited references in IEEE paper referencing format. Please review the IEEE referencing format document at the URL: 4 https://ieeedataport.org/sites/default/files/analysis/27/IEEE%20Citation%20Guidelines.pdf). It means that your references should be numbered, and these numbers properly cited in your project report. Also include:

**[1] (2010, August 11). Ebook Reader. Retrieved November 03, 2020, from https://apps.apple.com/us/app/ebook-reader/id381260755**

9. [10 POINTS] Non-recorded (no-voice) presentation slides. No min/max number of slides enforced. Please make sure that you can complete the presentation within 15 (fifteen) minutes. Following templates could be a good start to prepare your presentations.
As each project topic is different, a variety in presentation style is expected and welcome.
- Title of your project together with participants
- Objective of the project designed
- Cost estimation
- Project timeline (timeline of the project designed, NOT the time you've spent on it)
- Functional and non-functional requirements. If too long, select representative items.
- Use case diagram
- Sequence diagram for a selected representative operation of the project.
- Class diagram
- Architectural design
- Model-View-Controller (MVC) pattern (similar to Figure 6.6)
- Layered architecture pattern (similar to Figure 6.9)
- Repository architecture pattern (similar to Figure 6.11)
- Client-server architecture pattern (similar to Figure 6.13)
- Pipe and filter architecture pattern (similar to Figure 6.15)
- Preferably a demo of user interface design that shows screen to screen transitions though no full functionality is required.
- OPTIONAL: IF implemented the project, a demo of your implementation.

10. OPTIONAL PART. Your program code (if fully implemented the project, not required otherwise). Please note that implementation is not required for the final project. Groups are welcome to implement their work, if they choose to do so. [This part may qualify for extra credit, if you implement and submit the implementation code together with your project. The extra credit will be determined based on the quality of your implementation]

11. [5 POINTS] GitHub requirement: Make sure at least one member of your group commits everything for project deliverable 2 to your GitHub repository, i.e. **(Danh)**

- Your final project deliverable2 report

- Unit test code for a sample unit of your project
- Implementation code (if you have implemented your project)
- Non-recorded (no voice) presentation slides

Still, one member of your team should also submit the required project deliverable 2 materials to eLearning.

Please note: This is just a suggested outline. You are welcome to add more content if you feel necessary. (NO EXTENSION IS POSSIBLE ON FINAL PROJECT DELIVERABLE 2 SUBMISSION DUE DATE).

No min/max page, font type restrictions.
IMPORTANT NOTE: Please use an automated tool for drawing all diagrams required in the deliverables. No manual drawing please.

UML Editors: Following is a list of some freely available UML editors for your convenience:
• Sparks Enterprise Architect http://www.sparxsystems.com/. 30 day trial version only.
• Violet UML editor http://alexdp.free.fr/violetumleditor/page.php. Very simple features of UML design. Free.
• Omondo EclipseUML http://www.omondo.com/ (Academic License available for free) works with Eclipse http://www.eclipse.org/
• StarUML http://staruml.sourceforge.net/en/ see also StarUML @ Wikipedia Open-source UML modeling tool supports most of the diagram types specified in UML 2.0
• UMLet http://www.umlet.com/ Open-source UML tool; runs stand-alone or as an Eclipse plug-in on Windows, OS X, and Linux
• Visual Paradigm for UML (Community Edition) http://www.visualparadigm.com/product/vpuml/editions/community.jsp The Community Edition is free for non-commercial use; It puts a “Community Edition” watermark on your diagrams; Runs on Windows XP/Vista/7, Linux, Mac OS X, etc.
• Netbeans UML Plug-in http://www.netbeans.org/features/uml/ Does not support all UML diagram types, but supports forward and reverse engineering
• ArgoUML http://argouml.tigris.org/ see also ArgoUML @ Wikipedia
• Rational Rose http://www.rational.com/tryit/index.jsp
• http://www.microgold.com/
• Microsoft Visio and open-source Dia are diagramming tools with a library of UML shapes that may also be used for drawing UML diagrams.
• Creatly http://creately.com/ for drawing UML diagrams.

Making life easy when working as a group:

It is very important to make sure that you communicate and share common work with your teammates. Here are some URLs to help you on that:

• Github — a web-based Git or version control repository and Internet hosting service. This is the recommended version control software for this project. If by some reason you cannot use the Github platform, you may use any of the following similar platforms for sharing your project related material.
• Doodle—a tool for time management and meeting scheduling.
• GroupMe—a group messaging service that lets you be in touch with your team members via mobile phones.
• CVS, open source version control - helps you work on different versions of the same product and merge your versions.
• Slack — a web-based team communication service.
• Mercurial https://www.mercurial-scm.org/ for version control.

**About Presentation of your Project:**

Regardless of which option you choose, presentation should take maximum 15 (fifteen) minutes. No minimum time restriction. As we have many groups and limited time for presentations, we will have to time each presentation. So, thank you for your attention on the matter. Each team member is expected to talk during presentation. The presentation dates are: 11/12 11/17 11/19 11/24

The presentation schedule will be posted to eLearning once students submit their final deliverables. You may use any style in your presentations.

A slide show is recommended as it helps with displaying a summary of content you talk about to the audience as well as to yourself.

A suggested outline for presentations is listed below:

A brief introduction to your project topic
List of requirements
Use case diagram that contains use cases
Sequence diagram
Design Class Diagram (DCD)
User Interface Design Comparison with similar work (if any), or emphasizing its significance and uniqueness (if there is not any)
Conclusion and Future Work

You are welcome to enhance the minimum content listed above, provided that you stay within maximum 15 minute presentation time requirement. We will listen to multiple presentations per day, so please try not to exceed the allocated time for your presentations so as not to steal from other groups' presentation time.

Feel free to enrich your presentation with supporting figures, charts, documents, tables, similar work, etc.

Contribute from yourselves: Employ your own design layouts, color selections, animations, artistic perspectives to your presentations. Try to make them attractive. Think of commercials: We only remember the “interesting” ones.

It is a suggested tactic that in a presentation, each slide should remain min. 1 minute on display so that everybody reads and understands it. So, not too many slides maybe a good idea to start with.

Rehearsal will prevent unexpected surprises. Make sure you rehearse and time your presentation before you actually present it.

**Project Submission:**

Each group should designate one team member to submit his work via eLearning only. PLEASE SUBMIT ONE PROJECT PER TEAM. Each member of the group will receive the same grade, unless group members report a poor member performance.

What to submit? Please zip the following as one single file:
- Final project deliverable2 report (Please note that your deliverable2 report should also include your deliverable1 report as required in section 2 above.)
- Test code (section 5 above)
- Non-recorded (no voice) presentation slides (section 9 above)
- [Optional] If you have fully implemented your project, include your implementation code (section 10 above)

**Grading Policy:**
Your project grade will be the average of your report and in class presentation evaluation. So, please make sure to delegate tasks and contribute fairly. Remember that it is part of group work responsibility to delegate tasks and ensure that everybody contributes. Members of each group WILL receive the same grade. So, if there is any problem in your group, please act timely. Writing a paper at the end of your work and submitting it to a journal/conference is strongly encouraged. I will support students who want to proceed with implementation of their design and submit their work to a journal/conference.