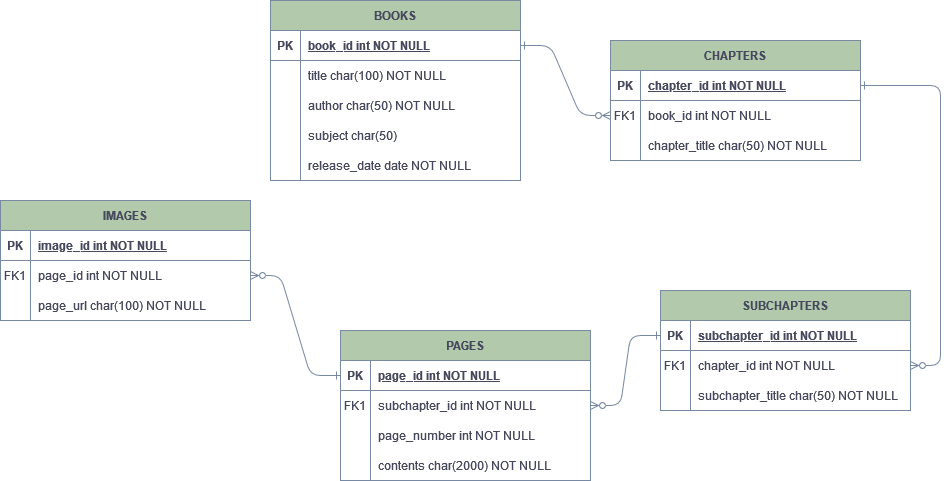
**Outline Testing strategy**:

* Verify the Content screen content
* Verify the Home screen content
* Verify the Library screen content
* Verify the Search screen content
* Verify that the **Switch** button is working in the Reading screen
* Verify that the **OutlinedTextField** search is working in the Search screen
* Verify that the **toggleReadingMode** is working in the Reading app view model
* Navigating to the Library screen by clicking the **go to library** button from the Home screen
* Navigating to the Contents screen by clicking the **Table of Contents** button from the Library screen
* Navigating to the Reading screen by clicking on one of the books in the Library screen
* Navigating to the Reading screen by clicking the **Read** button from the Library screen
* Verify that the **listDirectoryContents** is working in the FileSystem class
* Verify that the **deleteDirectoryContents** isdeleting the directory contents
* Verify that the **createFile** is creating a new file in the expected directory
* Verify that the **downloadFile** download file from URL and save it in the specified location

**Explanation:**

These tests will make sure that all the major features of the application work right: the user will feel comfort moving from screen to screen, performing different actions. The tests have touched upon content verification, navigation, and main functions like searching, reading modes, and file management. The testing strategy includes content verification, navigation across screens, like Home, Library, Search, and Reading. It also includes button and input field testing for expected functionality, among others. In addition, it includes essential backend functionality that involves file handling like creating, deleting, and downloading of files. Also, we plan on unit testing any new function or feature that we add right after implementing it, so that we catch errors early on and can fix them before they cause any trouble later.



In our database, we will have a table to store the general book information such as the title, author, subject, and release date. This information will be displayed in the library screen. When a book is clicked, the app will make a query to the database to find that book’s chapters and subchapters, and it will display them in the table of contents screen. When a subchapter is pressed, the app will make a query to the database to find that subchapter’s pages and contents, and any images contained in that page. Since images can’t be stored in text, we will have a separate table containing all the image URLs in a certain page in order to be able to access them. We chose this data model because it matches the layout of our app, with each table or set of tables being connected to a screen in our app, which will make it simple to implement the database part in our project. This will also make it easier to format the book for each page, by splitting it up by chapters and pages.

Library screen: Books table

Table of contents screen: Chapters and Subchapters tables

Reading screen: Pages and Images tables

Examples of queries:

SELECT \* FROM Books;

SELECT \* FROM Chapters WHERE book\_id = 1;

SELECT \* FROM Pages WHERE subchapter\_id = 3;

INSERT INTO Books (title, author, subject, release\_date) VALUES (“book1”, “author1”, “science”, “24-JAN-25”);

INSERT INTO Pages (subchapter\_id, page\_number, contents) VALUES (2, 3, “book contents go here”);

INSERT INTO Images (page\_id, page\_url) VALUES (5, “www.example.com/image”);