MULUNGUSHI UNIVERSITY

SCHOOL OF SCIENCE, ENGNEERINY AND TECHNOLOGY

Group 5 - Library Finder (Campus)
ICT 381
COURSE WORK

Student Name	Student IDs
1. Dennis Sampa	**202302571**
2. Gift Saya	**202302133**
3. Tabo Mwewa	**202302583**
4. Lloyd Bulaya	**202306274**
5. Joy Silishebo	**202302837**
6. Mwikisa kapalu	**202304618**
7. Cleopatra Ngoma	**202301392**
8. Mushilo Mwewa Mayuya	**202302315**
9. Dada Mshisha	**202200670**
10. Annette Sianongo	**202302878**
11. Chabala Kasonde	**202307569**

Systems Modeling & Design - Week 1 Deliverables

1. **Vision Summary**

The Campus Library Finder is an offline-first mobile application designed to provide Mulungushi University students with a seamless and efficient way to browse the campus library's book catalog. The core problem it solves is the difficulty students face in physically searching for books, checking their availability, and remembering their favorites without a reliable, centralized, and offline-accessible digital tool.

Our application will allow students to search for books by title or author, filter by category, view detailed information (including shelf location), and mark books as favorites for quick access later. An optional admin mode will enable librarians to add new books to the catalog. By digitizing this process, we aim to save students time, reduce frustration, and encourage greater use of the library's resources. The app will strictly follow the MVC architecture for maintainability and clarity.

2. **Requirements List & User Stories**

A) Functional Requirements

ID	*Requirement Description*	*Priority*	*User Story (As a... I want to... so that...)*	*Acceptance Criteria*
FR1	The system shall allow users to browse a list of all books in the catalog.	Must	As a student, I want to see a list of all available books, so I can casually browse the library's collection.	- List is displayed in a scrollable view. - Each list item shows at least the book title and author.
FR2	The system shall allow users to search for books by title or author.	Must	As a student, I want to search for books by title or author, so I can quickly find a specific book I need.	- A search bar is present. - Typing text filters the book list in real-time. - Results match partial text in title or author fields.
FR3	The system shall allow users to filter the book list by category.	Should	As a student, I want to filter books by category (e.g., "Computer Science", "Fiction"), so I can narrow down my options to a relevant subject.	- Selectable category chips or a spinner are available. - Selecting a category updates the list to show only books from that category.
FR4	The system shall allow users to view detailed information about a book.	Must	As a student, I want to view detailed information about a book, so I can learn its category, shelf location, and a description before going to find it.	- Details screen displays title, author, category, shelf code, and description. - Screen is accessible from the main list.
FR5	The system shall allow users to mark a book as a favorite.	Should	As a student, I want to mark books as favorites, so I can easily find them later without searching again.	- A toggle/heart icon is present on the book details and list items. - Favorited books are visually distinct. - A separate "Favorites" list is available.
FR6	The system shall provide a separate view to see all favorited books.	Should	As a student, I want to see a list of all my favorite books, so I can have a personalized, quick-access library.	- A "Favorites" screen exists. - It only shows books I have favorited.
FR7	The system shall allow an Admin user to add a new book to the catalog.	Could	As a librarian (admin), I want to add new books to the catalog, so the app's inventory stays up-to-date.	- An "Add Book" button (FAB) is visible in admin mode. - A form with fields for title, author, category, and shelf is presented. - Saving the form adds the book to the main catalog.
	The system shall	Could	As a student, I want to	- A "Share" button is

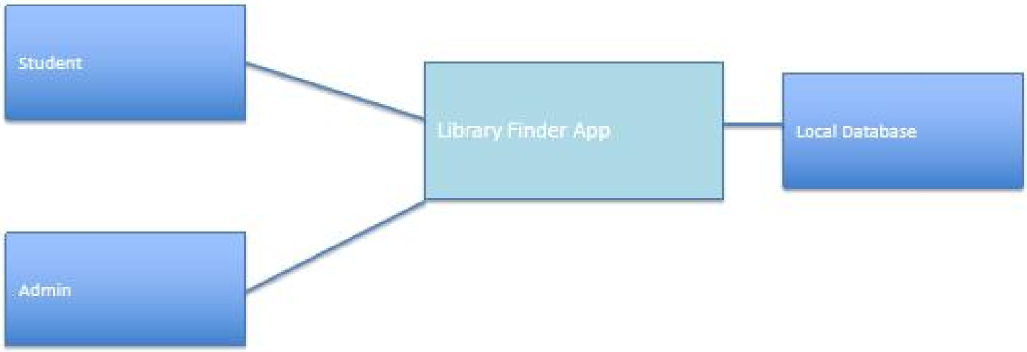
FR8	share a book's citation via other apps.		share a book's details, so I can send it to a friend or save it in my notes.	present on the details screen. - Tapping it opens the standard Android share sheet with pre-filled book text.

B) Non - Functional Requirements

ID	Requirement Description	Priority	User Story (As a... I want to... so that...)	Acceptance Criteria
NF R1	The system shall be fully functional without an internet connection (Offline-First).	Must	As a student, I need the app to work anywhere on campus, even in areas with poor or no internet.	- All core features (browse, search, view details, favorites) work offline. - Data is persisted locally on the device.
NF R2	The user interface shall be accessible, with clear text and descriptions for images.	Must	As a visually impaired student, I need the app to be readable by a screen reader, so I can use it independently.	- All images have content descriptions. - Text has sufficient contrast and size. - All interactive elements are focusable.
NF R3	The app shall have a responsive and smooth user interface.	Should	As a user, I expect the app to respond quickly to my touches and scrolls, so I don't get frustrated.	- The app does not freeze or lag during scrolling, searching, or navigation.

Priority Legend:
Must = Core functionality,
Should = Important but not vital,
Could = Nice-to-have.

3. **Context Diagram**



Description:

- The Student and Librarian (Admin) are the primary actors interacting with the system.
- The system's core boundary is the Campus Library Finder App.
- The app interacts with Device Storage (Local DB) to persistently save all book data, favorites, and settings.
- The Librarian actor interacts with the app in an optional admin capacity to add books.
- The app can use the External Share intent to send book details to other applications on the device.

4. Draft Backlog for Week 2

This backlog is prioritized based on the "Must/Should/Could" requirements from Week 1.

High Priority (Must Have)

1. ***Create Use Case Diagram:*** Identify all actors (Student, Librarian) and use cases (Browse Books, Search Books, View Book Details).
2. ***Detail 3 Core Use Cases:***
   Search for a Book (Primary Actor: Student)
   View Book Details (Primary Actor: Student)
   Browse Book Catalog (Primary Actor: Student)
3. ***Create Sequence Diagram*** for "**Search for a Book**" showing interaction between Student, Search Controller, and Book Repository/DB.

Medium Priority (Should Have)

4. ***Detail 2 Secondary Use Cases:***
   Mark Book as Favorite (Primary Actor: Student)
   View Favorites List (Primary Actor: Student)
5. ***Create Sequence Diagram*** for "**Mark Book as Favorite**".

Lower Priority (Could Have)

6. ***Detail Use Case:*** "**Add New Book**" (Primary Actor: Librarian).
7. ***Create Sequence Diagram*** for "**Add New Book**".

WEEK 2 AND 3 IMPLEMENTATION

Use Case Diagram

This diagram shows all the ways users can interact with our Library Finder app.It's like a menu of all possible actions.

Detailed Explanation:

PURPOSE:

- Shows all system functionalities from user's perspective
- Identifies who can do what in the system
- Defines boundaries between system and external users

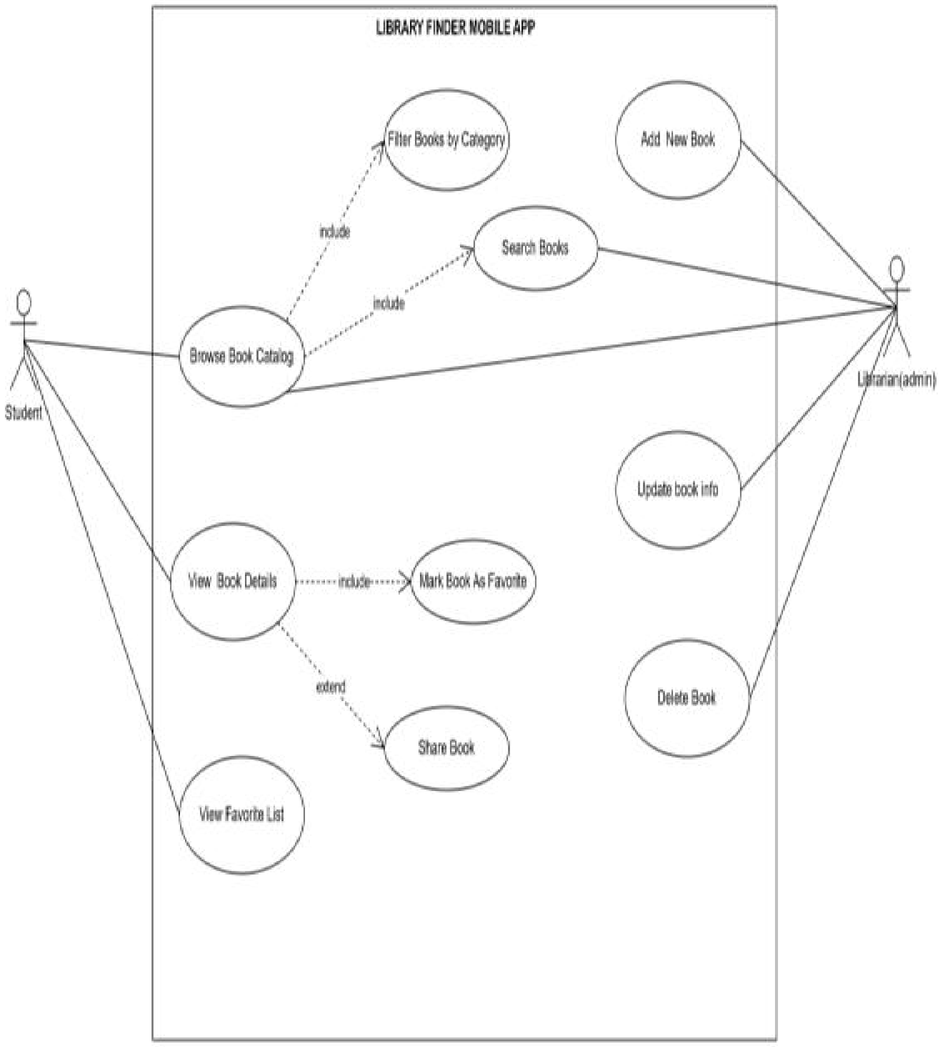
KEY COMPONENTS:

- ***ACTORS:*** Student (primary), Librarian (secondary)
- ***USE CASES***: Search Books, View Details, Mark Favorite, etc.
- ***RELATIONSHIPS***: Lines showing which actors perform which actions

WHAT IT SHOWS:- Students can search, browse, view details, and manage favorites- Librarians can do everything students can PLUS add/edit books- Clear separation between regular user and admin function

WHY IT'S IMPORTANT:

- Ensures we cover all user requirements
- Helps identify missing functionalities
- Guides development of user interfaces



Sequences Diagram

This diagram shows step-by-step what happens when a user performs a specific action,like searching for a book.

Detailed Explanation:

PURPOSE:

- Shows the order of operations between different system components
- Illustrates how objects interact over time
- Details the flow of messages between components

KEY COMPONENTS:

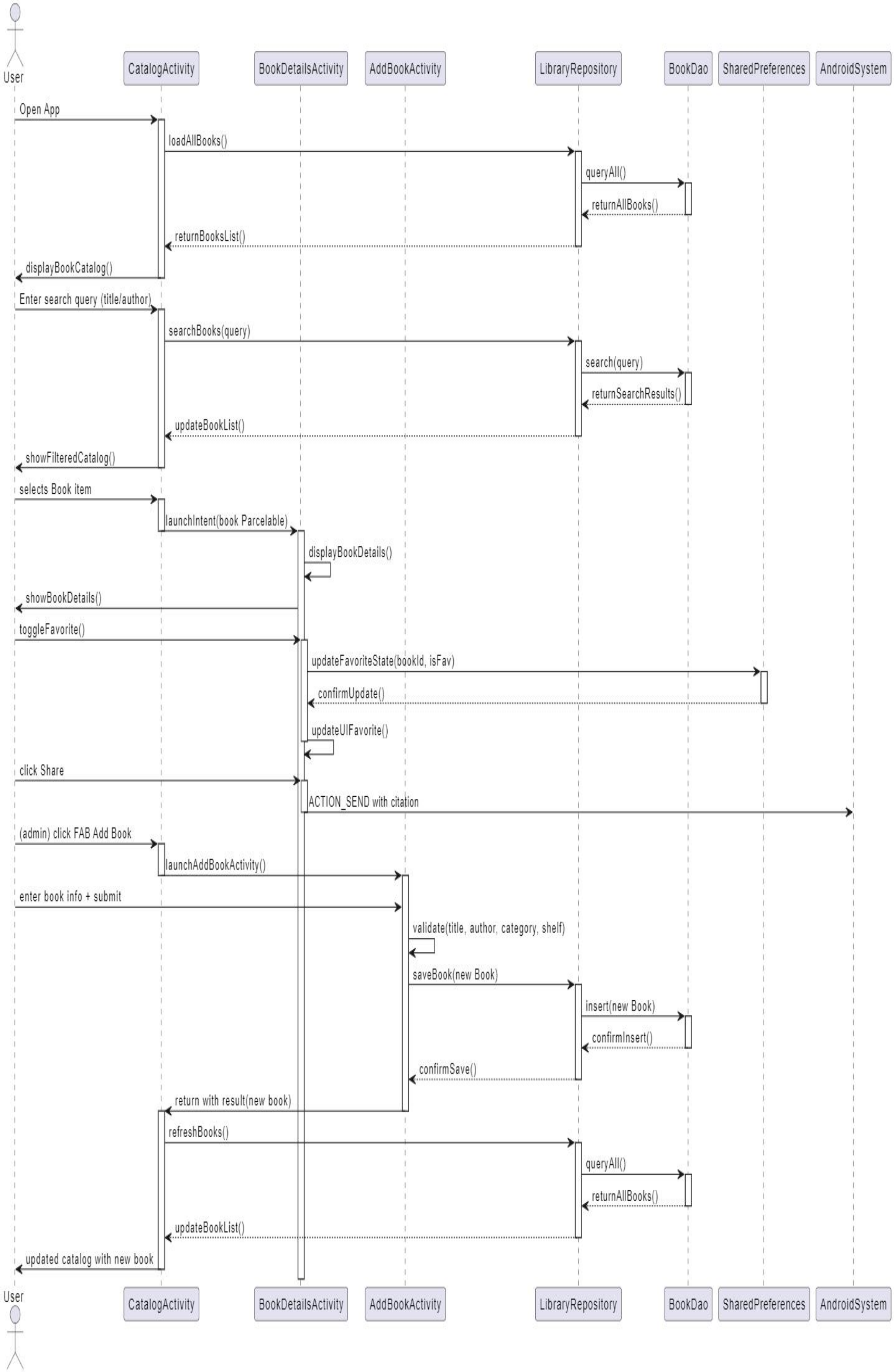
- ***LIFELINES***: Student, Controller, Repository, Database
- ***MESSAGES*****:** Method calls and responses between components
- ***ACTIVATION BARS*****:** When each component is active
- ***ALT FRAGMENTS***: Different paths for success/error cases

WHAT IT SHOWS:- How a search request flows from UI to database and back- What happens during normal operation vs error scenarios- Timing and order of all system interactions

WHY IT'S IMPORTANT:

- Identifies potential bottlenecks in the system
- Ensures all components communicate correctly
- Helps debug timing-related issues

LibraryFinder Sequence: Search, View, Favorite, Add Book (Admin) with Activation Boxes



Activity Diagram

This diagram shows the complete workflow of using the app,like a flowchart of all possible user journeys.

Detailed Explanation:

PURPOSE:

• Models the complete workflow of business processes

• Shows decision points and parallel activities

• Illustrates the flow from one activity to another

KEY COMPONENTS:

• ***START/END NODES***: Beginning and conclusion of processes

• ***ACTIVITIES***: Specific tasks or operations (Search, View Details)

• ***DECISION NODES***: Branching points (Yes/No, Success/Error)

• ***SWIMLANES***: Separates different actors (Student vs Admin and System)

WHAT IT SHOWS:

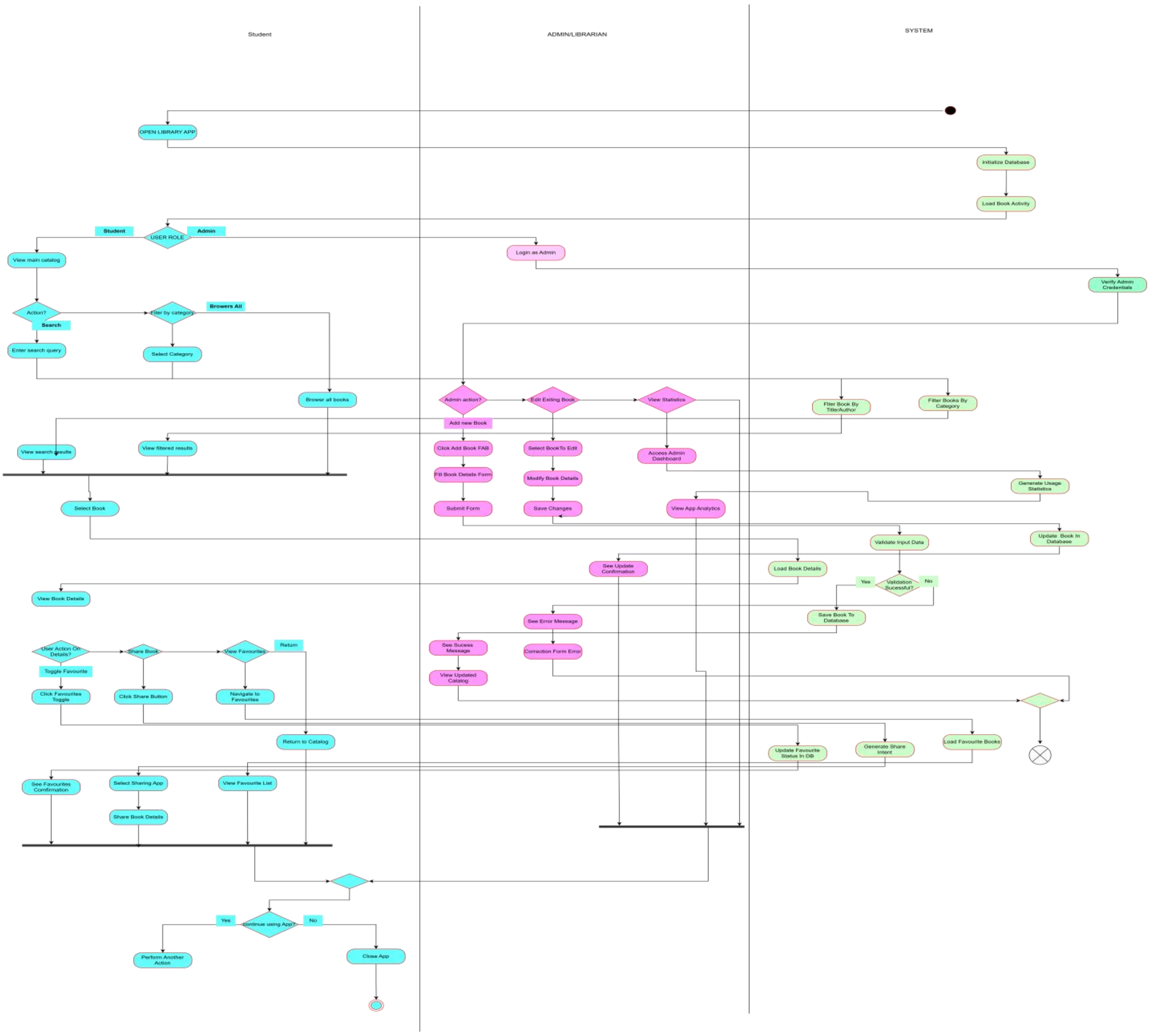
- Complete user journey from opening app to completing tasks
- All possible paths a user can take through the system
- How different user roles have different workflows
- Error handling and alternative flows

WHY IT'S IMPORTANT:

• Provides complete picture of system behavior

• Helps identify complex workflow scenarios

• Useful for testing all possible user paths



Class Diagram

This diagram shows the building blocks of our app- the classes, their attributes, and how they connect to each other.

Detailed Explanation:

PURPOSE:

- Shows the static structure of the system
- Defines classes, their attributes, methods, and relationships
- Illustrates the object-oriented design

KEY COMPONENTS:

- ***CLASSES***: Book, Student, Librarian, etc.
- ***ATTRIBUTES***: Properties of each class (title, author, etc.)
- ***METHODS****:* Operations each class can perform
- ***RELATIONSHIPS***: How classes are connected (inheritance, association)

WHAT IT SHOWS:

- All data entities in our system and their properties

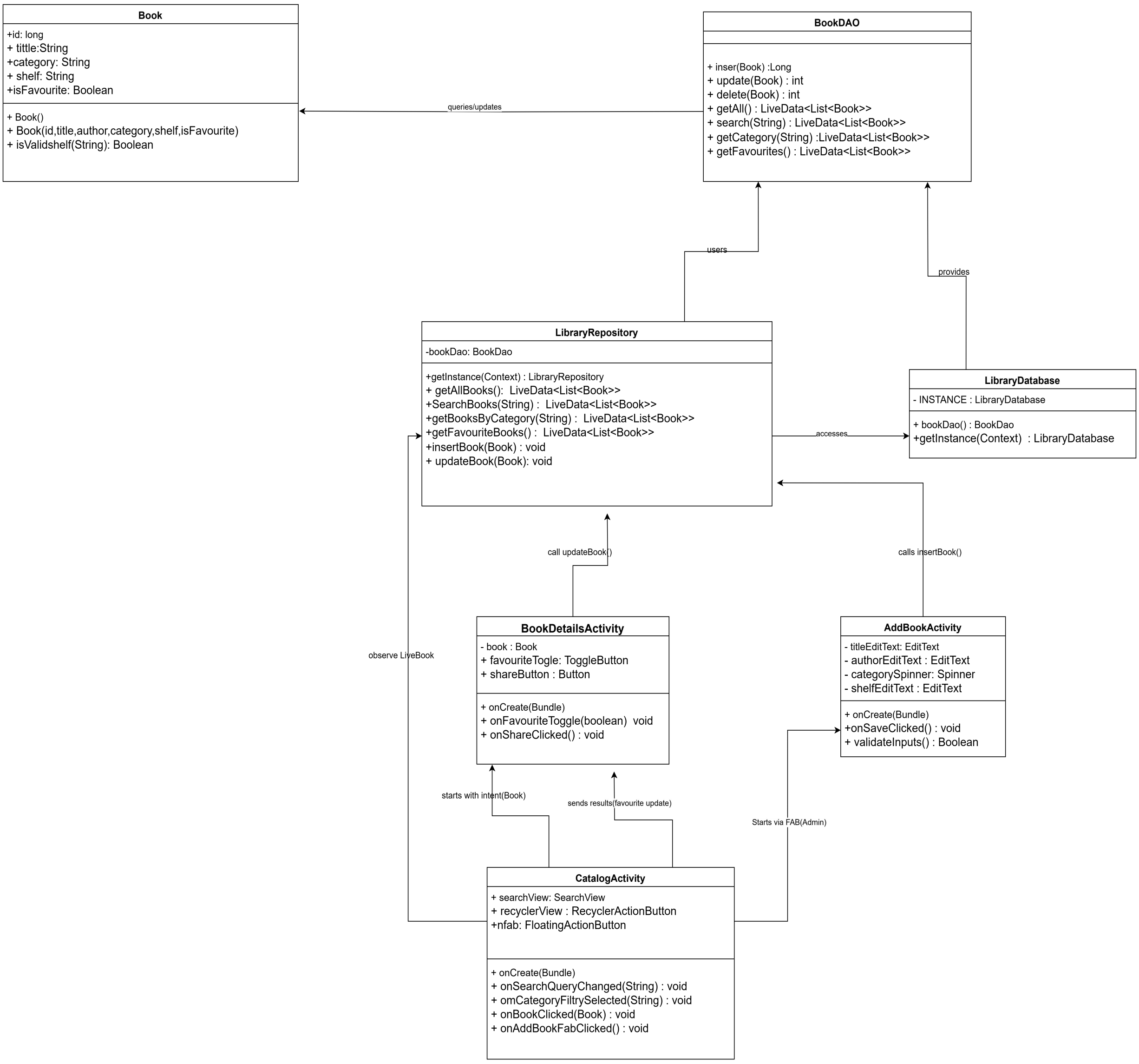
- How different classes relate to each other
- The complete data model structure
- MVC architecture separation (View, Controller, Model classes)

WHY IT'S IMPORTANT:

• Blueprint for actual code implementation

• Ensures data consistency across the system

• Guides database design and object relationships


Book
+id: long
+ tittle:String
+category: String
+ shelf: String
+isFavourite: Boolean
+ Book()
+ Book(id,title,author,category,shelf,isFavourite)
+ isValidshelf(String): Boolean
queries/updates
BookDAO
+ inser(Book) :Long
+ update(Book) : int
+ delete(Book) : int
+ getAll() : LiveData<List<Book>>
+ search(String) : LiveData<List<Book>>
+ getCategory(String) :LiveData<List<Book>>
+ getFavourites() : LiveData<List<Book>>
users
provides
LibraryRepository
-bookDao: BookDao
+getInstance(Context) : LibraryRepository
+ getAllBooks(): LiveData<List<Book>>
+SearchBooks(String) : LiveData<List<Book>>
+getBooksByCategory(String) : LiveData<List<Book>>
+getFavouriteBooks() : LiveData<List<Book>>
+insertBook(Book) : void
+ updateBook(Book): void
accesses
LibraryDatabase
- INSTANCE : LibraryDatabase
+ bookDao() : BookDao
+getInstance(Context) : LibraryDatabase
call updateBook()
calls insertBook()
observe LiveBook
BookDetailsActivity
- book : Book
+ favouriteTogle: ToggleButton
+ shareButton : Button
+ onCreate(Bundle)
+ onFavouriteToggle(boolean) void
+ onShareClicked() : void
AddBookActivity
- titleEditText: EditText
- authorEditText : EditText
- categorySpinner: Spinner
- shelfEditText : EditText
+ onCreate(Bundle)
+onSaveClicked() : void
+ validateInputs() : Boolean
starts with intent(Book)
sends results(favourite update)
Starts via FAB(Admin)
CatalogActivity
+ searchView: SearchView
+ recyclerView : RecyclerActionButton
+nfab: FloatingActionButton
+ onCreate(Bundle)
+ onSearchQueryChanged(String) : void
+ omCategoryFiltrySelected(String) : void
+ onBookClicked(Book) : void
+ onAddBookFabClicked() : void

Deployment Diagram

This diagram shows where all the pieces of our app will live and how they connect- like a map of our system's physical layout.

Detailed Explanation:

PURPOSE:

- Shows the physical deployment of software components
- Illustrates how system components are distributed across hardware
- Defines the runtime architecture

KEY COMPONENTS:

- ***NODES***: Physical devices (Mobile Phone, Server)
- ***COMPONENTS***: Software pieces (App, Database, Services)
- ***CONNECTIONS***: How components communicate with each other

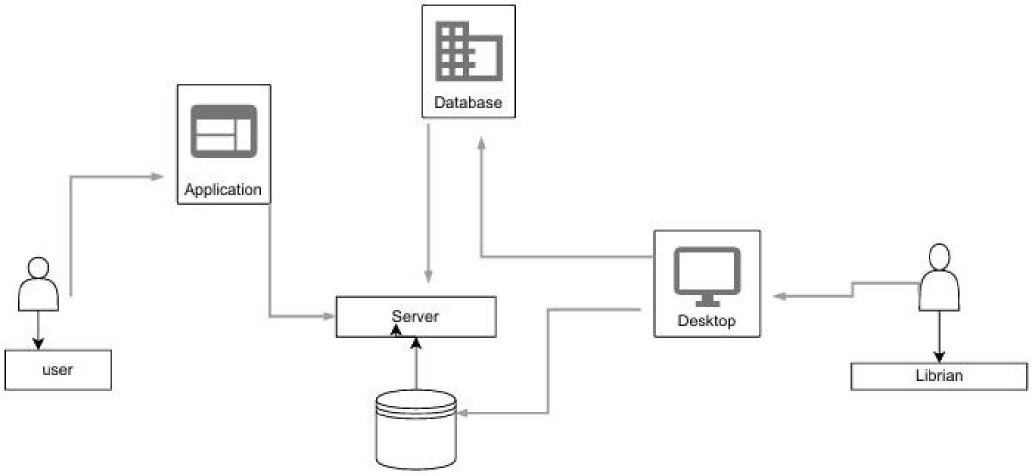
WHAT IT SHOWS:

- Our app runs entirely on the Android mobile device
- Room SQLite database is embedded in the same device
- External integrations with Camera and Share functionality
- No external servers required for core functionality

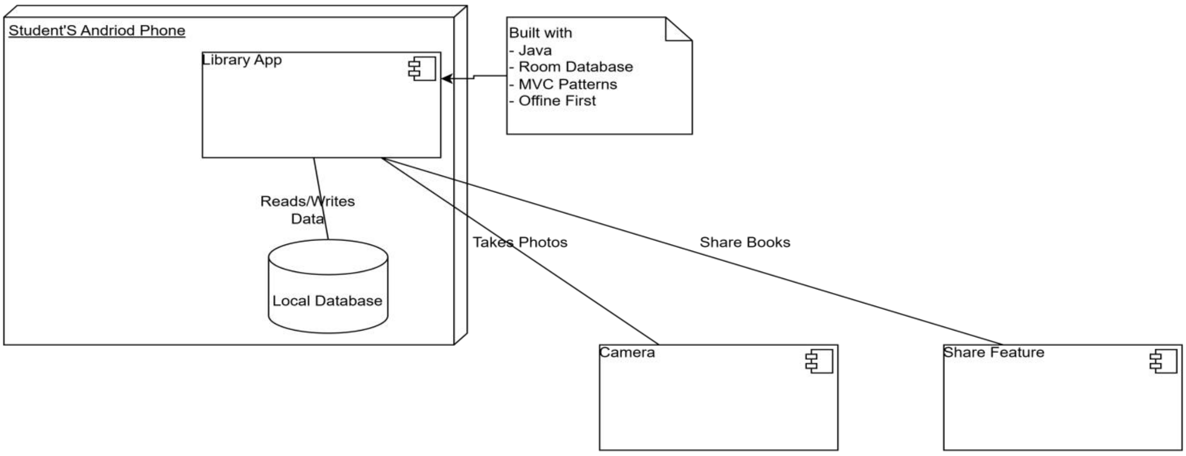
WHY IT'S IMPORTANT:

- Clarifies hardware/software requirements
- Shows system scalability limitations
- Identifies potential performance bottlenecks
- Guides installation and deployment decisions

.



LIBRARY FINDER DEPLOYEMENT DIAGRAM



Simple Diagram Summary Table

Diagrams	Week	Purpose
Context Diagram	1	Shows system boundaries
Use case Diagram	2	Shows User and Librarian functions
Sequence Diagram	2	Shows step by step flow
Activity Diagram	3	Shows User and Librarian work flows
Class Diagram	3	Shows system structure
Deployement Diagram	3	Shows app architecture