Ishika
Btech cse(Sec-D)
2301010242

Software Engineering

Assignment-2

1. 1. LOC Estimation:
   - Inputs / Outputs: 25 modules × 100 LOC = 2500 LOC
   - User Interfaces: 10 modules × 150 LOC = 1500 LOC
   - External Files: 8 modules × 200 LOC = 1600 LOC

   Total LOC ≈ 5600

   Function Point Estimation:
   - Inputs / Outputs = 25 × 4 = 100 FP
   - User Interfaces = 10 × 4 = 40 FP
   - External Files = 8 × 7 = 56 FP

   Total FP = 196

2. $n_1 = 15$, $n_2 = 25$, $N_1 = 80$, $N_2 = 100$

   Vocabulary $(n) = n_1 + n_2$
   $= 15 + 25 = 40$

   Program Length $(N) = N_1 + N_2$
   $= 80 + 100 = 180$

   Volume $= N \times \log_2(n)$
   $= 180 \times \log_2(40)$
   $= 180 \times 5.32$
   ≈ 957.6

   Effort = Difficulty × V
   $= \frac{n_1}{2} \times \frac{N_2}{n_2} \times V$
   $= \frac{15}{2} \times \frac{100}{25} \times 957.6$
   ≈ 28728

3. LOC = 5600, KLOC = 5.6

Using Basic COCOMO (Organic) coefficients:

$a = 2.4$, $b = 1.05$, $c = 2.5$, $d = 0.38$

Effort = $a \times (KLOC)^b$

$= 2.4 \times (5.6)^{1.05}$

≈ 14.649 person-months

Development Time (TDEV): $c \times (Effort)^d$

$= 2.5 \times (14.649)^{0.38}$

≈ 6.9334 months

Average staff = $\frac{Effort}{TDEV}$

$= \frac{14.649}{6.9334}$

≈ 2.1128 persons

2. 1. Top-Down:

Start with main modules (Order Management, Payment, Delivery Tracking, Customer) and refine into submodules.

Bottom-Up:

Begin with lower-level components (Login system, GPS tracking, Payment API) and integrate upwards.

Hybrid Approach:

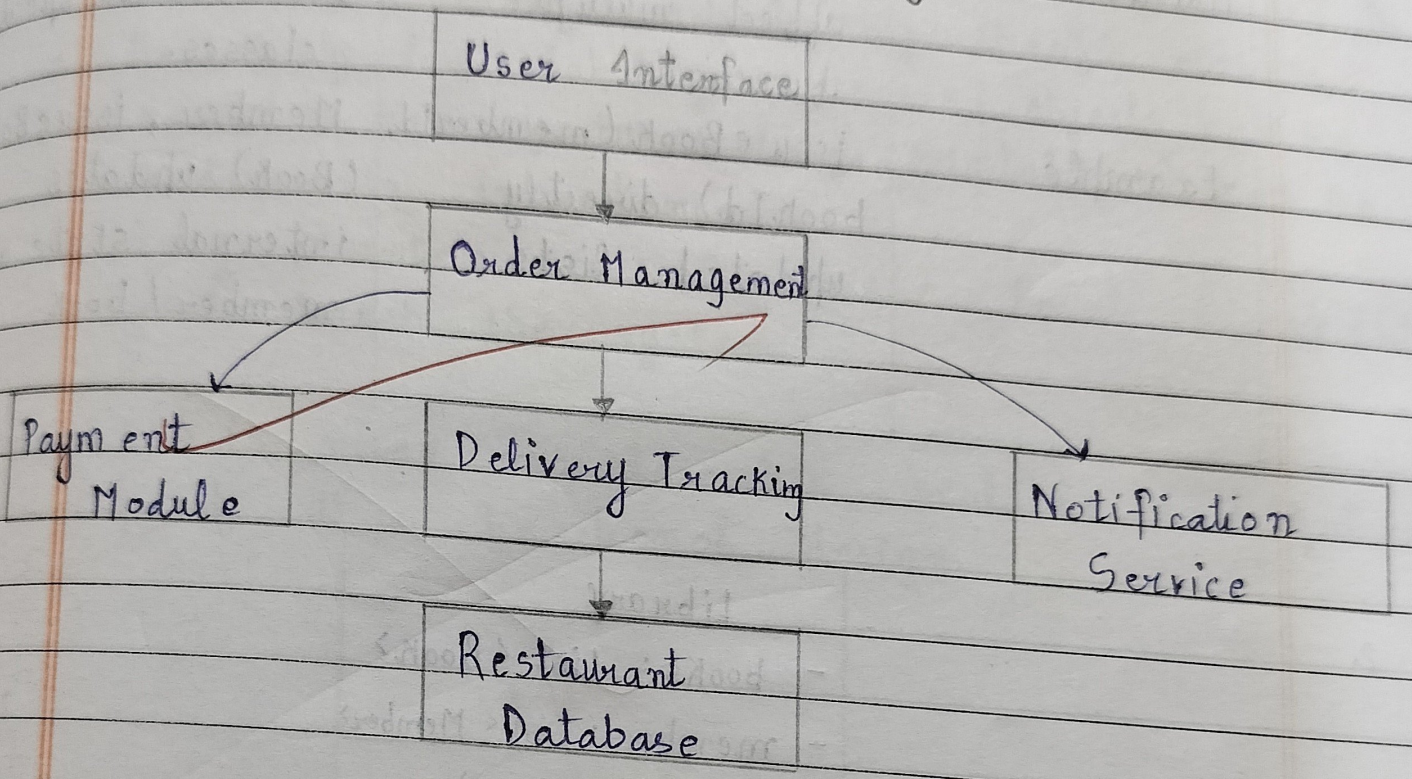
Combine both for efficiency.

2. Design Decisions:

- Cohesion: Each module should perform one well-defined task.

- Coupling: Keep modules independent (Delivery Tracking seperate from Payment).
- Abstraction: Use APIs and interfaces to hide internal details.

3. Component - Level Design Diagram:

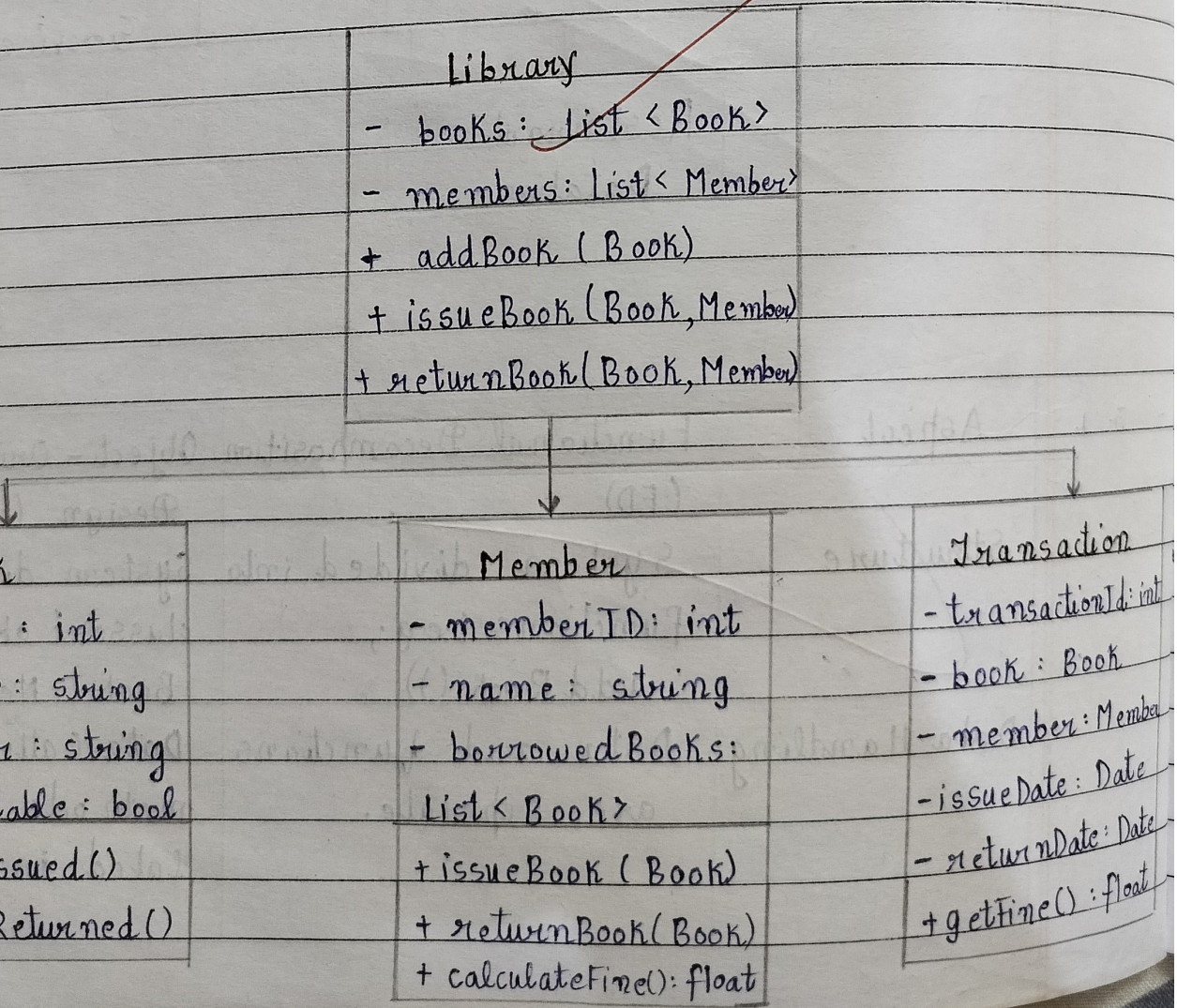


3. 1.

Aspect	Functional Decomposition (FD)	Object - Oriented Design (OOD)
Structure	System divided into functions (e.g., addBook())	System divided into classes/objects (e.g. Book, Member)
Data Handling	Data and functions are seperate.	Data (attributes) & methods encapsulated in classes

Extensibility	Adding features may require multiple function changes	Easy to extend via new classes or subclassing.
Reusability	Limited; functions often tied to specific data.	High; classes/objects reusable across projects.
Maintenance	Harder; changes may affect multiple functions.	Easier, changes localized to specific classes.
Example	issueBook(memberId, bookId) directly updated lists	Member, issueBook (Book) updates internal state of member / book.

2.



3. Best Design for Scalability & Maintenance: Object-Oriented Design (OOD)

Reasons:

- Encapsulation & Modularity: Each class manages its own data; changes affect only specific classes.
- Reusability & Scalability: Classes/objects can be reused; new features (e.g. DigitalBook) can be added via inheritance.
- Easy Maintenance: Bugs are localized, update won't break the system.

OOD is more maintainable, scalable and future proof than functional design.

Marwa
30/09