ABHINAV JHA
2301010211
B.TECH CSE CORE - 4

Page No.
Date

Assignment - 2

1.i)

Type	Quantity	Assumed Avg. LOC	Total LOC
Input/Output modules	25	200	5000
User Interfaces	10	300	3000
External files	8	150	1200
			10,200

I/O Modules = 25 (Assumed : 12 = EI, EO = 13)
Unique Operators : 15
Unique Operands = 25
Total Operators : 80
Total Operands : 100

ii) Function Point (FP)

Component	Quantity	Weight (Avg.)	FP
External Input (EI)	12	4	48
External Output (EO)	13	5	65
External Inquiries (EQ)	10	4	40
Internal Logical Files (ILF)	5	10	50
External Interface Files (EIF)	8	7	56
			259

1.2.

Metric	Formula	Result
Vocabulary (n)	$n = n1 + n2$	15+25 = 40
Program Length (N)	$N = N_1 + N_2$	80+100 = 180
Program Volume (V)	$V = N \log_2 n$	$180 \log_2 40 = 957.6$
Program Level (L)	$L = V^*/V$	2+5/25+100 = 0.03325
Effort (E)	$V \times D$ or $E = V/L$	= 957.6/0.03325 = 28800

1. 3. It is assumed to be organic Project

KLOC = 10,200 / 1000 = 10.2

Effort $E = 2.4 (KLOC)^{(1.05)}$

$= 2.4 (10.2)^{(1.05)}$

$= 27.49$ PM

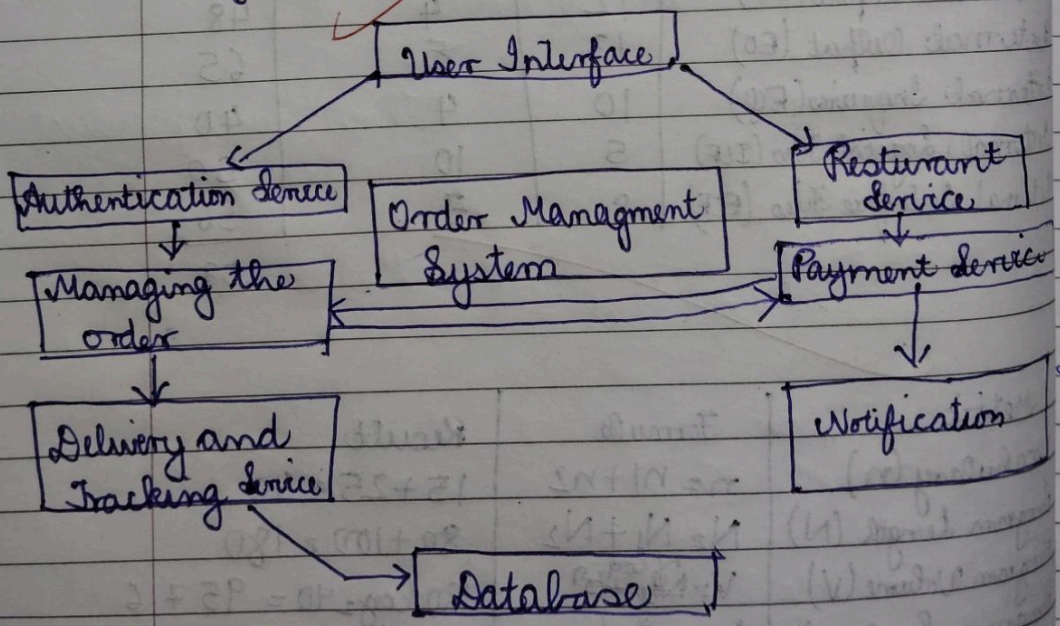
Development Time : $2.5 (E)^{(0.38)}$

$= 2.5 (27.49)^{(0.38)}$

$= 8.806$

2. 1. • Top-down approach : It is a decompositional approach starting with entire system and breaking it down.

• Bottom-up design approach : It is an integrative approach starting with smaller components & integrating them to make large system.



Page No.
Date

2.2. Cohesion: The delivery tracking should only calculate driver location and update status making the module simple. It must be high.

Coupling: Must be low the order service should only communicate with payment service by passing only minimal data.

Abstraction: It must be high defines an abstract of resturant inventory the order service uses this interface to check stock.

2.3. User → Authentication Service → Application Layer → External Services

Module: Menu, Cart, Tracking, Payment, Delivery Service, Resturant Database.

3.1. Feature	Functional Decomposition	Object-Oriented Design
Structure	Organised around functions that operate on data	Organized around objects that encapsulate data & behaviour.
Data Handling	Data is operate from behavior	Data & behaviour are encapsulated.

3.2.

Book	Member	Loan
• Book ID - string • Title: String • Attributes: ISBN	• Member ID: String • Name: String • Borrow Book()	• Loan ID: String due date: Date Loan Date: Date
Methods: checkout() Return book()	Get loan history()	Methods: Calculate fine() Mark as returned()

Date

3.3. Object-oriented design is superior for future scalability & maintainence because it supports inheritance, encapsulation and polymorphism making the system modular, extensible and easier to maintain.

7/10