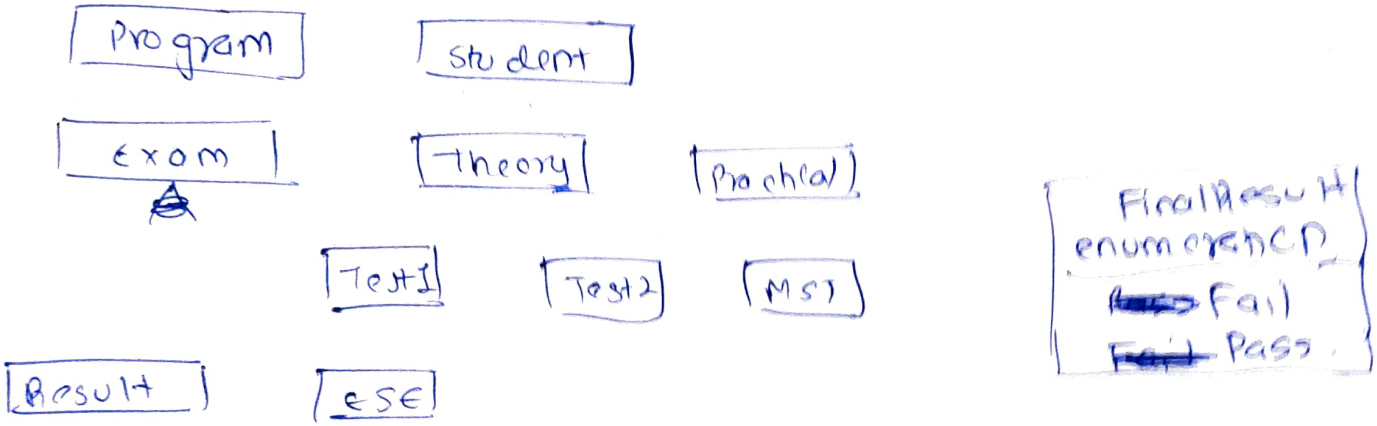
Srushti Shah
19107102
T.Y.B.Tech. C.S.

Software Engineering II - MST

Que. 1)

(i) Identify the classes



(ii) Identify attributes of each class

Program
+ program_id : int
+ program_name : string
+ program_department : string
+ program_instructor : String

Student
+ student_id : int
+ student_name : string
- phoneno : int
- date_of_birth : Date
/ age : int
+ department : string
+ year : int
+ gender : Gender
- email_id : string
- address : string

Test2
+ test_id : int
+ test_date : Date
+ test_time : Time
+ marks_outof : int
- marks_obtained : int

Exam
+ exam_id : int
+ exam_type : String
+ program_id : int

Test 2
+ test_id : int
+ test_date : Date
+ test_time : Time
+ marks_out_of : int
- marks_obtained : int

Theory
+ exam_id : int
+ program_id : int
+ marks_outof : int
- marks_obtained : int
+ result_theory : FinalResult

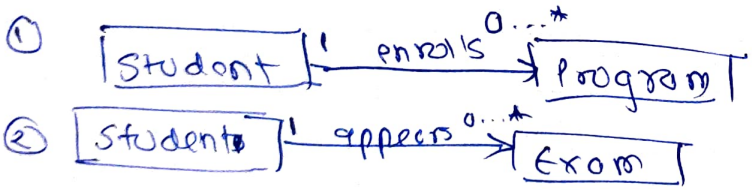
Practical
+ exam_id : int
+ exam_date : Date
+ exam_time : Time
+ marks_outof : int
- marks_obtained : int
+ result_prac : FinalResult

MST
+ mst_id : int
+ mst_subject : String
+ mstDate : Date
+ mstTime : Time
+ mstmarks : int
- mstobtained : int

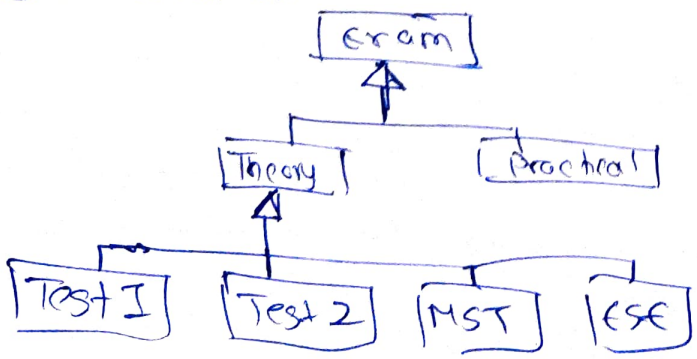
ESE
+ ese_id : int
+ ese_subject : String
+ eseDate : Date
+ eseTime : Time
+ esemarks : int
- esobtaind : int

Result
+ student_id : int
+ result_id : int
- total_prac : int
- total_theory : int
- total_obtained : int
- total_outof : int
+ finalResult : boolean
+ CGPA : float.
+ ~~resultprac : finalResult~~

(iii) Identify associations.



(iv) Generalisation



(iv) Identify methods and functions of each class

① Program

Program(program_id)
enrollForprogram()

② Student
getDetails()
updateDetails()
enrolltoProgram()
login()

③ Exam(examid, exam_type)
getDetails()
scheduleExam()
.

④ TestI
TestI(int marks_outof = 10);
scheduleTest();
obtainmarks(): int

⑤ Test 2(marks_outof = 10)
scheduleTest()
obtainMarks()

⑥ MST
MST(int marks outof = 20);
scheduleTest()
obtainmarks(): int

⑦ ESE(int marks_outof = 60);
scheduleTest()
obtainmarks(): int

⑧ Result
Result(maths_outof = 200)
obtainTotal(prac_marks, theory): int
calculateCGPA(): float
giveFinalresult(): FinalResult

⑨ Practical(int marks_outof = 100);
scheduleTest();
obtainmarks(): int
obtainresult(): FinalResult

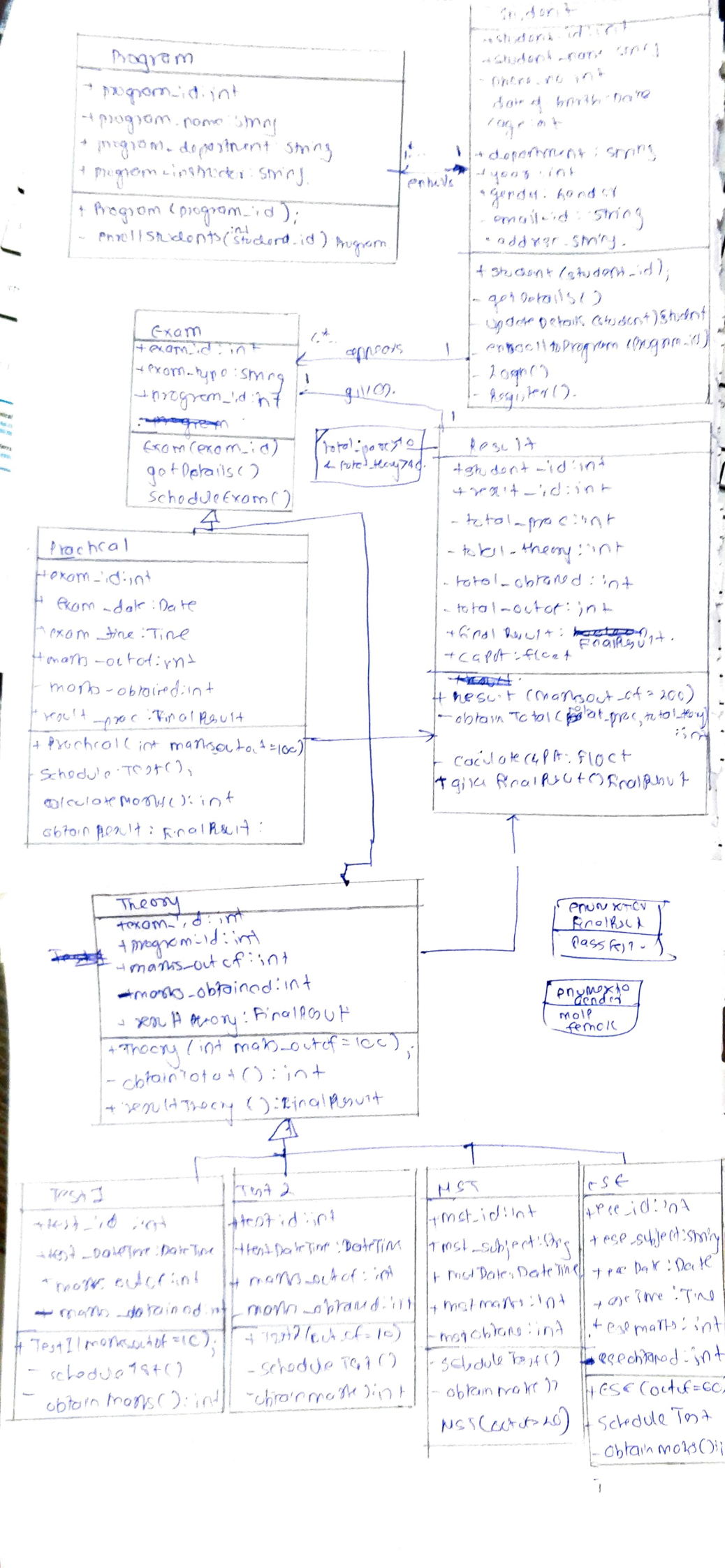
⑩ Theory(int marks_outof = 100);
obtainTotal(): int
resultTheory(): FinalResult

Constraints.

content Practical inv: self.marks obtained > 40

content Theory inv: self.marks obtained > 40

content Result inv: finalResult = PASS and totalprac > 40 and total theory > 40


Program
+ program_id: int
+ program_name: string
+ program_department: string
+ program_instructor: string
+ Program (program_id);
- enrollStudents(int student_id) Program
enrolls
+ student_id
+ student_id: int
+ student_name: string
- phone_no: int
date of birth: Date
+ department: string
+ year: int
+ gender: Gender
- email_id: string
+ address: string
+ Student (student_id);
- getDetails()
- updateDetails (student) Student
- enrollToProgram (program_id)
- login()
- Register()
Exam
+ exam_id: int
+ exam_type: string
+ program_id: int
Exam (exam_id)
getDetails()
scheduleExam()
appears
gives
total: parent
Result
+ student_id: int
+ result_id: int
- total_prac: int
- total_theory: int
- total_obtained: int
- total_outof: int
+ final Result: FinalResult
+ CGPA: float
+ Result (marksout_of = 200)
- obtainTotal (total_prac, total_theory): int
- caculateCGPA: float
+ giveFinalResult() FinalResult
Practical
+ exam_id: int
+ exam_date: Date
+ exam_time: Time
+ marks_outof: int
- marks_obtained: int
+ result_prac: FinalResult
+ Practical (int marksoutof=100)
- Schedule Test()
calculateMarks(): int
obtainResult: FinalResult
Theory
+ exam_id: int
+ program_id: int
+ marks_outof: int
+ marks_obtained: int
+ result Theory: FinalResult
+ Theory (int mark_outof=100);
- obtainTotal(): int
+ resultTheory (): FinalResult
enum Result
FinalResult
Pass/Fail
enum Gender
male
female
Test 1
+ test_id: int
+ test_DateTime: DateTime
+ marks_outof: int
+ marks_obtained: int
+ Test1 (marksoutof=10);
- schedule Test()
- obtain marks(): int
Test 2
+ test id: int
+ test DateTime: DateTime
+ marks_outof: int
- marks_obtained: int
+ Test2 (outof=10)
- schedule Test()
- obtain marks(): int
MST
+ mst_id: int
+ mst_subject: String
+ mstDate: DateTime
+ mst marks: int
- mst obtains: int
- schedule Test()
- obtain marks()
MST (outof=40)
ESE
+ ese_id: int
+ ese_subject: string
+ ese Date: Date
+ ese Time: Time
+ ese marks: int
- eseObtained: int
+ ESE (outof=60)
+ schedule Test
- obtain marks(): int

Qust 2: library Management.

(i) Use cases.

User - login
- register
- search book
- issue book (check max limit)
- return Book (check due date)

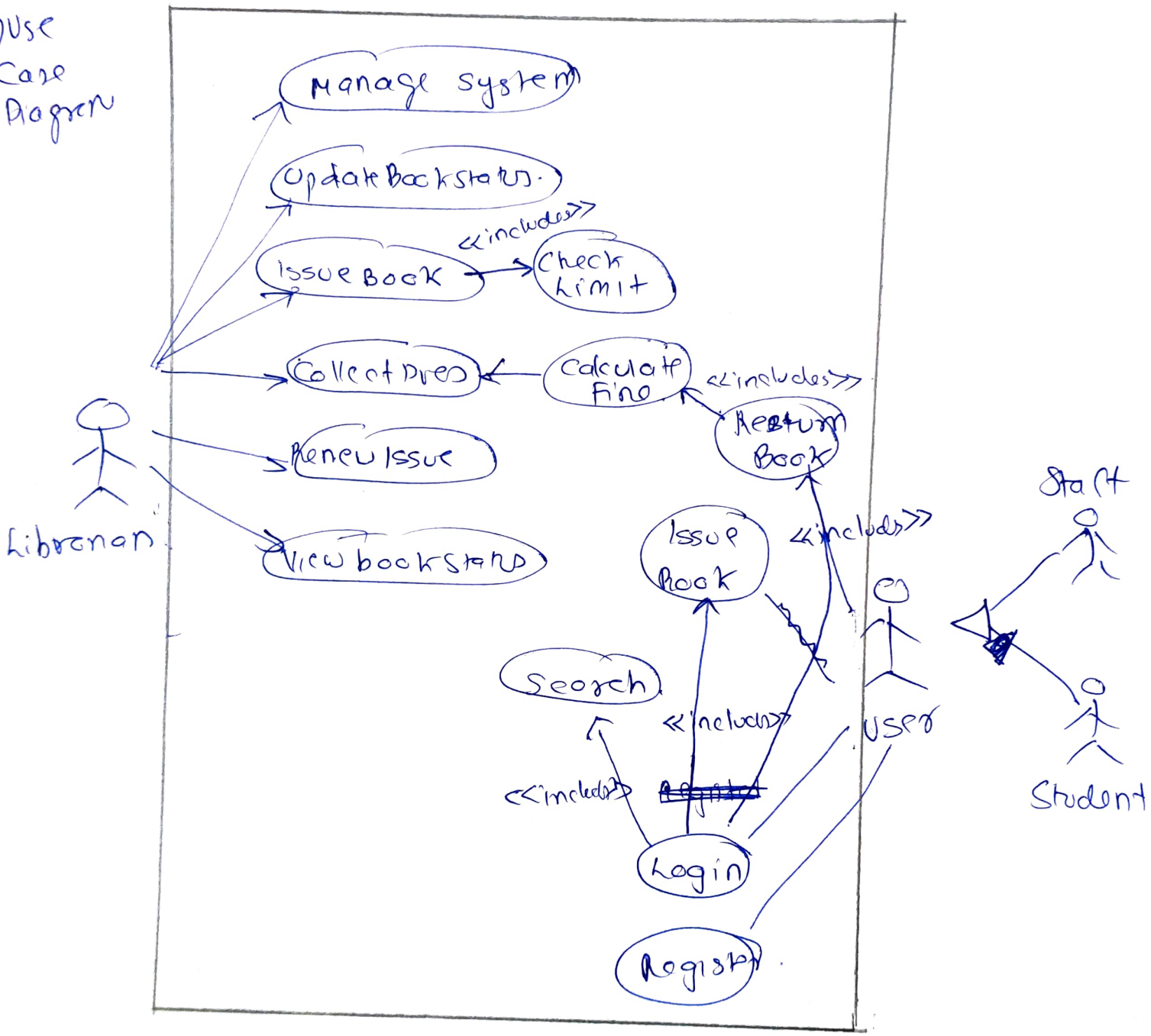
(ii) Actors

User ← Student, staff.
Librarian (Admin)

librarian - manage system.
- Update book status
- view book status
- add new book
- issue book.
- calculate fine.

(iii) use case template

(i) Use Case Diagram



Primary Actor is the User that facilitates the use case.

Secondary Actor is the Librarian that helps complete the use case.

(iii) use case template

Use Case : Issue Book

Description : user requests to issue book from an available no of books

Actor : user and librarian

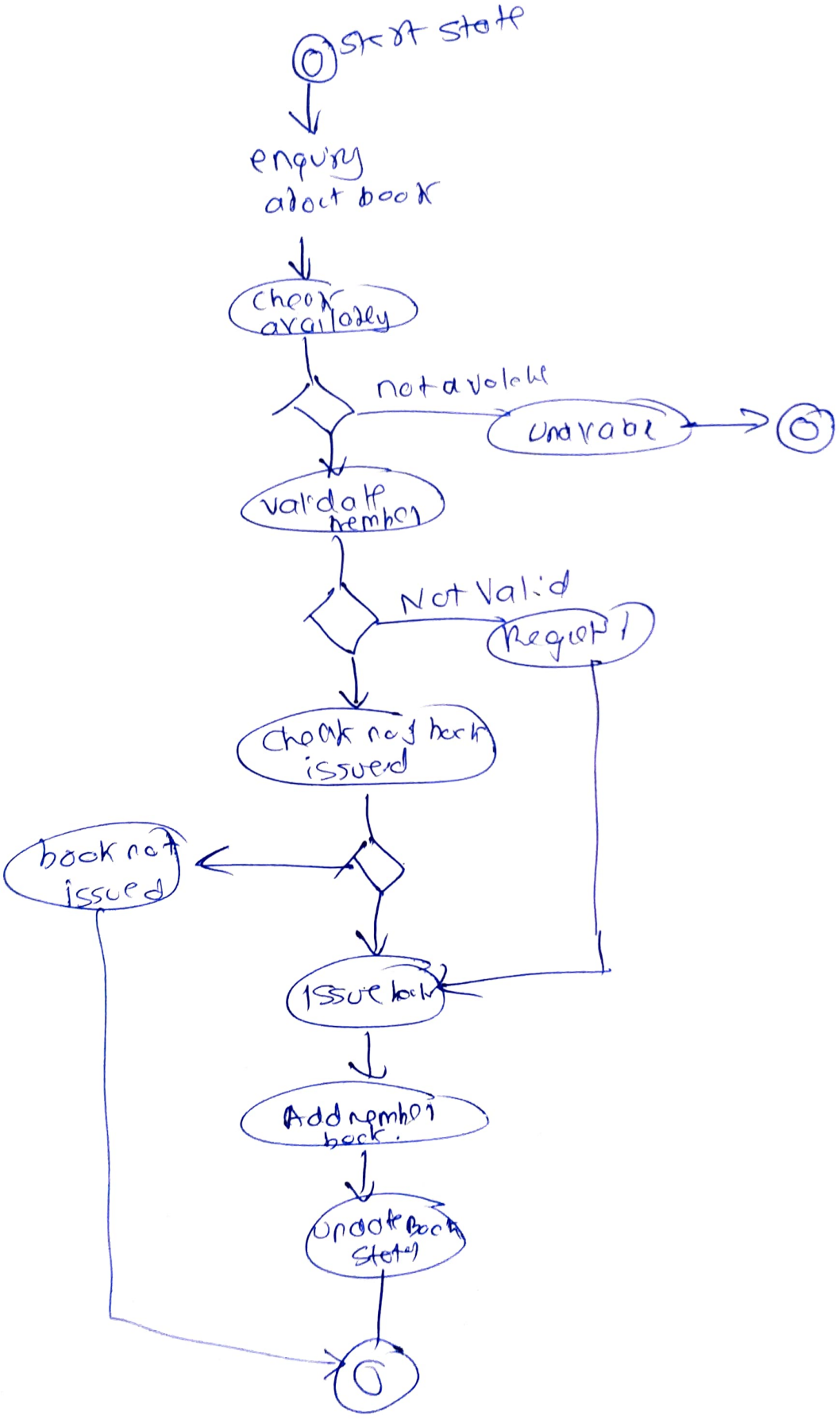
Precondition : The user doesnot have maximum limit of books
- The book is available.

Post condition : The book is issued and book status is updated.

Flow :
1. User searches for a book and requests for update
2. Enter book ID, Author, Book Name.
3. If book ID, author, bookname matches
   then get status of book
   else
   return 'no book'
4. If the status of book = available
   then.
   a. Check if user have 'maximum book issued' already
      if yes : "cannot issue"
   b. else : issue the book
5. update the book status.

Alternate Flow
1. Book not found
2. Maximum limit reached
   a. cannot issue the book
   b. Return a book.

(ii)


Start state
enquiry about book
Check availability
not available
Unavailable
Validate member
Not Valid
Request
Check not book issued
book not issued
Issue book
Add member back.
Update Book Status