Lab Topic: **Library Book Management App**

Theory Topic: **Machine Learning based implementation of Library Management system**

Submitted by:
Palak Singhal 16CO129
Sharanya Kamath 16CO140

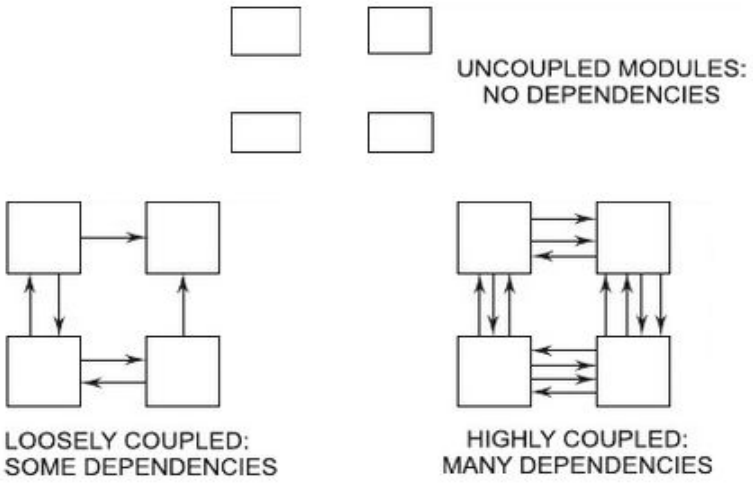
Q1. Coupling and Cohesion.

COUPLING

An indication of the strength of interconnections between program units. The interface complexity is basically determined by the number of types of parameters that are interchanged while invoking the functions of the module.

Highly coupled have program units dependent on each other.
Loosely coupled are made up of units that are independent or almost independent.
When the different modules have no interconnection among them then they are called uncoupled modules.

Coupling is always bad because it prevents the replacement or changes of components independently of the whole.



What are some of the consequences of high coupling?

- Developers / maintenance programmers need to understand potentially the whole system to be able to safely modify a single component.
- Changing requirements that affect the suitability of some component will potentially require wide ranging changes in order to accommodate a more suitable replacement component.
- More thought need to go into choices at the beginning of the lifetime of a software system in order to attempt to predict the long term requirements of the system because changes are more expensive.

Types of Coupling:

Data Coupling.

Two modules are data coupled if they communicate using an elementary data item that is passed as a parameter between the two; for example, an integer, a float, a character, etc. This data item should be problem related and not used for a control purpose.
When a non-global variable is passed to a module, modules are called data coupled. It is the lowest form of a coupling.
In our project, the number of library cards remaining with the user (i.e. the number of books he can issue) is passed from the system module to the display module.

Stamp Coupling.

Two modules are stamp coupled if they communicate using a composite data item, such as a record, structure, object, etc. When a module passes a non-global data structure or an entire structure to another module,they are said to be stamp coupled.

Control Coupling.

Control coupling exists between two modules if data from one module is used to direct the order of instruction execution in another. An example of control coupling is a flag set in one module that is tested in another module.
The sending module must know a great deal about the inner workings of the receiving module.

External Coupling.

It occurs when modules are tied to an environment external to software. External coupling is essential but should be limited to a small number of modules with structures.

Common Coupling.

Two modules are common coupled if they share some global data items (e.g., Global variables). Diagnosing problems in structures with considerable common coupling is time-consuming and difficult.

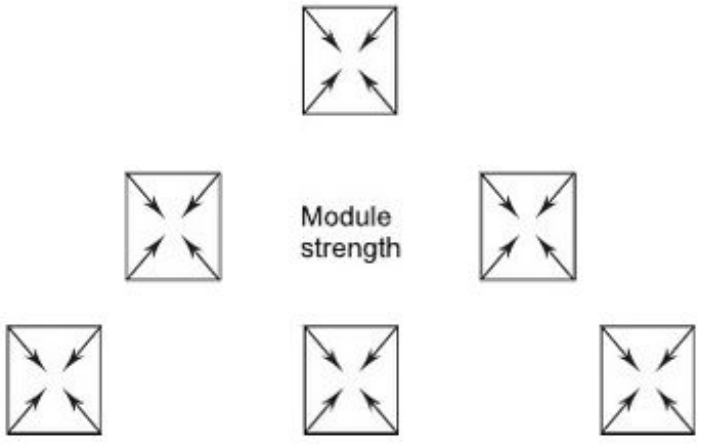
Content Coupling.

Content coupling exists between two modules if theircode is shared; for example, a branch from one module into another module.It is when one module directly refers to the inner workings of another module. Modules are highly interdependent on each other. It is the highest form of coupling. It is also the least desirable coupling as one component actually modifies another and thereby the modified component is completely dependent on the modifying one.

COHESION

Measure of how well module fits together. The cohesion of a component is a measure of the closeness of the relationships between its components.
A component should implement a single logical function or single logical entity. All the parts should contribute to the implementation.

A strongly cohesive module implements functionality that is related to one feature of the solution and requires little or no interaction with other modules. Cohesion may be viewed as the glue that keeps the module together. It is a measure of the mutual officity of the components of a module.



Types of cohesion:

Functional Cohesion.

Functional cohesion is said to exist if different elements of a module cooperate to achieve a single function. In our project, calculation of fine displays functional cohesion.

Sequential Cohesion.

A module is said to possess sequential cohesion if the elements of a module form the parts of a sequence, where the output from one element of the sequence is input to the next.
In our project, the Signup procedure displays sequential cohesion as the information that the user enters is passed on to the verification module, and then to the student info database.

Communicational Cohesion.

A module is said to have communicational cohesion if all the functions of the module refer to or update the same data structure.
In our project, communicational cohesion is depicted between the databases and the system. The system constantly keeps updating the status of students and books in the respective databases.

Procedural Cohesion.

A module is said to possess procedural cohesion if the set of functions of the module are all part of a procedure (algorithm) in which a certain sequence of steps has to be carried out for achieving an objective.
In our theory project, procedural cohesion is seen in the clustering algorithm. Proper order of step are followed in the algorithm which then finally decides what books to show in the user's feed.

Temporal Cohesion.

When a module contains functions that are related by the fact that all the functions must be executed in the same time span, the module is said to exhibit temporal cohesion.

Logical Cohesion.

A module is said to be logically cohesive if all elements of the module perform similar operations

Coincidental Cohesion.

A module is said to have coincidental cohesion if it performs a set of tasks that relate to each other very loosely. In this case, the module contains a random collection of functions. It means that the functions have been put in the module out of pure coincidence without any thought ordesign. It is the worst type of cohesion.

Goal: We want loosely coupled modules with high internal cohesion.

Q2. Conceptual and technical design, Exception handling, Fault tolerance.

Conceptual and Technical Designs

Conceptual Design

• We produce conceptual design that tells the customer exactly what the system will do. (The What of the solution).

First the conceptual design is produced that tells the customer exactly what the system will do. It belongs to the what part of the solution. Once the customer approves it ,it is translated into a much more detailed document i. e. the technical design which allows system builders to understand the actual hardware and software needed to solve the customers problem. It belongs to the how part of the solution.

In our project conceptual design can be used in the following ways:

1.Where will the data come from?

Ans: The data in our project will come from the student database and the book information database. The student database will be responsible for storing the student details. The book database will store all the information regarding the books present in the library.

2.What will happen to the data?

Ans: The data contained in the database will be transferred from the server to the client side which could resolve the query of the user. Any new user details or edits would flow to the database. Also the admin could add more book details in the book info database.

3.What will the system look like to the users?
Ans: The system will be an android application with a proper interface for login/signup, a profile window displaying user details and search options etc.

4.What choices will be given to the user?
Ans: The user will have the choice to register or log in. Also the user can edit personal details, or search for a book with fields like author name, book name,publication etc.

Technical Design

• We produce technical design that allows system builders (developers) to understand the actual hardware and software needed to solve the customer's problem. (The How of the solution).

The technical design describes the hardware configuration, the software needs. Communication interfaces. The input and output of the system, the software architecture, and anything's else that translates their requirements into a solution to the customer problem. That is the technical design description is a technical picture of the system specification .

In our project technical design will be used in the following ways:

1.Hardware components:

The existing Local Area Network (LAN) will be used for collecting data from the users and also for updating the Library Catalogue.
The client-server system that we will use is HTTP client-server.
The server should meet the following requirements:
-> minimum 2GB RAM
-> 2GB of available hard disk space
The client should meet the following requirements:
-> At least 500 MB free space
-> 512 MB of RAM

2. Software components

In a nutshell software components are loosely coupled pieces of software that encapsulate a group of closely coupled pieces of software.

-From the perspective of a software architect, our project has main components such as frontend and various backend services.

-From the perspective of a developer, our project has different components such as libraries and classes.

The major software tools which are going to be used by us for the lab project are:

- Android sdk
- Intellij IDEA (Java programming language)
- Apache server
- Mongo DB for database managemen

EXCEPTION HANDLING

Exception handling is the process of responding to the occurrence, during computation, of *exceptions*.

Exceptions are anomalous or exceptional conditions requiring special processing – often changing the normal flow of program execution. An exception breaks the normal flow of execution and executes a pre-registered *exception handle*.

Approaches to exception handling in software are error checking, which maintains normal program flow with later explicit checks for contingencies reported using special return values or some auxiliary global variable such as C's errno or floating point status flags; or input validation to preemptively filter exceptional cases.

Many failures in critical systems are due to missing or faulty exception handling. They were not tested under the exception conditions.The requirements were not specific about exceptions that had to be tolerated.

Why exception handling is difficult:

- Exception conditions arise from several levels.
- Exception conditions are more difficult to understand than main line requirements.
- Exceptions occur infrequently but require disproportionate effort.

Sources of exceptions:

Operational requirements

(loss of power, communication, thermal control)
In our project, if two users try to issue a book which has only one copy left, there may be need of exception handling.

Implementation detail

(calibration anomalies, actuator states, operator input)
In our project, this type of exception will occur if the user clicks on something on the screen when a process is going on.

Computing environment

(hardware failures, memory errors, executive, middleware)
In our project memory errors could occur as we have two databases- Student info database and Book info database.

Monitoring and self-test

(over-temperature sensors,system performance test)
In our project, if we don't conduct system testing thoroughly, then some exceptional cases may get missed.

Application software

(assertions, violation of timing constraints, mode changes)
In our project, if the admin makes too many modifications to the database at once, the system may take some time to update itself. This may lead to time lag at the user interface.

FAULT TOLERANCE

Software fault tolerance is the ability of computer software to continue its normal operation despite the presence of system or hardware faults.

Fault tolerance is the property that enables a system to continue operating properly in the event of the failure of (or one or more faults within) some of its components. If its operating quality decreases at all, the decrease is proportional to the severity of the failure, as compared to a naively designed system in which even a small failure can cause total breakdown. Fault tolerance is particularly sought after in high-availability or life-critical systems.

The importance of implementing a Fault Tolerance System is about service continuity or maintaining functionality in the event of system failure though operating speed might be affected.Fault Tolerance System is highly needed in most enterprise organization especially for life-critical systems to continue providing service in the event of system fault. You can implement it to a computer hardware, network system, applications etc. For example, hot swapping of a hard drive in a server rack.

Yes, there are limitations of a Fault Tolerance System. Examples are listed below.

1. You can't tolerate what you don't expect
2. 100% Operating level is not guaranteed in the event of system failure.
3. Fault Tolerance System is expensive.

How can faults be tolerated?

- Acceptance test is most effective if it can be calculated in a simple way and if it is based on a criteria that can be derived independently of the program application.
- The existing techniques include

1. timing checks
2. coding checks
3. reversal checks
4. reasonableness checks
5. structural checks

All these checks could be used in our project in following ways:

-> We can keep checking our code at frequent intervals and make changes accordingly.

-> We can use a watchdog timer .Watchdog timers are used to monitor the performance of a system and detect lost or locked out modules.

-> We can also implement system closure which wouldn't allow any unauthorised access to the data. Since its restricted all the changes would be visible and it would be easier to track any fault.

-> Partitioning or creating modules with few or no common dependencies could also be used for fault tolerance.

-> Structural check on the data structures used in our code like linked list used to store student details etc could be implemented with the help of extra pointers etc.

-> Checking overflow or underflow condition like display of random text on querying for a book not present in the database.
