## Tasks and requirements

Disclaimer: The requirements generated by Chat-GPT are based on its understanding of the input provided and the language model's training data. As an AI tool, Chat-GPT may generate contradictory or inconsistent requirements. If any contradictions are found in the requirements, developers should make reasonable assumptions and use the simplest possible approach to implement the functionality. The main goal is to create a functional application that meets the minimum requirements in the given time frame.

**Distributed library service**

As part of our company's initiative to promote education and learning, we have decided to implement a distributed library service. The library will not have a physical location and will rely on book contributions from individual contributors who are in various physical locations.

To ensure that the library has a diverse collection of books, we have decided to enlist the help of our company employees to act as contributors. This way, employees can share their personal book collections, which will be used to build the library's collection.

It is important to note that all books in the library are printed, and the service does not support digital assets. Given the nature of the library service, books will be passed from one contributor to another to maintain the collection. Our aim is to have a wide selection of books that cater to different interests and age groups.

To ensure proper organization and management of the collection, we need to create a system that tracks the books' locations and facilitates the book transfer process. Additionally, we need to establish clear guidelines for contributors on how to contribute books, how to request books, and how to pass books to other contributors.

Our ultimate goal is to create a library service that provides an accessible and diverse selection of books to our employees. This will promote a culture of learning and will encourage personal development within our company.

**Requirements:**

* The system should allow users to create and manage their accounts, including updating their personal information and viewing their book borrowing and book contributing history.
* The system should provide a submission form for contributors to enter information about the new book they want to add to the library's collection.
* The system should allow users to search and browse the library's collection of books by title, author, subject, or keyword.
* The system should provide personalized recommendations for items based on a user's book borrowing history, interests, and preferences.
* The system should enable users to borrow and pass items from the library's collection. Users should be able to place book holds or reserve items that are currently in use.
* The system should send notifications to users when items are available for pickup, when items are due, and when book holds or reserves are ready for pickup.

**Technical stack:** Java 17, Spring Boot (2.7.5, 3), JUnit and other supporting libraries, MySQL

**Step by step task instruction:**

**Task 1**

Implement the foundational structure of the application, including projects for different layers of the application, implementation of the base classes, and unit testing projects.

Requirements:

* Layered architecture for the distributed library service should consist of at least three layers: the presentation layer, the application layer (service layer), and the data layer.
* Implement the presentation layer, which provides the front-end user interface for the distributed library service, using a web frameworks (e.g. Thymeleaf, etc.). The front-end user interface, such as the home page, search page, and book detail page should be implemented using HTML, CSS, JS, etc.
* Write your tests using JUnit, including defining test classes, test methods, and test assertions. Use Mockito to create fake objects and mock dependencies for your tests.
* Use MySQL as database server.
* Use Spring Data or pure providers of JPA to connect to the database and manipulate data.
* Implement all necessary configurations for local run: dockerfile, docker compose with related technologies, README with instructions

**Task 2**

Implement the basic data models and database schema for the distributed library service, such as the Book model and User model, in the data layer. Implement the data access layer, including repositories or data access objects (DAOs), for interacting with the database and providing data to the higher layers. Implement data migrations.

Recommended data structures:

Book table - Contains information about books in the library

* BookID (Primary key)
* Title
* Author
* Publisher
* ISBN
* Publication Date
* Genres
* Number of Pages
* Tags
* ContributorId (Foreign key to User table)
* CurrentKeeperId (Foreign key to User table)

User table - Contains information about library members and contributors

* UserID (Primary key)
* Name
* Email
* Phone
* City
* Address

Loan History table - Contains information about books history

* LoanID (Primary key)
* BookID (Foreign key to Book table)
* UserID (Foreign key to User table)
* Loan Date
* Due Date

Reservation Queue table - Contains information about books that are currently reserved by library members

* ReservationID (Primary key)
* BookID (Foreign key to Book table)
* UserID (Foreign key to User table)
* Reservation Date

Add database migration with predefined data (to run application locally with some predefined data in DB). Can use Spring Boot, Flyway, Liquibase, etc.

**Task 3**

Implement authentication for application.

Requirements:

* The application must provide a way for users to register for an account, including entering their email address, username, and password.
* Once registered, users must be able to log in to their account using their username and password.
* The application must integrate with the selected third-party authentication service (e.g. Google, Facebook) to allow users to log in with their existing accounts.
* The "/login" path should lead to a login page where users can enter their login credentials.

**Task 4**

Implement a mechanism for adding and editing a book

Requirements:

* Users should be able to add new books to the library by filling out a form with book information such as title, author, description, and tags.
* The system should keep track of who added or edited the book, as well as the date and time of the change.
* Users who added books should be marked as contributor by default.
* The system should validate the input data before saving it to the database. For example, the title and author fields should not be empty, and the ISBN field (if present) should be valid.
* The system should allow users to edit existing book information, such as title, author, description, and tags.
* The system should handle any errors or exceptions that occur during the adding or editing process, and display an appropriate error message to the user.
* The book page should be accessible through the URL path "/book/{id}", where id – internal id in database

**Task 5**

Implement a dashboard with available books and the ability to filter them.

Requirements:

* The dashboard should display a list of available books in the library.
* The list of books should be filterable by various parameters such as book title, author, genre, publication date, etc.
* The filtering should be done via a user-friendly interface that allows users to select multiple filter criteria at once.
* The dashboard should display a brief summary of each book, including its title, author, description, cover image, and any other relevant information.
* The dashboard should be accessible through the URL path "/dashboard"

**Task 6**

Implement a user profile page that displays which books the user contributed and borrowed to the library and when.

Requirements:

* The application should provide a user profile page where users can view their activity in the library.
* The user profile page should display the list of books contributed to the library by the user, along with their respective dates.
* The user profile page should display the list of books borrowed by the user, along with their respective dates.
* The page should be accessible by permanent link.
* The user page should be accessible through the URL path "/user/{id}"

**Task 7**

Implement a mechanism for transferring a book from one user to another.

Requirements:

* Users should be able to initiate a book transfer request by selecting the book they want to transfer and specifying the recipient's email or username.
* The recipient should receive a notification about the transfer request and have the option to accept or reject it.
* If the recipient accepts the transfer request, the book should be removed from the sender's account and added to the recipient's account.
* If the recipient rejects the transfer request or does not respond within a certain timeframe, the transfer should be canceled, and the book should remain with the sender.
* Both the sender and recipient should be able to view the status of the transfer request and any updates or changes made to it.

**Task 8**

Implement a page with the history of book movements between users.

Requirements:

* Create a page that displays the history of book movements between users.
* The page should include information such as the book title, the borrower's name, and the date of borrowing
* Data on the page should be sorted by the date of borrowing.

**Task 9**

Implement unit tests with coverage between 50% and 60%.

Requirements:

* Develop a comprehensive suite of unit tests covering all the major functionalities of the system.
* The unit tests should be written using a framework such as JUnit and should have at least 50% code coverage.

**Task 10**

Implement a recommendation system that shows similar books using the N Nearest Neighbors algorithm.

Requirements:

* Recommendations block should be present on the book page.
* Use the N Nearest Neighbors algorithm to recommend similar books based on user behavior and book attributes.
* The system should take into account factors such as genre, author, publication date, and user behavior (books read).

**Expectations**

The main goal of the hackathon is to measure the time it takes for developers to complete regular tasks. To achieve this goal, we have several requirements for participants:

* We would like to receive the number of tasks implemented during the first 8 hours of work.
* We want to measure the execution time of tasks, so please accompany the execution of each task with two separate commits: one named TASK\_N\_START and one named TASK\_N\_FINISH, where N is the task number. You can create an empty start commit using --allow-empty flag in the git command line. For example:

`git commit --allow-empty -m " TASK\_2\_START: Start working on data model"`

* Also, please commit the average time to **Readme** file in the root of the repository. README file template will be provided for all participants.

Suggested example of README:

…... // might be developer notes

Spent time

Task 1: 2 hours

Task 2: 2 hours

….

Solution source code should be provided via a private repository at GitLab, GitHub, or BitBucket. Please, provide access to repo for your expert (see shared excel doc) and ‘vadym\_vlasenko@epam.com’.

**Nice to have:**

* Implement CI/CD scripts for any tool (e.g. GitHub actions, GitLab CI, etc.)
* Deploy application to Cloud and provide an access