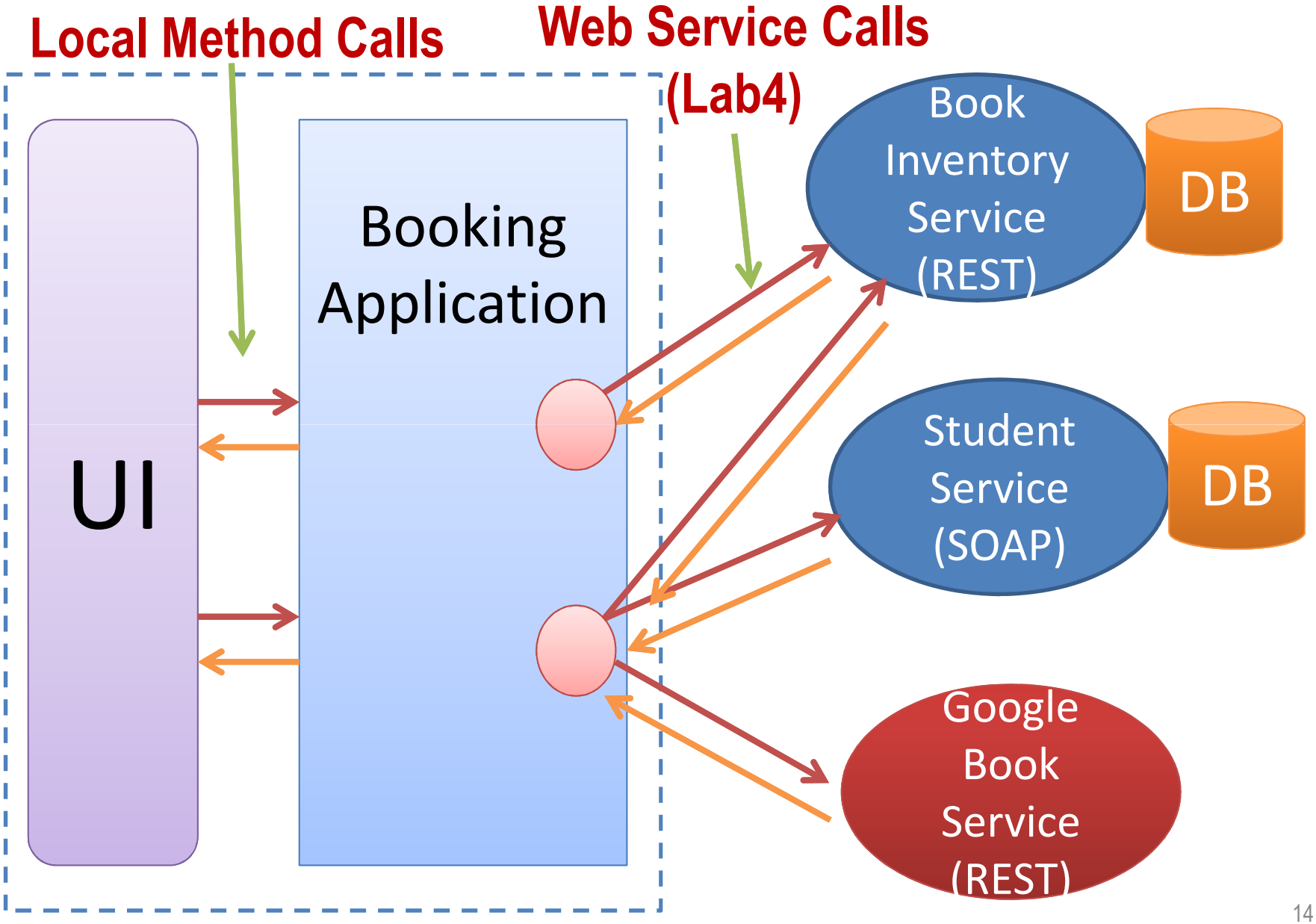
Overview


Local Method Calls
Web Service Calls
(Lab4)
UI
Booking
Application
Book
Inventory
Service
(REST)
DB
Student
Service
(SOAP)
DB
Google
Book
Service
(REST)

A. The business scenario and functional specification

A university library, *UniLib*, wants to establish ***a booking system*** that allows students to borrow books it already holds, or request the ordering of book(s) it does not already have in its collection. The booking system is not a closed system. Instead, it will be integrated with external book information providers in order to streamline the process of ordering new books.

A.1. End-user functionalities

After analysing requirements of the booking system, it has been decided that the *UniLib* booking system will provide students with the following functionalities:

1. View all books in the library stock

A student can request to view all books currently possessed by the library. The system will provide the following book detail:

- Book ID
- Title
- List of author(s)
- Publisher
- Published date
- ISBNs (both 10 and 13 digit formats)
- Status (whether the book is *"available"*, *"on loan"*, or *"back order"*)

Note: We will assume at this stage that library only keeps one copy of each book

2. Borrow a book

To borrow a book, a student needs to provide the following detail:

- Student ID
- PIN number
- Book ID

The submission of this detail will trigger the internal *book borrowing process* (see section A.2), which after its execution will return a response to the student whether the request was successful or not.

3. Request a new book

To request the library to purchase a new book, a student needs to provide the following detail:

- Student ID
- PIN number
- ISBN of the requested book in either 10 or 13-digit format

The submission of this detail will trigger another internal *book ordering process* which after its execution will return a response to the student indicating whether the request was successful or not.

4. View student records

A student is able to query her borrowing/requesting record with the library. The system will provide information related to what books are on loan and what books are being requested by the student for purchasing. Students need to be authenticated before being able to view this information.

A.2. Internal business processes

1. Book borrowing process

i. The student will be first authenticated based on the provided student ID and PIN number. If the authentication fails, the request is rejected.
ii. The requested book ID is then validated. If the ID is incorrect or the book is currently not available, a proper error message is returned to the student.
iii. Otherwise, the system will lend the book to the student and update the status of the book as well as the student's borrowing record.

2. Book ordering process

i. Similar to the previous process, the student will be first authenticated. If the authentication fails, the request is rejected.
ii. The system will only consider a request if
    a. the book is NOT already held by the library, and
    b. it has NOT already been ordered by someone else (i.e. there is no 'back order' already placed for the book).
iii. The requested ISBN is then validated. If the ISBN is incorrect (i.e. the number is not associated with any book available in the market), a proper error message is returned to the student.
iv. If the ISBN is valid the system will retrieve necessary information related to the book and assess the book quality (see section A.3). If the assessment result is negative, the system will notify the student of the justification.
v. Otherwise, the system will do the following
    a. update its database with the new book
    b. update the student's requesting record
    c. notify the purchasing department for executing the purchase order (at this stage we will assume that all requests that meet basic quality criteria are automatically approved for purchase.)
    d. notify the student that a purchase order has been placed

A.3. External business partner

UniLib has found that Google Books (http://books.google.com.au/) is a reliable source of book information that can be used to assess whether it should order a book or not. Therefore, *UniLib* has decided to partner with Google Books in order to streamline its book ordering process. In particular, *UniLib* relies on Google Books to perform the following two assessments:

1. ISBN validation

If the provided ISBN number is not associated with any book in the Google Books database, *UniLib* will regard it as *incorrect.*

2. Book quality assessment

A book needs to satisfy both of the following requirements in order to be regarded as *positive* for being purchased:

i. The average rating from readers is at least 3 out of 5. For example, a book with an "averageRating" of 3.5 according to Google Books is acceptable, but a book with this field missing (meaning "averageRating" being 0.0) is not.

ii. The book is available for sale in Australia. For example, if a book's saleInfo is as follows

```
"saleInfo": {
 "country": "AU",
 "saleability": "NOT_FOR_SALE",
 "isEbook": false
}
```

the book is not available in Australia.

B. Application architecture

UniLib wants to develop the booking system as an SOA solution to maximize the flexibility and reusability while shortening time-to-market (i.e. development time) of the system. As a result, the booking system is designed to be composed of the following four components:

1. Book inventory service

This is a Web service that allows its clients to access and manipulate *UniLib*'s book database. Information in the book database, such as book detail or student borrowing/requesting records, is exposed as resources that support CRUD operations from service clients. **As such, the book inventory service is designed to be a REST Web service that exchanges XML data** (for example, see Lab 2)**.** By wrapping the access and manipulation of the book database as a Web service, *UniLib* not only makes the booking system more modular but also opens the opportunity for the service to be used by other applications within the same university (e.g. purchase order application run by the purchase department).

2. Student service

Similar to the book inventory service, this service provides a Web service interface for validating student details by referencing the student database. **This is a SOAP Web service** (for example, see Lab 3).

3. Google Books service

As discussed in section A.3, this is an external service for validating ISBN numbers and assessing book quality. Google Books API (https://developers.google.com/books/docs/v1/using) is the Web service interface to access this service. **This is a REST Web service that exchanges JSON data.** To understand and use this service, see the "Resources" section below and a Lab 2 example.

4. Booking application

This is the end user-facing application. It provides end-user functionalities (see section A.1) by composing the above mentioned services. As such, this application plays the role of the service client with respect to all the three Web services discussed above.

C. Databases

The *UniLib* booking system requires the use of two ***separate*** databases (***external persistent*** data stores): book database and student database. These databases are referenced by the *Book inventory service* and the *Student service, **respectively***. There are many different ways to implement databases, such as relational databases, XML files, CSV files, Microsoft Excel sheets, Google spread sheets, text files, etc. For this assignment, you can choose one that you are most familiar with.

C.1. Book database

This database has two data sources: book data source and student borrowing record data source.

1. Book data source

This data source stores the following information for each book:

- Book ID,
- Book title,
- List of authors
- ISBN in both 10-digit and 13-digit format
- Publisher
- Published date
- Status of the book that might be one of *{"available", "on loan", "back order"}*

2. Student borrowing record data source

This data source stores the following borrowing information for each student:

- Student ID
- List of books that the student is borrowing
- List of books that the student has successfully requested

Note that the information in this data source needs to be consistent and synced with the information in the book data source as well as with the information in the student data source (discussed in section C.2 below).

C.2. Student database

This database has one data store: student data source.

1. Student data source

This data source stores the following information for each student:

- Student ID
- Full name
- 4-digit secret PIN number

D. Implementation, progress and submission requirements

You are required to implement the Booking application following the specification discussed in sections A-C. The use of GUI (Graphical User Interface) in developing this application is highly recommended but not mandatory. CLI (Command Line Interface) is acceptable. *Regardless of what technique you use, the Booking application needs to be flexible enough to allow students to carry out different transactions (e.g. view all books, borrow a book, view borrowing status, request a book, view requesting status).*

D.1. Coding requirements

1. Code should follow a consistent coding style, and be appropriately formatted and commented.
2. You should be able to set up and demonstrate your application (using the lab machines or your laptop), including the deployment of the services you developed and the run-time access to the external service.
3. Your application should provide appropriate error handling in the event that invalid data are supplied or if a Web service is unavailable.

D.4. Resources

Information on how to use Google Books API is available at https://developers.google.com/books/docs/v1/using. You need to have a Google account and acquire an API key in order to use the API. Registration for an API key is free of charge (https://developers.google.com/books/docs/v1/using#APIKey).

Google Books API returns data in JSON format. You are required to extract necessary information from this JSON data to implement the business logic of this application. For this purpose, you may follow the way how we handled service interaction withAxis2 JSON support (Lab 2), or use another non-Axis2 platform/ library for accessing JSON-based REST services that you are more familiar with.

1. Search book by ISBN

A book can be searched from its ISBN using a GET request to the following endpoint:
https://www.googleapis.com/books/v1/volumes?q=isbn:*{isbn_number}*&key=*{yourAPIKey}*

Note that in this URL, you need to replace *{isbn_number}* with the ISBN number you wish to search for and *{yourAPIKey}* with your own API key obtained from Google.

2. View book detail

Book detail can be retrieved using a GET request to the following endpoint:
https://www.googleapis.com/books/v1/volumes/*{volumeID}*?key=*{yourAPIKey}*

Note that in this URL, you need to replace *{volumeID}* with the ID of the book you wish to view and *{yourAPIKey}* with your own API key obtained from Google.

To simplify your implementation, you are recommended to use this volumeID as the unique ID for books in your book database.