1)Fetch All Lists (GET /api/lists)

```
public List<ListEntity> getAllLists() {
   return listRepository.findAll();
}
```

Time Complexity: O(N)

Retrieves all lists from the database.

Requires scanning N rows in the list_entity table.

Performance decreases as N grows.

2)Fetch a Single List by ID (GET /api/lists/{id})

```
public Optional<ListEntity> getListById(Long id) {
   return listRepository.findById(id);
}
```

Time Complexity: O(1)

Primary Key Lookup, which is fast due to indexing.

Always takes constant time, regardless of database size.

3)Save a New List (POST /api/lists)

```
public ListEntity saveList(ListEntity list) {
   return listRepository.save(list);
}
```

Time Complexity: O(1)

Inserts a new row into the database.

Direct insert operations run in constant time.

4)Delete a List (DELETE /api/lists/{id})

```
public void deleteList(Long id) {
    listRepository.deleteById(id);
}
```

Time Complexity: O(1)

Deletes a row using primary key indexing.

Runs in constant time, irrespective of dataset size.

5)Update a List (PUT /api/lists/{id})

```
public ListEntity updateList(Long id, String newName, List<ListItem> newItems) {
    ListEntity list = listRepository.findById(id)
        .orElseThrow(() -> new ResponseStatusException(HttpStatus.NOT_FOUND, "List not found"));

    list.setName(newName);
    list.setItems(newItems);
    return listRepository.save(list);
}
```

Time Complexity: O(M)

Fetching the list by ID → O(1)

Updating list name → O(1)

Updating M list items → O(M) (depends on number of items being updated)

Overall Complexity: O(M), where M is the number of items being modified.