**Challenge Application**

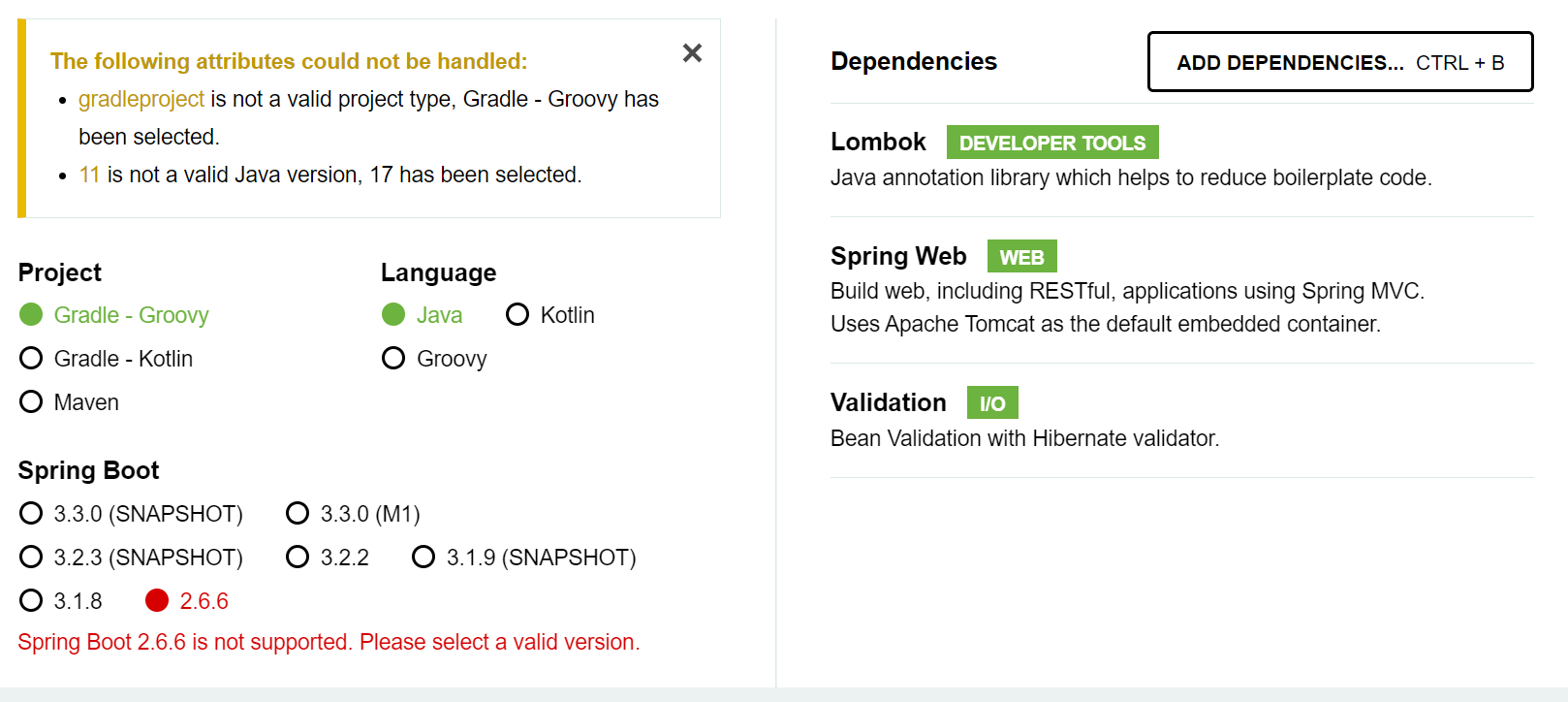
As per your suggestion link is for spring io https://start.spring.io/#!type=gradleproject&

language=java&platformVersion=2.6.6&packaging=jar&jvmVersion=11&groupId=com.d

ws&artifactId=challenge&name=challenge&description=DWS%20Dev%20Challenge&package

Name=com.dws.challenge&dependencies=lombok,web,validation But Spring Boot 2.6.6 is not supported so I chose latest 3.2.2 stable version with JDK 17.

See Error for 2.6.6 version.



Improvements needed in future:

Some feature improvements for the application could include:

Implement Kafka:

To send notification messages implement kafka.

Resource Bundle:

Read text from resource bundles then can remove hard coded text from classes.

Error Handling Improvement:

Implement more detailed error handling for various scenarios, such as handling unexpected errors gracefully and providing clearer error messages to the users.

Logging:

Introduce logging statements to track important events, such as account creations, transfers, and error conditions. This can be useful for debugging and auditing purposes.

Validation Enhancements:

Add additional validations to ensure data integrity and security, such as validating account IDs and transfer amounts against predefined criteria.

Performance Optimization:

Evaluate the performance of the repository methods and identify potential bottlenecks. Optimize data access operations to improve overall application performance.

Testing:

Enhance unit tests to cover more edge cases and scenarios, including negative test cases and boundary conditions. This can help ensure robustness and reliability of the repository implementation.

Concurrency Handling:

Implement more advanced concurrency control mechanisms to handle concurrent access to the repository data more efficiently. This could involve using advanced locking strategies or leveraging database transactions if applicable.

Transaction Management:

If the application involves multiple repository operations that need to be executed as a single transaction, consider implementing transaction management to ensure data consistency and integrity.

Documentation:

Improve code documentation, including method-level comments and class-level descriptions, to make the codebase more understandable and maintainable for other developers.

Code Refactoring:

Review the existing codebase for opportunities to refactor and improve code structure, readability, and maintainability. This could involve breaking down complex methods into smaller, more manageable components or applying design patterns where appropriate.

Data Persistence:

Consider integrating with a persistent data store (e.g., a relational database or NoSQL database) to store account information persistently. This can provide durability and scalability benefits over an in-memory data store.

**What implemented/Done High Level Information:**

* Used latest Spring Boot 3.2.2 and JDK 17
* Implemented validation logic to support in 3.2.2
* Implemented Validation and transfer amount logic (Added API Endpoint in Controller class)
* Implemented swagger integration with Application. Can see on swagger and API information in the URL <http://localhost:18080/swagger-ui/index.html> after up application.
* Implemented Concurrency – the balance change should be thread safe.
* Unit testing – the code covered from both positive, negative, and concurrency perspective with postman (Attached postman collection in application resources/files folder).
* Provided README file in project.
* Done concurrency testing through Postman Collection (with option Run collection and we have option How many iteration)
* Provided test cases.

Unit Testing:

Provided postman collection in Application Code (in resources/files folder). Most of the unit testing covered.

JFI see the below pic.

A screenshot of a computer

Description automatically generated

Swagger:

After Up the application using this url http://localhost:18080/swagger-ui/index.html can see API Docu. The below are the sample pics.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Test Cases:

Implemented positive and negative test cases.