aChecklist details and response : -

1. The numbers of the estimated hours and the actual hours. You can optionally write a short paragraph with details and explanations.

The estimated effort for this assignment was approximately 4 hours, and the actual development time closely matched this estimate. During the process, I allocated time for designing the architecture, implementing the Angular frontend with Tailwind CSS for a responsive and modern UI, and integrating it with a Spring Boot backend that leverages MySQL for data persistence. The task also included configuring robust error handling, form validations, and ensuring CORS and timezone considerations were addressed efficiently. Overall, the streamlined approach and careful planning allowed for the project to be completed within the estimated timeframe while meeting all functional and technical requirements.

2.Reasons for choosing your tech stack for this task. You can write a short paragraph to justify your choice among other alternatives.

For this task, I selected Angular for the frontend because it offers a robust, component-based architecture with excellent support for reactive forms and built-in HTTP communication, streamlining complex form validations and state management. The latest Angular standalone components reduce boilerplate code, making the development process more efficient. On the backend, I chose Spring Boot for its rapid application development capabilities, clear separation of concerns, and out-of-the-box support for building RESTful APIs. Its seamless integration with Spring Data JPA simplifies database interactions with MySQL, a mature, stable, and widely supported relational database. This combination of technologies not only ensures maintainability and scalability but also aligns with industry standards for building modern web applications efficiently.

3. Comments:

a. The assumptions you made during implementation.

Few assumptions made during implementation are :

- **Validation & Business Rules:**
  The implementation assumes that the receipt date must not be a future date, and the reimbursement amount must be greater than 0. Both validations are handled on the client-side using Angular reactive forms, along with native HTML input restrictions (e.g., setting max on the date picker).

- **File Handling:**
  I assumed that the receipt file can be stored as a BLOB in the database and that the backend would accept files via multipart form-data without additional file storage requirements (e.g., file system or cloud storage).
- **Error Handling:**
  It was assumed that a basic level of error handling, both on the frontend (displaying user-friendly error messages) and on the backend (using global exception handlers and logging), would be sufficient for this exercise.

b. The problems you came across, and how you solved them.

The problems I encountered and their solutions are as follows :

- **CORS Issues:**
  When connecting the Angular frontend (running on port 4200) with the Spring Boot backend (running on port 9020), CORS errors occurred. I resolved this by either applying the @CrossOrigin annotation on the backend controller and/or configuring a global CORS policy, ensuring that the requests from the frontend are allowed.
- **Timezone Discrepancy in Date Handling:**
  The date picker initially showed tomorrow's date as a valid option because the JavaScript toISOString() method returns the date in UTC, causing a mismatch in Eastern Standard Time (EST). I solved this by adjusting the date calculation using the timezone offset, ensuring the correct local date is set as the maximum allowable date.
- **File Upload Validations:**
  Ensuring that the file upload field accepted valid files and integrated properly into the form submission process required testing that the file was appended correctly to the FormData object on the frontend. Enhancements such as using the accept attribute were also considered to limit file types.
- **HTTP Client Configuration:**
  While configuring the Angular application with standalone components, I encountered an injector error for HttpClient. This was fixed by providing the correct module (using provideHttpClient() in Angular 16) at the application level.

c. The highlights in your code or coding practice that you want us to notice. d. Anything else.

- **Robust Form Validation:**
  The implementation combines both built-in Angular validators and custom validators (e.g., preventing future dates) to ensure data integrity before submission. This proactive validation minimizes errors and reduces backend load.
- **Use of Standalone Components:**
  By leveraging Angular's standalone component structure, the code is modular, cleaner, and easier to maintain. The integration of necessary modules (like ReactiveFormsModule and HttpClientModule) directly into the component's configuration improves clarity.
- **Comprehensive Error Handling:**
  On the backend, the use of global exception handling and specific error responses (such as for database exceptions) ensures that any issues are logged and communicated clearly, which is crucial for production-level applications.
- **Adoption of Industry Best Practices:**
  The overall design separates concerns by clearly dividing responsibilities between the presentation layer (Angular), business logic (Spring Boot service layer), and data access (Spring Data JPA repository). This separation leads to high maintainability and testability.