Traceability Document for Investment Management System

Author: [Your Name]
Date: [Current Date]
Version: 1.0
Logo: [Optional Logo Here]

Table of Contents

Objectives and Scope

The Investment Management System aims to provide a robust backend solution for managing investments and associated persons. The system will facilitate the creation, retrieval, updating, and deletion of investment records and person profiles, ensuring data integrity and security.

Functional Requirements

Endpoint	Method	Description
/api/investments	POST	Create a new investment.
/api/investments/person/{personId}	GET	Retrieve investments by person ID.
/api/investments/person/{personId}/total	GET	Calculate total invested amount by person.
/api/investments/{id}	PUT	Update an existing investment.
/api/investments/{id}	DELETE	Delete an investment.
/api/persons	POST	Create a new person.
/api/persons	GET	Retrieve all persons.
/api/persons/{id}	GET	Retrieve a person by ID.
/api/persons/{id}	PUT	Update a person by ID.
/api/persons/{id}	DELETE	Delete a person by ID.

HTTP Routes Summary

Route	Method	Controller	Description
/api/investments	POST	InvestmentController	Create a new investment.
/api/investments/person/{personId}	GET	InvestmentController	Get investments by person ID.
/api/investments/person/{personId}/total	GET	InvestmentController	Calculate total invested by person.
/api/investments/{id}	PUT	InvestmentController	Update an existing investment.
/api/investments/{id}	DELETE	InvestmentController	Delete an investment.
/api/persons	POST	PersonController	Create a new person.
/api/persons	GET	PersonController	Get all persons.
/api/persons/{id}	GET	PersonController	Get person by ID.
/api/persons/{id}	PUT	PersonController	Update person by ID.
/api/persons/{id}	DELETE	PersonController	Delete person by ID.

Controller, Service, and DTO Relationship

Controller	Service	DTOs
InvestmentController	InvestmentService	InvestmentDTO
PersonController	PersonService	PersonDTO

Technical Architecture

The system follows the Model-View-Controller (MVC) pattern, separating concerns into distinct layers:

- **Controllers:** Handle HTTP requests and responses.
- **Services:** Contain business logic and interact with repositories.
- **Repositories:** Manage data persistence.
- **Models:** Represent the data structure.
- **DTOs:** Facilitate data transfer between layers.

Architecture Diagram

```
[Client] --> [Controller] --> [Service] --> [Repository] --> [Database]
```

Data Validation Mechanisms

Data validation is implemented using annotations from the `javax.validation.constraints` package in DTOs. For instance, `@NotNull` ensures required fields are not empty, while `@Email` validates email formats.

Error Handling

The system employs a global exception handler to manage errors. Common error responses include:

- **404 Not Found:** When a resource is not found.
- **400 Bad Request:** When input validation fails.

Example response for a 404 error:

```
{
  "status": 404,
  "error": "Not Found",
  "message": "Resource not found"
}
```

System Security

The system uses Spring Security for authentication and authorization. Key features include:

- **Authentication:** Validates user credentials.
- **Authorization:** Controls access to endpoints based on roles.
- **Roles:** Define user permissions, e.g., ADMIN, USER.

Conclusions

The Investment Management System is designed to provide a secure and efficient backend for managing investments and associated persons. Future enhancements may include additional features such as reporting and analytics.

Next Steps:

- Implement unit and integration tests.
- Conduct security audits.
- Gather user feedback for further improvements.