# Technical Architecture Document (TAD) for Struts to Spring Boot Migration

## Table of Contents

# 1. Introduction

1. Introduction

- Purpose

- Scope

# 2. Current State Architecture

2. Current State Architecture

- Technical Flow

- System Diagram

# 3. Future State Architecture

3. Future State Architecture

- New Technology Approach

- Migration Strategy

- System Diagram

# 4. Parallel Run Strategy

4. Parallel Run Strategy

- Implementation Plan

- Parameterization for Flow Selection

- System Diagram

# 5. Conclusion

5. Conclusion

# 6. References

6. References

# 1. Introduction

## 1. Introduction

### 1.1 Purpose

The purpose of this document is to outline the migration of Struts actions to Spring Boot-based REST APIs while ensuring seamless transition with minimal impact on the existing system.

### 1.2 Scope

• Replace Struts actions with Spring Boot equivalents.

• Maintain both Struts and Spring flows during the transition period via configurable cloud properties.

• Ensure smooth API-based interaction between backend and React frontend.

# 2. Current State Architecture

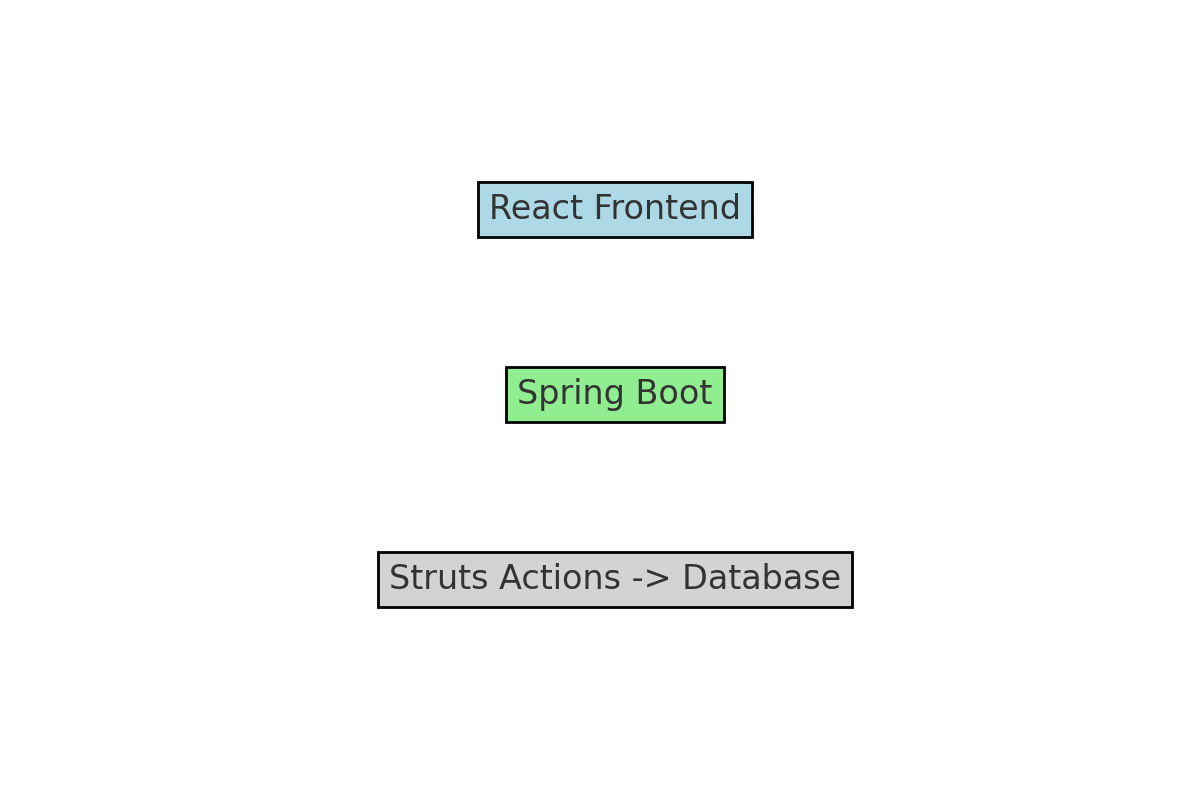
## 2. Current State Architecture

### 2.1 Technical Flow

• The backend is built using Spring Boot, which internally calls Struts actions.

• Struts actions interact with the database.

• The frontend is built using React, which communicates with the backend.



# 3. Future State Architecture

## 3. Future State Architecture

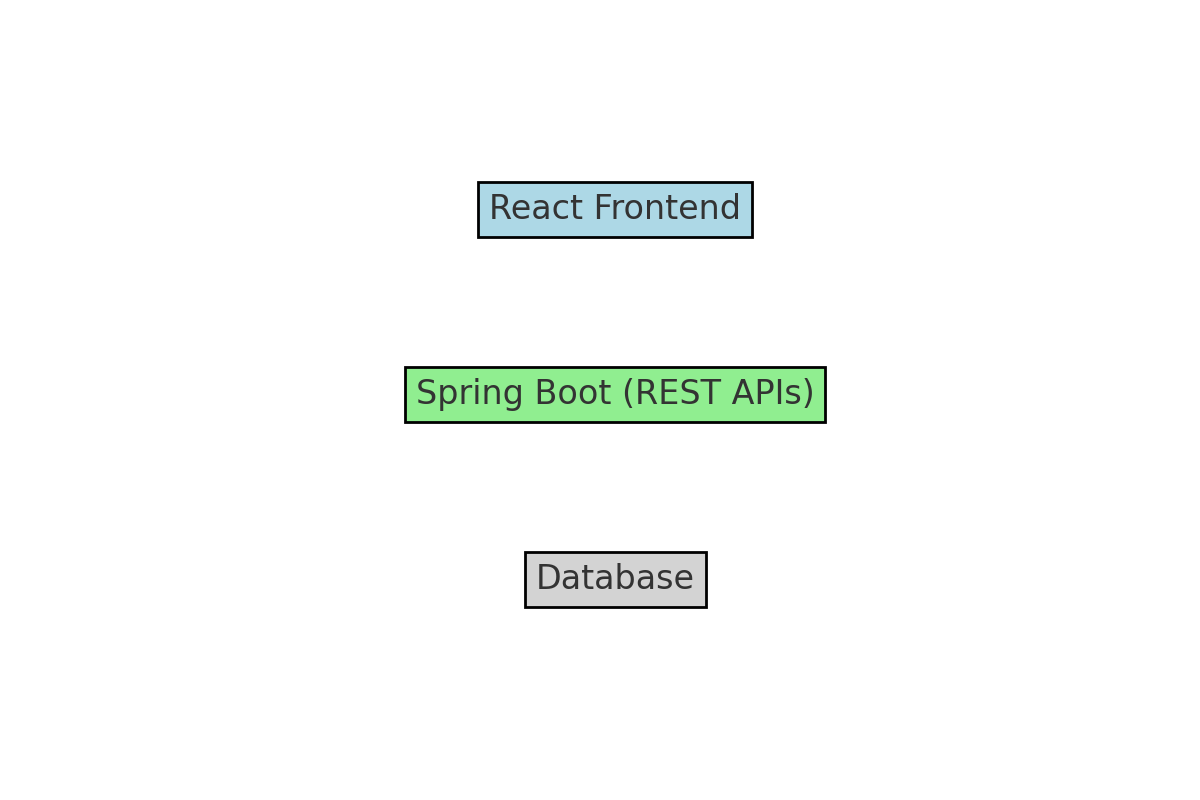
### 3.1 New Technology Approach

• Remove Struts actions and replace them with Spring Boot REST controllers.

• Ensure that all backend calls are API-driven.

• Keep the frontend as React, interacting with the new Spring Boot APIs.

• Database interactions will remain unchanged.



# 4. Parallel Run Strategy

## 4. Parallel Run Strategy

### 4.1 Implementation Plan

• Introduce a feature flag in cloud configuration to toggle between Struts actions and Spring Boot REST controllers.

• Initially route a small percentage of traffic to the new Spring Boot flow.

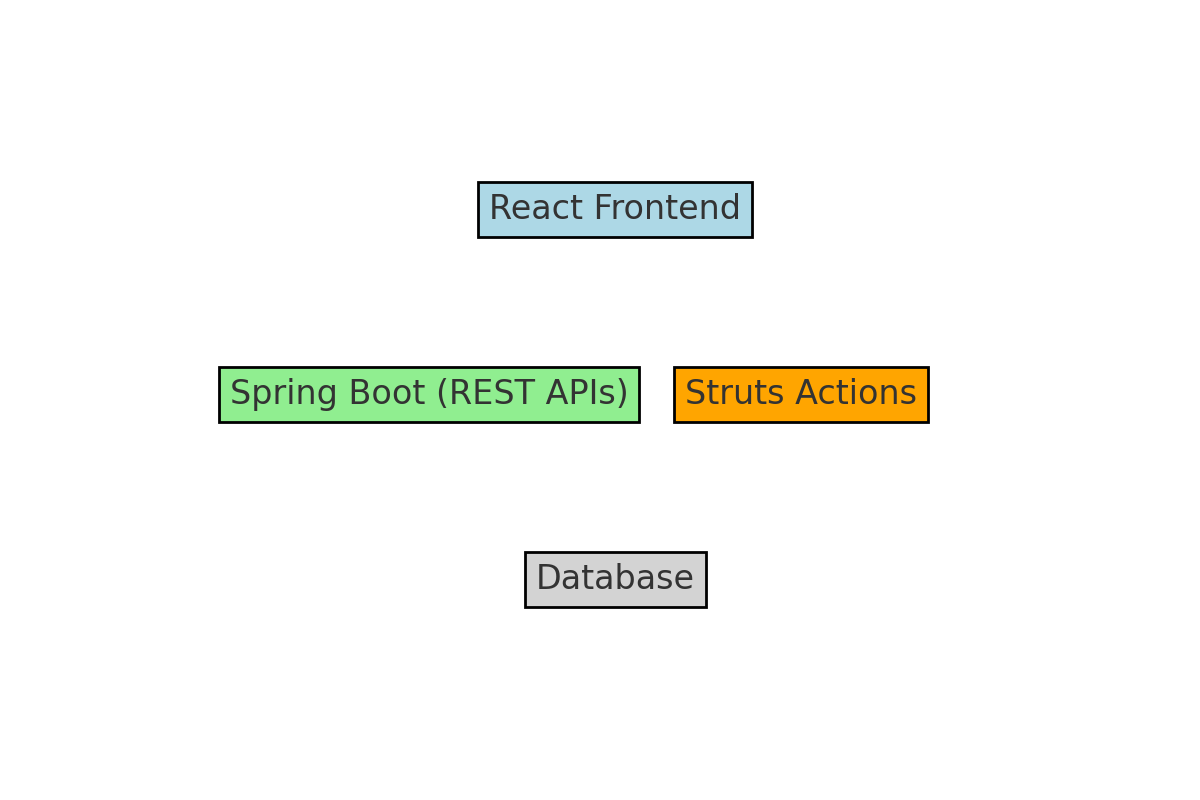
• Gradually transition all requests to the new flow while monitoring performance and stability.

### 4.2 Parameterization for Flow Selection

Define a configuration parameter in cloud properties to enable/disable Struts flow.

Example configuration:

enable.struts.flow=true # When true, Struts actions are used  
enable.struts.flow=false # When false, Spring Boot REST APIs are used



# 5. Conclusion

## 5. Conclusion

• The migration from Struts to Spring Boot REST improves maintainability, scalability, and performance.

• A parallel run strategy ensures minimal risk and a smooth transition.

• The React frontend remains unchanged, making migration seamless.

# 6. References

## 6. References

• Spring Boot Documentation

• Struts to Spring Migration Guide

• REST API Design Best Practices

## Appendix: Detailed Login Flow Migration

### Current Login Flow (Struts-based)

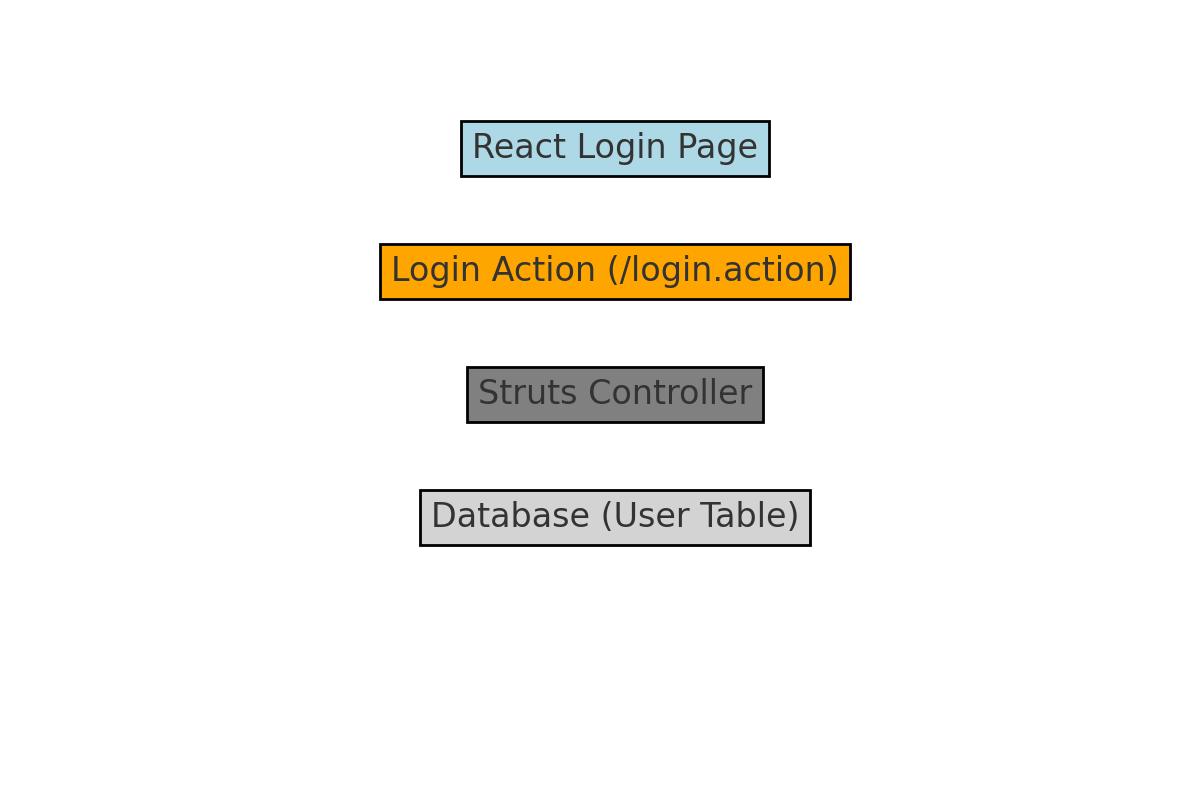
The current login flow follows these steps:

1. The user enters credentials in the React login page.

2. A request is sent to the backend endpoint `/login.action`.

3. Struts controller processes the request and authenticates the user.

4. If authentication is successful, a session is created, and the user is logged in.



### Future Login Flow (Spring Boot-based)

The new login flow will be implemented as follows:

1. The user enters credentials in the React login page.

2. A request is sent to the new Spring Boot REST API endpoint `/api/login`.

3. Spring Security handles authentication and generates a JWT token.

4. If authentication is successful, the token is returned to the frontend, and the user session is managed via JWT authentication.

