Experimental Report

Rebuilding a Bootstrap Cake Shop as a Java Web Application

Student Name: Mohammad Abdullah

Student ID: 202322240356

Course: Java Web

Date: 9 December 2025

Instructor: D.cora

Department of Computer Science
China West Normal University
Academic Year: 2025

Contents

Abstract

This experiment converts an existing Bootstrap-based front-end (*Sweet Delights* cake shopping website) into a full Java web application using **Spring Boot 3**, **Spring MVC**, and **Thymeleaf**, while preserving the original design, responsiveness, and user experience. The backend adds server-side routing, dynamic product rendering, and a session-backed shopping cart. The project also implements modern **light/dark/auto** color modes via Bootstrap 5.3's `data-bs-theme` attribute and respects the system preference with `prefers-color-scheme`. The final deliverable is a single runnable JAR and screenshots of core pages suitable for academic submission.

1 Objective

- Rebuild the multi-page Bootstrap cake shop into a Java web application.
- Use Spring Boot (embedded Tomcat), Spring MVC controllers, and Thymeleaf templates.
- Preserve design fidelity and responsiveness using Bootstrap 5.3.
- Implement a session-based cart and product detail routes.
- Support light/dark/auto color modes with persistence.
- Produce a reproducible build and deployment workflow (Maven).

2 Environment and Tools

Component	Specification
Operating System	Windows 11 (x64)
JDK	Temurin/OpenJDK 17
Build Tool	Apache Maven 3.9+
Framework	Spring Boot 3.3.x (Web, Thymeleaf)
Template Engine	Thymeleaf 3 (natural templates)
Front-end	Bootstrap 5.3 (color modes), Bootstrap Icons
IDE	Visual Studio Code (Java extensions)

Table 1: Software environment used.

Rationale. Spring Boot provides embedded Tomcat and auto-configuration; Thymeleaf supports "natural templates" that render as valid HTML even before server processing; Bootstrap 5.3 provides responsive components and first-class color-mode support.

3 System Design

3.1 Architecture

- **Routing/Controllers:** Spring MVC controllers map /, /shop, /product/{id}, /cart, /about, /contact.
- **Templates:** Thymeleaf templates in src/main/resources/templates with shared fragments (head, navbar, footer).
- **Static Assets:** CSS/JS/images under src/main/resources/static served by Spring Boot.
- **Data Layer:** In-memory repository (InMemoryProductRepository) for products (IDs, names, categories, prices, images).
- **Cart:** Session-backed map of productId → quantity, with add/update/remove operations.
- **Theming:** data-bs-theme + small JS to persist light/dark/auto to localStorage and to honor prefers-color-scheme.

3.2 Project Structure

```
sweet-delights-java/
  pom.xml
  src/main/java/com/example/sweetdelights/
    SweetDelightsApplication.java
    model/Product.java
    repo/InMemoryProductRepository.java
    web/{HomeController, ShopController, ProductController, CartController,
     SimplePageController}.java
  src/main/resources/
    application.properties
    templates/
      fragments/{head,navbar,footer}.html
      {index,shop,product,cart,about,contact}.html
    static/
      css/styles.css
      images/{hero-cake.jpg,cake1.jpg,...,about-us.jpg,map-placeholder.jpg}
```

4 Configuration and Implementation

4.1 Maven and Spring Boot

The `pom.xml` declares `spring-boot-starter-web`, `spring-boot-starter-thymeleaf`, and optional `spring-boot-devtools`. A single command produces a runnable JAR:

```
mvn clean package
java -jar target/sweet-delights-java-1.0.0.jar
```

4.2 Templates and Static Resources

Thymeleaf views are placed under `templates/` and static assets under `static/`. A shared `head` fragment loads Bootstrap CSS, Icons, and `styles.css`; `navbar` and `footer` fragments avoid duplication across pages.

4.3 Controllers and Session Cart

`CartController` exposes POST endpoints to add, update, and remove items from the session cart. `ShopController` lists products; `ProductController` resolves product details by ID.

4.4 Color Modes (Light/Dark/Auto)

A tiny script in `head.html` sets `data-bs-theme` *before* CSS loads, preventing a flash of the wrong theme. A navbar dropdown allows choosing Light, Dark, or Auto (system) and persists to `localStorage`. Custom CSS variables are overridden for dark mode.

5 Testing and Results

5.1 Functional Tests

Test Case	URL/Action	Expected	Status
Home page loads	/	Hero, featured items	PASS
Shop grid	/shop	Equal-height cards, buttons aligned	PASS
Product detail	/product/1	Correct image/price	PASS
Add to cart	POST /cart/add/1	Item appears in cart	PASS
Update/remove	POST /cart/update, /cart/remove	Quantity changed/removed	PASS
About/Contact	/about, /contact	Static content renders	PASS
Theme toggle	Navbar dropdown	Light/Dark/Auto persist	PASS

Table 2: Summary of functional verification.

5.2 UI/UX Validation

- Responsive navbar and grid across breakpoints.
- Uniform card imagery using Bootstrap ratio + `object-fit: cover`.
- Themed buttons and footer respecting light/dark mode.

6 Screenshots

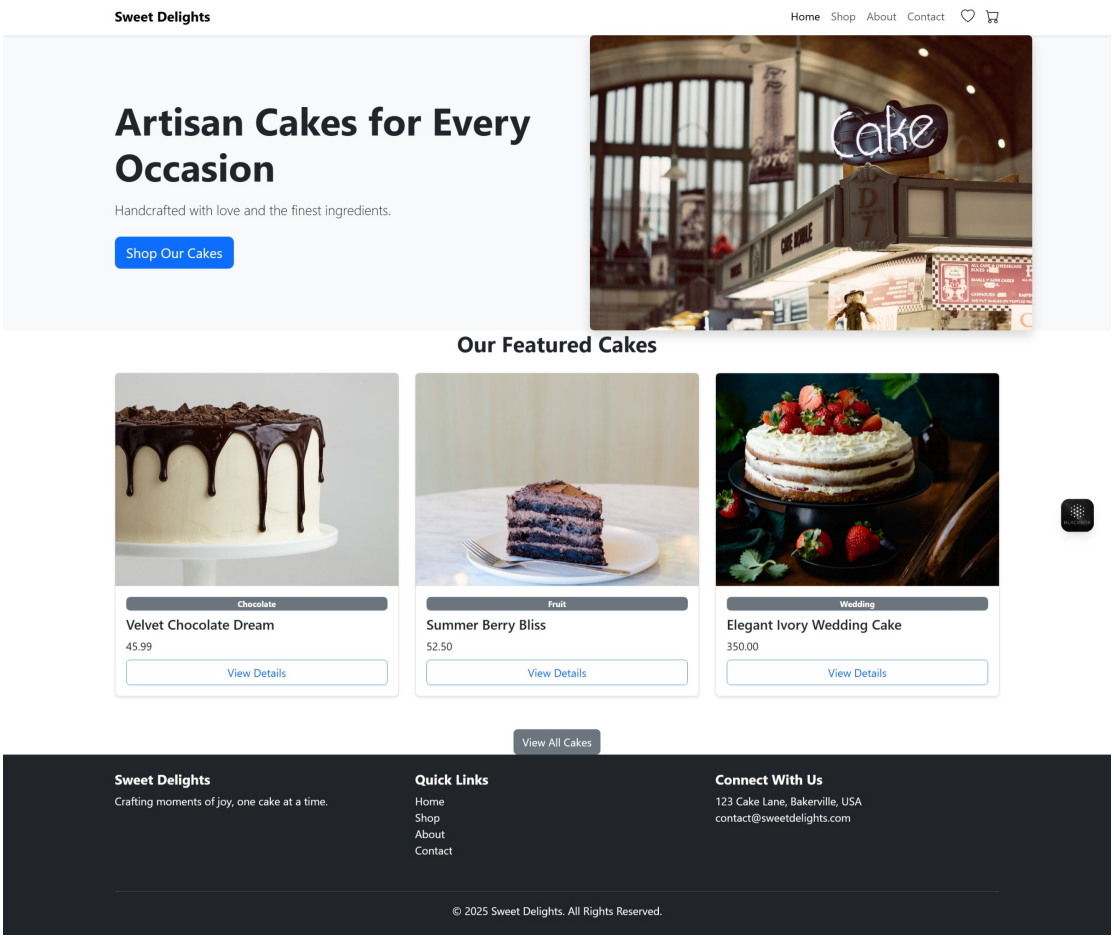

Figure 1: Home page with hero and featured products (Light mode).

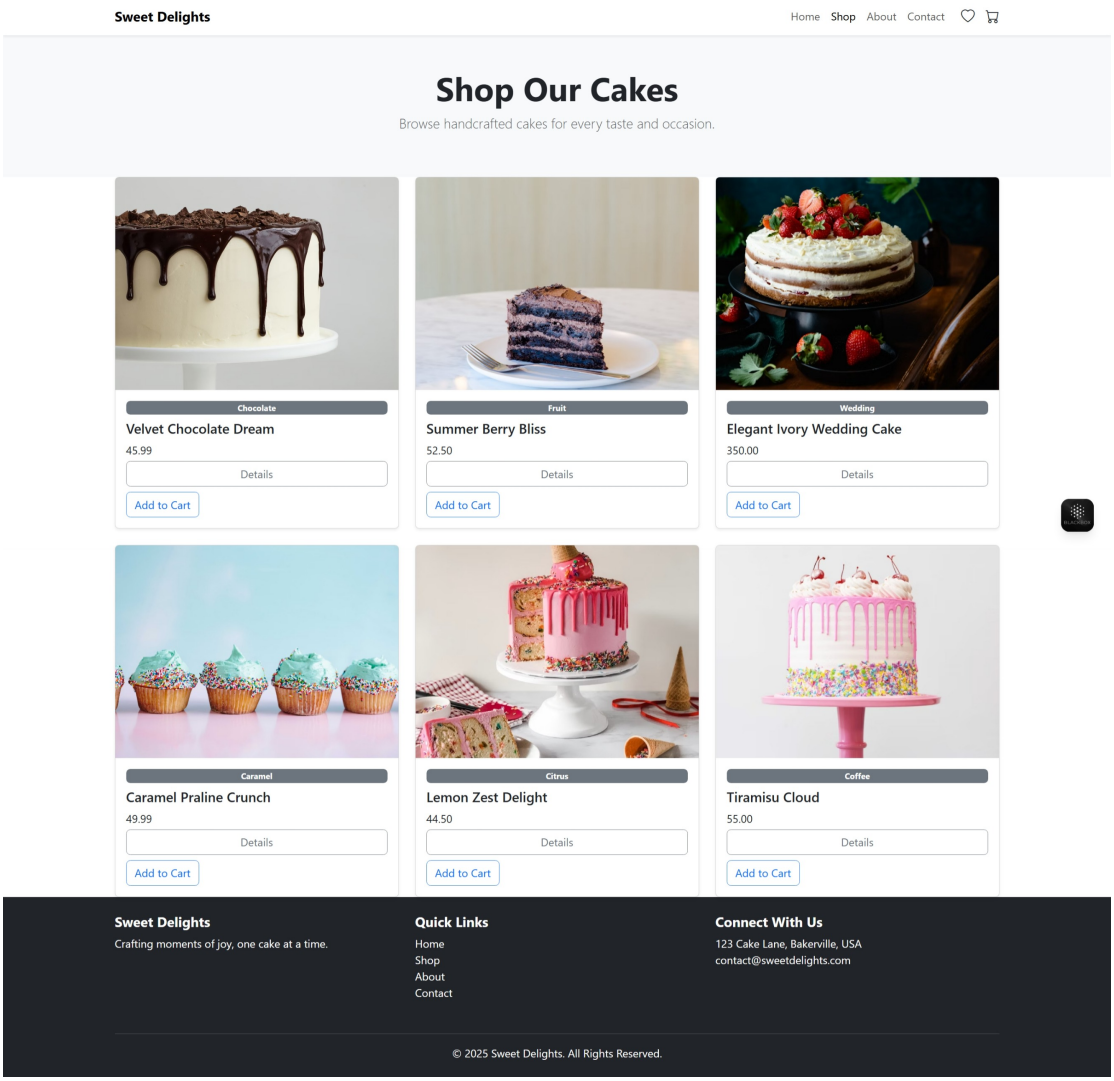

Figure 2: Shop grid with equal-height cards and consistent imagery.

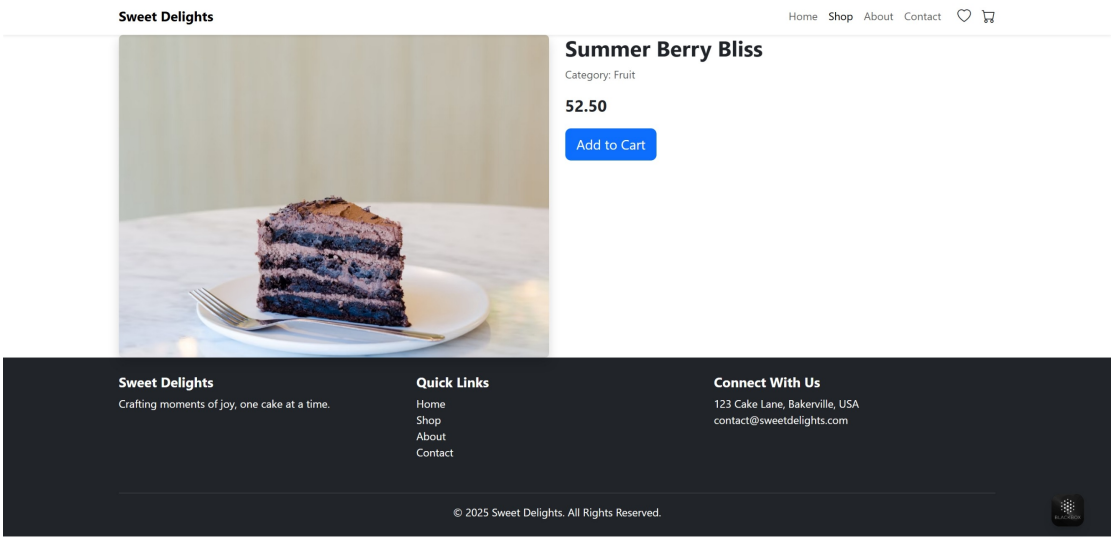

Figure 3: Product detail page with add-to-cart.

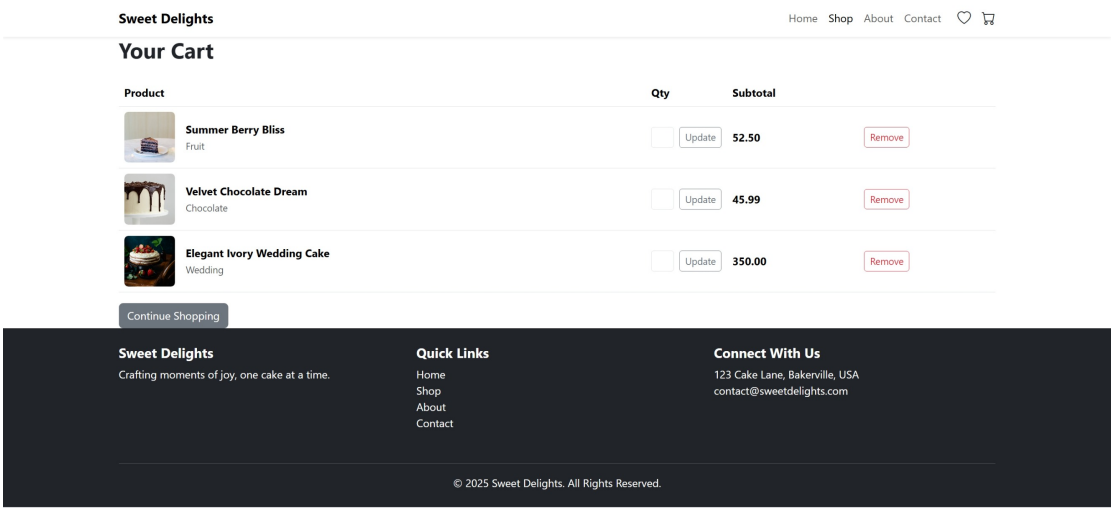

Figure 4: Cart view with update/remove actions.

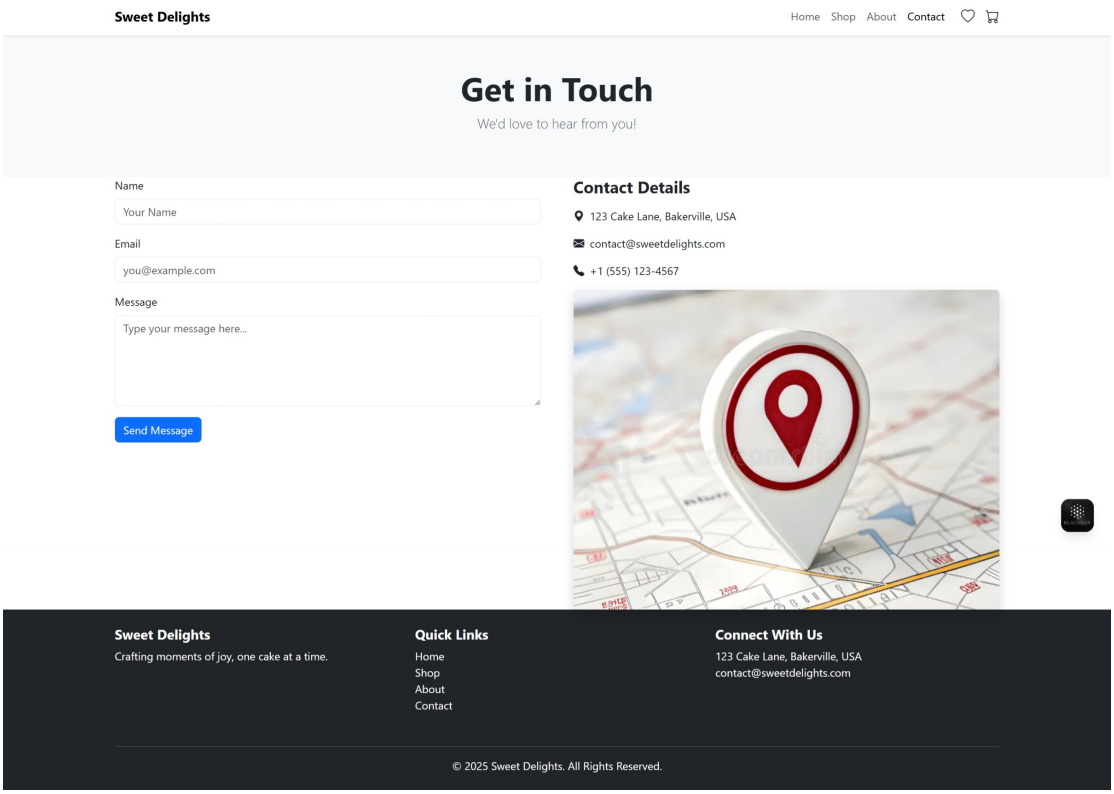

Figure 5: Theme dropdown (Light/Dark/Auto) in navbar.

7 Problems Encountered and Resolutions

7.1 Template Caching in Development

Issue: HTML changes not visible immediately.

Fix: Disable cache via `spring.thymeleaf.cache=false` (or include DevTools, which

disables cache automatically in dev).

7.2 Static Resources Not Loading

Issue: 404s on CSS/images.
Fix: Place assets under `src/main/resources/static` and reference with `/css/...`, `/images/...`. Thymeleaf's `@{...}` ensures correct paths.

7.3 Image Height Inconsistency

Issue: Product cards had varying heights.
Fix: Bootstrap `.ratio` wrappers (e.g., `ratio-4x3`) + `object-fit: cover` on `img` ensured uniform image areas and tidy layouts.

7.4 Theme Flicker on First Paint

Issue: Light-to-dark "flash" when toggling modes.
Fix: Early inline script sets `data-bs-theme` before CSS loads; navbar script persists user choice to `localStorage` and respects OS preference.

8 Conclusion

The conversion successfully preserves the visual design of the original Bootstrap cake shop while adding a robust Java backend with Spring Boot and Thymeleaf. Server-side routing, session cart, and reusable fragments improve maintainability. Bootstrap 5.3 color modes provide a polished Light/Dark/Auto experience. The project builds into a single executable JAR, making it straightforward to demo, deploy, and submit.

9 Appendix: Key Configuration Snippets

9.1 application.properties

```
spring.thymeleaf.cache=false
server.port=8080
```

9.2 Head Fragment (theme loader excerpt)

```
<script>
  (function () {
    const storageKey = 'theme';
    const getStoredTheme = () => localStorage.getItem(storageKey);
    const getPreferredTheme = () =>
      matchMedia('(prefers-color-scheme: dark)').matches ? 'dark' : 'light
    ';
    const theme = getStoredTheme() || getPreferredTheme();
    document.documentElement.setAttribute('data-bs-theme', theme);
  })();
</script>
```

9.3 Card Image (uniform height)

```
<div class="ratio ratio-4x3">
  <img class="card-img-top object-fit-cover" src="/images/cake1.jpg" alt="
    ...">
</div>
```