Babeș-Bolyai University

Faculty of Mathematics and Computer Science

SeatSurfer: A Database-Centric Seat Management System

Bachelor's Thesis

Author: Your Name Here
Supervisor: Lect. Dr. [Supervisor Name]
Specialization: Computer Science (English)
Year: 2025

Abstract

SeatSurfer is a modern web and mobile application designed to address the rising need for flexible and efficient seat management in office environments. The system allows employees to book seats in real time, view floor layouts, and manage reservations. Admins can configure office layouts, generate occupancy reports, and analyze usage patterns.

This thesis presents the theoretical and practical underpinnings of database systems applied in the development of SeatSurfer. We explore existing systems, compare architectural choices, and present the technologies used — including Flutter, Spring Boot, PostgreSQL, and Firebase. A relational data model was chosen to enable fast queries, maintain data integrity, and support multi-user interaction.

The development process follows Agile methodology with continuous integration and testing, ensuring modularity, scalability, and maintainability. The final product provides a robust infrastructure for seat reservation, occupancy analytics, and real-time collaboration between employees and administrators.

Keywords: databases, seat booking, Flutter, Spring Boot, reservation system, PostgreSQL, office management.

Contents

Chapter 1

Introduction

1.1 Context and Motivation

With the growing trend of hybrid and flexible work models, managing physical office spaces efficiently has become a new challenge. Organizations are looking for smart systems that offer visibility over seat occupancy and enable employees to plan their presence in shared environments. The SeatSurfer application addresses this demand by providing a seat reservation platform integrated with real-time data and analytics.

1.2 Purpose and Scope

The primary goal of this thesis is to present a practical and theoretical analysis of database systems applied to a real-world use case: SeatSurfer. The application allows users to view floor layouts, reserve seats, manage bookings, and generate occupancy reports.

From a technical perspective, the project focuses on:

- Designing an optimized relational data model
- Implementing a RESTful backend service using Spring Boot
- Building an intuitive frontend in Flutter
- Integrating user authentication and secure booking workflows
- Creating admin dashboards with occupancy insights

1.3 Structure of the Thesis

This paper is structured as follows:

- **Chapter 2** introduces similar systems and applications, discussing their functionalities and limitations.
- **Chapter 3** describes the technologies used in the development of SeatSurfer.
- **Chapter 4** presents the architecture of the system, including database design and API structure.

- **Chapter 5** showcases the functionalities implemented, such as booking, admin tools, analytics, and PDF reports.
- **Chapter 6** concludes the paper and proposes future development directions.

Chapter 2

Related Work and System Benefits

2.1 Existing Applications and Their Limitations

Several seat booking and office management platforms have emerged in recent years, such as Microsoft Outlook Room Finder, Robin, Deskbird, and Skedda. These tools help organizations manage office layouts and enable employees to book seats and meeting rooms. However, many existing systems suffer from:

- High costs of licensing
- Lack of customization
- Limited data analytics features
- Poor integration with internal tools

2.2 Innovation Introduced by SeatSurfer

SeatSurfer introduces:

- Full control over seat layouts and logic
- Visual seat mapping
- Smart PDF reports with date filtering
- Integration with Google and Outlook calendars
- Open-source and customizable deployment

2.3 Advantages of the Chosen Stack

By using Flutter, Spring Boot, PostgreSQL, and Firebase, SeatSurfer provides modular, scalable, and secure architecture suitable for real-time collaboration and future scalability.

Chapter 3

Technologies Used

3.1 Overview of the Tech Stack

SeatSurfer's architecture is based on:

- Flutter – frontend UI
- Spring Boot – backend API
- PostgreSQL – database
- Firebase – authentication

3.2 Flutter: UI Development

Flutter is an open-source framework by Google that allows cross-platform development from a single codebase [3]. With it, SeatSurfer ensures a responsive user experience on Android, iOS, and web using a single codebase.

3.3 Spring Boot: Backend Services

Spring Boot is a powerful Java-based backend framework that simplifies the development of RESTful APIs [4]. It enables rapid application development using dependency injection, controller routing, and service-based architecture.

3.4 PostgreSQL: Relational Database

PostgreSQL is an open-source relational database system known for its strong support of transactions, constraints, and indexing [1]. SeatSurfer stores seat, booking, and user data in PostgreSQL using a normalized schema.

3.5 Firebase: Authentication and Messaging

Firebase offers secure authentication and session control for both users and admins [2]. It integrates easily with Flutter and provides token-based authentication used by the backend for access control.

3.6 Development Tools

The following tools were used throughout the project:

- Visual Studio Code – frontend
- IntelliJ IDEA – backend
- Postman – API testing
- GitHub – version control

Chapter 4

System Architecture

4.1 Overview

The architecture of SeatSurfer follows a layered and modular design to ensure scalability, maintainability, and clarity. The system consists of the following components:

- A Flutter-based frontend for mobile and web clients
- A Spring Boot REST API backend
- A PostgreSQL relational database
- Firebase authentication for login and role management

Communication between components is stateless and performed via RESTful HTTP endpoints. Booking logic, seat layout logic, and occupancy reporting are handled exclusively on the backend.

4.2 Entity-Relationship Diagram

The core data model is relational and consists of the following entities:

- **User** — a person with an email, role (user/admin), and associated bookings
- **Floor** — a uniquely identified floor layout in a building
- **Seat** — a specific seat on a floor, defined by coordinates
- **Booking** — a user's reservation for a given seat and date

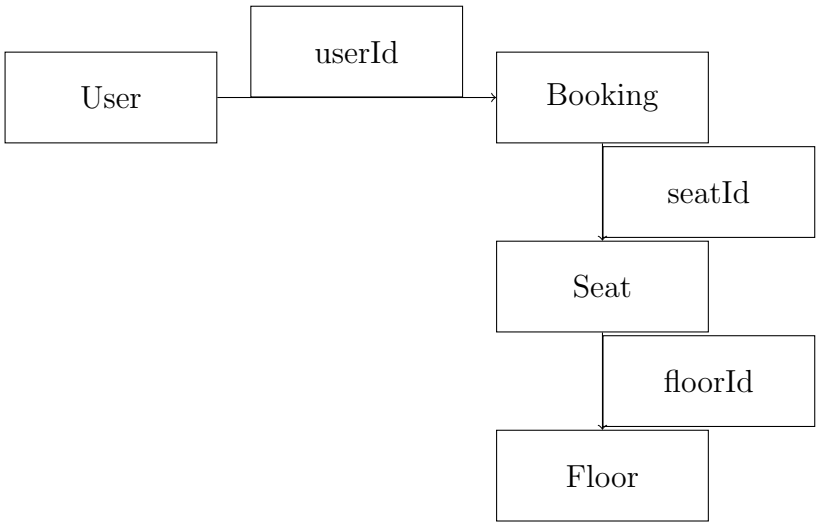

Figure 4.1: Entity-Relationship Diagram of the SeatSurfer Database

This schema is normalized, with foreign key relationships ensuring data consistency. Additional constraints like unique bookings per user per date are enforced at the database level.

4.3 Backend Architecture (Spring Boot)

The backend follows a clean separation of concerns using the standard structure:

- **Controller Layer** — handles incoming HTTP requests
- **Service Layer** — contains business logic
- **Repository Layer** — interacts with the database via Spring Data JPA

Each entity (e.g., ‘Seat‘, ‘Booking‘) has its own ‘Controller‘, ‘Service‘, and ‘Repository‘. Example:

- ‘BookingController.java‘ exposes ‘/api/bookings‘
- ‘BookingService.java‘ contains logic to prevent double bookings
- ‘BookingRepository.java‘ provides query methods using JPA

4.4 Frontend Integration (Flutter)

The Flutter application interacts with the backend through HTTP REST API calls. Users are authenticated via Firebase, and the access token is sent in the request headers.

The frontend maintains the current session, makes API calls to:

- Display available seats per floor/date
- Send booking/cancel requests
- Display future bookings, occupancy charts

4.5 System Interaction Diagram

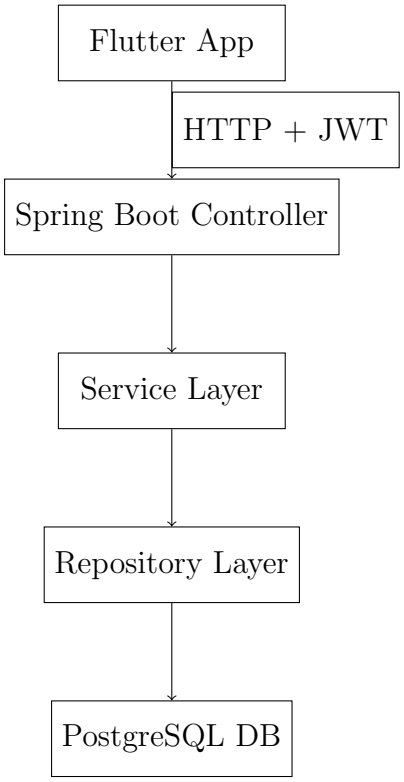

Figure 4.2: System Architecture Flow

The design allows each layer to evolve independently. For example, switching from PostgreSQL to MySQL, or adding a caching layer like Redis, would not affect the controller or frontend.

Bibliography

[1] PostgreSQL Global Development Group. *PostgreSQL Documentation*, 2024.

[2] Google Inc. *Firebase Documentation*, 2024.

[3] Google Inc. *Flutter Documentation*, 2024.

[4] Pivotal Software. *Spring Boot Reference Guide*, 2024.