MIDDLE EAST TECHNICAL UNIVERSITY
NORTHERN CYPRUS CAMPUS

CNG 495

Fall – 2025

Capstone Project Final Report

Name: Nurlan

Surname: İldırımlı

SID: 2745438

Project Title: University Sports Center Reservation System

Date of Submission: 25.12.2025

Contents

1 Introduction

This project proposes a cloud-based reservation platform for the University Sports Center. Students and staff can browse facilities (Football Pitch, Tennis Court, volleyball / Basketball court outdoor, Table Tennis), view available time slots, and create/cancel reservations. Authorized personnel can define working hours, capacities, maintenance windows, and policies (e.g., per-week booking limits).

The system will provide real-time availability, double-booking prevention. The web application will be implemented with a modern front-end and a serverless backend on Firebase. Data will be stored in a cloud database with transactional safeguards to ensure consistency during peak demand.

The main benefits of this system include eliminating manual reservation procedures, preventing double-booking and overcrowding of facilities, and providing a real-time, transparent scheduling experience. The platform offers practical advantages over traditional methods such as physical logs or phone-based reservations, where human errors and communication delays are common. The system introduces an automatic student penalty mechanism that restricts students from making new reservations for one week if they do not attend their reserved session, which helps increase responsible usage and reduce facility misuse.

This project also highlights the integration of scalable cloud technologies. Firebase Authentication enables secure user access control, while Firestore stores reservation data in real time. Firebase Cloud Functions automate the notification process via email and ensure business rules are consistently applied. The frontend is built using React and TypeScript, allowing a responsive and maintainable codebase that supports future extensions such as reservation history analytics and campus mobile app integration.

A comparable institutional system currently in use is the METU Ankara Reservation System, which allows students to book campus sports facilities through an online portal. The platform can be accessed at: https://rez.metu.edu.tr/

This project was inspired by the METU reservation system in terms of its core functionality and purpose.

2 Structure of the Project

This chapter explains the full architecture of the University Sports Center Reservation System. It covers system modules, user roles, cloud services, screenshots of implemented functions, and technologies used.

2.1 System Overview

The system provides two main user roles:

Role	**Main Capabilities**
Student	Browse facilities, check availability, book slots, cancel reservations, update profile
Admin	Manage facilities and weekly slot templates, update reservation status, view all reservations

Table 1: User roles and capabilities

Real-time synchronization is achieved using Firebase services, including Firestore and Cloud Functions, together with a React + TypeScript frontend.

2.2 Main Components and Responsibilities

Component	**Description**	**Key Files**
Authentication	Secure login and role-based routing	Firebase Auth, RequireStudent.tsx, RequireAdmin.tsx
Facility Management	CRUD management of facilities	ManageFacilities.tsx, facilityService.ts
Weekly Slot Scheduling	Weekly slot templates with visibility and availability	ManageSlots.tsx, facilityService.ts
Student Booking	Weekly grid display and reservation creation	StudentHome.tsx, reservationService.ts
User Profile	Editable personal information	UserProfilePage.tsx, userService.ts
My Reservations	View and cancel own reservations	StudentReservations.tsx
Admin Panel	Update reservation statuses and view all bookings	AdminHome.tsx
Cloud Functions	Send reservation emails automatically	functions/src/index.ts

Table 2: Main modules and responsibilities overview

2.3 Utilized Cloud Services

Cloud Service	Purpose	Benefit
Firebase Authentication	Identity and login management	Secure role-based access
Cloud Firestore	Store facilities, slots, reservations, and profiles	Real-time scalable database
Cloud Functions	Email notifications via triggers	No server maintenance required
Firebase Hosting	Deployment of the web application	HTTPS, CDN, global access
SMTP (Nodemailer)	Email delivery	Reliable communication

Table 3: Cloud services used in the project

2.4 Information Flow and Architecture

- User logs in via Firebase Authentication.
- UI directs to student/admin pages using role guards.
- Student selects facility and date.
- System filters weekly slot templates by weekday and checks availability.
- Reservation data is stored in Firestore.
- Cloud Functions send confirmation/cancellation emails.
- Admin updates reservation status when needed.

Figures included in the report:

2.5 Use Case Diagram

Here you can see the use case diagram in Figure 1.

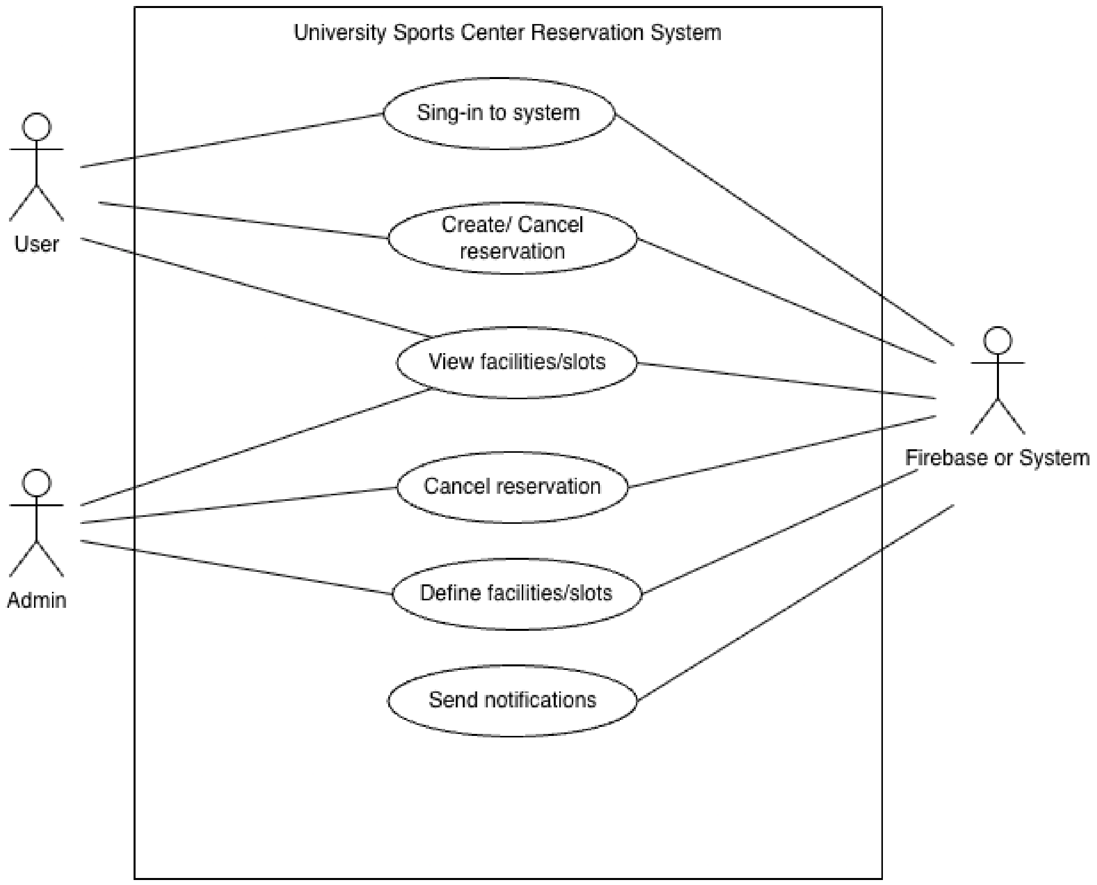

Figure 1: Use case diagram.

2.6 Data Flow Diagram

Here you can see the data flow diagrams in Figure 2 and Figure 3.

Context Level:

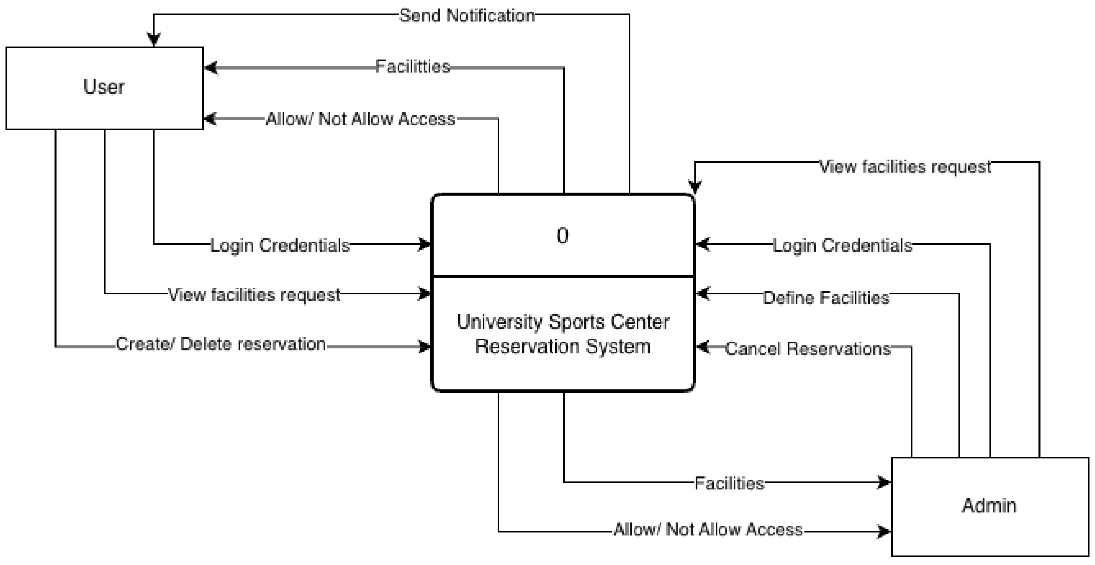

Figure 2: Context Level.

Level 0:

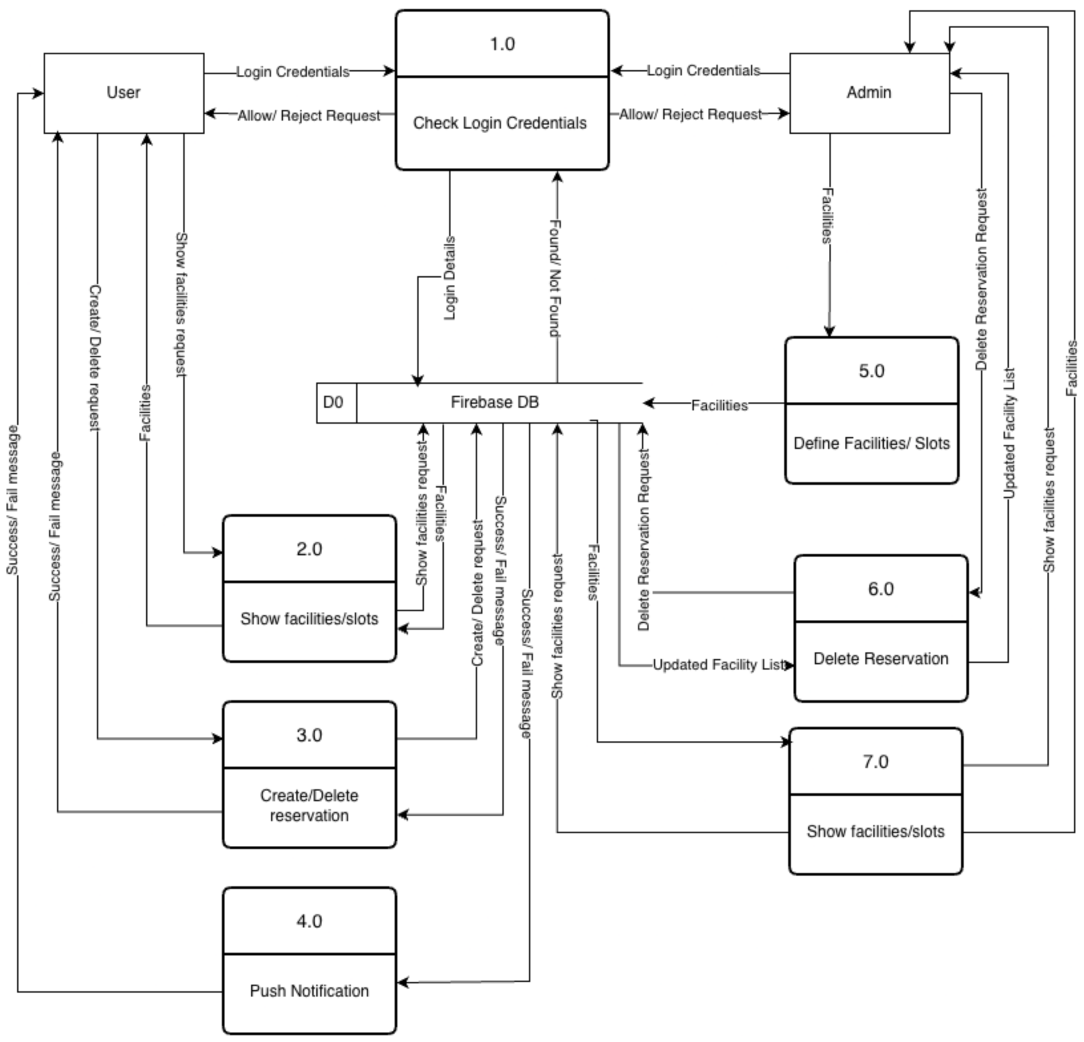

Figure 3: Level 0.

2.7 User Manual for Implemented Features

Student Interface

- Login using email and password
- View availability grid and select slots based on date
- Book reservations via a modal form
- Manage reservations with cancel option
- Update personal information in profile page

Admin Interface

- View all reservations in a table
- Mark reservations as completed or not attended

- Manage facilities (CRUD)
- Manage weekly slot templates

2.8 Technologies Used

Layer	Technology
Frontend	React (Vite), TypeScript, CSS
Backend	Firebase Cloud Functions (Node.js + TS)
Database	Cloud Firestore
Authentication	Firebase Authentication
Deployment	Firebase Hosting
Tools	GitHub, VS Code

Table 4: Technologies and tools used

2.9 Tutorials and Development Notes

- Reservation creation checks per-day availability without modifying slot template structure.
- Weekly slot templates use weekday-based configuration for reuse every week.
- Admin status updates trigger Cloud Functions to send emails.
- Role guards enforce secure interface separation.
- Serverless architecture reduces deployment and maintenance overhead.

This section demonstrated the complete structure of the system and how each cloud component contributes to an efficient and maintainable web platform.

3 Project Statistics

This section summarizes the development effort, timeline, code metrics, and technologies used in the project.

3.1 Development Timeline & Responsibilities

The project was fully implemented by Nurlan İldırımlı. The following table presents each milestone and the completed responsibilities.

Timeline	**Completed Tasks**
Week of October 13–20	Project setup (React + Vite + TypeScript). Firebase initialization. Authentication implemented.
Week of October 20–27	Firestore data model designed. Facilities and slots fetching implemented.
Week of October 27–November 3	Student reservation flow completed. “My Reservations” and cancellation added.
Week of November 3–10	Admin dashboard created and restricted to admins only.
Week of November 10–17	Reservation status update logic added. Cloud Functions coded for notifications.
Week of November 17–24	SMTP setup for email delivery. Functions compiled successfully.
Week of November 24–30	UI improvements and full integration of trigger-based email flow.
December – January	Weekly slot template system, redesigned UI for student/admin pages, date-based reservation filtering, user profile updates, and improved data consistency completed.

Table 5: Project development timeline

3.2 Code Metrics

- Total lines of code (excluding lockfiles): **2,821**
- With lockfiles included: **18,127** (package-lock files dominate)

Breakdown by language:

- TypeScript & TSX (frontend + Cloud Functions): **2,358 lines**
- CSS: **111 lines**
- HTML: **13 lines**
- Other (config, JSON, env, scripts): Remaining portion

3.3 Programming Languages and Cloud Technologies

- TypeScript / TSX for application logic and serverless backend
- CSS for styling
- HTML for entry page
- JavaScript minimal usage for config scripts

3.4 Database Types and Storage Details

Firestore NoSQL database is used with the following collections:

- `facilities` – facility definitions
- `slots` – weekly slot templates
- `reservations` – booking records
- `users` – profile information
- `reservationBans` – (ban logic currently disabled)

Firebase Authentication is used for secure user identity management.

3.5 Cloud Function Runtime & Memory Notes

- Cloud Functions implemented in TypeScript using **Node.js 24**
- Default memory and compute configuration (e.g., **256 MB** for Gen 2)
- No custom scaling rules applied
- Frontend has standard browser-level resource requirements

3.6 Automated Functionality Status

- **onReservationCreated**: Sends confirmation e-mail upon reservation
- **onReservationStatusUpdated**: Sends email notifications for cancellation and no-show status
- Scheduled functions for weekly resets exist but remain **deactivated** until Firebase Blaze plan upgrade

Overall, the project reflects a complete cloud-based web system fully developed by a single engineer, successfully demonstrating scalable frontend and serverless backend expertise.

4 GitHub Repository

Clone Command: git clone https://github.com/nurlanildirimli/University-Sports-Reservation.git

5 References

1. Firebase Documentation – Authentication, Firestore, Cloud Functions. `https://firebase.google.com/docs`
2. Google Cloud Storage – Object Storage Best Practices. `https://cloud.google.com/storage/docs/best-practices`