**Airport Simulation Exercise**

**Back-End:** Developed in C# ASP.NET Core Web API

**Front-End:** developed in TypeScript AngularJS version 11.2.6

**Design patterns**: Controller, Service, Repository, Dependency Injection

**Additional features**: PostgreSQL Database in cloud, EntityFramework to access database

----------------------------------------------------------------------

**Considerations and Assumptions:**

**BackEnd**

Regarding landings and departures, a flight has to request them to the control tower and receive either an approval or denial.

Realistically there can be setbacks with flight therefore a plane can land and depart even if there is another scheduled to do the same at that time, meaning that the runway works in a first come first serve fashion.

In case of landing and departure approval the runway becomes locked and since this is a simulation, around every 15 seconds the runway will unlock itself.

The control tower is considered to be a Request Entity in the database and every action it does is considered a request.

**FrontEnd**

Navbar shows 3 seperate tabs:

* **Requests** - Shows all currently active requests as well as the status of the runway and the status of the airplanes.
* **History** - Shows communication history between control tower and flights and vice-versa.
* **Create Request**  Allows the creation of requests for testing purposes.

**Possible problems when running**

* If many requests are made to the database in rapid succession they might not complete since the database has a limit of 5 connections and no verification is made to alert that. This will not break the application
* Not every functionality mainly creating a request have full validation, which can lead to unpredictable results.