**Session 5 – Box office**

**Duration: 150 minutes**

# 1 Instructions

This document contains the special tasks for the 2nd session. We advise you to read this document carefully and to get an overview over the provided media like attached images, appendices and text.

The maximum time to complete as many tasks as possible is **150 minutes**. **Within this time,** you must read and understand all tasks and to store the results **into the folder** “***Session-5-yourname***” onto the desktop of your computer.

This is the first session – that means you will write the basis for the others. You are allowed to use all components, source code and windows of this session for the following sessions of the competition. Sometimes it makes sense to use parts of it also for other tasks in the future.

Please for every session create a file containing a short user guide and a technical description of how to configure and start up the application.

Make sure that you follow the provided style guide throughout all parts of the system.

Make sure that you provide appropriate validation and error messages throughout all parts of the system.

Make sure that all relevant buttons/links are working at the end of the session.

Make sure that you use appropriate naming conventions for all parts of the system as needed.

Make sure that your work is on the state of the art of Skill 09.

# 2 Contents

Session 5 of this Test Project consists of the following documentation/files:

1. AEC2024-Skill09-Session 5.pdf
2. Session5.sql
3. Server.zip

# 3 Project

## 3.1 Introduction

The **Event** **Agency** **NeuBrandenburg** (**EANB**) is a modern enterprise which organizes, manages and offers cultural events in Neubrandenburg and its environment.

This agency needs a desktop application for box office workers.

## 3.2 The Task

The database and a server for live data are given. Box office workers can get a real-time overview of the seats and sell tickets.

# 4 Application Structure

The application only consists of the ***Main Window***.

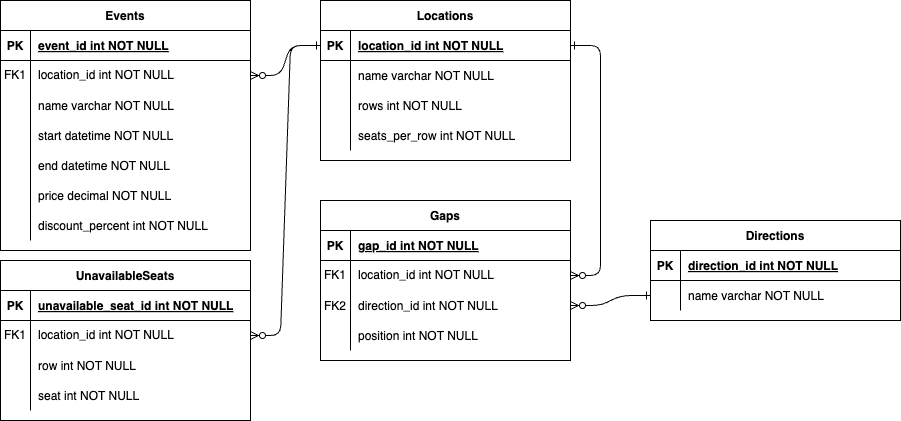
# 5 Tasks

## 5.1 Database import

**Quality tested** Database

**Work requested**  General

* Execute Session5.sql to import the given database
* The database structure provided cannot be altered
* Do not remove tables, add or delete any fields on the tables or change their data types
* Refer to the ERD below for the database structure



## 5.2 Server connection

**Quality tested** Server Connection

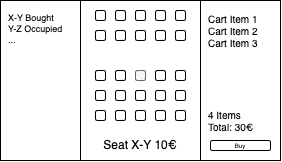
**Work requested**  General

* There is an existing server providing real time data for this application
* Executing Server.exe will start the server at localhost:3000
* The server uses the TCP protocol, this means you can connect to it using a TCP socket
* The protocol used by the server works as follows:
  + After connecting to the server, the event-id must be sent as a 16-bit integer with the most significant bit first. For a more detailed explanation consider this example where the number 4154 is transmitted. 4154 in binary is 0001 0000 0011 1010. The bits on the left have a higher value compared to the bits on the right. This means that the bits must be transmitted in order from left to right: 0, 0, 0, 1, 0, 0, and so on.
  + The server then sends a sequence of seat states each consisting of three 16-bit integers with the most significant bit first. The first integer is the row number, the second the column number and the third is the seat state. By default the seat is empty (State 0; a ticket for the seat can still be sold), but it can also be bought (State 1; someone owns a ticket for the seat, but he hasn't entered the event yet) and occupied (State 2; someone owns a ticket for the seat and has entered the event). The seat state sent by the server signals a change of the state in real time. Messages with invalid seat states or seat locations should be ignored.
  + The client can also send seat locations consisting of two 16-bit integers with the most significant bit first, to signal that the seat got bought at the box office. The first integer is the row number and the second the column number. These messages update the state of the corresponding seat to occupied.

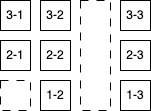
## 5.3 Main Screen

**Quality tested** Programming, UI Design, DB Access

**Mockup**



**Work requested** General

* Create the Main Screen.
  + The user should be able to select an event at the top
  + The screen is divided in three parts: the log on the left and the location overview in the middle and the cart on the right
  + The log shows the history of changes to the seat state
  + Each log entry consists of the new state and the seat location
  + When clicking a log entry, the corresponding seat should be selected in the location overview
  + The location overview should show a grid of all the seats with the size specified in the location table
  + There need to be gaps according to the entries in the gaps table referencing the location
  + Seats of the location listed in the unavailable Seats table should not be shown
  + For details refer to the example below. The example shows a layout with 3 rows and 3 seats per row. There is a vertical gap at position 2. Also seat 1 in row 1 is unavailable.
  + Each seat is coloured based on the current state
  + When a seat is selected, it is highlighted, and the price is shown at the bottom
  + The discount stored in the event table is applied, if the event has already started
  + If it can be sold there should be a button for adding it to the card next to the price
  + A seat can be sold if it is empty or if it is bought, but the ticket owner has not occupied it and the event has already started
  + A seat cannot be sold if the event is already over
  + A seat cannot be added to the cart, if the cart already contains the seat
  + When a seat is added to the cart it gets deselected
  + The cart should list the seat location and price of each seat that got added to the cart
  + If a seat cannot be sold anymore because the state changed or the event is over, it should be crossed out
  + There is a button for clearing the cart
  + There is a button next to each seat for removing the seat from the cart
  + The total number of seats and total price is shown
  + There is a checkout button next to the total price that is only enabled if each seat is available to be sold
  + If the checkout button gets pressed, the cart gets cleared and the seat locations get sent to the server

## 5.4 Protocol Design

**Quality tested** Software Design, Protocol Design

**Mockup** to be useful design

**Work requested**  General

* Extend the protocol description to support additional functionality required for a public ticket shop application
* The extended protocol needs to fulfil the following requirements
* Only authenticated users should have access to the data from the server
* Users get real-time information on the state of the seats
* Users can reserve seats without buying them for up to 30 minutes
* Users can buy seats before the start of the event

## 5.5 Documentation / Notes

**Quality tested** Documentation

**Work requested**

* Please create a file containing a short user guide and a technical description of how to configure the server, how to get access to your database and how to start up the application.
* In this file you may also leave *notes* that could be useful for the evaluation. *Notes* in this document are not part of the evaluation process.