

Oblig-I 2024:

Modeling aspects of EventTicket application



Ketil Stølen

Based on a case suggested by Andre Hauge, Vikash Katta, HiØ

General

- In this compulsory exercise (Oblig I) you will be trained in modelling the structure and behaviour of processes and technology focusing particularly on interaction and communication. We have selected a case that should be easy to grasp without prior domain knowledge. In a later compulsory exercise (Oblig III) you will be asked to conduct a security risk assessment of the same case.
- The description is underspecified, and there are many ways to do the tasks. This is intentional and reflects how things are in practise. You are free to make assumptions if they are stated explicitly by you in your solution-paper.
- The solution-paper in the form of a single pdf-document should be sent to [be submitted via Devilry](#) by midnight (24:00) September 23. You may work in groups of **maximum three students** or **you may work alone**. Hence, one solution may have up to three names. There should be no collaboration or copying **between different groups**. Hence, each group should solve the exercise independently.

General

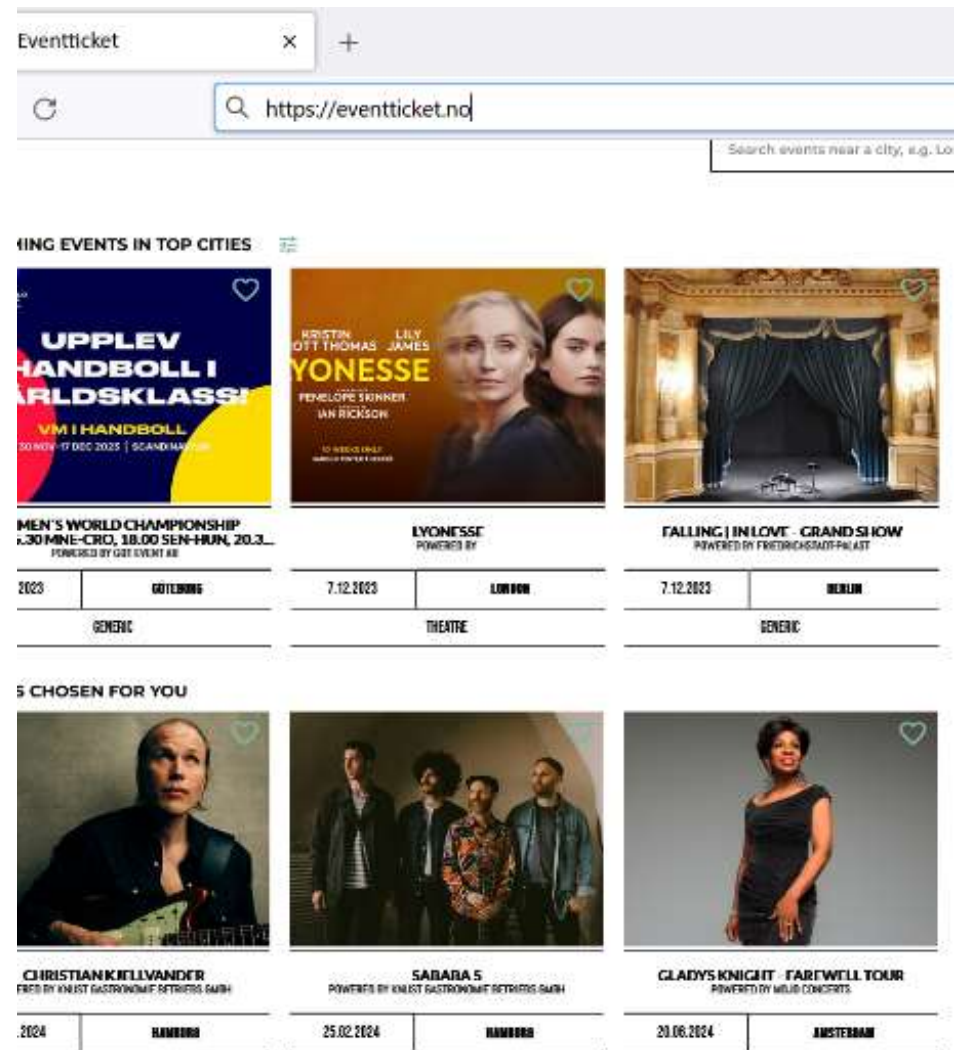
- You may use the tool of your preference. Some alternatives:
 - <https://www.eclipse.org/papyrus/> (powerful but involves a lot to install and use)
 - <https://app.diagrams.net/> (light weight)
 - VISIO with UML stencil
- **PLEASE NOTE THE FOLLOWING REGULATIONS:**
<https://www.uio.no/english/studies/examinations/compulsory-activities/mn-ifi-guidelines.html>

Case

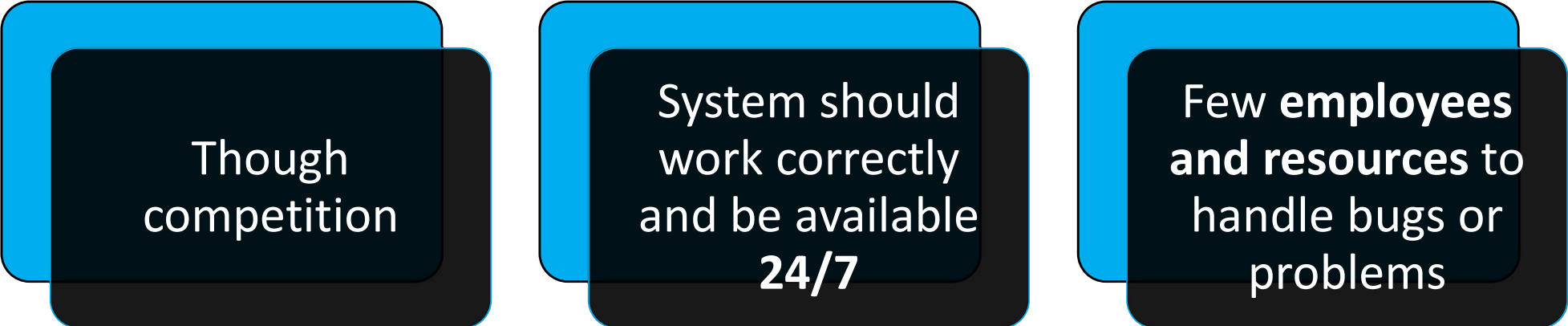
- In this oblig you will model aspects of an imaginary ticket-booking system developed by an imaginary company called EventTicket AS.
- The company and its system is described in the following eight slides.

EventTicket AS

- Imaginary IT-company from Norway
- Established 2021
- Small team, 4 employees
- Offers online ticket booking system
 - Organizers may create events, sell and track tickets for their events (concerts, sport, theatre etc.)
- EventTicket AS has more than 1000 events available at any point in time across the Nordic countries
- Turnover 50M NOK



Marked and criticality



Though
competition

System should
work correctly
and be available
24/7

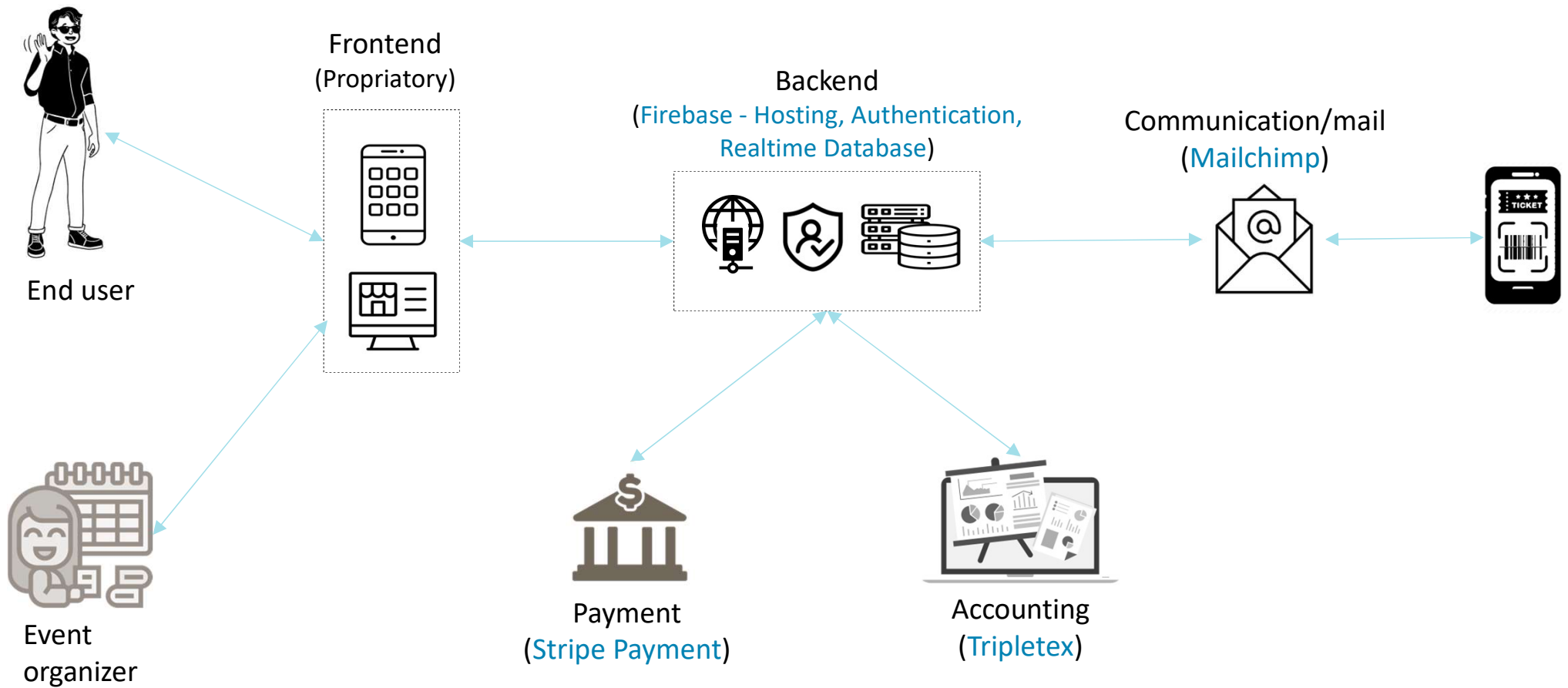
Few **employees
and resources** to
handle bugs or
problems

Employees and their responsibilities

- Per (CEO)
 - Strategic management
 - Personnel responsibility
 - Financial control
 - Stakeholder management
- Emma (Senior developer)
 - System development and management
 - Patching and coding
- Daniel (Marketing)
 - Marketing strategy
 - Customer engagement
 - Brand management
- Hanne (Sales)
 - Sales
 - Customer relations
 - Collection of user feedback

System

- Supports the full ticketing process
 - Establish and promote events
 - Ticket sale
 - Send/mail tickets to end-users (ticket buyers)
 - Check in/validate tickets when the event takes place
 - Communication with end-users
 - Managing the financial aspects with the event organizers



System/components



Frontend: User friendly interface available via web and mobile devices that makes it simple to navigate and select and buy tickets



Event database: Contains data for the full set of events



Ticketsystem: Robust software managing the issuing and validation of tickets



Payment: Secure payment gateway offering reliable payment transactions. Users may choose between different payment alternatives



Organizer's dashboard: Event organizers have access to a dedicated dashboard where they can set up, manage and promote their events

Third party providers

The ticket system integrates third party solutions via APIs:

- **Google Firebase**
 - **Hosting**, host web & app
 - **Authentication**, login and onboarding
 - **Realtime Database**, storage and management of critical event data
- **Stripe Payment**, for online ticket purchase
- **Mailchimp**, for distribution of tickets and updating end users via
- **Tripletex**, for invoicing and accounting

EventTicket is particularly worried about



Phishing Attacks



Ransomware



Imposter Scams



Third party dependencies

Task I

- Make a class-diagram describing the relevant concepts. The class-diagram should **at least** represent the following concepts: event, event organizer, ticket buyer, third party provider, and the five components on slide 10.
- The class-diagram **must contain** the generalization relation and illustrate correct use of both composition and aggregation. All associations **must** be decorated with multiplicities.

Task II

- Make one main sequence diagram describing a booking session. There **must at least** be lifelines for event, ticket buyer, frontend, event database and two of the three other components on slide 10.
- The sequence diagram **must** illustrate correct use of at least referencing (the referenced sequence diagrams **must** also be specified in full), loop, lifeline creation and termination, and opt. The sequence diagram should also describe some negative behaviour. Please avoid excessive complexity. The sequence diagram being made in this task will be used further in the next task.
- Describe the first ten events of at least one positive trace represented by your sequence diagram.

Task III

- Make a communicating state-machine that correctly reflects all the traces specified by the sequence diagram for the frontend.
- Make a communicating state-machine that correctly reflects all the traces specified by the sequence diagrams for the event database.
- The two state-machines must allow any positive trace allowed by the respective lifelines in the sequence diagrams but may have additional traces.
- Remember that the machines should communicate the messages specified by the sequence diagrams. Hence, for a lifeline, an input **signal (output signal)** in a state machine should be the **reception (sending)** of a message in the sequence diagram.