# Express.js Exam Retake – Airline System

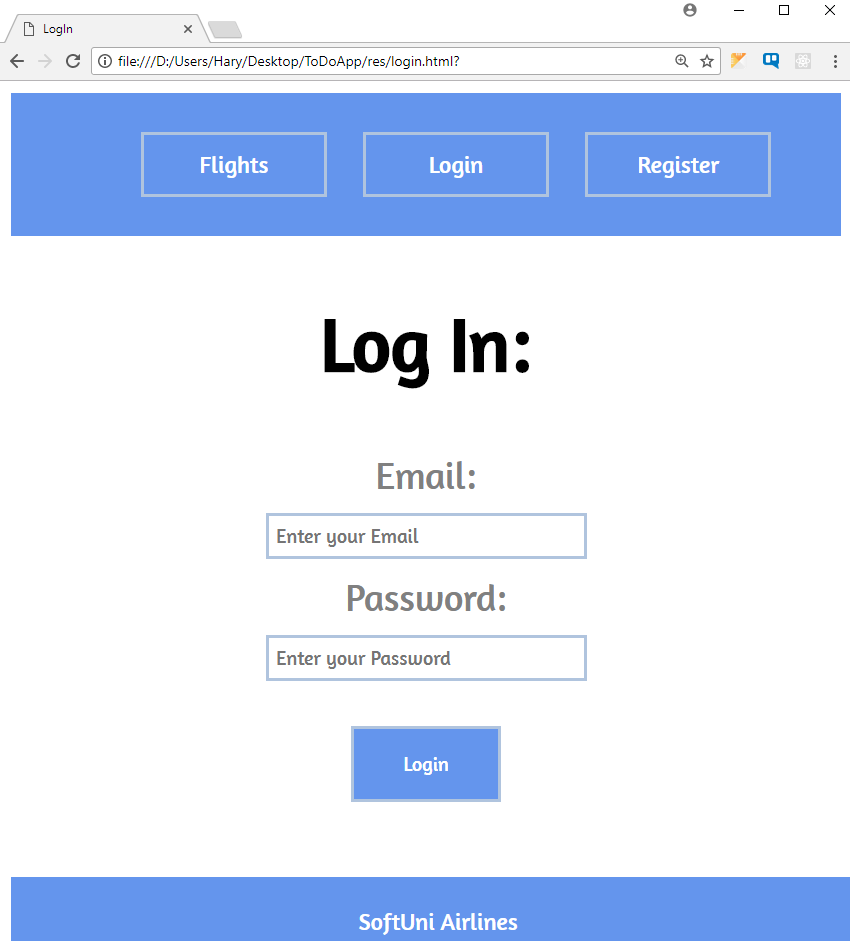
# Exam rules:

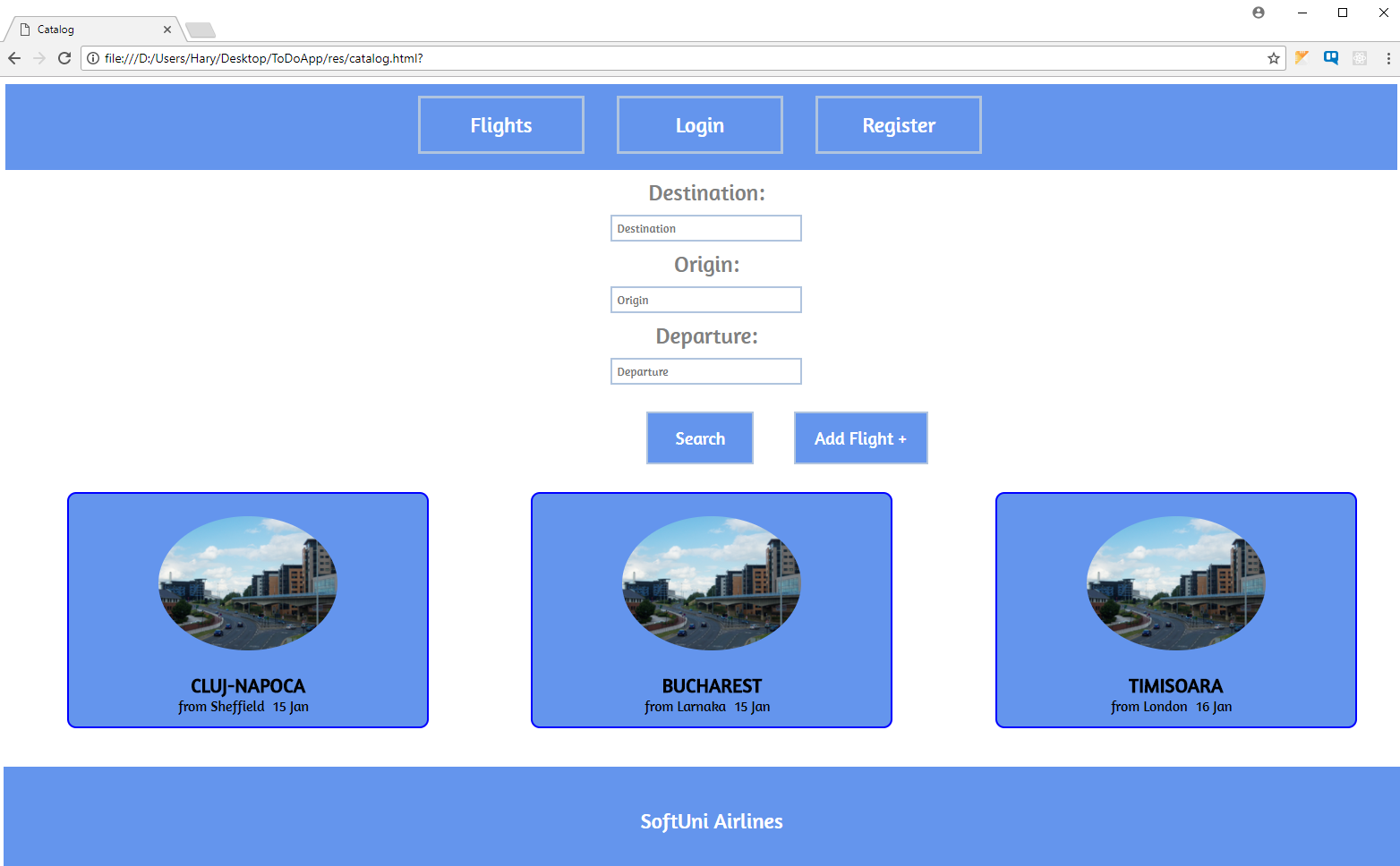
* You have 6 hours – from 9:00 to 15:00
* When you are ready, delete the *node\_modules* folder, make sure all dependencies are listed in the *package.json* file and submit your archived project at <https://softuni.bg/trainings/1732/expressjs-fundamentals-september-2017#lesson-7565>
* There will be no breaks during the exam but you can go to the toilet or to breathe some fresh air outside **if you are alone**
* **You may modify and extend the provided HTML as needed.**

## Home Page & Navigation (25 points)

Get familiar with the provided **html & css** and create an application that allows users to buy flight tickets. An anonymous user should be able to access **register**/**login** and **flights**. Your **application** should also have **administrators**. Admins can **create** flights and change flight **properties**.The **home/flights** page should list all **public** **flights** in the database (if the current **user** is **admin** all **public** and **private** flights should be **listed**). Only **authenticated** users can **book** a ticket. If the user is **not** authenticated **redirect** him to the login page. Admins see additional functionality on the front page to **add** a new flight (see provided HTML).

Make sure only the **necessary** elements are displayed in the navigation menu and on every page.





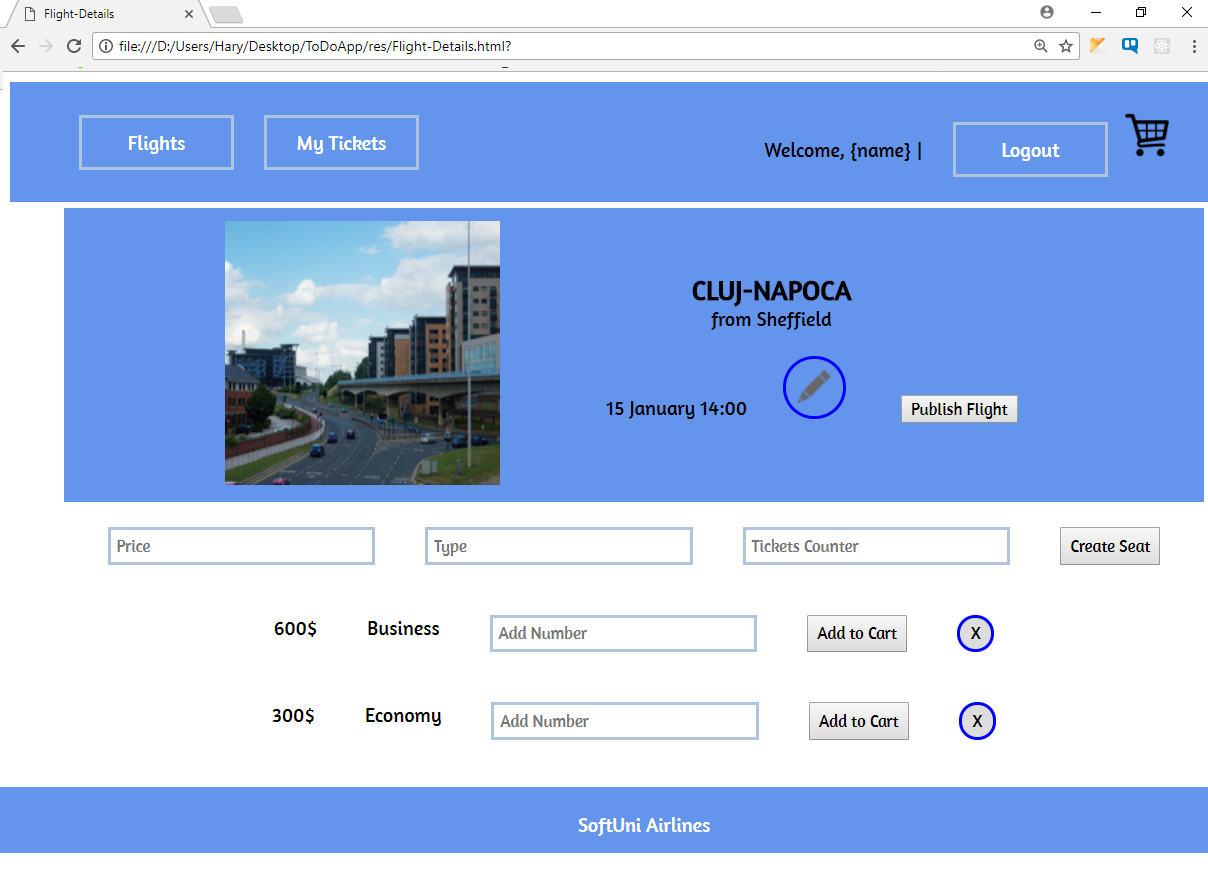
## Problem 2. Create a Flight (For admins only) (15 points)

Only **admins** have the ability to **create** a flight, by accessing the create flight menu from the ' **Add Flight** ' button on the home page. Each flight has an **origin, destination, departure date and time, image (img url) and public flag (Boolean).** When a flight is created, it’s public flag is set to **false**.

## Problem 3. Flight Details (For authenticated users) (10 points)

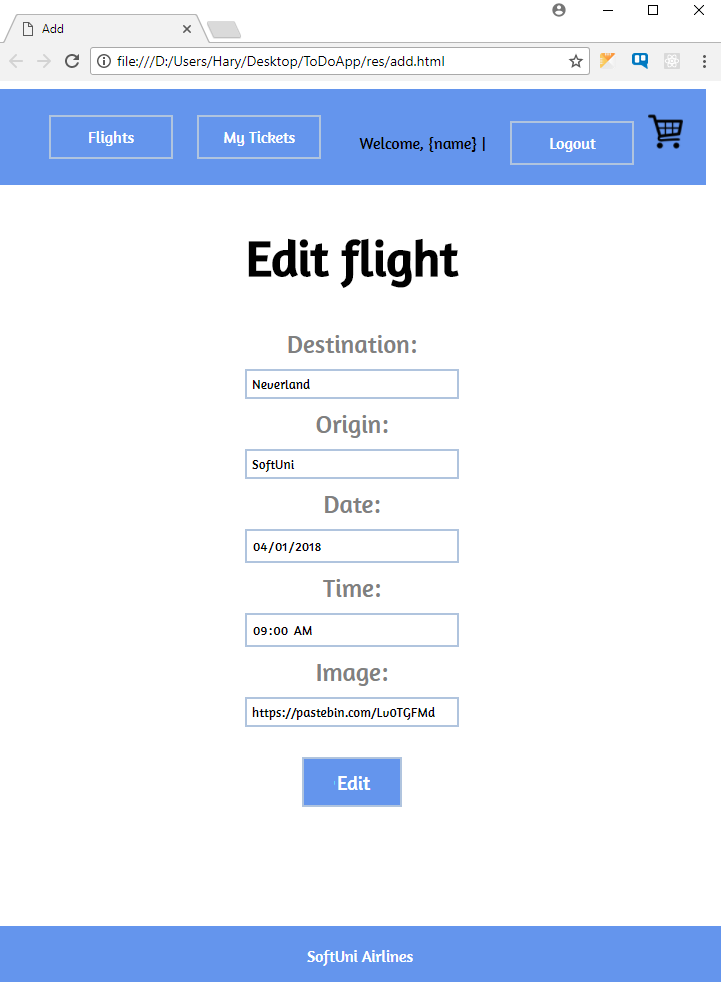
By **clicking** on a **flight** (in the home page view) your app should **open** **details** page of the selected flight, which **displays** the **origin**, **destination**, **departure** time, **image** and list of all **available** tickets. Only admins can see flights that are marked as private (**public** flag set to **false**).

If the current user is **admin**, additional **functionality** should be **available** to **add** or **remove** seats and to **edit** the current flight.



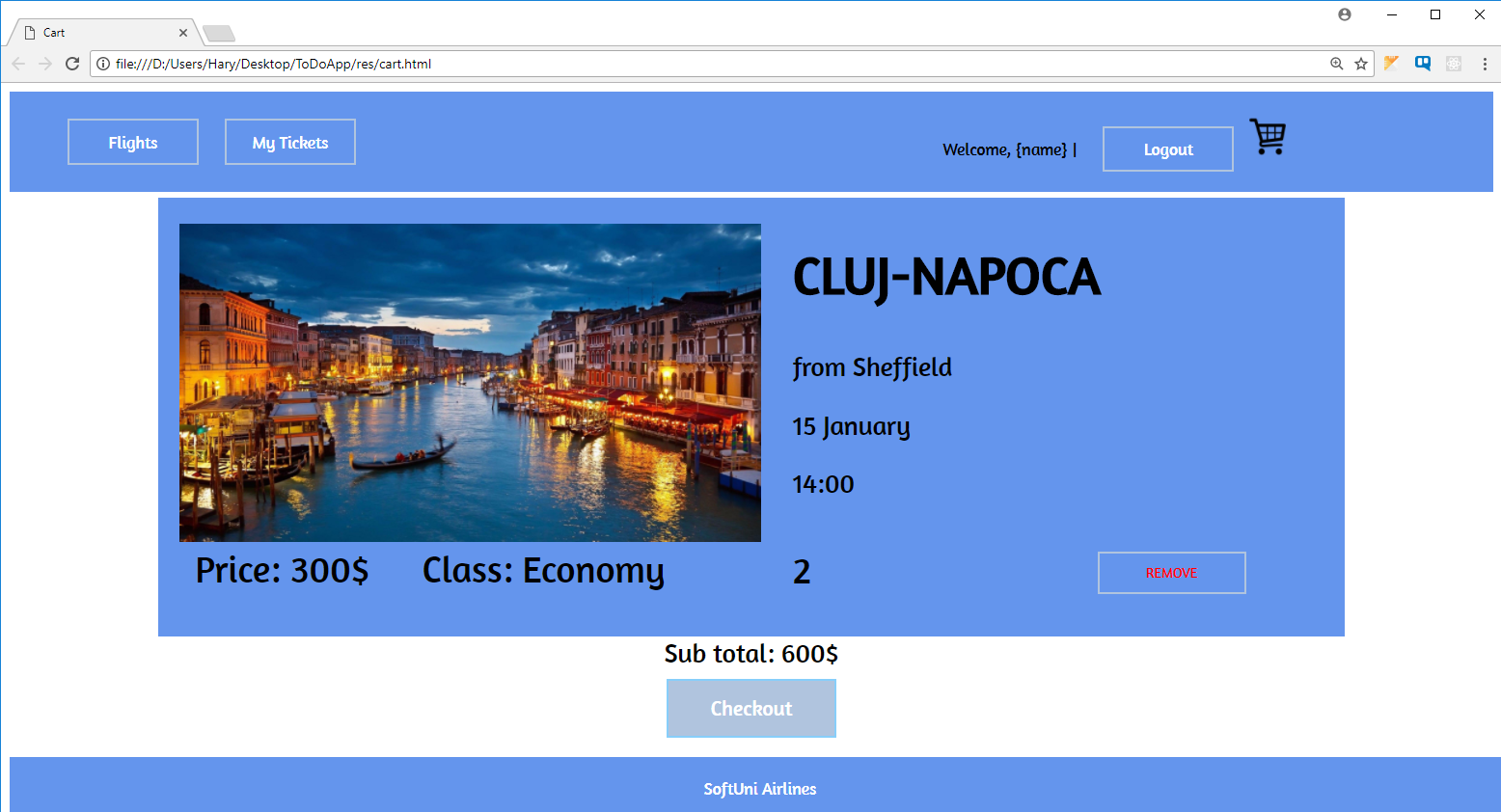
## Problem 4. Edit Flight (For admins only) (25 points)

Each Admin should be able to add **seats**, with different **count**, **price** and **type** or **delete existing ones** (see the **picture** from **problem 3**). Admins must also be able to **publish** flights (changestatusfrom **private to public**) and **edit** them (change flight **destination**, **departure** time, **origin** or **image**).



## Problem 5. Book a Ticket (For authenticated users) (15 points)

All **authenticated** users can **book** tickets for a given **flight** by clicking the [**Add to cart**] button in flight details page (**problem 3 image**). The user should be able to choose the **count** of seats, that he wants to book. After that the user should be able to **pay** for his **tickets**, by opening his **shopping cart**.



## Problem 6. My Flights (For authenticated users) (10 points)

Each user can view their booked (only **purchased**) tickets by clicking [**My Tickets**]. The following view should **display** all tickets, owned by the user and **all** available **flight** **information** about them.

**OTHER REQUIREMENTS**

* Use **Node.js** as a web server
* Use **MongoDB** as a database storage
* Use **Express.js** as a routing framework
* You may use whatever frameworks you like