**Client.js**

// Q1 replace the ????? so that this event listener handles click events on

// the 'search for flight' button

document.getElementById('btnFlightSearch').addEventListener('click', () => {

    // Q2 complete these statements to read destination and date from form

    const city = document.getElementById('theDest').value;

    const date = document.getElementById('theDate').value;

    // Q3 complete the fetch API call to send the user's chosen city and

    // date to the 'flight search' route in server.js. You will need to look

    // at the server.js code to complete this successfully.

    fetch(`/search/${city}/${date}`).then(response => response.json())

        .then(json => {

            console.log(`Flight from Destination ${city} is on the ${date}`);

        // Q6 complete so that it parses the JSON returned and outputs the

        // data to the searchResults <div> in the format shown on the paper.

            renderFlightSearchResultToHTML(json);

        // Q8 update with a book button - see question paper

        document.createElement('button').addEventListener('click', () => {

            fetch(`/flightbook/${json[0].ID}`)

        })

    });

// Q9 replace the ????? so that this event listener handles click events on

// the 'add flight' button

// Note the event listener has been setup to be an async function - this may help you

document.getElementById('btnFlightAdd').addEventListener('click', async() => {

    // Q9 complete these statements to read flight details from the form

    const number = document.getElementById('fNumber').value;

    const dest = document.getElementById('destCity').value;

    const date = document.getElementById('theDate2').value;

    const deptime = document.getElementById('departTime').value;

    const arrtime = document.getElementById('arriveTime').value;

    const thePrice = document.getElementById('price').value;

    const numSeats = document.getElementById('numSeats').value;

    const flightStatusDiv = document.getElementById('flightAddStatus');

    // Q9 complete the fetch API call to send the data to the 'add flight'

    // route on the server as a POST request...

    newFlight = {

        number: number,

        dest: dest,

        date,

        deptime,

        arrtime,

        thePrice,

        numSeats

    }

    fetch('/flightadd', {

        method: 'POST',

        headers: {

            'Content-Type': 'application/json'

        },

        body: JSON.stringify(newFlight)

    })

        .then(alert(`Title ${newFlight.number} addedd successfully:\n ${JSON.stringify(newFlight)}`))

    // .then()

    .catch(error => {

        const errorP = document.createElement('p');

        p.innerHTML = `<p>Error happened: ${error}</p>`;

        flightStatusDiv.appendChild(errorP);

    });

    // Q12 modify Q9 answer to handle non-200 status codes. Ensure that

    // user-friendly error messages are displayed to the user in the

    // 'flightAddStatus' <div>.

});

// Q13 replace the ????? so that this event listener handles click events on

// the login button

// Note the event listener has been setup to be an async function - this may help you

document.getElementById('btnLogin').addEventListener('click', async() => {

    // Q13 complete these statements to read login details from the form

    const user = document.getElementById('user').value;

    const password = document.getElementById('pass').value;

    const loginStatusDiv = document.getElementById('loginStatus');

    // Q13 complete the fetch API call to send the data to the login

    // route on the server as a POST request...

    user = {

        uname: user,

        password

    }

    fetch('/login', {

        method: 'POST',

        headers: {

            'Content-Type': 'application/json'

        },

        body: JSON.stringify(user)

    }).catch(error => {

        const errorP = document.createElement('p');

        p.innerHTML = `<p>Error happened: ${error}</p>`;

        loginStatusDiv.appendChild(errorP);

    })

    // Q14 modify Q13 answer so that if the user did not log in correctly,

    // a user-friendly error message is displayed to the user in the

    // 'loginStatus' <div>.

});

function renderFlightSearchResultToHTML(data) {

    const nbFlights = data.length;

    var html = `<p> <b>${nbFlights}</b> flight(s) found:</p>`;

    if (nbTitles) {

        html += '<ul>';

        data.forEach(flight => {

            html += `<li>

            <b>Flight number:</b> ${flight.fNumber}<br/>

            <b>Departure:</b> ${flight.depart}<br/>

            <b>Arrival:</b> ${flight.arrive}</br>

            <b>Price:</b> ${flight.price}</br>

            <input type='button' id='bookFlight-${flight.ID}' value='Book Flight' />

            </li>`;

        });

        html += '</ul>';

    }

    document.getElementById('searchResults').innerHTML = html;

}

**Server.js**

const express = require('express');

const app = express();

const sqlite3 = require('sqlite3').verbose();

// Q14 you need to add the full code to use session variables to enable a login system.

const conn = new sqlite3.Database('euroair.db', err => {

    if (err) {

        console.error(err);

    }

    console.log('Connected to db');

});

app.get('/', (req, res) => {

    res.send('Hello World from Express!');

});

// Q4 complete the route to find all flights to the user's chosen

// destination on the user's chosen date

app.get('/search/:flightDestination/:flightDate', (req, res) => {

    try {

        const stmt = 'SELECT \* FROM eaflights WHERE endcity=? AND date=?';

        conn.get(stmt, [req.params.flightDestination, req.params.flightDate], (err, row) => {

            res.json(row);

        })

    } catch(error) {

        res.status(500).json(error)

    }

});

// Q7 complete the route to book the flight for 1 passenger and a hard-coded

// username by inserting a record in the bookings table

app.post('/flightbook/:flightId', (req, res) => {

    conn.run("INSERT INTO bookings(uname, password) VALUES (userName,?)", [req.params.flightId],

        (err, results, fields) => {

            if(err) {

                res.status(500).json({'error': 'Internal error'});

            } else {

                res.json({'success': 1});

            }

        });

});

app.post('/flightadd', (req, res) => {

    // Q11 send back an error if any of the details are blank (you need to add this code...)

    const params = req.params.newFlight;

    Object.keys(params).forEach(param => {

        if (param === null || param === undefined) {

            res.status(500).json({ error: 'Fill in all parameters'})

            return;

        }

    })

    // Q10 complete the 'add flight' route as described in the paper

    conn.run('INSERT INTO eaflights(fnumber, endcity, date, depart, arrive, price, nseats) VALUES (?,?,?,?,?,?,?)',

        [params.number, params.dest, params.date, params.deptime, params.arrtime, params.thePrice, params.numSeats],

        (err, results, fields) => {

            if(err) {

                res.status(500).json({error: 'Internal error'});

            } else {

                res.json({'success': 1});

            }

        });

});

// Q14 complete the login route on the server

app.post('/login', (req, res) => {

    // see week 8

});

app.listen(3002, () => {

    console.log('server is online.');

});