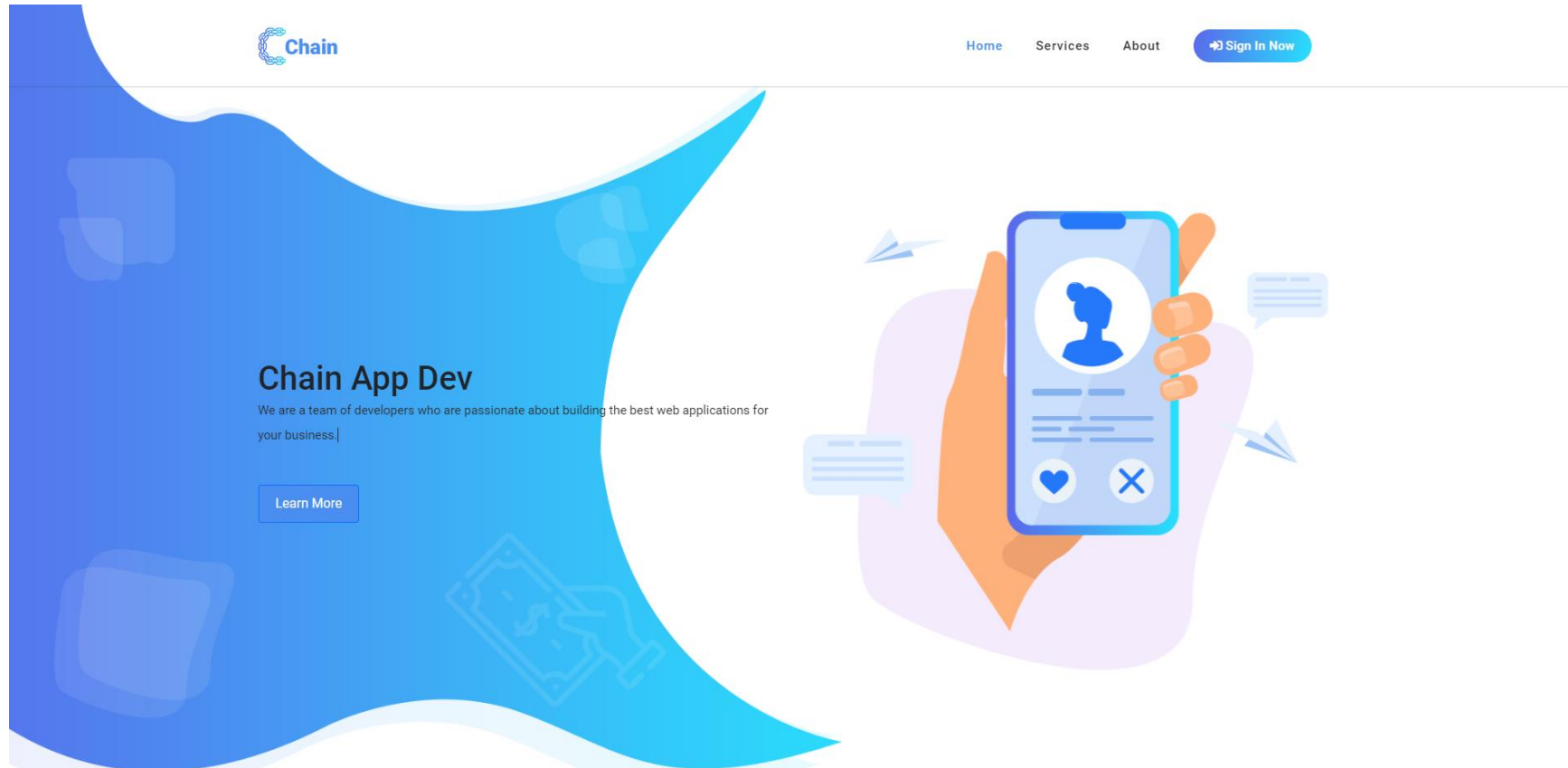




LAB 4 ASSIGNMENT

Daniel Krasovski – C18357323

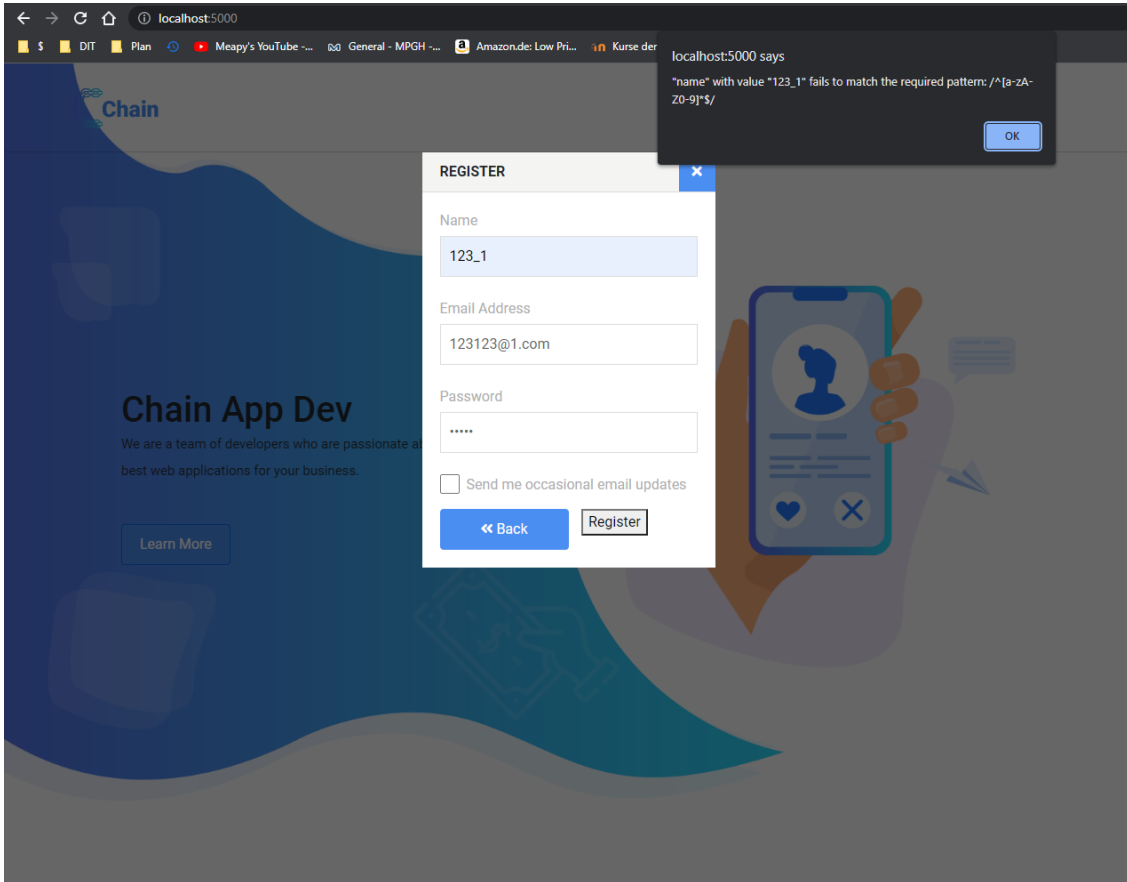
HOMEPAGE



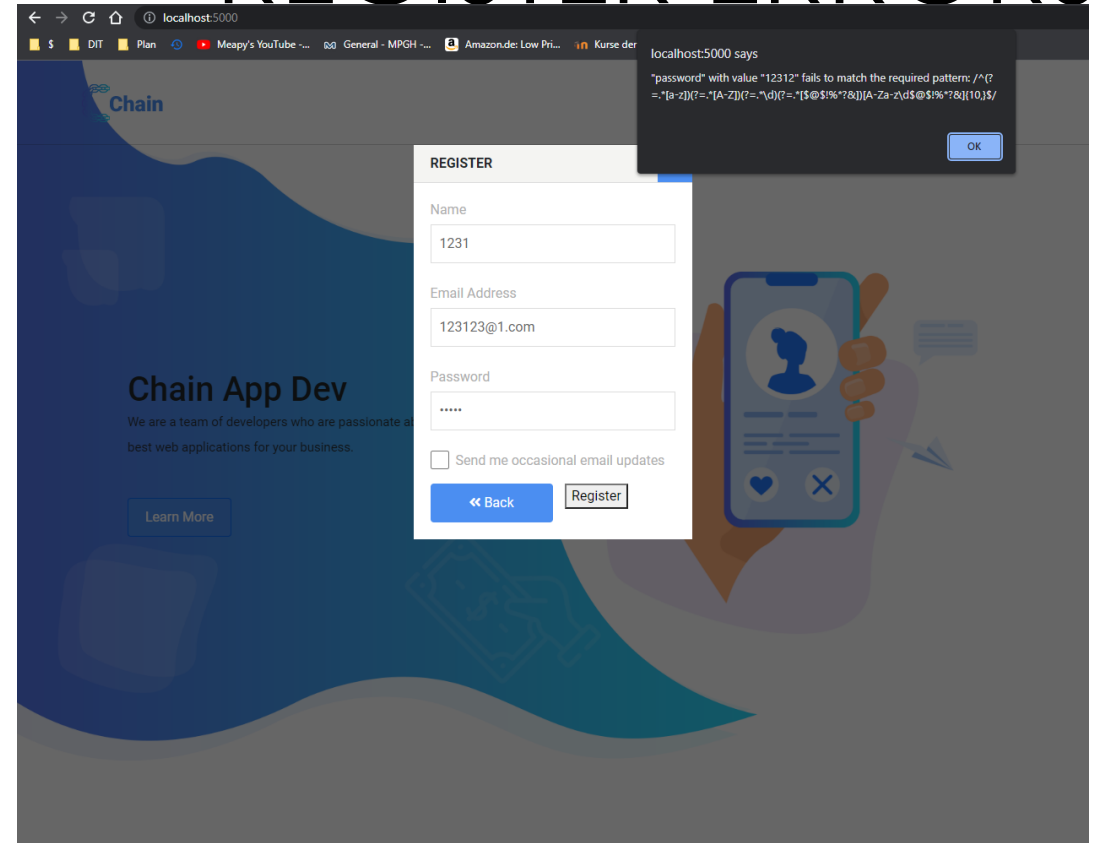
Bootstrap Template from:

<https://themewagon.com/themes/chain-free-bootstrap-5-html5-landing-page-template/>

REGISTER ERRORS

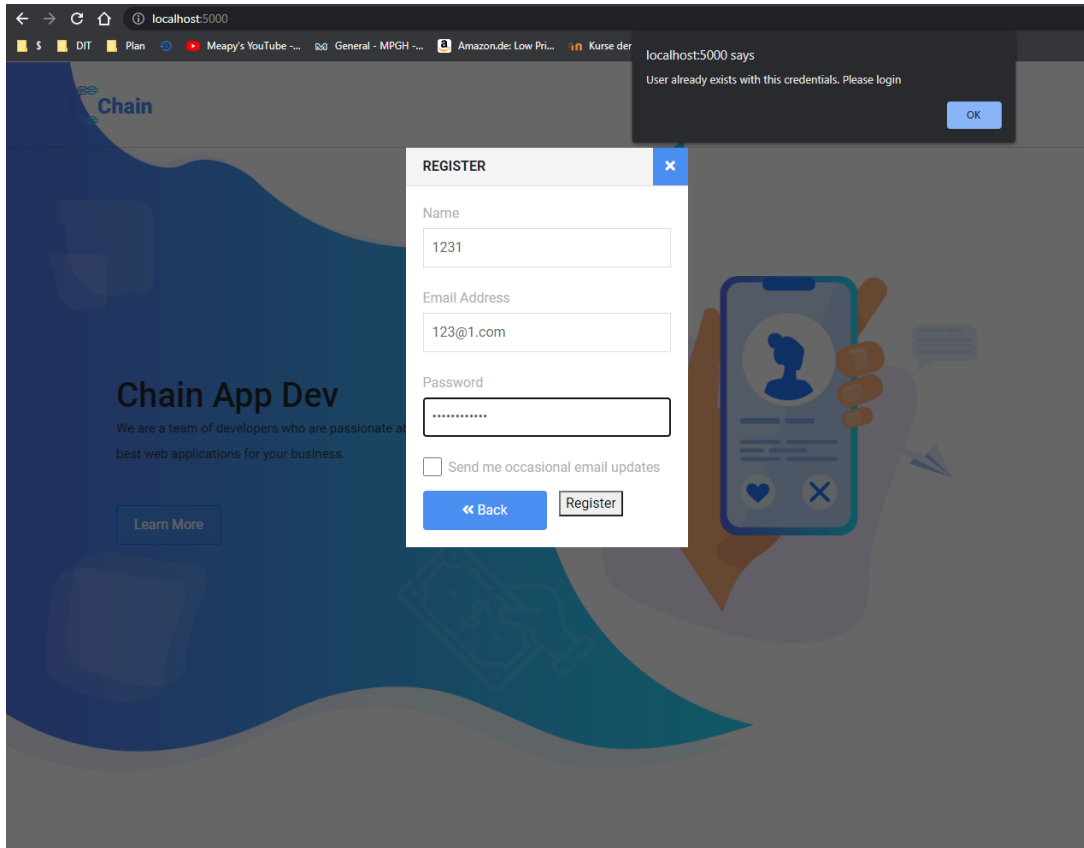


Username doesn't only contain numbers and letters



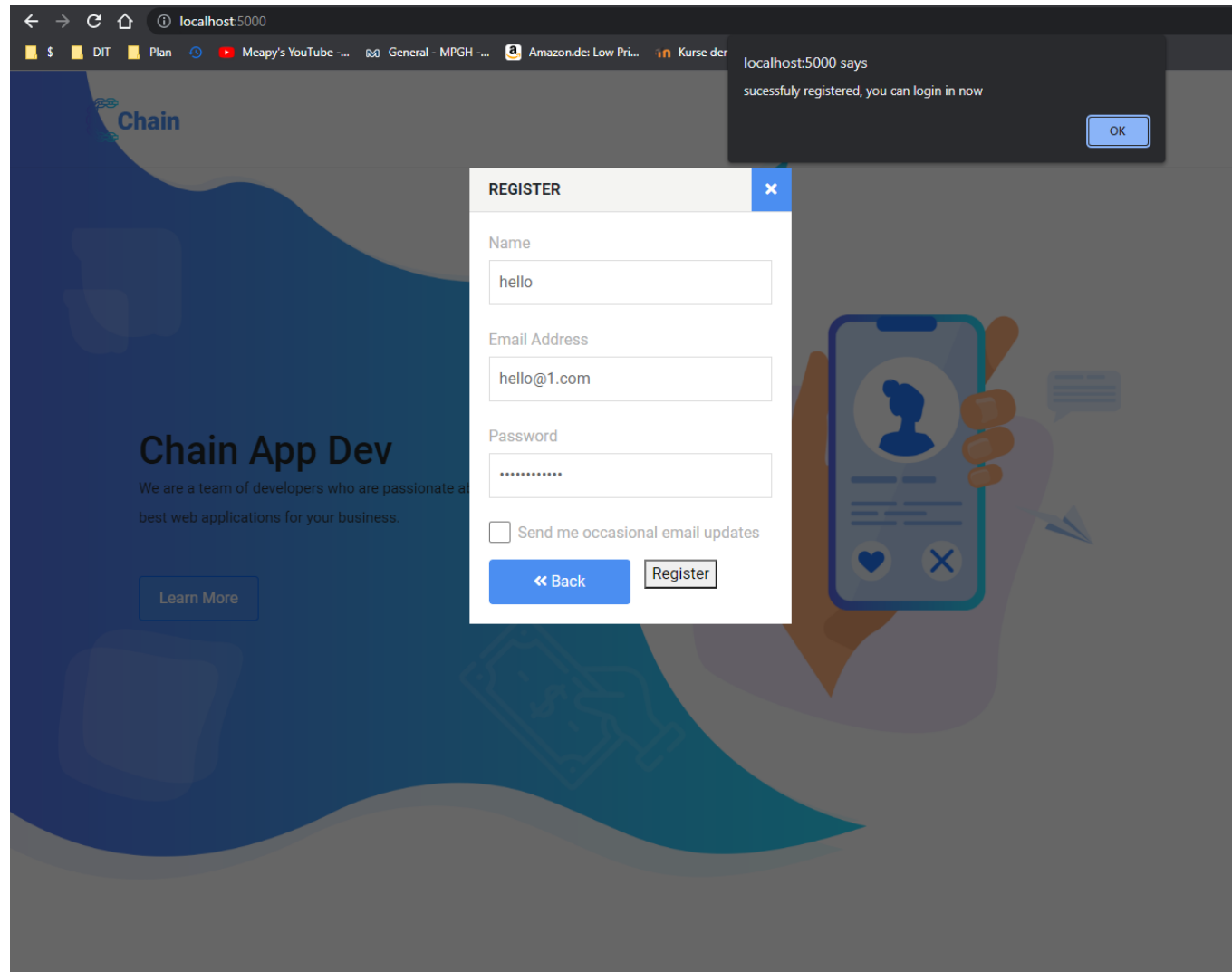
Password doesn't match the requirements

REGISTER ERRORS



Email already in use

REGISTER SUCCESS



LOGIN

Chain

Home Services About Sign In Now

Chain App Dev

We are a team of developers who are passionate about creating the best web applications for your business.

Learn More

LOGIN

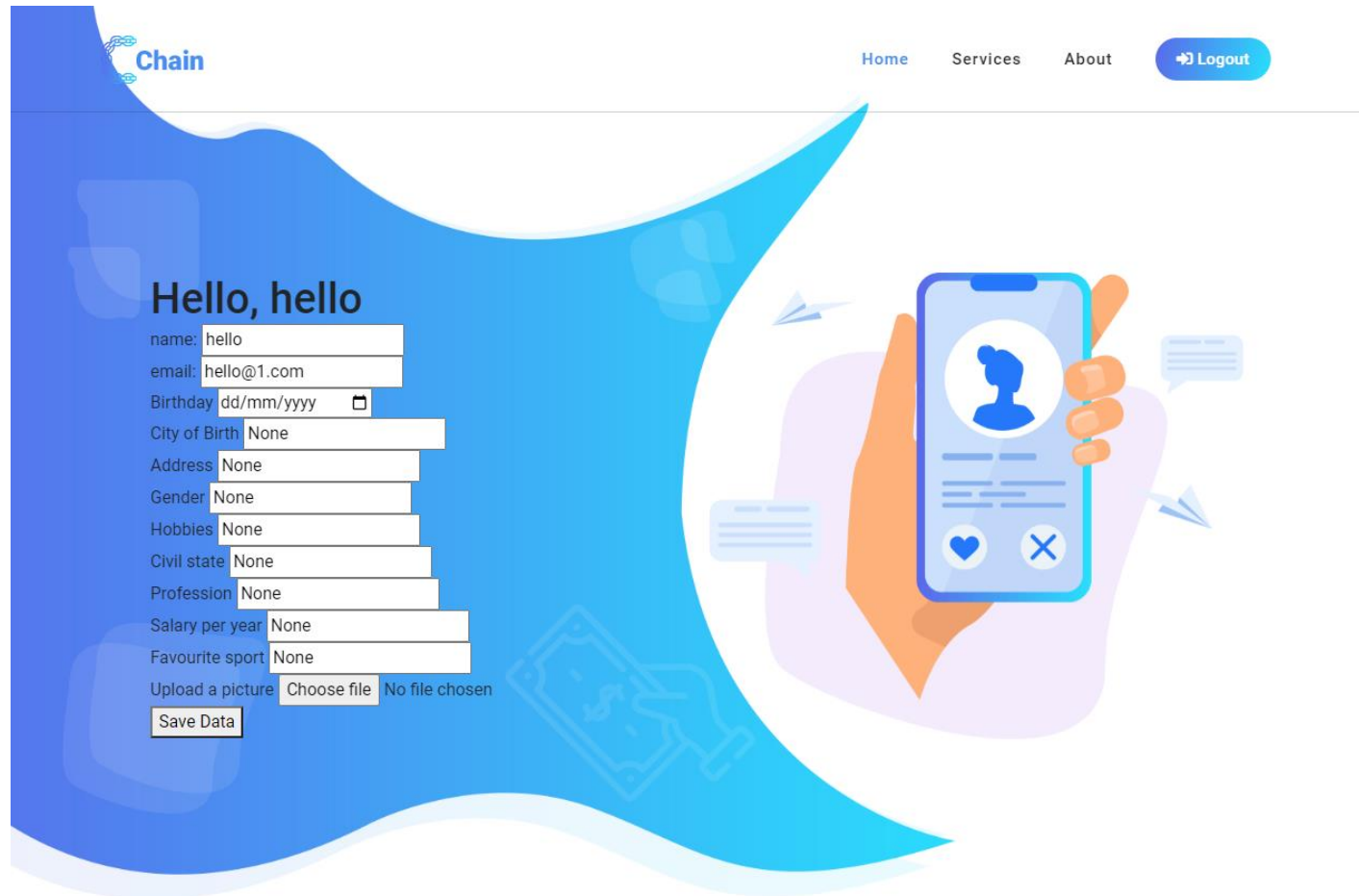
Email

Password

<< Back Login

[Forgot password?](#)

ONCE LOGGED IN



The screenshot shows a web application interface. At the top left is a logo with a chain icon and the text "Chain". To the right of the logo are navigation links: "Home", "Services", "About", and a blue "Logout" button with a right-pointing arrow. The main content area has a large blue abstract shape on the left. Inside this shape, the text "Hello, hello" is displayed. Below it is a form with the following fields: "name:" with value "hello", "email:" with value "hello@1.com", "Birthday" with a date picker showing "dd/mm/yyyy", "City of Birth:" with value "None", "Address:" with value "None", "Gender:" with value "None", "Hobbies:" with value "None", "Civil state:" with value "None", "Profession:" with value "None", "Salary per year:" with value "None", "Favourite sport:" with value "None", and "Upload a picture:" with a "Choose file" button and the text "No file chosen". At the bottom of the form is a "Save Data" button. To the right of the blue shape is a large illustration of a hand holding a smartphone. The phone screen shows a profile card with a blue silhouette of a person's head and shoulders, and two buttons at the bottom: a heart icon and a close (X) icon. There are also some faint, stylized icons of a paper plane and a speech bubble around the phone.

Chain

Home Services About Logout

Hello, hello

name: hello

email: hello@1.com

Birthday dd/mm/yyyy

City of Birth: None

Address: None

Gender: None

Hobbies: None

Civil state: None

Profession: None

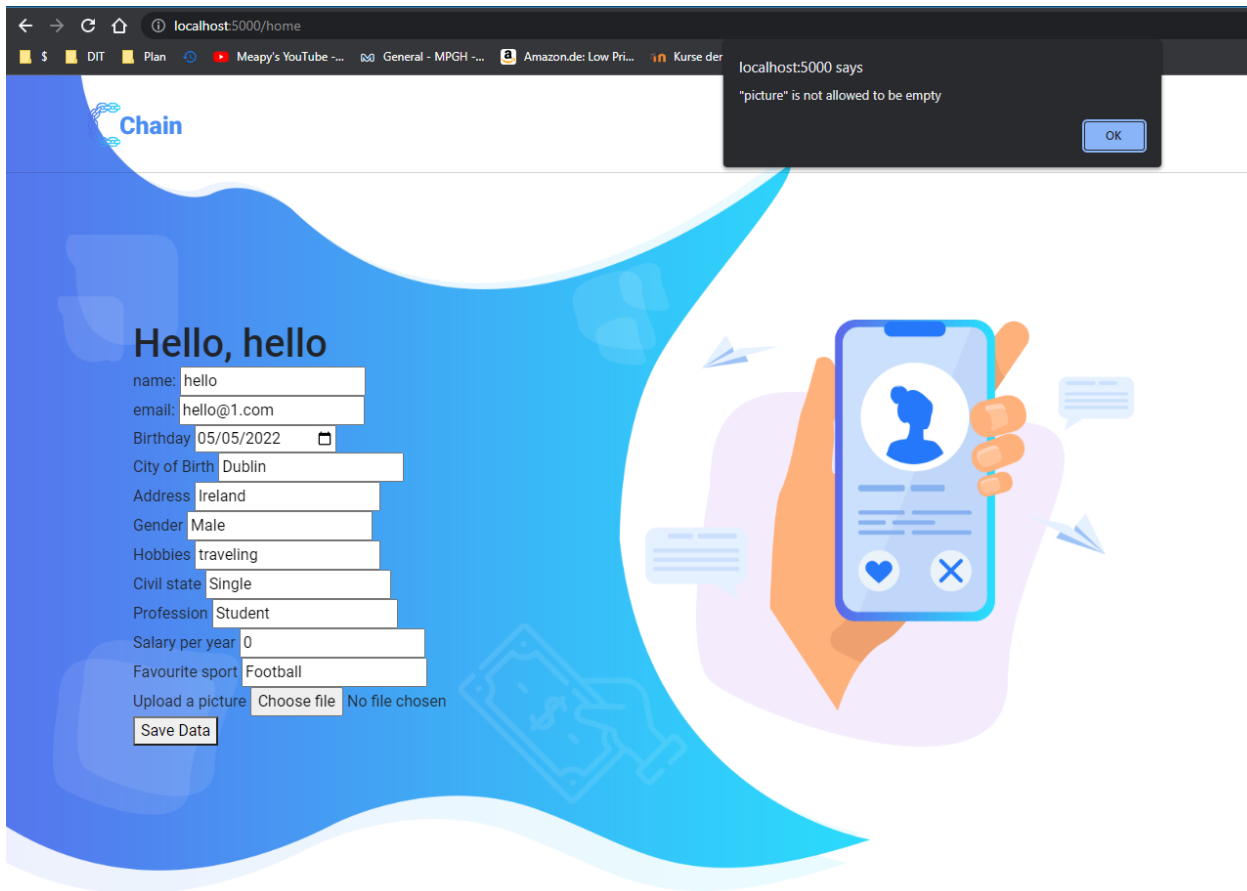
Salary per year: None

Favourite sport: None

Upload a picture Choose file No file chosen

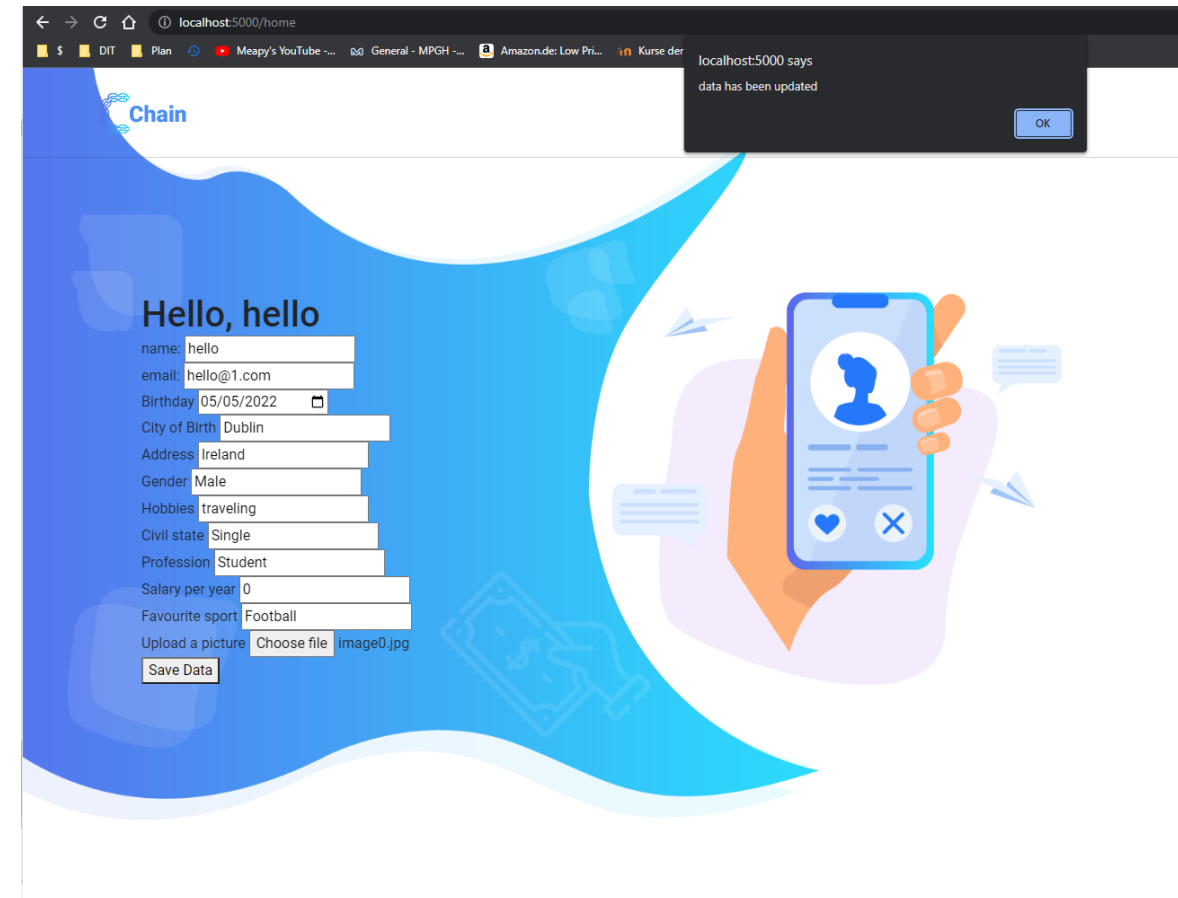
Save Data

All fields must be filled in



The screenshot shows a web browser window with the address bar displaying 'localhost:5000/home'. The browser's tab bar includes several tabs: '\$', 'DIT', 'Plan', 'Meapy's YouTube ~...', 'General - MPGH ~...', 'Amazon.de: Low Pri...', and 'Kurse der'. A dark notification box at the top right of the page contains the text 'localhost:5000 says "picture" is not allowed to be empty' and an 'OK' button. The main content area features a blue wavy background with a 'Chain' logo in the top left. Below the logo, the text 'Hello, hello' is displayed. A form with various input fields is shown, including 'name: hello', 'email: hello@1.com', 'Birthday: 05/05/2022', 'City of Birth: Dublin', 'Address: Ireland', 'Gender: Male', 'Hobbies: traveling', 'Civil state: Single', 'Profession: Student', 'Salary per year: 0', and 'Favourite sport: Football'. At the bottom of the form, there is an 'Upload a picture' section with a 'Choose file' button and the text 'No file chosen'. A 'Save Data' button is located at the bottom left of the form. To the right of the form, there is an illustration of a hand holding a smartphone displaying a profile card with a heart and a close button icon.

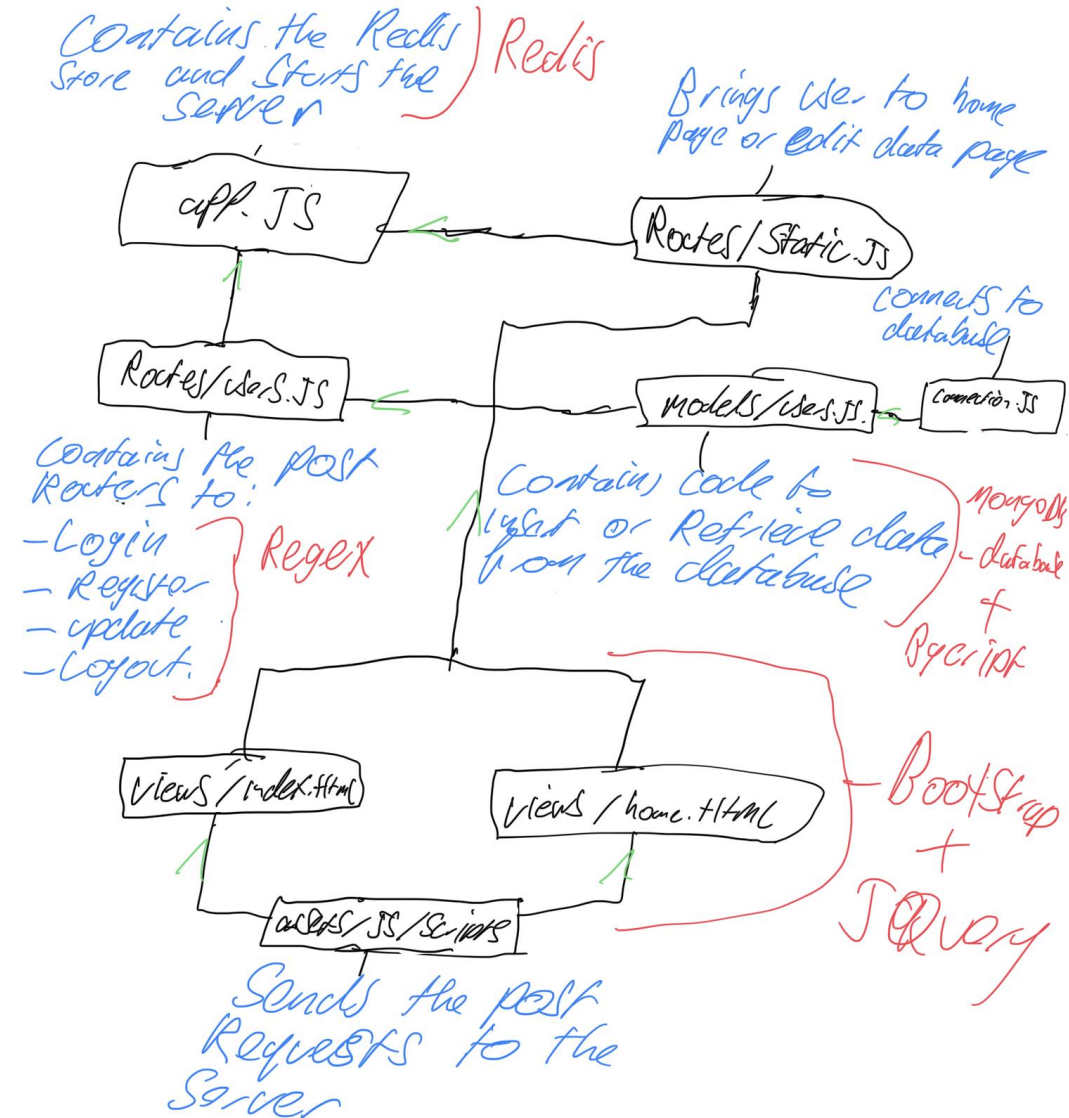
SAVING DATA



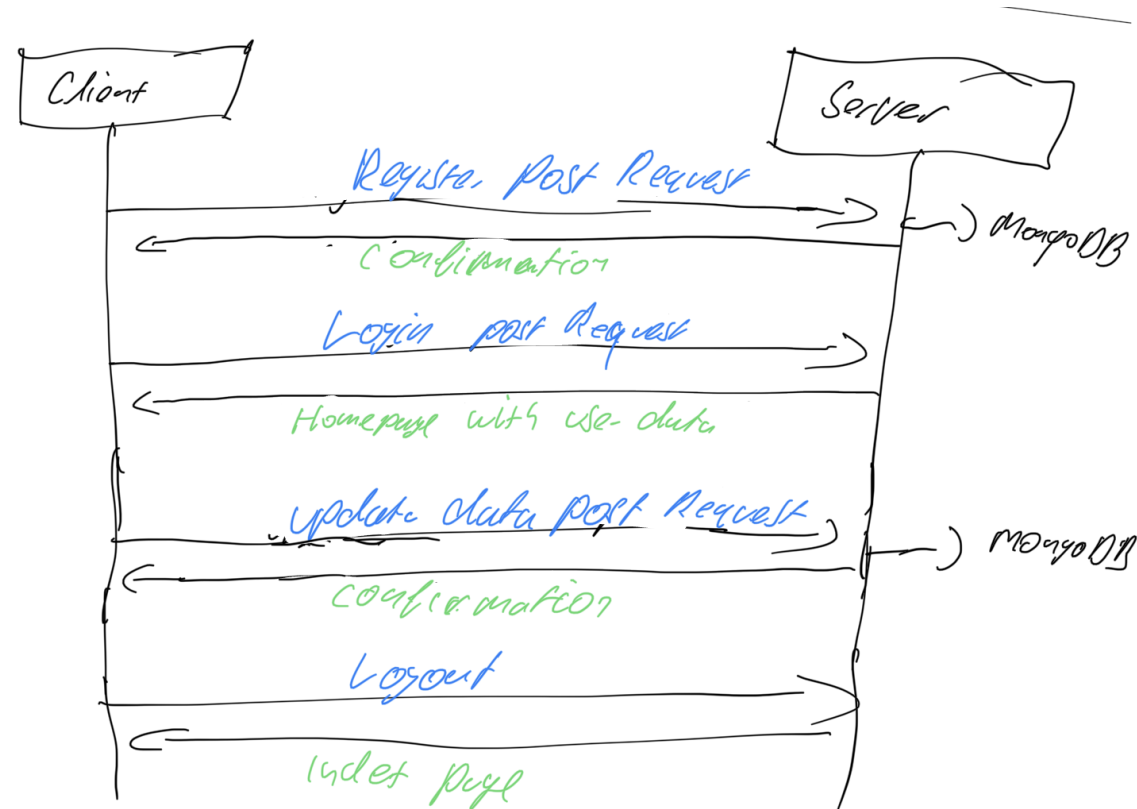
The screenshot shows the same web browser window as the previous one, but with a different notification box. The notification box at the top right now says 'localhost:5000 says data has been updated' and has an 'OK' button. The form content is identical to the previous one, but the 'Upload a picture' section now shows 'image0.jpg' next to the 'Choose file' button. The 'Save Data' button is still present at the bottom left. The illustration of the hand holding the smartphone remains on the right side of the form.

Success on saving data

STRUCTURE DIAGRAM



INTERACTION DIAGRAM





PARTS COMPLETED

- Everything has been completed

SCREENSHOTS

MongoDB

The screenshot shows the MongoDB Compass web interface. At the top, there's a 'Documents' tab for the 'userDemo.users' collection. Below this, there are tabs for 'Documents', 'Aggregations', 'Schema', 'Explain Plan', 'Indexes', and 'Validation'. The 'Documents' tab is active, showing a list of documents. The first document has fields: _id, name, email, password, address, city, civilS, dob, gender, hobbies, picture, salary, and sport. The second document has similar fields but with different values. The interface includes a filter bar, a 'FIND' button, and a 'REFRESH' button.

```
{ "_id": ObjectId("626ef57cc6d317bfe219c6cf"), "name": "123", "email": "123@1.com", "password": "$2b$15$grery6SO6zWZggPxemERfOU3YkrLNUS00TOtqQdXKpbe//012J/G", "address": "123", "city": "City", "civilS": "None", "dob": "2022-05-12", "gender": "tes", "hobbies": "None", "picture": "C:\\fakepath\\image0.jpg", "salary": "None", "sport": "None" }
```

```
{ "_id": ObjectId("627064a8268de25ceecd2e80"), "name": "12312", "email": "12321@1.com", "password": "$2b$15$icV6HkrOvy9H7ckeazIFfuVja4p.EOpZyN.YkXESOrjy4jJsnWs/O", "address": "Dublin", "city": "Vilnius", "civilS": "Single Pringle", "dob": "2022-05-20", "gender": "Male", "hobbies": "Money123", "picture": "C:\\fakepath\\image0.jpg", "salary": "100000", "sport": "F1" }
```

Update user code

```
function updateUser(userData) {  
  return new Promise(async (resolve, reject) => {  
    try {  
      let checkUserData = await checkIfUserExists({ email: userData.email });  
      if (checkUserData.data && checkUserData.data.length > 0) {  
        let updateUserData = await mongoConnection  
          .collection("users")  
          .updateOne(  
            { email: userData.email },  
            {  
              $set: {  
                name: userData.name,  
                email: userData.email,  
                dob: userData.dob,  
                city: userData.city,  
                address: userData.address,  
                gender: userData.gender,  
                hobbies: userData.hobbies,  
                civilS: userData.civilS,  
                salary: userData.salary,  
                picture: userData.picture,  
                sport: userData.sport,  
              }  
            },  
            (err, results) => {  
              if (err) {  
                console.log(err);  
                throw new Error(err);  
              }  
              resolve({  
                error: false,  
                data: results,  
              });  
            }  
          );  
      }  
    }  
  });  
}
```

REGISTER CODE

Server side

```
router.post("/register", async (req, res) => {
  try {
    const schema = joi.object().keys({
      //name can only contain letters and numbers
      name: joi.string().regex(/^[a-zA-Z0-9]*$/).required(),
      email: joi.string().email().required(),
      // password has to have 10 symbols and it is required to have at least one number, one Upper letter and one symbol between [$,%,&,@].
      password: joi.string()
        .regex(/^(?=[a-z])(?=.*[A-Z])(?=.*\d)(?=.*[$@!%*?&])[A-Za-z\d$@!%*?&]{10,}$/)
        .required(),
    });
    const result = schema.validate(req.body);
    if (result.error) {
      throw result.error.details[0].message;
    }
    let checkUserRegister = await models.registerUser(result.value);
    if (checkUserRegister.error) {
      throw checkUserRegister.message;
    }
    res.json(checkUserRegister);
  } catch (e) {
    console.log(e);
    res.json({ error: true, message: e });
  }
});
```

Client side post request sending

```
$('.register').click(function (event) {
  event.preventDefault();
  var name = $('#nameRegister').val();
  var email = $('#emailRegister').val();
  var password = $('#passwordRegister').val();
  $.ajax({
    url: '/users/register',
    type: 'POST',
    data: {
      name: name,
      email: email,
      password: password
    },
    function(data) {
      if (data.error) {
        alert(data.error);
      } else {
        alert(data.success);
        window.location.href = '/';
      }
    }
  });
});
```

SCREENSHOTS

Register user with Bcrypt for password

```
function registerUser(userData) {
  return new Promise(async (resolve, reject) => {
    try {
      // check if user does not exists
      let checkUserData = await checkIfUserExists({ email: userData.email });
      if (checkUserData.data && checkUserData.data.length > 0) {
        // user already exists, send response
        return resolve({
          error: true,
          message: "User already exists with this credentials. Please login",
          data: [],
        });
      }
      // generate password hash
      let passwordHash = await bcrypt.hash(userData.password, 15);
      userData.password = passwordHash;

      // add new user
      mongoConnection
        .collection("users")
        .insertOne(userData, async (err, results) => {
          if (err) {
            console.log(err);
            throw new Error(err);
          }
          //return data
          resolve({
            error: false,
            data: results.ops,
          });
        });
    } catch (e) {
      reject(e);
    }
  });
}
```

Session with Redis

```
// connect to redis session store
const redisSessionStore = redis.createClient(
  nconf.get("redisPort"),
  nconf.get("redisHost"),
  {
    db: 0,
  }
);

redisSessionStore.on("connect", () => {
  console.log(
    `${chalk.green("✓")} Connected to ${chalk.green("Redis")} Session Store`
  );
});

app.use(bodyParser.json());
app.use(bodyParser.urlencoded({ extended: false }));

// session store
app.use(
  session({
    secret: nconf.get("sessionSecret"),
    cookie: {
      maxAge: 1000 * 60 * 60 * 24 * 7, // 1 week
    },
    store: new redisStore({ client: redisSessionStore }),
    resave: false,
    saveUninitialized: false,
  })
);
```


SCREENSHOTS

Default values for user data

```
router.post("/login", async (req, res) => {
  try {
    const schema = joi.object().keys({
      email: joi.string().email().required(),
      password: joi.string().min(6).max(20).required(),
    });
    const result = schema.validate(req.body);
    console.log(result.value);
    if (result.error) {
      throw result.error.details[0].message;
    }
    let checkUserLogin = await models.verifyUser(result.value);

    if (checkUserLogin.error) {
      throw checkUserLogin.message;
    }
    // set session for the logged in user
    req.session.user = {
      name: checkUserLogin.data.name || "None",
      email: checkUserLogin.data.email || "None",
      dob: checkUserLogin.data.dob || "None",
      city: checkUserLogin.data.city || "None",
      address: checkUserLogin.data.address || "None",
      gender: checkUserLogin.data.gender || "None",
      hobbies: checkUserLogin.data.hobbies || "None",
      civilS: checkUserLogin.data.civilS || "None",
      job: checkUserLogin.data.job || "None",
      salary: checkUserLogin.data.salary || "None",
      picture: checkUserLogin.data.picture || "None",
      sport: checkUserLogin.data.sport || "None",
    };
    res.json(checkUserLogin);
  } catch (e) {
    res.json({ error: true, message: e });
  }
});
```

Edit data page

Chain

Home Services About Logout

Hello, hello

name:

email:

Birthday:

City of Birth:

Address:

Gender:

Hobbies:

Civil state:

Profession:

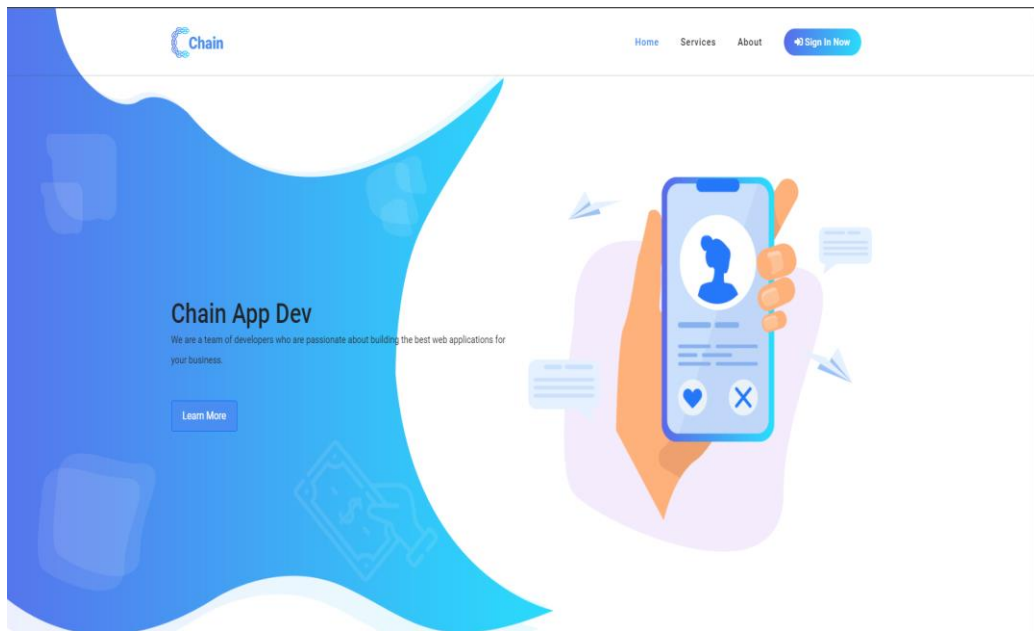
Salary per year:

Favourite sport:

Upload a picture: No file chosen

SCREENSHOTS

Home page



Login or sign up

