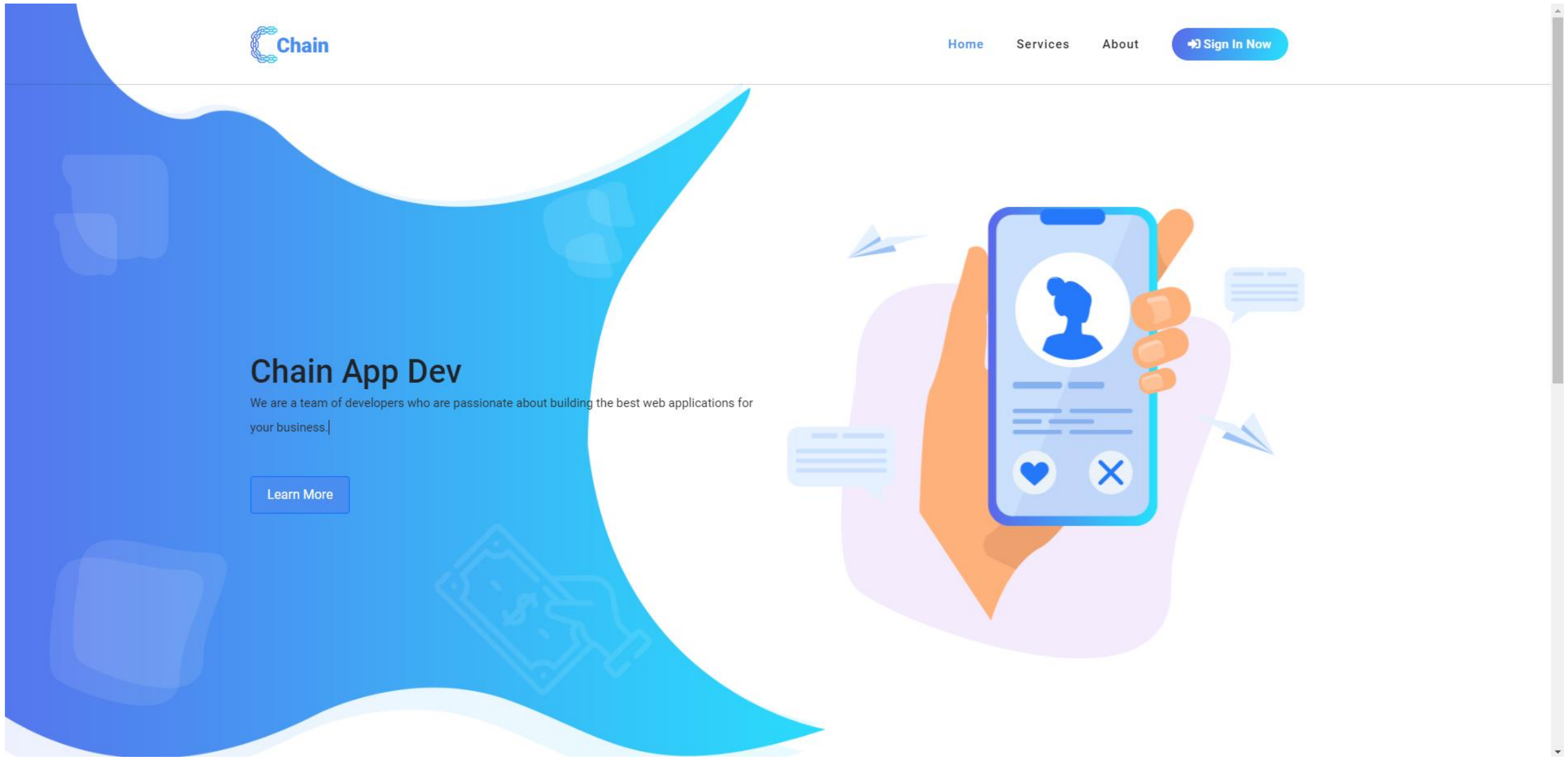
LAB 4 ASSIGNMENT

Daniel Krasovski – C18357323

HOMEPAGE

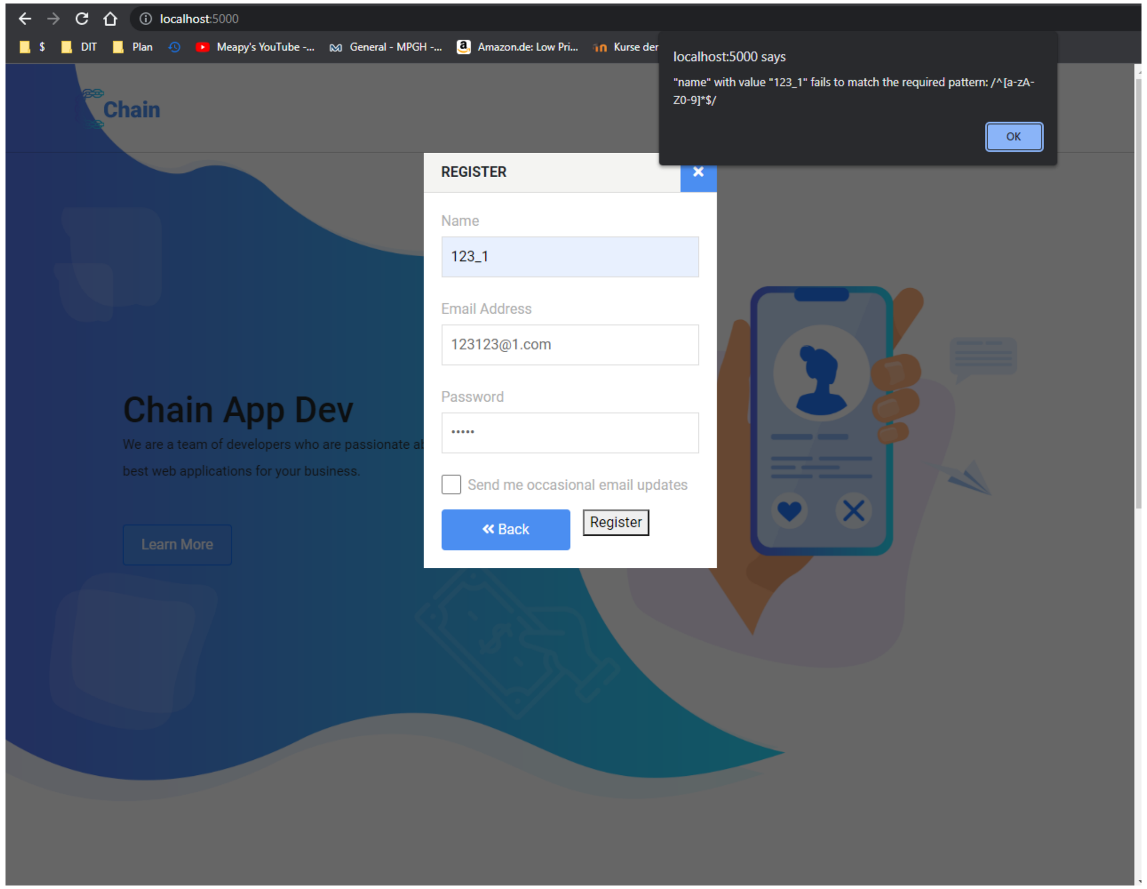


Chain App Dev

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

Bootstrap Template from:
https://themewagon.com/themes/chain-free-bootstrap-5-html5-landing-page-template/

REGISTER ERRORS



```
localhost:5000 says
"name" with value "123_1" fails to match the required pattern: /^[a-zA-Z0-9]*$/
```

REGISTER

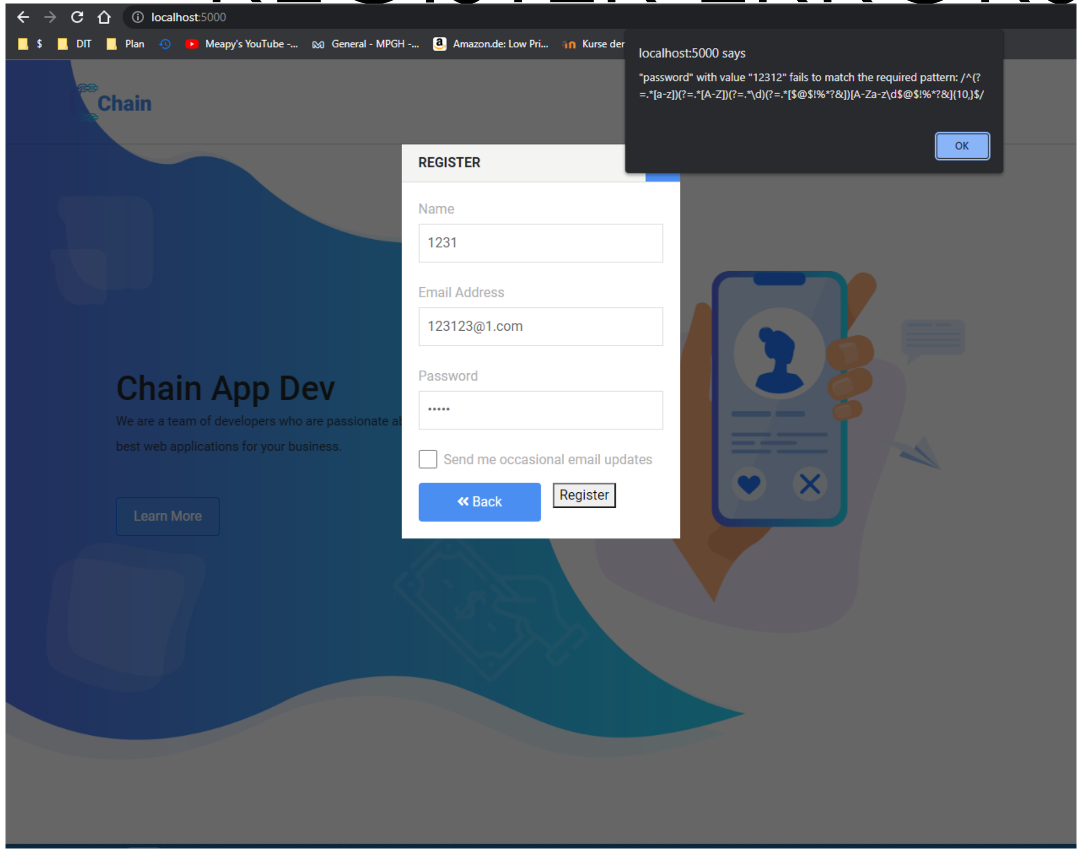
Name: 123_1

Email Address: 123123@1.com

Password: •••••

[ ] Send me occasional email updates

localhost:5000
OK
Chain
Chain App Dev
We are a team of developers who are passionate a
best web applications for your business.
Learn More
« Back
Register



```
localhost:5000 says
"password" with value "12312" fails to match the required pattern: /^(?=.*[a-z])(?=.*[A-Z])(?=.*\d)(?=.*[$@$!%*?&])[A-Za-z\d$@$!%*?&]{10,}$/
```

REGISTER

Name: 1231

Email Address: 123123@1.com

Password: •••••

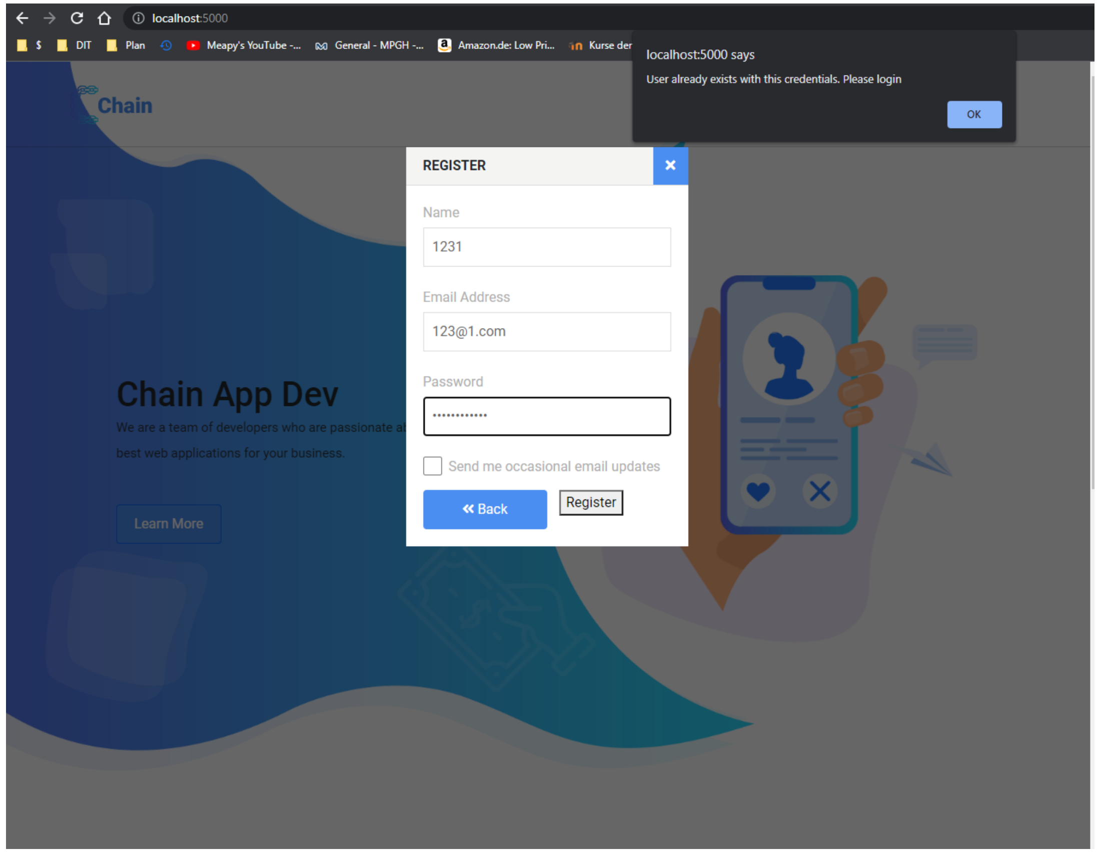
[ ] Send me occasional email updates

localhost:5000
OK
Chain
Chain App Dev
We are a team of developers who are passionate a
best web applications for your business.
Learn More
« Back
Register

Username doesn't only contain numbers and letters

Password doesn't match the requirements

REGISTER ERRORS

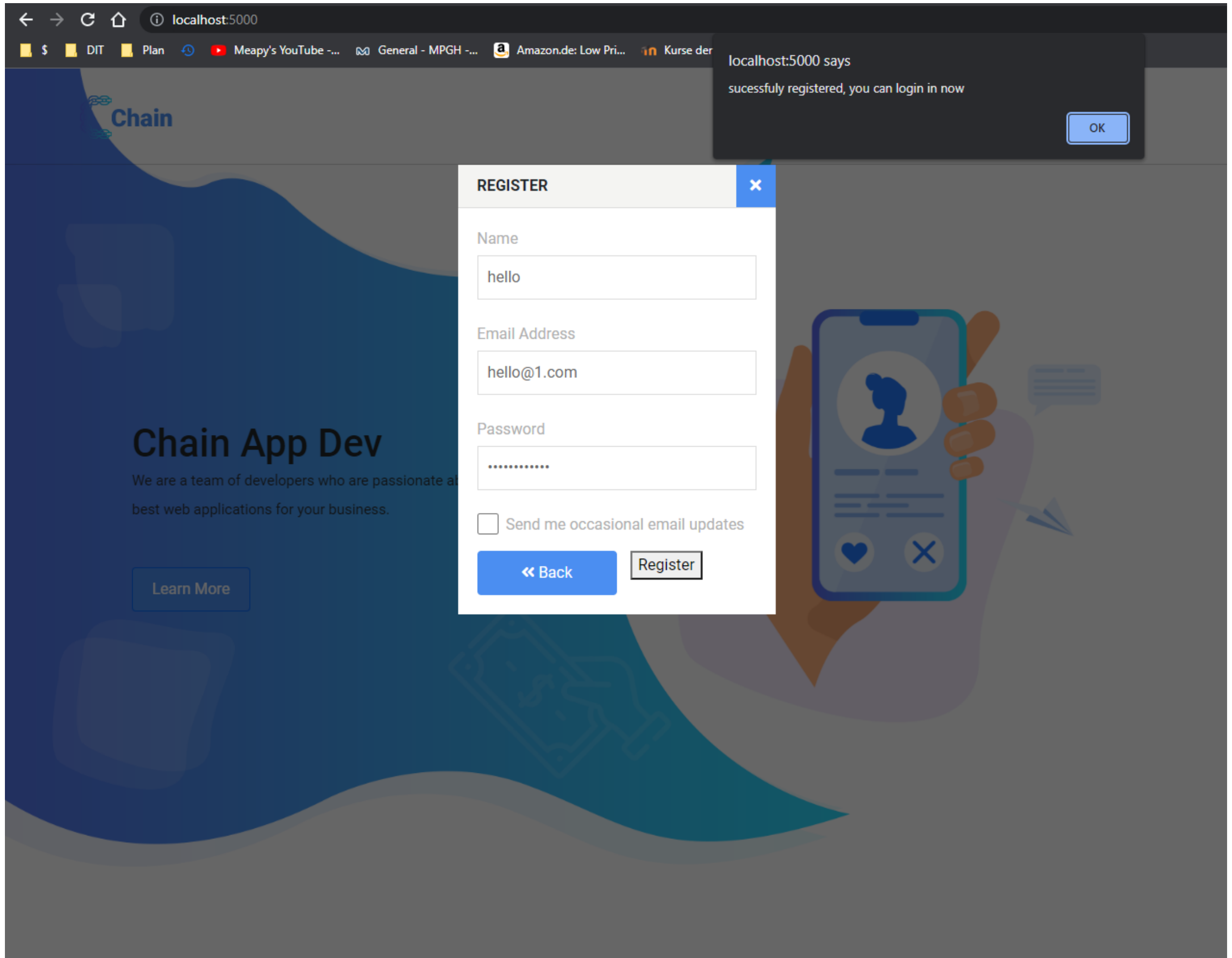

localhost:5000 says

User already exists with this credentials. Please login

OK
localhost:5000
Chain
REGISTER
Name
1231
Email Address
123@1.com
Password
Send me occasional email updates
Back
Register
Chain App Dev
We are a team of developers who are passionate al
best web applications for your business.
Learn More

Email already in use

REGISTER SUCCESS


localhost:5000
$ DIT Plan Meapy's YouTube -... General - MPGH -... Amazon.de: Low Pri... Kurse der
Chain
Chain App Dev
Learn More
Back
Register

localhost:5000 says

sucessfuly registered, you can login in now

OK

REGISTER

Name

hello

Email Address

hello@1.com

Password

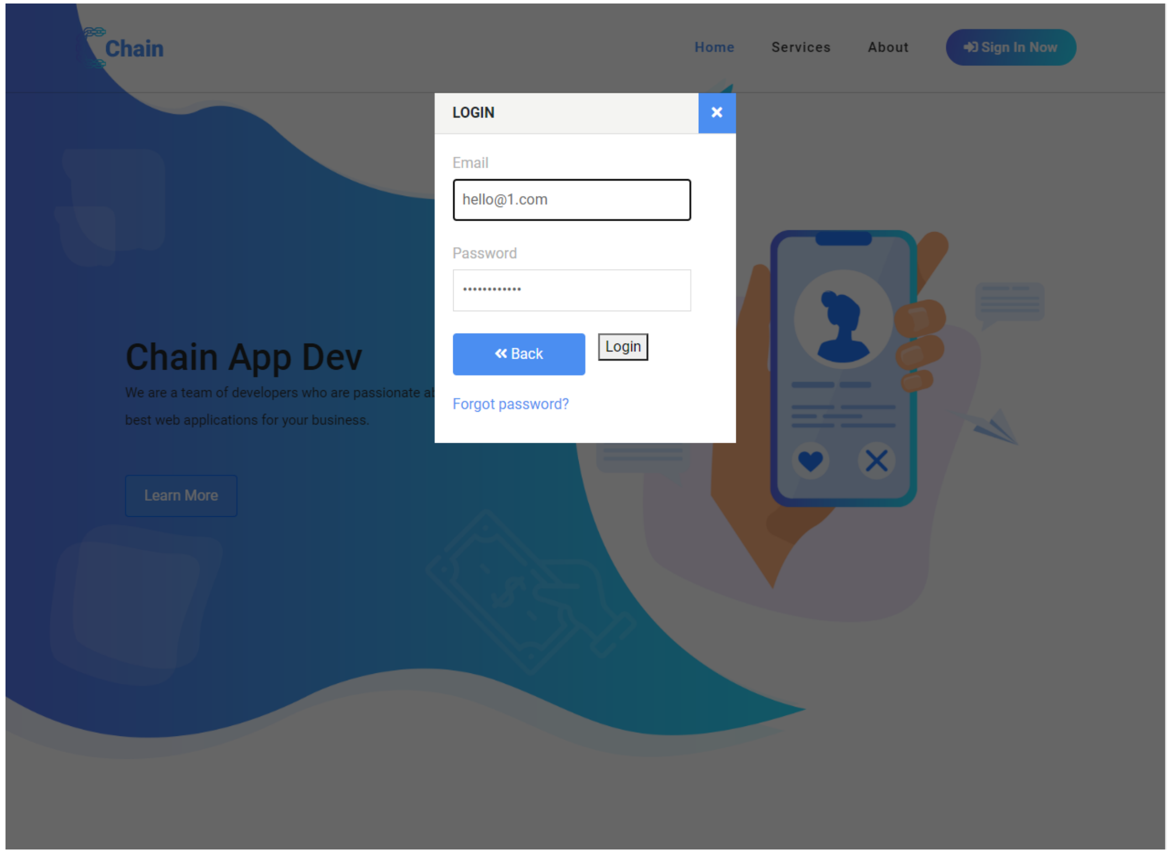
••••••••••••

Send me occasional email updates

We are a team of developers who are passionate a[…]

best web applications for your business.

LOGIN



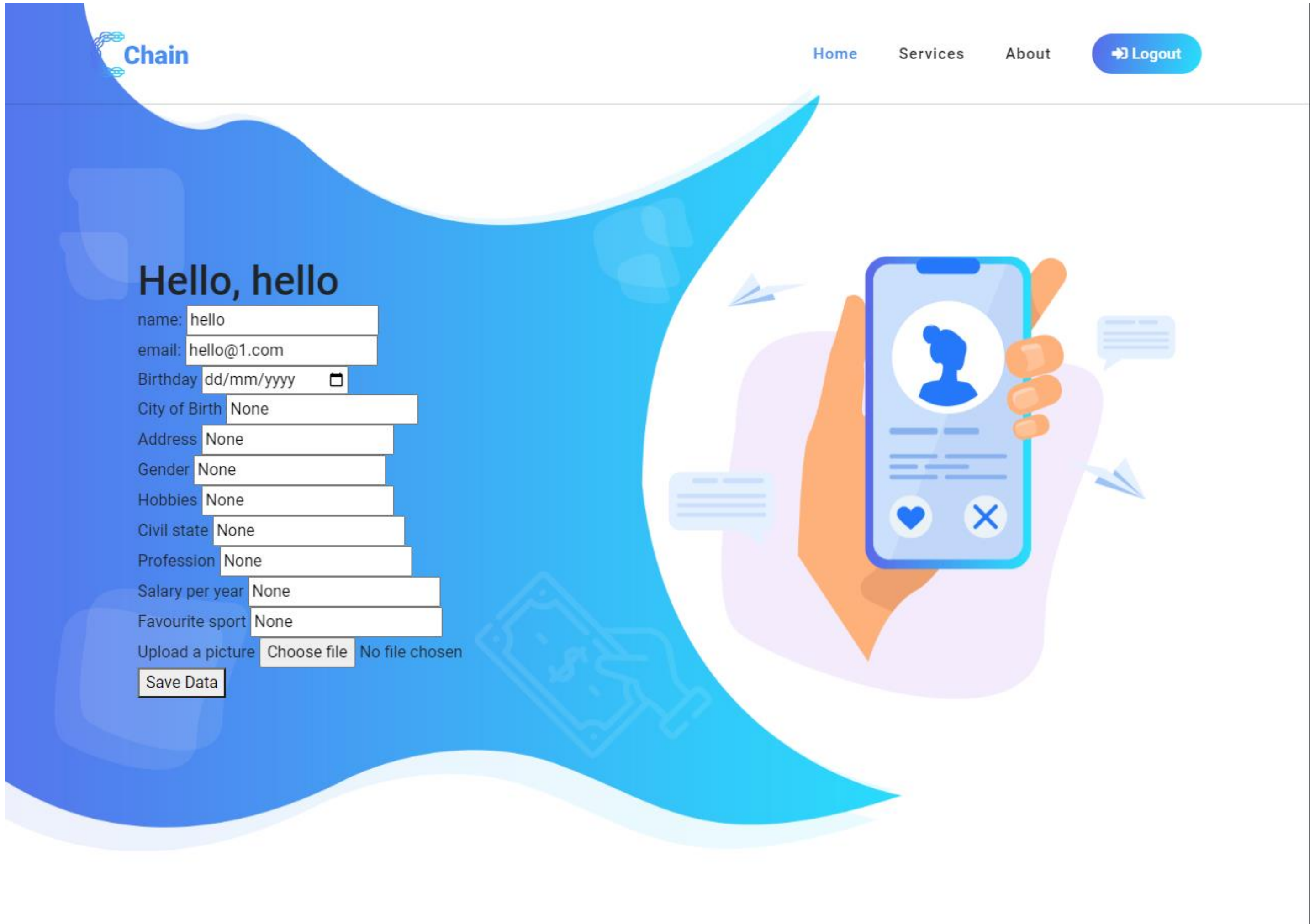
Email
hello@1.com
Password
·············

Chain App Dev

We are a team of developers who are passionate a

best web applications for your business.

ONCE LOGGED IN



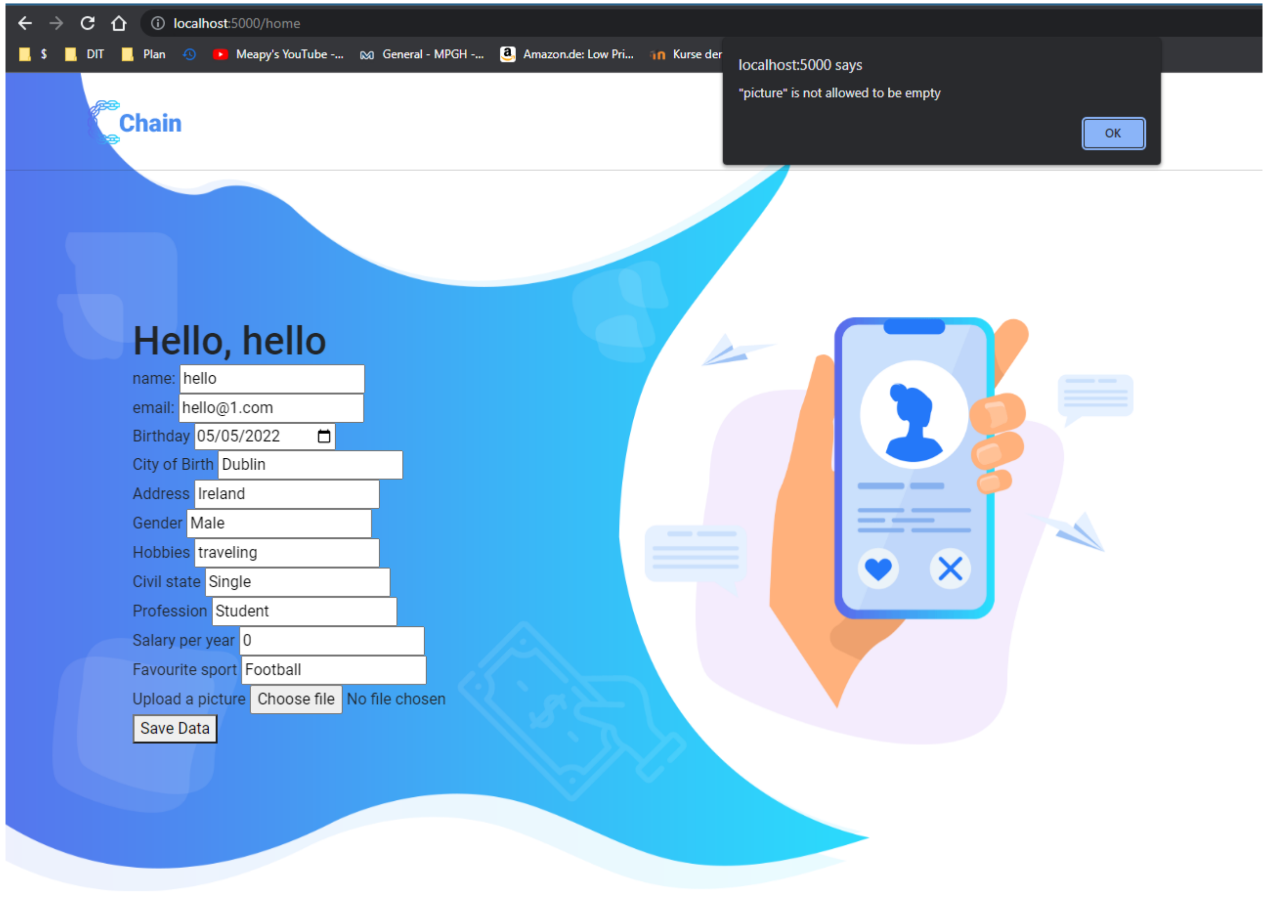
Hello, hello

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

SAVING DATA

All fields must be filled in

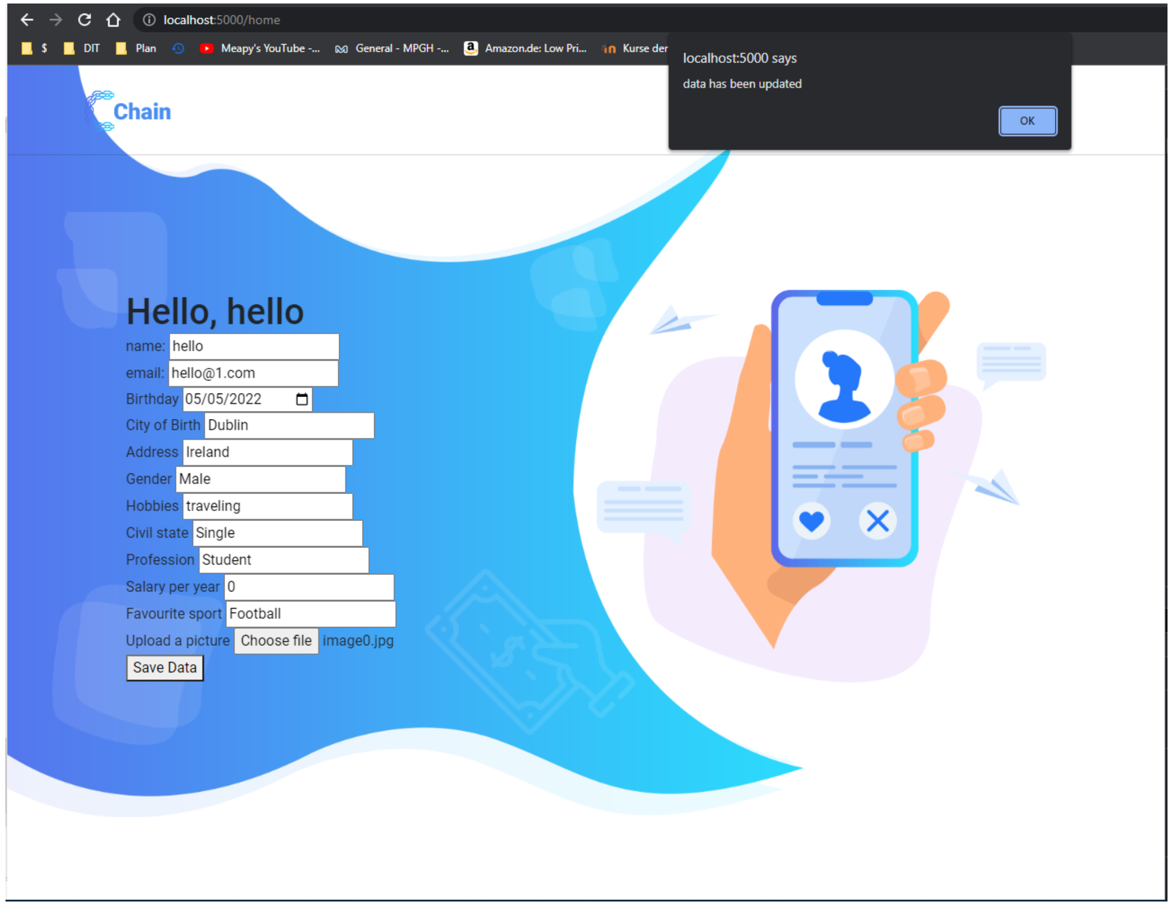

localhost:5000/home

localhost:5000 says

"picture" is not allowed to be empty

Hello, hello

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
Favourite sport	Football
Upload a picture	Choose file No file chosen


localhost:5000/home

localhost:5000 says

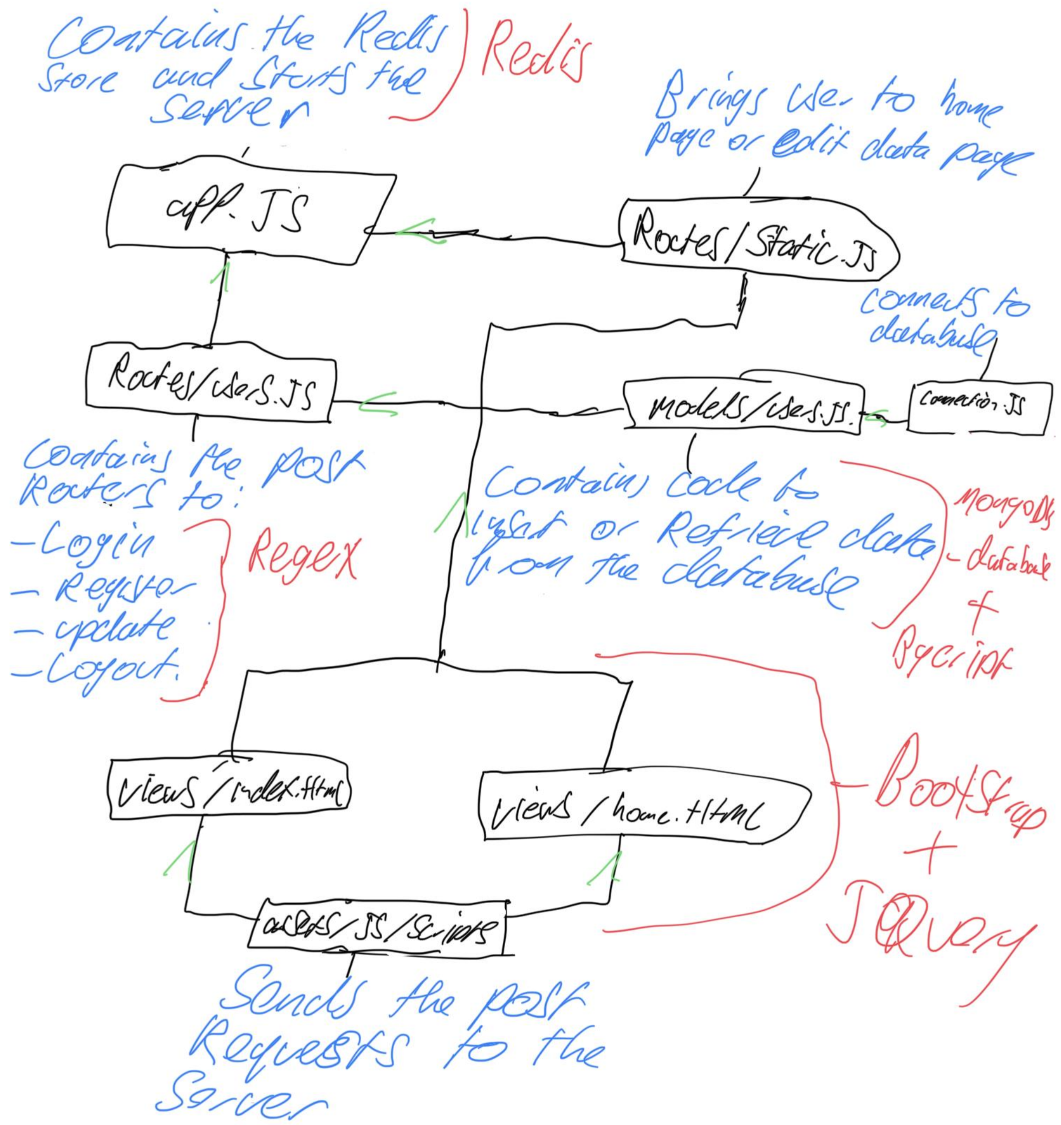
data has been updated

Hello, hello

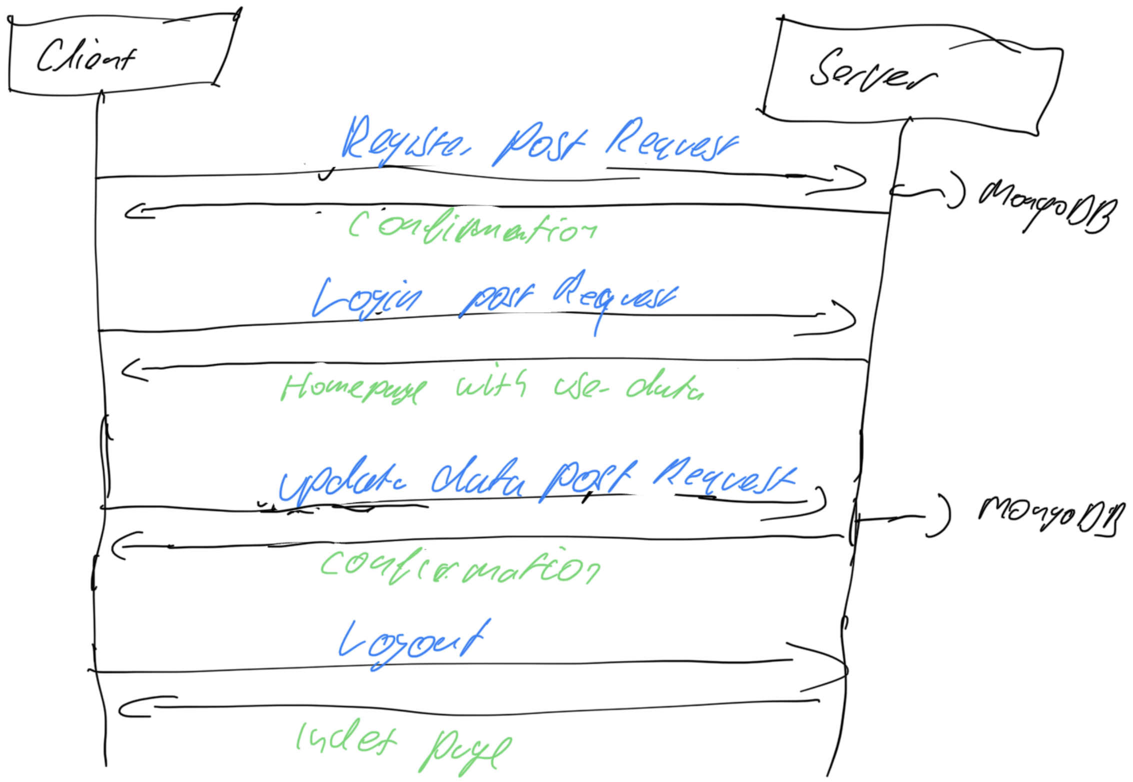
name:	hello
email:	hello@1.com
Birthday	05/05/2022
City of Birth	Dublin
Address	Ireland
Gender	Male
Hobbies	traveling
Civil state	Single
Profession	Student
Salary per year	0
Favourite sport	Football
Upload a picture	Choose file image0.jpg

Success on saving data

STRUCTURE DIAGRAM


Contains the Redis Store and Starts the Server
Redis
Brings user to home page or edit data page
app.JS
Routes/Static.JS
connects to database
Routes/users.JS
models/users.JS
connection.JS
Contains the post Routers to:
-Login
-Register
-update
-Logout.
Regex
Contains code to Insert or Retrieve data from the database
MongoDb
-database
+
Bcript
views/index.Html
views/home.HTML
Bootstrap
+
JQuery
assets/JS/Scripts
Sends the post Requests to the Server

INTERACTION DIAGRAM

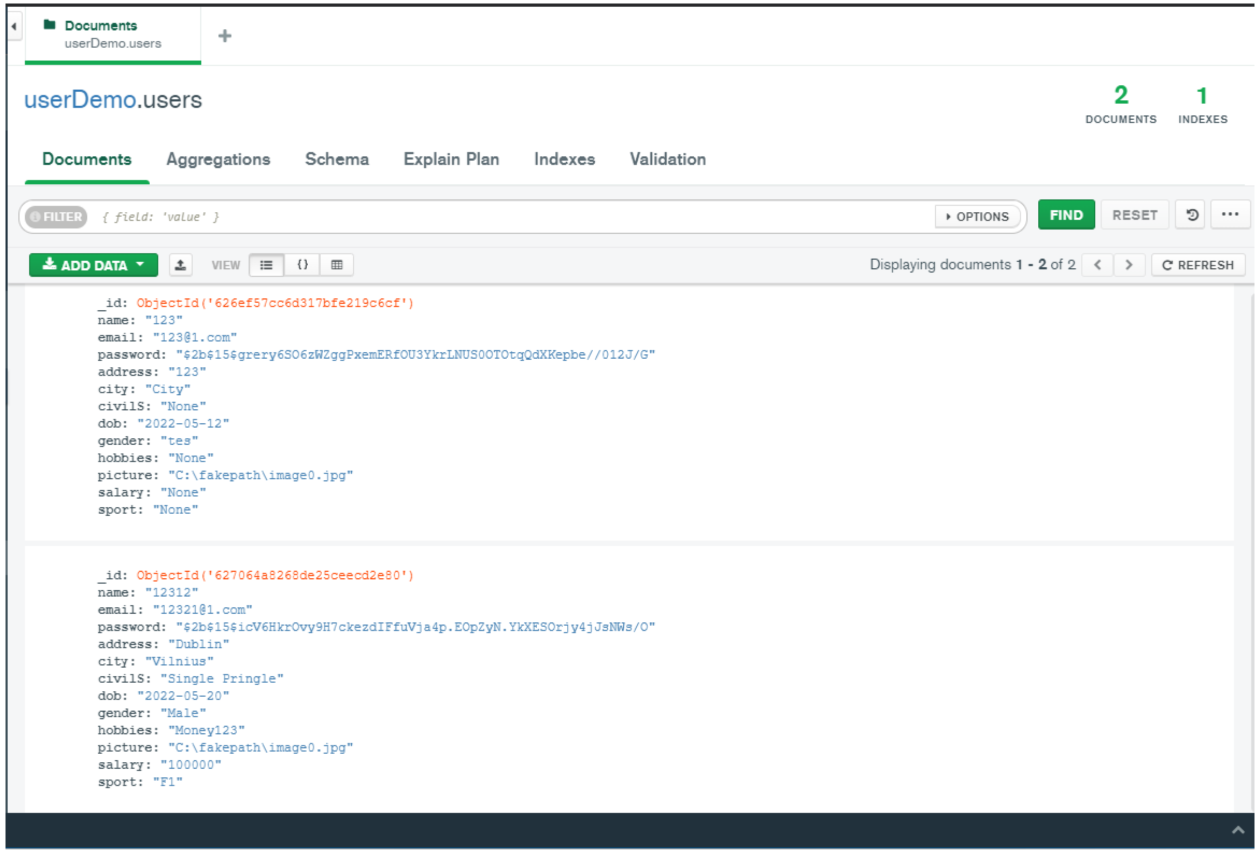


PARTS COMPLETED

- Everything has been completed

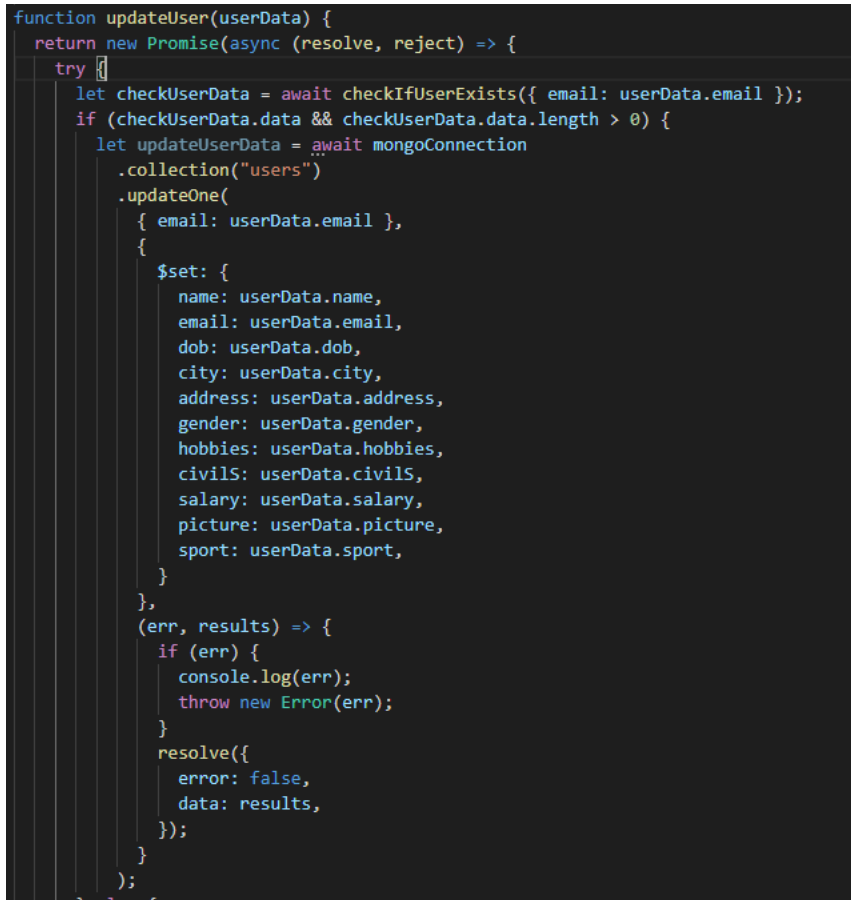
SCREENSHOTS

MongoDB



```
_id: ObjectId('626ef57cc6d317bfe219c6cf')
name: "123"
email: "123@1.com"
password: "$2b$15$grery6SO6zWZggPxemERfOU3YkrLNUS0OTOtqQdXKepbe//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$15$icV6HkrOvy9H7ckezdIFfuVja4p.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"
```

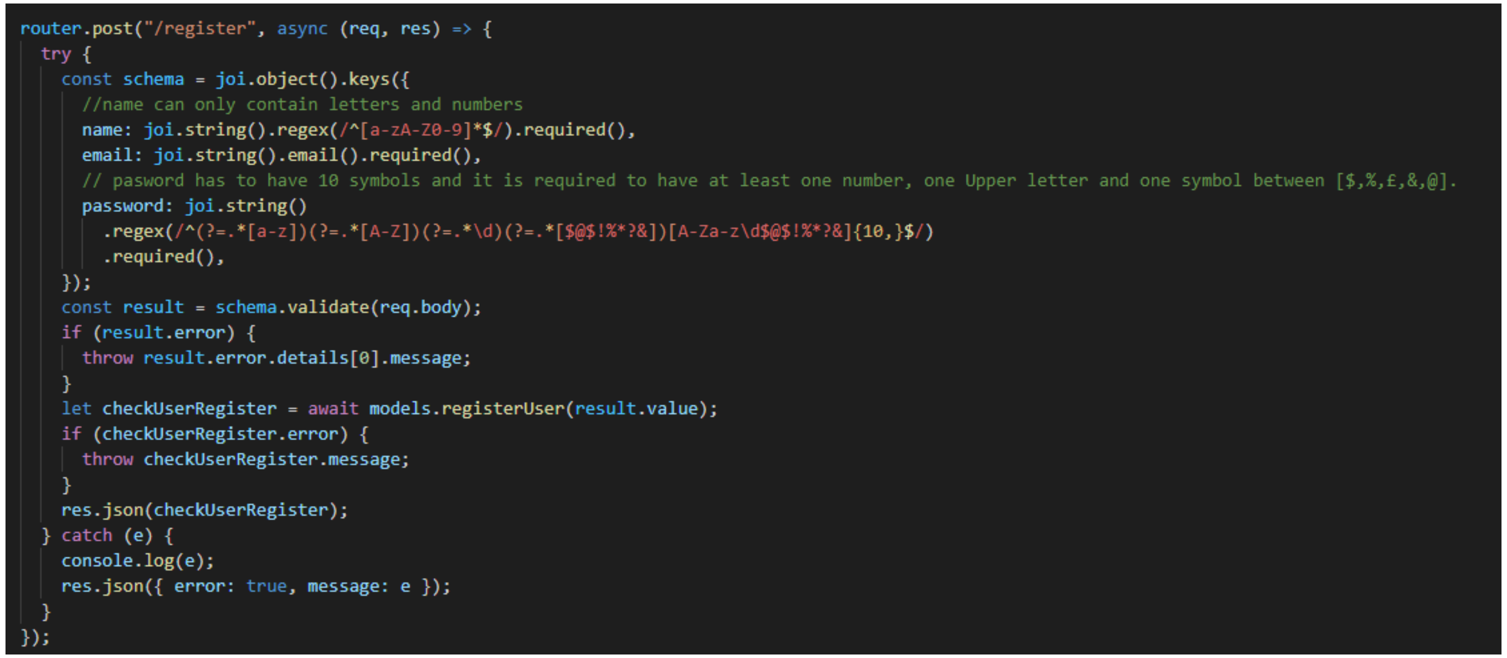
Update user code



```
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,
              });
            }
          );
```

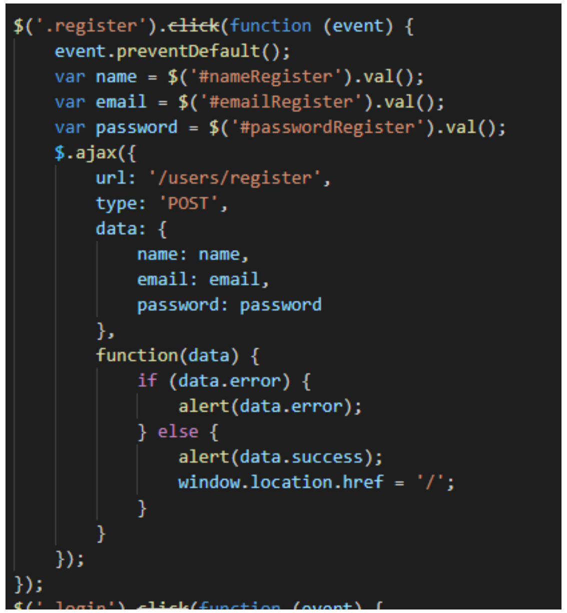
REGISTER CODE

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]*$/).required(),
      email: joi.string().email().required(),
      // pasword has to have 10 symbols and it is required to have at least one number, one Upper letter and one symbol between [$,%,£,&,@].
      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 });
  }
});
```

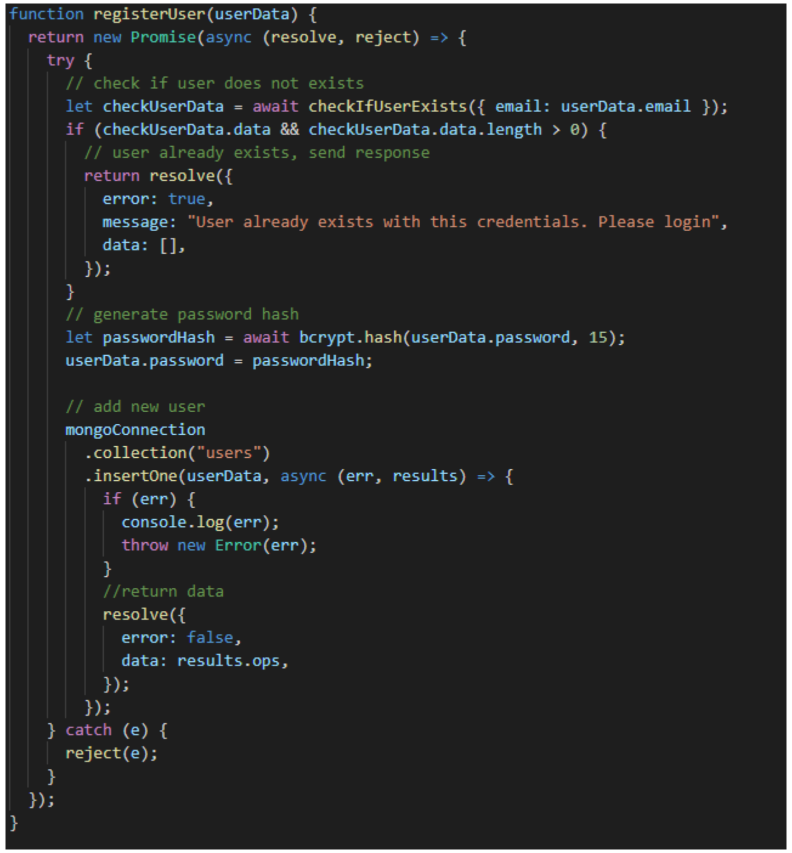
Client side post request sending



```
$('.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 = '/';
            }
        }
    });
});
$('.login').click(function (event) {
```

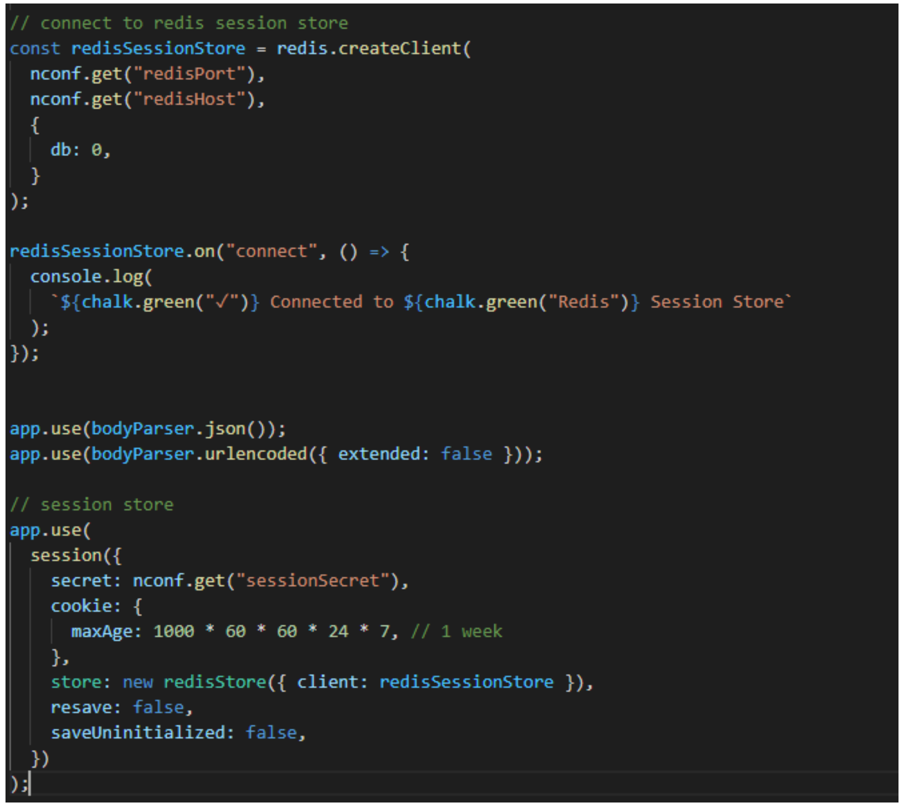
SCREENSHOTS

Register user with Bycript for password



```javascript
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);
    }
  });
}
```

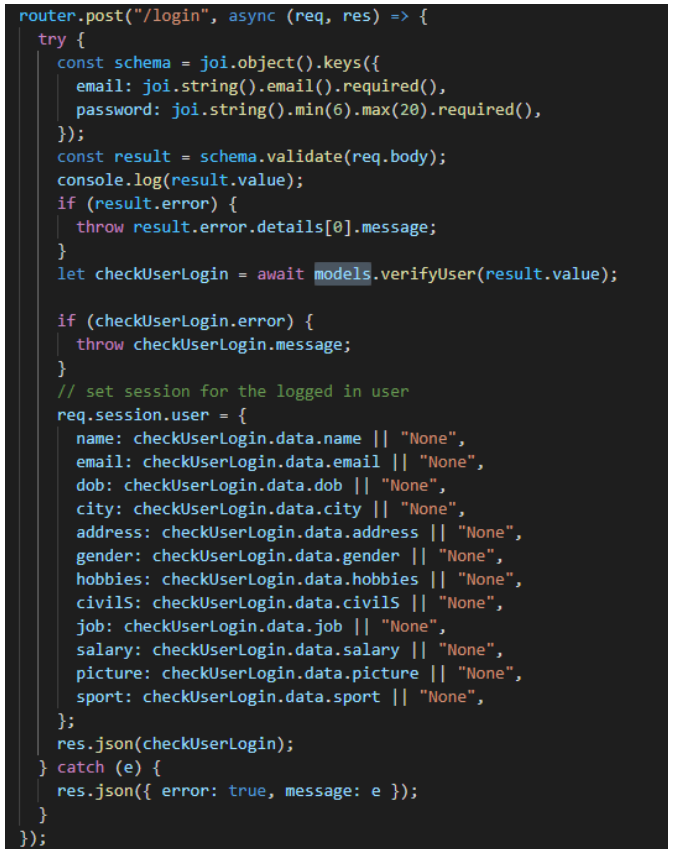
Session with Redis



```javascript
// 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,
  })
);
```

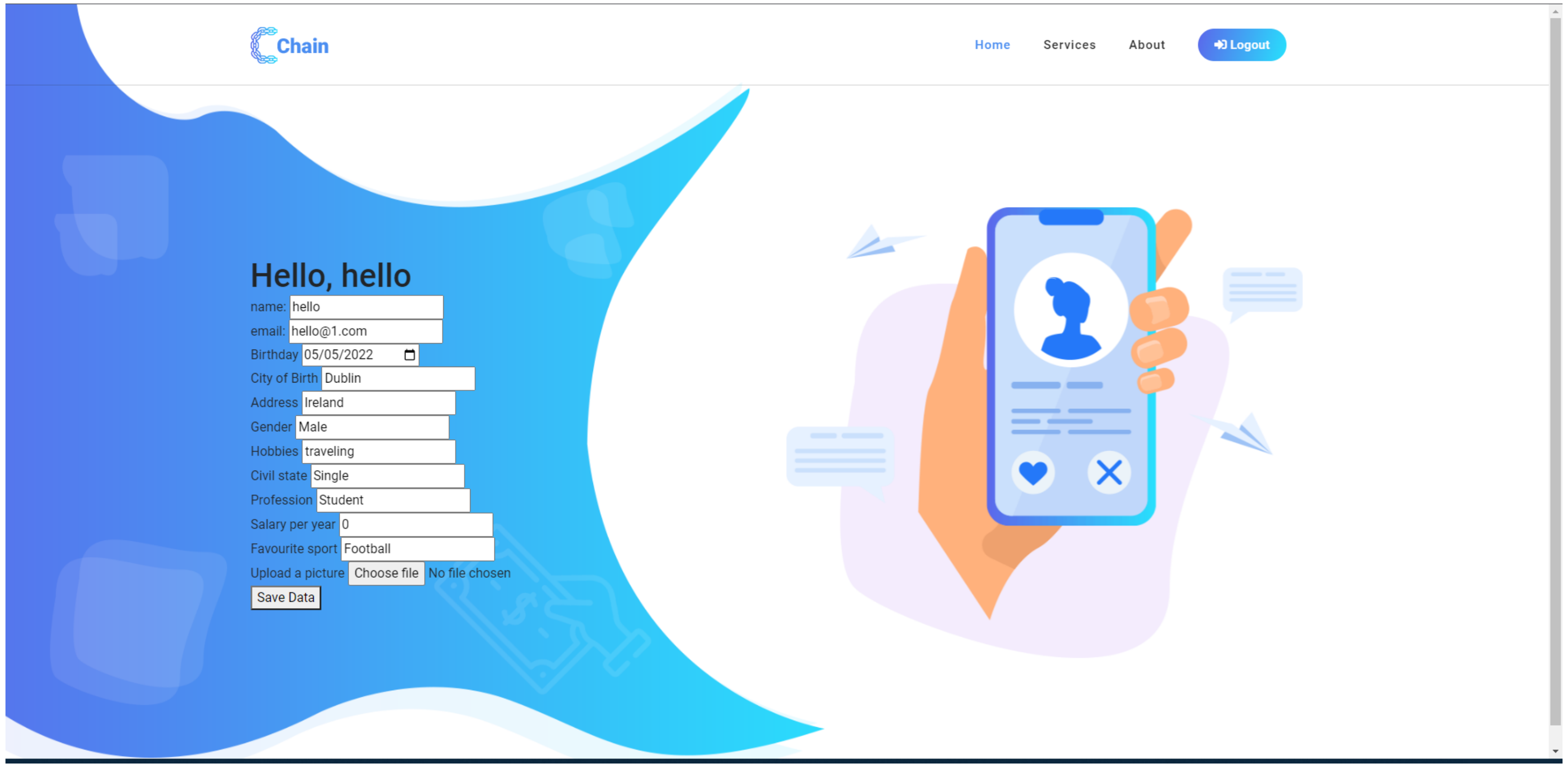
SCREENSHOTS

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

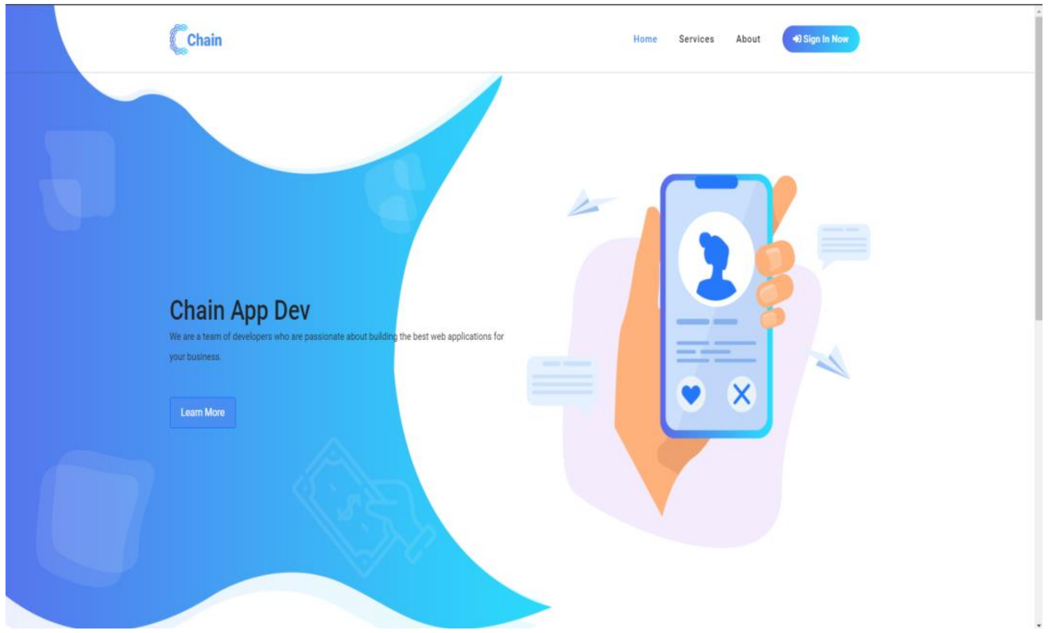


Hello, hello

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
Favourite sport	Football
Upload a picture	Choose file No file chosen

SCREENSHOTS

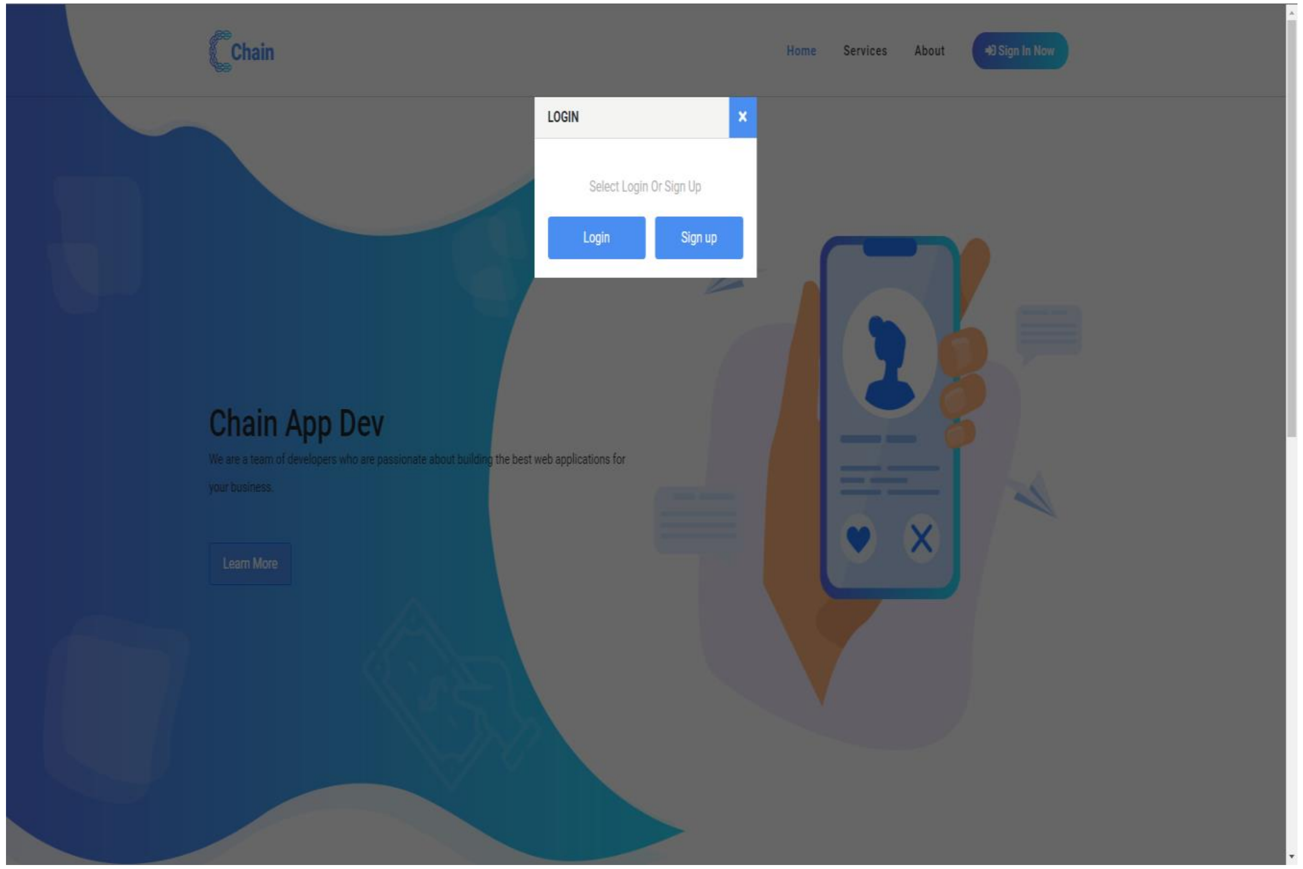
Home page



Chain App Dev

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

Login or sign up



Chain App Dev

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