Western Norway
University of
Applied Sciences

DAT152 – Advanced Web Applications

Backend Web Development
Client-Side Rendering vs. Server-Side Rendering

Tentative Schedule for Web Frameworks & Services

- **Lecture F10 (08.09)**: CSR vs. SSR
- ***Exercise F11 (08.09)***: Evaluation of CSR vs. SSR
- **Lecture F12 (12.09)**: Backend Web Development (MVC)
- ***Exercise F14 (15.09)***: MVC patterns + Derby DB
- **Lecture F15 (19.09):** Web Frameworks (SpringMVC)
- **Lecture F16 (22.09)**: Web Services – Part 1
- ***Exercise F17 (22.09)***: MVC Spring App + DB
- **Lecture F18 (26.09)**: Web Services – Part 2
- **Lecture F19 (29.09)**: Web Services – Part 3
- ***Exercise F20 (29.09)***: RESTful Web Services (Spring REST Framework)
- **Lecture F21 (03.10)**: Web Auth Framework – Part 1 (OpenId & OAuth2.0)
- ***Lecture F22 (06.10)*****:** Web Auth Framework – Part 2
- ***Exercise F23 (06.10)***: RESTful Web Services (Spring REST Framework)
- ***Oblig-2* F24 (10.10):** Web framework + Web services + Auth

Tools

- IDE
  - Eclipse, VS Code2, IntelliJ, ++
- Java 17 or greater
- Apache Tomcat 10.1.x
- Databases (MySql, H2 etc)
- Postman
- Curl
- Maven
- Docker
- GitHub
- Keycloak
- ++

Plan for Today

- Client-Side Rendering (CSR) vs. Server-Side Rendering (SSR)
- Metrics and Quality Assessments

Further reading:

Beke, M. (2018). On the comparison of software quality attributes for client-side and server-side rendering.

Soitinaho, J. (2024). Migrating Enterprise Single-Page Application to Server-side Renderable Application.

Client-Side vs. Server-Side Rendering

- Both technologies are used to develop web applications running on different platforms (mobile, desktop, etc)
- What are example technologies for CSR (aka SPA)?
- What are example technologies for SSR (aka MPA/traditional web dev)?
- What are the metrics for accessing the quality of CSR or SSR?

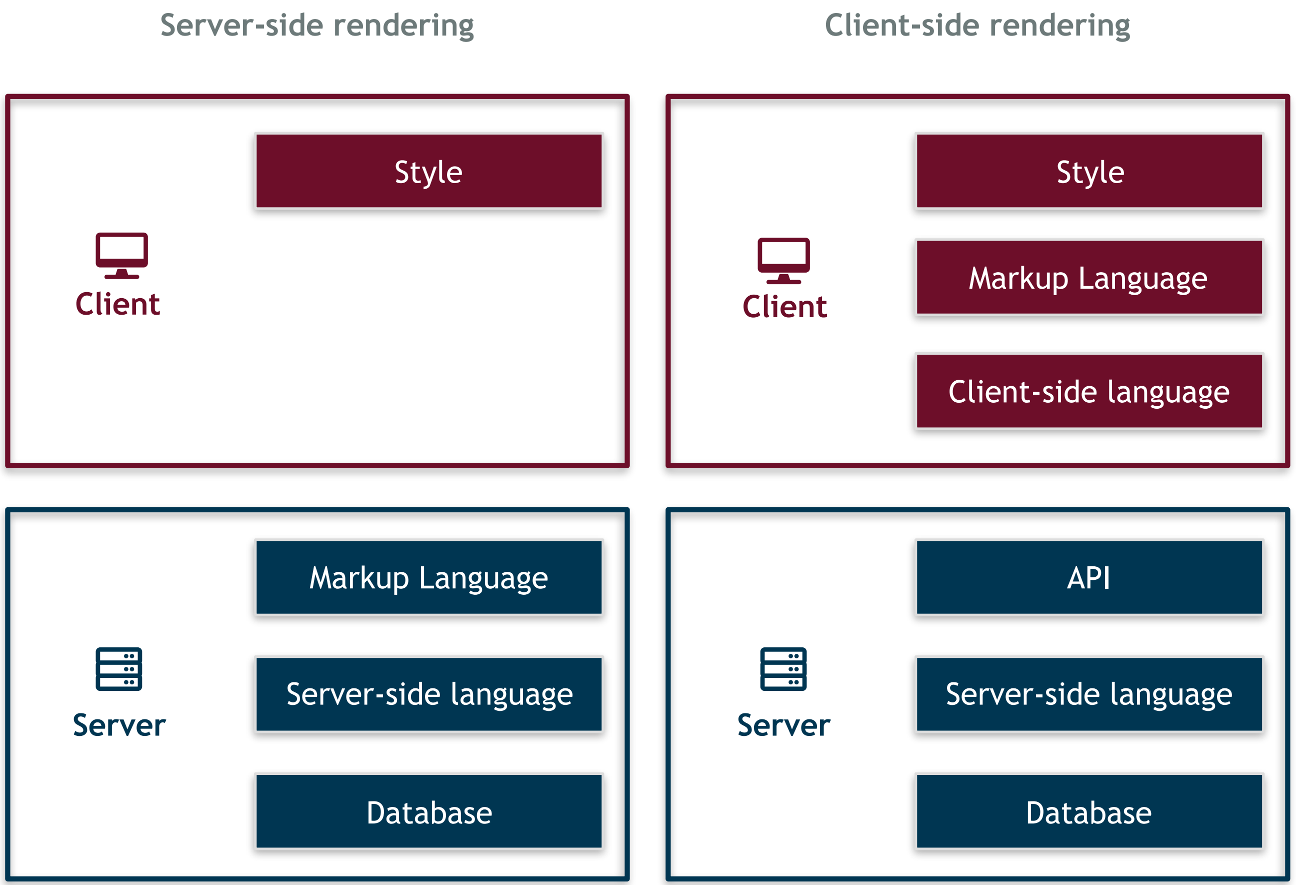
Technologies for CSR (Examples)

- Language
  - Javascript (mainly) – for frontend code/logic
- Frontend Javascript frameworks
  - React, Angular, Vue, …
- Web server (for serving data)
  - Node.js (Javascript)

Technologies for SSR (Examples)

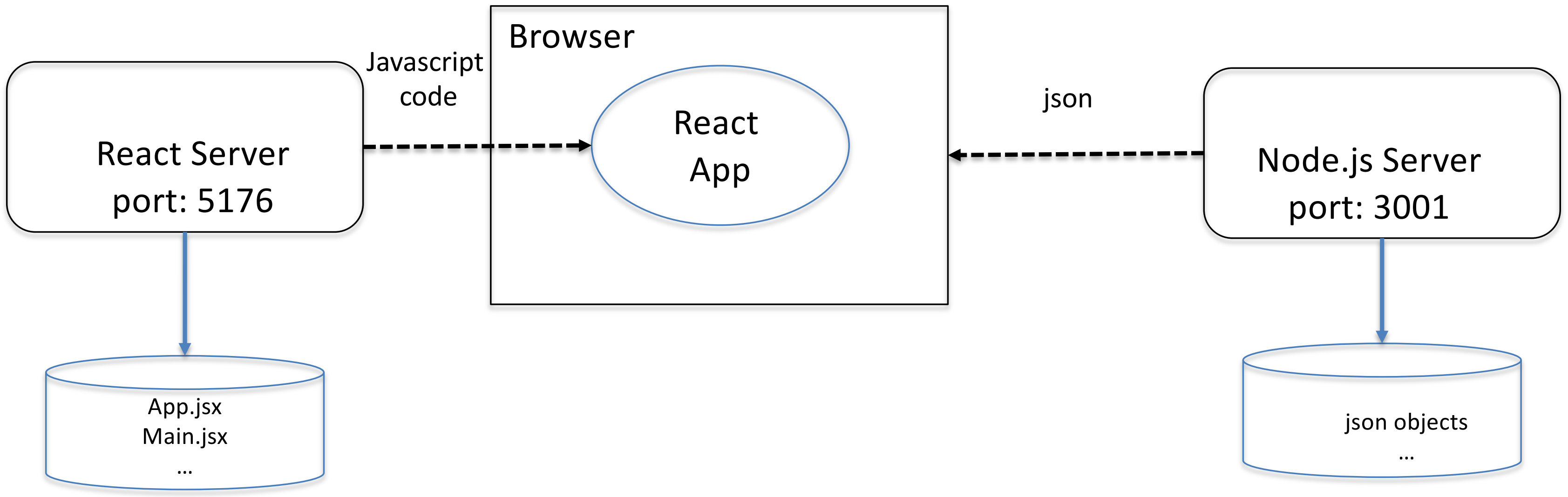
- Language
  - Go, PHP, Java, Python, (Javascript)
- Frameworks
  - Laravel, Wordpress, Ruby-on-Rails, Spring, Django, (Next.js)
- Web/App Servers
  - Apache HTTP Server, Nginx, Apache Tomcat, Glassfish, (Node.js)

Technology Stacks for CSR vs. SSR

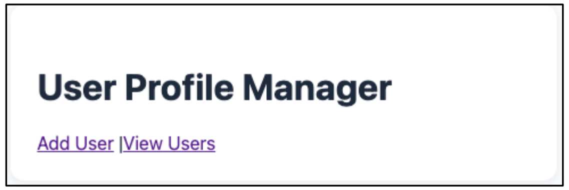


Beke, M. (2018). On the comparison of software quality attributes for client-side and server-side rendering.

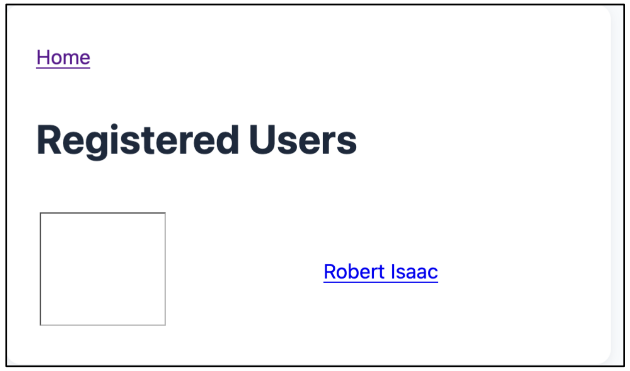
CSR - Architecture



The Views - CSR

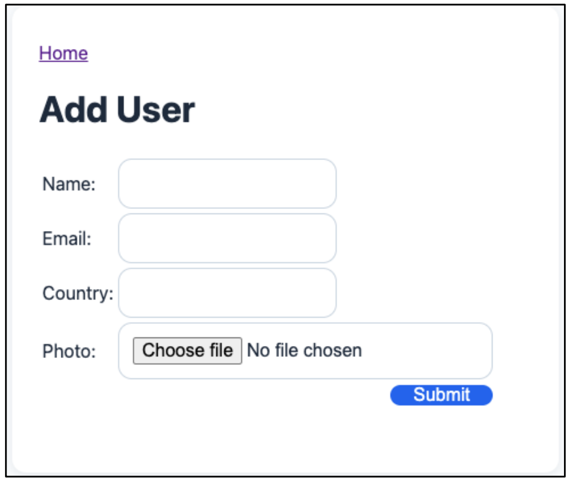

User Profile Manager

Add User |View Users


Home

Registered Users

Robert Isaac


Home

Add User

Name:

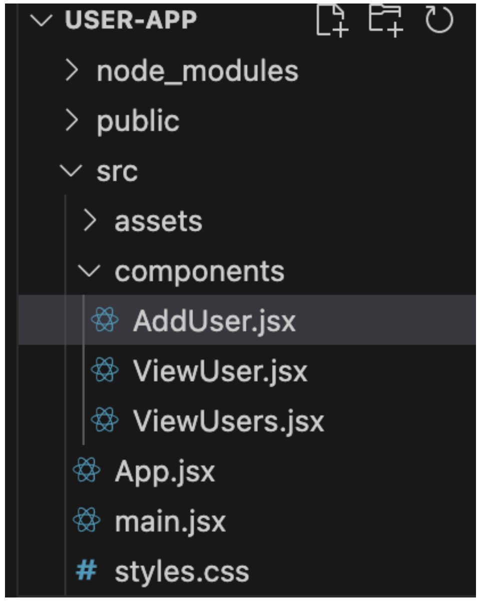
Email:

Country:

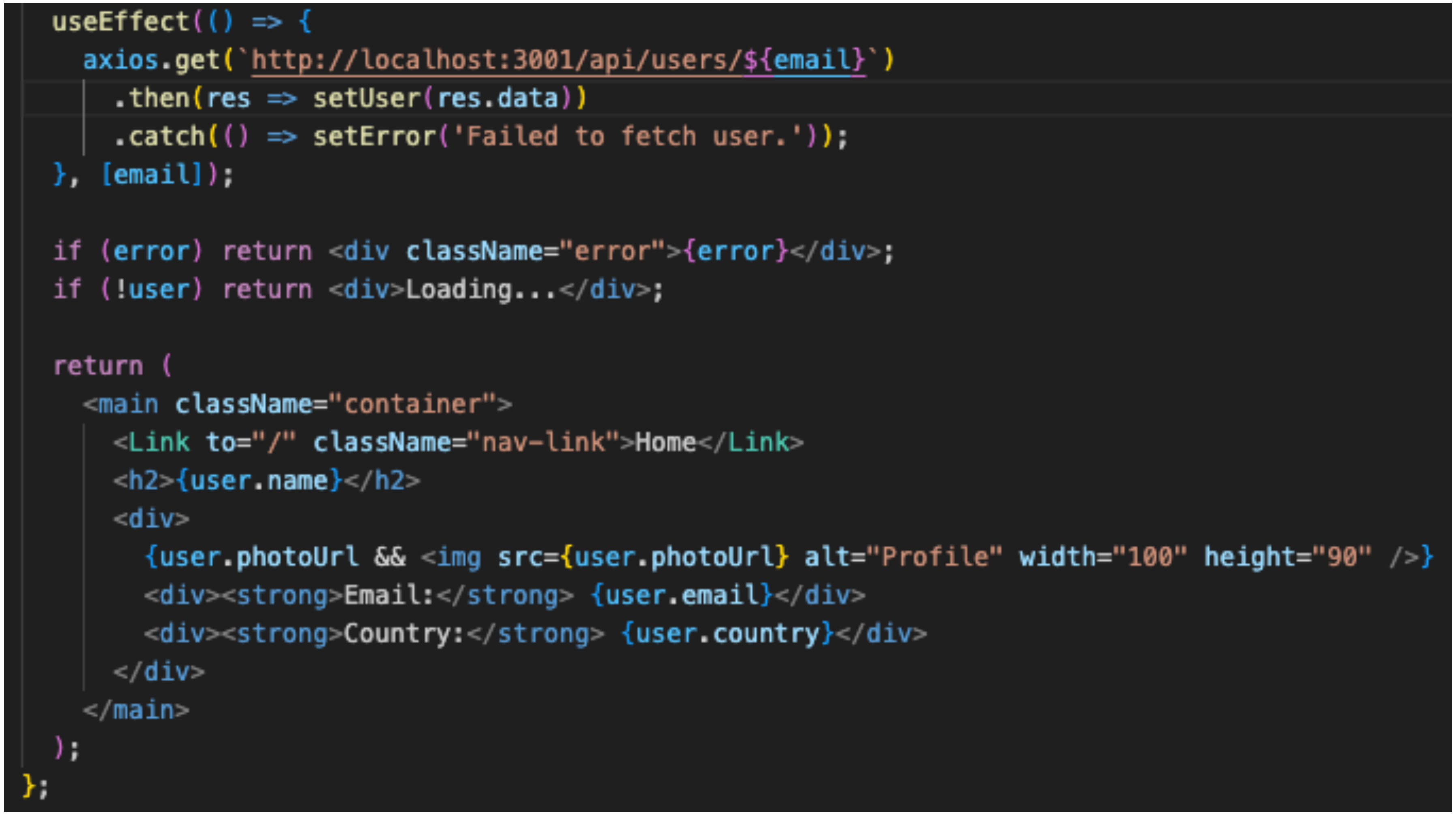
Photo: Choose file No file chosen

Submit

JavaScript – Manipulating HTML DOM

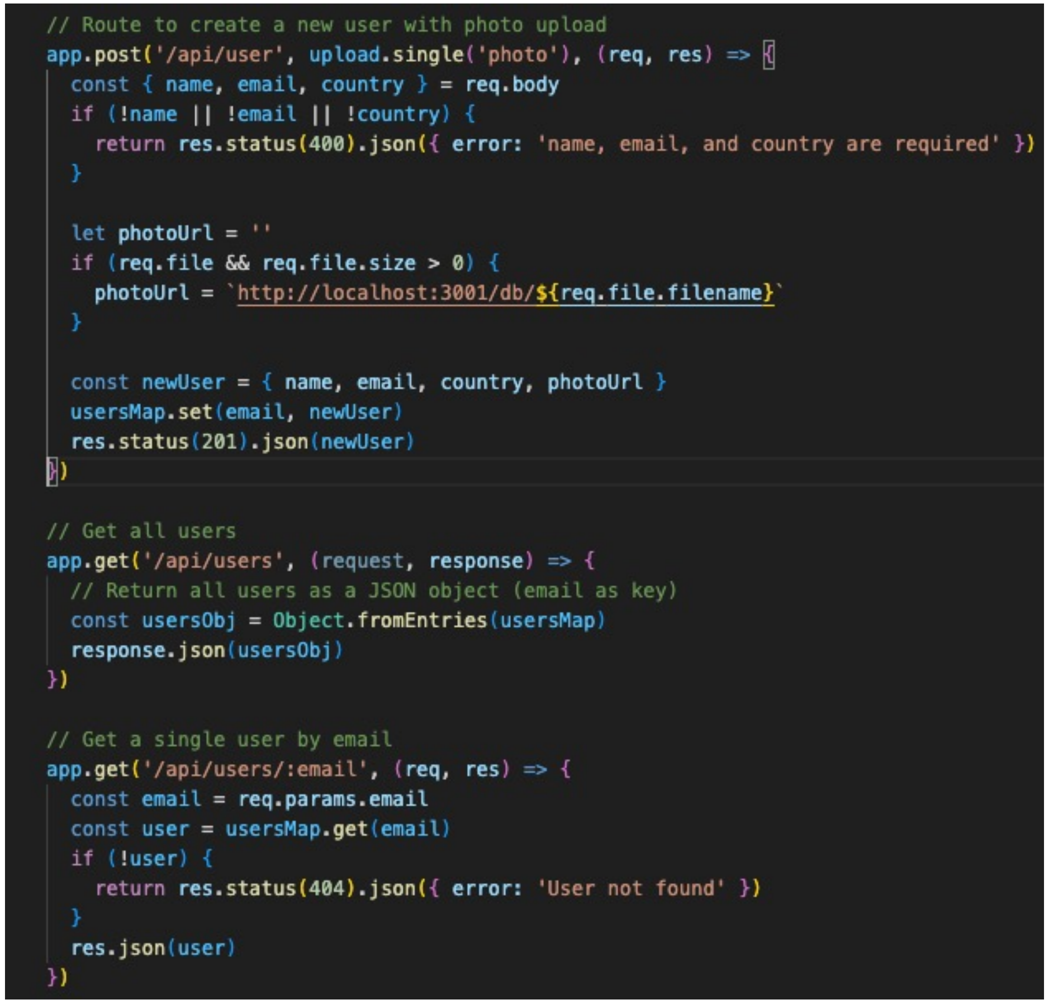

```
USER-APP
  > node_modules
  > public
  v src
    > assets
    v components
        AddUser.jsx
        ViewUser.jsx
        ViewUsers.jsx
      App.jsx
      main.jsx
    # styles.css
```

Client-Side: Javascript manipulating DOM



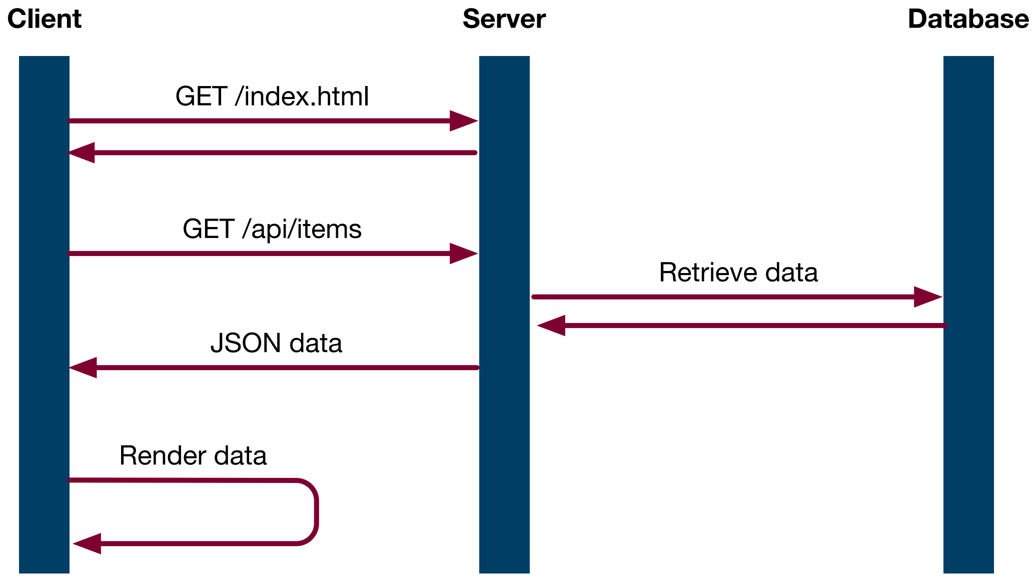
```
  useEffect(() => {
    axios.get(`http://localhost:3001/api/users/${email}`)
      .then(res => setUser(res.data))
      .catch(() => setError('Failed to fetch user.'));
  }, [email]);

  if (error) return <div className="error">{error}</div>;
  if (!user) return <div>Loading...</div>;

  return (
    <main className="container">
      <Link to="/" className="nav-link">Home</Link>
      <h2>{user.name}</h2>
      <div>
        {user.photoUrl && <img src={user.photoUrl} alt="Profile" width="100" height="90" />}
        <div><strong>Email:</strong> {user.email}</div>
        <div><strong>Country:</strong> {user.country}</div>
      </div>
    </main>
  );
};
```

CSR – Web Server (Node.js)

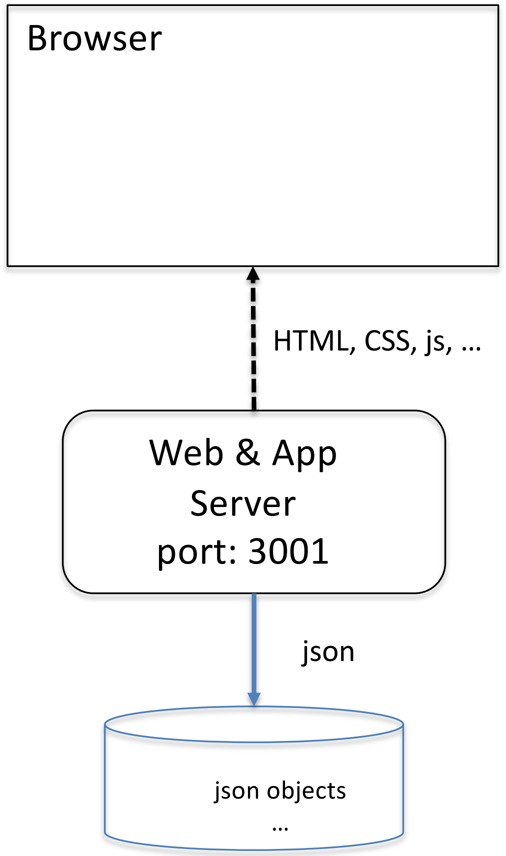


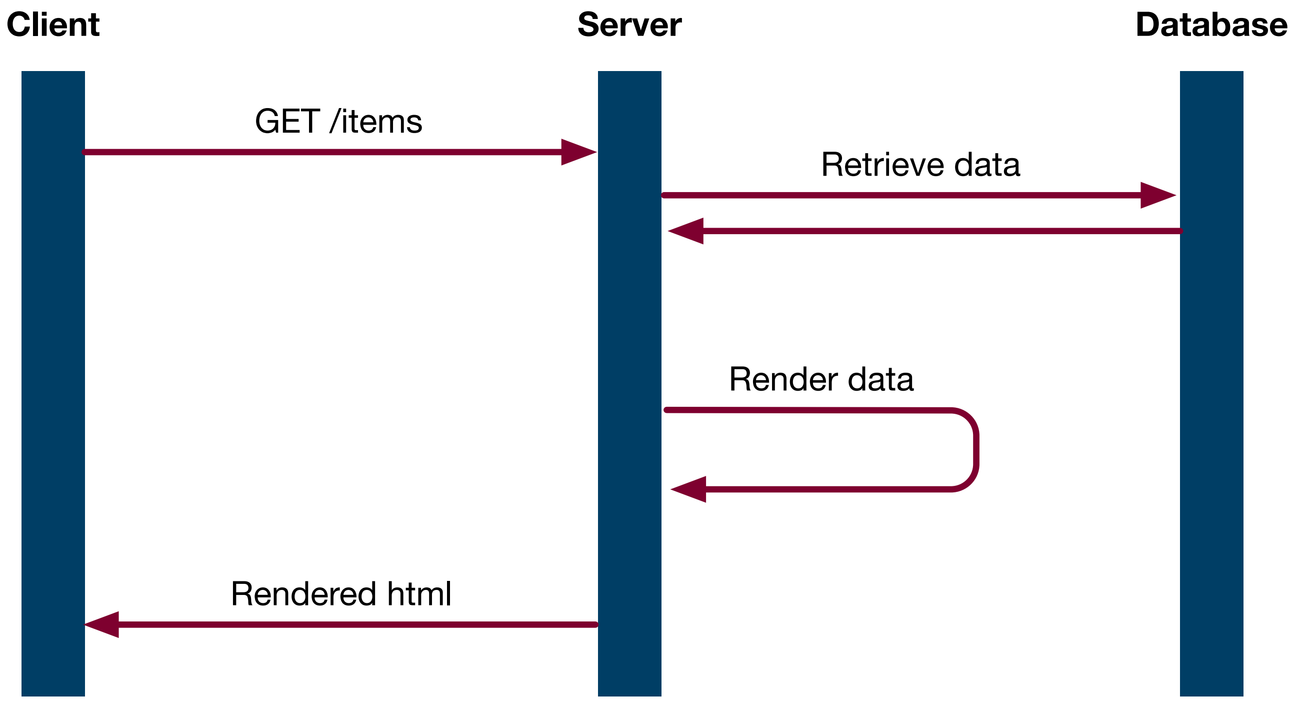
```
// Route to create a new user with photo upload
app.post('/api/user', upload.single('photo'), (req, res) => {
  const { name, email, country } = req.body
  if (!name || !email || !country) {
    return res.status(400).json({ error: 'name, email, and country are required' })
  }

  let photoUrl = ''
  if (req.file && req.file.size > 0) {
    photoUrl = `http://localhost:3001/db/${req.file.filename}`
  }

  const newUser = { name, email, country, photoUrl }
  usersMap.set(email, newUser)
  res.status(201).json(newUser)
})

// Get all users
app.get('/api/users', (request, response) => {
  // Return all users as a JSON object (email as key)
  const usersObj = Object.fromEntries(usersMap)
  response.json(usersObj)
})

// Get a single user by email
app.get('/api/users/:email', (req, res) => {
  const email = req.params.email
  const user = usersMap.get(email)
  if (!user) {
    return res.status(404).json({ error: 'User not found' })
  }
  res.json(user)
})
```

Allows the client app to fetch json data using the defined routes

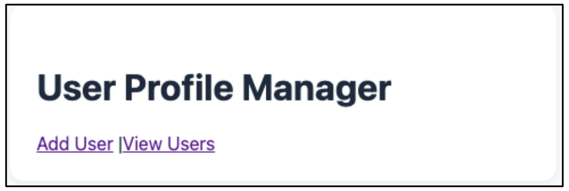


SSR - Architecture



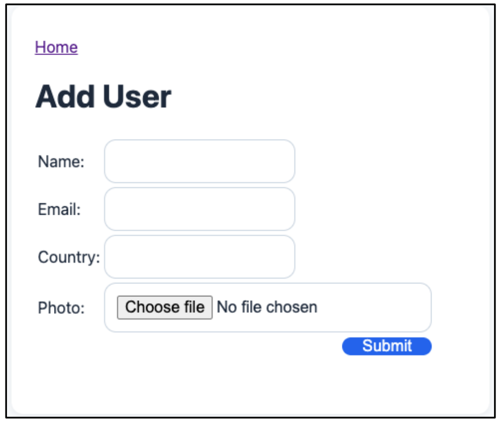


The Views - SSR



User Profile Manager

Add User |View Users



Home

Add User

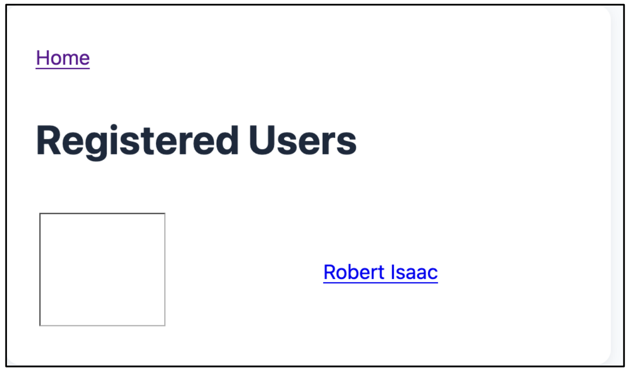
Name:

Email:

Country:

Photo: Choose file No file chosen

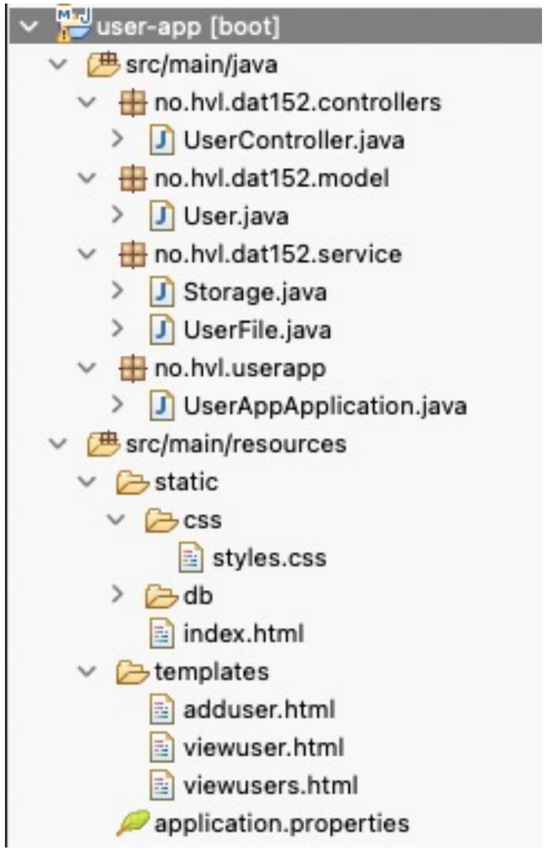
Submit



Home

Registered Users

Robert Isaac

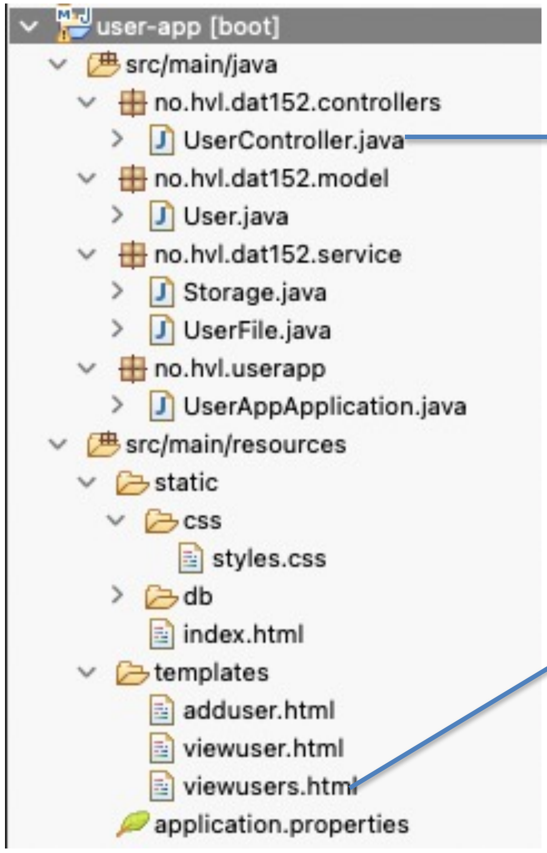


```
user-app [boot]
  src/main/java
    no.hvl.dat152.controllers
      UserController.java
    no.hvl.dat152.model
      User.java
    no.hvl.dat152.service
      Storage.java
      UserFile.java
    no.hvl.userapp
      UserAppApplication.java
  src/main/resources
    static
      css
        styles.css
      db
      index.html
    templates
      adduser.html
      viewuser.html
      viewusers.html
    application.properties
```

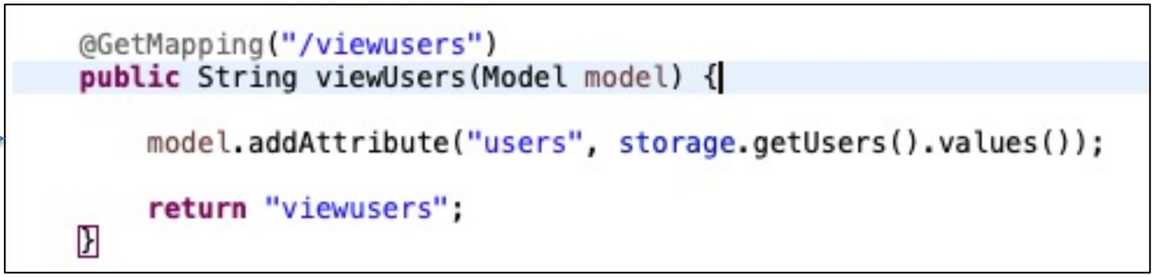
- Each HTML page is prepared at the server e.g., using Thymeleaf template, jsp, etc
- A client browser requests for the page
- The server sends back a full HTML page as response to the client browser
- The client browser renders the HTML

HTML, CSS, js, resources are located on the server

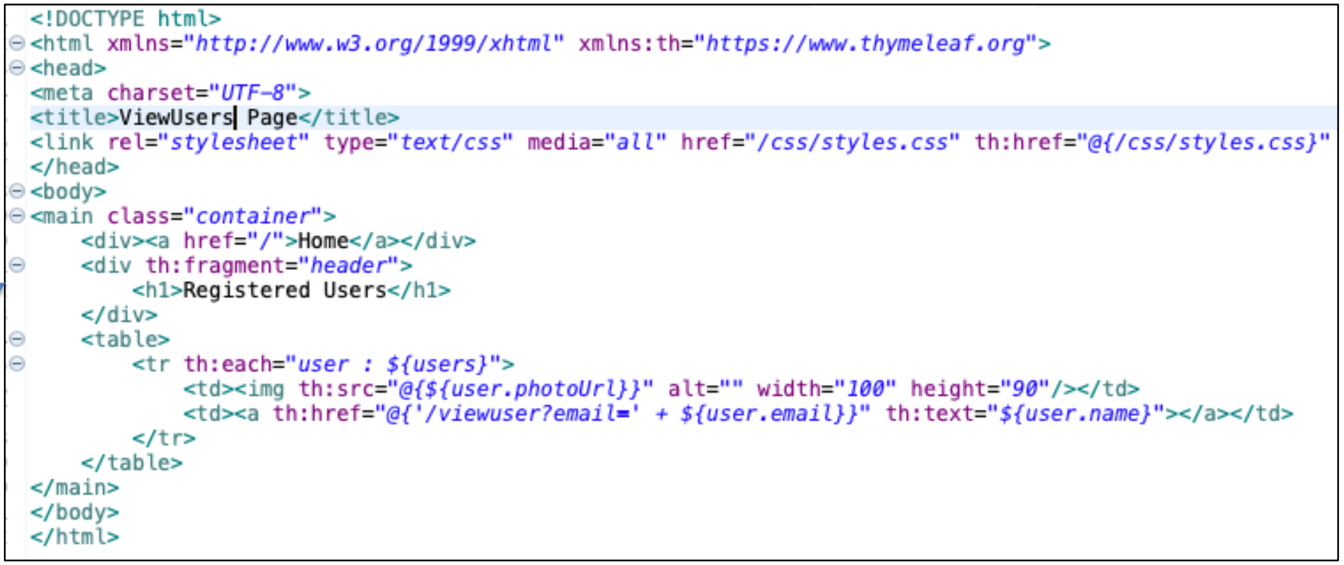
SSR – Data + HTML



```
user-app [boot]
  src/main/java
    no.hvl.dat152.controllers
      UserController.java
    no.hvl.dat152.model
      User.java
    no.hvl.dat152.service
      Storage.java
      UserFile.java
    no.hvl.userapp
      UserAppApplication.java
  src/main/resources
    static
      css
        styles.css
      db
      index.html
    templates
      adduser.html
      viewuser.html
      viewusers.html
    application.properties
```



```java
@GetMapping("/viewusers")
public String viewUsers(Model model) {

    model.addAttribute("users", storage.getUsers().values());

    return "viewusers";
}
```



```html
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml" xmlns:th="https://www.thymeleaf.org">
<head>
<meta charset="UTF-8">
<title>ViewUsers Page</title>
<link rel="stylesheet" type="text/css" media="all" href="/css/styles.css" th:href="@{/css/styles.css}"
</head>
<body>
<main class="container">
    <div><a href="/">Home</a></div>
    <div th:fragment="header">
        <h1>Registered Users</h1>
    </div>
    <table>
        <tr th:each="user : ${users}">
            <td><img th:src="@{${user.photoUrl}}" alt="" width="100" height="90"/></td>
            <td><a th:href="@{'/viewuser?email=' + ${user.email}}" th:text="${user.name}"></a></td>
        </tr>
    </table>
</main>
</body>
</html>
```

Software Quality Attributes

- User Experience
- Performance
- Development Effort
- Compatibility
- Maintainability
- Scalability
- Security
- Reliability & Availability

User Experience

- How user feels when interacting with the web app.

- CSR
  - Allows rich interaction by the user
    - e.g., entry and exit animations for static elements
    - can fetch new dynamic content on a webpage without loading the entire page
    - allows for a more usable way to submit forms or other data
    - Enables lazy-loading of content and images
  - Client needs to wait for all Javascript frontend to download before page content can be rendered.
- SSR
  - On first load, user experience a faster initial load time since a fully rendered page is sent to the client

Performance

- How responsive and stable a system is under a certain load
  - Response time, resource utilization, throughput, and efficiency
- Client
  - Initial page load time
  - Subsequent page load time
- Server
  - Bandwidth: Amount of data transferred (per sec)
  - Throughput: Number of requests processed per sec by server

Performance

Metric	**CSR**	**SSR**
Initial Page Load Time	**Slower**: frontend Javascript code first downloaded and executed by the client. Then request is made to fetch data from the server. Returned data from server is then rendered by the client. Many roundtrips.	**Faster**: since a fully rendered page with data is returned to the client.
Subsequent Page Load Time	**Faster**: Often, only data is requested in async mode and page is updated with data. Partial page update.	**Slower**: Complete new HTML page is fetched from server and rendered by the client
Throughput	**Higher**: Server only returns serialized data object (json) and not a fully rendered HTML. Less processing and thus more requests can be processed	**Lower**: Since fully rendered HTML page has to be returned by server, more processing is needed by server for each request.
Bandwidth	**Lower**: Less data (json) is transferred between the client and server.	**Higher**: Complete HTML page is sent to the client.

Development Effort

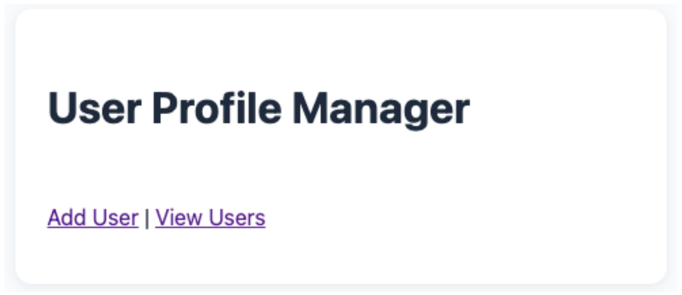
- Programming effort requires to develop an application

Metric	**CSR**	**SSR**
Complexity	**High** - Requires effort for creating complete and secure server-side API and client-side communication with the API. - Duplicated code can occur, e.g., models created on server side may be duplicated on client side - Different languages may be used client-side and server-side - Build environment must be configured and maintained for both the server-side and client-side code	**Low**

Compatibility

Metric	**CSR**	**SSR**
Browser support	Comes with challenges. - New features of Javascript, - users can disable Javascript	**Better**: Browser has less work
Search Engine Optimization (SEO)	Possible but difficult, since crawlers needs to access the Javascript	**Excellent**: Crawlers can access and index all URLs

Lab 1 – Exercise


User Profile Manager

Add User | View Users

- **Client-Side Rendering (CSR) vs. Server-Side Rendering (SSR)**
  - **Task 1: Run the web applications**
  - **Task 2: Browse through the source code**
  - **Task 3: Run the Google Developer Tools to measure performance metrics (initial load time and subsequent load time)**

Next Lecture - Friday

- Backend web development
- Strategies for MVC