Report SpringApp Project Group 4

Jacek Antoni Wegrzynowski, Rashaad Wells Iversen, Veronicha C. T. Pettersen

November 20, 2023

Abstract

10-15 lines with the software technology and the highlights from the project that has been undertaken.

1 Introduction

Approximately 1 page on:

- A brief introduction to the prototype implementation and topic of the project. In this project, an IoT-cloud software system has been developed. The system takes feedback from users in the form of yes/no votes on polls. The polls can be voted on either via an web interface or via an IoT-device.
- Discuss (briefly) the technology stack that has been selected, mention related technologies (if relevant), primary arguments for choice of technology stack.

  Here we can maybe add an illustration from the powerpoint
- A brief account of the results that have been obtained in the project.
- A one paragraph overview at the end, explaining how the rest of the report is / has been organised

This rest of this report is organised as follows: Section 2 gives an ....

2 Software Technology Stack

Introduce in (sufficient) depth the key concepts and architecture of the chosen software technologies. As part if this, you may consider using a running example to introduce the technology.

Emphasize the "new" software technologies that was selected by the group and which has not been covered in the course.

This part and other parts of the report probably needs to refer to figures. Figure 1 from [**?**] just illustrates how figure can be included in the report.

2.1 Angular:

We chose Angular as our web application framework to develop the frontend of the FeedApp prototype. Angular is a popular, open source TypeScript based framework that is used to create Single Page Applications (SPAs). SPAs are web applications that only loads one single page, and then changes the content of that page depending on how the user interacts with the page.

Fundamental Consepts of Angular:

Components : components are many places described as the main building blocks for Angular applications. Each component will contain one HTML-file with the UI-template of the component and one TypeScript- file that contains the components logic. It can also contain CSS or SCSS files, defining the styles of the component, as well as test files and configuration files. A component defines a specific view, as well as the functionality that goes into that view. In other words, it contains both what the user sees in the UI and the logic that goes into the component.

Services :It is also possible to share different functions and logic between different components. This can be done by creating a service, which is a TypeScript file that is used for tasks such as for example business logic and handling of data. In our FeedApp implementation, we created services dedicated to managing authentication and handling poll data.

Dependency Injection (DI) : Services can also be used to inject dependencies into multiple components, with the use of DI. This is useful for connecting the different parts of the application.

Routing : The Angular Router handles the navigation between different views as users performs different tasks. This is a key element in SPAs since instead of reloading the page every time the view is changed,

One of the main advantages of creating the application as a SPA is that it speeds up the development process.

Resourses used for writing this paragraph: https://angular.io/guide/architecture

2.2 Spring Boot

We have chosen Spring Boot as our enterprice software framework when developing our application. Spring Boot is an extension of the Spring Framework that simplifies the development process, making it possible to create a functioning web application fast. To fully understand the benefits of the Spring Boot extension, we are first going to talk a little bit about some of Springs main consepts.

Spring Configurations Explained:

Bean Definitions : In Spring, objects managed by the Spring Inversion of Control (IoC) container are referred to as beans. Configurations involve specifying these beans and managing their lifecycle within the application.

Dependency Injection (DI) : Spring's DI mechanism manages dependencies among application components. This setup is crucial for injecting required services or modules into different parts of the application.

Aspect-Oriented Programming (AOP) : Configurations in Spring also include setting up aspects for handling cross-cutting concerns like logging or transaction management.

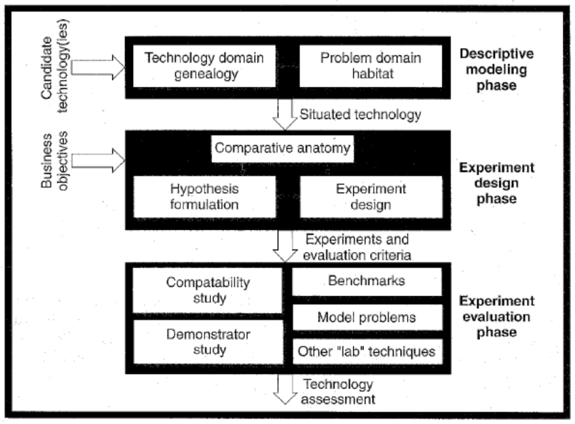

Figure 1: Software technology evaluation framework.

Data Source and Transaction Management : For applications interacting with databases, configurations encompass setting up database connections and managing transactions effectively.

Deploying a Spring Application:

Packaging : The application is compiled and packaged, typically into a JAR or WAR file, ready for deployment.

Running on a Server : The packaged application is deployed on a web server. Spring Boot, with its embedded server capability, simplifies this by allowing the application to run independently without needing a separate server setup.

Spring Boot's Role in FeedApp:

Auto-Configuration : Spring Boot automatically configures the application based on the included libraries, reducing the need for extensive manual configuration.

Simplified Deployment : The embedded server feature of Spring Boot allows our application to be deployed as a standalone unit, enhancing ease of deployment and portability.

In the implementation of our FeedApp prototype, the use of SpringBoots autoconfiguration and deployment functionalities has meade it possible for us to spend more time on the applications buissness logic.

2.3 JSON Web Tokens

2.4 H2

2.5 Hibernate

2.6 Java Persistence API

2.7 Mosquitto MQTT

3 Demonstrator Prototype

About 4 pages on:

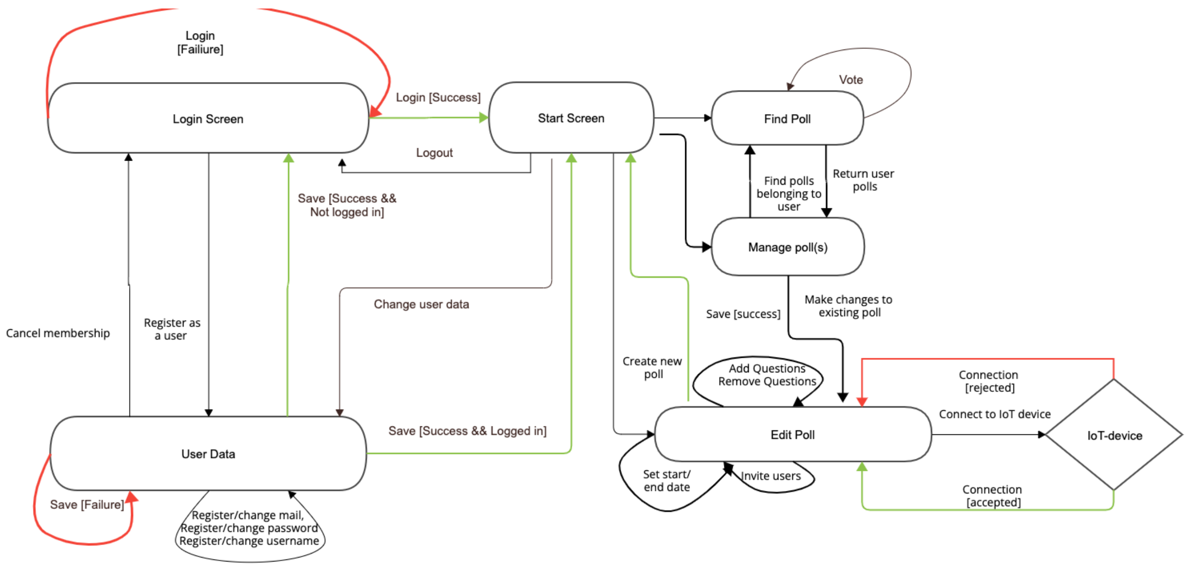

Figure 2: Application Flow Diagram

1. An architectural overview of the application that has been implemented
2. High-level design, domain model, . . . (App assignment A)
3. May involve selected models from Chaps. 5 of the IoT and cloud books

To get an overview of how we wanted to implement the user interface, an application flow diagram was modeled. The diagram displays how the user navigates through the different frames in the front end of the application:

Each screen has in the diagram been modelled as a state, and for every state, the user is able to do some action or provide some input. These actions or inputs are modelled as transitions, and are in the figure above illustrated with arrows. Transitions that results in errors are colored red, and transitions that does not result in errors are colored green. Our application consists of six screens. We have a login screen, where the user can either login, or create a new user. Once the user is logged in, he is directed to a start screen, where he gets the option to do multiple actions. He can search for a poll, which then can be voted on, or he can view and manage his own polls. He can create new polls, and in this state he gets the option to pair the poll with an IoT device.

The example below shows how you may include code. There are similar styles for many other langages - in case you do not use Java in your project. You can wrap the listing into a figure in case you need to refer to it. How to create a figure was shown in Section 2.

```java
public class BoksVolum {

  public static void main(String[] args) {

    int b, h, d;
    String btext, htext, dtext;

    [ ... ]

```

Config	Property	States	Edges	Peak	E-Time	C-Time	T-Time
22-2	A	7,944	22,419	6.6 %	7 ms	42.9%	485.7%
22-2	A	7,944	22,419	6.6 %	7 ms	42.9%	471.4%
30-2	B	14,672	41,611	4.9 %	14 ms	42.9%	464.3%
30-2	C	14,672	41,611	4.9 %	15 ms	40.0%	420.0%
10-3	D	24,052	98,671	19.8 %	35 ms	31.4%	285.7%
10-3	E	24,052	98,671	19.8 %	35 ms	34.3%	308.6%

Table 1: Selected experimental results on the communication protocol example.

```
        int volum = b * h * d;

        String respons =
            "Volum [" + htext + "," + btext + "," + dtext + "] = " + volum;

    }
}
```

4 Prototype Implementation

This section should provide details of how the prototype has been implemented which may involve presentation of suitable code snippets.

5 Test-bed Environment and Experiments

About 2 pages that:

Explains how the prototype has been tested the test-bed environment.

Explains what experiments have been done and the results.

For some reports you may have to include a table with experimental results are other kinds of tables that for instance compares technologies. Table 1 gives an example of how to create a table.

6 Conclusions

Concludes on the project, including the technology, its maturity, learning curve, and quality of the documentation.

The references used throughput the report should constitute a well chosen set of references, suitable for someone interesting in learning about the technology.