Voiland College of Engineering & Architecture
WASHINGTON STATE UNIVERSITY

Project/Research Proposal – Spring 2024

Jeremiah Lynn; 11711049
Jeremiah.lynn@wsu.edu
Faculty Sponsor: Dr. Balasubramanian Kandaswamy

Introduction

Background

Web development is an evolving field, with technology consistently being introduced that solves yesterday's problems and creates tomorrow's difficulties. Once advancement in the field has been the adoption of the microservices architecture. Microservices enable developers to break each application domain into a single service that can be independently deployable. A user's request can be routed through many microservices to create a complete response [1]. The problems microservices seek to solve include, but are not limited to:

1. Moving away from monolithic design to enable less coupling, leading to higher fault tolerance [2].
2. The ability to independently scale overused components of a program easily without scaling underused components [2].
3. The ability to better utilize person hours when developing programs, as each domain can remain separate from other domains [2].

However, microservices come at a cost. Here is a short list of problems that microservices suffer from:

1. Increased complexity due to having multiple separate projects to maintain.
2. Decreased performance as each service will operate independently and require communication instead of having everything happen on a single device. [3]
3. Integration testing becomes very difficult as the system is spread across multiple machines. [2]

Student Preparation

Jeremiah Lynn is a senior at Washington State University completing his major in Computer Science BS. He is passionate about and enjoys studying web technologies. He has work experience developing SDK's and consuming API's building consumable software for various

clients using different web technologies. His love for building software and refining his skill in software development has led him to this project. His coursework at WSU have also aided in his understanding of the basics of web technologies. CptS 489 gave him foundational knowledge on the working of JavaScript, HTML, and CSS. Throughout his coursework, he has utilized various web technologies to create multiple projects.

Student Involvement and Tasks

This project will be developed in its entirety by Jeremiah, under the supervision of Dr. Subu. Half of the project will be developing a recipe sharing application which will be used to study technologies not covered in his coursework, and the other half will be conducting research on various questions centered around developer experience, deployment strategies, and performance metrics when developing a microservices application.

Primarily, he will use docker for container application development [4] and Azure Container Apps [5] for deploying each microservice as a container. Some other technologies might include Dapr [6] for abstracting the communication between microservices, React [7] for developing the front end and SASS [8] for styling.

For the research side of the project, he will be testing the difference between using REST and gRPC for the API. To do this, he will first develop the entire application using REST, perform metric testing, clone the project, and change the API to use gRPC, and perform metric testing. The goal is to better understand these technologies and their use in the development of a real application, including the benefits and drawbacks.

The application he will be developing will be a recipe sharing platform, one that acts a sort of social media for recipe sharing and discovery. The scope of the project will be limited to allow for adequate time for research. However, if time permits, there is a backlog of features to be implemented found at the end of this document. Below are the basic application requirements.

Functional Requirements	Non-Functional Requirements
1. Sign in / signup 2. Recipe creation 3. Recipe querying 4. List of recipes 5. Recipe saving 6. Recipe sharing	1. Visually pleasing 2. Quick loading 3. Simple and intuitive

Though he will be using the microservices architecture, he will be developing a monolithic front end. This will allow him to quickly develop all the front-end features in a single location.

The following milestones have been created to track project and estimate the timeline for this project.

Milestone 1 – Design
The planning phase during which the boundaries for each microservice will be defined, the data design will be written for specific components, and the specific technologies will be selected (i.e. database, ORM, front-end library/framework, etc.).

Milestone 2 – Develop the Application Using REST API
The implementation phase during which the entire application, both frontend and backend, will be developed. During this milestone, some design decisions may need to be refined, which will require updates to the design documents.

Milestone 3 – Performance Testing
The first of the two testing phases during which the performance tests will be written and run using the current API implementation. The testing will use Graphana's k6 to write and perform the tests.

Milestone 4 – Switch to gRPC API
The research phase during which the API will be switched from REST to gRPC. This will require some changes to the application, as REST uses plain text or json and gRPC uses binary data.

Milestone 5 – Performance Comparison and Final Report
The second testing phase during which the same tests from milestone 3 will be used to conduct a final performance review. After which, the research can be compiled into a final report discussing the developer experience when using the microservices architecture and the performance difference between REST and gRPC.

Real World Impact
The impact will be very limited. This project will mostly be for educational purposes. However, the application developed during this project should be publicly accessible, and perhaps iterated on in the future to create a fully-fledged application.

Time Commitment
This class will be used as a 3-credit class. As such, the basic time requirement will be 135 hours. This breaks down to a minimum of 9 hours a week over a 15-week period. This is a reasonable and accomplishable expectation. Within this time commitment, the deliverables will include a working application and a final report.

Outline

Week	To-do
1; 1/8 – 1/12	• Initial Meeting and Planning • Start MS1
2; 1/15 – 1/19	Monday – Holiday • Finish MS1
3; 1/22 – 1/26	• Start MS2
4; 1/29 – 2/2	• Work on MS2 • Biweekly Meeting
5; 2/5 – 2/9	• Work on MS2
6; 2/12 – 2/16	• Work on MS2 • Biweekly Meeting
7; 2/19 – 2/23	Monday – Holiday • Work on MS2
8; 2/26 – 3/1	• Work on MS2
9; 3/4 – 3/8	• Finish MS2 • Midterm App Review and Assessment
10; 3/11 – 3/15	Spring Break
11; 3/18 – 3/22	• Start MS3 • Biweekly Meeting
12; 3/25 – 3/29	• Finish MS3
13; 4/1 – 4/5	• Start MS4 • Biweekly Meeting
14; 4/8 – 4/12	• Work on MS4
15; 4/15 – 4/19	• Finish MS4
16; 4/22 – 4/26	• Start MS5
17; 4/29 – 5/3	• Finish MS5

Feature Catalog

Recipes

- Tags
- User ratings
- User comments
- Modified recipes
- Scalable recipes
- Unit conversion
- Ingredient substitutes
- Recipe creator UI

Cookbooks

- Multiple cookbooks for saved recipes.
- Sharable cookbooks
- Downloadable cookbooks (for offline usage)
- Printable cookbooks

Searching

- Search by ingredients
- Search by chef
- Search by tags

Content Moderation

- Roles for content moderators
- Reporting
- Blocking posts and banning accounts
- Content Moderator UI

Chefs

- Customizable homepage
- User icon
- Ability to Follow

Notifications

- Likes
- Follows
- Comments
- Ratings
- Sharing

Advertisement

- In feed advertisement

Authentication

- SSO (Google, Facebook, etc.)

References

[1] “What Is Microservices Architecture?,” Google Cloud. https://cloud.google.com/learn/what-is-microservices-architecture#:~:text=Microservices%20architecture%20(often%20shortened%20to

[2] GitLab, “What are the benefits of a microservices architecture?,” GitLab, Sep. 29, 2022. https://about.gitlab.com/blog/2022/09/29/what-are-the-benefits-of-a-microservices-architecture/

[3] “10 disadvantages of microservices you’ll need to overcome | TheServerSide,” TheServerSide.com. https://www.theserverside.com/answer/What-are-some-of-the-disadvantages-of-microservices

[4] Docker, “Enterprise Application Container Platform | Docker,” Docker, 2018. https://www.docker.com/

[5] “Azure Container Apps | Microsoft Azure,” azure.microsoft.com. https://azure.microsoft.com/en-us/products/container-apps

[6] D. Maintainers, “Dapr - Distributed Application Runtime,” dapr.io. https://dapr.io/

[7] Meta Open Source, “React,” react.dev. https://react.dev/

[8] “Sass: Syntactically Awesome Style Sheets,” Sass-lang.com, 2019. https://sass-lang.com/