**A picture containing text

Description automatically generated**

**Department of Informatics**

**University of Leicester**

**CO7201 Individual Project**

**Preliminary Report**

**Developing Platform-independent Reusable Functional Components**

**Zewen Wang**

**zw224@student.le.ac.uk**

**229008877**

**Project Supervisor: [Radwan Marwan]**

**Second Marker: [Free Rover C]**

**Word Count: [1853]**

**[June 25, 2023]**

**DECLARATION**

All sentences or passages quoted in this report, or computer code of any form whatsoever used and/or submitted at any stages, which are taken from other people’s work have been specifically acknowledged by clear citation of the source, specifying author, work, date and page(s). Any part of my own written work, or software coding, which is substantially based upon other people’s work, is duly accompanied by clear citation of the source, specifying author, work, date and page(s). I understand that failure to do this amounts to plagiarism and will be considered grounds for failure in this module and the degree examination as a whole.

Name: [Zewen Wang]

Date:[June 25, 2023]

Contents

[**1.** **Aims and Objectives** 3](#_Toc65510906)

[**2.** **Requirements** 3](#_Toc65510907)

[**3.** **Technical Specification** 3](#_Toc65510908)

[**4.** **Requirements Evaluation Plan** 3](#_Toc65510909)

[**5.** **Background Research and Reading list** 3](#_Toc65510910)

[**6.** **Time-plan and Risk Plan** 3](#_Toc65510911)

[**7.** **References** 3](#_Toc65510912)

# **Aims and Objectives**

**Aim:**

When it comes to companies developing their own software, they often encounter a myriad of complex and tedious tasks that need to be addressed. These tasks encompass various aspects, ranging from establishing a private Maven[1] repository and GitLab[2] repository to securely store source code, to setting up a comprehensive CI/CD[3] pipeline using tools like Jenkins[4]. Automating unit tests and ensuring seamless code deployment and launch are also critical aspects of the software development process.

However, the work doesn't end there. Once the code is deployed to the production environment, it becomes essential to have a mature monitoring system in place. This system enables continuous monitoring of the software's operation to promptly detect any abnormal errors or issues that might arise. To accomplish this, setting up effective metrics becomes crucial, allowing for the monitoring of the system's health and performance. In the event of an error or anomaly, it is imperative to activate a notification mechanism that can promptly alert the relevant staff members through email or other means.

These tasks, although merely scratching the surface, cannot be overlooked. The project at hand aims to deliver a comprehensive platform and tutorial that seamlessly integrates all of these components into a single product. The goal is to provide an out-of-the-box solution that addresses the diverse needs of software development, encompassing everything from repository management to CI/CD, automated testing, code deployment, and monitoring. By streamlining these processes, companies can focus more on their core objectives and accelerate their software development endeavors.

**Objectives:**

1. Out-of-box user, permission management and other modules like notification systems, and schedule Job module should be presented.
2. Modules should be scalable and high availability
3. CI/CD pipeline should be set up and can be used to deploy artefact
4. At least integrate with one business module like a photo-sharing system to prove it works well and can save effort when developing another kind of business system.
5. All module should are presented in a microservice architecture

**Challenge:**

1. Research and learn many different fields of knowledge then integrate them
2. Design and develop a sophisticated web solution architecture
3. Design and develop a Good-looking and practical UI interface

# **Requirements**

**Essential**:

* User and permission management modules, login/logout, permission management based on Role with fine-grained control, i.e., method-level permission control
* CI/CD pipeline should be set up, using Jenkins and GitHub, artefact can be deployed to the test environment or production environment automatically
* The monitor system, not only includes os metrics but also includes custom metrics, once an error occurs an email should be sent to the related person
* Build a home photo-sharing module on top of all the base-component, details about the function of this photo-sharing system will depend on subsequent research
* All components should be scalable and high availability, so these components should be a stateless system

**Recommended:**

* + - Build another serverless GP system with the help of reusable components

**Optional:**

* Rebuild these components in a serverless way

# **Technical Specification**

**Methodology: Scrum**

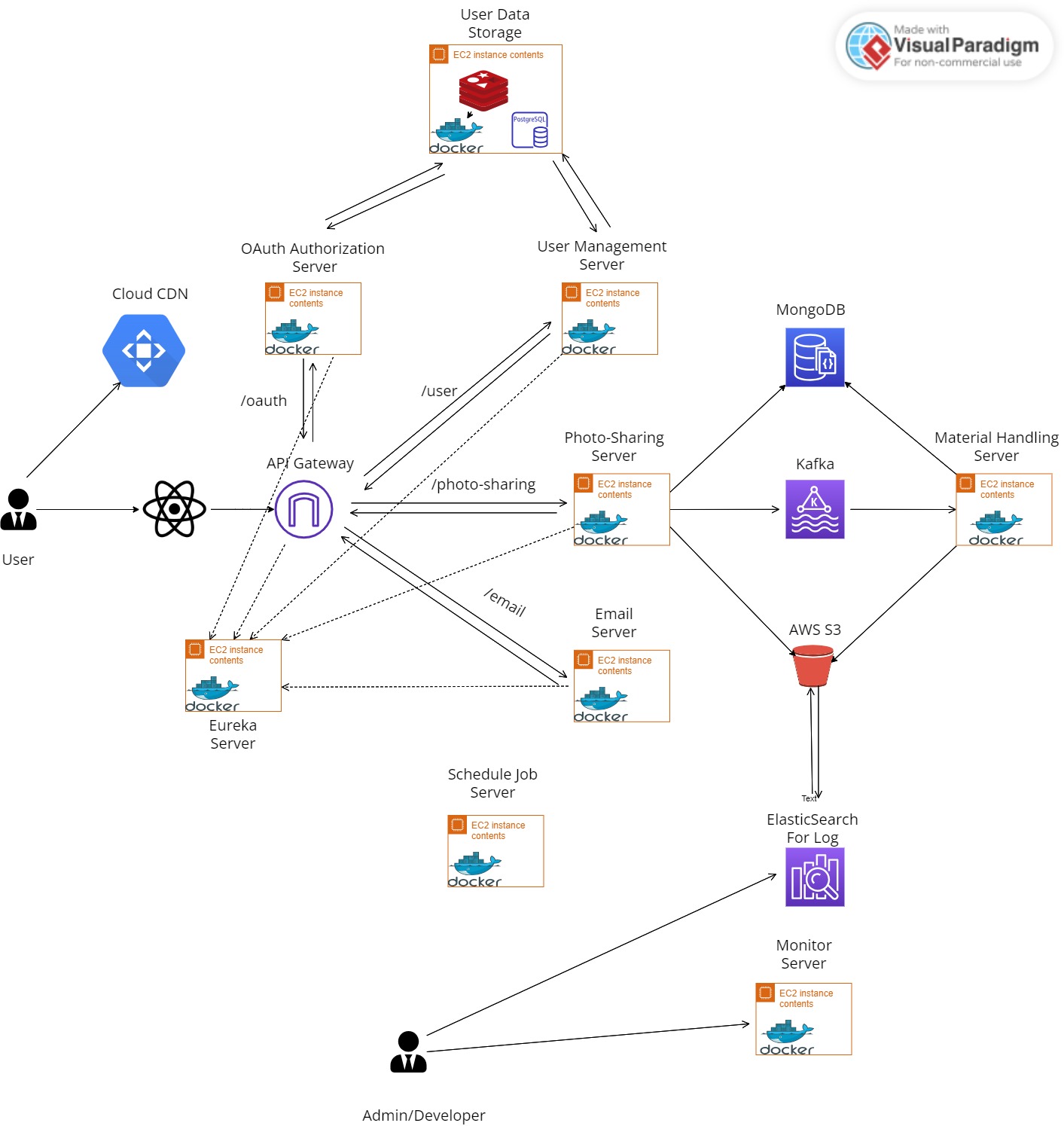
Scrum is an agile project management framework that helps teams effectively collaborate and deliver value in complex projects. It is based on iterative and incremental development principles and promotes adaptive planning, continuous improvement, and self-organizing teams. Scrum divides the project into multiple time-boxed iterations called "sprints," typically lasting 1-4 weeks, during which a potentially shippable increment of the product is produced.

Scrum consists of the following phases:

1. Product Backlog: The product backlog is the prioritized list of user stories, features, and requirements that define the project. It represents the work to be completed and is managed by the Product Owner.
2. Sprint Planning: At the beginning of each sprint, the Development Team, in collaboration with the Product Owner, selects a set of backlog items to work on. They determine the sprint goal, define the tasks needed to accomplish the selected items, and estimate the effort required.
3. Daily Scrum: The Daily Scrum is a short, time-boxed meeting held each day during the sprint. The Development Team members synchronize their work, discuss progress, and plan the work for the next 24 hours. The Scrum Master facilitates the meeting but does not actively participate.
4. Sprint Execution: During this phase, the Development Team works on the tasks identified during the Sprint Planning. They self-organize to accomplish the work and collaborate closely to deliver the planned increment.
5. Sprint Review: At the end of each sprint, a sprint review is conducted. The Development Team presents the completed work to the stakeholders, including the Product Owner and other relevant parties. Feedback is gathered, and the Product Owner may choose to update the product backlog based on the review.
6. Sprint Retrospective: Following the sprint review, the team holds a retrospective meeting to reflect on the sprint process and identify areas for improvement. They discuss what went well, what could be improved, and create action items to enhance their performance in future sprints.
7. These phases are repeated for each sprint, allowing the team to continuously deliver increments of the product and adapt based on feedback and changing requirements.

**Technical Solutions:**

Below is the core system architecture including the main tech stack.



|  |  |  |  |
| --- | --- | --- | --- |
| Component | Type | Name | Summary |
| Language | Backend | Java/Python/Golang |  |
| Frontend | ReactJs |  |
| Framework | Backend | Springboot |  |
| OAuth 2.0 | Permission, SSO |
| Spring Gateway | Traffic Routing |
| Spring Security | Role-based permission Control |
| Spring Eureka | Service registration |
| Spring Cloud Sleuth | Tracing and Debugging |
| Kafka | MQ |
| Spring Cloud Config | Configuration Center |
| Hibernate | ORM framework |
| Django | The web framework for python |
| Gin | The web framework for Golang |
| GORM | ORM framework for Golang |
| Material-UI | UI for React |
| React Router | Traffic routing for React |
| Storage | PostgreSQL | RDBMS |
| Redis | Cache |
| MongoDB | No-SQL |
| AWS S3 | File Storage |
| CI/CD |  | Jenkins | Automation Deploy tool |
| GitHub | Source code Version Control |
| Maven | Jar repository |
| Containerized |  | Docker | Container |
| Kubernetes | Orchestration |
| OS |  | AWS EC2 | Virtual Machine |
| Monitor |  | Prometheus | Time-series database |
| Grafana | UI |
| ElasticSearch | Log searching |
| IDE |  | IntelliJ |  |

# **Requirements Evaluation Plan**

1. User&Perimisson Module provide full function and can be run successfully, providing Role-based control
   1. Login/Logout, SSO
   2. Role-based control
   3. Method level permission
2. CI/CD pipeline successfully automates test and deploy
   1. One-click then all things happened automatically
3. All components run within Docker and are scaled by K8S
   1. The module should be scaled automatically
4. A log collect-search Module
5. Monitor Module provides overall monitoring of the whole system
   1. Detect abnormal metrics and send notifications to related people
6. Photo-Sharing Module provide full function and can be run successfully
   1. Upload, storage, and share photos/Video
   2. Photo editing like watermark, resize function
   3. Materials filter based on metadata/upload date etc.
7. Reusable components should prove to support large-scale calls which means the API interface should act normally even under pressure tests.

The above criteria are the main part to evaluate whether my project is a success or not.

# **Background Research and Reading list**

Most web applications have gone through the same evolution, from a monolithic architecture to multi-tier and then microservice architecture, although business models are ever-changing, while the underlying architecture systems have certain similarities. They should have some fundamental capabilities, in a big way, availability, durability, resilience and disaster recovery, these are the aspect that must be considered when designing a software architecture[5].

About Methodology – scrum is widely used in industry, it is a tool dedicated to improving development efficiency while ensuring software quality[6].

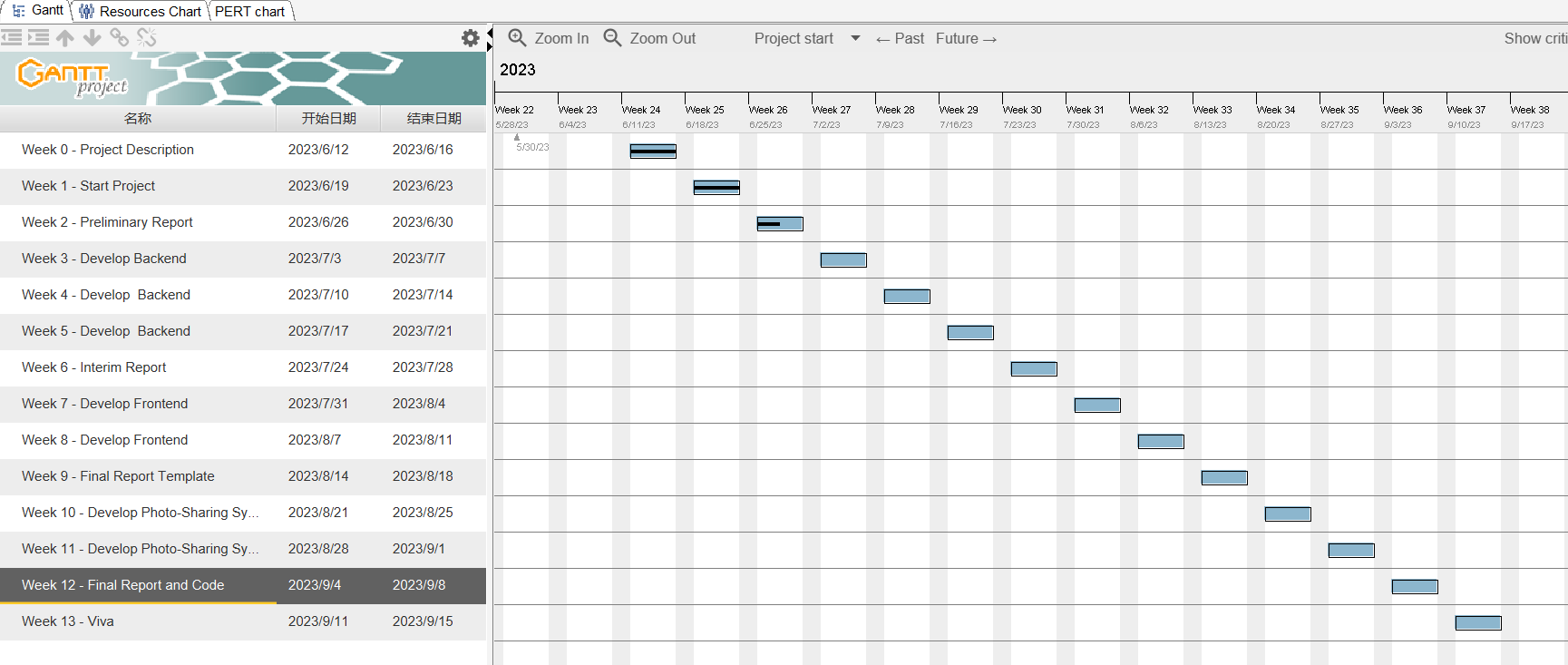
Also, there are some best practices when talking about developing web applications, it can be followed, such as design for failure, redundancy and fault recovery etc[7].

# **Time-plan and Risk Plan**

**Time-plan:**

|  |  |  |
| --- | --- | --- |
| **Week** | **Task** | **Delivery** |
| Week 0(12 – 16 June) | Research | Project Description |
| Week 1(19 – 23 June) | Build base architecture and Integrate OAuth 2.0 |  |
| Week 2(26 – 30 June) | Integrate Spring Security, PostgreSQL, Redis and Hibernate | Preliminary Report |
| Week 3(3 – 7 July) | Integrate Docker and K8S, deploy all components to AWS EC2 as docker image |  |
| Week 4(10 – 14 July) | Develop a Role-based permission function and Build CI/CD pipeline environment(Jenkins) |  |
| Week 5(17 – 21 July) | Build log collection platform and ElasticSearch, Learnning Scrum |  |
| Week 6(24 – 28 July) | Build Monitor System and Develop collect metrics from Reusable Component and Write Interm Report | Interim Report |
| Week 7(31 – 4 Aug) | Learn ReactJs and Develop Front UI |  |
| Week 8(7 – 11 Aug) | Learn ReactJs and Develop Front UI |  |
| Week 9(14 – 18 Aug) | Learn ReactJs and Develop Front UI and Final Report Template | Final Report Template |
| Week 10(21 – 25 Aug) | Develop Photo-Sharing System |  |
| Week 11(28 – 1 Sep) | Develop Photo-Sharing System |  |
| Week 12(4 – 8 Sep) | Develop Photo-Sharing System and write Final Report | Final Report and Code |
| Week 13(11 – 15 Sep) |  | Viva |

**Gantt-Chart**

****

Below is the original Gantt project file

****

**Risk-plan:**

1. As I list above, there are a lot of things that should be integrated with the platform, in this scenario, time is the key resource and the project will be developed according to its priority. And the priority based on the Requirements Evaluation Plan
2. ReactJs: The author has no experience with front-end language, and due to time limitations, probably can’t build the whole module UI. And some modules may have a rough UI.
3. The choice of the technology stack is based on the most popular technologies currently on the market and the technology selection may change as the development progresses.

# **References**

1. Porter, B. (no date) *Maven – Welcome to Apache Maven*. Available at: https://maven.apache.org/.
2. *The DevSecOps Platform* (no date). Available at: https://about.gitlab.com/.
3. *What is CI/CD?* (no date). Available at: https://www.redhat.com/en/topics/devops/what-is-ci-cd.
4. *Jenkins* (no date). Available at: https://www.jenkins.io/.
5. *阿里电商架构演变之路-阿里云开发者社区* (no date). Available at: https://developer.aliyun.com/article/161190.
6. *What is Scrum?* (no date). Available at: https://www.scrum.org/resources/what-scrum-module.
7. *On Designing and Deploying Internet-Scale Services* (no date). Available at: https://www.usenix.org/legacy/event/lisa07/tech/full\_papers/hamilton/hamilton\_html/index.html.