S2 SUBMISSION DOCUMENT

1.Tech Report

What is Docker?

In simple terms docker is a way to containerize applications (putting code in boxes that can work on their own). It magically makes a virtual computer, but guess what - they aren't really virtual computers. Containers are boxes that have no host Operating system, so they are independent of the device they run on.

How Does Docker Work?

At its core, Docker utilises containerization technology to package and isolate applications and their dependencies into containers. These containers leverage the host system's kernel and share resources efficiently while remaining isolated from one another. Docker employs a client-server architecture, where the Docker client interacts with the Docker daemon to build, manage, and run containers.

Containerized Development Environment Example:

Create a dockerfile(plain text-file that specifies instructions for building a Docker image) in the root directory of the project with the following content.

Dockerfile Copy code

```
# Use an official OpenJDK 11 image as the base image
FROM adoptopenjdk/openjdk11:alpine-jre

# Set the working directory in the container
WORKDIR /app

# Copy the JAR file generated by Maven/Gradle to the container
COPY target/*.jar app.jar

# Expose the port the application runs on
EXPOSE 8080

# Command to run the application
CMD ["java", "-jar", "app.jar"]
```

Key advantages/Features of Docker

1.Dependency Management & Isolation

Docker allows developers to isolate application dependencies within containers, eliminating compatibility issues and dependency conflicts. Each container operates in isolation from others, ensuring that changes or updates to one component do not impact others.

2.Continuous Integration and Deployment (CI/CD)

Docker facilitates seamless integration with CI/CD pipelines, enabling automated testing, building, and deployment of JHipster web applications. Containers provide a consistent environment for testing and deployment, reducing the likelihood of errors and discrepancies between environments.

3.Scalability and Resource Efficiency

Docker enables horizontal scaling of JHipster web applications by spinning up multiple containers to handle increased traffic or workload.

4.Cross-Platform Compatibility

Docker containers are platform-independent and can run consistently across different operating systems and cloud platforms.

Common Docker Commands

1.*docker build -t my-jhipster-app*: Builds a Docker image for the application i.e “my-jhipster-app”

2.*docker run -d -p 8080:8080 --name jhipster-app my-jhipster-app*: Launches a Docker container based on the built image.

3.docker-compose up --scale web=3: Scales the application by spinning up multiple container instances.

4.*docker stop*: Stops a running Docker container.

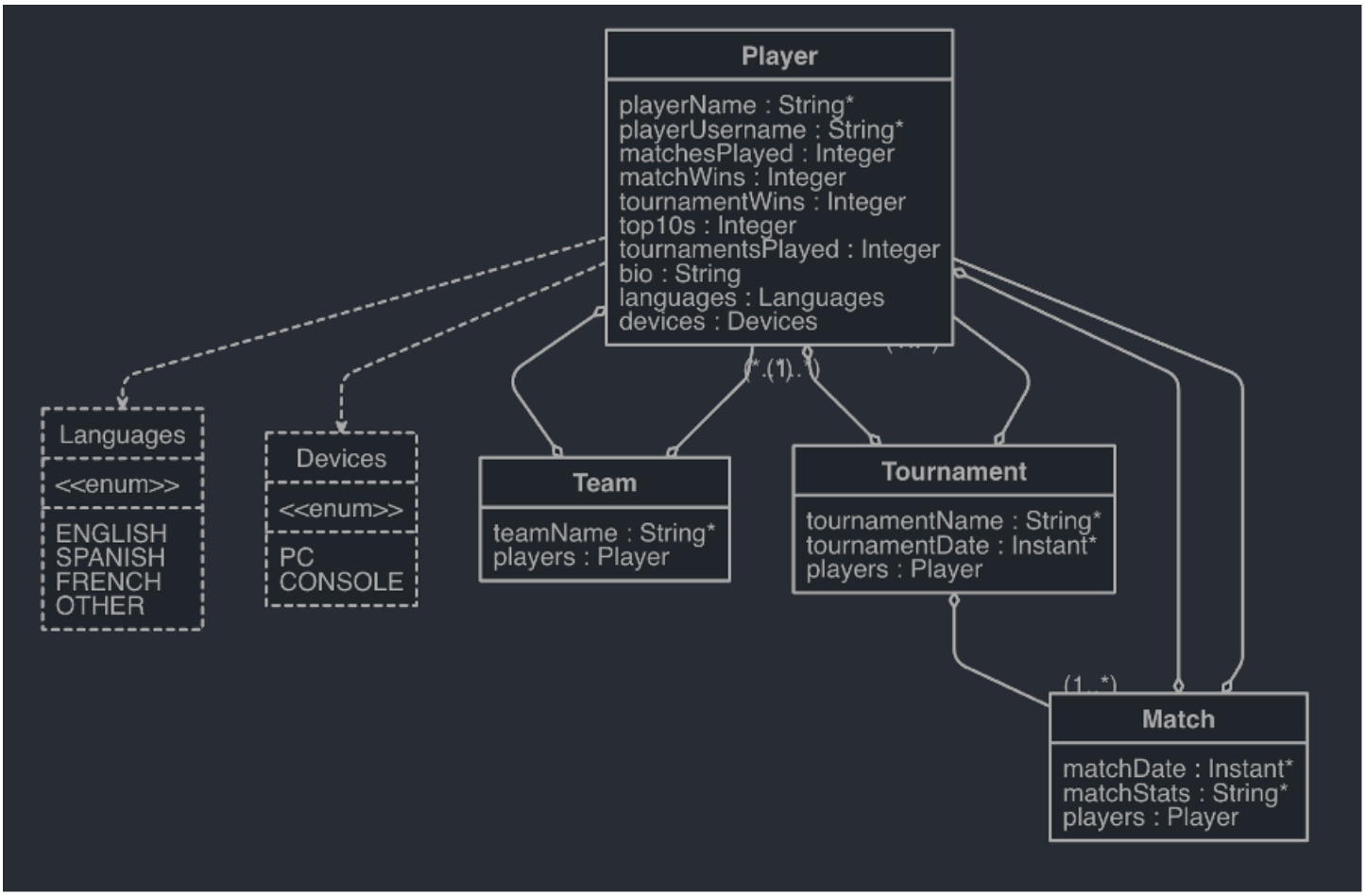
5.*docker remove*: Removes a Docker container.

6.*docker search*: Searches for specific Docker images available in repositories.

Conclusion

In summary, Docker simplifies the deployment and management of applications by encapsulating them into portable and scalable containers. Its flexibility, efficiency, and compatibility make it a preferred choice for modern software development and deployment workflows.

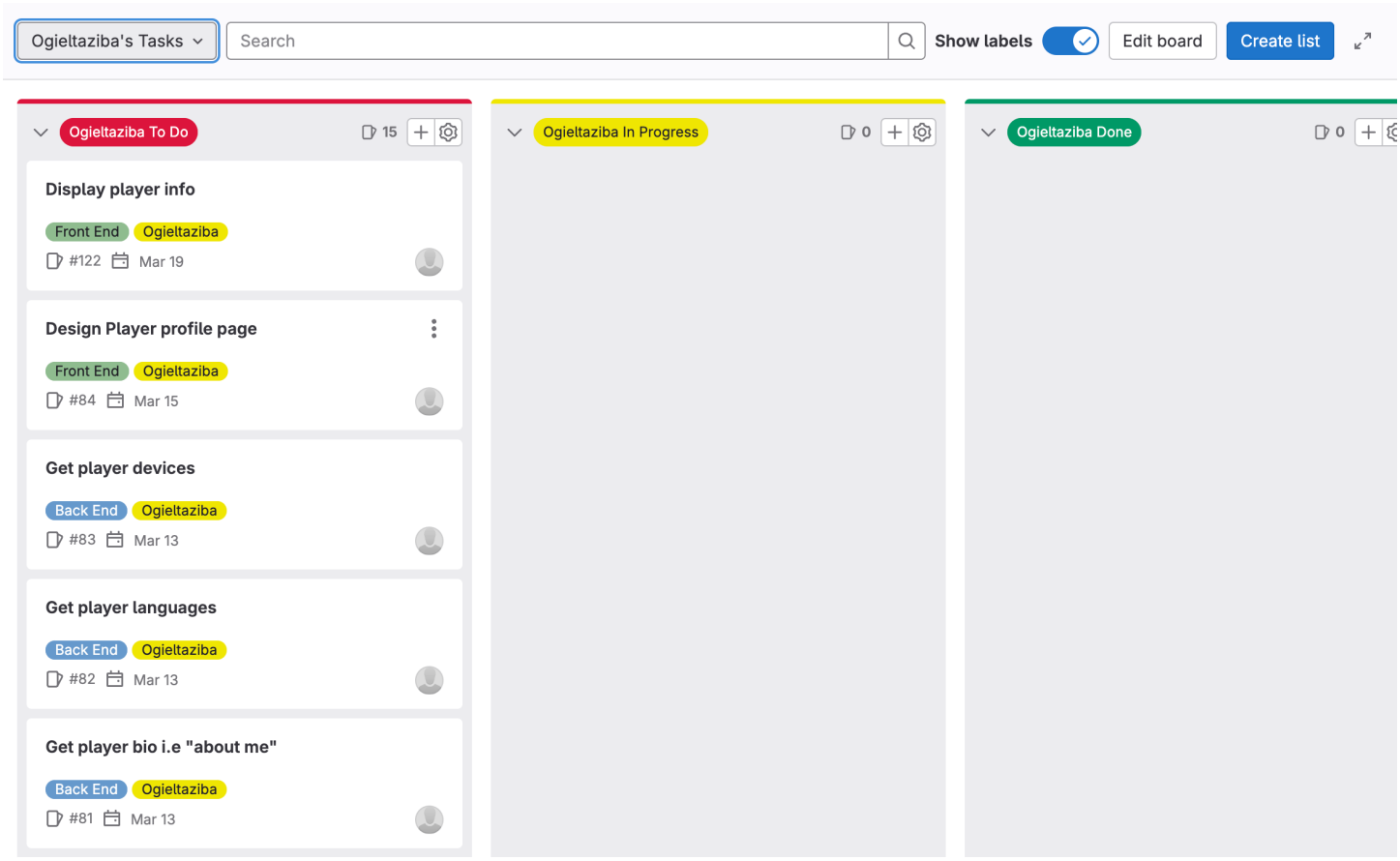
2.JDL

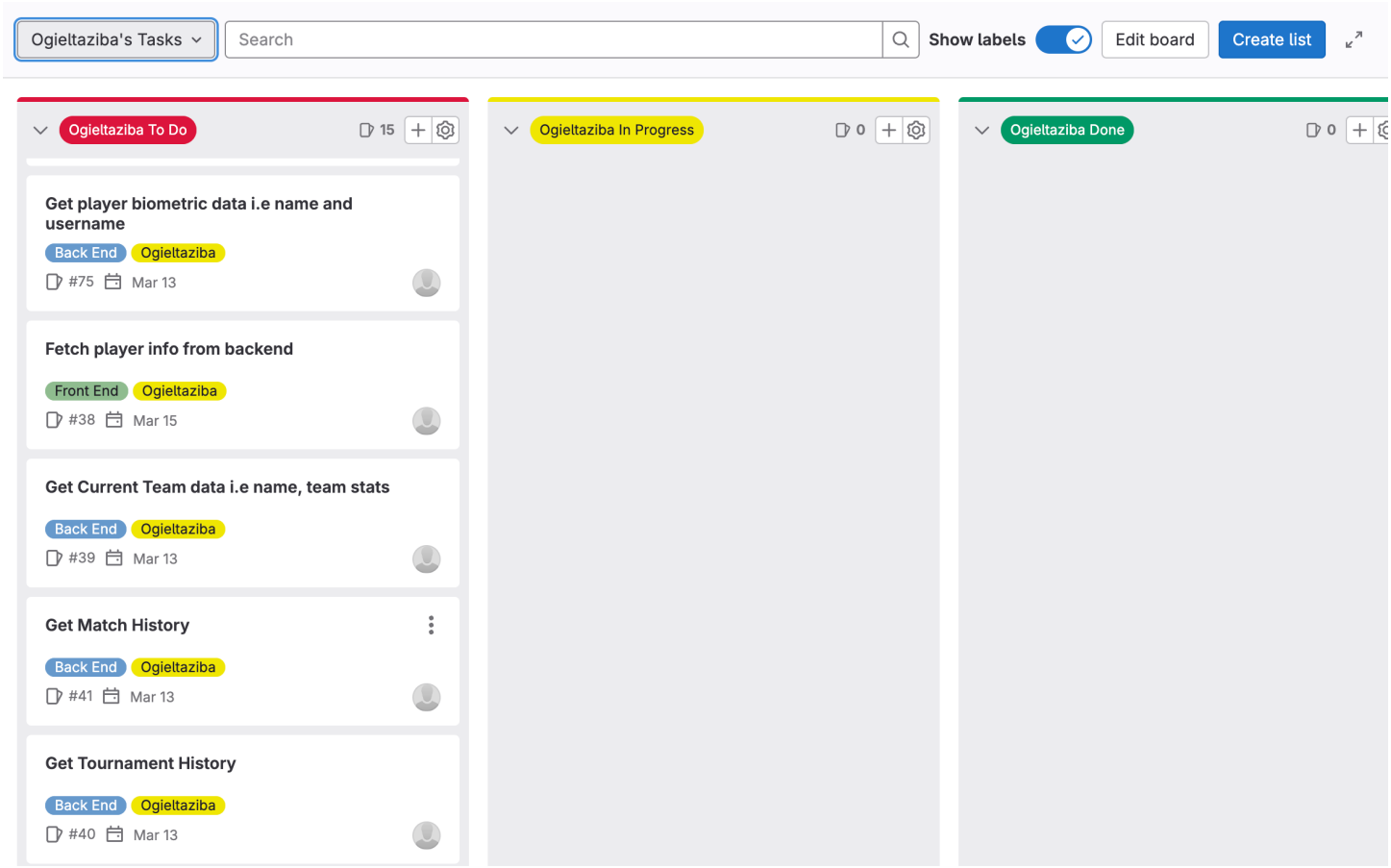


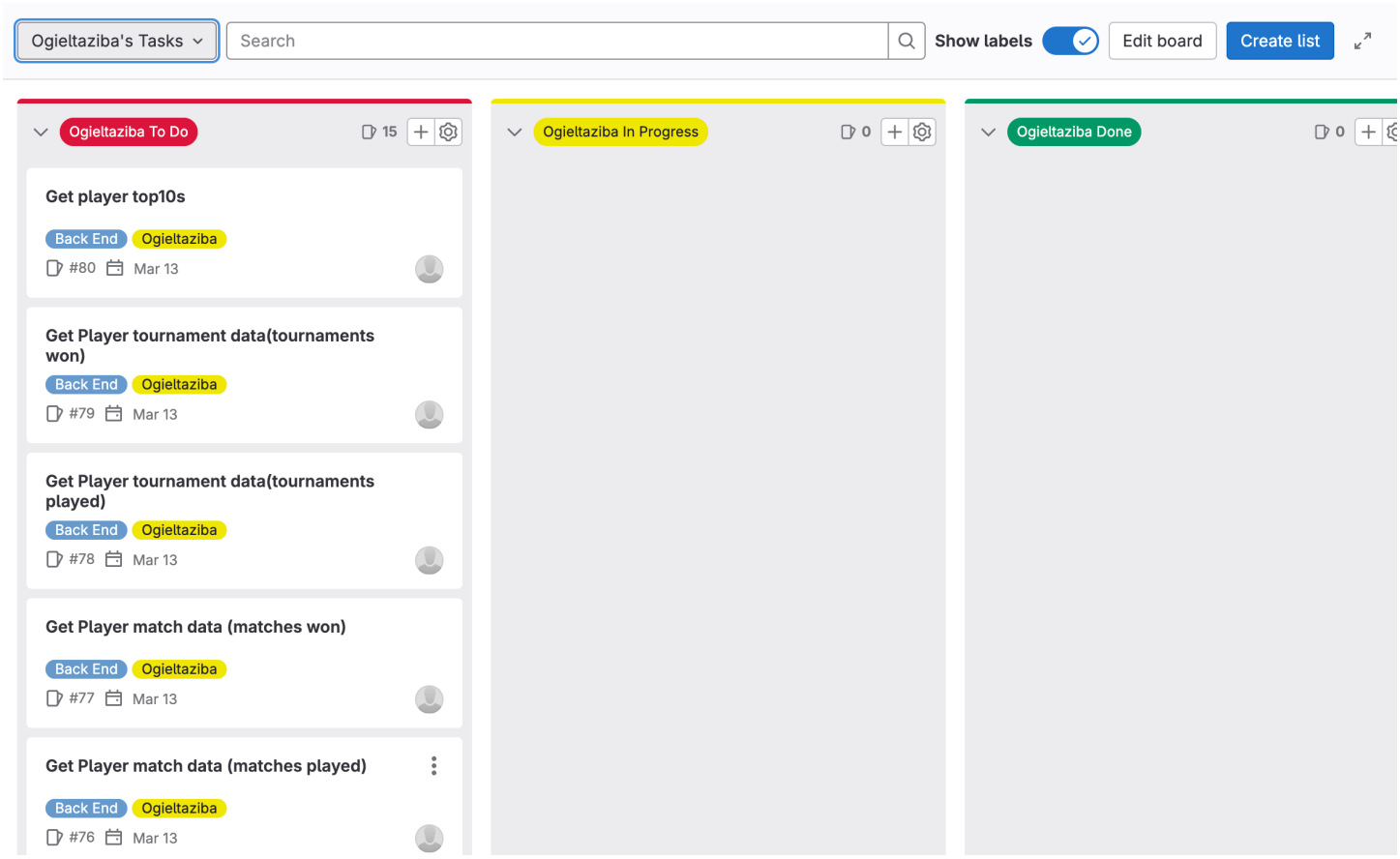

```
entity Player {
 playerName String required,
 playerUsername String required,
 matchesPlayed Integer,
 matchWins Integer,
 tournamentWins Integer,
 top10s Integer,
 tournamentsPlayed Integer,
 bio String,
 languages Languages,
 devices Devices
}
entity Team {
 teamName String required,
 players Player
}
```

```
entity Tournament {
 tournamentName String required,
 tournamentDate Instant required,
 players Player
}
entity Match {
 matchDate Instant required,
 matchStats String required,
 players Player
}
enum Languages {
 ENGLISH, SPANISH, FRENCH, OTHER
}
enum Devices {
 PC, CONSOLE
}
relationship ManyToMany {
 Player to Team,
 Player to Game,
 Player to Tournament,
 Match to Player
}
relationship OneToMany {
 Team to Player,
 Game to Player,
 Tournament to Player,
 Tournament to Match,
 Match to Player
}
```

3.KANBAN Cards







4.Timesheets

Team sheet Number/ID: ODouglas03

Team member name: Ogieltaziba Douglas

Team representative (secratary) Haiwei

Team meeting sign off date:

Date from: 13.02.24

Date until:

Task	Date	Start time	End time	Total Hours
Team meeting with tutor, discussed tech reports	13.02.24	2:00 PM	3:00 PM	1:00
Conducted research on docker for tech report	15.02.24	4:00 PM	6:30 PM	2:30
Team meeting discussed tech reports again	16.02.24	12:30 PM	1:30 PM	1:00
More research on docker, and first draft complete	17.02.24	3:00 PM	5:30 PM	2:30
				0:00
				0:00
				0:00

Total Hours **7:00**

Team sheet Number/ID: ODouglas4

Team member name: Ogieltaziba Douglas

Team representative (secratary) Ogieltaziba Douglas **Date from:** 20.02.24

Team meeting sign off date: **Date until:**

Task	Date	Start time	End time	Total Hours
Team meeting with tutor	20.02.24	2:00 PM	3:00 PM	1:00
Working on JDL and tech report for S2	21.02.24	2:00 PM	5:00 PM	3:00
Team meeting combining JDLs, discussing format and Kanban	23.02.24	12:00 PM	3:00 PM	3:00
Working on S2- Kanban cards	25.02.24	6:00 PM	7:00 PM	1:00
Working on S2- Finalizing tech report and S2	26.02.24	6:00 PM	10:00 PM	4:00
				0:00
				0:00

Total Hours **12:00**