Full-stack Web Development

for Auto-Assessment Platform (AASP)

Final Year Project
Lee Jun Wei

Agenda

01 **Introduction**
Background, problems & objective

02 **Related Products**
Summary of findings

03 **Design**
Requirements & considerations

04 **Implementation**
Overview of technologies used

04 **Conclusion**
Achievements & future work

05 **Live demo!**
Demo of main features

01

Introduction

Background

Assessments

- Important feedback channel for **educators**
- Provide **learners** with measure of progress

Benefits of online platforms

- Automated grading
- Ease of distribution
- Objectivity of marking

Background

SC1007 Data Structures and Algorithms

- Assessments distributed with **HackerEarth**
  - Commercial platform
  - High cost
- In-house solution is desired
  - Full control over the platform and its data

hackerearth

Prior Works

Kenneth Soh

Designed and developed AASP

Yap Guan Sheng

MCQ questions
Quiz feedback feature
Improved user interface

Lee Jun Wei

Problems

SECURITY

Secure coding practices not enforced

Backend services exposed

RELIABILITY

Bugs that lead to a complete crash

MAINTAINABILITY

Complicated architecture

Poor backwards-compatibility of dependencies

Built on a depreciated projects last updated 7 years ago

Objectives

- To **redesign and develop** a new in-house Automated Assessment Platform (AASP)

- Developed with **Security, Reliability, Maintainability** in mind

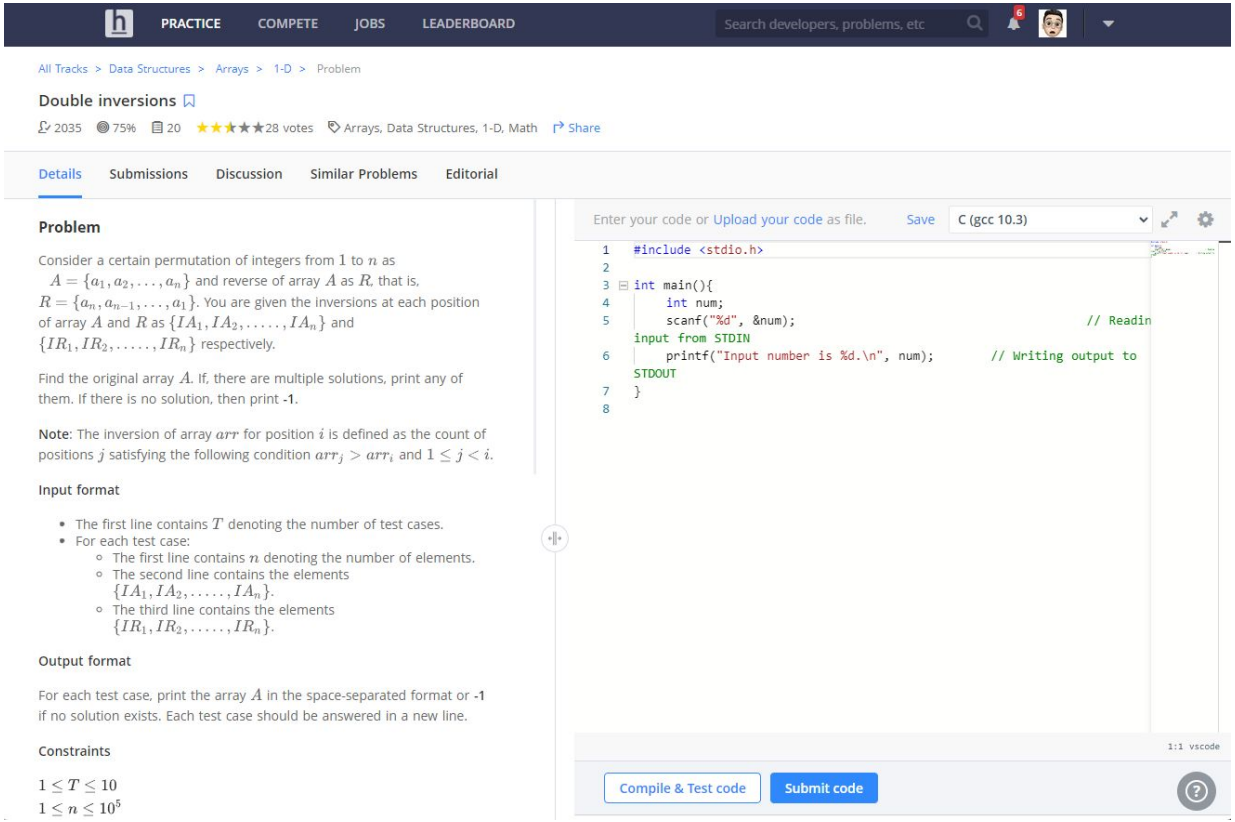
02

Review of Related Products

HackerRank, HackerEarth, Leetcode

User Interface Summary

of related products



Positive traits

- Clean and uncluttered
- Horizontally-split layout
- Text formatting support

Useful Features

of related products

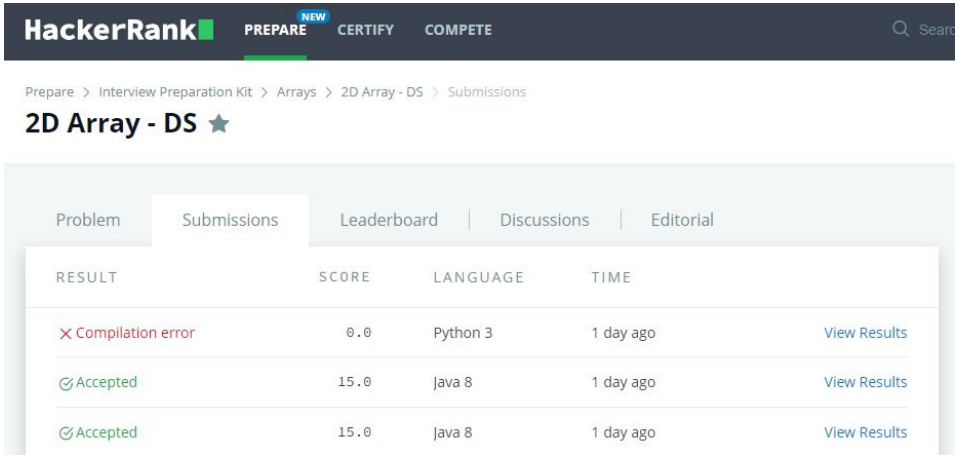
```
gest
(longest, r - l + 1)
```

Run Code ^ Submit

- Run code against sample test case

Useful Features

of related products



- Submission history and test case details

Useful Features

of related products

HackerRank Prepare > Interview Preparation Kit > Arrays > 2D Array - DS

Exit Full Screen View

Problem

Submissions

Leaderboard

Given a 6×6 2D Array, arr:

```
1 1 1 0 0 0
0 1 0 0 0 0
1 1 1 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0
```

An hourglass in A is a subset of values with indices falling in this pattern in arr's graphical representation:

```
a b c
 d
e f g
```

There are 16 hourglasses in arr. An hourglass sum is the sum of an hourglass' values. Calculate the hourglass sum for every hourglass in arr, then print the maximum hourglass sum. The array will always be 6×6.

Example

$arr =$

```
-9 -9 -9  1 1 1
 0 -9  0  4 3 2
```

Change Theme Language Python 3

```python
#!/bin/python3

import math
import os
import random
import re
import sys

#
# Complete the 'hourglassSum' function below.
#
# The function is expected to return an INTEGER.
# The function accepts 2D_INTEGER_ARRAY arr as parameter.
#

def hourglassSum(arr):
    # Write your code here
    pass

if __name__ == '__main__':
    fptr = open(os.environ['OUTPUT_PATH'], 'w')

    arr = []
```

Line: 33 Col: 1

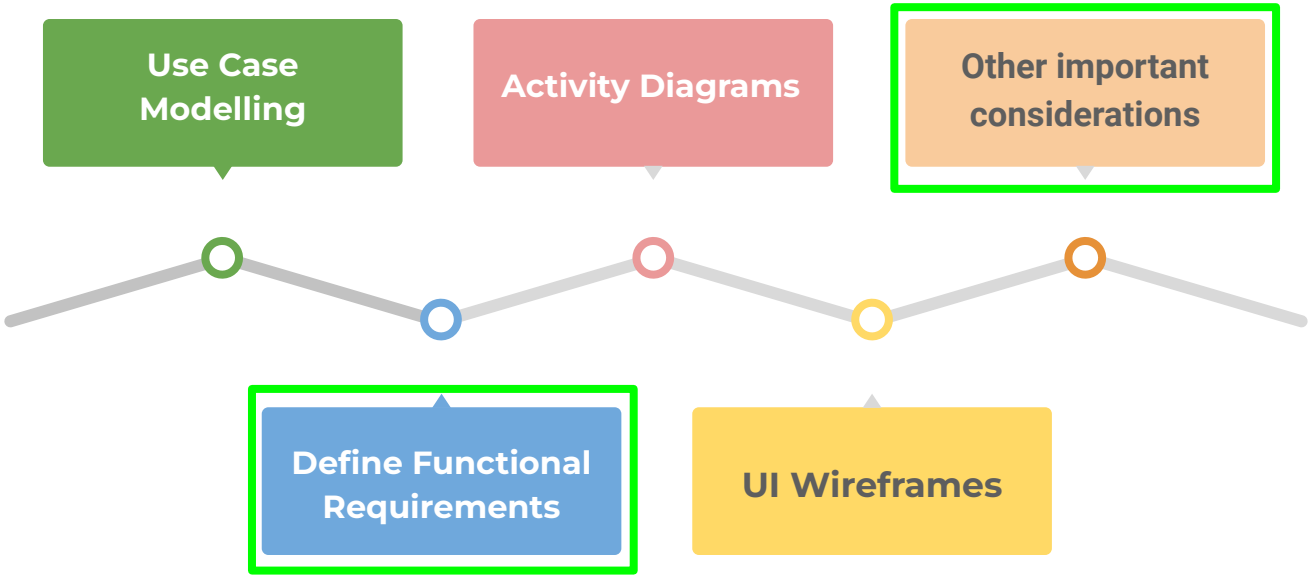
Upload Code as File Test against custom input Run Code Submit Code

- Integrated code editor

03

Design and Considerations

Design Methodology



Summary of Functional Requirements

AASP

- Automated grading
- User Authentication
- Strict access control to resources

Educators

- **Question banks** to store questions
- **Courses** to organise students
- Distribute **Assessments** to students
- View **Reports** of assessment attempts

Students

- Take assessments

Security Considerations

Web Application Security

- Main attack vector for attackers
- Proper security measures must be implemented

Network Exposure

- Minimise attack surface
- Only the web application should be exposed
- Internal services should not be exposed

Maintainability Considerations

Primary Programming Language

- Select language with lower learning curve
- Taught as part of the curriculum

Web Framework Selection

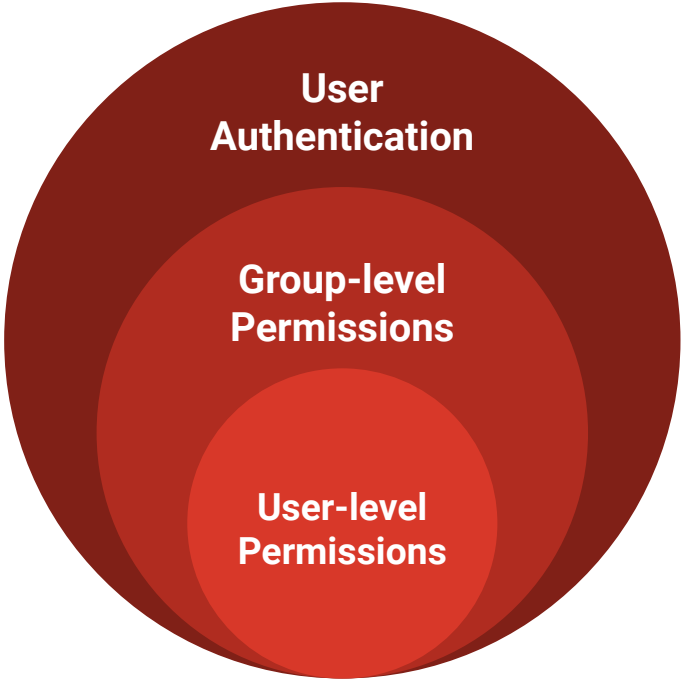
- Easy to learn
- Well-documented
- Large community

Don’t reinvent the wheel

- Integrate good existing projects if possible

Access Control

3 layered approach

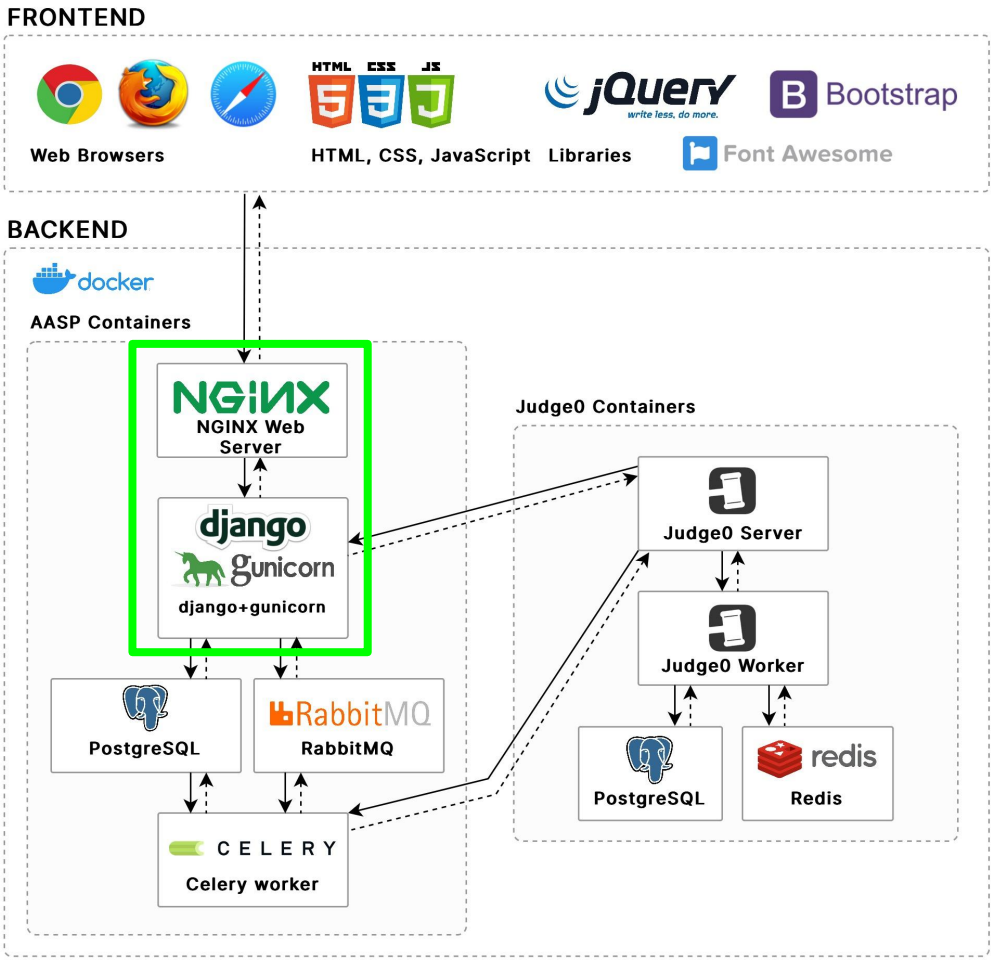


04

Implementation

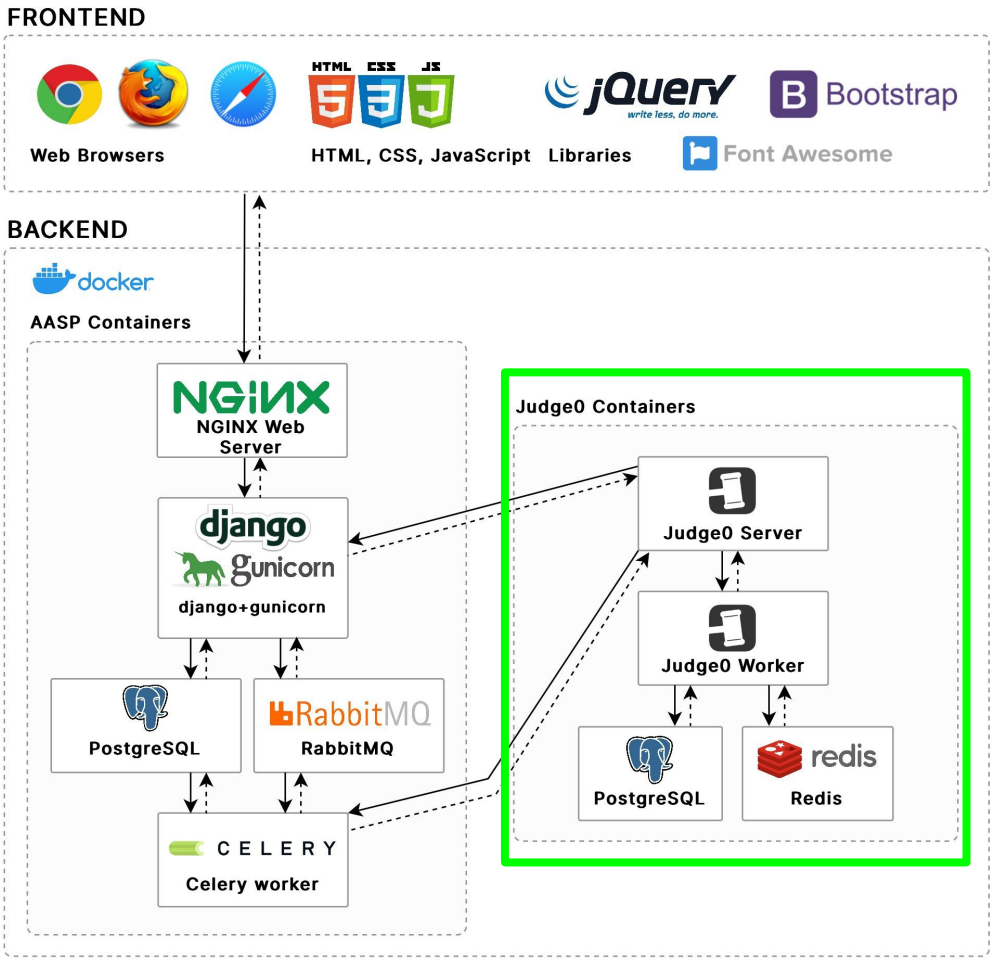
Technologies used

Backend Server



Django Web Framework

- Most popular Python web framework
- Focused on **API stability** and **forwards-compatibility**
- Built-in security features

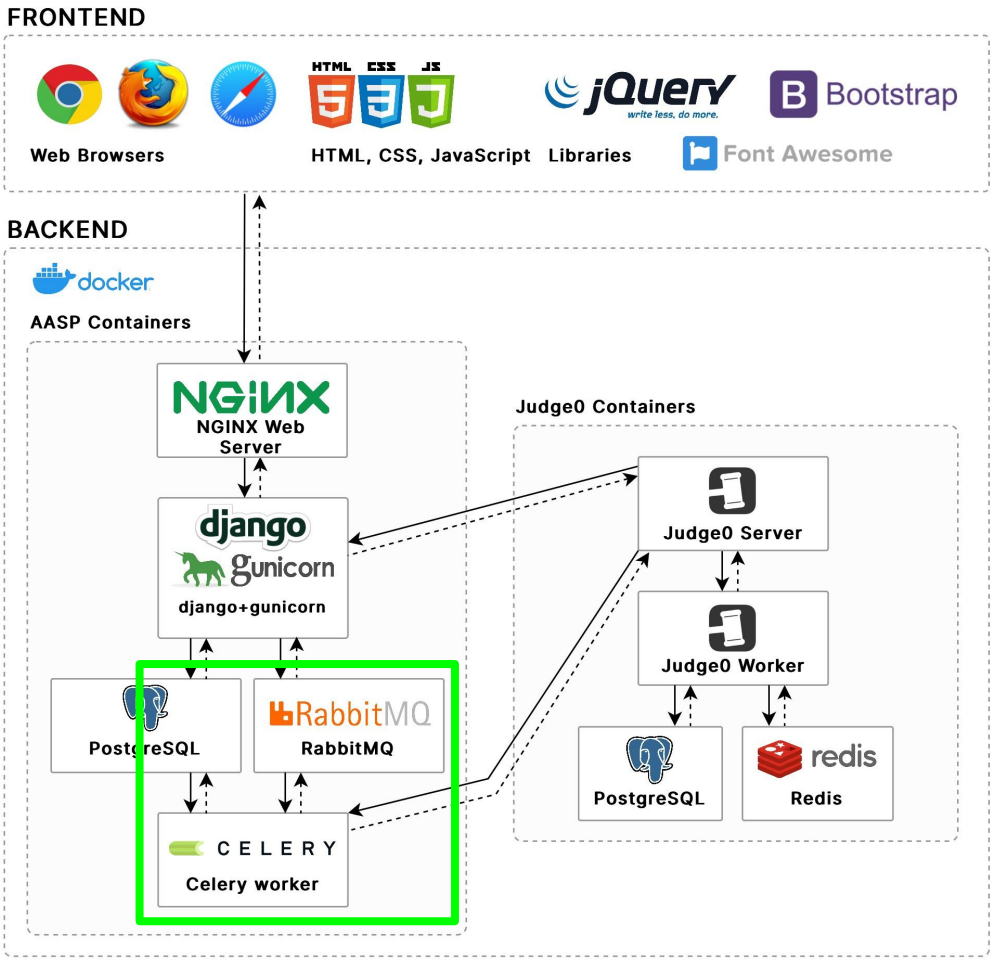


Code Execution Engine

Judge0 API

Executes code securely in a sandboxed environment

- Open-source
- Actively maintained
- Well documented



Asynchronous Task Queue

Python Celery and RabbitMQ

- Run background tasks required for automated grading

Deployment

Docker Engine

- AASP is containerised
- Consistent deployment on various platforms
- Eliminates hassle of installing dependencies

3-Step Process

1. Clone the repository
2. Update configuration file to set secret keys
3. Run `docker-compose up -d`

05

Conclusion

Conclusion

What was achieved

- **Redesign and development** of the AASP
- Design choices that promotes **Security, Reliability and Maintainability**
- **Simplified deployment process**

Future works

- More question types
- Email notifications & reminders

06

Live demo

http://172.21.148.184

Thank You!

Any questions?