



# **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



# 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

The screenshot shows a web-based coding interface. At the top is a dark navigation bar with a logo, links for PRACTICE, COMPETE, JOBS, and LEADERBOARD, a search bar, and a user profile icon. Below this is a breadcrumb trail: All Tracks > Data Structures > Arrays > 1-D > Problem. The problem title is 'Double inversions' with a bookmark icon. Below the title are statistics: 2035 views, 75% solved, 20 submissions, 5 stars (28 votes), and tags: Arrays, Data Structures, 1-D, Math. A 'Share' button is also present. A horizontal tab bar contains 'Details' (selected), 'Submissions', 'Discussion', 'Similar Problems', and 'Editorial'. The 'Details' tab shows the problem description, input/output formats, and constraints. The right side of the interface is a code editor with a light blue theme, showing a C program that reads an integer and prints it. The editor has a toolbar with 'Save', a compiler selector set to 'C (gcc 10.3)', and icons for undo, redo, and settings. At the bottom of the editor are 'Compile & Test code' and 'Submit code' buttons, and a help icon.

**Problem**

Consider a certain permutation of integers from 1 to  $n$  as  $A = \{a_1, a_2, \dots, a_n\}$  and reverse of array  $A$  as  $R$ , that is,  $R = \{a_n, a_{n-1}, \dots, a_1\}$ . You are given the Inversions at each position of array  $A$  and  $R$  as  $\{IA_1, IA_2, \dots, IA_n\}$  and  $\{IR_1, IR_2, \dots, IR_n\}$  respectively.

Find the original array  $A$ . If, there are multiple solutions, print any of them. If there is no solution, then print -1.

**Note:** The inversion of array  $arr$  for position  $i$  is defined as the count of positions  $j$  satisfying the following condition  $arr_j > arr_i$  and  $1 \leq j < i$ .

**Input format**

- The first line contains  $T$  denoting the number of test cases.
- For each test case:
  - The first line contains  $n$  denoting the number of elements.
  - The second line contains the elements  $\{IA_1, IA_2, \dots, IA_n\}$ .
  - The third line contains the elements  $\{IR_1, IR_2, \dots, IR_n\}$ .

**Output format**

For each test case, print the array  $A$  in the space-separated format or -1 if no solution exists. Each test case should be answered in a new line.

**Constraints**

$$1 \leq T \leq 10$$
$$1 \leq n \leq 10^5$$

## Positive traits

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

# Useful Features

## of related products

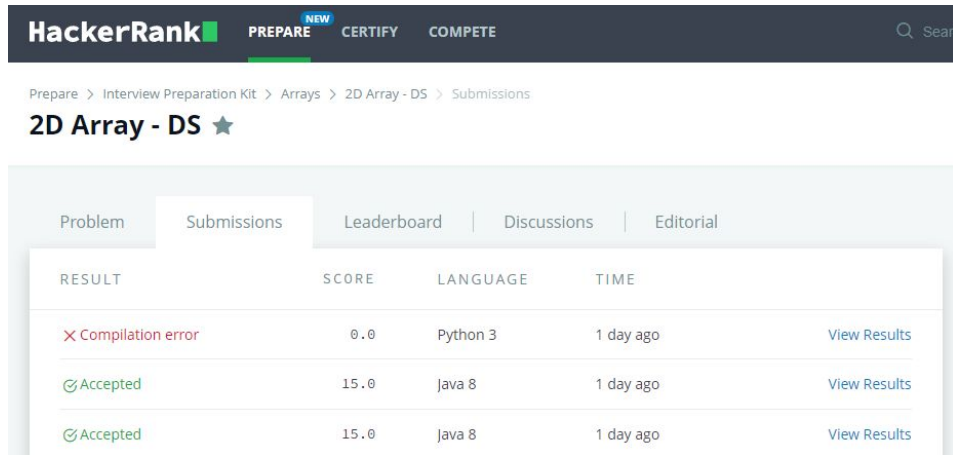
```
test  
c(longest, r - 1 + 1)
```

▶ Run Code ^

Submit

- Run code against sample test case

# Useful Features of related products



The screenshot shows the HackerRank interface for the '2D Array - DS' problem. The navigation bar includes 'HackerRank', 'PREPARE' (with a 'NEW' badge), 'CERTIFY', and 'COMPETE'. A search bar is on the right. The breadcrumb trail is 'Prepare > Interview Preparation Kit > Arrays > 2D Array - DS > Submissions'. The problem title '2D Array - DS' is followed by a star icon. Below the title are tabs for 'Problem', 'Submissions', 'Leaderboard', 'Discussions', and 'Editorial'. The 'Submissions' tab is active, displaying a table of submission history.

RESULT	SCORE	LANGUAGE	TIME	
✗ Compilation error	0.0	Python 3	1 day ago	<a href="#">View Results</a>
✓ Accepted	15.0	Java 8	1 day ago	<a href="#">View Results</a>
✓ Accepted	15.0	Java 8	1 day ago	<a href="#">View Results</a>

- Submission history and test case details

# Useful Features of related products

The screenshot displays the HackerRank interface for a problem titled "2D Array - DS". The interface is divided into several sections:

- Problem Statement:** Given a  $6 \times 6$  2D Array, `arr`. It shows a grid of values: `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`, and `0 0 0 0 0 0`. It defines an hourglass in `arr` as a subset of values with indices falling in a specific pattern and asks to calculate the maximum hourglass sum.
- Example:** Shows an example array `arr` with values: `-9 -9 -9 1 1 1` and `0 -9 0 4 3 2`.
- Code Editor:** A Python 3 code editor with a dark theme. It contains a template for a function `hourglassSum(arr)` that needs to be completed. The code includes imports for `math`, `os`, `random`, `re`, and `sys`. It also shows a main block that opens an output file and prints the result.
- Buttons:** At the bottom, there are buttons for "Run Code" and "Submit Code", along with options to "Upload Code as File" and "Test against custom input".

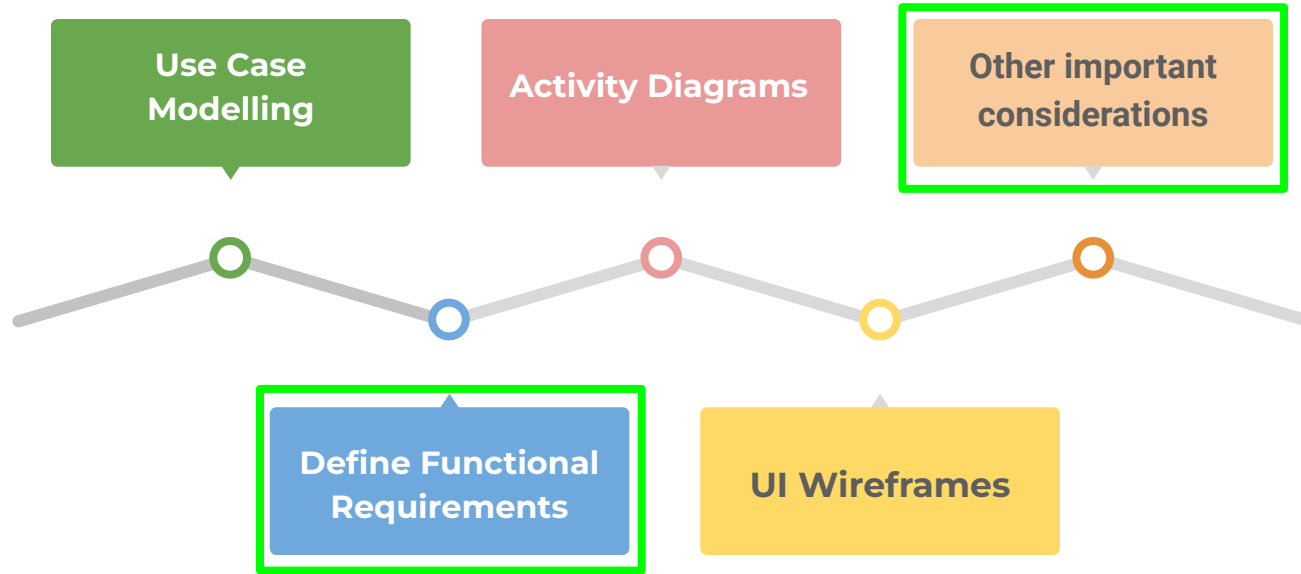
- Integrated code editor

# 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

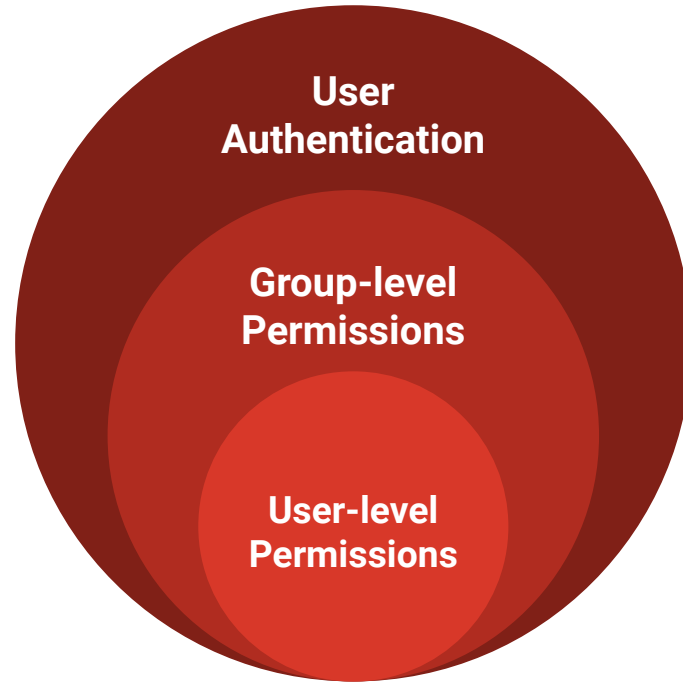
- Easy to learn
- Well-documented
- Large community

## Don't reinvent the wheel

- Integrate good existing projects if possible

# Access Control

3 layered approach





04

# Implementation

**Technologies used**

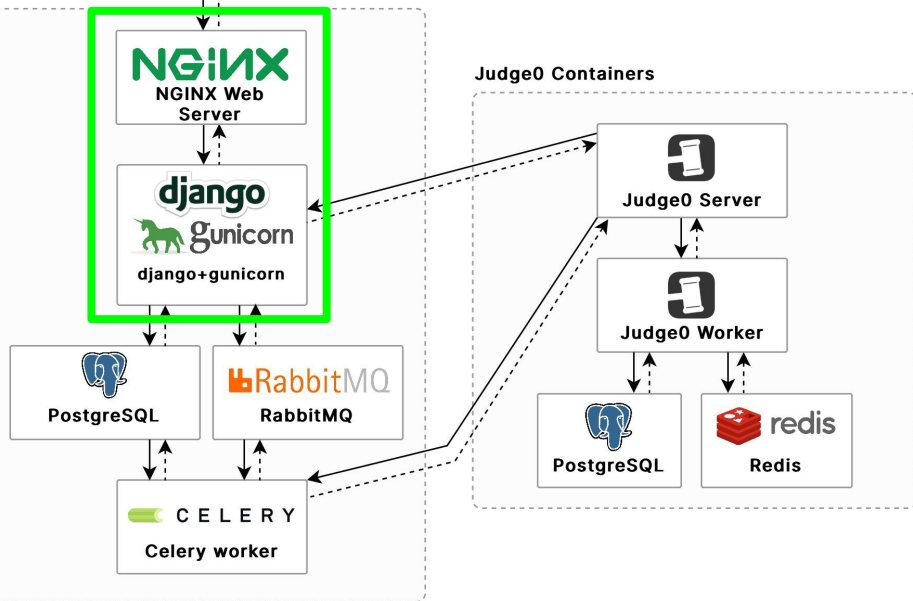
## FRONTEND



## BACKEND



AASP Containers



# Backend Server

## Django Web Framework

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

## FRONTEND



Web Browsers



HTML, CSS, JavaScript



Libraries



Font Awesome

## BACKEND



AASP Containers

**NGINX**

NGINX Web Server

**django**

Gunicorn

django+gunicorn



PostgreSQL

**RabbitMQ**

RabbitMQ

**CELERY**

Celery worker

Judge0 Containers



Judge0 Server



Judge0 Worker



PostgreSQL



Redis

# Code Execution Engine

## Judge0 API

**Executes code securely in a sandboxed environment**

- Open-source
- Actively maintained
- Well documented

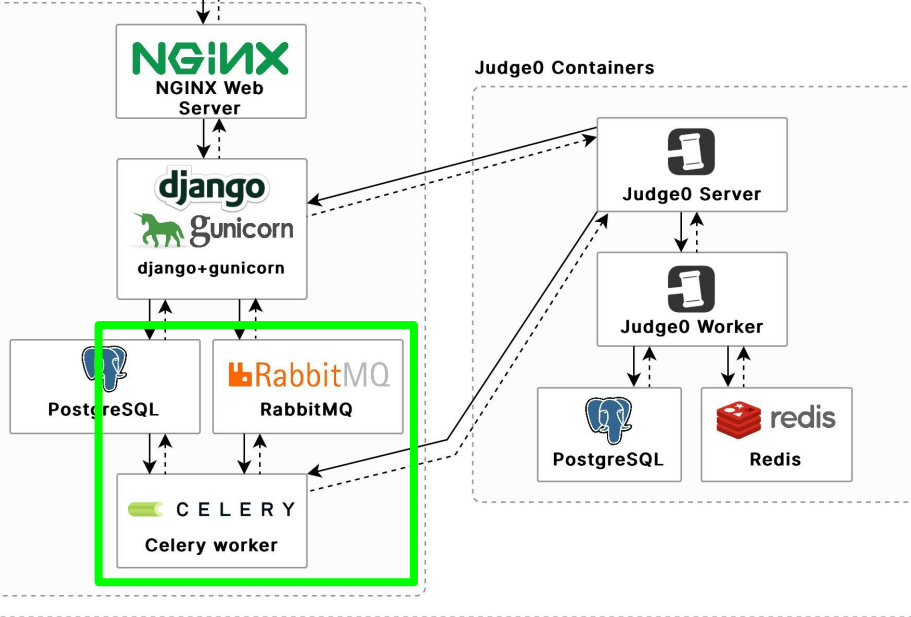
## FRONTEND



## BACKEND



AASP Containers



# 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

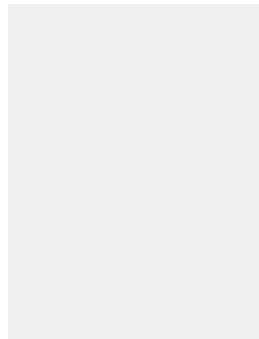
1. Clone the repository
2. Update configuration file to set secret keys
3. Run `docker-compose up -d`





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?

**CREDITS:** This presentation template was created by Slidesgo, including icons by **Flaticon**, and infographics & images by **Freepik**