

# ECLIPSE

ONLINE JUDGE

---

## TEAM ECLIPSE

Vighnesh Reddy Konda | Naman Jain | Satti Vamsi Krishna Reddy

# What's an OJ (Online Judge)?

- An online judge is an online system to test programs in programming contests. They are also used to practice for such contests.
- The system can compile and execute code, and test them with pre-constructed expected output data.
- Submitted code may be run with restrictions, including time limit, memory limit, security restriction and so on.
- The output of the code will be captured by the system, and compared with the standard output. The system will then return the result.
- Online Judges have rank lists based on user ratings

# ABSTRACT

---

# ABSTRACT

- Create a website for hosting programming contests and maintaining a database of problems in a categorized fashion.
- Login for existing users / Registration facility for new users and storing their details in a database.
- Implement a judge/checker that supports C, C++, Python, Java.
- Implement a Discussion Forum where users can discuss and post regarding various questions and their solutions.
- Implement a proper admin portal, where questions can be uploaded and contests hosted.

# ABSTRACT

- Implement a rating feature, rating will get updated after every contest, a statistical approach to calculate ratings appropriately.
- Extend the rating feature to leaderboards ranked according to ratings and categorised in various ways (according to say institution).
- Prevent execution of potential 'malware/untrusted' code that can be uploaded by a user intentionally.
- Implement a private messaging interface which also supports file-transfers for users to interact and collaborate.\*

*\*To be done if time permits*

# PRIOR ARTWORK & SURVEY

1. Popular OJs out there!
  2. Security (★)
  3. Why Django?
  4. Ranking System
  5. Django Channels\*
-

# POPULAR OJs

Some popular OJ's are SPOJ, Codeforces, Codechef, HackerRank, Hackerearth *etc.*

## CODEFORCES

- Users are divided into two major divisions based on their rating with each div. having its own difficulty.
- Virtual participation in a contest.
- Hacking someone else's solution who is in the same room during a contest.

## SPOJ

- Huge collection of problems.
- Problems are well categorized based on the tags given to them.
- Supports upto 5 spoken languages.  
(world-wide community)

# Codeforces - Messaging System

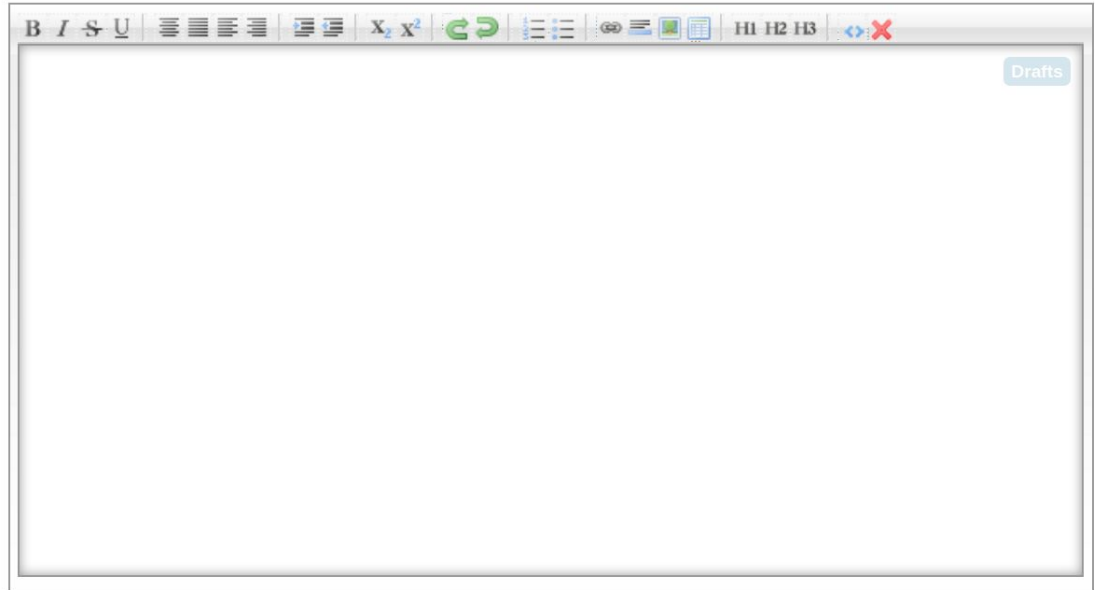
- Not so comfortable and appealing visually.
- Tough to collaborate in real-time with such an UI.

Message to



anonymousbuggy

anonymousbuggy: Yo chaman! :P  
naman\_ntc: round XD





# SECURITY

We intend to prevent any harm to server caused by execution of ‘untrusted code’. This is commonly called ‘Sandboxing’ in Computer Security. So, what we stumbled across as some possible options for this:

- Chroot jail - Isolates a particular directory - mimics itself as the root.
- LXB Containers (Linux Containers)
- Docker.io (an interesting open-source extension to Linux Containers, easier to handle) - something awesome! - but we felt it's beyond our ‘scope’ :(
- Libsandbox (not OS specific - more related to code itself)
- Using VMWare / Virtual Box - even run an OS inside another? - you know what I mean.

# Why Django over Rails?

A popular alternative available to Django is Ruby-on-Rails. Despite that, we choose Django because:

- Ruby-on-Rails is based on Ruby while Django on Python. We are relatively more familiar with python, so Django became a preference.
- Django takes care of user authentication system (*i.e.* `django.contrib.auth`), and their security — right out of the box unlike Rails.

# Why Django?

Other reasons to choose django were:

- It's Open Source (so is Rails).
- Databases are simple to be managed on frameworks like Django.
- We felt Django's Model-Template-View framework system and it being python based makes it a lot simple to code.
- Also, django has additional extensions like Channels which can be used for developing messaging systems.

And finally because what its creators call it:

'the web framework of perfectionists with **deadlines**'

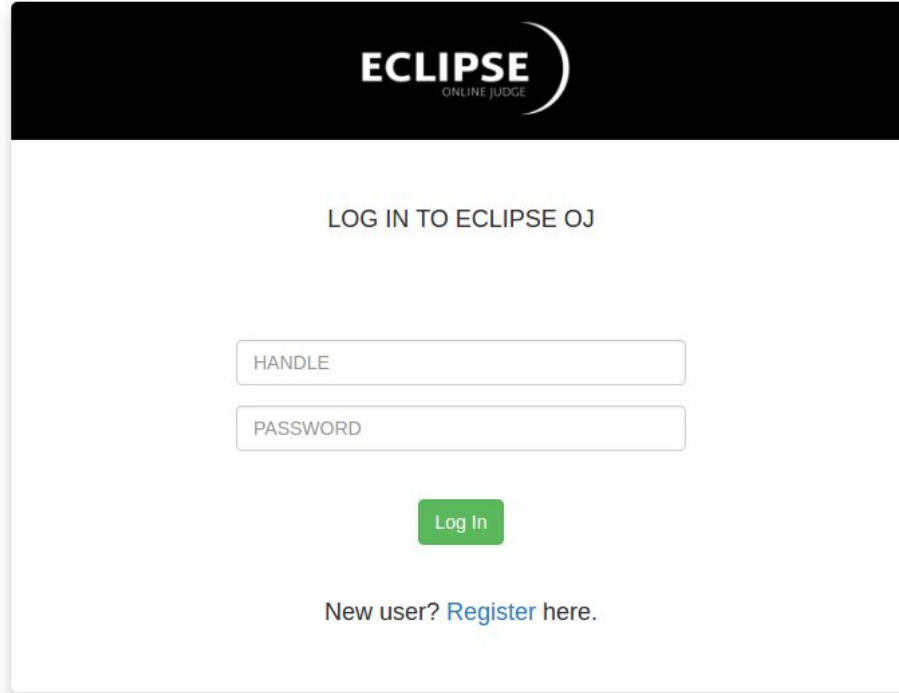
The Django logo, featuring the word "django" in a white, lowercase, sans-serif font on a dark green background.

The web framework for  
perfectionists with deadlines.

# Why Django? - Security of Users

- Django by-default uses ‘hashing’ to manage passwords of users securely.
- The choice of hashing algorithm is left to the user.
- Many of which popular ones like md5, bcrypt, pbkdf2, argon2, sha1 etc. already implemented in-the-box at ‘*django.contrib.auth.hashers*’.

# A SNEAK-PEEK



The image shows a login interface for 'ECLIPSE ONLINE JUDGE'. It features a black header with the logo, a white login form area, and a green 'Log In' button. The form includes input fields for 'HANDLE' and 'PASSWORD', and a link to 'Register here' for new users.

**ECLIPSE**  
ONLINE JUDGE

LOG IN TO ECLIPSE OJ

HANDLE

PASSWORD

Log In

New user? [Register here.](#)

# RATING SYSTEM

We want to put a Leaderboard such that it keeps people motivated in order to improve. For this we need a rating system, some of the popular ones include Elo, Chess, Glicko rating system ...etc.

- The Elo rating system is a method for calculating the relative skill levels of players in multiplayer type of games.
- We found that Elo rating system is most appropriate for us and also Codechef and Hackerrank implement their rating system mainly based on Elo (though not entirely the same)

Most of the popular OJs do not keep their rating mechanisms open and the number of factors the rating depends on is huge

# Django Channels

In order to create a modern chat application using Django we found that there are two approaches one using jQuery and the other using Websockets. Django channels is to make Django able to handle more than just plain HTTP requests, including WebSockets and HTTP2.

We chose websockets, because we didn't want our website to use jQuery for messaging which makes tons of useless state-check requests to the server. Also, websockets work instantly—opposing to jQuery approach, which can only pull data every N milliseconds.

# WHY ECLIPSE OJ?

---



# Why Eclipse OJ?

- Most of the popular online judges today ask for a price to hold a contest.
- No big and famous ‘open-source OJs’ - so felt the need of one :P
- Additionally IIT Bombay doesn't have an online judge of it's own, so we decided to make a state-of-the-art Online judge for our institute, which we strongly believe will increase the community of sport programmers in our insti.
- Team collaboration has been difficult, specifically for learners initially. Messaging systems (if any) already seem to be naive and not user-friendly.

# (OPEN-SOURCE) SOFTWARE REQUIREMENTS

---

# SOFTWARE REQUIREMENTS

## FOR PROJECT DEVELOPMENT

- Django (♥) - Python based
- HTML5
- CSS3 (using Sass)
- Javascript (including jQuery, \*React)
- BASH
- Others like: MathJax, Bootstrap ... etc.

## FOR PROJECT MANAGEMENT

- Git (GitHub repository)
- LaTeX (Documentation)

*\*To be used for messaging if time permits*

# IMPLEMENTATION

---

# IMPLEMENTATION

- All the frontend UI for our website is done using HTML, CSS, JS , Bootstrap.
- All the various modules like login, registration, discussion forum, database of questions, leaderboard, messaging ...etc are done as 'separate apps' thus accounting for reusability.
- We would implement a basic login/registration page with Django, using Django's authentication packages at '*django.contrib.auth*'
- We plan to use Django's argon2 hashing algorithm to provide secure hashing for passwords present at (*django.contrib.auth.hashers*)

# IMPLEMENTATION

- Customize the '*admin.py*' in Django specifically for adding problems and hosting contests in a simple way using sqlite3 databases.
- Every code is sandboxed using chroot jail to avoid any harm done by running untrusted/malicious codes.
- Implement a Rating system that reflects the user's performance based on a simplified Elo Rating System.

All user start with 1500 rating. The probability of a user with a rating  $R_A$  being placed higher than a person with a rating  $R_B$  is equal to:

$$E_A = \frac{1}{1 + 10^{(R_B - R_A)/400}}.$$

(Image Credits: [http://en.wikipedia.org/wiki/Elo\\_rating\\_system#Mathematical\\_details](http://en.wikipedia.org/wiki/Elo_rating_system#Mathematical_details))

# FEASIBILITY & LIMITATIONS

---

# FEASIBILITY & LIMITATIONS

- Due to time constraints we might not be able to add additional security features like docker for sandboxing our code.
- Chroot is known to be bypassed using relatively simple hacks.
- On reading about these features we realized implementing Advanced Security might be overambitious in the given time, so they are not in proposal draft.
- Also other online judges have been developed over such a long period have some nice features like virtual contest and hacking that we discussed above, which we don't aim to because of time constraints.



# FEASIBILITY & LIMITATIONS

- Many real-world OJ's use Rating Systems much more complicated than the one we intend to implement.
- We are also not implementing auto-construction of random test-cases for some given constraints in a problem statement. (something like codeforces-polygon)
- Our OJ doesn't check for plagiarism and thereby identify similar codes.
- Rest given, the project seems feasible. (especially because of django)

# TESTING & DEMONSTRATION

---

# TESTING

- We will perform rounds of testing after each iterative feature addition process.
- For the initial testing, we will have basic submissions and organize a small contest for our friends who we know will be eager to participate :D
- Later on, post the addition of other features like ratings, ranking, messaging interface\* and other constructs we'll perform same steps to verify their working.
- Specifically, we ensure that our project develops iteratively and hence, features are added one-by-one to the base project, similar to the way many open-source projects do. Thus having a working-model at every major commit.

*\*To be done if time permits*

# DEMONSTRATION

- Since mars server doesn't seem to support Django based websites we plan to demonstrate our project in a LAN
- Demonstrate the UI, create users and show that admin can add questions and respective test-cases, outputs. And also host a contest.
- Demonstrate the contest in the point of view of users and demonstrate our rating system.
- Verify and check whether our judge is evaluating properly.
- Demonstrate our discussion forum by adding posts from (say) 2 users.
- Show the leaderboard and it's categorization based on institute, region ...etc

# TIMELINE & WORK ALLOTMENT

---

Getting to understand Django - implementing a basic login page and it's UI

Iterate the procedure by adding leaderboard, developing the rating system.

Debugging the code, formatting and the documentation and making videos



Developing a basic online judge along with database and minimal features and security.

Customizing the admin interface, adding the discussion forum. Also, more functionalities like messaging if time permits

# WORK ALLOTMENT

## 19 - 31 September

- Naman: Works mainly on the front end such that the User Interface looks decent enough
- Vighnesh: Helps Naman in making UI and Vamsi in backend
- Vamsi: Works mainly on back end. Builds a small database of problems. Implements sandboxing with chroot jail

## 1 - 7 October

- Naman: Works on the leaderboard which can be searched by category
- Vighnesh: Works on the implementation of Elo rating system
- Vamsi: Helps Naman in leaderboard, Vighnesh in Elo rating system and improves the database of problems

## 7 - 14 October

- Naman: Helps Vamsi on admin portal and Vighnesh on discussion forum
- Vighnesh: Works on the discussion forum with comments facility.
- Vamsi: Works on customising the admin portal.

THANK YOU!