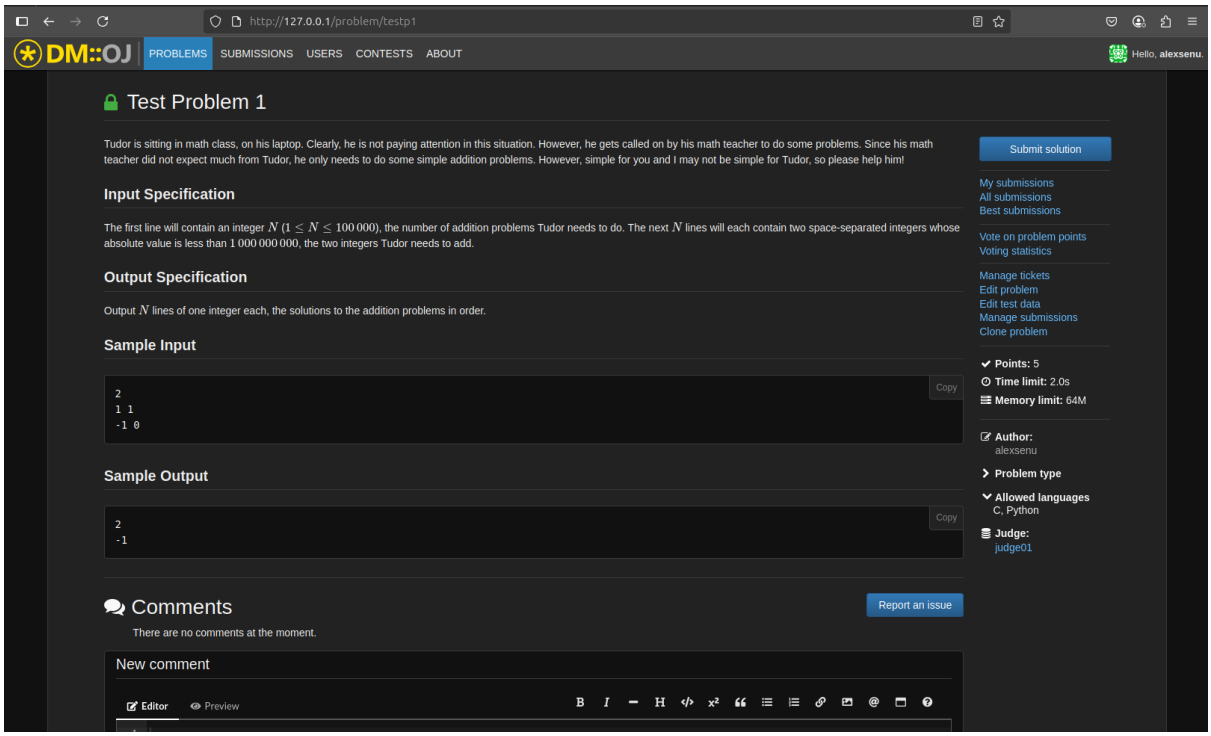
ΠΑΝΕΠΙΣΤΗΜΙΟ ΙΩΑΝΝΙΝΩΝ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ & ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΕΓΚΑΤΑΣΤΑΣΗ ΚΑΙ ΛΕΙΤΟΥΡΓΙΑ ΤΟΥ DMOJ, ΕΝΟΣ ΣΥΣΤΗΜΑΤΟΣ ΑΥΤΟΜΑΤΗΣ ΒΑΘΜΟΛΟΓΗΣΗΣ ΠΡΟΓΡΑΜΜΑΤΙΣΤΙΚΩΝ ΕΡΓΑΣΙΩΝ



Αλέξανδρος Σένου ΑΜ: 1127/15019

Επιβλέπων καθηγητής : Γκόγκος Χρήστος

Άρτα 2025

Πίνακας περιεχομένων

ΔΗΛΩΣΗ ΠΝΕΥΜΑΤΙΚΗΣ ΙΔΙΟΚΤΗΣΙΑΣ

Η παρούσα εργασία αποτελεί προϊόν αποκλειστικά δικής μου προσπάθειας. Όλες οι πηγές που χρησιμοποιήθηκαν περιλαμβάνονται στη βιβλιογραφία και γίνεται ρητή αναφορά σε αυτές μέσα στο κείμενο όπου έχουν χρησιμοποιηθεί.

ΠΕΡΙΛΗΨΗ

Στην εργασία αυτή μελετάται πώς εγκαθίσταται και λειτουργεί το σύστημα DMOJ, ένα σύγχρονο, ελεύθερο λογισμικό που αξιοποιείται για την αυτόματη αξιολόγηση ασκήσεων προγραμματισμού. Η εργασία ξεκινά με τη θεωρητική ανασκόπηση της έννοιας των online judge συστημάτων, παρουσιάζοντας αντίστοιχες υπάρχουσες πλατφόρμες και τα βασικά χαρακτηριστικά τους. Στη συνέχεια, αναλύεται η αρχιτεκτονική του DMOJ, η τεχνολογική του στοίβα, και οι διαδικασίες παραμετροποίησης σε περιβάλλον Ubuntu/WSL. Η εργασία εστιάζει σε πρακτικές πτυχές όπως η εγκατάσταση και η σύνδεση τεχνολογιών όπως Django, MariaDB, Redis, Celery, Node.js και Docker, με στόχο την πλήρη λειτουργικότητα του συστήματος αξιολόγησης. Τέλος, εξετάζονται πιθανές εφαρμογές του DMOJ σε ακαδημαϊκό πλαίσιο, για τη διευκόλυνση της διδασκαλίας και της αξιολόγησης φοιτητών, καθώς και προτεινόμενες βελτιώσεις και μελλοντικές κατευθύνσεις επέκτασης της πλατφόρμας.

ABSTRACT

This thesis focuses on the installation, operation, and practical use of DMOJ (Don Mills Online Judge), a modern open-source system for the automatic evaluation of programming assignments. This thesis starts with a theoretical overview of online judge systems, featuring a comparison of existing platforms and their core features. It then explores the system architecture of DMOJ, its technological stack, and the installation process within an Ubuntu/WSL environment. Emphasis is placed on the embodiment of the key components (Django, MariaDB, Redis, Celery, Node.js, Docker) to build a fully functional and secure evaluation platform. Finally, the thesis discusses academic use, cases of DMOJ for programming courses and student assessment, along with potential improvements and future directions for expansion.

1. Εισαγωγή

1.1 Υπόβαθρο και Κίνητρα

Στη σύγχρονη εκπαίδευση Πληροφορικής και στους διαγωνισμούς προγραμματισμού, η αξιολόγηση των προγραμματιστικών λύσεων, αποτελεί μια απαιτητική διαδικασία. Η παραδοσιακή χειροκίνητη βαθμολόγηση προγραμματιστικών εργασιών, είναι χρονοβόρα και επιρρεπής σε σφάλματα ή ασυνέπειες, ειδικά όταν ο αριθμός των φοιτητών ή συμμετεχόντων αυξάνεται. Τα συστήματα αυτόματης αξιολόγησης γνωστά και ως “online judges” (αυτόματοι βαθμολογητές), αναπτύχθηκαν για να αντιμετωπίσουν αυτές τις προκλήσεις. Τα συστήματα online judge αποτελούν πλατφόρμες που ελέγχουν αυτόματα τον κώδικα ενός προγράμματος, συγκρίνοντας τα αποτελέσματα που προκύπτουν με τα προκαθορισμένα σωστά αποτελέσματα για συγκεκριμένες εισόδους. Η υποβολή του κώδικα γίνεται συνήθως μέσω διαδικτύου, για αυτό και ο όρος online, ενώ μπορούν να τεθούν περιορισμοί σε πόρους όπως ο χρόνος εκτέλεσης και η μνήμη που χρησιμοποιεί το πρόγραμμα. Με αυτόν τον τρόπο εξασφαλίζεται, ότι η αξιολόγηση κάθε λύσης βασίζεται σε αντικειμενικά κριτήρια και προδιαγεγραμμένα tests cases, χωρίς ανθρώπινη μεροληψία.

Στο ακαδημαϊκό περιβάλλον, τα συστήματα αυτόματης βαθμολόγησης, έχουν γίνει απαραίτητα εργαλεία, καθώς προσφέρουν άμεση και συνεπή αξιολόγηση των εργασιών προγραμματισμού. Οι εκπαιδευτικοί, μπορούν να διαχειριστούν ευκολότερα μεγάλο όγκο υποβολών, ξοδεύοντας λιγότερο χρόνο στη διόρθωση και περισσότερο χρόνο στην υποστήριξη των φοιτητών[1]. Παράλληλα, οι φοιτητές, λαμβάνουν ταχύτατα απαντήσεις για τις λύσεις τους, γεγονός που ενισχύει την μαθησιακή διαδικασία: έχουν τη δυνατότητα να διορθώσουν λάθη καθώς και να βελτιώσουν τον κώδικά τους σε σύντομο χρόνο. Αυτό, είναι ιδιαίτερα σημαντικό στη σύγχρονη πληροφορική εκπαίδευση, όπου η έμφαση στην πρακτική εξάσκηση και στις δεξιότητες προγραμματισμού, απαιτεί συχνή αξιολόγηση με δίκαιο και αποδοτικό τρόπο.

Αντίστοιχα, στο ανταγωνιστικό περιβάλλον των προγραμματιστικών διαγωνισμών, τα συστήματα online judge αποτελούν τον ακρογωνιαίο λίθο για τη διεξαγωγή και την προετοιμασία των διαγωνισμάτων. Σε εθνικές και διεθνείς διοργανώσεις, όπου εκατοντάδες ή και χιλιάδες λύσεις υποβάλλονται μέσα σε περιορισμένο χρόνο, η αυτόματη αξιολόγηση, είναι η μόνη ρεαλιστική λύση για την ομαλή ροή τους. Ακόμα και σήμερα, η αξιολόγηση προγραμματιστικών εργασιών πραγματοποιείται σε πολλές περιπτώσεις χειροκίνητα, με αποτέλεσμα να υπάρχει σημαντική καθυστέρηση στην ανακοίνωση των αποτελεσμάτων. Η καθυστέρηση αυτή στερεί από τον φοιτητή τη δυνατότητα να λάβει άμεση και ουσιαστική ανατροφοδότηση, τη στιγμή που έχει ακόμη φρέσκες τις έννοιες και τους αλγορίθμους που εφάρμοσε. Καθώς μεσολαβούν άλλα μαθήματα διαφορετικού αντικειμένου (π.χ. αναλογικά ή ηλεκτρονικά κυκλώματα), η σύνδεση με το αρχικό προγραμματιστικό πρόβλημα χάνεται και η ανατροφοδότηση δεν έχει πλέον την ίδια αξία ή παιδαγωγική αποτελεσματικότητα. Τα σύγχρονα online judges επιτρέπουν την άμεση βαθμολόγηση κάθε υποβολής με προκαθορισμένα test cases, παρέχοντας στους συμμετέχοντες συνεχή ενημέρωση για την επίδοσή τους και διασφαλίζοντας παράλληλα ότι όλοι κρίνονται δίκαια με τα ίδια κριτήρια. Επιπλέον, χαρακτηριστικά όπως η μερική βαθμολόγηση (partial scoring) – όπου δίνεται τμηματική πίστωση για μερικώς σωστές λύσεις σε ένα πρόβλημα – και η ανάλυση περίπλοκων περιπτώσεων εισόδου επιτρέπουν μια πιο λεπτομερή αποτύπωση της επίδοσης των διαγωνιζομένων. Αυτό αντανακλάται και στις Ολυμπιάδες Πληροφορικής ή σε διαγωνισμούς τύπου ACM-ICPC, όπου οι

πλατφόρμες αυτόματης αξιολόγησης αποτελούν προαπαιτούμενο για την ομαλή διεξαγωγή των αγώνων.

Η ανάγκη για τέτοια συστήματα πηγάζει και από τις απαιτήσεις της σύγχρονης εκπαίδευσης στην Πληροφορική. Πέραν από τους παραδοσιακούς διαγωνισμούς, πολλά πανεπιστήμια εντάσσουν διαγωνιστικές δραστηριότητες ή ασκήσεις προγραμματισμού στο πρόγραμμα σπουδών τους, δημιουργώντας ένα ανταγωνιστικό αλλά εκπαιδευτικό πνεύμα μεταξύ των φοιτητών. Η ύπαρξη μιας πλατφόρμας όπου οι φοιτητές μπορούν να εξασκηθούν σε προβλήματα, να υποβάλλουν λύσεις και να λαμβάνουν άμεσα αποτελέσματα, ευθυγραμμίζεται με τις αρχές της ενεργητικής μάθησης και της άμεσης ανατροφοδότησης που θεωρούνται βέλτιστες πρακτικές στη σύγχρονη διδακτική. Δεν είναι τυχαίο ότι έχουν δημιουργηθεί πολυάριθμες πλατφόρμες αυτόματης αξιολόγησης, για να εξυπηρετήσουν αυτές τις ανάγκες[2]. Ενδεικτικά, συστήματα όπως το HUST Online Judge, το Vjudge, το Mooshak, το UVa Online Judge, το SPOJ (Sphere Online Judge) και πολλά άλλα έχουν χρησιμοποιηθεί ευρέως, προσφέροντας δυνατότητες αξιολόγησης προγραμματιστικών προβλημάτων σε πληθώρα γλωσσών. Κάθε σύστημα παρέχει διαφορετικά χαρακτηριστικά, όμως κοινός στόχος είναι η αξιόπιστη και γρήγορη βαθμολόγηση με αυτοματοποίηση.

Μια από τις πλέον σύγχρονες πλατφόρμες που έχουν κερδίσει έδαφος στην κοινότητα είναι το DMOJ (Don Mills Online Judge). Το DMOJ είναι ένα έργο ανοικτού κώδικα που σχεδιάστηκε με βάση τόσο τους διαγωνισμούς όσο και την εκπαιδευτική χρήση, συνδυάζοντας ποικίλες λειτουργίες που ικανοποιούν τις αυτές τις ανάγκες. Διαθέτει μια απλή και μοντέρνα διεπαφή χρήστη και συνοδεύεται από εκτενή τεκμηρίωση, στοιχεία που διευκολύνουν την υιοθέτησή του. Τεχνικά, υποστηρίζει μεγάλη ποικιλία γλωσσών προγραμματισμού και είναι αρκετά ώριμο και σταθερό ως λογισμικό. Σημαντικές λειτουργίες του DMOJ, όπως η μερική βαθμολόγηση, η μαζική αξιολόγηση πολλών υποβολών παράλληλα (batch evaluation), η δυνατότητα διεξαγωγής διαγωνισμών με προκαθορισμένα χρονικά παράθυρα και η ύπαρξη συστήματος κατάταξης και εικονικών διαγωνισμών, το καθιστούν ιδιαίτερα ελκυστικό για χρήση τόσο σε ανταγωνιστικά όσο και σε εκπαιδευτικά περιβάλλοντα. Επιπλέον, η ενεργή κοινότητα ανάπτυξης του DMOJ και η δυνατότητα επικοινωνίας με τους δημιουργούς του προσφέρουν ένα επίπεδο υποστήριξης που βοηθά στην επίλυση προβλημάτων ή στην προσθήκη νέων δυνατοτήτων.

Συνοψίζοντας, η αυτοματοποιημένη αξιολόγηση προγραμματισμού μπορεί να προσφέρει λύσεις σε θέματα που προκαλούνται κατά την διεξαγωγή διαγωνισμών Πληροφορικής, τόσο σε πανεπιστημιακά μαθήματα όσο και σε προγραμματιστικούς διαγωνισμούς. Το πλαίσιο αυτό δημιουργεί το υπόβαθρο και το κίνητρο για την παρούσα εργασία: η εξέταση ενός σύγχρονου συστήματος αυτόματης βαθμολόγησης – εν προκειμένω του DMOJ – και η διερεύνηση του τρόπου εγκατάστασης και λειτουργίας του, ώστε να αξιοποιηθεί ως εργαλείο στην εκπαιδευτική διαδικασία και πέραν αυτής.

1.2 Στόχοι της Εργασίας

Βασικός σκοπός της πτυχιακής αυτής είναι η μελέτη και πρακτική αξιολόγηση της πλατφόρμας DMOJ ως λύση για αυτόματη βαθμολόγηση προγραμματιστικών εργασιών. Οι κύριοι στόχοι που προσπαθεί να καλύψει η εργασία συνοψίζονται ως εξής:

- **Κατανόηση και παρουσίαση των αρχών λειτουργίας του DMOJ**: Ανάλυση σε βάθος στην αρχιτεκτονική και τον τρόπο λειτουργίας του συστήματος. Αυτό περιλαμβάνει την επεξήγηση των βασικών του υποσυστημάτων (π.χ. πυρήνας online judge, βάση δεδομένων, queue διαχείρισης εργασιών, διεπαφή χρήστη) και τον τρόπο με τον οποίο συνεργάζονται για να αξιολογούν αυτόματα τις υποβολές προγραμμάτων. Με την επίτευξη αυτού του στόχου, θα αποκτηθεί μια σφαιρική εικόνα του πώς το DMOJ χειρίζεται μια υποβολή από τη στιγμή που υποβάλλεται ο κώδικας μέχρι την παραγωγή του αποτελέσματος βαθμολόγησης.

- **Εγκατάσταση και παραμετροποίηση του συστήματος DMOJ**: Πρακτική υλοποίηση της πλατφόρμας σε τοπικό περιβάλλον. Ο στόχος αυτός περιλαμβάνει τη βήμα προς βήμα εγκατάσταση του DMOJ σε ένα σύστημα (σε περιβάλλον Ubuntu Linux), την ρύθμιση όλων των απαιτούμενων εξαρτήσεων (βάση δεδομένων, διακομιστής Redis, γλώσσες προγραμματισμού, κ.λπ.) και την προσαρμογή των ρυθμίσεων του DMOJ. Μέσα από αυτή τη διαδικασία, θα προβληθούν και οι τυχόν δυσκολίες ή ζητήματα συμβατότητας που μπορεί να προκύψουν, καθώς και οι καλύτερες πρακτικές για την αντιμετώπισή τους. Το αποτέλεσμα θα είναι ένας ολοκληρωμένος οδηγός εγκατάστασης και ρύθμισης, χρήσιμος για οποιονδήποτε επιθυμεί να αναπτύξει το σύστημα αυτό σε τοπικό περιβάλλον.

- **Αξιολόγηση των δυνατοτήτων και της απόδοσης του DMOJ**: Διερεύνηση και δοκιμή της λειτουργικότητας της πλατφόρμας στην πράξη. Αυτός ο στόχος περιλαμβάνει τη δημιουργία σεναρίων χρήσης, όπως η προσθήκη ενδεικτικών προβλημάτων στον κριτή, η υποβολή λύσεων από χρήστες και η παρακολούθηση της διαδικασίας βαθμολόγησης. Θα αξιολογηθούν οι δυνατότητες του DMOJ ως προς την υποστήριξη πολλαπλών γλωσσών, τον χειρισμό ταυτόχρονων υποβολών, την ακρίβεια και ταχύτητα βαθμολόγησης, καθώς και χαρακτηριστικά όπως το σύστημα βαθμολόγησης διαγωνισμών (π.χ. κατάταξη, μερική βαθμολόγηση). Παράλληλα, θα εξεταστεί η ευχρηστία της διεπαφής και η εμπειρία χρήστη. Μέσα από την αξιολόγηση αυτή αναμένεται να αναδειχθούν τα ισχυρά σημεία του συστήματος, αλλά και τυχόν περιορισμοί ή πεδία βελτίωσης.

Με την ολοκλήρωση των παραπάνω, η εργασία θα έχει καλύψει μια ολοκληρωμένη εικόνα του DMOJ – από τη θεωρητική περιγραφή και εγκατάστασή του, μέχρι τη δοκιμή του στην πράξη – προσφέροντας χρήσιμα συμπεράσματα για την αξία του ως εργαλείο αυτόματης αξιολόγησης προγραμματισμού.

1.3 Επισκόπηση της Δομής της Εργασίας

Η πτυχιακή εργασία οργανώνεται σε έξι κεφάλαια, τα οποία περιγράφονται συνοπτικά παρακάτω, με έμφαση στο περιεχόμενό τους:

Κεφάλαιο 2 – Θεωρητικό Υπόβαθρο και Σχετική Έρευνα: Παρουσιάζεται το γενικότερο θεωρητικό υπόβαθρο γύρω από τα συστήματα αυτόματης αξιολόγησης προγραμμάτων. Γίνεται αναφορά στην εξέλιξη των συστημάτων online judge διαχρονικά και σε παρόμοιες πλατφόρμες που έχουν

χρησιμοποιηθεί για εκπαιδευση, διαγωνισμούς αλλά και επαγγελματικούς λόγους(όπως η προετοιμασία για μια συνέντευξη). Επιπλέον, εξετάζονται οι βασικές τεχνολογίες και έννοιες που σχετίζονται με το αντικείμενο (όπως οι αρχές αξιολόγησης κώδικα, τα είδη test cases, ζητήματα ασφάλειας στην εκτέλεση υποβολών κ.ά.).

Κεφάλαιο 3 – Η Πλατφόρμα DMOJ: Αρχιτεκτονική και Λειτουργίες: Εστιάζει αποκλειστικά στο ίδιο το σύστημα DMOJ. Περιγράφεται η αρχιτεκτονική του σε βάθος, συμπεριλαμβανομένων των κύριων λειτουργικών μονάδων και υπηρεσιών του. Αναλύονται τα επιμέρους συστατικά, όπως η βάση δεδομένων και το σχήμα της, το web interface (frontend) και η επικοινωνία του με τον server (backend Django εφαρμογή), η ουρά εργασιών (task queue) και οι *workers* που εκτελούν τις αξιολογήσεις, καθώς και ο μηχανισμός του κεντρικού "κριτή"(judge) που διαχειρίζεται τις υποβολές. Επίσης, παρουσιάζεται ο τρόπος με τον οποίο το DMOJ υποστηρίζει τη βαθμολόγηση διαφορετικών γλωσσών προγραμματισμού και πώς επιτυγχάνει την ασφαλή εκτέλεση του κώδικα των χρηστών. Το κεφάλαιο αυτό προσφέρει μια ολοκληρωμένη εικόνα των εσωτερικών διεργασιών του DMOJ και θέτει τις βάσεις για την πρακτική υλοποίηση που ακολουθεί.

Κεφάλαιο 4 – Οδηγός Εγκατάστασης της Πλατφόρμας DMOJ: Αποτελεί ένα πρακτικό εγχειρίδιο για την εγκατάσταση και ρύθμιση του DMOJ σε ένα τοπικό σύστημα. Αρχικά, δίνονται οι προαπαιτούμενες τεχνικές απαιτήσεις (υλικό και λογισμικό) που πρέπει να διαθέτει το σύστημα φιλοξενίας, καθώς και όλες οι απαραίτητες εξαρτήσεις (βάση δεδομένων MariaDB, διακομιστής Redis, περιβάλλον Python/Django, Node.js για το frontend, Docker για τους κριτές, κ.ά.) Στη συνέχεια, περιγράφεται βήμα προς βήμα η διαδικασία εγκατάστασης: από την προετοιμασία του περιβάλλοντος (ενημέρωση συστήματος, εγκατάσταση απαραίτητων πακέτων), μέχρι τη λήψη του πηγαίου κώδικα του DMOJ, την δημιουργία και ρύθμιση της βάσης δεδομένων, την παραμετροποίηση των αρχείων ρυθμίσεων της πλατφόρμας και την εκκίνηση των υπηρεσιών της. Κάθε στάδιο συνοδεύεται από επεξηγήσεις και συμβουλές, ενώ επισημαίνονται πιθανά προβλήματα που μπορεί να προκύψουν (π.χ. ζητήματα δικαιωμάτων, ρυθμίσεις firewall) και οι τρόποι αντιμετώπισής τους. Το κεφάλαιο ολοκληρώνεται με την επιβεβαίωση της επιτυχούς λειτουργίας του συστήματος, καθοδηγώντας τον αναγνώστη στον έλεγχο ότι το DMOJ λειτουργεί σωστά μετά την εγκατάσταση.

Κεφάλαιο 5 – Περιπτώσεις Χρήσης και Υλοποίηση του DMOJ: Το κεφάλαιο αυτό εξετάζει πώς μπορεί να αξιοποιηθεί το DMOJ σε πραγματικές συνθήκες, κυρίως στο πλαίσιο της πανεπιστημιακής εκπαίδευσης. Παρουσιάζονται σενάρια χρήσης της πλατφόρμας σε μαθήματα όπως «Προγραμματισμός 1», «Αντικειμενοστραφής Προγραμματισμός» κ.ά., όπου η αυτόματη αξιολόγηση ενισχύει την εκπαιδευτική διαδικασία μέσω άμεσης ανατροφοδότησης και αντικειμενικής βαθμολόγησης. Αναλύεται η συμβολή του DMOJ στη διαχείριση και οργάνωση ασκήσεων, στη διεξαγωγή προγραμματιστικών διαγωνισμών (με δυνατότητα ρύθμισης χρόνου, βαθμολόγηση σε πραγματικό χρόνο, μερική αξιολόγηση), καθώς και στην ατομική εξάσκηση των φοιτητών. Ιδιαίτερη έμφαση δίνεται στην υποστήριξη πολλαπλών γλωσσών, στην ενίσχυση της αυτοεκπαίδευσης και στην αποτελεσματική παρακολούθηση της προόδου των συμμετεχόντων. Το κεφάλαιο αναδεικνύει τις δυνατότητες του DMOJ ως εργαλείου που ενισχύει τη μάθηση, προάγει τη διαφάνεια στην αξιολόγηση και υποστηρίζει διαδραστικά μοντέλα διδασκαλίας στον χώρο της πληροφορικής.

Κεφάλαιο 6 – Συμπεράσματα και Μελλοντικές Κατευθύνσεις: Το τελευταίο κεφάλαιο συνοψίζει τα κύρια ευρήματα και συμπεράσματα της εργασίας. Ανακεφαλαιώνεται ο βαθμός εκπλήρωσης των αρχικών στόχων και συζητείται πώς η εγκατάσταση και λειτουργία του DMOJ μπορεί να προσφέρει θετικά, στην εκπαιδευτική διαδικασία και τη διοργάνωση προγραμματιστικών διαγωνισμών. Γίνεται αναφορά στα σημαντικότερα πλεονεκτήματα που παρουσιάστηκαν από τη χρήση του συστήματος, όπως η αξιοπιστία, η ευκολία χρήσης και η επεκτασιμότητα. Παράλληλα, επισημαίνονται οι προκλήσεις που αντιμετωπίστηκαν – για παράδειγμα, οποιεσδήποτε τεχνικές δυσκολίες κατά την εγκατάσταση ή περιορισμοί του συστήματος που αναδείχθηκαν στις δοκιμές. Βάσει των παραπάνω, διατυπώνονται προτάσεις για μελλοντική εργασία και βελτιώσεις: αυτές μπορεί να περιλαμβάνουν πιθανές επεκτάσεις λειτουργικότητας του DMOJ, ενσωμάτωσή του σε ευρύτερα εκπαιδευτικά περιβάλλοντα (π.χ. συνδυασμός με συστήματα e-learning) ή τρόπους βελτίωσης της απόδοσης και της ασφάλειάς του. Το κεφάλαιο κλείνει με μια συνολική αποτίμηση της εμπειρίας και της συμβολής της παρούσας πτυχιακής, υπογραμμίζοντας τη σημασία της αυτόματης αξιολόγησης προγραμματισμού για τη συνεχή εξέλιξη της εκπαίδευσης στην πληροφορική.

Κεφάλαιο 2: Θεωρητικό υπόβαθρο και βιβλιογραφική ανασκόπηση

2.1 Επισκόπηση συστημάτων Online Judge

Τα συστήματα **Online Judge (OJ)** έχουν σχεδιαστεί για να αξιολογούν αυτόματα λύσεις προγραμματιστικών ασκήσεων μέσω του διαδικτύου. Αυτά τα συστήματα επιτρέπουν την αυτόματη αξιολόγηση κώδικα ή εκτελέσιμων αρχείων σε πραγματικό χρόνο, χωρίς την ανάγκη παρέμβασης βαθμολογητή[3]. Σε γενικές γραμμές, ένα σύστημα OJ, δέχεται ως είσοδο τον κώδικα που υποβάλλει ο χρήστης, τον μεταγλωττίζει (για γλώσσες που το απαιτούν) και τον εκτελεί σε ένα ομοιογενές περιβάλλον με προκαθορισμένα σύνολα δοκιμαστικών δεδομένων. Στη συνέχεια συγκρίνει την έξοδο του προγράμματος με τις αναμενόμενες απαντήσεις και παράγει ένα αποτέλεσμα ή βαθμολογία για τη λύση. Όλη αυτή η διαδικασία πραγματοποιείται **άμεσα**, επιτρέποντας στον προγραμματιστή να λάβει γρήγορη ανατροφοδότηση για την ορθότητα και την αποτελεσματικότητα της λύσης του.

Οι online judge πλατφόρμες εμφανίστηκαν αρχικά ως λύση για την αποτελεσματική υποστήριξη διαγωνισμών προγραμματισμού. Από τις πρώτες πλατφόρμες που ξεχώρισαν ήταν το UVa Online Judge, που δημιουργήθηκε το 1995 από το Ισπανικό Πανεπιστήμιο της Βαγιαδολίδ. Το UVa OJ συγκέντρωσε μια τεράστια συλλογή αλγοριθμικών προβλημάτων από τους διαγωνισμούς ACM και προσέλκυσε παγκοσμίως ενδιαφερόμενους λύτες. Με βάση αυτό το αρχείο προβλημάτων, εκδόθηκαν ακόμη και βιβλία που βοήθησαν φοιτητές να προετοιμαστούν για ομαδικούς προγραμματιστικούς διαγωνισμούς. Τα τελευταία χρόνια, οι διαγωνισμοί προγραμματισμού έχουν

εξελιχθεί σε μαζικές διοργανώσεις: ενδεικτικά, το 2015 συμμετείχαν πάνω από **40.000** φοιτητές από περίπου 3.000 πανεπιστήμια σε περιφερειακούς γύρους του ACM ICPC. Για την αποτελεσματική διεξαγωγή τέτοιων διοργανώσεων, η ύπαρξη αυτοματοποιημένων κριτών είναι καθοριστική – κανένα ανθρώπινο δυναμικό δεν θα μπορούσε να βαθμολογήσει σε πραγματικό χρόνο τόσο μεγάλο όγκο υποβολών λύσεων με την απαιτούμενη ταχύτητα και αξιοπιστία.

Εκτός από τους αγώνες ανταγωνιστικού προγραμματισμού, τα συστήματα OJ διαδραματίζουν σημαντικό ρόλο και στην **εκπαίδευση**. Έχουν πλέον καθιερωθεί ως βασικές πλατφόρμες για την εκμάθηση προγραμματισμού, δίνοντας στους μαθητευόμενους τη δυνατότητα να εξασκηθούν πρακτικά σε προγραμματιστικές ασκήσεις, να υποβάλουν τον κώδικά τους και να λάβουν άμεση ανάδραση στα αποτελέσματα[4]. Αυτή η άμεση ανατροφοδότηση λειτουργεί ευεργετικά στη μαθησιακή διαδικασία, επιτρέποντας στους φοιτητές να διορθώνουν τα λάθη τους και να βελτιώνουν τις δεξιότητές τους σε σύντομο χρόνο. Πράγματι, έρευνες έχουν δείξει ότι τα αυτοματοποιημένα συστήματα αξιολόγησης προσφέρουν αντικειμενικότητα και συνέπεια στη βαθμολόγηση, άμεση ενημέρωση του φοιτητή για την ορθότητα της λύσης, δυνατότητα εξάσκησης ανά πάσα στιγμή (24/7), καθώς και βελτίωση της ποιότητας του παραγόμενου κώδικα μέσω της συνεχούς ανατροφοδότησης[5]. Κατά συνέπεια, τα online judge συστήματα αποτελούν πλέον αναπόσπαστο κομμάτι τόσο των προγραμματιστικών διαγωνισμών όσο και των σύγχρονων μεθόδων διδασκαλίας του προγραμματισμού, καθιστώντας την διαδικασία αξιολόγησης πιο γρήγορη, αντικειμενική και προσβάσιμη σε μεγάλη κλίμακα.

2.2 Υπάρχουσες πλατφόρμες Online Judge

Με την εξάπλωση των προγραμματιστικών διαγωνισμών και την αυξανόμενη ανάγκη για αυτόματη αξιολόγηση, έχουν αναπτυχθεί πολλές πλατφόρμες τύπου online judge, καθεμία με τις δικές της ιδιαιτερότητες και κοινότητα χρηστών. Παρακάτω παρουσιάζονται συνοπτικά μερικές από τις πιο δημοφιλείς πλατφόρμες και τα χαρακτηριστικά τους:

- **Codeforces**: Μία από τις μεγαλύτερες διεθνείς κοινότητες ανταγωνιστικού προγραμματισμού, γνωστή για τους τακτικούς αγώνες που διοργανώνει σε εβδομαδιαία βάση. Το Codeforces εισήγαγε σύστημα κατάταξης με **rating** τύπου Elo, χωρίζοντας τους συμμετέχοντες σε δύο κύριες κατηγορίες (Div.1 και Div.2) ανάλογα με το επίπεδό τους. Κάθε αγώνας έχει διάρκεια συνήθως 2 ώρες και περιλαμβάνει πολλαπλά προβλήματα προς επίλυση. Ένα ιδιαίτερο χαρακτηριστικό της πλατφόρμας είναι η φάση των "**hack**": μετά το πέρας του διαγωνισμού, οι συμμετέχοντες έχουν τη δυνατότητα να δουν τις λύσεις των αντιπάλων τους και να δοκιμάσουν επιθετικά test cases για να εντοπίσουν αποτυχίες σε αυτές. Αν μια λύση ανταγωνιστή αποδειχθεί λανθασμένη υπό το νέο τεστ, ο «hacker» κερδίζει επιπλέον πόντους ενώ ο αντίπαλος χάνει βαθμούς. Αυτό το σενάριο κάνει τους αγώνες του Codeforces ιδιαίτερα δυναμικούς, προάγοντας την εύρεση οριακών περιπτώσεων και την περαιτέρω βελτίωση των λύσεων.

- **CodeChef**: Μια δημοφιλής πλατφόρμα που εστιάζει κυρίως στην πρακτική εξάσκηση και σε διαγωνισμούς προγραμματισμού διαφόρων επιπέδων δυσκολίας. Το CodeChef παρέχει στους χρήστες έναν φιλικό online κώδικα-συντάκτη (editor) και υποστηρίζει την εκτέλεση

κώδικα σε **πάνω από 40 γλώσσες προγραμματισμού** δίνοντας μεγάλη ευελιξία στους διαγωνιζόμενους να επιλέξουν τη γλώσσα που προτιμούν. Τα προβλήματα της πλατφόρμας είναι ταξινομημένα ανά βαθμό δυσκολίας, από εύκολα έως πολύ δύσκολα, έτσι ώστε οι χρήστες να μπορούν να προοδεύουν σταδιακά. Η πλατφόρμα διοργανώνει τακτικά διαγωνισμούς (monthly challenges, cook-offs κ.ά.) και περιέχει πίνακες κατάταξης που βοηθούν στην κατανόηση της σχετικής απόδοσης κάθε διαγωνιζόμενου.

- **HackerRank**: Σε αντίθεση με τις καθαρά ανταγωνιστικές πλατφόρμες, το HackerRank φτιάχτηκε με κύριο στόχο την προετοιμασία των χρηστών, για τη διαδικασία **πρόσληψης προγραμματιστών** από εταιρείες. Αποτελεί ένα εργαλείο στο οποίο οι εργοδότες μπορούν να δημιουργήσουν εξατομικευμένα tests προγραμματισμού και οι υποψήφιοι να τα λύσουν μέσω του φυλλομετρητή. Η πλατφόρμα προσφέρει μια μεγάλη γκάμα θεμάτων (αλγόριθμοι, δομές δεδομένων, ανάπτυξη λογισμικού, SQL, machine learning κλπ.) και άμεση βαθμολόγηση των υποβολών, γεγονός που την καθιστά ιδιαίτερα χρήσιμη για αξιολόγηση δεξιοτήτων σε συνεντεύξεις. Επιπλέον, το HackerRank διαθέτει πίνακες κατάταξης και διαγωνισμούς ανοικτούς προς όλους, λειτουργώντας και ως χώρος εξάσκησης για φοιτητές και επαγγελματίες, αν και η κύρια έμφασή του είναι στις **διαδικασίες πρόσληψης** και όχι στους ανοιχτούς προγραμματιστικούς αγώνες.

- **AtCoder**: Πρόκειται για μια πλατφόρμα που το έτος της δημιουργίας του ήταν το 2012 στην Ιαπωνία και έχει εξελιχθεί σε μία από τις πλέον αξιόπιστες πηγές προγραμματιστικών διαγωνισμών υψηλής ποιότητας. Οι διαγωνισμοί του AtCoder (συνήθως οι σειρές **ABC**, **ARC** και **AGC**) πραγματοποιούνται σε εβδομαδιαία ή μηνιαία βάση, προσελκύοντας τόσο αρχάριους όσο και έμπειρους διαγωνιζόμενους. Όλοι οι αγώνες προσφέρονται με δίγλωσσες εκφωνήσεις (αγγλικά και ιαπωνικά), διευκολύνοντας τη διεθνή συμμετοχή[6]. Το σύστημα επιτρέπει την χρήση πολλών διαφορετικών γλωσσών προγραμματισμού (ουσιαστικά οποιαδήποτε δημοφιλή γλώσσα είναι αποδεκτή για τις λύσεις) και ακολουθεί το κλασικό μοντέλο αξιολόγησης: οι υποβολές κώδικα ελέγχονται αυτόματα απέναντι σε κρυφά test cases, με στόχο την πλήρη κάλυψη όλων των σεναρίων. Η βαθμολόγηση είναι συνήθως τύπου **ACM-ICPC** (πλήρης επιτυχία ή αποτυχία ανά πρόβλημα), χωρίς τη δυνατότητα της μερικής βαθμολόγησης, ενώ δεν υπάρχει φάση 'hack' όπως στο Codeforces. Το AtCoder είναι ιδιαίτερα φημισμένο για τα καλοσχεδιασμένα προβλήματά του, πολλά εκ των οποίων εμβαθύνουν σε προχωρημένες αλγοριθμικές τεχνικές, και θεωρείται εξαιρετικό πεδίο εξάσκησης για όσους στοχεύουν σε διεθνείς διαγωνισμούς.

- **LeetCode**: Η LeetCode έχει καθιερωθεί ως η κορυφαία διαδικτυακή πλατφόρμα για την **προετοιμασία τεχνικών συνεντεύξεων εργασίας** και την εξάσκηση σε αλγοριθμικά προβλήματα. Προσφέρει μια τεράστια συλλογή από προβλήματα κάθε είδους (πάνω από 2500 αυτή τη στιγμή), τα οποία εστιάζουν κυρίως σε κλασικές προγραμματιστικές προκλήσεις που τίθενται σε συνεντεύξεις μεγάλων εταιρειών. Το LeetCode παρέχει ένα φιλικό περιβάλλον όπου ο χρήστης μπορεί να γράψει, να μεταγλωττίσει και να εκτελέσει τον κώδικά του απευθείας μέσω του ιστότοπου, λαμβάνοντας άμεσο αποτέλεσμα για το αν η λύση είναι σωστή ή όχι. Θεωρείται από πολλούς ως η καλύτερη πλατφόρμα για να βελτιώσει κανείς τις δεξιότητές του και να κάνει την απαραίτητη προετοιμασία για τεχνικές συνεντεύξεις. Παράλληλα, το LeetCode διοργανώνει εβδομαδιαίους διαγωνισμούς (weekly contests) και πίνακες κατάταξης, προσφέροντας και μια ανταγωνιστική διάσταση για όσους

το επιθυμούν. Ο κύριος σκοπός της όμως είναι η εκμάθηση μέσα από την πράξη και στην εξοικείωση με προβλήματα που συναντώνται συχνά στη βιομηχανία λογισμικού.

- **DMOJ (DMOJ: Modern Online Judge)**: Το DMOJ είναι ένα σύγχρονο σύστημα online judge που ξεχωρίζει για το λογισμικό **ανοιχτού κώδικα**. Η πλατφόρμα ξεκίνησε το 2013 από μια κοινότητα στον Καναδά και εμπνεύστηκε από προγενέστερα εγχειρήματα όπως το PEG Judge. Σήμερα, λειτουργεί τόσο ως ιστότοπος ανοικτών διαγωνισμών, όσο και ως αποθετήριο προβλημάτων. Το DMOJ φιλοξενεί προβλήματα από διάφορους εθνικούς διαγωνισμούς (π.χ. τον Καναδικό διαγωνισμό CCC, τον Κροατικό COCI, προβλήματα από Ολυμπιάδες Πληροφορικής IOI/JOI κ.ά.) και διοργανώνει τους δικούς του μηνιαίους ανοικτούς διαγωνισμούς (**DMOJ Monthly Open Programming Contest - DMOPC**) στους οποίους μπορεί να συμμετάσχει οποιοσδήποτε. Ως πλατφόρμα, υποστηρίζει επίσης πολλές γλώσσες προγραμματισμού και προηγμένα είδη προβλημάτων (συμπεριλαμβανομένων διαδραστικών προβλημάτων). Χάρη στο ότι είναι ανοικτού κώδικα, το DMOJ μπορεί να εγκατασταθεί και να τροποποιηθεί ελεύθερα από εκπαιδευτικούς ή οργανισμούς που επιθυμούν να δημιουργήσουν τον δικό τους τοπικό online judge server. Αυτό το χαρακτηριστικό το έχει κάνει δημοφιλές σε πανεπιστήμια και σχολεία που θέλουν να προσφέρουν στους μαθητές τους μια εντός έδρας εμπειρία online judge για εκπαιδευτικούς σκοπούς.

(Σημείωση: Εκτός από τις παραπάνω, υπάρχουν πολλές ακόμη πλατφόρμες online judge, όπως η κλασική TopCoder, που από το 2001 διοργανώνει σύντομους προγραμματιστικούς αγώνες με σύστημα κατάταξης, η Sphere Online Judge (SPOJ), με τεράστιο αρχείο προβλημάτων και διαφορετικές κατηγορίες προκλήσεων. Κάθε σύστημα έχει τη δική του κοινότητα και στόχους, αλλά οι βασικές αρχές λειτουργίας τους παραμένουν παρόμοιες.)

2.3 Βασικά χαρακτηριστικά σύγχρονων συστημάτων Online Judge

Παρά την ποικιλία υλοποιήσεων, τα σύγχρονα συστήματα online judge μοιράζονται ορισμένα χαρακτηριστικά που τα κάνουν αποτελεσματικά και δημοφιλή. Τα κυριότερα από αυτά συνοψίζονται ως εξής:

- **Υποστήριξη πολλαπλών γλωσσών προγραμματισμού**: Τα περισσότερα online judge συστήματα δίνουν τη δυνατότητα για υποβολές λύσεων σε πολλές διαφορετικές γλώσσες, ώστε να μην υπάρχει περιορισμός για τους διαγωνιζόμενους ή μαθητές σε μία μόνο τεχνολογία. Τυπικά, υποστηρίζονται όλες οι δημοφιλείς γλώσσες (C, C++, Java, Python, JavaScript, C#, Ruby, κ.ά.), ενώ ορισμένες πλατφόρμες περιλαμβάνουν ακόμη και πιο εξειδικευμένες ή εκπαιδευτικές γλώσσες. Για παράδειγμα, η πλατφόρμα CodeChef διαθέτει περιβάλλον εκτέλεσης για πάνω από **40 γλώσσες προγραμματισμού**, δίνοντας μεγάλη ευελιξία στους χρήστες. Η πολυγλωσσική υποστήριξη επεκτείνεται συχνά και στο περιβάλλον χρήστη (UI) – δηλαδή το ίδιο το site μπορεί να είναι διαθέσιμο σε περισσότερες από μία γλώσσες για να εξυπηρετεί διεθνές κοινό.

- **Αυτόματη εκτέλεση και βαθμολόγηση υποβολών**: Ο πυρήνας κάθε online judge αποτελείται από το να **εκτελεί αυτόματα** τον κώδικα που υποβάλλει κάθε χρήστης και να τον αξιολογεί

σύμφωνα με προκαθορισμένα κριτήρια. Αυτό μπορεί να επιτευχθεί μέσω ειδικών διεργασιών (κριτές) που φορτώνουν τον εκτελέσιμο κώδικα σε ένα προστατευμένο περιβάλλον (sandbox), το τρέχουν με διάφορα σύνολα εισόδων (test cases) και ελέγχουν την έξοδο. Αν η έξοδος συμφωνεί με την αναμενόμενη για κάθε δοκιμή, τότε η λύση θεωρείται επιτυχής. Σε διαφορετική περίπτωση, το σύστημα απορρίπτει την λύση, και καταγράφει το είδος του σφάλματος (π.χ. λάθος αποτέλεσμα, υπέρβαση χρόνου κλπ). Η διαδικασία αυτή είναι πλήρως αυτοματοποιημένη και ομοιόμορφη, εξασφαλίζοντας δίκαιη και αντικειμενική αξιολόγηση για όλους τους χρήστες. Επιπλέον, εξαλείφει την ανάγκη ανθρώπινης βαθμολόγησης, μειώνοντας σημαντικά τον χρόνο αλλά και τον κόπο που απαιτείται για την αξιολόγηση εκατοντάδων ή και χιλιάδων λύσεων σε μεγάλους διαγωνισμούς.

- **Άμεση ανατροφοδότηση (real-time feedback)**: Ένα από τα πιο σημαντικά πλεονεκτήματα των online judges είναι ότι προσφέρουν άμεσα αποτελέσματα στον χρήστη μετά την υποβολή μιας λύσης. Μέσα σε λίγα δευτερόλεπτα από την υποβολή του ο χρήστης ενημερώνεται αν η λύση του ήταν σωστή ή όχι, καθώς και για τον λόγο αποτυχίας σε περίπτωση λάθους. Τα συστήματα χρησιμοποιούν σύντομους **κωδικούς αποτελέσματος** για να δηλώσουν την κατάληξη μιας υποβολής: π.χ. Accepted (AC) για αποδεκτή σωστή λύση, Wrong Answer (WA) για λανθασμένη έξοδο, Time Limit Exceeded (TLE) όταν το πρόγραμμα υπερβαίνει το επιτρεπόμενο χρονικό όριο, Runtime Error (RE) για σφάλμα κατά την εκτέλεση, Memory Limit Exceeded (MLE) για υπέρβαση μνήμης, κ.ά.. Αυτή η άμεση ανατροφοδότηση σε πραγματικό χρόνο επιτρέπει στους προγραμματιστές να μάθουν από τα λάθη τους γρήγορα. Στο πλαίσιο των διαγωνισμών, δίνει τη δυνατότητα στους συμμετέχοντες να γνωρίζουν την πορεία τους (π.χ. αν ένα πρόβλημα λύθηκε επιτυχώς ή αν χρειάζεται διορθώσεις) και να μπορούν να διαχειρίζονται καλύτερα τον χρόνο τους. Στο πλαίσιο εκπαίδευσης, η συνεχής άμεση ανάδραση έχει αποδειχθεί ότι βελτιώνει την αποτελεσματικά την εκμάθηση προγραμματισμου, καθώς οι φοιτητές μπορούν να αντιλαμβάνονται τα λάθη τους και να τα διορθώνουν αμέσως.

- **Φιλική διεπαφή χρήστη**: Οι σύγχρονες πλατφόρμες online judge διαθέτουν προσεγμένα σχεδιασμένες διεπαφές, που διευκολύνουν τον χρήστη στην πλοήγηση και τη χρήση. Συνήθως παρέχουν μια βιβλιοθήκη προβλημάτων, με δυνατότητες επιλογής φίλτρων (ανά δυσκολία, κατηγορία αλγορίθμου, κ.λπ.), προφίλ χρήστη με στατιστικά και επιτεύγματα, καθώς και πίνακες κατάταξης (leaderboards) για διαγωνισμούς. Πολλές πλατφόρμες προσφέρουν ενσωματωμένο **editor κώδικα στο φυλλομετρητή** έτσι ώστε ο χρήστης να μπορεί γράψει και να δοκιμάσει τη λύση του εκείνη τη στιγμή, χωρίς να χρειάζεται εξωτερικά εργαλεία. Για παράδειγμα, σε πλατφόρμες όπως το HackerRank και το CodeChef, ο διαγωνιζόμενος μπορεί πριν κάνει την τελική υποβολή, να επεξεργαστεί τον κώδικα σε ένα online περιβάλλον και να εκτελέσει δοκιμαστικά το πρόγραμμά του με είσοδο της επιλογής του. Επιπλέον, το περιβάλλον χρήστη παρέχει σαφείς επεξηγήσεις για τα μηνύματα σφαλμάτων ή τους κωδικούς αποτελέσματος (status codes) που λαμβάνει ο χρήστης, βοηθώντας τον στο να κατανοήσει τι πήγε στραβά και πώς μπορεί να το διορθώσει. Συνολικά, η προσοχή που δίνεται στην απλότητα και την ευχρηστία καθιστά αυτά τα συστήματα προσβάσιμα ακόμη και σε αρχάριους, επιτρέποντάς τους να επικεντρωθούν στην επίλυση προβλημάτων αντί να παλεύουν με περίπλοκα εργαλεία.

- **Επεκτασιμότητα και απόδοση**: Τα online judge συστήματα έχουν φτιαχτεί με τρόπο που να τους επιτρέπει να μπορούν να εξυπηρετούν μεγάλο αριθμό χρηστών και υποβολών ταυτόχρονα, κάτι που είναι κρίσιμο σε περιόδους αυξημένου φόρτου, όπως κατά τη διάρκεια διαγωνισμών. Η αρχιτεκτονική τους βασίζεται σε κατανεμημένα υποσυστήματα, όπου ένας κεντρικός web server παραλαμβάνει τις υποβολές και τις προωθεί μέσω **ουρών εργασιών** σε πολλούς αυτόνομους "κριτές" για ταυτόχρονη αξιολόγηση. Ένα παράδειγμα είναι η αρχιτεκτονική του DMOJ, η οποία περιλαμβάνει ξεχωριστά στοιχεία όπως web server, ουρά εργασιών και πολλούς **κριτές** (processes) που τρέχουν τις υποβολές, γεγονός που της επιτρέπει να χειρίζεται πολύ μεγάλο αριθμό υποβολών παρέχοντας ταυτόχρονα γρήγορη και αξιόπιστη βαθμολόγηση[7].Επιπλέον, αξιοποιούνται τεχνολογίες όπως το **cloud** και τα containers (Docker) ώστε το σύστημα να προσαρμόζεται αυτόματα στις απαιτήσεις κάθε στιγμής. Αυτό σημαίνει ότι ακόμη και όταν υποβάλλονται εκατοντάδες λύσεις σχεδόν ταυτόχρονα, η πλατφόρμα μπορεί να διατηρήσει γρήγορους χρόνους απάντησης. Η αξιοπιστία ενισχύεται περαιτέρω με περιορισμούς σε χρόνο και μνήμη, ώστε να προστατεύεται το σύστημα από ακραίες ή εσφαλμένες συμπεριφορές υποβληθέντος κώδικα.

- **Ανοικτός κώδικας και κοινότητα**: Ένα θετικό χαρακτηριστικό ορισμένων συστημάτων online judge είναι ότι διατίθενται με **άδεια ανοικτού κώδικα**. Αυτό επιτρέπει σε ακαδημαϊκά ιδρύματα ή ομάδες χρηστών να εγκαταστήσουν και να παραμετροποιήσουν δικό τους server αξιολόγησης σύμφωνα με τις ανάγκες τους. Το DMOJ, για παράδειγμα, διατίθεται δημόσια και μπορεί να μελετηθεί ή να τροποποιηθεί από οποιονδήποτε. Παρόμοια, υπάρχουν και άλλα έργα όπως το DOMjudge, το Mooshak και το HUST Online Judge, τα οποία υποστηρίζουν διαγωνισμούς και προσφέρουν τεχνική διαφάνεια. Ο ανοικτός κώδικας ενισχύει την εμπιστοσύνη στον τρόπο λειτουργίας των συστημάτων και επιτρέπει σε τεχνικές κοινότητες να βελτιώνουν τις δυνατότητες του λογισμικού, να επιδιορθώνουν σφάλματα ή να προσθέτουν νέες λειτουργίες, όπως μηχανισμούς εντοπισμού λογοκλοπής ή εξειδικευμένους τύπους προβλημάτων. Η δυνατότητα χρήσης χωρίς κόστος αδειοδότησης ενισχύει την εκπαιδευτική χρήση, ειδικά από ιδρύματα με περιορισμένους πόρους.

Συμπερασματικά, τα σύγχρονα online judge συνδυάζουν όλα τα παραπάνω χαρακτηριστικά για να μπορούν να προσφέρουν, ένα ολοκληρωμένο περιβάλλον αυτόματης αξιολόγησης. Υποστηρίζοντας πολλαπλές γλώσσες και χιλιάδες χρήστες, με γρήγορη και αξιόπιστη βαθμολόγηση σε πραγματικό χρόνο, αποτελούν αναπόσπαστο εργαλείο τόσο για την κοινότητα των διαγωνισμών προγραμματισμού όσο και για εκπαιδευτικούς σκοπούς. Μέσω αυτών των δυνατοτήτων, η διαδικασία επίλυσης αλγοριθμικών προβλημάτων γίνεται πιο προσιτή, διαδραστική και **συνεργατική**, δημιουργώντας νέες ευκαιρίες μάθησης και εξέλιξης για προγραμματιστές όλων των επιπέδων.

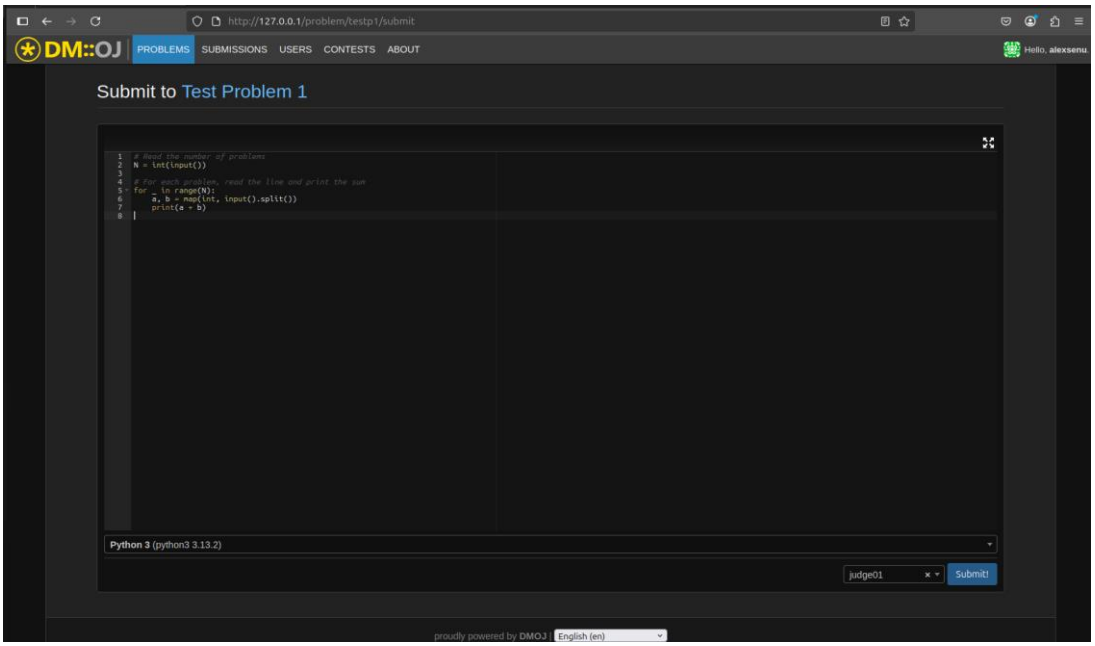
Κεφάλαιο 3: Αρχιτεκτονική συστήματος DMOJ

3.1 Επισκόπηση συστήματος

Η πλατφόρμα DMOJ (Don Mills online judge) έχει σχεδιαστεί για την αυτόματη αξιολόγηση προγραμματιστικών λύσεων, αποτελούμενη από πολλά υποσυστήματα που συνεργάζονται μεταξύ τους. Η αρχιτεκτονική του, έχει σχεδιαστεί να διαχειρίζεται μεγάλο πλήθος υποβολών ταυτόχρονα. Αυτό το καταφέρνει δημιουργώντας πρώτα μια εικόνα δικαστή (βαθμολογητή) στο Docker, με τη βοήθεια ενός shell script (εντολής κελύφους που μπορεί να χρησιμοποιηθεί μόνο σε Linux/Unix shell), και μέσω αυτής της εικόνας, μπορεί να τρέξει πολλά containers παράλληλα που θα περιέχουν από έναν δικαστή, έτοιμα να παραλάβουν τους υποβληθέντες κωδικούς και να τους αξιολογήσουν. Σε υψηλό επίπεδο, η πλατφόρμα περιλαμβάνει μια διαδικτυακή εφαρμογή (frontend και backend) *[Πίνακας 1]* για την παρουσίαση προβλημάτων και λήψη λύσεων, μία ουρά εργασιών για την προώθηση των υποβολών προς αξιολόγηση, αυτόνομα προγράμματα-δικαστές που εκτελούν και ελέγχουν τον κώδικα, καθώς και βάσεις δεδομένων και άλλες υπηρεσίες υποστήριξης. Όλα αυτά τα μέρη συνεργάζονται αρμονικά, επιτρέποντας στη ροή πληροφορίας να ξεκινά από την υποβολή μιας λύσης και να καταλήγει στην τελική βαθμολόγηση και ανατροφοδότηση προς τον χρήστη.

Ροή αξιολόγησης μιας υποβολής : από τη στιγμή που ένας χρήστης υποβάλει τη λύση του μέσω της διεπαφής web, ξεκινά μια ακολουθία βημάτων για την αυτόματη αξιολόγησή της.

1. **Καταχώρηση υποβολής :** η λύση αποστέλλεται στον διακομιστή web και καταγράφεται στη βάση δεδομένων ως νέα υποβολή προς αξιολόγηση. Ο χρήστης ενημερώνεται ότι η λύση του ελήφθη και βρίσκεται σε κατάσταση **«Pending»** (αναμονή προς βαθμολόγηση).

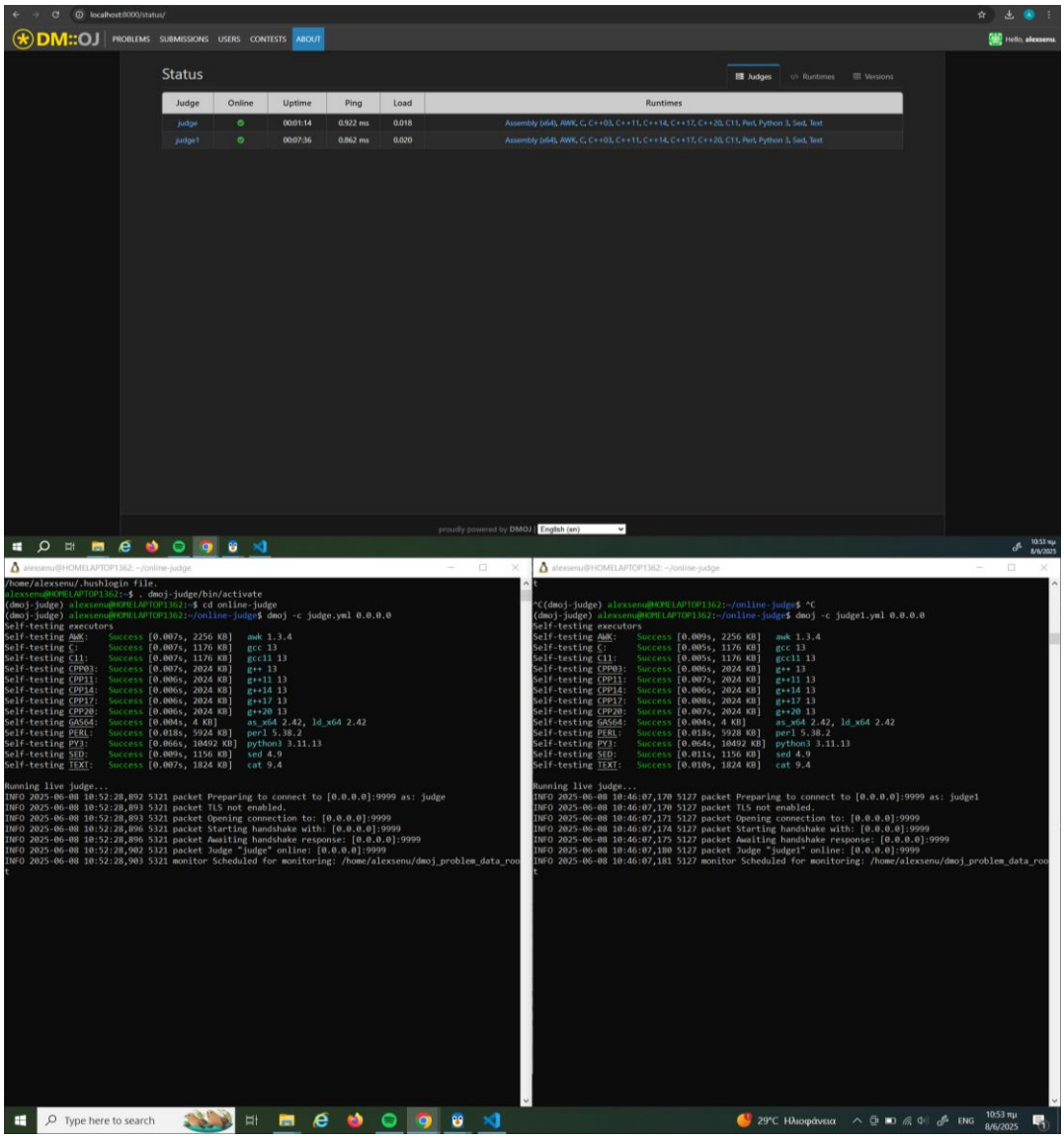


Εικόνα 1: Το πλαίσιο του DMOJ στο οποίο ο χρήστης υποβάλλει το κώδικα του για αξιολόγηση

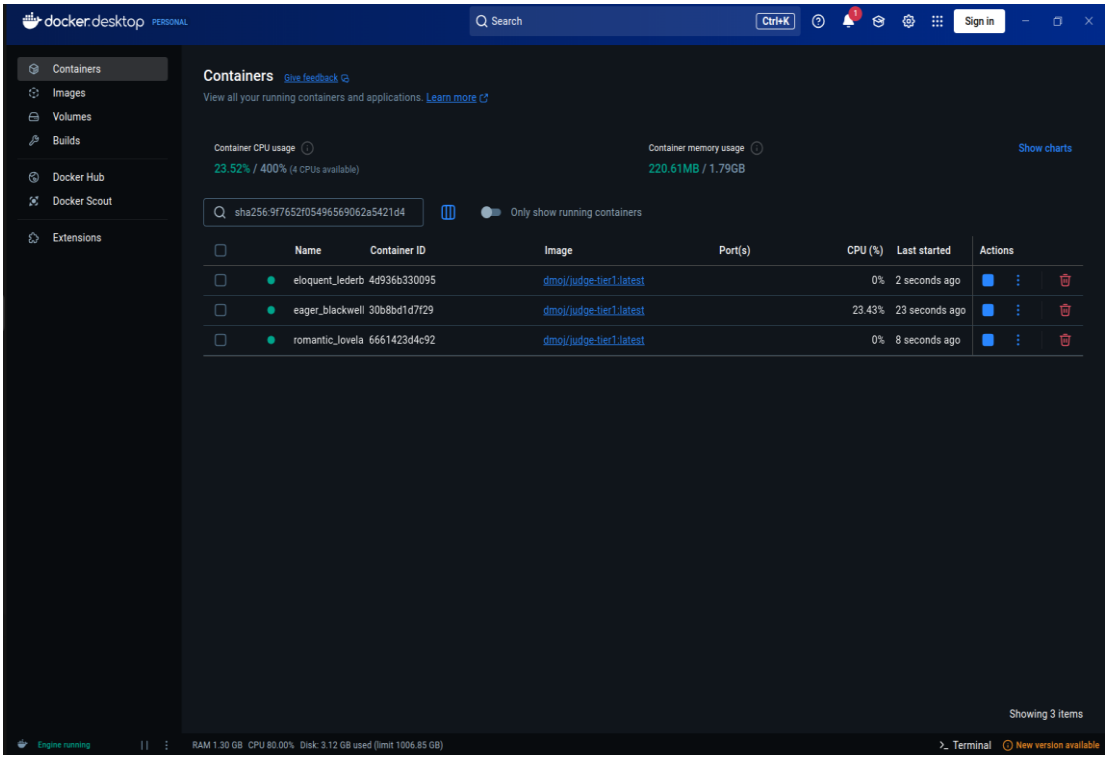
2. **Προώθηση στην ουρά εργασιών :** η εφαρμογή backend τοποθετεί την υποβολή σε μια ουρά εργασιών (task queue) ή την διαβιβάζει σε ένα ενδιάμεσο σύστημα διαχείρισης υποβολών. Αυτό το υποσύστημα (γνωστό ως *judge bridge*) αναλαμβάνει να συντονίσει την επικοινωνία

μεταξύ του κεντρικού συστήματος και των διαθέσιμων δικαστών. Έτσι διασφαλίζεται ότι οι υποβολές θα αξιολογηθούν σειριακά και δίκαια, χωρίς υπερφόρτωση συγκεκριμένου δικαστή.

3. **Ανάθεση σε δικαστή :** μόλις ένας διαθέσιμος δικαστής (αυτόνομη διαδικασία αξιολόγησης) είναι ελεύθερος, λαμβάνει από την ουρά την υποβολή προς εκτέλεση. Συνήθως εκτελούνται πολλαπλές ανεξάρτητες διαδικασίες δικαστών παράλληλα, ώστε να μπορούν να αξιολογούνται ταυτόχρονα περισσότερες από μία λύσεις και να αποφεύγεται η συσσώρευση μεγάλου φόρτου σε ουρά. Για παράδειγμα, στη περίπτωση που τρέχουν τρεις δικαστές σε containers μέσω του Docker(όπως φαίνεται στην εικόνα 2), η ιστοσελίδα θα μπορεί να διαχειριστεί την ίδια χρονική στιγμή τρεις διαφορετικες υποβολές-απαντήσεις. Επίσης, κάθε δικαστής έχει ρυθμιστεί με το κατάλληλο περιβάλλον ώστε να μπορεί να μεταγλωττίσει και να τρέξει προγράμματα σε διάφορες γλώσσες προγραμματισμού.

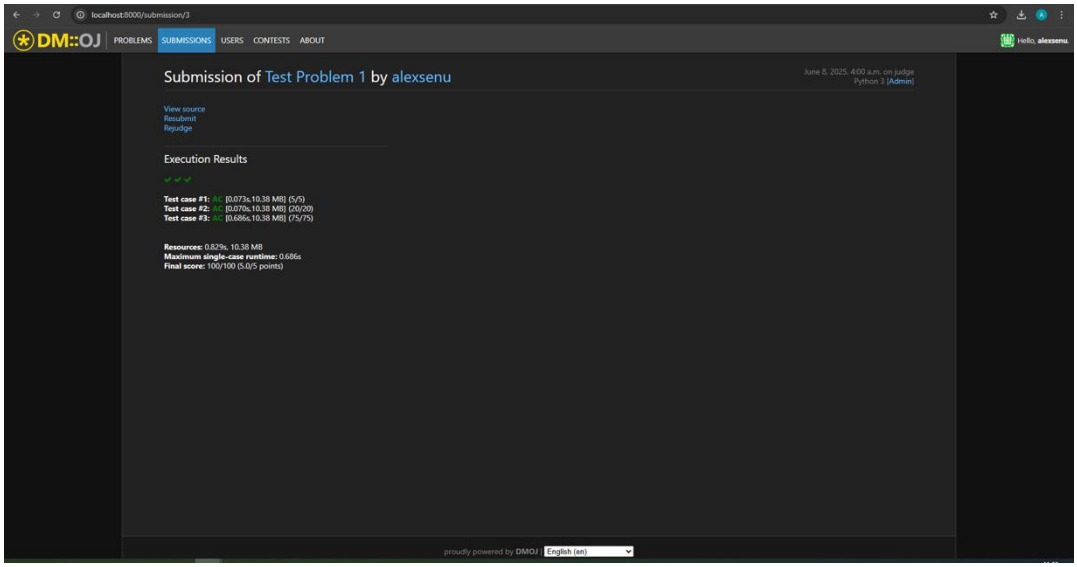


Εικόνα 2: παράδειγμα με 2 δικαστές να τρέχουν παράλληλα στα virtual environment τους όπως φαίνεται στο κάτω μέρος της εικόνας και το status στο πάνω μέρος της εικόνας που τα δείχνει ενεργά



Εικόνα 3: Παράδειγμα με 3 δικαστές να τρέχουν παράλληλα από μια εικόνα στο Docker

4. **Μεταγλώττιση και εκτέλεση :** ο δικαστής μεταγλωττίζει (compile) τον υποβληθέντα κώδικα στην γλώσσα που επιλέχθηκε και στη συνέχεια εκτελεί το πρόγραμμα, τρέχοντας το με τα προκαθορισμένα test cases (αρχεία εισόδων). Κατά την εκτέλεση του, το σύστημα παρακολουθεί τον χρόνο και τη μνήμη που καταναλώνεται από το πρόγραμμα.
5. **Έλεγχος αποτελεσμάτων :** με τη δοκιμή κάθε test case, η έξοδος που παράγει το πρόγραμμα συγκρίνεται με την αναμενόμενη έξοδο (output) του προβλήματος. Εάν σε οποιοδήποτε test case η έξοδος δεν ταιριάζει με το **.out** αρχείο ή παραβιάζεται κάποιος περιορισμός (π.χ. υπέρβαση χρονικού ορίου), η υποβολή χαρακτηρίζεται με το αντίστοιχο αποτέλεσμα. Αν όλα τα test cases επιτύχουν, η λύση χαρακτηρίζεται **Accepted**.
6. **Ενημέρωση και αποθήκευση αποτελέσματος :** ο δικαστής θα επιστρέψει το τελικό αποτέλεσμα πίσω στο κεντρικό σύστημα μέσω του bridged. Η βάση δεδομένων ενημερώνεται με την βαθμολογία/κατάσταση της υποβολής και η πληροφορία αυτή προωθείται στην frontend διεπαφή.
7. **Παρουσίαση ανατροφοδότησης :** ο χρήστης βλέπει στην ιστοσελίδα το αποτέλεσμα της υποβολής του (για παράδειγμα Accepted , Wrong Answer κλπ) μαζί με άλλα στοιχεία όπως ο χρόνος εκτέλεσης και η μνήμη που καταναλώθηκε. Σημαντικό ρόλο σε αυτό παίζει και ο ***διακομιστής συμβάντων*** (event server), μια ξεχωριστή υπηρεσία που στέλνει ζωντανά ενημερώσεις στον φυλλομετρητή του χρήστη κατά τη διάρκεια της αξιολόγησης. Μέσω αυτής της υπηρεσίας ο χρήστης μπορεί να βλέπει σε ζωντανά ποια test cases εκτελούνται και πότε ολοκληρώνονται.



Εικόνα 4: Ανατροφοδότηση αποτελέσματος ζωντανά

Κάτα αυτό το τρόπο, η πληροφορία ρέει ομαλά από την υποβολή μιας λύσης έως τη βαθμολόγηση της και την ενημέρωση του χρήστη. Η **συνολική δομή του συστήματος** αποτελείται από ένα σύνολο διακριτών στοιχείων, όπως ο web server (για την εξυπηρέτηση των αιτημάτων και τη φιλοξενία της εφαρμογής Django), η ίδια η εφαρμογή Django που είναι ο πυρήνας, πολλαπλοί δικαστές για την βαθμολόγηση των προγραμμάτων, μια βάση δεδομένων, καθώς και ο event server για ζωντανές ενημερώσεις. Πρακτικά, τα στοιχεία αυτά εκτελούνται ξεχωριστά ως διεργασίες/υπηρεσίες που επικοινωνούν όμως μεταξύ τους μέσω αιτημάτων ή ουρών μηνυμάτων. Για την καλύτερη λειτουργία και επίβλεψη σε ένα παραγωγικό περιβάλλον, χρησιμοποιούνται εργαλεία με ικανότητες παρακολούθησης των υπόλοιπων διεργασιών (όπως το Supervisor) ώστε να εκκινούν αυτόματα όλες τις απαραίτητες υπηρεσίες (Django server, bridged instance, Celery workers, event server)και να επανεκκινούν σε περίπτωση σφαλμάτων. Με την αρχιτεκτονική αυτή, το DMOJ επιτυγχάνει αξιοπιστία (αστοχία ενός δικαστή δεν καταρρίπτει ολόκληρο το σύστημα βαθμολόγησης) και ταχύτητα στην αξιολόγηση των λύσεων.

3.2 Βασικά συστατικά

Στην ενότητα αυτή παρουσιάζονται τα κύρια υποσυστήματα που αποτελούν την πλατφόρμα DMOJ, ο ρόλος τους καθώς και τα χαρακτηριστικά του καθενός. Η γενική δομή του συστήματος, ακολουθεί το κλασικό μοντέλο μιας web εφαρμογής backend-frontend, με επιπλέον στοιχεία για την ανάθεση και εκτέλεση των υποβολών. Τα βασικά συστατικά χωρίζονται ως εξής:

- **Διεπαφή ιστοσελίδας (Frontend)** : είναι το τμήμα της πλατφόρμας με το οποίο αλληλεπιδρά ο τελικός χρήστης μέσω του φυλλομετρητή. Περιλαμβάνει τις ιστοσελίδες για παρουσίαση των προβλημάτων και τις φόρμες υποβολής λύσεων, τις κατατάξεις/ranking και γενικά ό,τι βλέπει και μπορεί να διαχειριστεί ένας χρήστης. Η διεπαφή υλοποιείται ως ένα web περιβάλλον, με σελίδες που δημιουργούνται από την Django σε συνδυασμό με static assets (CSS/JavaScript) για το παρουσιαστικό και τη διαδραστικότητα. Υποστηρίζονται πολλές γλώσσες, όπως και τα ελληνικά, ώστε οι χρήστες να μπορούν να χρησιμοποιούν την

πλατφόρμα στη γλώσσα που επιθυμούν. Επιπλέον, μέσω JavaScript και της υπηρεσίας συμβάντων (event server), η διεπαφή προσφέρει ζωντανές ενημερώσεις, όπως σε μια σελίδα προβολής της υποβολής μιας απάντησης, ο χρήστης βλέπει την κατάσταση εκτέλεσης σε πραγματικό χρόνο χωρίς ανανέωση σελίδας (live status).

Front-End	Back-End
Η εμφάνιση του ιστότοπου	Ο τρόπος λειτουργίας του ιστότοπου
HTML, CSS, JavaScript	Python, MariaDB, Redis
Απόδοση από την πλευρά του χρήστη της ιστοσελίδας	Απόδοση από την πλευρά του διακομιστή

Πίνακας 1: Ένα μικρό μέρος των στοιχείων που αποτελούν το backend-frontend του συστήματος DMOJ

- **Εφαρμογή Django (Backend) :** αποτελεί το κεντρικό πυρήνα του DMOJ. Είναι μια web εφαρμογή γραμμένη σε Python και βασισμένη στο framework Django. Σε τοπική παραγωγή το Django μπορεί να τροφοδοτήσει ένα μικρό WSGI(Web Server Gateway Interface). Για global παραγωγή φορτώνεται σε έναν application server (uWSGI), πίσω από τον web server (NGINX), διαχειρίζοντας όλα τα αιτήματα των χρηστών. Με το Django **ORM** η εφαρμογή επικοινωνεί με τη βάση δεδομένων για την αποθήκευση και ανάκτηση πληροφοριών (π.χ. στοιχεία χρήστη, δήλωση προβλήματος, αποτελέσματα υποβολών). Επιπλέον, το Django παρέχει και μια διοικητική διεπαφή (admin interface) που επιτρέπει σε διαχειριστές του συστήματος να επεξεργάζονται περιεχόμενο (να εισάγουν νέα προβλήματα, να ελέγχουν χρήστες κ.ά.), κάτι που προσφέρει ευελιξία στη διαχείριση της πλατφόρμας.
  *Σημείωση : το **ORM** (Object-Relational Mapping) είναι το σύστημα που παρέχει το Django για την αλληλεπίδραση της εφαρμογής με βάσεις δεδομένων, χρησιμοποιώντας αντικείμενα και κλάσεις Python αντί για την παραδοσιακή SQL.*

- **Βάση δεδομένων :** το σύστημα χρησιμοποιεί μια σχεσιακή βάση δεδομένων (SQL) για να αποθηκεύει όλα τα απαραίτητα δεδομένα. Στην συγκεκριμένη εγκατάσταση, η MariaDB (συμβατή με MySQL) λειτουργεί ως σύστημα διαχείρισης βάσης δεδομένων. Η βάση δεδομένων, περιέχει πίνακες για τους χρήστες, τα δικαιώματά τους, τις δηλώσεις των προβλημάτων και τις υποβολές λύσεων, βαθμολογίες, πληροφορίες διοργανώσεων, τα σχόλια στις συζητήσεις κ.ά. Η ακεραιότητα αυτών των δεδομένων είναι σημαντική, καθώς αυτά είναι και η πληροφορία της Django εφαρμογής.
  Σημείωση : το DMOJ μπορεί να λειτουργήσει και με άλλες βάσεις δεδομένων που υποστηρίζονται από το Django, αλλά η MySQL-based λύση είναι η πλέον δοκιμασμένη στην πράξη.

- **Ουρά εργασιών (Task Queue)** : το σύστημα υποστηρίζει ασύγχρονη εκτέλεση εργασιών στο παρασκήνιο, μέσω μιας ουράς εργασιών ειδικά για απαιτητικές ή χρονοβόρες λειτουργίες. Συγκεκριμένα, χρησιμοποιεί το σύστημα **Celery** (Python task queue) σε συνδυασμό με έναν μεσίτη μηνυμάτων (Redis), για την υλοποίηση αυτών των εργασιών. Η εφαρμογή Django,

στέλνει εργασίες στην ουρά και οι **Celery workers** τις επεξεργάζονται αυτόνομα. Με αυτόν τον τρόπο, η κύρια εφαρμογή δεν μπλοκάρει αναμένοντας την ολοκλήρωση μιας εργασίας που μπορεί να καθυστερεί ή να έχει "κολλήσει" – αντίθετα, η εργασία εκτελείται στο παρασκήνιο, και το αποτέλεσμα, επιστρέφει όταν είναι έτοιμο. Το Celery ενσωματώνεται μέσα στο Django project (με το **dmoj_celery** module στο συγκεκριμένο project) και χρησιμοποιείται η **Redis** ως μεσίτη μηνυμάτων για την ανταλλαγή των δεδομένων της ουράς. Πρακτικά, η Redis λειτουργεί ως μια in-memory βάση τύπου NoSQL που καταχωρεί τις εργασίες και διαμεσολαβεί μεταξύ παραγωγών (Django) και καταναλωτών (Celery workers). Η υποδομή Celery/Redis εκτελεί επίσης εργασίες όπως αποστολή email, περιοδικές εκκαθαρίσεις ή άλλες διαδικασίες που δεν είναι επιθυμητό να εκτελούνται στον βασικό server της εφαρμογής.

- **Bridge και δικαστές (Judges)** : αυτά τα υποσυστήματα αποτελούν την καρδιά της αυτόματης αξιολόγησης. Ο Bridge – ή αλλιώς γέφυρα των δικαστών – είναι μια υπηρεσία συντονισμού, που μεσολαβεί μεταξύ του κεντρικού Django site και των επιμέρους δικαστών. Ο Bridge διατηρεί ανοικτή την επικοινωνία με κάθε ενεργό δικαστή όπου τους αναθέτει νέες υποβολές όταν είναι διαθέσιμοι. Μπορεί να θεωρηθεί ως ένας διακομιστής ουράς για τις υποβολές προς βαθμολόγηση, με επιπλέον λογική δρομολόγησης και ελέγχου. Οι δικαστές είναι ανεξάρτητες διεργασίες που εκτελούν το λογισμικό αξιολόγησης του DMOJ (**dmoj/judge-server** στο github). Κάθε δικαστής συνδέεται στο bridged μέσω δικτύου (συνήθως TCP socket σε καθορισμένη θύρα) και περιμένει για ανάθεση εργασιών. Όταν λάβει μια υποβολή τη μεταγλωττίζει αν χρειάζεται και την τρέχει όπως περιγράφηκε νωρίτερα, μέσα σε ασφαλές περιβάλλον, και επιστρέφει το αποτέλεσμα. Οι δικαστές είναι σχεδιασμένοι να εκτελούν μία υποβολή τη φορά για να μην παρεμβάλλονται πολλαπλά προγράμματα στην ίδια διεργασία. Για να αυξηθεί ο συνολικός ρυθμός αξιολόγησης, το σύστημα υποστηρίζει την ταυτόχρονη λειτουργία πολλών δικαστών – είτε στον ίδιο server, είτε σε πολλούς απομακρυσμένους κόμβους. Επιπλέον, αν κάποιος δικαστής παρουσιάσει βλάβη ή κολλήσει, οι υπόλοιποι συνεχίζουν και ο bridge μπορεί να επαναδρομολογήσει τις εργασίες που δεν ολοκληρώθηκαν.
- **Υπηρεσία συμβάντων (Event Server) :** πρόκειται για μια υπηρεσία υπεύθυνη για τις ενημερώσεις (real-time notifications) προς τους επισκέπτες. Στην πράξη, όταν ένας χρήστης έχει ανοιχτή την σελίδα μιας υποβολής που βρίσκεται υπό αξιολόγηση, θέλει να βλέπει άμεσα το στάδιο της εκτέλεσης (π.χ. "Running on test 3...") και το τελικό αποτέλεσμα χωρίς να χρειαστεί χειροκίνητη ανανέωση. Το DMOJ πετυχαίνει αυτή τη λειτουργεία με έναν ξεχωριστό event server, υλοποιημένο σε **Node.js**. Ο event server ανοίγει μια μόνιμη σύνδεση με τον browser του χρήστη (μέσω WebSocket) και επίσης είναι συνδεδεμένος με την κεντρική εφαρμογή και τη βάση δεδομένων για να πληροφορείται τις αλλαγές κατάστασης. Για παράδειγμα, καθώς ο δικαστής αξιολογεί μια υποβολή και ολοκληρώνει test cases, το backend δημοσιεύει σχετικά γεγονότα τα οποία ο event server μεταφέρει στον browser. Στη παρούσα εγκατάσταση, ο event server ρυθμίστηκε να λειτουργεί στην θύρα 15102 και έγινε κατάλληλη ρύθμιση του NGINX για proxy προς αυτόν (όλα τα αιτήματα των clients στο URL **/channels/** προωθούνται στο Node event server). Έτσι, ο event server ενσωματώνεται ομαλά στην ίδια διεύθυνση web του site, παρέχοντας ασφαλή και συντονισμένη επικοινωνία.
- **Στατικά αρχεία (static files)** : ένα υποσύστημα που δεν είναι λειτουργικό μεν αλλά απαραίτητο για την ολοκλήρωση της εμφάνισης της πλατφόρμας, είναι η διαχείριση των

στατικών αρχείων – δηλαδή των αρχείων JavaScript, CSS, εικόνων και λοιπού πολυμεσικού υλικού που χειρίζεται η διεπαφή. Το DMOJ, όντας Django εφαρμογή, οργανώνει τα στατικά του αρχεία μέσα στο έργο και κάνει χρήση εργαλείων frontend για τη τοποθέτηση τους. Συγκεκριμένα χρησιμοποιείται η γλώσσα **Sass** για τους ορισμούς των CSS και εργαλεία όπως τα **PostCSS** και **Autoprefixer** για τη βελτιστοποίηση των CSS καθώς και bundling/συμπίεση των JavaScript αρχείων. Για τον σκοπό αυτό, απαιτείται η παρουσία Node.js και του npm, τα οποία χρησιμοποιούνται για να γίνει η εγκατάσταση των σχετικών εργαλείων και εκτέλεση των build scripts (π.χ. **npm install** ακολουθούμενο από εντολές για compile Sass σε CSS). Μετά την παραγωγή των static assets, γίνεται συλλογή τους (collectstatic) σε ένα παραγωγικό περιβάλλον και σερβίρονται απευθείας από τον web server (NGINX) για λόγους απόδοσης. Τα στατικά αρχεία, εξασφαλίζουν μια φιλική και σύγχρονη εμπειρία χρήστη με όμορφο UI και άμεση απόκριση, χωρίς να επιβαρύνουν τον backend κατά τη λειτουργία του(σερβίρονται ως αρχεία cache).

3.3 Τεχνολογική στοίβα

Η υλοποίηση του DMOJ είναι βασισμένη σε μια σύγχρονη τεχνολογική στοίβα, που επιλέχθηκε ώστε να καλύπτει τόσο τις ανάγκες μιας web εφαρμογής όσο και ενός συστήματος αυτόματης αξιολόγηση κώδικα. Παρακάτω αναφέρονται οι τεχνολογίες και τα εργαλεία που χρησιμοποιεί το σύστημα, καθώς και ο ρόλος τους:

- **Γλώσσα προγραμματισμού Python :** ο κώδικας της πλατφόρμας είναι γραμμένος σε Python, κάτι που προσφέρει ευελιξία και αλλά και ταχεία ανάπτυξη. Η Python επιλέχθηκε για την καθαρή σύνταξη, τις ισχυρές βιβλιοθήκες της και το μεγάλο οικοσύστημα που έχει στη διάθεση της. Θεωρείται κατάλληλη για την υλοποίηση web servers (μέσω Django), για scripting ολικής διαχείρισης, αλλά και για τον συντονισμό των κριτηρίων αξιολόγησης στον δικαστή. Παρόλο που η Python είναι διερμηνευόμενη και όχι τόσο γρήγορη όσο οι γλώσσες συστήματος, το γεγονός αυτό αντισταθμίζεται από το ότι οι κρίσιμες λειτουργίες, όπως η εκτέλεση του υποβαλλόμενου κώδικα,συμβαίνει εκτός του κύριου loop της εφαρμογής. Η Python προσφέρει αρκετή απόδοση για το χειρισμό των web requests και του συντονισμού.
- **Framework Django :** το Django αποτελεί τη ραχοκοκαλιά του web κομματιού του DMOJ. Ως πλήρες MVC(Model-View-Controller) framework, όλα τα εργαλεία για την κατασκευή μιας ασφαλούς και επεκτάσιμης web εφαρμογής παρέχονται. Για την πλατφόρμα DMOJ, το Django επιλέχθηκε λόγω της εύκολης οργάνωσης της πολύπλοκης λειτουργικότητας (π.χ. προβλήματα, υποβολές, contests) σε επαναχρησιμοποιήσιμες εφαρμογές. Μέσω **admin interface** το Django αξιοποιείται για τη διαχείριση περιεχομένου (π.χ. ένας admin μπορεί να εισάγει από εκεί νέα προβλήματα αντί να χρειάζεται πρόσθετο custom UI).
- **Βάση δεδομένων MariaDB/MySQL :** για την αποθήκευση δεδομένων χρησιμοποιείται μια MySQL-συμβατή βάση. Η **MariaDB** είναι fork της MySQL με ανοικτό κώδικα και παρέχει βελτιωμένη απόδοση σε πολλές περιπτώσεις και ενεργή ανάπτυξη. Σε επίπεδο λειτουργικότητας για το DMOJ, η επιλογή μεταξύ MySQL και MariaDB είναι ως επί το πλείστον διαφανής και δεν υπάρχουν διαφορές. Η χρήση μιας σχεσιακής βάσης δεδομένων διασφαλίζει ότι μπορούν να γίνονται πολύπλοκα ερωτήματα (π.χ. εύρεση ποσοστών επιτυχίας ανά πρόβλημα) με αποδοτικό τρόπο. Η λήψη backup, η αναπαραγωγή (replication) και άλλες διοικητικές λειτουργίες είναι γνώριμες και υποστηρίζονται πλήρως από την

τεχνολογία MariaDB. Τέλος, αξίζει να σημειωθεί, πως η επίσημη τεκμηρίωση του DMOJ προτείνει τη χρήση της τελευταίας σταθερής έκδοσης της MariaDB.

- **Redis :** είναι ένας ελαφρύς ενσωματωμένος διακομιστής δομών δεδομένων τύπου key-value. Στο DMOJ, η Redis χρησιμοποιείται κυρίως για να διαχειρίζεται τα αιτήματα του Celery (ουρά εργασιών), επιτρέποντας την ταχεία προώθηση εργασιών και την παραλαβή αποτελεσμάτων. Λόγω του ότι τα δεδομένα τα διατηρεί στη μνήμη RAM, έχει χαμηλή καθυστέρηση αλλά υψηλή απόδοση, χαρακτηρηστικά που βοηθούν στην άμεση διάθεση εργασιών προς τους δικαστές. Επιπλέον, η Redis μπορεί να χρησιμοποιηθεί και ως cache για τη Django (π.χ. caching σελίδων), αν και αυτό είναι προαιρετικό και εξαρτάται από τη γενική ρύθμιση του έργου. Στην αρχιτεκτονική του DMOJ η Redis λειτουργεί συμπληρωματικά, δηλαδή δεν αποθηκεύει μόνο κρίσιμα δεδομένα, αλλά δρα και ως αγωγός ταχείας μετάδοσης των μηνυμάτων. Επίσης, μπορεί να λειτουργήσει και ως event server αφού η Django εφαρμογή μπορεί να δημοσιεύει γεγονότα (events) σε κάποιο Redis channel και ο Node.js event server να φιλοξενείται λαμβάνοντας γεγονότα και μεταδίδοντάς στους web clients.
- **Node.js & Socket.io :** η παρουσία του **Node.js** στην τεχνολογική στοίβα του DMOJ συνδεέται αποκλειστικά με τον event server. Το Node σαν μια πλατφόρμα εκτέλεσης JavaScript στον server, η οποία φημίζεται για το event-driven μοντέλο της αλλά και για τις non-blocking I/O ιδιότητες της, την καθιστά ιδανική για εφαρμογές όπου υπάρχει μεγάλη δραστηριότητα εισόδου και εξόδου, και διατηρεί πολλές ανοιχτές συνδέσεις, όπως ακριβώς απαιτείται από ένα διακομιστή πραγματικού χρόνου. Στο DMOJ ο event server, που είναι γραμμένος σε Node, χρησιμοποιεί τη βιβλιοθήκη **Socket.io** για να διαχειρίζεται WebSocket συνδέσεις με τους φυλλομετρητές. Με αυτό το τρόπο μπορεί να στέλνει άμεσα δεδομένα (π.χ. ένα μήνυμα JSON με τα αποτελέσματα των test cases) σε έναν συγκεκριμένο χρήστη. Με το Node να λειτουργεί ως ξεχωριστό service, απομονώνεται και η ευθύνη του real-time κομματιού. Σημαντικό είναι και ότι ο Node event server φιλοξενείται πίσω από το ίδιο το Nginx, ώστε να μην αντιλαμβάνονται οι χρήστες διαφορετικό origin ή θύρα και όλα να δρομολογούνται μέσω κεντρικού domain.
- **Nginx + uWSGI (Web Server Layer) :** αν και δεν αναφέρεται στις προηγούμενες περιγραφές, η παραγωγική ανάπτυξη του DMOJ έχει ανάγκη από έναν σταθερό web server. Το **Nginx** μπορεί να χρησιμοποιηθεί ως αντίστροφος διακομιστής μεσολάβησης και static file server, μαζί με τον **uWSGI** ως application server του Django. Το Nginx λαμβάνει τα HTTP αιτήματα από τους πελάτες και είτε θα τα σερβίρει απευθείας αν είναι static file το περιεχόμενο είτε θα τα προωθήσει στον uWSGI που θα εκτελέσει τον κώδικα Django. Ο uWSGI (Gunicorn) διαχειρίζεται πολλαπλές διεργασίες των workerd για το Django, φροντίζοντας για την επεξεργασία παράλληλων αιτημάτων. Συνολικά, η υποδομή αυτή είναι στάνταρ για τις Django εφαρμογές και εξασφαλίζει το σύστημα ώστε να μπορεί να εξυπηρετεί πολλούς χρήστες γρήγορα και με ασφάλεια.
- **Docker (περιβάλλον container)** : αν και δεν είναι αναπόσπαστο κομμάτι για τη πλατφόρμα DMOJ, αξίζει να αναφερθεί η χρησιμότητα του Docker. Η επιλογή πολλών διαχειριστών είναι να τρέξουν τον δικαστή μέσα σε Docker container. Η λειτουργική ομάδα του DMOJ έχει χτίσει έτοιμες Docker images που περιλαμβάνουν όλα τα απαραίτητα runtime περιβάλλοντα(C++, Java, Python κ.λπ.) ταξινομημένα σε tiers. Με αυτή τη προσέγγιση απλοποιείται η εγκατάσταση, καθώς επιτρέπει την ανάπτυξη ενός νέου δικαστή, από το pull μιας εικόνας και τρέχοντάς την. Επίσης, ενισχυείται η απομόνωση κάθε δικαστή αφού κάθε container τρέχει

σε δικό του sandbox περιβάλλον με περιορισμένους πόρους αλλά και το σημαντικότερο χωρίς πρόσβαση στο host σύστημα.

Συνοψίζοντας τη τεχνολογική στοίβα του DMOJ, με τη Python/Django ως βασικό web application και διαχειριστή κώδικα του δικαστή, MariaDB/MySQL για την αποθήκευση των δεδομένων, την Redis για την γρήγορη αποθήκευση και επικοινωνία, Celery για την ασύγχρονη επεξεργασία εργασιών, Node.js για τις ζωντανές ανατροφοδοτήσεις, και Docker για την ασφαλή λειτουργία των δικαστών. Κάθε τεχνολογία παίζει το ρόλο της και συνεργάζονται μεταξύ τους – η Django μαζί με τη MariaDB για τα δεδομένα, η Django με Celery και Redis για tasks, η Django/bridged για τους δικαστές, ο Node.js με τη Django/Redis για τα real-time events. Αυτή η αρχιτεκτονική πολλών επιπέδων, μπορεί να καλύψει το ιδιαίτερο πρόβλημα που λύνει το DMOJ: να κρίνει αυτόματοποιημένα με ασφάλεια τον κώδικα.

3.4 Θέματα ασφαλείας

Το πιο κρίσιμο ζητήμα σε ένα σύστημα αυτόματης βαθμολόγησης σαν το DMOJ είναι η **ασφάλεια που προσφέρεται κατά την εκτέλεση αυθαίρετου κώδικα**. Δεδομένου και ότι οι χρήστες υποβάλλουν τα δικά τους προγράμματα, τα οποία το σύστημα θα μεταγλωττίσει και θα τρέξει, καθιστά απαραίτητη την ύπαρξη μηχανισμών που θα αποτρέψουν κακόβουλες ενέργειες και διασφαλίσουν ότι η εκτέλεση γίνεται σε ελεγχόμενο περιβάλλον. Το DMOJ αντιμετωπίζει τις προκλήσεις αυτές με έναν συνδυασμό λειτουργιών όπως η τεχνική απομόνωσης (sandboxing) και ο περιορισμών πόρων.

Sandbox εκτέλεσης : κατά την λειτουργία τους οι δικαστές του DMOJ ενσωματώνουν έναν μηχανισμό sandbox, ένα *περιορισμένο περιβάλλον* δηλαδή, μέσα στο οποίο ο κώδικας του χρήστη τρεχει. Πιο συγκεκριμένα, όταν ο δικαστής ξεκινήσει την εκτέλεση προγράμματος, εφαρμόζει διάφορους περιορισμούς. Μπορεί να θέσει όρια (time limits) στον χρόνο CPU που μπορεί να χρησιμοποιηθεί, στη ποσότητα της μνήμης RAM που μπορεί να δεσμευτεί, στο μέγεθος των αρχείων εξόδου, καθώς και τον αριθμό διεργασιών που μπορεί να διαχειριστεί το πρόγραμμα. Με αυτόν τον τρόπο αντιμετωπίζονται κακόβουλες ή προβληματικές συμπεριφορές όπως ένα πρόγραμμα σε άπειρη επανάληψη(infinite loop) που θα τερματιστεί από τον χρονομετρητή (Time Limit Exceeded) ή ένα πρόγραμμα που προσπαθεί να καταναλώσει όλη τη διαθέσιμη μνήμη και θα διακοπεί μόλις ξεπεράσει το όριο (Memory Limit Exceeded), ενώ προσπάθειες δημιουργίας υπερβολικών διεργασιών (fork bombs)επίσης διακόπτονται από αντίστοιχα όρια που έχουν εφαρμοστεί. Επιπλέον, το sandbox μπορεί να **απομονώσει το πρόγραμμα**, εκτελώντας ως χαμηλών δικαιωμάτων χρήστης στο λειτουργικό και χωρίς πρόσβαση σε ευαίσθητες τοποθεσίες, όπως τα αρχεία συστήματος. Ο δικαστής ***δεν τρέχει ποτέ ως root***, και ο λογαριασμός χρήστη στον οποίο εκτελούνται τα υποβαλλόμενα προγράμματα είναι προσεκτικά περιορισμένος από τον διαχειριστή του συστήματος (π.χ. μπορεί να ανήκει σε cgroup με όριο CPU/MEM, να έχει jail περιορισμούς στο filesystem, και ο δικτυακός του τόνος να είναι φραγμένος ώστε να μην επικοινωνεί με τυχαίες IP). Έτσι, ακόμα και αν κάποιο πρόγραμμα που υποβάλλεται επιχειρήσει ενέργειες εκτός πλαισίου (π.χ. να κάνει κάποια τροποποίηση στα αρχεία συστήματος ή να ανοίξει δίκτυο), οι ενέργειές του είτε θα αποτυγχάνουν είτε θα περιορίζονται αυστηρά εντός του sandbox όπου και θα τερματίζεται.

Περιορισμοί πόρων και έλεγχος κακόβουλων υποβολών : οπως αναφέρθηκε προηγουμένως, τα βασικά όπλα της εφαρμογής ενάντια σε κακόβουλο κώδικα είναι τα όρια χρόνου και μνήμης. Για

παράδειγμα, αν ένας χρήστης κάνει υποβολή ένος προγράμματος που κολλά σε ατέρμονα βρόχο ή προσπαθεί να κάνει υπολογισμούς πολύ πέραν του επιτρεπόμενου χρόνου, ο χρονομετρητής θα το διακόψει όταν φτάσει το όριο (π.χ. 2 δευτερόλεπτα CPU χρόνου για ένα test case)[9]. Ομοίως, αν επιχειρήσει να δημιουργήσει τεράστιες δομές δεδομένων στη μνήμη ή να φορτώσει ολόκληρα αρχεία στην RAM, θα εμφανιστει σφάλμα μνήμης. Μια πιο ύπουλη μορφή επίθεσης είναι η προσπάθεια υπερφόρτωσης της πλατφόρμας με **εσκεμμένα βαριές υποβολές** – π.χ. υποβολή πολλαπλών λύσεων που θα κάνουν 5-6 δευτερόλεπτα η καθεμία, με σκοπό την καθυστέρηση του συστήματος). Το DMOJ αποτρέπει αυτές τις προσπάθειες αφενός με τους περιορισμούς ανά υποβολή και αφετέρου με την **κλιμάκωση του συστήματος**. Σε περιπτώση διαγωνισμών υψηλού φόρτου, ο διαχειριστής μπορεί να προσθέσει επιπλέον δικαστές ώστε η συνολική χωρητικότητα αξιολόγηση να αυξηθεί. Επίσης υπάρχουν και **ρυθμίσεις στο επίπεδο εφαρμογής** (αν και αυτές δεν είναι δημοσίως γνωστές) όπως το όριο στις υποβολές ανά λεπτό για να αποφευχθούν spam submissions.

Αποτροπή προσπέλασης και δικτυακές περιορίσεις : το sandbox των δικαστών φροντίζει για τον **ελεγχο της πρόσβασης σε συστήματα εκτός του πεδίου αξιολόγησης**. Το υποβαλλόμενο πρόγραμμα δεν πρέπει να μπορεί να έχει επικοινωνία με τον διακομιστή της βάσης δεδομένων ή με τον κεντρικό server πέραν του επιτρεπόμενου. Στην υλοποίηση με Docker αυτό επιτυγχάνεται ευκολότερα, καθώς το container του δικαστή μπορεί να έχει μόνο πρόσβαση στη διεύθυνση/θύρα του bridge (με firewall κανόνες που να το περιορίζουν). Έτσι εάν ένας κακόβουλος κώδικας που θα επιχειρούσε να ανοίξει socket σε κάποιον εξωτερικό server, δεν θα έβρισκε δίκτυο. Επίσης, το filesystem που βλέπει το πρόγραμμα είναι περιορισμένο σε δικό του φάκελο. Συνήθως ο δικαστής δημιουργεί έναν προσωρινό κατάλογο και αντιγράφει εκεί το εκτελέσιμο αλλά και τα test cases, και οριοθετεί αυτόν ως working directory. Δεν έχει καμία πρόσβαση σε **/etc/** ή σε αρχεία άλλων χρηστών.

Φυσική απομόνωση δικαστών : σε αρκετές εγκαταστάσεις, οι διαχειριστές επιλέγουν να τρέχουν τους δικαστές σε ξεχωριστά μηχανήματα. Αυτή η πρακτική σημαίνει ότι ακόμη και αν κάτι πάει τελείως στραβά και ένας κακόβουλος κώδικας καταφέρει να αποκτήσει πρόσβαση στο σύστημα του δικαστή, δεν θα επηρεάσει τον κεντρικό server ή τη βάση δεδομένων, παρά μόνο τον μεμονωμένο αυτό host. Στην περίπτωση του DMOJ, η αρχιτεκτονική με το bridged το καθιστά εύκολο καθώς οι δικαστές συνδέονται μέσω δικτύου, οπότε μπορούν να φιλοξενούνται από οπουδήποτε. Η χρήση του Docker είναι μια ενδιάμεση λύση (απομόνωση σε επίπεδο λογισμικού), ενώ η απομόνωση σε επίπεδο hardware/VM είναι το επόμενο βήμα για ακόμη μεγαλύτερη σιγουριά. Φυσικά, για ακαδημαϊκές ή μικρότερες εγκαταστάσεις, ένας δικαστής στο ίδιο μηχάνημα με το site είναι πιο πρακτικός – αλλά ακόμη και εκεί, η εκτέλεση υπό διαφορετικό χρήστη και με τα κατάλληλα όρια παραμένει σε ισχύ.

Ασφάλεια της ίδιας της πλατφόρμας : πέραν του κομματιού της εκτέλεσης κώδικα, το DMOJ ως web εφαρμογή υιοθετεί και τα μέτρα ασφαλείας των web frameworks. Το Django παρέχει προστασία σε επιθέσεις SQL injection μέσω ORM, προστασία έναντι σε XSS επιθέσεις μέσω της αυτόματης απόδρασης ειδικών χαρακτήρων στα templates. Οι κωδικοί των χρηστών αποθηκεύονται με κρυπτογραφημένα hash στη βάση, ενώ υποστηρίζεται και two-factor authentication για πρόσθετη ασφάλεια λογαριασμών. Επιπλέον, το DMOJ διαθέτει ρυθμίσεις ρόλων και δικαιωμάτων μέσω των οποίων επιτρέπει τον ορισμό curators, testers κ.λπ. με περιορισμένες αρμοδιότητες αντί όλοι οι admin χρήστες να έχουν πλήρη δικαιώματα. Αυτά συμβάλλουν στην **εσωτερική ασφάλεια** του συστήματος, αποτρέποντας κατάχρηση από καλοπροαίρετους αλλά περίεργους χρήστες. Τέλος, αξίζει να αναφερθεί ότι η κοινότητα του DMOJ αντιμετωπίζει σοβαρά τα ζητήματα ασφαλείας·

υπάρχουν παραδείγματα **CTF διαγωνισμών** που στήθηκαν πάνω στο DMOJ με στόχο να βρεθούν τρωτά σημεία (π.χ. **Java Sandbox Breakout** διαγωνισμός), κάτι που δείχνει ότι το σύστημα δοκιμάζεται και βελτιώνεται συνεχώς σε θέματα sandbox ασφάλειας.

Συμπερασματικά, τα θέματα ασφαλείας στο DMOJ καλύπτονται σε πολλαπλά επίπεδα. Από το ασφαλές sandboxing και τους περιορισμούς πόρων στο επίπεδο του λειτουργικού, μέχρι την ασφαλής ανάπτυξη web εφαρμογών στο Django. Η εκτέλεση αυτών των προγραμμάτων, γίνεται σε τόσο απομονωμένο πλαίσιο, ώστε να προστατεύεται το σύστημα από κακόβουλες ή εσφαλμένες συμπεριφορές. Ετσι η πλατφόρμα έχει τη δυνατότητα να παρέχει ένα αξιόπιστο περιβάλλον διαγωνισμών και εξάσκησης, όπου οι συμμετέχοντες μπορούν να εμπιστεύονται ότι οι λύσεις τους θα αξιολογηθούν δίκαια και με ασφάλεια, χωρίς να διακινδυνεύεται η σταθερότητα του συστήματος.

Κεφάλαιο 4: Οδηγός Εγκατάστασης της Πλατφόρμας DMOJ

4.1 Απαιτήσεις συστήματος

Για την επιτυχημένη εγκατάσταση και απρόσκοπτη λειτουργία της πλατφόρμας DMOJ σε τοπικό περιβάλλον (localhost), είναι απαραίτητο το υπολογιστικό σύστημα να πληροί ορισμένες τεχνικές προδιαγραφές και να διαθέτει τις κατάλληλες εξαρτήσεις[10].

Η εγκατάσταση προτείνεται να γίνει σε λειτουργικό σύστημα βασισμένο σε Linux καθώς προσφέρει σταθερότητα, μεγάλη υποστήριξη και συμβατότητα με τις τεχνολογίες που χρησιμοποιεί το DMOJ. Ο δικαστής για παράδειγμα μόνο μέσω λειτουργικών συστημάτων βασισμένα σε Linux μπορεί να εγκατασταθεί και να λειτουργήσει. Ωστόσο, όπως αναφέρεται και στο επίσημο αποθετήριο του DMOJ, η πλήρης υποστήριξη δίνεται στο Debian, με χρήση των προεπιλεγμένων εκδόσεων πακέτων μέσω του apt. Αυτό δεν αποκλείει τη χρήση Ubuntu, καθώς στην πράξη έχει αποδειχθεί απόλυτα συμβατό, αρκεί να χρησιμοποιούνται οι ίδιες ή συμβατές εκδόσεις πακέτων. Το εργό μπορεί να πραγματοποιηθεί και με WSL(windows subsystem for linux).

Οι βασικές απαιτήσεις του συστήματος περιλαμβάνουν:

- Linux - το λειτουργικο σύστημα
- MariaDB – ως σύστημα διαχείρισης βάσεων δεδομένων (MySQL-compatible)
- Redis – λειτουργεί ως μεσίτης στη διαχείριση μηνυμάτων της εφαρμογής
- Python 3.11 για το δικαστή – Python 3.12 για την ιστοσελίδα
- Node.js και npm – για την παραγωγή και διαχείριση αρχείων CSS/JS (frontend assets)
- Supervisor – για την παρακολούθηση και αυτόματη εκκίνηση υπηρεσιών (bridged, site, celery, event server).
- Git – για λήψη και διαχείριση του πηγαίου κώδικα από το αποθετήριο του DMOJ.
- Εργαλεία μεταγλώττισης (π.χ. gcc, g++, make) και βιβλιοθήκες ανάπτυξης για Python και MySQL.
- Docker - για τη διαχείριση της λειτουργίας του δικαστή(judge image)

Επιπλέον, για την πλήρη αξιοποίηση του συστήματος, είναι αναγκαίο να υπάρχουν προεγκατεστημένες γλώσσες προγραμματισμού (όπως Python, C++, Java) για να μπορεί ο δικαστής να αξιολογεί τις υποβολές. Οι γλώσσες αυτές πρέπει να εγκατασταθούν μέσω apt.

4.2 Προετοιμασία εγκατάστασης

Εγκατάσταση προαπαιτούμενων

Πριν προχωρήσουμε στην πλήρη εγκατάσταση και παραμετροποίηση της πλατφόρμας DMOJ, είναι απαραίτητο να διαμορφωθεί σωστά το περιβάλλον του συστήματος ώστε να καλύπτει όλες τις βασικές εξαρτήσεις που απαιτεί η εφαρμογή. Η διαδικασία αυτή περιλαμβάνει την εγκατάσταση εργαλείων προγραμματισμού, βιβλιοθηκών συστήματος, καθώς και πακέτων που σχετίζονται με Python, Node.js και Redis.

Η προετοιμασία του συστήματος είναι καθοριστικής σημασίας, καθώς πολλές από αυτές τις εξαρτήσεις είναι απαραίτητες όχι μόνο για την ομαλή εκτέλεση της εφαρμογής, αλλά και για τη μεταγλώττιση πακέτων Python και την παραγωγή αρχείων CSS/JS. Η διαδικασία υλοποιείται μέσω του διαχειριστή πακέτων **apt** και των αντίστοιχων εργαλείων Node.js και npm.

```
$ sudo apt update-upgrade
```

Ανανεώνει τη λίστα διαθέσιμων πακέτων από τα αποθετήρια του Ubuntu.

```
$ sudo apt install git gcc g++ make python3-dev python3-pip libxml2-dev
libxslt1-dev zlib1g-dev gettext curl redis-server -y
```

Η παραπάνω εντολή χρησιμοποιείται για την εγκατάσταση όλων των απαραίτητων εργαλείων και βιβλιοθηκών που απαιτούνται για την ανάπτυξη και λειτουργία της πλατφόρμας DMOJ σε περιβάλλον Ubuntu. Συγκεκριμένα:

- **git** : Σύστημα ελέγχου έκδοσης για την ανάκτηση και διαχείριση πηγαίου κώδικα από αποθετήρια (όπως το GitHub).
- **gcc** : ο βασικός μεταγλωττιστής γλώσσας C, απαραίτητος για τη μεταγλώττιση πακέτων και βιβλιοθηκών.
- **g++** : ο μεταγλωττιστής C++, χρήσιμος σε περιπτώσεις όπου απαιτείται υποστήριξη C++.
- **make** : εργαλείο αυτοματισμού κατασκευής λογισμικού, που χρησιμοποιείται για τη διαχείριση μεταγλώττισης και εξαρτήσεων.
- **python3-dev** : Header αρχεία και static βιβλιοθήκες για Python 3, απαραίτητα για την μεταγλώττιση πακέτων που βασίζονται σε Python.
- **python3-pip** : ο διαχειριστής πακέτων της Python (pip), χρησιμοποιείται για την εγκατάσταση Python βιβλιοθηκών
- **libxml2-dev** : βιβλιοθήκη για την επεξεργασία αρχείων XML, απαραίτητη για Django και άλλες web εφαρμογές
- **libxslt1-dev** : εργαλεία μετατροπής XML με XSLT, συμπληρωματικά στη libxml2

- **zlib1g-dev** : βιβλιοθήκη συμπίεσης δεδομένων, συχνά απαραίτητη για πακέτα που κάνουν χρήση συμπίεσης
- **gettext** : εργαλείο διεθνοποίησης, χρησιμοποιείται για την υποστήριξη μεταφράσεων στο Django
- **curl :** Εργαλείο για την ανάκτηση δεδομένων από URLs, χρήσιμο για την εγκατάσταση Node.js και άλλων εργαλείων.
- **redis-server** : Υπηρεσία μνήμης τύπου NoSQL, χρησιμοποιείται από το Celery για τη διαχείριση ουρών εργασιών.
- **-y** : η εντολή -y απαντάει αυτόματα ναι(yes) σε τυχόν προτροπές που ενδέχεται να εμφανιστούν κατά τη διαδικασία εγκατάστασης.

$ curl -sL https://deb.nodesource.com/setup_18.x | sudo -E bash -

Κατεβάζει και εκτελεί το πρωτότυπο ρύθμισης του αποθετηρίου Node.js έκδοσης 18.x στο σύστημα, με δικαιώματα υπερχρήστη (όμως διατηρώντας τα περιβαλλοντικά στοιχεία του χρήστη).

$ sudo apt-get install nodejs –y

Εγκαθιστά την πλατφόρμα Node.js και το npm (Node Package Manager).

$ sudo npm install -g sass postcss-cli postcss autoprefixer

Εγκαθιστά εργαλεία CSS για την παραγωγή και μετατροπή εμφάνισης του DMOJ.

Δημιουργία βάσης δεδομένων

Το **DMOJ**, σε έλεγχο που έχει γίνει από την υπεύθυνη ομάδα του λειτουργεί μόνο με την **MySQL**, και πιθανόν να μην μπορεί να λειτουργήσει με κάποια άλλη εφαρμογή βάσης δεδομένων.

$ sudo apt install mariadb-server libmysqlclient-dev -y

Η παραπάνω εντολή εγκαθιστά τη βάση δεδομένων MariaDB και τις απαραίτητες επικεφαλίδες(development headers) για σύνδεση μέσω Python.

Και αφού είναι διαθέσιμη η εφαρμογή της MariaDB, με τις παρακάτω εντολές, θα δημιουργηθεί η βάση δεδομένων για την ιστοσελίδα του DMOJ, το οποίο σημαίνει δημιουργία ενός προορισμού(φακέλου) στον υπολογιστή, με όλα τα δεδομένα, όπως λογαριασμοί χρηστών και αποτελέσματα εξετάσεων, που θα αποθηκευτούν για την εξυπηρέτηση της εφαρμογής

$ sudo mariadb

Εισέρχεται στο διαχειριστικό περιβάλλον της βάσης MariaDB.

```
CREATE DATABASE dmojdb DEFAULT CHARACTER SET utf8mb4
DEFAULT COLLATE utf8mb4_general_ci;
```

Η εντολή δημιουργεί μια βάση δεδομένων με το όνομα **dmojdb**, χρησιμοποιώντας την κωδικοποίηση "utf8mb4" και την ταξινόμηση "utf8mb4_general_ci". Αυτό διασφαλίζει ότι η βάση δεδομένων μπορεί να αποθηκεύσει όλους τους χαρακτήρες Unicode και να κάνει συγκρίσεις και ταξινόμηση χωρίς να λαμβάνει υπόψη αν οι χαρακτήρες είναι κεφαλαίοι ή πεζοί.

```
GRANT ALL PRIVILEGES ON dmojdb.* TO 'dmojhost'@'localhost'
IDENTIFIED BY 'password';
```

Η εντολή αυτή παραχωρεί στον χρήστη **dmojhost** όλα τα δικαιώματα για την βάση δεδομένων **dmojdb**, επιτρέποντάς του να διαχειρίζεται όλα τα αντικείμενα της βάσης δεδομένων. Ο χρήστης dmojhost μπορεί να συνδεθεί μόνο από τον υπολογιστή στον οποίο είναι εγκατεστημένη η βάση δεδομένων (localhost) και για κωδικό πρόσβασης θα χρησιμοποιεί το **password**. ΣΗΜΕΙΩΣΗ: Ο “dmojhost” είναι καθαρά λογαριασμός της MariaDB(MySQL) και δεν έχει καμία σχέση με τον χρήστη συστήματος Linux.

```
exit;
```

Έξοδος από το περιβάλλον της βάσης MariaDB.

```
$ mariadb-tzinfo-to-sql /usr/share/zoneinfo | sudo mariadb -u root
mysql
```

Όταν εκτελείται αυτή η εντολή, το εργαλείο **mariadb-tzinfo-to-sql** διαβάζει τα αρχεία ζώνης ώρας από τον φάκελο **/usr/share/zoneinfo** και τα μετατρέπει σε SQL εντολές που μπορούν να εισαχθούν στη βάση δεδομένων MariaDB.

4.3 Διαδικασία εγκατάστασης Βήμα προς Βήμα

4.3.1 Εγκατάσταση ιστοσελίδας

Δημιουργία εικονικού περιβάλλοντος

Για την εγκατάσταση και διαχείριση της πλατφόρμας DMOJ, είναι απαραίτητη η **δημιουργία και ενεργοποίηση ενός εικονικού περιβάλλοντος Python** (virtual environment), μέσα στο οποίο θα εγκατασταθούν οι αναγκαίες βιβλιοθήκες και εξαρτήσεις. Η χρήση εικονικού περιβάλλοντος, μέσω του εργαλείου **venv** , προσφέρει απομόνωση από το υπόλοιπο σύστημα και διασφαλίζει ότι η εγκατάσταση θα είναι σταθερή και απαλλαγμένη από ενδεχόμενες συγκρούσεις πακέτων. Η προσέγγιση αυτή είναι ιδιαίτερα σημαντική σε έργα βασισμένα στην Python, καθώς επιτρέπει την ανεξάρτητη διαχείριση διαφορετικών εκδόσεων πακέτων και εξαρτήσεων.

$ sudo apt install python3.12-venv

Με αυτή την εντολή γίνεται εγκατάσταση του πακέτου python3-venv, της υπηρεσίας εικονικού περιβάλλοντος.

$ sudo python3 -m venv dmojsite

Δημιουργεί εικονικό περιβάλλον Python με όνομα dmojsite χρησιμοποιώντας δικαιώματα υπερχρήστη.

$. dmojsite/bin/activate

Ενεργοποιεί το εικονικό περιβάλλον Python που δημιουργήθηκε με όνομα dmojsite(το αποθετήριο του εικονικού περιβάλλοντος είναι ο φάκελος με όνομα dmojsite). Σε αυτό το σημείο, αν το εικονικό περιβάλλον dmojsite έχει ενεργοποιηθεί σωστά, θα εμφανιστεί μέσα σε παρένθεση, όπως φαίνεται παρακάτω, στην αρχή της εντολής του τερματικού. Το εικονικό περιβάλλον βοηθά στη διατήρηση των απαραίτητων αρχείων, ξεχωριστά, από τα υπόλοιπα πακέτα του συστήματος.

ΣΗΜΑΝΤΙΚΟ : Με τη δημιουργία, πλέον του εικονικού περιβάλλοντος, θα πρέπει να δίνεται μεγάλη προσοχή στο τερματικό σας, αν βρίσκεστε εντός εικονικού περιβάλλοντος ή εκτός, όταν πραγματοποιείται τις επόμενες εντολές. Κάποιες από αυτές πρέπει να βρίσκονται μέσα στο εικονικό περιβάλλον, δηλαδή να εμφανίζεται το **(dmojsite)**, που ενδικνύει ότι το εικονικό περιβάλλον είναι ενεργοποιημένο.

(dmojsite) $ git clone https://github.com/DMOJ/online-judge

Κατεβάζει τον πηγαίο κώδικα του DMOJ site από το GitHub, και δημιουργείται ένα αντίγραφο του αποθετηρίου της σελίδας DMOJ με όνομα online-judge. Ο φάκελος online-judge θα περιέχει πολλά χρήσιμα εργαλεία που χρειάζονται για να δημιουργηθεί η εμφάνιση της ιστοσελίδας του DMOJ.

(dmojsite) $ cd online-judge

Μεταβαίνει στον φάκελο με τον πηγαίο κώδικα, που δημιουργήθηκε από την προηγούμενη εντολή.

(dmojsite) $ git checkout v4.0.0

Εφόσον η εγκατάσταση του δικαστή θα πραγματοποιηθεί μέσω εικονικού περιβάλλοντος (virtual environment) θα πρέπει, να φροντίσετε οι εκδόσεις στο PyPi του site και του judge, να ειναι παρόμοιες. Η τελευταία εκδοση του site είναι v4.0, οπότε και του judger θα πρέπει να είναι v4.0. Ενημερωτικά, υπάρχει και διαθέσιμος judge v4.1 αλλά από τη στιγμή που το site βρίσκεται ακόμα στην v4.0, θα πρέπει να γίνει η επιλόγη παλαιότερου δικαστή για να αποφευχθούν μη συμφωνίες.

(dmojsite)~/online-judge$ sudo git submodule init

Το αποθετήριο του DMOJ περιέχει υπομονάδες που ορισμένες φορές δεν έχουν κλωνοποιηθεί πλήρως στον τοπικό υπολογιστή. Με την εντολή git submodule init θα διαβαστεί το .gitmodules του

αποθετηρίου και θα προσθέσει τις εγγραφές του submodule τοπικά, ώστε το Git να γνωρίζει ότι το αποθετήριο περιέχει submodules.

(dmojsite)~/online-judge$ sudo git submodule update

Με αυτή την εντολή πραγματοποιείται η λήψη του περιεχομένου του submodule και προσαρμόζεται στην αντίστοιχη εκδοχή που καθορίζεται από το κύριο αποθετήριο. Αν όλα έχουν πάει καλά μέχρι εκεί θα εμφανίζονται τα μηνύματα όπως φαίνεται και στη παρακάτω εικόνα.

Εικόνα 5: Μήνυμα σφάλματος

Με αυτή την εντολή πραγματοποιείται η λήψη του περιεχομένου του submodule και προσαρμόζεται στην αντίστοιχη εκδοχή που καθορίζεται από το κύριο αποθετήριο.

(dmojsite)~/online-judge$ pip3 install -r requirements.txt

Η εντολή χρησιμοποιείται για να εγκαταστήσει τις βιβλιοθήκες που ορίζονται στο αρχείο requirements.txt, το οποίο βρίσκεται στο αποθετήριο online-judge.

(dmojsite)~/online-judge$ sudo apt install pkg-config libmysqlclient-dev build-essential

(dmojsite)~/online-judge$ sudo apt install pkg-config libmariadb-dev build-essential

(dmojsite)~/online-judge$ pip3 install mysqlclient

Δίνοντας αυτές τις τρεις εντολές το σύστημα εγκαθιστά τη βιβλιοθήκη mysqlclient στο εικονικό περιβάλλον, επιτρέποντας το να επικοινωνήσει με τη βάση δεδομένων MySQL/MariaDB.

Συνοπτικά, το εικονικό περιβάλλον με όνομα **dmojsite**, είναι η ιστοσελίδα που χτίζεται, και το κλωνοποιημένο αποθετήριο με όνομα online-judge, είναι το κατάστημα που έχει όλα τα διαθέσιμα εργαλεία που χρειάζεται η ιστοσελίδα για να χτιστεί. Μέσω εντολών, το εικονικό περιβάλλον, αποκτά όλα αυτά τα διαθέσιμα εργαλεία που θα το βοηθήσουν να παράξει ένα παρόμοιο περιβάλλον με αυτό του επισημου DMOJ, έτσι ώστε να χρησιμοποιηθεί τοπικά.

Δημιουργία αρχείου ρυθμίσεων της εφαρμογής

Σε αυτό το σημείο, πρέπει να δημιουργηθεί ένα αρχείο **local_settings.py** στη τοποθεσία (home/user/online-judge/dmoj/local_settings.py) πραγματοποιώντας αντιγραφή του **ολικού κώδικα local_settings.py** της σελίδας https://github.com/DMOJ/docs/blob/master/sample_files/local_settings.py, και οι **πρώτες**

ρυθμίσεις θα αφορούν τα στοιχεία σύνδεσης με την εφαρμογή βάσης δεδομένων **MariaDB**.* Στη γραμμή 36 του αρχείου local_settings.py που αφορά τα database credentials, προσθέστε στο 'NAME' το όνομα του φακέλου της βάσης δεδομένων('dmojdb'), στο πεδίο 'USER' δώστε το όνομα χρήστη της UNIX πλατφόρμας ('alexsenu') και για 'PASSWORD' δώστε το κωδικό που δώσατε κατά τη δημιουργία της βάσης δεδομένων ('dmojdb').

**Η υλοποιήση του εγγράφου, στη παρούσα εργασία πραγματοποιήθηε μέσω Visual Studio Code. Στη καρτέλα Explorer του αριστερού πίνακα εργαλειών, ανοιξτέ το φάκελο online-judge, προσθέστε το στο απαραίτητο μονοπάτι, οπως περιγράφεται και παραπάνω, και επεξεργαστείτε από εκει τα αρχεία σας.*

Μέσω του αρχείου **local_settings.py**, **θα προστεθούν** όλες οι απαραίτητες ρυθμίσεις, αλλά το αρχείο που θα τις **πραγματοποιεί**, είναι το **settings.py**, αφού θα διαβάζει και το αρχείο **local_settings.py**, οπότε προτιμάται όλες οι **προσθέσεις** να γίνονται στο αρχείο **local_settings.py** προς αποφυγή τυχόν παραμετροποιήσεων του κύριου αρχείου ρυθμίσεων(περιέχει πολλές γραμμές κώδικα και δεν είναι ασφαλές να επεξεργάζεται συχνά), που είναι και το κύριο γρανάζι της ιστοσελίδας που θα δημιουργηθεί.

```
(dmojsite)~/online-judge$ pip install --upgrade setuptools

(dmojsite)~/online-judge$ python3 manage.py check
```

Αφού πρώτα κάνει εγκατάσταση στο **setuptools** που είναι απαραίτητο για να φορτώσει τα pkg_resources, εκτελεί έλεγχο για τη διασφάλιση ορθότητας ρυθμίσεων. Ουσιαστικά, η εντολή αυτή είναι ένα εργαλείο εντοπισμού λαθών πριν την εκτέλεση ή την ανάπτυξη της εφαρμογής σε παραγωγή, παρέχοντας μια αναφορά για προβλήματα που πρέπει να διορθωθούν πριν από την εκτέλεση ή την ανάπτυξή της.

Σύνταξη της μορφής των σελίδων

Το DMOJ χρησιμοποιεί **sass** και **autoprefixer** για τη δημιουργία των φύλλων στυλ της ιστοσελίδας. Το DMOJ συνοδεύεται από ένα σενάριο **make_style.sh** που μπορεί να εκτελεστεί για τη μεταγλώττιση και τη βελτιστοποίηση των φύλλων στυλ.

```
(dmojsite)~/online-judge$ ./make_style.sh
```

Μεταγλωττίζει και βελτιστοποιεί τα CSS της σελίδας.

Στα πλαίσια ανάπτυξης web εφαρμογών με Django, τα **static αρχεία** είναι αρχεία που **δεν αλλάζουν δυναμικά** και εξυπηρετούνται όπως είναι στον τελικό χρήστη. Τέτοια αρχεία περιλαμβάνουν:

- **CSS** αρχεία (στυλ)
- **JavaScript** αρχεία (διαδραστικότητα)
- **Εικόνες** (λογότυπα, icons, backgrounds)
- **Fonts** και άλλοι πόροι (π.χ. SVG, PDF)

Αυτά τα αρχεία είναι ζωτικής σημασίας για την **εμφάνιση, διάταξη και διαδραστικότητα** της ιστοσελίδας. Για παράδειγμα, το DMOJ χρειάζεται static αρχεία για να εμφανίζει σωστά τα κουμπιά, τις φόρμες υποβολής, τη δομή της σελίδας και πολλά ακόμη.

Πριν προχωρήστε στη παρακάτω εντολή δημιουργήστε ένα φάκελο με όνομα **static** μέσα στο ήδη υπάρχον φάκελο **tmp** που βρίσκεται στο Linux σύστημα(ή σε έναν φάκελο ο οποίος είναι σταθερός και δεν επηρεάζεται αλλά θα πρέπει να προσθεσετε και το μονοπάτι), όπου και θα προστεθούν όλα τα παραπάνω στατικά αρχεία. Έπειτα στη **γραμμή 125** του **local_settings.py** δώστε το απόλυτο μονοπάτι για το **φάκελο static** που δημιουργήσατε('/tmp/static').

(dmojsite)~/online-judge$ python3 manage.py collectstatic

Η παραπάνω εντολή συγκεντρώνει όλα τα static αρχεία(όπως εικόνες ,CSS, Javascript) και τα μεταφέρει στο STATIC_ROOT του αρχείου local_settings.py και **ετοιμάζει τα αρχεία για εξυπηρέτηση** από web server όπως το **Nginx**(παρακάτω που θα γίνει χρήση του, θα ρυθμίσουμε στο υπεύθυνο αρχείο το μονοπάτι για το φάκελο static) ή το runserver που θα γίνει για πρώτη δοκιμή λειτουργίας της ιστοσελίδας. Η συλλογή των static αρχείων με την εντολή **collectstatic** συμβάλλει στην ταχύτερη εξυπηρέτησή τους από τον web server, στη συγκέντρωσή τους σε έναν οργανωμένο φάκελο και διευκολύνει την ανανέωσή τους μετά από κάθε αλλαγή στο frontend. Επίσης κάντε comment out στη **γραμμή 128** το **STATIC_URL.**

(dmojsite)~/online-judge$ python3 manage.py compilemessages

Μεταγλωττίζει αρχεία μεταφράσεων που είναι συνήθως αρχεία **.po**, που περιέχουν τις μεταφράσεις κειμένων από την μία γλώσσα στην άλλη.

(dmojsite)~/online-judge $ python3 manage.py compilejsi18n

Μεταγλωττίζει τα αρχεία μετάφρασης για το JavaScript σε μορφή **.json** για χρήση από το Django στο frontend.

Ρύθμιση πινάκων βάσης δεδομένων

Πρέπει να δημιουργηθεί το σχήμα για τη βάση δεδομένων, καθώς σε αυτό το σημείο είναι κενή.

(dmojsite)~/online-judge$ python3 manage.py migrate

Εφαρμόζει τις αλλαγές βάσης δεδομένων σύμφωνα με τα μοντέλα Django.

Με τις επόμενες εντολές φορτώνονται ορισμένα αρχικά δεδομένα, ώστε η εγκατάστασή σας να μην είναι κενή.

(dmojsite)~/online-judge$ python3 manage.py loaddata navbar

Φορτώνει δεδομένα για την πλοήγηση της εφαρμογής.

(dmojsite)~/online-judge$ python3 manage.py loaddata language_small

Φορτώνει ένα βασικό σύνολο υποστηριζόμενων γλωσσών προγραμματισμού. Εάν επιθυμείτε όλες τις γλώσσες προγραμματισμού αλλάξτε το **language_small** της παραπάνω εντολής με **language all**.

```
(dmojsite)~/online-judge$ python3 manage.py loaddata demo
```

Εισάγει δεδομένα επίδειξης όπως προβλήματα και χρήστες.

Δημιουργία λογαριασμού για πρωταρχική σύνδεση

```
(dmojsite)~/online-judge$ python3 manage.py createsuperuser
```

Δημιουργεί λογαριασμό διαχειριστή με πρόσβαση στο πίνακα διαχείρισης. Στη **δοκιμαστική** εκδοχή δημιουργείται ένας λογαριασμός με όνομα και κωδικό **admin**. Στη συνέχεια της παραγωγής, θα μπορείτε να αλλάξετε τα στοιχεία σας μέσω της ιστοσελίδας, και αφου δεν αναφέρονται σε κάποιο αρχείο στο έργο, αποθηκεύστε τα κάπου να είναι ασφαλή.

Ρύθμιση Celery

Το DMOJ χρησιμοποιεί την υπηρεσία **Celery** για να πραγματοποιήσει δύσκολες ενέργειες, όπως τη συλλογή των υποβολών που έχουν επαναβαθμολογηθεί. Το **Redis** θα χρησιμοποιηθεί ως **μεσίτης**. Επίσης, για το επόμμενο κομμάτι θα χρειαστεί να βγείτε από το εικονικό περιβάλλον με την εντολή **deactivate** και θα ξαναμπείτε παρακάτω με την εντολή **. dmojsite/bin/activate.**

```
~/online-judge$ service redis-server start
```

Η παραπάνω εντολή, εκκινεί το διακομιστή **Redis**, τον οποίο χρειάζονται οι εργαζόμενοι της **Celery**(στο τερματικό δε θα εμφανιστεί κάποια απάντηση για την έναρξη του Redis, απλά θα περιμένει την επομένη εντολή).

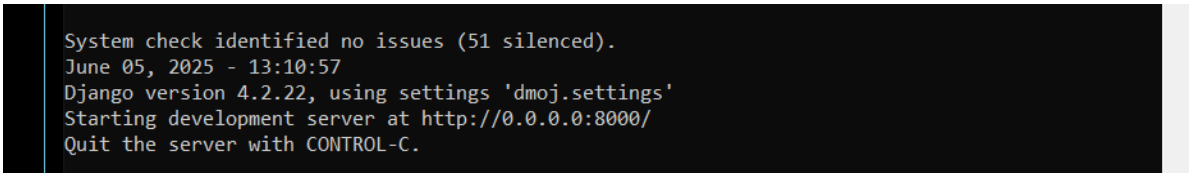
Στο αρχείο **local_settings.py** θα πρέπει να καταργηθεί το σχόλιο από το **CELERY_BROKER_URL** και **CELERY_RESULT_BACKEND** στη γραμμή **194** και **195** αντίστοιχα. Από προεπιλογή, το **Redis** ακούει στη **θύρα localhost 6379**, η οποία αντικατοπτρίζεται στο **local_settings.py**. Στη συνέχεια, θα δοκιμαστεί και η **Celery** αν λειτουργεί σωστά.

Εκτέλεση του διακομιστή

Σε αυτό το σημείο, θα δοκιμαστεί η εκτέλεση διακομιστή, και αν όλα λειτουργούν σωστά.

```
(dmojsite)~/online-judge$ python3 manage.py runserver 0.0.0.0:8000
```

Εκκινεί τον Django development server στην τοπική IP. Στο τερματικό θα εμφανιστεί η ημερομηνία, η ώρα, και που τρέχει η ιστοσελίδα στιγμιαία όπως φαίνεται και στην εικόνα. Με το σύνδεσμο **http://localhost:8000** , μπορείτε να επισκεφτείται την αρχική σελίδα του DMOJ. Με το συνδυασμό **Ctrl+C**, θα τερματίσετε το **runserver**. Να σημειωθεί ότι με το **runserver** μπορεί να γίνουν όλες οι απαραίτητες δοκιμές, τοπικά, για την πλατφόρμα του DMOJ, όπως η σύνδεση με το δικαστή.



*Εικόνα 6: Εμφάνιση μηνύματος ενεργοποιήσης της ιστοσελίδας στο τερματικό από το **runserver***

(dmojsite)~/online-judge$ python3 manage.py runbridged

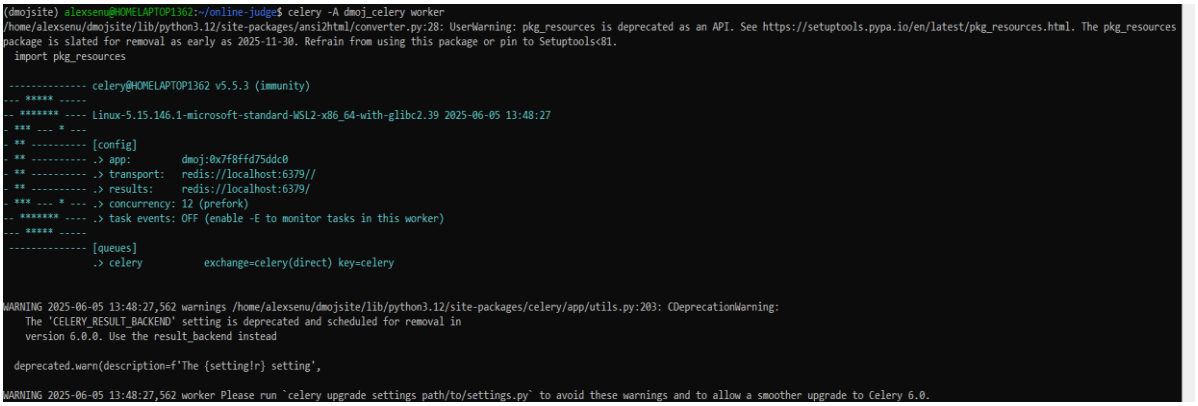
Εκκινεί προσωρινά την υπηρεσία bridged, η οποία επιτρέπει τη διασύνδεση του DMOJ site με το σύστημα που εκτελεί τις υποβολές (δικαστής). Αν δεν εμφανιστούν σφάλματα για περίπου **10 δευτερόλεπτα**, τότε η υπηρεσία λειτουργεί σωστά. Τερματισμός με **Ctrl+C**.

(dmojsite)~/online-judge$ pip install redis

(dmojsite)~/online-judge$ celery -A dmoj_celery worker

Αφού κάνει την απαραίτητη εγκατάσταση του redis που λειτουργεί σαν “οδηγός” που επιτρέπει σε Python εφαρμογές όπως το Celery, να στέλνουν εντολές προς το redis-server για την ομαλή λειτουργία της ιστοσελίδας, εκκινεί τον worker του Celery, ο οποίος διαχειρίζεται χρονικά απαιτητικές διεργασίες. Αν λειτουργεί χωρίς σφάλματα θα εμφανιστεί κάτι όπως στη παρακάτω εικόνα και τερματίζεται με Ctrl+C.

Αν επιθυμείτε, μπορείτε να επεργαστείτε τη πλατφόρμα όπως έχει, εφόσον λειτουργούν όλα χωρίς προβλήματα.



Εικόνα 7: Μήνυμα επιβεβαίωσης λειτουργίας Celery

Ρύθμιση του uWSGI

Το runserver δεν είναι ασφαλές και δεν προορίζεται για παραγωγικούς σκοπούς και δε θα πρέπει να χρησιμοποιείται, πέρα από τη δοκιμή λειτουργίας. Οπότε αφού το στάδιο λειτουργίας πέρασε με επιτυχία, για την εξυπηρέτηση της ιστοσελίδας θα γίνει εγκατάσταση του **Uwsgi** και **nginx**, χρησιμοποιώντας τον επόπτη **supervisord**, για να κρατάει σε λειτουργία το **site** και **bridged**. Πρώτα, θα πρέπει να δημιουργηθεί αρχείο **uswgi.ini** στον φάκελο του κλωνοποιημένου αποθετηρίου και να αντιγραφεί το **https://github.com/DMOJ/docs/blob/master/sample_files/uwsgi.ini** μέσα στο αρχείο αλλάζοντας τα **chdir**(γραμμή 12) και **pythonpath**(γραμμή 13) με το απόλυτο μονοπάτι για το

κλωνοποιημένο αποθετήριο **/home/user(alexsenu)/online-judge** και το **virtualenv**(γραμμή 14) με το μονοπάτι του εικονικού περιβάλλοντος **home/user(alexsenu)/dmojsite,** και κάντε αποθήκευση αρχείου. Αν εμφανιστεί κάποιο πρόβλημα με τις παρακάτω εντολές θα χρειαστεί να διαγραφεί χειροκίνητα το αρχείο **tmp/dmoj-site.sock** γιατι προκαλει προβλημα στη φόρτωση των εργατών, αφου προσπαθεί να το διαγράψει το σύστημα με τη δεύτερη εντολή, αλλα αποτυγχάνει.

```
(dmojsite)~/online-judge$ pip3 install uwsgi
```

Εγκαθιστά το uWSGI, υπεύθυνο για την επικοινωνία του web server (Nginx) με την εφαρμογή Django.

```
(dmojsite)~/online-judge$ uwsgi --ini uwsgi.ini
```

Εκκινεί τον uWSGI server με παραμέτρους από το αρχείο uwsgi.ini. Αν λέει, **workers are spawned**, λειτουργεί σωστά. Τερματισμός με Ctrl+C.

Ρύθμιση επόπτη

```
~/online-judge$ sudo apt install supervisor
```

Εγκαθιστά το εργαλείο **supervisor**, που κρατά σε λειτουργία υπηρεσίες όπως uWSGI, bridged, celery. Έπειτα, **τρεις κινήσεις** θα πρέπει να πραγματοποιηθούν στο **φάκελο** του λειτουργικού συστήματος Ubuntu με διαδρομή **/etc/supervisor/conf.d/** και είναι η δημιουργία των αρχείων **site.conf**, **bridged.conf** και **celery.conf.** Να δημιουργηθεί αρχείο με όνομα **site.conf** που θα περιέχει τα στοιχεία του **https://github.com/DMOJ/docs/blob/master/sample_files/site.conf**, αρχείο με όνομα **bridged.conf** που θα περιέχει τα στοιχεία του **https://github.com/DMOJ/docs/blob/master/sample_files/bridged.conf** και αρχείο με όνομα **celery.conf** που θα περιέχει τα στοιχεία του **https://github.com/DMOJ/docs/blob/master/sample_files/celery.conf**(αν χρησιμοποιείτε WSL, με τις εντολές **touch site.conf, touch bridged.cond** και **touch celey.conf δημιουργείται τα απαραίτητα αρχεία και sudo chmod 660** για να μην υπάρξουν θέματα δικαιωμάτων κατά την εγγραφή τους στο Visual Studio Code. Αφού έχετε μετακινηθεί στο απαραίτητο μονοπάτι. Όπου **<path to virtualenv>** αντικαταστήστε με το φάκελο προορισμού **εικονικού περιβάλλοντος(/home/user(alexsenu)/dmojsite/)** και όπου **<path to site>** αντικαταστήστε με το φάκελο προορισμού του **αποθετηρίου φακέλου** που δημιουργήθηκε στο σύστημα**(home/user(alexsenu)/online-judge/).** Στα αρχεία **celery.conf** και **bridged.conf,** εντοπίστε το **user** και το **<user to run under>** αντικαταστήστε με το όνομα λογαριασμό χρήστη(alexsenu) και στο **group** το **<user to run under>** με το όνομα του λειτουργικού συστήματος(ubuntu).

Στη συνέχεια, φορτώστε ξανά τον επόπτη (supervisor) και ελέγξτε ότι το **site**, **bridged** και **celery** τρέχουν.

```
$ sudo supervisorctl update
```

Ανανεώνει τη λίστα υπηρεσιών ώστε να αναγνωριστούν οι νέες.

Η παρακάτω εντολή εμφανίζει την κατάσταση όλων των υπηρεσιών του Supervisor. Αν το **site**, **bridged** και **celery** τρέχουν κανονικά(running). Φροντίστε να έχετε σώσει τα αρχεία που δημιουργήσατε και αν κάτι δεν λειτουργεί κανονικά (site FATAL) δοκιμάστε να κάνετε επανεκκίνηση τον υπολογιστή, καθώς κάποιες φορές χρειάζεται επανεκκίνηση του συστήματος για να πιάσουν τόπο οι νέες ρυθμίσεις

$ sudo supervisorctl status

.

Ρύθμιση του nginx

$ sudo apt install nginx

Εγκαθιστά τον web server **Nginx**. Έπειτα, πρέπει να δημιουργηθεί αρχείο με όνομα **nginx.conf** στο φάκελο **/etc/nginx/conf.d/** με τα στοιχεία του **https://github.com/DMOJ/docs/blob/master/sample_files/nginx.conf** και να γίνουν οι κατάλληλες επεξεργασίες στο αρχείο όπου απαιτείται. Για παράδειγμα στη **γραμμή 21 και 25** θα γίνει αντικατάσταση του <site code path> με **(/home/user(alexsenu)/online-judge)** και στη διαδικασία /static και στη **γραμμή 38** του αρχείου **nginx.conf,** θα χρειαστεί να προσθέσετε το μονοπάτι **/tmp/static.** Επίσης στη γραμμή **10 του αρχείου server_name** θα πρέπει να δοθεί η τιμή που χρησιμοποιείται για την επίσκεψη της ιστοσελίδας και εφόσον η δοκιμή γίνεται τοπικά, το όνομα του σέρβερ μπορεί να είναι **server_name 127.0.0.1;,** αφαιρώντας το **<hostname>**.

$ sudo nginx -t

Ελέγχει αν η σύνταξη του αρχείου ρυθμίσεων είναι σωστή.

$ sudo service nginx reload

Αν κάτι δε πάει καλά δοκιμάστε επανεκκίνηση το Nginx για να φορτωθούν οι νέες ρυθμίσεις.

$ http://127.0.0.1/

Επισκέπτεστε την τοπική σελίδα για να επιβεβαιώσετε ότι η Nginx σελίδα λειτουργεί όπως στη παρακάτω εικόνα.

Εικόνα 8: Επιβεβαίωση λειτουργίας Nginx

Διαμόρφωση διακομιστή συμβάντων

Για την υποστήριξη ζωντανής ενημέρωσης των υποβολών (real-time feedback), το DMOJ χρησιμοποιεί έναν **Event Server** βασισμένο σε **WebSocket** τεχνολογία. Ο Event Server διαχειρίζεται αιτήματα μεταξύ του χρήστη και της εφαρμογής για γεγονότα όπως νέα υποβολή, ενημέρωση κατάστασης, και άλλα.

Η ρύθμιση του απαιτεί τη δημιουργία ενός αρχείου παραμετροποίησης με την ονομασία **config.js** και την εγκατάσταση συγκεκριμένων εξαρτήσεων τόσο σε Node.js όσο και σε Python. Η δημιουργία του αρχείου **websocket/config.js** γίνεται με την ακόλουθη εντολή:

```
(dmojsite) $ cat > websocket/config.js
```

Το περιεχόμενο που εισάγετε είναι:

```
module.exports = {
  get_host: '127.0.0.1',
  get_port: 15100,
  post_host: '127.0.0.1',
  post_port: 15101,
  http_host: '127.0.0.1',
  http_port: 15102,
  long_poll_timeout: 29000,
};
```

Το **<event server websocket port>** στο αρχείο **nginx.conf** και στη διαδικασία **location /event/** θα αντικατασταθεί με τον αριθμό θύρας που βρίσκεται στο **get_port, 15100** και θα πρέπει να τους **καταργηθούν** σχόλια από το **location /event** από τη γραμμή 65 εως 71**.** Το **<event server http port>** στο αρχείο **nginx.conf** και στη διαδικασία **location /channels/** θα αντικατασταθεί με τον

αριθμό που βρίσκεται στο **http_port, 15102** και θα πρέπει να του **καταργηθούν** όλα τα σχόλια από τη γραμμή 73 εως 77**.** Στο **local_settings.py** στη **γραμμή 173** θα καταργηθεί το σχόλιο στο **EVENT_DAEMON_POST = 'ws://127.0.0.1:15101/'** το οποίο βλέπει στη θύρα **15101** του **post_port.** Επίσης στη **γραμμή 183** κάντε comment out και αντικαταστήστε το **<your domain> με 127.0.0.1(EVENT_DAEMON_GET = 'ws://127.0.0.1/event/')** και κάντε comment out τη **γραμμή 185,** χωρίς να αλλάξετε κάτι**(EVENT_DAEMON_POLL = ' /channels/').**

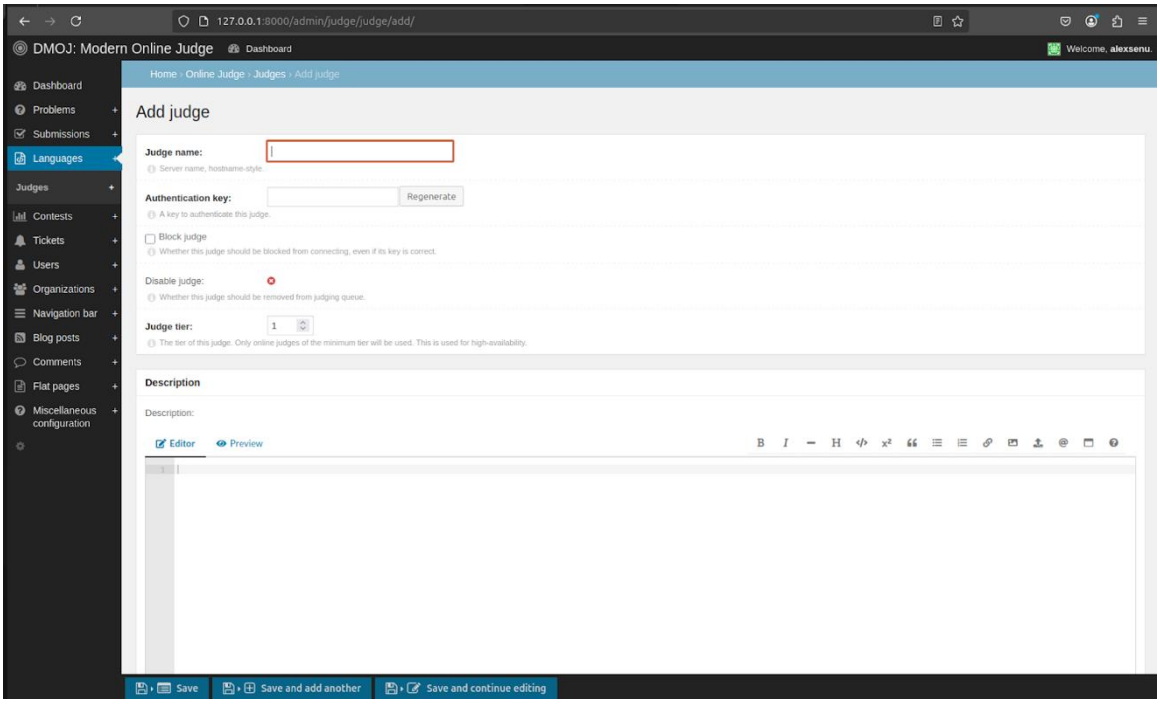
4.3.2 Εγκατάσταση δικαστή

Ο δικαστής προορίζεται για εγκατάσταση αποκλειστικά σε συστήματα βασισμένα σε Linux. Υποτίθεται ότι έχουν ακολουθηθεί οι οδηγίες εγκατάστασης της ιστοσελίδας και ότι εκτελείται μια παρουσία γέφυρας(bridged). Και ακολουθούν δύο διαδικασίες εγκατάστασεις.

1.ΕΓΚΑΤΑΣΤΑΣΗ ΑΠΟ ΤΗ ΜΕΡΙΑ ΤΗΣ ΙΣΤΟΣΕΛΙΔΑΣ

Η ιστοσελίδα πρέπει να γνωρίζει ότι υπάρχει διαθέσιμος ένας ή περισσότεροι "δικαστές" (judgers).

Μέσα στην ιστοσελίδα, στη τοποθεσία **admin**, εμφανίζονται διάφορες επιλογές στο κεντρικο πλαίσιο, όπου στη καρτέλα **ONLINE JUDGE** θα βρείτε την επιλογή **Judges** και θα πατήσετε **+ADD**. Θα ονομάσετε το δικαστή και ο μοναδικός κωδικός για το δικαστή, τον οποίο κωδικό μπορεί η ιστοσελίδα να το δημιουργήσει μέσω του κουμπιού Regenerate για να είναι πολύπλοκος και να παρέχει μεγαλύτερη ασφάλεια, περάστε το σε ένα **text document** για να το χρησιμοποιήστε παρακάτω. Μόλις προσθέσετε τις πληροφορίες κάνετε **Save** με το κουμπί στο κάτω μέρος της πλατφόρμας.



Εικόνα 9: Τρόπος πρόσθεσης δικαστή στη πλατφόρμα

Έπειτα στο αρχείο local_settings.py, στο **BRIDGED_JUDGE_ADDRESS [('0.0.0.0', 9999)]** δείχνει την διεύθυνση στην οποία θα συνδεθεί ο δικαστής. Το 0.0.0.0 δείχνει οτί ακούει τοπικά σε οποιαδήποτε συσκευή και το 9999 είναι η θύρα που χρησιμοποιείται. Εάν η σύνδεση δικαστή γίνεται από άλλη συσκευή τότε θα πρέπει να δοθεί η πραγματική IP διεύθυνση της συσκευής που τρέχει το δικαστή. Επίσης για αποφυγή τυχόν προβλημάτων να γίνει έλεγχος διαθεσιμότητας της θύρας.

Τέλος, ελέγχουμε αν η BRIDGED λειτουργεί με την εντολή:

$ supervisorctl status

και η γραμμή: **bridged RUNNING pid <pid>, uptime <uptime> ,** επιβεβαιώνει ότι λειτουργεί, αλλιώς εκτελείτε **sudo supervisorctl restart bridged** για να δοκιμάσετε να επανεκκινήσετε τη γέφυρα που θα συνδέσει το δικαστή με την ιστοσελίδα. Αν τελικά δείτε **bridged RUNNING** συνεχίστε με την εγκατάσταση του δικαστή.

2. Εγκατάσταση δικαστή μέσω PyPi

Η εγκατάσταση δικαστή μέσω PyPi, περιλαμβάνει την δημιουργία ενός εικονικού περιβάλλοντος το οποίο θα περιέχει τον δικαστή(**dmoj-judge**) που θα συνδέεται με το εικονικό περιβάλλον της ιστοσελίδας(**dmojsite**), καθώς και τη ενημέρωση της μεταβλητής **DMOJ_PROBLEM_DATA_ROOT** στο αρχείο **settings.py,** με το μονοπάτι που θα αποθηκεύεται τα test cases για τα προβλήματα σας που θα δημιουργείτε από τη πλατφόρμα.

Σε ένα τερματικό θα λειτουργει το εικονικό περιβάλλον της ιστοσελίδας και σε ένα άλλο τερματικό θα λειτουργεί το εικονικό περιβάλλον της ιστοσελίδας, και η σύνδεση τους θα πετυχένεται μέσω των ρυθμίσεων του **local_settings.py,** στη γραμή **BRIDGED_JUDGE_ADDRESS = [('0.0.0.0' , 9999)]**. Σε

αυτή τη γραμμή περιέχεται το IP αλλά και το port μέσω του οποίου θα γίνεται η σύνδεση του δικαστή με την ιστοσελίδα.

Πρώτα κάνετε εγκατάσταση τις απαραίτητες βιβλιοθήκες για το χτίσιμο του δικαστή. Θα γίνει προσθήκη deadsnakes PPA(χρείαζεται η Python3.11 για λόγους συμβατότητας με το δικαστή) που θεωρείται η πιο εύκολη και ασφαλής λύση για Python πολλων εκδόσεων.

$ apt install python3-dev python3-pip build-essential libseccomp-dev

$ sudo apt install software-properties-common –y

$ sudo add-apt-repository ppa:deadsnakes/ppa

$ sudo apt update

$ sudo apt install python3.11 python3.11-dev python3.11-venv

Έπειτα, δημιουργήστε και ενεργοποιήστε ένα εικονικό περιβάλλον στο οποίο θα κάνετε την εγκατάσταση του DMOJ judge(ΠΡΟΣΟΧΗ: η προεπιλεγμένη εντολή εγκατάστασης του δικαστή απο pypi είναι pip3 install dmoj, αλλά αυτό θα εγκαταστήσει νεοτερη έκδοση και στο παρόν έργο χρειάζεται η v4.0, η **pip install dmoj==4.0.0** , που ειναι παλαιότερη αλλά είναι συμβατή με την ιστοσελίδα).

$ python3.11 -m venv dmoj-judge

$. dmoj-judge/bin/activate

(dmoj-judge) $ pip install dmoj==4.0.0

Στη συνέχεια, θα τρέξετε το αρχείο dmoj-autoconf το οποίο θα παράγει μέσα στο τερματικό το "**Configuration result"** από το οποίο θα πάρετε το runtime και θα το προσαρμόσετε στο **judge.yml** αρχείο(*https://github.com/DMOJ/docs/blob/master/sample_files/judge_conf.yml).* Τα runtimes είναι οι γλώσσες προγραμματισμού που θα μπορεί ο δικαστής να βαθμολογήσει. Για χάρη της δημιουργίας του δικαστή χρησιμοποιήθηκε η Python3.11, μιας και στη Python3.12 δεν υπήρχαν συγκεκριμένα εργαλεία που χρειάζονται στο χτίσιμο του δικαστή. Οπότε στο configuration result, θα πρέπει να δηλώθεί η Python3.12 χειροκίνητα στο **judge.yml**, προσθετωντας την εξης γραμμή: **python3: /usr/bin/python3.12,** αφού έχει ήδη εγκατασταθεί απο προηγούμενες εντολές στο έργο. Το judge.yml μπορείτε να το αποθηκεύσετε στο φάκελο **online-judge.**

Εικόνα 10: Μορφή αρχείου judge.yml

Παρακάτω θα γίνει δοκιμή του δικαστή με τα runtimes που του δόθηκαν στο judge.yml. Αυτό σημαίνει ότι στο τερματικό θα πρέπει να βρίσκεστε στο φάκελο που περίεχει το judge.yml. Με Ctrl+C τερματίζετε.

```
(dmoj-judge) ~/online-judge$ dmoj-cli -c judge.yml
```

Αφού δεν έχει αλλαχτεί κάτι στο local_settings.py η παρακάτω εντολή πρέπει να λειτουργήσει(dmoj -c judge.yml -p "$PORT" "$IP" είναι η μη τροποποιημένη εντολή με όλες τις πληροφορίες που χρειάζεται) για ενεργοποιήση του δικαστή.

```
dmoj -c judge.yml 0.0.0.0
```

3. Εγκατάσταση δικαστή στο Docker

Για να λειτουργήσει ο δικαστής θα χρειαστεί να γίνει η εγκατάσταση του Docker. Σε αυτό τον οδηγό εγκατάστασης θα χρησιμοποιηθεί το **αποθετήριο apt** και σε περίπτωση επιθυμίας διαφορετικού τρόπου μπορούν να βρεθούν οι λεπτομέρειες στην επίσημη σελίδα της **https://docs.docker.com/engine/install/ubuntu/**.

Αρχικά, πρεέπει να γίνει η αποεγκατάσταση των ακόλουθων πακέτων, για να μην υπάρξει κάποια συμπλοκή με τα νέα πακέτα που συνοδευονται με το Docker. **Docker.io**, **docker-compose**, **docker-compose-v2**, **docker-doc** και **podman-docker** είναι τα πακέτα και επιπλέον η μηχανή Docker βασίζεται στο **containerd** και **runc**, το οποίο σημαίνει ότι η εγκατάσταση της συνοδευεται με αυτά τα δύο σε ένα πακέτο **containerd.io**. Με την παρακάτω εντολή γίνονται όλες οι απαραίτητες αποεγκαταστάσεις για να αποφευχθούν τυχόν συγκρούσεις.

```
$ for pkg in docker.io docker-doc docker-compose docker-compose-v2
podman-docker containerd runc ;  do sudo apt-get remove $pkg; done
```

apt-get μπορεί να αναφέρει ότι κανένα από αυτά τα πακέτα είχαν εγκατασταθεί και αν επιθυμείτε η καθαρή και ομαλή εγκατάσταση οι παρακάτω εντολές διαγράφουν κάποια αρχεία που χρειάζονται

τις εντολές καθώς δεν διαγράφονται αυτόματα.

```
$ sudo apt-get purge docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin docker-ce-rootless-extras

$sudo rm -rf /var/lib/docker

$sudo rm -rf /var/lib/containerd

$sudo rm /etc/apt/sources.list.d/docker.list

$sudo rm /etc/apt/keyrings/docker.asc
```

Και αφού δεν υπάρχει κάποιο παλιό αρχείο Docker, σειρά έχει η προετοιμασία του **αποθετηρίου apt Docker.** Έπειτα θα εγκατασταθεί και θα ενημερωθεί το Docker από το αποθετήριο. Οι παρακάτω εντολές δημιουργούν αποθετήριο για τα keyrings αν δεν υπάρχουν ήδη, κάνουν εγκατάσταση και αποθηκεύουν το GPG κλειδί του Docker.

```
$ sudo apt-get update

$sudo apt-get install ca-certificates curl

$sudo install -m 0755 -d /etc/apt/keyrings

$sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o /etc/apt/keyrings/docker.asc

$sudo chmod a+r /etc/apt/keyrings/docker.asc
```

Με την παρακάτω εντολή προστίθεται το αποθετήριο που δημιουργήθηκε στις πηγές **Apt**:

```
$ echo "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc] https://download.docker.com/linux/ubuntu $(. /etc/os-release && echo "${UBUNTU_CODENAME:-$VERSION_CODENAME}") stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
```

```
$ sudo apt-get update
```

Και πλέον μπορεί να γίνει το τελικό βήμα εγκατάστασης της τελευταίας έκδοσης του **Docker** και δοκιμής της ομαλής λειτουργίας της μηχανής με την παρακάτω εντολή:

$ sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin

$ sudo docker run hello-world

Η τελευταία εντολή κατεβάζει μια δοκιμαστική εικόνα που θα τρέξει στο τερματικό και θα εμφανίσει μήνυμα επιβεβαίωσης πως η εγκατάσταση λειτουργεί κανονικά.

Αφού έγινε η εγκατάσταση του Docker σειρά έχει η δοκιμή λειτουργίας δικαστή από το Docker. Το DMOJ έχει αποθετήριο με όλα τα runtimes που υποστηρίζουν το δικαστή και μπορούν να βρεθούν στη σελίδα https://github.com/DMOJ/runtimes-docker. Τα runtimes χωρίζονται σε τρεις κατηγορίες, όπου η Tier 1 υποστηρίζει Python 2/3, C/C++ (GCC μόνο), Java 8, και Pascal. Η Tier 3 υποστηρίζει όλες τις γλώσσες και η Tier 2 είναι κάπου στη μέση. Στη σελίδα μπορεί να βρεθεί το **Dockerfile** με όλες τις λεπτομέρειες για τα Tiers. Όλες οι εικόνες ενημερώνονται κάθε εβδομάδα για να λειτουργούν με βάση τα τελευταία δεδομένα σε κάθε γλώσσα προγραμματισμού. Παρακάτω θα δοθεί εκτέλεση μιας γραμμής εντολών, για να γίνει η δοκιμασία της λειτουργίας δικαστή μέσω **CLI instance** πριν συνδεθεί με την ιστοσελίδα. Αναμένει τα προβλήματα να τοποθετούνται στη διεύθυνση **/mnt/problems/** του διαχειριστή και τα αρχεία που αφορούν θέματα προσαρμογής δικαστή στο φάκελο **/mnt/problems/judge.yml**(να δημιουργηθούν οι αντίστοιχοι φάκελοι στα **έγγραφα του συστήματος και όχι στο /judge/ φάκελο** και αν υπάρχουν αντίστοιχοι παλιοί φάκελοι να διαγραφούν γιατί θα πρέπει να έχετε τα πλήρη δικαιώματα στο φάκελο).

$ git clone --recursive https://github.com/DMOJ/judge.git

$ sudo systemctl enable docker

$sudo systemctl start docker

$ cd judge/.docker

$ make judge-tier1

$ docker run \-v /mnt/problem> -v /mnt/problems:/problems \--cap-add=SYS_PTRACE \dmoj/judge-tier1:latest \cli -c /problems/judge.ymlE

Με το πέρας της προηγούμενης εντολής docker run... το τερματικό δείχνει ότι γίνεται self-testing σε όλες τις γλώσσες που περιέχει η tier 1 και επιτυγχάνουν όλες εκτός από την Objective C, η οποία στέλνει και traceback, αλλά δεν είναι ανησυχητικό, καθώς όλες οι αλλες γλώσσες δοκιμάστηκαν με επιτυχία.

Έπειτα, θα φτιαχτεί μια εικόνα δικαστή Tier 1 στη παρακάτω ακολουθία εντολών αφού πρώτα με την εντολη ip addr στο τερματικό βρούμε το όνομα διεπαφής του συστήματος, και σε αυτή τη περίπτωση είναι e. Στο παρακάτω κείμενο πρέπει να αλλαχτεί το "$PORT" και το "$IP" με τη θύρα στην οποία θα ακούει ο δικαστής 9998(από εδώ δέχεται εντολές ο δικαστής το bridged όμως που είναι συνδεδεμένη η σελίδα είναι στο '9999') και τη διεύθυνση 192.168.1.45 στην οποία ακούει ο δικαστής. Το "$JUDGE_NAME" και "$JUDGE_AUTHENTICATION_KEY" θα πάρει το όνομα και το κωδικό που δόθηκε στη σελίδα /admin/judge/judge/add αντίστοιχα. Το παρακάτω κείμενο είναι το template για τη δημιουργία εικόνας Tier 1. Μπορείτε να βρείτε τα scripts με την απαραίτητη μορφή τους στη θέση **https://docs.dmoj.ca/#/judge/setting_up_a_judge?id=with-docker**.

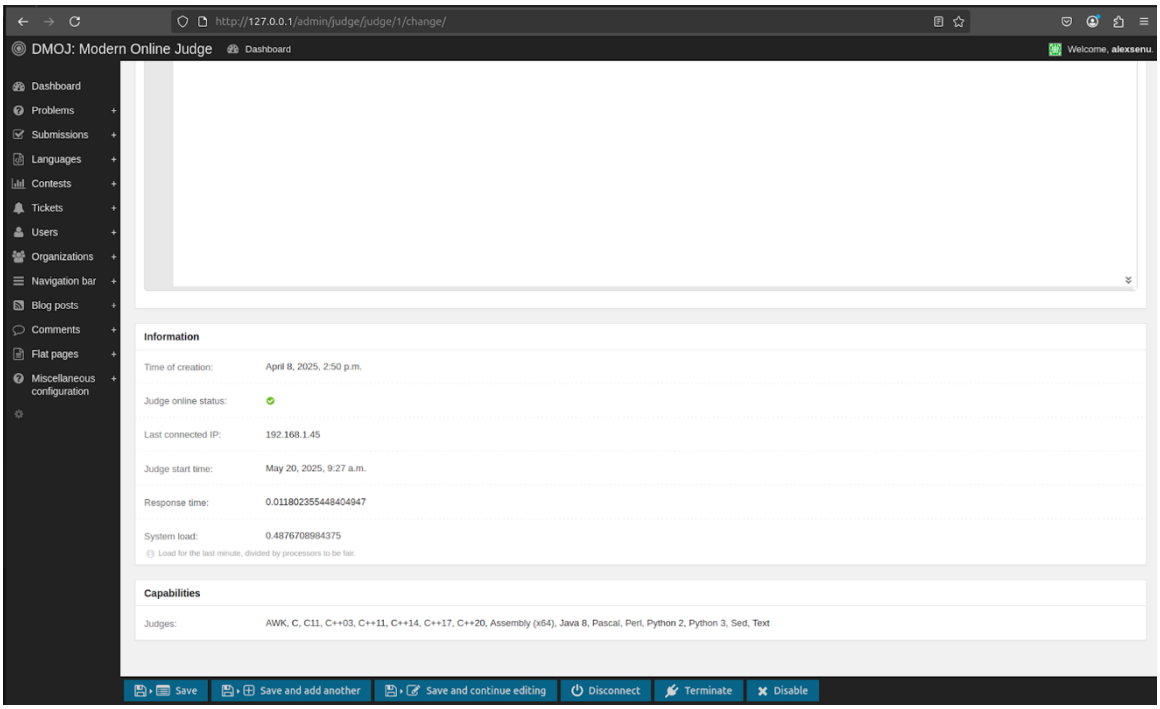
Εικόνα 11: Script δημιουργίας Tier 1 judger

Παρακάτω, είναι η τροποποιημένη έκδοση για το συγκεκριμένο έργο.

$ docker run \ --name judge \ -p "$(ip addr show dev eth0 | perl -ne 'm@inet(.*)/.*@ and print$1 and exit')":9998:9998 \ -v /mnt/problems:/problems\ --cap-add=SYS_PTRACE \ -d \ --restart=always \ dmoj/judge-tier1:latest \ run -p 9998 -c /problems/judge.yml \ 0.0.0.0 judge YgOr4h4zwALPObllwUujSIHwAWrsB/YNOwd1DRUzIcPLP57WRhMmbVZIiWZvlEWQ3HlmpvN0ltZZb1m7Ahwct2dKo85Zh3Mj1XVx

Για να επιβεβαιώσουμε το δικαστή θα δημιουργηθεί ένα .**yaml** αρχείο που περιέχει τους χρόνους εκτέλεσης, τους φακέλους προβλημάτων και άλλες πληροφορίες το οποίο μπορείτε να το τοποθετήστε στην ίδια τοποθεσία με το local_settings.py. Ένα δείγμα αρχείου διαμόρφωσης είναι διαθέσιμο στη σελίδα https://github.com/DMOJ/docs/blob/master/sample_files/judge_conf.yml στο οποίο δίνεται στο **id** ως τιμή το όνομα του δικαστή(judge01 στο συγκεκριμένο έργο) και στο **key** το κλειδί.

Για να δούμε αν θα λειτουργήσει ο δικαστής στην ιστοσελίδα θα πρέπει να ανοίξετε την εφαρμογη Docker και θα μπείτε στη καρτέλα images για να βρείτε την εικόνα με το όνομα δικαστή που δημιουργήθηκε(dmoj/judge-tier1) από την παραπάνω εντολή και θα βρούμε το σημάδι που λέει RUN και αφού ολοκληρωθεί, θα δημιουργηθεί ένα container, το οποίο βρίσκεται στη καρτέλα 'Containers'. Με αυτο το τρόπο ολοκληρώνεται η σύνδεση δικαστή με την ιστοσελίδα και μπορείτε να το επιβεβαιώσετε, ανοίγοντας τη καρτέλα Languages/Judges στην ιστοσελίδα και εισέρχοντας στο όνομα του δικαστή σας για να δείτε τη παρακάτω εικόνα που λέει Judge online status, με πράσινο τσεκ, και ο δικαστής είναι έτοιμος για να αξιολογήσει υποβολές(όπως φαίνεται και στη παρακάτω εικόνα).



Εικόνα 12: Επιβεβαίωση σύνδεσης δικαστή με την ιστοσελίδα

4.4 Ρυθμίσεις παραμετροποίησης

Διόρθωση σφάλματος με την είκονα DMOJ μέσω πλατφόρμας

Στη καρτέλα **Navigation bar-> Sites,** αλλάξτε τη διεύθυνση σε localhost:8000 για να σας εμφανίζεται η αρχική σελίδα με το πάτημα της εικόνας DMOJ.

Πρόσθεση επιπλέον δικαστών

Σε περίπτωση επιθυμίας αύξησης αριθμού των δικαστών, μπορεί να επιτευχθεί. Αν χρησιμοποιείται το PyPi για την εγκατάσταση των δικαστών, ανοίγειται τον ανάλογο αριθμό τερματικών με τον αριθμό των δικαστών που χρειάζεστε, φτιάχνεται ξεχωριστό αρχείο judge.yml για καθε δικαστή(judge1.yml, judge2.yml, judge3.yml κ.ο.) με το δικό του ξεχωριστό όνομα, κωδικό αλλά και runtimes. Μπορείτε να χρησιμοποιήστε το ίδιο port και ip για όλους τους δικαστές, απλά προσαρμόστε το judge.yml. Μέσω Docker εκκινείται το judge image, δημιουργώντας κάθε φορά ένα καινούργιο container με το δικό του ξεχωριστό judger.

Ρύθμιση Email της ιστοσελίδας

Στη γραμμή ~86 του αρχείου local_settings.py βρίσκεται έτοιμο template, με το οποίο μπορείτε να βάλετε λογαριασμό Gmail να λειτουργεί ως αλληλογράφος με χρήστες της ιστοσελίδας. Αν έχετε δοκιμάσει να κάνετε εγγραφή χρήστη στην ιστοσελίδα και η εγγραφή δεν ολοκληρώνεται, είναι επειδή δεν υπάρχει ο ανταποκριτής που στέλνει στα email των χρηστών, τα μηνύματα επιβεβαίωσης/ενεργοποίησης λογαριασμού. Κάντε uncomment απο το EMAIL_BACKEND μέχρι το EMAIL_PORT. Θα χρειαστεί να συμπληρώσετε το EMAIL_HOST_USER με το λογαριασμό email, που επιθυμείτε να είναι υπεύθυνο για την ιστοσελίδα (dmojexample@gmail.com) και στο EMAIL_HOST_PASSWORD τον κωδικό που θα λάβετε μετά την ενεργοποίηση του ελέγχου ταυτότητας δύο παραγόντων ή επαλήθευση σε 2 βήματα(2FA-Two Factor Authentication) που θα πραγματοποιήστε μέσω της καρτέλας 'Ασφάλεια' στο λογαριασμό του email σας. Με την ενεργοποίηση της θα εμφανιστεί και η επιλογή 'App passwords' στην οποία θα δώσετε το όνομα της ιστοσελίδας στο 'App name' και με το κουμπί 'create' θα δημιουργήσετε έναν κωδικό τον οποίο θα το δώσετε ως τιμή στη μεταβλητή EMAIL_HOST_PASSWORD.

Εγκατάσταση ADD-ON reCAPTCHA

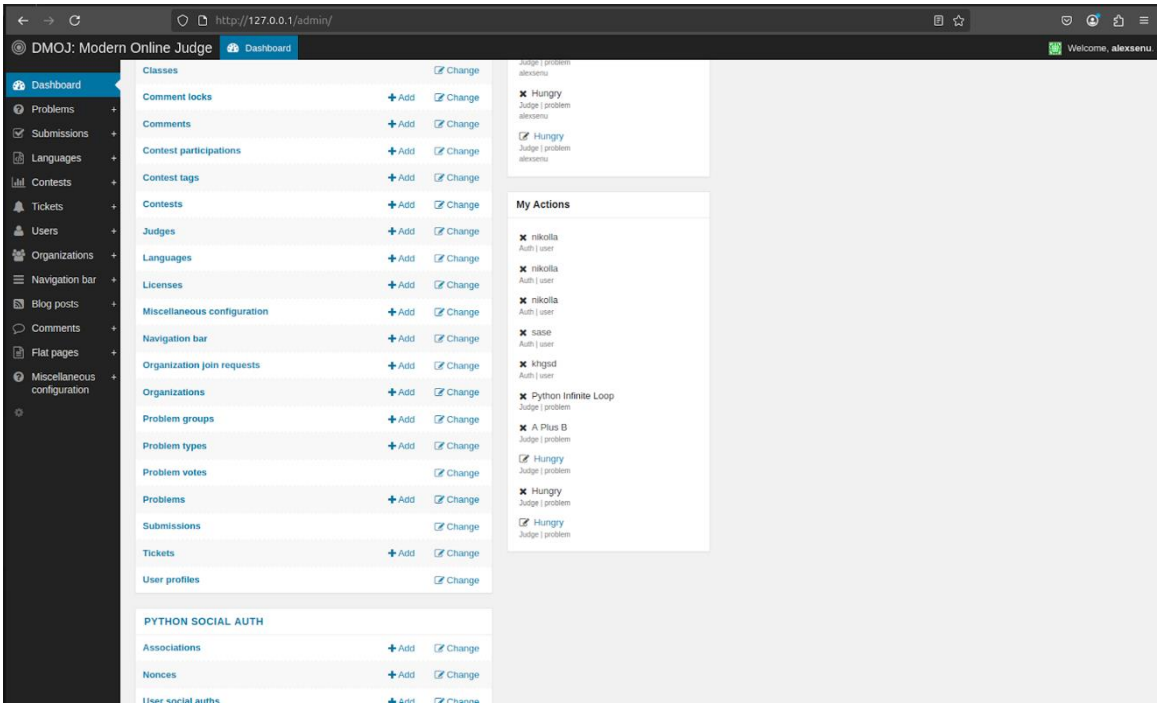
Η εφαρμογή reCaptcha της Google προσφέρει προστασία από αυτοματοποιημένες επιθέσεις(spam) και βοηθά στην ασφάλεια της ιστοσελίδας. Όταν κάποιος χρήστης πατήσει το κουμπί **Εγγραφής Λογαριασμού**, η reCaptcha ελέγχει αν πρόκειται για άνθρωπο ή bot. Για να μπει σε λειτουργία αυτός ο μηχανισμός στην ιστοσελίδα DMOJ θα επισκεφτείτε την διεύθυνση **https://www.google.com/recaptcha/admin/create** για να δημιοργήσετε λογαριασμό **reCaptcha(v2)**, στη περιοχή **'Domains'** δώστε τη τιμη **127.0.0.1** και ένα **Label** με όνομα **DMOJ** και θα κάνετε **Submit.** Μετά θα εμφανιστούν δύο κλειδιά, το **Site Key** και το **Secret Key,** και αφου δημιουργήσετε 2 νέες μεταβλητές μέσα στο local_settings.py, τις **RECAPTCHA_PUBLIC_KEY**=**Site Key** και **RECAPTCHA_PRIVATE_KEY**= **Secret Key**(κάτω-κάτω στη γραμμή ~330 του αρχείου), θα βάλετε τα αντίστοιχα κλειδιά στο καθένα. Έπειτα, στη γραμμή 23 του local_settings.py θα

προσθέσετε την εξής γραμμή **INSTALLED_APPS += (‘snowpenguin.django.recaptcha2’,),** η οποία δηλώνει τη λίστα με τις πρόσθετες εγκατεστημένες εφαρμογές της ιστοσελίδας(εκτός απί τις ήδη εγκατεστημένες που βρισκόνται στο αρχείο settings.py το οποίο δια΄βαζει και το local_settings.py). Επόμενο βήμα είναι η τοποθέτηση του widget της εφαρμογής reCaptcha στην φόρμα εγγραφής λογαριασμού του DMOJ. Το αρχείο που είναι ο σχεδιαστής αυτής της φόρμας(εγγραφής λογαριασμού) βρίσκεται στο μονοπάτι **/home/user(alexsenu)/online-judge/templates/registration/registration_form.html** στο οποίο θα προσθέσετε στη **γραμμή 115 τη γραμμή <script src=”https:www.google.com/recaptcha/api.js async defer></script>** και στη **γραμμή 207 τη γραμμή <div class=”g-recaptcha” data-sitekey=”το site key”> **. Τελευταία προσθήκη είναι στο **/home/user(alexsenu)/online-judge/requirements.txt** στη **γραμμή 42** τη γραμμή **django-recaptcha2**.

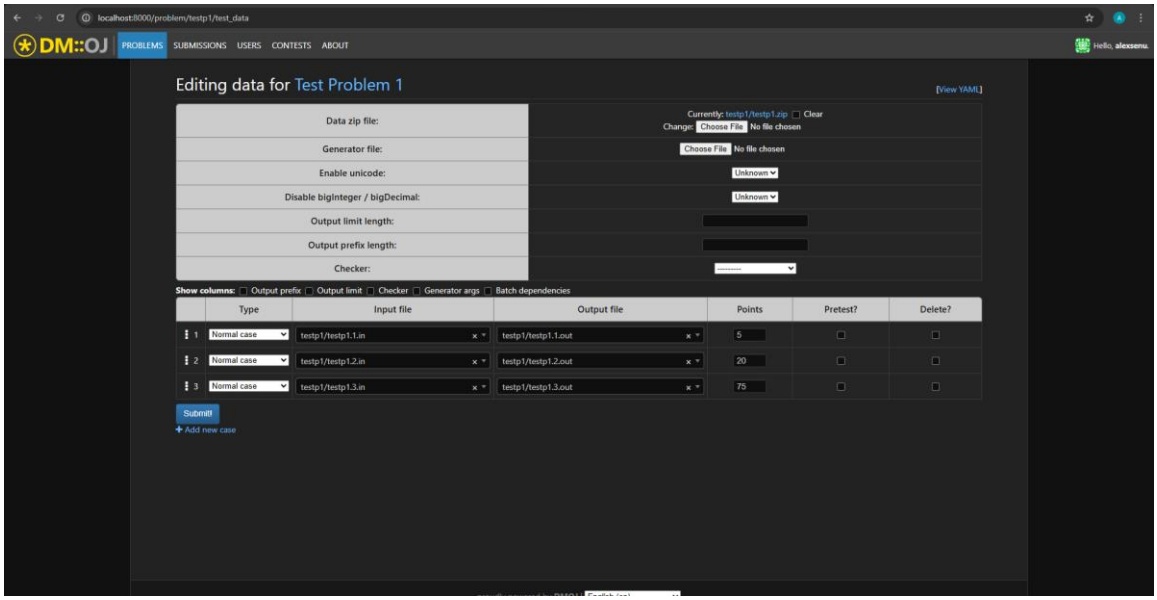
ΣΗΜΕΙΩΣΗ: εάν εμφανιστεί **traceback** που να αναφέρεται στο ‘**Did you mean: 'gettext_lazy'?’** συμβαίνει λόγω του ότι έχει αφαιρεθεί πλήρως η συνάρτηση **‘ugettextlazy’** και έχει αντικατασταθεί από τη **‘gettextlazy’** στο Django 4.0. Έτσι θα πρέπει να εντοπίσετε το αρχείο **/home/alexsenu/dmojsite/lib/python3.12/site-packages/snowpenguin/django/recaptcha2/templatetags/recaptcha2.py** και θα αλλάξετε στη **γραμμή 5 από ‘ugettext_lazy’ σε ‘gettext_lazy’**.

Διαχείριση προβλημάτων μέσω της διεπαφής του ιστότοπου

Το DMOJ συνοδεύεται από μια ηλεκτρονική διεπαφή για τη δημιουργία και την επεξεργασία δηλώσεων προβλημάτων καθώς και δεδομένων. Ορίστε το **DMOJ_PROBLEM_DATA_ROOT** σε έναν φάκελο της επιλογής σας. Τα δεδομένα δοκιμής(test_cases) για όλα τα προβλήματα με δεδομένα διαχείρισης τοποθεσίας θα αποθηκευτούν σε αυτόν τον φάκελο. Για να προσθέσετε ένα πρόβλημα κατευθυνθείτε εδώ και κάντε ADD στο Problems.



Εικόνα 13: Μέρος δημιουργίας προβλημάτων στην πλατφορμα



Εικόνα 14: Διαμόρφωση των test cases και data zip file μέσω της επιλογής Edit test data στη πλατφόρμα

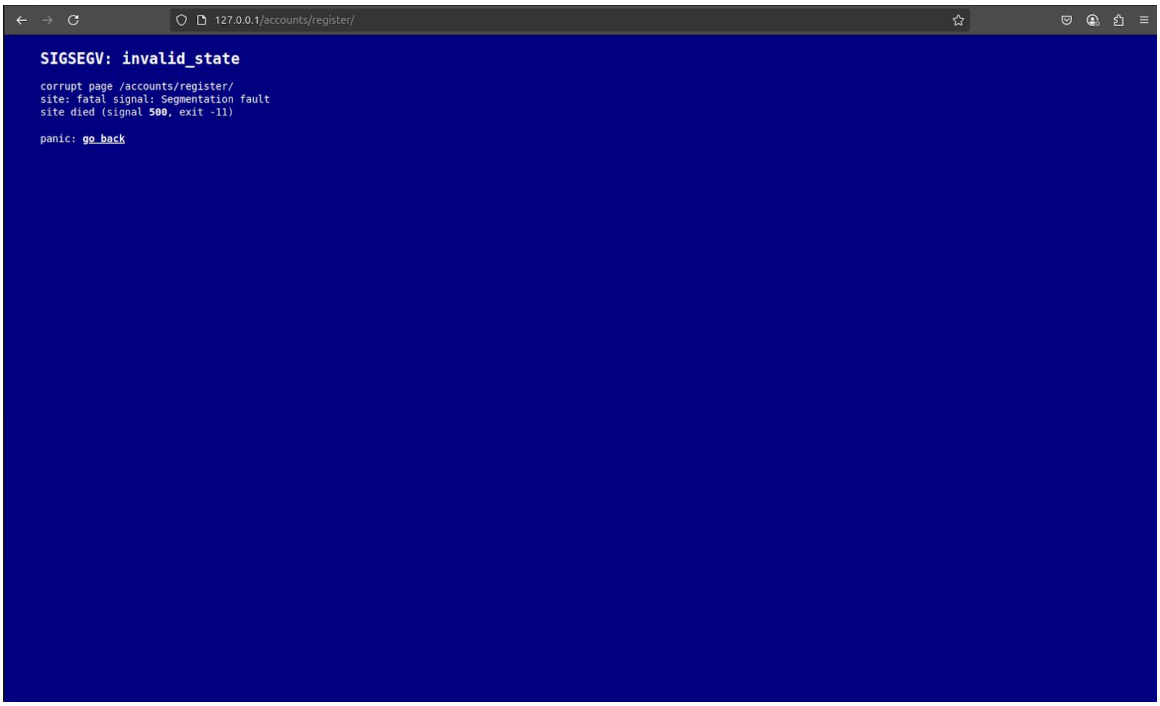
Έπειτα δώστε ένα όνομα και ένα κωδικό όνομα για το πρόβλημα και μη ξεχάσετε στους creators να αναφέρεται το όνομα σας, καθώς θα κλειδωθείτε έξω από το δικό σας πρόβλημα. Στο Problem body κάνετε τη περιγραφή του προβλήματος χρησιμοποιώντας το αρχείο https://raw.githubusercontent.com/DMOJ/docs/master/sample_files/problem_markdown_example.md.txt που προσφέρεται με διάφορες επιλογές μορφοποιήσης κειμένου. Επιλέξτε Problem types & Problem group παρακάτω. Αναφέρεται τους πόντους που κερδίζει όποιος λύνει το πρόβλημα σωστά, και αν χωρίζεται σε υποθέματα το πρόβλημα, όπου το καθένα, έχει το δικό του βαθμό, κάντε check το Allows partial points. Time Limit:2.0 & Memory limit:65537 ενδεικτικά για να μη σας βγάλει κάποιο πρόβλημα και επιλέξτε τις επιθυμητές γλώσσες προγραμματισμού στις οποίες θέλετε να εξεταστεί

το πρόβλημα σας και πατήστε Save. Έπειτα θα γυρίσετε πάλι στη καρτέλα Problems, και θα επιλέξετε να κάνετε **View on site** στο πρόβλημά σας. Εκεί θα βρείτε στα δεξιά σας τη καρτέλα **Edit test data** όπου και θα αναφέρεται στο **Data zip file** το μονοπάτι για το ξεχωριστό αρχείο του προβλήματος που περιέχει το αρχείο .zip με τα test cases. Στον ίδιο φάκελο με το zip θα υπάρχει και ένα αρχείο init.yaml. που θα αναφέρει ποιο αρχείο είναι in και ποιό το αντίστοιχο out. Αφού κάνετε submit θα σας πει το σύστημα να επιλέξτε input file & output file έτσι ώστε να γίνει η αντιστοίχιση των test cases. Μπορείτε για το πρώτο σας πρόβλημα να χρησιμοποιήστε ένα έτοιμο template απο το DMOJ που θα βρείτε εδώ: https://dmoj.ca/problem/aplusb

Ρύθμιση της λειτουργίας εγγραφής νέου χρήστη

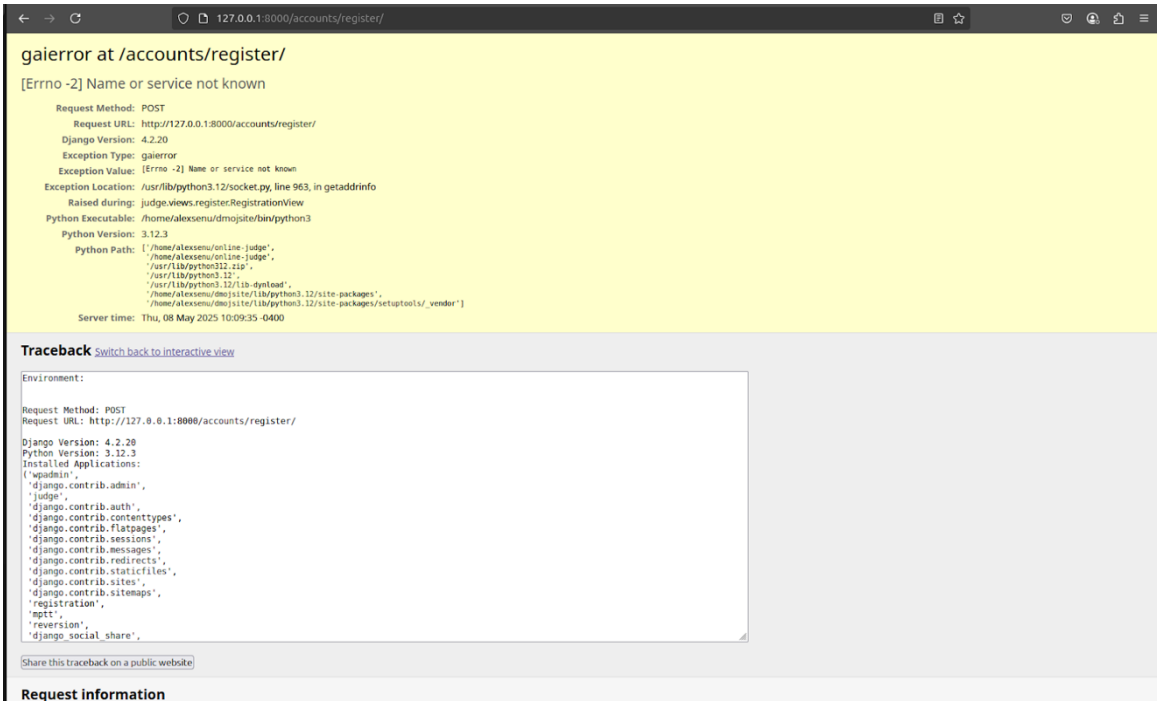
Στη προσπάθεια εγγραφής νέων χρηστών στην ιστοσελίδα και αφού συμπληρωθούν ολα τα απαραιτητα στοιχεια εγγραφης, αφού πατηθεί το κουμπί ‘’Register’’ η ιστοσελιδα μας βγάζει το εξης σφάλμα που φαίνεται στις φωτογραφίες, και επειδή μπορεί να δοκιμαστεί και απο **runserver** αλλα και απο **nginx** οι απαντήσεις μπορεί να διαφέρουν. Το Debug=True στο local_settings.py μας δείχνει τι μπορεί να πηγαίνει στραβά σε κάποιο σφάλμα.

Από **nginx**:



Εικόνα 15: Μήνυμα σφάλαματος στην εγγραφή χρήστη(Debug=False)

Από **runserver**:



Εικόνα 16: Μήνυμα σφάλματος στην εγγραφή χρήστη(Debug=False)

Στη δεύτερη εικόνα που λειτουργεί με **runserver** η **παραγωγή της ιστοσελίδας** η DEBUG μεταβλητή στο αρχείο local_settings.py είναι ακόμα στο True και για αυτό παίρνουμε το traceback το οποίο βοηθάει στο να βρεθει το πρόβλημα. Αυτο συμβαίνει γιατι δεν εχουν δηλωθεί στο αρχείο local_settings.py οι ρυθμίσεις για το υπεύθυνο λογαριασμό email που θα στέλνει μήνυμα επιβεβαίωσης στους χρήστες που θα δημιουργούν καινούργιο λογαριασμό στην ιστοσελίδα.

4.5 Αντιμετώπιση Προβλημάτων και Συντήρηση

Κατά τη διαδικασία εγκατάστασης και λειτουργίας του DMOJ, ενδέχεται να προκύψουν προβλήματα που σχετίζονται με λανθασμένες παραμετροποιήσεις ή με απουσία βιβλιοθηκών, είτε με δυσλειτουργίες υπο μοντέλων Git (submodules), είτε με την εκτέλεση του κώδικα σε εικονικά περιβάλλοντα Python. Παρακάτω περιγράφονται τα πιο χαρακτηριστικά ζητήματα που καταγράφηκαν στην πράξη κατά την υλοποίηση του συστήματος:

Πρόβλημα με Git Submodules

Ένα από τα πιο συνηθισμένα σφάλματα που εμφανίστηκε κατά την κλωνοποίηση του αποθετηρίου ήταν η ελλιπής ανάκτηση των Git submodules. Η πλατφόρμα DMOJ εξαρτάται από αυτά τα υπομοντέλα για κρίσιμες λειτουργίες (π.χ. bridged, judge workers).

$ git submodule update

Σε περιπτώσεις όπου η παραπάνω εντολή δεν λειτουργεί όπως αναμένεται, μπορεί να βοηθήσει η ανανέωση του submodule:

$ git submodule update –remote

Επιπλέον, καλό είναι να βεβαιωθεί κανείς πως εργάζεται στο σωστό branch με:

$ git checkout master

και αν υπάρχουν αλλαγές στο .gitmodules, να τις επικυρώσει με:

$ git submodule sync

Σφάλμα με pkg_resources / setuptools

Κατά την εκτέλεση της εντολής:

(dmojsite) $ python3 manage.py check

μπορεί να παρουσιαστεί σφάλμα ModuleNotFoundError: No module named 'pkg_resources'. Αυτό υποδεικνύει πως το setuptools δεν είχε εγκατασταθεί. Η παρακάτω εντολή εγκαθιστά το πακέτο και το σφάλμα δεν εμφανίζεται ξανά.

(dmojsite) $ pip install --upgrade setuptools

Αποτυχία δοκιμής των Celery workers

Παρατηρήθηκε ότι με την εντολή δοκιμής των Celery workers, το σύστημα δεν εντοπίζει σωστά την υπηρεσία **Redis,** που συνδεέται με τους Celery workers(επειδή η Celery δεν απαιτεί τη βοήθεια της Redis για αυτό εμφανίζεται και το σφάλμα), έτσι θα πρέπει να γίνει η εγκατάσταση του Redis με την παρακάτω εντολή:

(dmojsite)~/online-judge$ pip install redis

Πρόβλημα με το εικονικό περιβάλλον στο VS Code

Σε ορισμένες περιπτώσεις, κατά την εκκίνηση του VSCode, το τερματικό εμφανιζόταν σαν να ήταν ενεργό το dmojsite ενώ δεν στη πραγματικοτητα δεν ηταν ενεργοποιημένο. Παράλληλα, η εντολή deactivate δεν λειτουργούσε όπως αναμενόταν. Η παρακάτω εντολή ανανεώνει το περιβάλλον του terminal και "αδειάζει" το prompt από την ψευδή ενεργοποίηση.

```
$ exec bash
```

Θέματα με mysqlclient και συνδέσεις MariaDB

Απαιτείται εγκατάσταση συγκεκριμένων εργαλείων ανάπτυξης ώστε το mysqlclient να μπορεί να μεταγλωττιστεί σωστά. Αν δεν εγκατασταθούν, η εντολή pip3 install mysqlclient αποτυγχάνει. Αυτό επιτυγχάνεται με τη παρακάτω εντολή:

```
$ sudo apt install pkg-config python3-dev default-libmysqlclient-dev
build-essential
```

Ενημέρωση DMOJ & μετανάστευση βάσης δεδομένων

Καθώς το DMOJ είναι ενεργό έργο, προτείνεται ανά τακτά χρονικά διαστήματα η ανανέωση του κώδικα. Πριν την ενημέρωση, συνιστάται backup της βάσης δεδομένων. Τα βήματα ενημέρωσης είναι:

```
(dmojsite)~/online-judge$ git pull origin master

(dmojsite)~/online-judge$ pip3 install -r requirements.txt

(dmojsite)~/online-judge$ python3 manage.py makemigrations

(dmojsite)~/online-judge$ python3 manage.py migrate

(dmojsite)~/online-judge$ python3 manage.py check

(dmojsite)~/online-judge$ ./make_style.sh

dmojsite)~/online-judge$ python3 manage.py collectstatic

dmojsite)~/online-judge$ python3 manage.py compilemessages
```

dmojsite)~/online-judge$ python3 manage.py compilejsi18n

Αποτυχία φόρτωσης ζωντανής φόρτωσης του αποτελέσματος σε προβλήματα

Σημαντικό είναι να γίνει και comment out στο **local_settings.py** στη γραμμή **EVENT_DAEMON_USE** για να ενεργοποιηθεί η ζωντανή ενημέρωση στην ιστοσελίδα. Αφού γίνουν οι απαραίτητες αλλαγες μπείτε στο εικονικό περιβάλλον και ανανεώστε την λειτουργια της ιστοσελίδας με τις παρακάτω εντολες:

$ (dmojsite) $ sudo supervisorctl update

$ (dmojsite) $ sudo supervisorctl restart bridged

$ (dmojsite) $ sudo supervisorctl restart site

$ (dmojsite) $ sudo service nginx restart

$ (dmojsite) $ python3 manage.py collectstatic

Αδυναμία φόρτωσης Redis server

Μετά την εντολή φόρτωσης της τεχνολογίας Redis “service redis-server start” μπορεί να εμφανιστεί πρόβλημα μη ανίχνευσης κάποιου αρχείου που βοηθάει στην ενεργοποίηση της Redis. Αυτό λύνεται με την παρακάτω εντολή:

$ systemctl daemon-reload

Κεφάλαιο 5: Περιπτώσεις χρήσης και υλοποίηση του DMOJ

5.1 Εφαρμογή σε ακαδημαϊκό περιβάλλον

Το σύστημα του DMOJ παρέχει εξαιρετικές δυνατότητες στην αυτοματοποιημένη αξιολόγηση προγραμματιστικών λύσεων. Ένα από τα χρησιμότερα πεδία εφαρμογής του είναι τα ακαδημαϊκά περιβάλλοντα στα οποία η αξιολόγηση των προγραμματιστικών εργασιών είναι χρονοβόρα και απαιτεί αντικειμενικότητα. Σε πανεπιστημιακά και σχολικά περιβάλλοντα διεθνώς έχουν αναπτυχθεί ή χρησιμοποιηθεί online judges για την υποστήριξη μαθημάτων και διαγωνισμών προγραμματισμού. Χαρακτηριστικό παράδειγμα αποτελεί το Πανεπιστήμιο Marta Abreu του Las Villas (UCLV) στην

Κούβα, όπου εγκαταστάθηκε εσωτερική πλατφόρμα DMOJ (γνωστή ως DMOJ-UCLV) για την διεξαγωγή του εθνικού διαγωνισμού πληροφορικής και την εκπαίδευση μαθητών. Στα πλαίσια του Πανεπιστημίου Ιωαννίνων, ο καθηγητής που ανέθεσε την πτυχιακή εργασία υπέδειξε ότι το DMOJ θα μπορούσε να χρησιμοποιηθεί σε μαθήματα όπως «Προγραμματισμός 1», «Προγραμματισμός 2», «Αντικειμενοστραφής Προγραμματισμός» και «Αρχές Γλωσσών Προγραμματισμού», με στόχο την αξιολόγηση των φοιτητών που παρακολουθούν τα αντίστοιχα μαθήματα.

Η δυνατότητα αυτόματης αξιολόγησης καθιστά το DMOJ ιδανικό για τη βαθμολόγηση και την παρακολούθηση της προόδου των φοιτητών. Οι καθηγητές μπορούν να δημιουργήσουν ένα σύνολο από προβλήματα διαφορετικών τύπων που θα αξιολογηθούν αυτόματα από το σύστημα, εξασφαλίζοντας έτσι την αντικειμενικότητα στην αξιολόγηση, αλλά και τη δυνατότητα επαναλαμβανόμενης εξάσκησης από τους φοιτητές, μέχρι να σιγουρευτούν ότι ολοκλήρωσαν για το μάθημα «Προγραμματισμός 1» και μπορούν να λύσουν ασκήσεις που για τη σωστή εφαρμογή βασικών εννοιών, όπως οι αλγόριθμοι αναζήτησης.

Το DMOJ μέσω της άμεσης ανατροφοδότηση για τις λύσεις καταφέρνει να βελτιώνει την αποτελεσματικότητα της μάθησης και να ενθαρρύνει την αυτοεκπαίδευση. Με αυτόν το τρόπο το σύστημα μπορεί να ενισχύσει την εκπαιδευτική διαδικασία, επιτρέποντας στους φοιτητές να κερδίζουν πολύτιμο χρονο κατανοώντας τα λάθη τους και να τα διορθώνουν..

5.2 Μαθήματα προγραμματισμού

Η εφαρμογή του DMOJ σε μαθήματα προγραμματισμού έχει συγκεκριμένα πλεονεκτήματα, κυρίως λόγω της δυνατότητας να καλύψει μεγάλο όγκο φοιτητών και εργασιών σε μικρότερο χρόνο. Για παράδειγμα, σε μάθημα όπως το «Αντικειμενοστραφής Προγραμματισμός», το σύστημα μπορεί να αναθέτει και να αξιολογεί αυτόματα ασκήσεις που περιλαμβάνουν την υλοποίηση αντικειμενοστραφών προγραμμάτων, τη χρήση κλάσεων και τη διαχείριση κληρονομικότητας. Οι καθηγητές μπορούν να χρησιμοποιούν το DMOJ για να διαχειρίζονται το σύνολο των ασκήσεων, να παρέχουν παρατηρήσεις και να ενσωματώνουν νέες προκλήσεις.

Το DMOJ υποστηρίζει πολλαπλές γλώσσες προγραμματισμού, γεγονός που επιτρέπει στους φοιτητές να επιλέξουν τη γλώσσα που προτιμούν για την επίλυση των ασκήσεων τους, εξασφαλίζοντας έτσι την ευχέρεια να χρησιμοποιούν το εργαλείο χωρίς περιορισμούς. Επίσης, η δυνατότητα για ανατροφοδότηση σε πραγματικό χρόνο βοηθά τους φοιτητές να κατανοήσουν τα λάθη τους και να βελτιώσουν τις λύσεις τους.

5.3 Προετοιμασία Διαγωνισμών

Το DMOJ θεωρείται ένα εξαιρετικό εργαλείο για τη διοργάνωση διαγωνισμών προγραμματισμού. Η πλατφόρμα θα μπορούσε να βοηθήσει στη δημιουργηθια προγραμματιστικών διαγωνισμών για φοιτητές, είτε σε επίπεδο τάξης είτε σε επίπεδο πανεπιστημίου. Με την υποστήριξη τέτοιων διαγωνισμών, οι καθηγητές μπορούν να οργανώνουν και να παρακολουθούν τον ανταγωνισμό μεταξύ των φοιτητών, επιβραβεύοντας την ταχύτητα και την ορθότητα των λύσεων.

Με τη δυνατότητα αξιολόγησης και βαθμολόγησης σε πραγματικό χρόνο επιτρέπεται στους συμμετέχοντες να βλέπουν την πρόοδό τους, κάτι που προσδίδει μια πιο δυναμική διάσταση στο

διαγωνισμό. Η ανατροφοδότηση που παρέχεται στους διαγωνιζόμενους συμβάλλει στην επιθυμία τους για συνεχής βελτίωση, ενώ οι καθηγητές μπορούν να παρακολουθούν τις υποβολές σε πραγματικό χρόνο και να δίνουν feedback μέσω σχόλιων για διορθώσεις και για την απόδοση των συμμετεχόντων. Η πλατφόρμα υποστηρίζει επίσης τη δυνατότητα να διοργανώνονται διαγωνισμοί με προκαθορισμένα χρονικά παράθυρα και μερική βαθμολόγηση, δίνοντας την ευχέρεια στους φοιτητές να επιλύσουν επιμέρους μέρη των προβλημάτων.

5.4 Αξιολόγηση Φοιτητών

Με αυτή τη δυνατότητα οι φοιτητές θα μπορούν να έχουν στο προφίλ που έχουν δημιουργησει, ένα αρχείο με τα αποτελέσματα όλων των εξετάσεων αναλυτικά, και να μπορούν να κατευθύνονται αναλόγως, αν υπάρχουν αδυναμίες. Επιπλέον, με τη δυνατότητα βαθμολόγησης μερικών λύσεων, οι φοιτητές μπορούν να δουν την πρόοδό τους και να κατανοήσουν τα σημεία που χρειάζονται βελτίωση, ακόμα και αν η λύση τους δεν είναι πλήρως σωστή. Οι καθηγητές μπορούν να δημιουργήσουν ειδικά προβλήματα προγραμματισμού που να βοηθούν στην αντιμετώπιση καταστάσεων, όπως για φοιτητές που επιθυμούν να εξελίξουν τις ικανότητες τους, έπειτα άπο έλεγχο των βαθμολογιών τους, που είναι σώστα διαμορφωμένες στα προφίλ χρηστών.

5.5 Διαχείριση Διαγωνισμών Προγραμματισμού

Το DMOJ παρέχει τη δυνατότητα να οργανώνονται διαγωνισμοί προγραμματισμού τόσο σε τοπικό όσο και σε διεθνές επίπεδο. Η πλατφόρμα υποστηρίζει τη δημιουργία προβλημάτων, τη διαχείριση υποβολών και τη βαθμολόγηση συμμετοχών σε πραγματικό χρόνο. Οι διοργανωτές του διαγωνισμού μπορούν να ρυθμίσουν τα όρια χρόνου και μνήμης για κάθε πρόβλημα, προσδιορίζοντας έτσι το πλαίσιο αξιολόγησης.

5.6 Ατομική Εξάσκηση και Μάθηση

Ένα από τα πιο σημαντικά πλεονεκτήματα του DMOJ είναι η δυνατότητα για ατομική εξάσκηση και μάθηση. Οι φοιτητές μπορούν να χρησιμοποιούν την πλατφόρμα για να επιλύουν ασκήσεις προγραμματισμού με διάφορους βαθμούς δυσκολίας και να λαμβάνουν άμεση ανατροφοδότηση για την ορθότητα των λύσεών τους. Η συνεχής άσκηση μέσω του DMOJ μπορεί να βοηθήσει τους φοιτητές να βελτιώσουν τις ικανότητές τους στον προγραμματισμό, ενώ ταυτόχρονα τους ενθαρρύνει να επιλύουν πιο δύσκολα προβλήματα καθώς αναπτύσσουν τις δεξιότητές τους.

Η δυνατότητα να υποβάλλουν επανειλημμένα λύσεις και να λαμβάνουν άμεσα αποτελέσματα τους επιτρέπει να εντοπίζουν και να διορθώνουν τα λάθη τους σε πραγματικό χρόνο. Με αυτόν τον τρόπο, η ατομική εξάσκηση ενισχύεται και οι φοιτητές ενθαρρύνονται να αναπτύξουν ταχύτητα και ακρίβεια στην επίλυση προβλημάτων.

Κεφάλαιο 6: Συμπεράσματα και μελλοντικές επεκτάσεις

6.1 Σύνοψη

Συνοψίζοντας, η παρούσα εργασία πέτυχε την εγκατάσταση και διαμόρφωση του συστήματος **DMOJ (Modern Online Judge)**, αναδεικνύοντας τόσο τις δυνατότητες όσο και τις προκλήσεις μιας τέτοιας υλοποίησης. Μέσα από τα προηγούμενα κεφάλαια παρουσιάστηκε το θεωρητικό υπόβαθρο των συστημάτων αυτόματης βαθμολόγησης, η αρχιτεκτονική του DMOJ και ένας αναλυτικός οδηγός εγκατάστασης, καθώς και παραδείγματα χρήσης σε ακαδημαϊκό πλαίσιο. Τα αποτελέσματα καταδεικνύουν ότι το DMOJ αποτελεί μια ισχυρή πλατφόρμα αυτόματης αξιολόγησης προγραμματιστικών εργασιών, η οποία μπορεί να ενσωματωθεί αποτελεσματικά στην εκπαιδευτική διαδικασία.

Επιπλέον, η εμπειρία εγκατάστασης προσέφερε πολύτιμα συμπεράσματα. Παρά την φαινομενική πολυπλοκότητα (συνδυασμός πολλών τεχνολογιών όπως Django, MariaDB, Redis, Celery, Docker κ.ά.), αποδείχθηκε εντυπωσιακό το πώς όλα αυτά τα ετερογενή συστατικά **συνδέονται αρμονικά μέσω scripts και αρχείων ρυθμίσεων**. Απλές εντολές σε περιβάλλον τερματικού, αποδείχθηκαν ικανές να συντονίσουν μια σύνθετη εφαρμογή. Αυτό το γεγονός αναδεικνύει τη δύναμη των σύγχρονων εργαλείων ανάπτυξης και αυτοματισμού. Με τη σωστή σειρά ενεργειών, μια περίπλοκη πλατφόρμα μπορεί να στηθεί και να λειτουργήσει. Συνολικά, το **DMOJ** ως ένα ολοκληρωμένο σύστημα που πέτυχε το στόχο του έργου της αυτοματοποιημένης, αντικειμενικής και άμεσης αξιολόγησης κώδικα σε πραγματικό χρόνο, διευκολύνει τόσο τους εκπαιδευτές όσο και τους φοιτητές.

6.2 Προτεινόμενες βελτιώσεις

Παρά τη λειτουργικότητα και τη σταθερότητα που επέδειξε το DMOJ, η ανάλυση και η πρακτική εφαρμογή ανέδειξαν ορισμένα πεδία όπου μπορούν να γίνουν βελτιώσεις. Οι παρακάτω προτάσεις αποσκοπούν στο να κάνουν το σύστημα πιο φιλικό, ευέλικτο και ευρέως εφαρμόσιμο:

- **Γραφική διαδικασία εγκατάστασης (GUI Installer) :** η τρέχουσα εγκατάσταση απαιτεί αρκετά βήματα μέσω γραμμής εντολών και προϋποθέτει εξοικείωση με διάφορα εργαλεία. Η ανάπτυξη ενός φιλικού γραφικού περιβάλλοντος εγκατάστασης θα απλοποιούσε σημαντικά τη διαδικασία. Ένα **GUI installer** θα μπορούσε να καθοδηγεί τον διαχειριστή βήμα-βήμα, να ελέγχει αυτόματα τις εξαρτήσεις (π.χ. έλεγχος για MariaDB, Redis, Docker) και να ρυθμίζει τις παραμέτρους (όπως αρχεία ρυθμίσεων του Django) χωρίς να απαιτείται χειροκίνητη επεξεργασία, έτσι που ακόμη και χρήστες με περιορισμένη εμπειρία σε Linux ή διαχείριση συστημάτων θα μπορούσαν να εγκαταστήσουν το DMOJ πιο εύκολα, μειώνοντας τα πιθανά σφάλματα και εξοικονομώντας χρόνο.
- **Περαιτέρω αυτοματοποίηση με Docker Compose :** μια βελτίωση στην εγκατάσταση είναι η αξιοποίηση εργαλείων όπως το *Docker Compose* για την αυτοματοποίηση της ανάπτυξης. Αν και στο πλαίσιο αυτής της εργασίας χρησιμοποιήθηκε Docker container για τον δικαστή (judge) του DMOJ, θα ήταν ωφέλιμο να παρασχεθεί ένα έτοιμο σενάριο *Docker Compose* που να εκκινεί όλα τα απαραίτητα components (βάση δεδομένων, Redis, Django server, judge containers) με μια εντολή. Αυτό θα εξαλείψει την ανάγκη ξεχωριστών ρυθμίσεων για κάθε

υπηρεσία και θα διασφαλίσει ότι το περιβάλλον είναι αναπαραγώγιμο και συνεπές μεταξύ διαφορετικών εγκαταστάσεων.

- **Υποστήριξη επιπρόσθετων τύπων υποβολών (π.χ. αρχεία κειμένου/PDF) :** το DMOJ είναι σχεδιασμένο κυρίως για αξιολόγηση πηγαίου κώδικα, ωστόσο θα ήταν χρήσιμη η επέκτασή του ώστε να δέχεται και άλλους τύπους αρχείων προς υποβολή. Για παράδειγμα, η δυνατότητα υποβολής αρχείων **.txt** ή **.pdf** θα επέτρεπε τη χρήση της πλατφόρμας και σε μαθήματα θεωρητικού ή μαθηματικού χαρακτήρα, όπου οι φοιτητές θα μπορούσαν να υποβάλουν γραπτές λύσεις ή αποδείξεις. Μια τέτοια δυνατότητα θα προϋπέθετε διαφορετικό τρόπο αξιολόγησης (ενδεχομένως μη αυτόματο ή ημιαυτόματο, με σχόλια από τους διδάσκοντες), όμως η ενσωμάτωση όλων των εργασιών σε μία ενιαία πλατφόρμα θα διευκόλυνε τη διαχείριση των εργασιών ενός μαθήματος. Ως εκ τούτου, προτείνεται η μελλοντική προσθήκη υποστήριξης για αρχεία κειμένου/PDF, έστω και αν αυτά δεν βαθμολογούνται αυτόματα, ώστε το DMOJ να λειτουργεί και ως γενικό σύστημα διαχείρισης εργασιών.
- **Υποστήριξη ερωτήσεων πολλαπλής επιλογής και quizzes :** μια ακόμη λειτουργική βελτίωση θα ήταν η ενσωμάτωση τύπων προβλημάτων ακόμα και προγραμματιστικών, όπως ερωτήσεις **πολλαπλής επιλογής** (multiple choice) ή σύντομα quizzes γνώσεων. Με την προσθήκη αυτής της δυνατότητας, το DMOJ θα μπορούσε να χρησιμοποιηθεί για την αυτοματοποιημένη αξιολόγηση θεωρητικών γνώσεων, παρόμοια με ένα σύστημα διαχείρισης quiz. Για παράδειγμα, οι εκπαιδευτικοί θα μπορούσαν να δημιουργούν ερωτήσεις με προκαθορισμένες απαντήσεις και το σύστημα να βαθμολογεί αυτόματα τις επιλογές των φοιτητών. Αυτή η βελτίωση θα απαιτούσε την προσθήκη νέων μοντέλων δεδομένων (για τις ερωτήσεις/απαντήσεις) και διεπαφών χρήστη, ωστόσο θα μετέτρεπε το DMOJ σε μια πολυμορφική εκπαιδευτική πλατφόρμα που καλύπτει τόσο πρακτική προγραμματισμού όσο και θεωρητική αξιολόγηση γνώσεων.
- **Βελτίωση τεκμηρίωσης και υποστήριξης**: Τέλος, προτείνεται η συνεχής βελτίωση της τεκμηρίωσης (documentation) και των εργαλείων υποστήριξης της κοινότητας. Κατά τη διάρκεια της εγκατάστασης διαπιστώθηκαν μικρά ζητήματα (π.χ. ανάγκη αναβάθμισης του **setuptools**, ρυθμίσεις του VS Code για εικονικά περιβάλλοντα, εξαρτήσεις για το mysqlclient) τα οποία ξεπεράστηκαν μέσω αναζήτησης λύσεων. Η επίσημη τεκμηρίωση θα μπορούσε να δίνει περισσότερες πληροφορίες σε τέτοια κοινά προβλήματα ή να διατηρεί μια ενημερωμένη λίστα “γνωστών προβλημάτων” και λύσεων. Να σημειωθεί ότι στο επίσημο αποθετήριο του DMOJ(https://github.com/DMOJ) όποια προβλήματα έχουν προκύψει, αναφέρονται στη καρτέλα issues του αντιστοιχου υποαποθετηρίου που έχει το πρόβλημα και υπάρχει υπεύθυνη ομάδα να τα διευθετήσει. Επίσης, έχουν δημιουργήσει και στο Discord ομάδα, όπου συχνά προβλήματα που μπορεί να προκύπτουν κατά την εγκατάσταση της ιστοσελίδας, αναφέρονται με επισήμανση λέξεων-κλειδιών(π.χ σφάλμα ενεργοποίησης reCaptcha) και τα επίσημα μέλη της ομάδας DMOJ, μέσω σχόλειων προσπαθούν να κατατοπίσουν τους ενδιαφερόμενους.

6.3 Δυνατότητες επέκτασης

Οι πολυδιάστατες δυνατότητες του DMOJ ανοίγουν τον δρόμο για επέκταση του πεδίου εφαρμογής του, τόσο εντός όσο και εκτός του αρχικού πλαισίου χρήσης. Βάσει των όσων παρουσιάστηκαν, διακρίνονται οι εξής βασικές δυνατότητες επέκτασης της πλατφόρμας:

- **Υιοθέτηση σε περισσότερα ελληνικά πανεπιστήμια :** μια άμεση επέκταση αφορά τη διάδοση και υιοθέτηση του DMOJ και από άλλα ακαδημαϊκά ιδρύματα στην Ελλάδα. Θα μπορούσε να αναπτυχθεί ένα δίκτυο πανεπιστημίων που χρησιμοποιούν από κοινού την πλατφόρμα για την αξιολόγηση φοιτητικών εργασιών. Αυτό θα μπορούσε να γίνει είτε μέσω ανεξάρτητων εγκαταστάσεων σε κάθε ίδρυμα, είτε μέσω μιας κεντρικής εγκατάστασης που θα εξυπηρετεί πολλαπλά πανεπιστήμια με διακριτούς χώρους για το καθένα. Η επέκταση αυτή θα προαγάγει την ομοιομορφία και συνεργασία στην εκπαιδευτική αξιολόγηση σε εθνικό επίπεδο, ενώ παράλληλα θα δημιουργήσει μια κοινότητα χρηστών και προγραμματιστών που θα συμβάλλουν στη βελτίωση του συστήματος.
- **Εφαρμογή σε ευρύτερα αντικείμενα και επίπεδα εκπαίδευσης :** αν και το DMOJ σχεδιάστηκε για προγραμματιστικά προβλήματα, υπάρχει δυνατότητα επέκτασης της χρήσης του και σε άλλα αντικείμενα. Ήδη αναφέρθηκε η πιθανή αξιοποίησή του σε θεωρητικά ή μαθηματικά μαθήματα(με την αποδοχή γραπτών λύσεων), κάτι που θα το μετέτρεπε σε ένα γενικότερο σύστημα διαχείρισης ασκήσεων. Το DMOJ θα μπορούσε να χρησιμοποιηθεί και σε άλλα επίπεδα εκπαίδευσης, όπως στην δευτεροβάθμια εκπαίδευση (π.χ. σε πρότυπα ή πειραματικά σχολεία με προγράμματα προγραμματισμού) ή σε διατμηματικά μαθήματα όπου απαιτείται αξιολόγηση αλγοριθμικής σκέψης. Η προσαρμογή του περιβάλλοντος(π.χ. με πιο απλοποιημένο UI) θα επέτρεπε την ομαλή ένταξη του συστήματος και σε αυτούς τους χώρους.
- **Δημιουργία κεντρικού αποθετηρίου και ανταλλαγή προβλημάτων :** μια σημαντική δυνατότητα επέκτασης έγκειται στην αξιοποίηση του DMOJ ως πυρήνα για ένα **κεντρικό αποθετήριο προβλημάτων** προσβάσιμο από πολλούς φορείς. Για παράδειγμα, τα ελληνικά πανεπιστήμια που θα υιοθετήσουν το σύστημα θα μπορούσαν να συνεισφέρουν σε μια κοινή βιβλιοθήκη ασκήσεων προγραμματισμού (ή και θεωρίας), διαθέσιμη σε όλους τους χρήστες. Αυτό θα μείωνε τον κατακερματισμό της προσπάθειας δημιουργίας νέων προβλημάτων και θα επέτρεπε την επαναχρησιμοποίηση και βελτίωση των θεμάτων από πολλούς εκπαιδευτικούς. Η πλατφόρμα ήδη υποστηρίζει πολλαπλές γλώσσες προγραμματισμού και τύπους προβλημάτων· επεκτείνοντάς την με μία κοινή **τράπεζα** θεμάτων, θα ενισχυθεί η χρησιμότητά της ως εργαλείο συνεργασίας στην εκπαίδευση. Παράλληλα, η ανταλλαγή δεδομένων (π.χ. στατιστικών απόδοσης φοιτητών σε κοινά προβλήματα) θα μπορούσε να δώσει πολύτιμες πληροφορίες στους διδάσκοντες για τη δυσκολία των ασκήσεων και τα συχνότερα λάθη των φοιτητών, οδηγώντας σε βελτιώσεις στο διδακτικό υλικό.
- **Ολοκλήρωση με υπάρχουσες πλατφόρμες μάθησης :** για να διευκολυνθεί η υιοθέτηση του DMOJ στα πανεπιστήμια, μπορεί να εξεταστεί η ενσωμάτωσή του με πλατφόρμες e-learning (π.χ. Ασύγχρονη Τηλεκπαίδευση). Μια τέτοια ολοκλήρωση θα επέτρεπε στους διδάσκοντες να διαχειρίζονται εργασίες και quizzes του DMOJ απευθείας μέσα από το ήδη γνωστό περιβάλλον, και στους φοιτητές να υποβάλλουν λύσεις χωρίς να χρειάζεται ξεχωριστή σύνδεση/εγγραφή στην πλατφόρμα DMOJ. Τεχνικά, αυτό θα μπορούσε να υλοποιηθεί μέσω ανάπτυξης plugins ή χρήσης του API του DMOJ για την ανταλλαγή δεδομένων (π.χ. βαθμολογιών, κατάστασης υποβολών). Με την επέκταση αυτή, το DMOJ θα καταστεί μέρος ενός ολοκληρωμένου οικοσυστήματος εκπαιδευτικών εργαλείων, αυξάνοντας την αξία και την απήχησή του στην εκπαιδευτική κοινότητα.

6.4 Μελλοντικές κατευθύνσεις έρευνας

Η ολοκλήρωση της παρούσας εργασίας ανοίγει νέες προοπτικές για περαιτέρω έρευνα και εξέλιξη, τόσο στο επίπεδο της τεχνολογίας των αυτοματοποιημένων κριτών όσο και στο επίπεδο της εκπαιδευτικής μεθοδολογίας. Ακολουθούν ορισμένες μελλοντικές κατευθύνσεις έρευνας που θα μπορούσαν να διερευνηθούν βάσει των συμπερασμάτων:

- **Αξιολόγηση εκπαιδευτικής αποτελεσματικότητας :** μια σημαντική κατεύθυνση είναι η παιδαγωγική έρευνα γύρω από τη χρήση του DMOJ (ή γενικά των Online Judge συστημάτων) στη μάθηση. Θα ήταν ωφέλιμο να μελετηθεί συστηματικά ο αντίκτυπος της άμεσης ανατροφοδότησης και της αυτόματης βαθμολόγησης στην επίδοση των φοιτητών. Για παράδειγμα, με συγκριτικές μελέτες σε τάξεις που χρησιμοποιούν το DMOJ έναντι τάξεων με παραδοσιακή αξιολόγηση, μπορεί να εξαχθούν συμπεράσματα σχετικά με τη βελτίωση των δεξιοτήτων προγραμματισμού, την αύξηση του ενδιαφέροντος ή τη μείωση του χρόνου βαθμολόγησης. Τέτοιου είδους έρευνα θα τεκμηριώσει με ποσοτικά και ποιοτικά δεδομένα τα εκπαιδευτικά οφέλη (ή τυχόν προκλήσεις) από την ενσωμάτωση αυτόματων κριτών στην διδασκαλία.
- **Βελτιστοποίηση και κλιμάκωση του συστήματος :** σε τεχνολογικό επίπεδο, μια μελλοντική έρευνα θα μπορούσε να επικεντρωθεί στη **βελτιστοποίηση της απόδοσης** και στην ικανότητα κλιμάκωσης του DMOJ. Αυτό περιλαμβάνει τη διερεύνηση πιο αποδοτικών μεθόδων **sandboxing** και εκτέλεσης κώδικα για να μειωθεί ο χρόνος αξιολόγησης ανά υποβολή, τη βελτίωση του καταμερισμού φόρτου μεταξύ πολλαπλών δικαστών (load balancing) και τη διασφάλιση ότι το σύστημα μπορεί να διαχειριστεί μεγάλο όγκο ταυτόχρονων χρηστών (π.χ. σε έναν διαγωνισμό πανελλαδικής εμβέλειας) χωρίς προβλήματα. Παράλληλα, μπορεί γίνει έρευνα πάνω σε νέες τεχνικές ανίχνευσης κακόβουλου κώδικα ή αποτροπής επιθέσεων από υποβολές, ώστε να ενισχυθεί ακόμη περισσότερο η προστασία του συστήματος έναντι εξελιγμένων απειλών.
- **Επεκτάσεις με τεχνητή νοημοσύνη στην αξιολόγηση:** Μια συναρπαστική μελλοντική κατεύθυνση είναι η ενσωμάτωση μεθόδων **Τεχνητής Νοημοσύνης (AI)** ή *machine learning* για την αξιολόγηση των υποβολών. Ενώ το DMOJ σήμερα κρίνει κυρίως με βάση **προκαθορισμένες δοκιμές (test cases)**, θα μπορούσε να διερευνηθεί η χρήση AI για πιο ποιοτική αξιολόγηση, όπως η ανάλυση του κώδικα για βέλτιστες πρακτικές ή η παροχή έξυπνης ανατροφοδότησης. Για παράδειγμα, ένα σύστημα machine learning θα μπορούσε να εκπαιδευτεί να **ανιχνεύει μοτίβα προβλημάτων στον κώδικα** (π.χ. συνηθισμένα λάθη λογικής ή μη αποτελεσματικές υλοποιήσεις) και να δίνει εξατομικευμένα σχόλια στους φοιτητές πέρα από το “σωστό/λάθος”. Επίσης, τεχνικές επεξεργασίας φυσικής γλώσσας (NLP) θα μπορούσαν να χρησιμοποιηθούν για την αυτόματη αξιολόγηση σύντομων ανοικτών απαντήσεων ή περιγραφών αλγορίθμων, επεκτείνοντας έτσι το φάσμα των δυνατοτήτων της πλατφόρμας.
- **Σύγκριση και συνδυασμός με άλλα συστήματα :** τέλος, μια μελλοντική μελέτη θα μπορούσε να εστιάσει στη **συγκριτική αξιολόγηση** του DMOJ με άλλα διαθέσιμα συστήματα online judge ή πλατφόρμες αξιολόγησης. Παράγοντες όπως η ευχρηστία, η απόδοση, το εύρος των λειτουργιών και η ευκολία ενσωμάτωσης σε εκπαιδευτικά προγράμματα μπορούν να αποτελέσουν κριτήρια σύγκρισης. Επιπλέον, μπορεί να εξεταστεί η δυνατότητα

συνδυαστικής χρήσης εργαλείων – για παράδειγμα, η διασύνδεση του DMOJ με εργαλεία ανίχνευσης λογοκλοπής κώδικα (plagiarism detection) ή με πλατφόρμες ανάπτυξης (π.χ. GitHub Classroom). Η έρευνα σε αυτή την κατεύθυνση θα βοηθήσει να εντοπιστούν βέλτιστες πρακτικές και να καθοριστεί πώς το DMOJ μπορεί να εξελιχθεί περαιτέρω ώστε να παραμείνει ανταγωνιστικό και συμβατό με τις σύγχρονες ανάγκες.

Συμπερασματικά, το κεφάλαιο αυτό ανέδειξε ότι το **DMOJ** έχει εκπληρώσει τους αρχικούς στόχους της πτυχιακής εργασίας, ενώ παράλληλα φέρνει στο προσκήνιο πολλές δυνατότητες για βελτίωση και διεύρυνση. Οι προτάσεις που συζητήθηκαν – από πρακτικές βελτιώσεις στην εγκατάσταση και τις λειτουργίες έως ευρύτερες επεκτάσεις χρήσης και ερευνητικές κατευθύνσεις δείχνουν τον δρόμο για το μέλλον του συστήματος. Η υλοποίηση ενός αυτόματου κριτή σε επίπεδο πανεπιστημίου είναι μόνο η αρχή· με συστηματική προσπάθεια και συνεργασία, το DMOJ μπορεί να εξελιχθεί περαιτέρω, συμβάλλοντας ουσιαστικά στην αναβάθμιση της εκπαιδευτικής διαδικασίας και στην ενίσχυση της κουλτούρας του προγραμματισμού στην Ελλάδα

Βιβλιογραφία

[1] «EXPLAINABLE AI: PROVIDING STUDENT FEEDBACK VIA ONLINE JUDGE,» 2024. [Ηλεκτρονικό] Διαθέσιμο: https://www.jetir.org/papers/JETIR2408659.pdf
[2]"The Cuban Olympiad in Informatics: A New Stage from the DMOJ Online Judge,"2021. [Ηλεκτρονικό] Διαθέσιμο: https://ioinformatics.org/journal/v15_2021_133_141.pdf
[3] «A Survey on Online Judge Systems and Their Applications,» 2017 [Ηλεκτρονικό] Διαθέσιμο: https://ar5iv.labs.arxiv.org/html/1710.05913
[4]"How problem difficulty and order influence programming education outcomes in online judge systems," 2023.[Ηλεκτρονικό] Διαθέσιμο : https://pubmed.ncbi.nlm.nih.gov/37954262/
[5]"Student perception and usage of an automated programming assessment tool,"2013 [Ηλεκτρονικό] Διαθέσιμο: https://www.sciencedirect.com/science/article/abs/pii/S0747563213001040
[6]"About AtCoder,"2022. [Ηλεκτρονικό] Διαθέσιμο : https://www.codeandpastries.dev/blog/post/atcoder
[7]"About LeetCode" [Ηλεκτρονικό] Διαθέσιμο : https://leetcode.com/
[8]"Introduction to the programming skill assessment system DMOJ,"2023. [Ηλεκτρονικό] Διαθέσιμο: https://dlib.phenikaa-uni.edu.vn/bitstream/PNK/9806/1/Implement%20a%20Progamming%20Skill%20Assesment%20website%20-%20DMOJ.pdf
[9]"Olympiads in Informatics"2021. [Ηλεκτρονικό] Διαθέσιμο: https://ioinformatics.org/files/volume15.pdf
[10]" Dmoj: Modern Online Judge, documentation"2022. [Ηλεκτρονικό] Διαθέσιμο: https://docs.dmoj.ca/#/