UNIVERSITATEA POLITEHNICA DIN BUCUREȘTI
FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE
DEPARTAMENTUL DE CALCULATOARE

PROIECT DE DIPLOMĂ

AcadNet.dev
Platformă online pentru rezolvarea problemelor de informatică

Dimitrie David

Coordonator științific:

Prof. Dr. Ing. Răzvan Victor Rughiniș

BUCUREȘTI
2023

UNIVERSITY POLITEHNICA OF BUCHAREST
FACULTY OF AUTOMATIC CONTROL AND COMPUTERS
COMPUTER SCIENCE AND ENGINEERING DEPARTMENT

DIPLOMA PROJECT

AcadNet.dev
Online platform for solving computer science problems

Dimitrie David

Thesis advisor:

Prof. Dr. Ing. Răzvan Victor Rughiniș

BUCHAREST
2023

CONTENTS

SINOPSIS

Această lucrare prezintă AcadNet.dev, o platformă online pentru rezolvarea de probleme de informatică. Platforma oferă un mediu de lucru complet, care permite utilizatorilor să creeze probleme, să le rezolve și să le evalueze automat. Platforma oferă și un mediu de lucru online, care permite utilizatorilor să rezolve problemele direct în browser, fără a fi nevoie să configureze un mediu de dezvoltare local.

ABSTRACT

This paper presents AcadNet.dev, an online platform for solving programming problems. The platform offers a complete working environment, which allows users to create problems, solve them and automatically evaluate them. The platform also provides an online working environment, which allows users to solve problems directly in the browser, without the need to configure a local development environment.

1 INTRODUCTION

1.1 Motivation and problem statement

For the past half year, I was responsible for coordinating the development of problems for the Software Interoperability section of the National Olympiad of Applied Informatics - Acadnet. This section is different than the regular informatics olympiad, as it focuses more on the engineering side of informatics, rather than the theoretical side. The problems are more practical and require students to be more creative in order to solve them.

The problems are formulated as real life scenarios, where a code is given that should have a certain behavior, but it does not. The students have to find bugs in the code and fix them.

As of right now, there is no accessible methods for students to train for this olympiad. The only way to practice is to solve the problems from the previous years, by downloading their statement and original source. There are no tests to check if the solution is correct. The students have to compile and run the code themselves, and check if the output is correct.

The platform's goal is to create an environment where students can train and prepare for the olympiad in a more efficient way. Moreover, we want to create an online workspace for students, where they can solve problems directly in the browser. This allows us to get more creative with the engineering problems, as we can use more programming languages and configurations, without putting the students through the hassle of setting up a local development environment.

1.2 Objectives

During the last half year, I have been gathering insights on what the students and authors want from a platform like this. I have also been researching the available technologies, and I have been experimenting with different approaches. Based on this, the objectives of this project are defined as follows.

The objective of this project is to create a universal platform for solving engineering tasks. This should include the ability for authors to extend the platform and implement any new language that they want to write a problem in. In addition, the platform should give students the opportunity to solve the tasks directly in the browser, without requiring anything more than an account. In terms of functionality, the platform should allow authors to create problems, and students to solve them. The platform should also provide a way to automatically evaluate the solutions, and give feedback to the students.

In terms of security, because the solutions will be evaluated by executing user-written code, the platform should be able to run the code in a sandboxed environment, and prevent malicious code from being executed. The platform should also prevent students from cheating, by not allowing them to see the test cases, other users submissions, or the source code of the solutions.

The platform should be easily maintainable, and should be able to scale to a large number of users. It should use containerization to allow for easy deployment and scaling.

1.3 Proposed solution and achieved results

The core of the platform is a web application, which acts as the interface between the users and the platform. The web application is responsible for managing the users, problems, submissions, and for evaluating the submissions. In addition to this, the web application will also be a proxy for online workspaces.

The web application is backed by a SQL database and an S3 file storage. The database is used to store the users, problems, submissions, and other metadata. The file storage is used to store the source code of the problems, and the submissions.

The web application is written in C# using the .NET Core framework. The database is a PostgreSQL database, and the file storage is an S3 compatible storage, provided by DigitalOcean. The web application is deployed using Helm Charts on a Kubernetes cluster, also provided by DigitalOcean. The checker is written in Python and it is more like an sandbox manager that spawns a new container for each submission, and runs the tests inside the container. Like the checker, the workspace manager is also written in Python, and it is responsible for spawning new workspaces. Workspaces are Docker images based on a Visual Studio Code fork, that allows us to run the code directly in the browser.

The working platform can be found at AcadNet.dev and the source code is available on Github inside the acadnet-dev organization.

2 SYSTEM DESIGN AND ARCHITECTURE

2.1 User journey

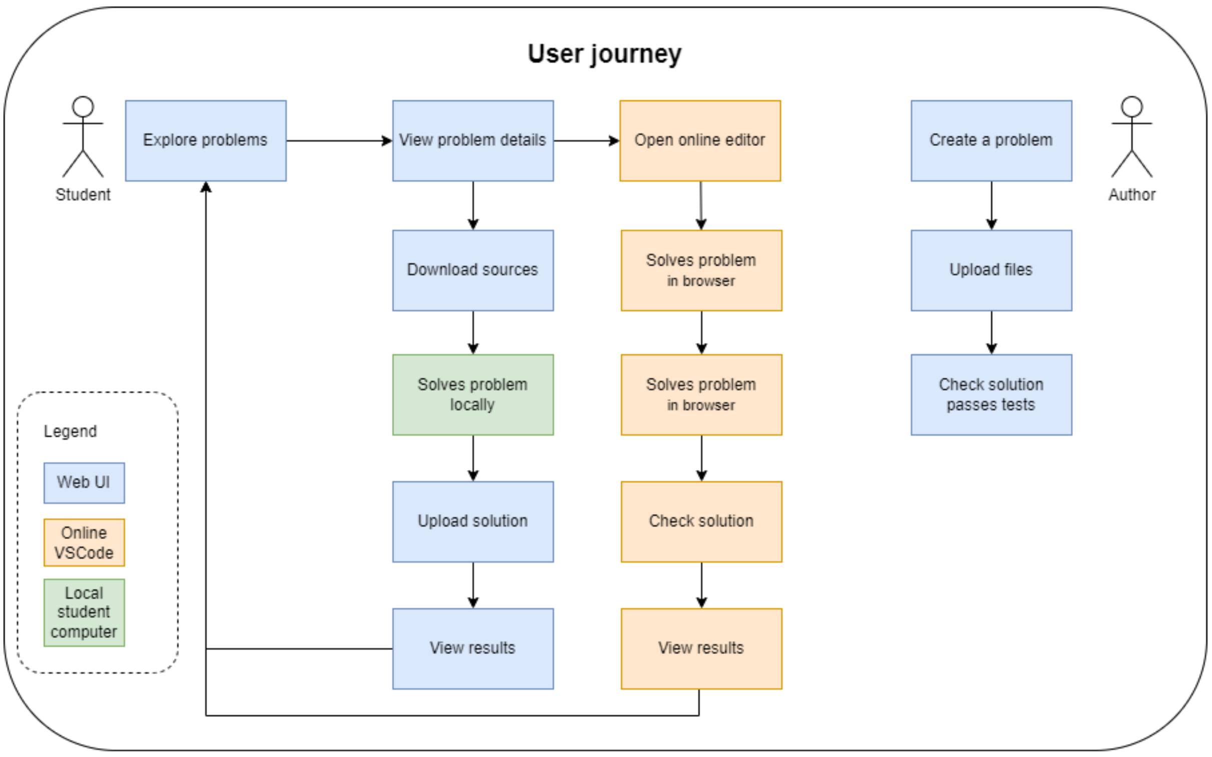

Figure 1: User journey

The platform's main actors are the users that want to learn and practice programming, by solving the problems available. The students can browse the problems, but in order to solve them, they have to create an account. After that, they can choose to solve the problems directly in the browser or they can download the source code and solve them locally. After they solve the problem, they can submit their solution for evaluation. The platform will automatically evaluate the solution, and give feedback to the them.

The other actors are the authors, who are responsible for creating the problems. The authors can create problems, and upload relevant files for the problem. These files include the statement, the source code of the problem, the test cases, and the solution.

2.2 Architecture overview

The architecture is composed of 3 main components: the web application, the checker, and the VSCode workspaces. The web application is the main component, and it is responsible for managing the users, problems, submissions, and for evaluating the submissions. The checker is responsible for evaluating the submissions, and the VSCode workspaces are used to allow users to solve the problems directly in the browser. The block diagram of the architecture can be seen in the figure below.

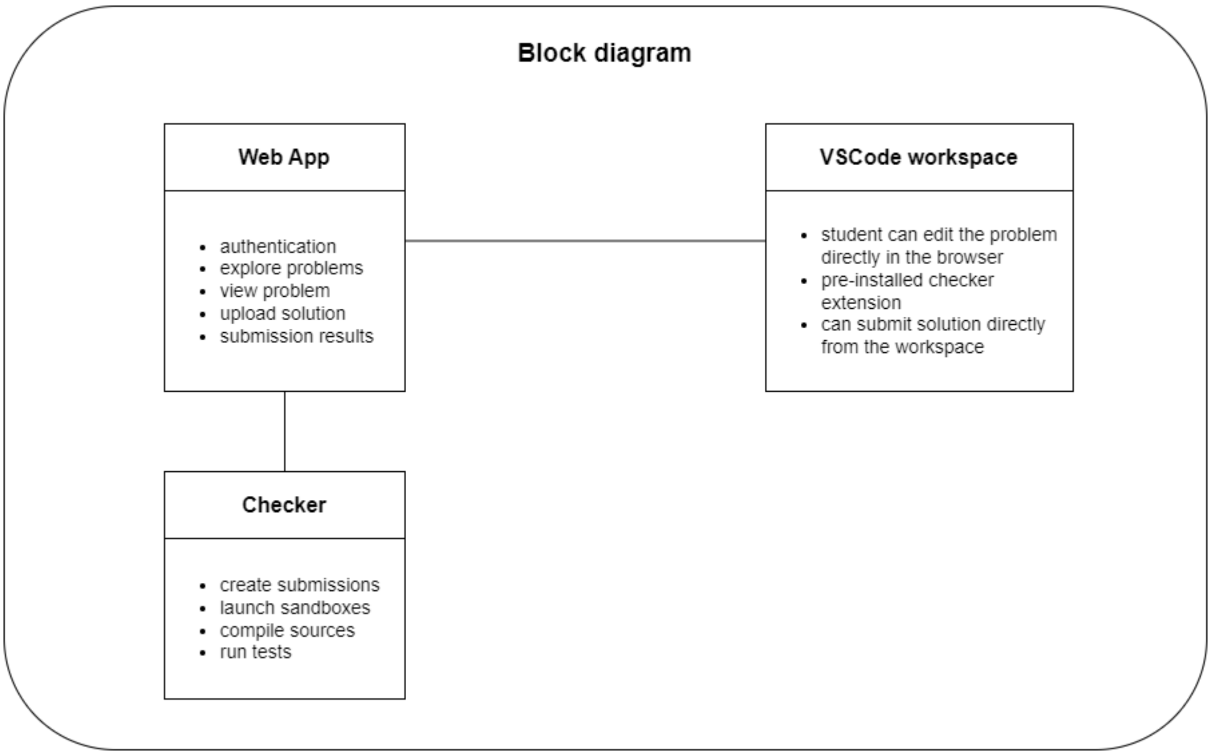

Figure 2: Software block diagram

Table 1: Component's responsibilities

Component	Responsibilities
Web App	Manage users, problems and submissions; send submissions to the checker and present results; manage VSCode workspaces
Checker	Test submissions by spawning a new sandbox in the K8s cluster and compare reference outputs with actual outputs
VSCode workspace	Allow users to solve problems directly in the browser

2.3 System components and interactions

The web app is the brain of the platform. It is interacting with all the other components. With regards to the checker, the web app is responsible for receiving the submissions from the user or from the online workspace and to send them to the checker. While the checker is evaluating the submission, the web app constantly polls the checker for the updates and results. When the checker is done, the web app will receive the results and will update the submission with the status. On the other hand, when it comes to the interaction with the online workspaces, the web app is responsible for spawning new workspaces via the workspace manager and it acts as a HTTP and websocket proxy between the user's browser and the workspace.

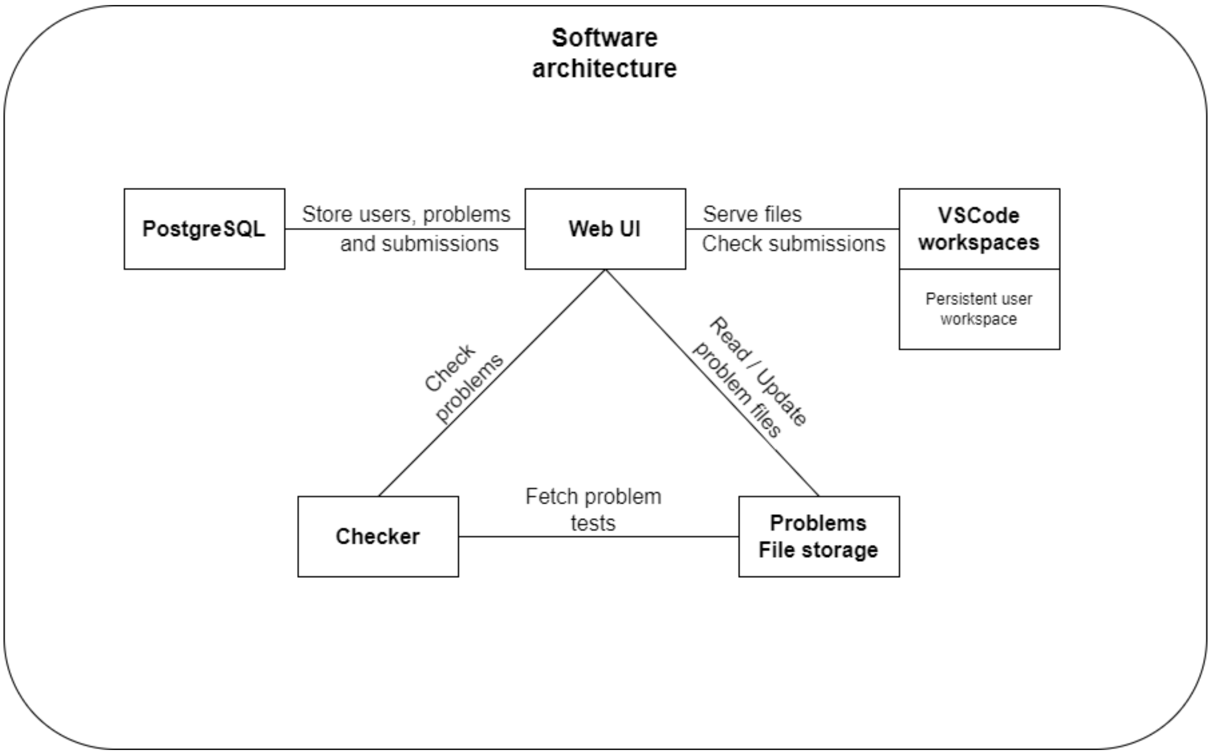

Figure 3: System components and their interactions

2.3.1 SQL and file storage

Other than the 3 main components, the architecture also includes a SQL database and an S3 file storage. The database is used to store the users, problems, submissions, and other metadata. The file storage is used to store the source code of the problems, and the submissions. The interactions between the components can be seen in the figure below. The web application is the only component that interacts with the database. With the file storage, the web application and the checker interact with it. The web application uses it to store the files uploaded by the authors, and the checker uses it to pull the test cases for the problem.

2.4 Database design

3 IMPLEMENTATION DETAILS

3.1 Web application

3.2 SQL database

3.3 File storage

3.4 Checker

3.5 VSCode workspaces

4 FEATURES AND FUNCTIONALITIES

4.1 User authentication

4.2 Problem creation

4.3 Problem browsing

4.4 Problem solving

4.5 Submission evaluation

4.6 Online workspaces

5 DEPLOYMENT AND MAINTENANCE

5.1 Infrastructure

5.2 Continuous integration

5.3 Continuous deployment

5.4 Monitoring

6 TESTING AND VALIDATION

Read in template.pdf

6.1 Evaluation criteria

6.2 Performance

6.3 Security

6.4 User feedback

7 CONCLUSIONS AND FUTURE WORK

7.1 Conclusions

7.2 Future enhancements

7.3 Lessons learned

BIBLIOGRAPHY

- NU utilizați referințe la Wikipedia sau alte surse fără autor asumat.
- Pentru referințe la articole relevante accesibile în web (descrise prin URL) se va nota la bibliografie și data accesării.
- Mai multe detalii despre citarea referințelor din internet se pot regăsi la:
  - http://www.writinghelp-central.com/apa-citation-internet.html
  - http://www.webliminal.com/search/search-web13.html
- Note de subsol se utilizează dacă referiți un link mai puțin semnificativ o singură dată; Dacă nota este citată de mai multe ori, atunci utilizați o referință bibliografică.
- Dacă o imagine este introdusă în text și nu este realizată de către autorul lucrării, trebuie citată sursa ei (ca notă de subsol sau referință - este de preferat utilizarea unei note de subsol).
- Referințele se pun direct legate de text (de exemplu "KVM [1] uses", "as stated by Popescu and Ionescu [12]", etc.). Nu este recomandat să folosiți formulări de tipul "[1] uses", "as stated in [12]", "as described in [11]" etc..
- Afirmațiile de forma "are numerous", "have grown exponentially", "are among the most used", "are an important topic" trebuie să fie acoperite cu citări, date concrete si analize comparative.
  - Mai ales în capitolele de introducere, "state of the art", "related work" sau "background" trebuie să vă argumentați afirmațiile prin citări. Fiți autocritici și gândiți-vă dacă afirmațiile au nevoie de citări, chiar și cele pe care le considerați evidente.
  - Cea mai mare parte dintre citări vor fi în capitolele de introducere "state of the art", "related work" sau "background".
- Toate intrările bibliografice trebuie citate în text. Nu le adăugați pur și simplu la final.
- Nu copiați sau traduceți niciodată din surse de informație de orice tip (online, offline, cărți, etc.). Dacă totuși doriți să oferiți, prin excepție, un citat celebru - de maxim 1 frază- utilizați ghilimele și evident menționați sursa. .
- Dacă reformulați idei sau creați un paragraf rezumat al unor idei folosind cuvintele voastre, precizați cu citare (referință bibliografică) sau cu notă de subsol sursa sau sursele de unde ați preluat ideile.

Trebuie respectat un singur standard de trimiteri bibliografice (citare), dintre următoarele alternative:

- APA (http://pitt.libguides.com/c.php?g=12108&p=64730)

- IEEE (`https://ieee-dataport.org/sites/default/files/analysis/27/IEEE%20Citation%20Guidelines.pdf`)
- Harvard (`https://libweb.anglia.ac.uk/referencing/harvard.htm`)
- Cu numerotarea referințelor în ordine alfabetică sau în ordinea apariției în text (de exemplu, stilul cu numere folosit de unele publicații ACM - `https://www.acm.org/publications/authors/reference-formatting`)

În Latex este foarte ușor să folosiți referințe într-un mod corect și unitar, fie prin adăugarea unei secțiuni `\begin{thebibliography}` (vezi la sfârșitul acestei secțiuni), fie printr-un fișier separat de tip bib, folosind comanda `\bibliography{}`, așa cum procedăm mai jos prin folosirea fișierului "bibliography.bib". În orice caz, în Latex va trebui să folosiți comanda `\cite{}` pentru a adăuga referințe, iar această comandă trebuie folosită direct în text, acolo unde vreți sa apară citația, ca în exemplele următoare:

- Articol jurnal: [3];
- Articol conferință: [1];
- Carte: [2];
- Weblink: [4];

Important: în această secțiune de obicei apar doar intrările bibliografice (adică doar listarea referințelor). Citarea lor prin comanda cite și explicații legate de ele trebuie facute în secțiunile anterioare. Citarea de mai sus a fost facută aici doar pentru exemplificare.

BIBLIOGRAPHY

[1] *Proc. 23rd International Symposium on Distributed Computing (DISC, Elche, Spain, September 2009)*, volume 5805 of *Lecture Notes in Computer Science*, Berlin, Germany, 2009. Springer.

[2] Michel Goossens, Frank Mittelbach, and Alexander Samarin. *The LaTeX Companion*. Addison-Wesley, Reading, Massachusetts, 1993.

[3] A. Amira H. Baali, H. Djelouat and F. Bensaali. Empowering technology enabled care using iot and smart devices: A review. *IEEE Sensors Journal*, 322(10):891–921, 1905.

[4] J. Silva-Martinez. Elen-325. introduction to electronic circuits: A design approach,. http://www.ece.tamu.edu/~spalermo/ecen325/Section%20III.pdf. Last accessed: 28 February 2018.

APPENDICES

A EXTRASE DE COD

. . .