GSoC 2025 Proposal
Public Courses on Github

Anas Iqbal

<*mohd.abd.6602@gmail.com*>

<*https://github.com/iqbalcodes6602/*>

<*omegaUp username: iqbal6602*>

Technical skills

I am a highly skilled and passionate software developer with a strong foundation in **full-stack development, backend systems, and cloud technologies**. My expertise spans multiple programming languages, including **Python, C++, JavaScript, SQL, and PHP**, along with frameworks like **Node.js, React.js, and Express.js**. I have extensive experience working with databases such as **MySQL and MongoDB**, and I am proficient in deploying applications using **Docker, Nginx,** and **Azure Cloud**.

My problem-solving skills are reflected in my achievements on **competitive programming** platforms like **LeetCode**, where I have solved **over 600 problems** with a **Knight** rating of **1971**, and **CodeChef**, where I hold a **4-star rating** with a **top 84th contest rank**. Additionally, I was a finalist in the Evolumin national-level hackathon.

At **Multiplicity Pepsales**, I am currently developing **microservices** for an **analytics dashboard** using **Elasticsearch** and **real-time push notifications** with **Flask and Pusher**, deploying them with **GitHub Actions on AWS**. During my **internship at Etechmy**, I built scalable applications with **React, Node.js, and Docker**. At **HUBX**, I developed **Express.js APIs** and **optimized MongoDB** queries for performance.

With a deep understanding of system architecture, real-time applications, and microservices, I am eager to apply my skills to challenging software development projects that push technological boundaries

Link to omegaUp merged PRs:

https://github.com/omegaup/omegaup/pull/8165

https://github.com/omegaup/omegaup/pull/8164

https://github.com/omegaup/omegaup/pull/8013

https://github.com/omegaup/omegaup/pull/8021

https://github.com/omegaup/omegaup/pull/8158

https://github.com/omegaup/omegaup/pull/8047

https://github.com/omegaup/omegaup/pull/8004

https://github.com/omegaup/omegaup/pull/8005

https://github.com/omegaup/omegaup/pull/7474

https://github.com/omegaup/omegaup/pull/7412

https://github.com/omegaup/omegaup/pull/7485

https://github.com/omegaup/omegaup/pull/7518

https://github.com/omegaup/omegaup/pull/7545

https://github.com/omegaup/omegaup/pull/7434

https://github.com/omegaup/omegaup/pull/7457

https://github.com/omegaup/omegaup/pull/7425

https://github.com/omegaup/omegaup/pull/7424

https://github.com/omegaup/omegaup/pull/8172 (pending)

https://github.com/omegaup/omegaup/pull/8173 (pending)

Education

- National Institute of Technology Rourkela, India
- Bachelors of Technology in Industrial Design

- December 2021
- June 2025

Motivation

Managing **public courses** on **GitHub** is a crucial step in making omegaUp's educational content more collaborative and scalable. Currently, course content is solely maintained by the **omegaUp staff**, limiting contributions from the broader community. By **migrating public course management to GitHub**, we enable **open-source contributions**, allowing **educators, students, and developers** to suggest improvements through **pull requests**.

This shift fosters community-driven content enhancement, ensuring that courses remain up-to-date, high-quality, and widely accessible. The **Mexican Olympiad in Informatics (OMI)** has already implemented a similar model, demonstrating its effectiveness in **crowdsourcing improvements**. By extending this approach to all **public courses**, we empower more contributors to **collaborate, refine, and expand** omegaUp's educational resources.

Additionally, integrating **GitHub-based course management** aligns with best practices in open-source development. With **continuous integration (CI) pipelines**, proposed changes can be **automatically validated**, maintaining content quality and consistency. The use of **GitHub's issue tracking and version control** will streamline feedback and iteration, reducing the maintenance burden on omegaUp staff.

Overall, this project supports **omegaUp's mission** of **expanding access to high-quality computer science education in Latin America** by making courses more interactive, community-driven, and sustainable.

Approach to follow

How the OMI Script Works-

OMI have uploaded the courses to a github repo, for the first time the course is added it has to be done manually, after that whenever a change is done to the content then the changes gets pushed automatically with the github api.

There are 3 important files responsible for deploying the changes in the public OMI course to omegaup -

1. Github actions CI/CD file - .github/workflows/continuous-integration.yaml - https://github.com/ComiteMexicanoDeInformatica/Curso-OMI/blob/main/.github/workflows/continuous-integration.yaml
2. Upload.py file in the utils directory (utils directory is omegaup's omegaup-deploy

repository)- https://github.com/omegaup/omegaUp-deploy/blob/89c78d38de4396343c3d6ee6bf5f696257f8af3b/upload.py

3. Problems.json in the root folder- https://github.com/ComiteMexicanoDeInformatica/Curso-OMI/blob/main/problems.json

Whenever a new commit is pushed the Github actions start executing, it runs a set of steps, one of such step is

```
- name: Deploy to omegaUp

  run: pipenv run python3 ./utils/upload.py --ci --verbose

  if: github.ref == 'refs/heads/main'
```

As evident this step then runs the upload.py file from the utils folder (which is omegaup-deploy repo). This upload.py file runs a bunch of functions. One of such functions is getting the list of problems, which have been changed. This function scans the problems.json file in the root directory and then checks if any of the problems have been updated. If updated it adds it to the list of problems that need to be uploaded to omegaUp else skips it.

These are sample logs from the Github actions of a problem being uploaded to omegaup.

```
2025-03-30 18:48:12,366: Loading problems...
142025-03-30 18:48:12,367: Loading git diff.
152025-03-30 18:48:12,370: Loading Demo de Objetivo.
162025-03-30 18:48:12,370: Loading Contest Start.
172025-03-30 18:48:12,370: No changes to Contest Start. Skipping.
182025-03-30 18:48:12,371: Uploading problem: Demo de Objetivo
```

As it's evident that the Demo de Objetivo problem had some changes hence its the only one to be uploaded the other problem have been skipped i.e. **only the changed problems get uploaded.**

4. Updating course details through PR itself -
    a. We can replicate what the problems.json is doing for problems similarly on the course level itself.
    b. All we need to do is define a settings.json file for Public Course which we are considering.
    c. But doing so we also need to update the **upload.py** file in the **omegaup-deploy** repository to consider the root settings.json file as the settings for each Course.
    d. Thus we make use of the **update course api** we would update each course whenever a PR is submitted.

Overall plan for the project-

The overall plan is to replicate what the OMI is doing for their course, we need to do this for all the omegaup courses. We just need some modification in the script -

1. The folder structure would be same, with-
    a. The ci/cd file for the Github Action
    b. omegaup-deploy repo being connected to the course repo and,
    c. a problems.json file at the root of the repo
2. If we are allowing the text-only-lectures to be updated as well, then the ci/cd file will need some changes , since currently the file runs a set of tests for the test cases for problems, but since there are no test cases in the text-only-lectures it would give error.
3. The repository structure -
    a. OMI has only one course in one repository.
    b. But we can put all the courses in the same repository.
    c. The only **change** needed would be in the **folder structure of repo** and **problems.json** file. The problems.json file just reflects the entire repo's folder structure.
        i. OMI just has one course so all the folders inside the repos represent the course content. Currently the OMI's folder structure is like this. (problem.json will have similar structure)

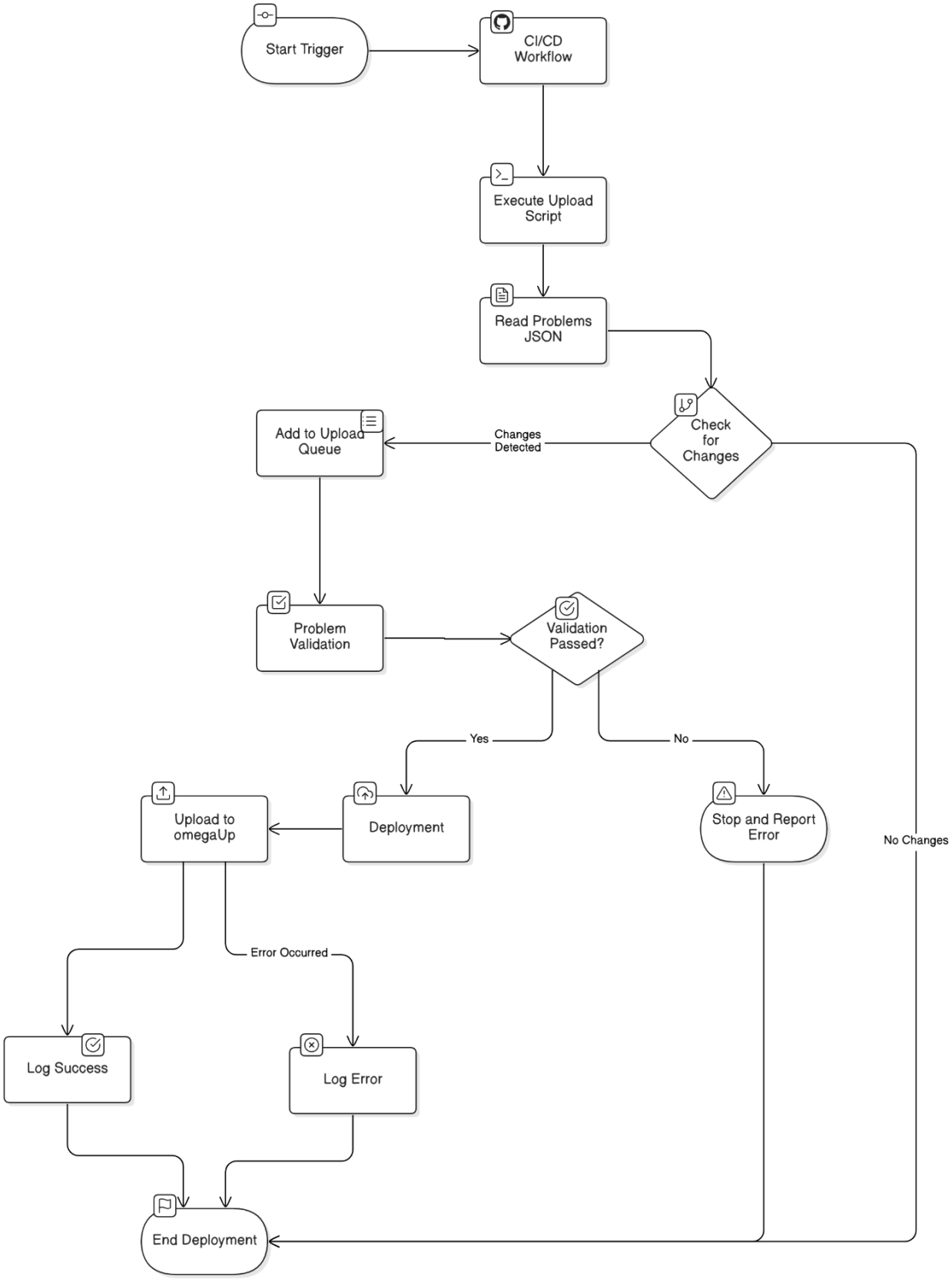
```
Course-Content-1/
      Problem A
      Lecture B
Course-Content-2/
      Problem C
      Lecture D
```

        ii. But since we are having multiple courses our each folder inside the repo will represent each Course. So our folder structure would be like this. (problem.json will have similar structure)

```
Course 1
     Course-Content-1/
          Problem A
          Lecture B
     Course-Content-2/
          Problem C
          Lecture D
Course 2
     Course-Content-1/
          Problem E
          Lecture F
     Course-Content-2/
          Problem G
          Lecture H
```

```
and so on for all the courses
```

Automated Deployment Pipeline for omegaUp Courses



Detailed project description

Current Status of omegaUp Public Courses

Currently, omegaUp provides a platform for programming education and competitive programming contests. However, managing public courses within omegaUp is largely a manual process. This makes version control, collaboration, and deployment consistency challenging. Unlike problem submissions, which follow structured workflows, public courses lack an automated pipeline for updates, version tracking, and review before publishing.

Step-by-Step Process to Complete the Project

Step 1: Uploading all the Omegaup Public Courses to new Github Repository:

We would write a python script to automate the process. The script would fetch all the courses for the user and all the course content and problems in it. We can make use of Course API, which would give us the list of courses owned by a particular user. In our case it's omegaUp courses admin. Based on this we can replicate the folder structure in the Github Repo itself to represent all the courses.

This is the initial step where we need to transfer all the lectures, homework, problems and all the course content of each Omegaup Public Course on Github -

- Each content in each course has a download zip file option.
- We will download each content from there and them upload it to the github repository

After doing it for all the course the Github repo will look something like this

```
-  .github/workflows/  # CI/CD configurations
-  Introducción a C++/
      -  0. - Introducción al Curso/
            - A. Introducción a omegaUp/
            - B. Curso de Introducción a C++ - Conoce a tu Instructor/
      -  1-A. ¡Hola Mundo!•/
            - A. Curso de Introducción a C++ - ¡Hola Mundo!/
            - B. ¡Hola Mundo!/
      -
-  Introducción a Algoritmos - Parte I/
      -  1. Búsqueda Binaria/
            - A. Búsqueda Binaria/
            - B. Implementar la Búsqueda Binaria de un Arreglo/
            - C. Tiempo de Ejecución de la Búsqueda Binaria/
      -  2. Práctica de Búsqueda Binaria•/
            - A. Alicia y las llaves doradas de las puertas/
            - B. Posicion Fibonacci/
```

```
  -  problems.json  # Defines the problems in the courses to be updated
```

The problems.json file will list all the problems which we need to watch for, i.e. when any of the mentioned problems in the file gets changed they will be deployed to the omegaUp website.
The problem.json will look something like this

```
{
  "problems": [
    {
      "path": "Introducción a C++/0-Introducción al Curso/A. Introducción a
omegaUp/ "
    },
    {
      "path": "Introducción a C++/0-Introducción al Curso/B. Curso de
Introducción a C++ - Conoce a tu Instructor/"
    },
    {
      "path": "Introducción a C++/1-A-¡Hola Mundo!/A. Curso de Introducción a
C++ - ¡Hola Mundo!/"
    },
    {
      "path": "Introducción a C++/1-A-¡Hola Mundo!/B. ¡Hola Mundo!/"
    }
  ]
}
```

The path is defined like - `Course-Name/Content-Name/ Problem`

This way we will watch for each file for changes when a new PR is submitted, hence we will decide whether we need to deploy it to omegaUp or not.

This path goes to the problem which is actually the zip file for the problem and it contains all the necessary files needed for this problem like cases, tests etc
Here is a sample structure for a problem

```
cases/
settings.distrib.json
settings.json
solutions/
statements/
```

The settings.json file contains all the necessary configuration for the problem including the problem alias which would be used in API calls.

Step 2: Set Up CI/CD Workflow

The repository includes a GitHub Actions workflow (`.github/workflows/main.yml`) that:

- Triggers on pull requests and commits to the `main` branch to ensure only reviewed content is published.
- Runs validation scripts to check course structure.
- Uses `omegaUp-deploy` scripts to interact with the omegaUp API.
- Deploys approved course changes to omegaUp upon merging.

Step 3: API Integration with omegaUp

- Uses `OMEGAUP_API_TOKEN` (stored as a GitHub secret) for authentication.
- Executes `upload.py` from `omegaUp-deploy` to push course changes.
- Ensures that only authorized maintainers can trigger deployments.
- But if the authorisation fails, it would throw an error and would not deploy the problem.

Step 4: Implement Review and Approval Mechanism

- Contributors submit a pull request (PR) with new or updated course content.
- The PR is reviewed manually or via automation to check:
    - Formatting consistency
    - Metadata correctness
    - Code integrity (if applicable)
- If approved, merging triggers the deployment step.

Step 5: Handling Course Updates

- Contributors can update existing courses by modifying content files.
- The CI/CD pipeline detects changes and redeploys only the modified sections.

- Versioning via GitHub ensures rollback capability if issues arise.

Step 6: Deployment to Public Courses

- Once changes pass validation, `upload.py` pushes them to omegaUp.
- The public branch is updated with the latest course version.
- The repository serves as the **single source of truth** for course content.

Upon full implementation, public courses on omegaUp will:

1. **Benefit from GitHub's collaboration model** (pull requests, reviews, history tracking).
2. **Be automatically deployed** to omegaUp using a secure and structured pipeline
3. **Allow community contributions** while maintaining quality control
4. **Reduce manual workload** for course maintainers, ensuring faster updates.

By leveraging the Mexican Olympiad's existing CI/CD model, we ensure a **scalable, efficient, and structured approach** to managing public courses on omegaUp.

Miscellaneous

Additionally there are some of aspects with respect to type of contents which need to be covered as well -

a. **Text only lectures** - omegaUp public courses have problems which are only texts and do not have any problems to solve like introduction OMI Course - https://omegaup.com/course/Curso-OMI/assignment/COMI-intro#problems/COMI-Objetivo , these kind of content in OMI public do not allow repo as they are not mentioned in the problems.json file in the root directory here - https://github.com/ComiteMexicanoDeInformatica/Curso-OMI/blob/main/problems.json , so we have to avoid the Test step in the github action workflow file for these kind of content in the first run

b. **Problems** - omegaUp courses have problems as content as well, these have test cases which also can be updated, hence we need to add a step in the CI CD pipeline in Github Actions workflow file to test these. This testing step is also present in the OMI Public Course.

Overall we need to decide whether we want to allow updating the text-only-lectures to be updated or not. Accordingly we can decide if we need to update the ci cd to handle these cases. If we choose to not update the text-only-lectures then we need to upload all these types of lectures first, without the test. After this step we can upload all the Problems and then add them to problems.json file in order to check for updates and let them get updated when a new PR gets merged with update for these problems.

Alternatives considered

Alternative 1: Allowing External Contributors to Suggest Changes Through a Custom omegaUp Interface

Instead of using **GitHub**, omegaUp could develop an internal **content contribution system**, allowing users to suggest edits directly on the platform.

Advantages:

- **Seamless Integration:** Users wouldn't need to leave omegaUp to contribute, maintaining a unified experience.
- **More User-Friendly:** Some contributors may find submitting changes via a custom omegaUp interface easier than using GitHub.
- **Full Control Over Features:** omegaUp could design the system to match its exact needs, rather than adapting to GitHub's workflows.

Disadvantages:

- **Higher Development Effort:** Building a fully functional contribution system would require significant engineering resources.
- **Lack of Established Version Control:** Unlike GitHub, a custom system would likely lack advanced version control, making it harder to track and manage content changes.
- **Scalability Issues:** Maintaining a separate contribution platform may become difficult as more courses and contributors join.

Alternative 2: Keeping Course Management Limited to omegaUp Staff

This approach would maintain the current workflow, where only **omegaUp staff** manage public courses directly on the platform without external contributions.

Advantages:

- **Quality Control:** Since only omegaUp staff handle course updates, the risk of low-quality or incorrect content being added is minimized.
- **Simplified Workflow:** No need to set up GitHub-based content management, reducing complexity for maintainers.
- **Avoids External Dependencies:** No reliance on GitHub for managing course content, ensuring all data remains within omegaUp's infrastructure.

Disadvantages:

- **Limited Contributions:** Prevents open-source collaboration, reducing the potential for improvements and refinements from the community.
- **Slower Updates:** Course updates depend entirely on the availability of omegaUp staff,

potentially delaying fixes or enhancements.
- **Higher Maintenance Burden:** Staff members bear full responsibility for content management, increasing their workload over time.

While both alternatives offer certain advantages, **migrating public course management to GitHub** provides the best balance between **openness, efficiency, and sustainability**. It enables **community-driven improvements** while leveraging **GitHub's robust version control and CI/CD features**, ensuring high-quality content with minimal overhead for omegaUp staff.

Security impact

Since this project involves **migrating public course management to GitHub**, it does not introduce any direct risks related to **cheating or unfair advantages** in competitive programming on omegaUp. However, certain security considerations need to be addressed:

1. Preventing Unauthorized or Malicious Edits

- **Pull Request Review Process:** All proposed changes to course content will go through a review process before being merged, preventing users from injecting misleading or incorrect information.

- **Maintainer Approval:** Only trusted maintainers will have merge permissions, ensuring that only high-quality contributions are accepted.

2. Avoiding Information Leaks

- **No Exposure of Private Course Data:** This system is only for public courses, so private courses or sensitive user data remain unaffected.

- **GitHub Permissions Management:** Access to course repositories will be restricted to prevent unauthorized edits outside of the pull request workflow.

3. Preventing Abuse of GitHub for Spoilers or Answer Sharing

- **Content Moderation:** GitHub repositories will be monitored to **prevent users from uploading solutions or spoilers** for omegaUp's problem sets.

- **Reporting Mechanism:** A **structured process** will be in place to report and remove any inappropriate or unfair content.

By enforcing strict **review, approval, and moderation policies**, this project does not introduce new security vulnerabilities. Instead, it enhances omegaUp's **content quality and accessibility**

while ensuring a **fair learning environment** for all users.

Deployment plan

The deployment process will be fully automated using GitHub Actions and omegaUp's API. Since this project does not involve changes to omegaUp's backend or database, the deployment will follow a normal deployment cycle.

Testing plan

Automated Testing

To ensure the reliability of the CI/CD pipeline and the correctness of course content, we will implement:

1. **CI/CD Pipeline Tests**
   - The GitHub Actions workflow will **validate course structure** before deployment.
   - Each PR will trigger:
     - **Syntax checks** for metadata files (`problems.json`, YAML, etc.).
     - **Validation of problem files** (ensuring test cases exist, checking input/output format).
     - **CI checks for omegaUp API requests** (ensuring valid authentication and request format).
2. **Content Verification Tests**
   - Before deployment, scripts will check:
     - If all referenced problem paths exist in the repository.
     - If test cases are properly structured for problem content.
     - If static lecture files are correctly formatted.

Manual Testing

Since course content can include complex elements, some manual review is required:

1. **Pre-Merge Review**
   - PRs will be reviewed manually to check:
     - **Correct course structure** (folders, filenames).
     - **Properly formatted lecture content** (Markdown, plain text).
     - **Accurate metadata for problem descriptions and test cases**.
2. **Post-Deployment Verification**
   - After each deployment, a **sample test course** will be checked on omegaUp to:
     - Ensure the **course structure is preserved**.
     - Validate that **problems work correctly** with expected test cases.
     - Confirm that **lecture materials display properly**.

Schedule

The project will be divided into three milestones, ensuring steady progress while keeping the implementation structured and manageable.

1. **Milestone 1 [May 10 – June 10] - Setting Up the Repository & Basic CI/CD Implementation**
    - Finalize repository structure (single repo vs. multiple repos).
    - Implement GitHub Actions to deploy static lecture content automatically.
    - Set up omegaUp API authentication for secure course updates.
    - Ensure basic course deployment from GitHub to omegaUp is functional.

2. **Milestone 2 [June 10 – July 19] - Expanding CI/CD for Problems & Handling Updates**
    - Add automation for problem content (test cases, metadata).
    - Ensure CI/CD first deploys static lectures, then problems.
    - Implement logic to detect and deploy only modified content.
    - Improve error handling and rollback mechanisms.

3. **Milestone 3 [July 19 – August 19] - Final Testing, Optimization & Documentation**
    - Run full end-to-end tests with multiple courses.
    - Optimize CI/CD pipeline for performance and reliability.
    - Document setup, best practices, and troubleshooting.
    - Ensure the system is ready for long-term maintenance.

(**for the reviewers**, if you want to know how will things works when we have successfully deployed the changes, please follow these steps and **submit a PR** to my repo to understand the workflow)

Proof of Concept**:- (you can try to raise a PR now itself to check)**

I have set up a proof-of-concept repository that mirrors the structure and workflow used by the Mexican Olympiad in Informatics (OMI) for managing contests. The OMI repo utilizes GitHub, GitHub Actions (CI/CD), and omegaUp's API to automate problem deployment.

I have created a sample course "Demo Course" in which we can simulate the entire workflow which

we want to achieve in this project. All the course content is available on Github, and we can manage and update the course content through PRs submitted by contributors

Inspired by this -

- I created a demo course, this is the link to course on omegaUp (this is public course you can join by going to this link) -
  - https://omegaup.com/course/omi-public-course/

- I have also created a sample GitHub repository to simulate a public course deployment pipeline.
  - https://github.com/iqbalcodes6602/omi_public_course

- I have configured a CI/CD workflow using GitHub Actions to automate course deployment.
  - https://github.com/iqbalcodes6602/omi_public_course/blob/main/.github/workflows/continuous-integration.yaml

- I used a separate GitHub account to simulate external contributors submitting courses.
  - Update the course content and submit a PR
    - https://github.com/iqbalcodes6602/omi_public_course/pull/2
  - You can also create a PR with updated content and test it-
    - https://github.com/iqbalcodes6602/omi_public_course/compare

- Integrated with the omegaUp API to push course changes automatically upon approval.
  - Once the PR is merged, view the changes in omegaUp course, the admin of the problem can also view the latest version and how it was deployed, in this case automatically deployed when PR got merged.

Problem settings Demo de Objetivo – Go to problem

Help

Problem settings | Edit statement | View version history | Edit Solution | Admins | Tags | Download | Delete

Show diff Files Only show changes

	Commit date	Account name	Problem version	Commit message
◉	3/23/2025, 11:34:58 PM ecccc707c	iqbal6602	2c03a569	Deployed automatic 9adf090819a5e29fe
○	3/23/2025, 11:16:58 PM 773566b1	iqbal6602	2c03a569	Deployed automatic 656c035d9d6f07ef0
○	3/23/2025, 11:04:16 PM 0e512f16	iqbal6602	2c03a569	Deployed automatic c7d7578645e17a01
○	3/23/2025, 11:01:55 PM 9524c3ff	iqbal6602	8d9d498c	Initial commit

- .gitattributes
- .gitignore
- settings.distrib.json
- settings.json
- statements/es.markdown