**ACKNOWLEDGEMENT**

Any achievement does not depend solely on the individual efforts but on the guidance, encouragement and co-operation of intellectuals, elders and friends. A number of personalities, in their own capacities have helped us in carrying out this mini project work. We would like to take this opportunity to thank them all.

We would like to express my profound thanks to **Sri. G Dayanand,** Chairman, Sapthagiri College of Engineering Bangalore, for his continuous support in providing amenities to carry out this Mini Project.

Special Thanks to **Dr. N. Srinivasan,** Director, Sapthagiri College of Engineering Bangalore, for his valuable suggestion.

Also we would like to express our immense gratitude to **Dr. SHIVABASAPPA K L** Principal, Sapthagiri College of Engineering Bangalore, for his help and inspiration during the tenure of the course.

We also extend our sincere thanks to **Dr. Yogish H K**, Professor and Head, Department of Computer Science and Engineering, Sapthagiri College of Engineering, for his constant support.

We would like to express our gratitude to **Mr. Ramangouda. S. Patil,** Assistant professor, Department of Computer Science and Engineering, Sapthagiri College of Engineering, for their timely advice on the mini project and regular assistance throughout the work.

We also extend our sincere thanks to all the **Faculty members** and **supporting staff** Department of Computer Science and Engineering, Sapthagiri College of Engineering, for their constant support and encouragement.

Finally, we thank our parents and friends for their moral support.

**ABHISHEK VARMA (1SG15CS004)**

**DYLAN SALDANHA (1SG15CS026)**

# **ABSTRACT**

WeByte is a programming colosseum that allows programmers to battle against other programmers in competitive programming contests. This Flask based website can be used by an individual who wishes to host a competitive coding event in one's locality, school, college or university. Companies can also host this website on their servers and use it for competitive coding tests for hiring purposes, instead of using a middleman to conduct the tests.

Some of the noticeable features that one would be able to find here are, webpages designed to be user friendly and intuitive. Authentication of user's email during signup. Excellent test coverage. Will work on Python 3.6 and above. Allows execution of Python programs. Finally, every user's data stored using SHA256 for privacy concerns.

**Table of Contents**

[**CHAPTER 1** 1](#_Toc531302060)

[**INTRODUCTION** 1](#_Toc531302061)

[**1.1 Overview of the Mini Project** 3](#_Toc531302062)

[**1.2 Aim of the Mini Project** 4](#_Toc531302063)

[**CHAPTER 2** 5](#_Toc531302064)

[**REQUIREMENT SPECIFICATION** 5](#_Toc531302065)

[**2.1 Functional Requirements:** 5](#_Toc531302066)

[**2.2 Non-Functional Requirements:** 5](#_Toc531302067)

[**2.3 HARDWARE AND SOFTWARE REQUIREMENTS:** 6](#_Toc531302068)

[**CHAPTER 3** 7](#_Toc531302069)

[**DESIGN** 7](#_Toc531302070)

[**CHAPTER 4** 8](#_Toc531302071)

[**IMPLEMENTATION** 8](#_Toc531302072)

[**Aggregation and segregation of data:** 8](#_Toc531302073)

[**Entry into the database:** 8](#_Toc531302074)

[**Data Security:** 8](#_Toc531302075)

[**I. Sign Up** 8](#_Toc531302076)

[**II. Sign In** 9](#_Toc531302077)

[**III. Upcoming and Ongoing Battles** 9](#_Toc531302078)

[**IV. Battleground** 9](#_Toc531302079)

[**V. Results** 9](#_Toc531302080)

[**CHAPTER 5** 11](#_Toc531302081)

[**TESTING** 11](#_Toc531302082)

[**CHAPTER 6** 13](#_Toc531302083)

[**SNAPSHOTS** 13](#_Toc531302084)

[**CHAPTER 7** 18](#_Toc531302085)

[**CONCLUSION AND FUTURE WORKS** 18](#_Toc531302086)

[**CONCLUSION** 18](#_Toc531302087)

[**FUTURE WORKS** 18](#_Toc531302088)

[**BIBLIOGRAPHY** 19](#_Toc531302089)

# **List of Figures**

|  |  |  |  |
| --- | --- | --- | --- |
| **Sl. No.** | **Figure No.** | **Title of figure** | **Page No.** |
| 1 | 3.1 | Flowchart for Code Arena | 7 |
| 2 | 6.1 | Home Page | 13 |
| 3 | 6.2 | Sign Up Page | 13 |
| 4 | 6.3 | User Login | 14 |
| 5 | 6.4 | Forgot Password Page | 14 |
| 6 | 6.5 | Forgot Password Confirmation | 15 |
| 7 | 6.6 | Upcoming Battles Page | 15 |
| 8 | 6.7 | Battleground Page | 16 |
| 9 | 6.8 | Battleground Editor | 16 |
| 10 | 6.9 | Battle Occurred Page | 17 |
| 11 | 6.10 | Results Page | 17 |

# **List of Tables**

|  |  |  |  |
| --- | --- | --- | --- |
| **Sl. No.** | **Table No.** | **Title of table** | **Page No.** |
| 1 | 5.1 | Testing Table | 11 |

# **CHAPTER 1**

## **INTRODUCTION**

Web technology refers to the means by which computers communicate with each other using markup languages and multimedia packages. It gives us a way to interact with hosted information, like websites. Web technology involves the use of hypertext markup language (HTML) and cascading style sheets (CSS). In order to make websites look and function a certain way, web developers utilize different languages. The three core languages that make up the World Wide Web are HTML5, CSS, and JavaScript.

In the IT world, the internet is an essential platform, whether its for developing or for consumer use. When developing a website, typically three main languages come into play. These languages are JavaScript, CSS, and HTML. HTML is the backbone of most webpages. Essentially, it is used to create the structure of how a specific website would look like, from the headings, to the paragraphs, the body, links, and even images. Markup languages are the languages in which the web is written. The most common markup language used is HTML, which uses tags to annotate text so that a computer can then manipulate the text. Most markup languages are human readable, and use annotations that are distinguishable from the annotated text. There are many different kinds of markups and languages, but all are consistent in the way in which they annotate documents.

Hypertext is defined as the arrangement of information inside a database that allows the user to receive information and to navigate from one document to another by clicking on highlighted words or pictures inside the primary document. Hypertext is the base of the World Wide Web, because it enables user to click on other links to get more information. Hypertext is a term used for all links, whether it appears as texts or other graphical part.

HTML is the conventional markup language used to create and edit web pages and web applications. HTML is used for creating the basic structure of a website. HTML consists of different elements preceded by an opening tag, <tag>, and a closing tag, </tag>. The content between the tags, <html> and </html>, is the content of the webpage. The content between the tags, <head> and </head>, is the title of the webpage. This text is displayed between the <title> and </title> tags. The content between the tags, <body> and </body>, is the main content of the webpage. The content can include links, paragraphs, headings, and various other elements.

CSS is a style sheet language standard set by W3C (World Wide Web Consortium) used to create and edit the visual presentation of web pages. CSS allows web developers to isolate a web page's content and visual styles into separate documents and gives better page layout control. An external CSS sheet is generally linked to HTML and XHTML, it also can be linked to XML, SVG, and XUL. HTML and JavaScript, with CSS, is a vital part of technology used by the majority of interfaces for websites. This is also used in interfaces for mobile devices making the websites more engaging.

CSS can be incorporated with HTML in 3 different ways; Inline, Internal, and External.

1. **Inline styles** add style to a single element on the page by placing 'style' after the element you wish to be styled.

*Ex: h2 style = "color: blue"*

1. **Internal styles** create a style for a single document because the CSS is stored in the head of the HTML document. Internal styles are placed using a *<style>* tag around all style selectors.

*Ex: <style>  
body {background-color: white;}  
/\*This is a comment!  
'Body' is the selector,  
'background-color' is the declaration\*/  
h2 {color: blue;}  
</style>*

1. **External style sheets** exist in separate documents from HTML documents, allowing for better organization of style and structure. An external style sheet can be linked to all HTML documents making up a web site, allowing a web developer to style the entire site (all pages) using one document.
2. **JavaScript** is a scripting language that is used along with HTML and CSS as the three core components of the World Wide Web. JavaScript has first-class functions and is used in most websites. JavaScript does not have any I/O which means that it has to be embedded in the host environment. JavaScript is also used in PDF documents, game development, and desktop and mobile applications. JavaScript is most commonly used to make DHTML by adding client-side behavior to HTML pages.
3. **PHP** stands for Hypertext Preprocessor (no, the acronym doesn't follow the name). It is an open source, server-side, scripting language used for the development of web applications. By scripting language, we mean a program that is script-based (lines of code) written for the automation of tasks.
4. **Python** is an interpreted high-level programming language for general-purpose programming. We have used the flask framework.
5. **Flask** is a micro web framework written in Python. It is classified as a microframework because it does not require particular tools or libraries. It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions.

**1.1 Overview of the Mini Project:**

WeByte’s story has been about perseverance and focus towards one goal - making technical recruitment simple and efficient. And this journey has been no walk in the park - we have had our share of challenges and we know there are more to come in the future.

An online programming platform where anyone in the world with access to the internet can solve problems to their heart’s content and know where they stand in the current 21st century. We provide developers across the globe with an opportunity to exercise or demonstrate their problem skills and compete with each other. This platform can be utilized by companies across the globe to find talented developers to suit their need and culture of the organization.

**1.2 Aim of the Mini Project:**

Our mini project aims to facilitate young, budding, creative programmers with an Open Source Programming Colosseum, where they can battle to their heart’s content and emerge victorious among the rubble. In addition to that we also aim to allow companies to find the Cream of the Crop.

From our past experiences we have noticed how hard it is for universities to organize programming events, therefore we aim to simplify the process of hosting a platform for competitive programming events held at universities. Providing a source of entertainment for the intellectuals.

# **CHAPTER 2**

## **REQUIREMENT SPECIFICATION**

A software requirement definition is an abstract description of the services which the

System should provide, and the constraints under which the system must operate. It should only specify the external behavior of the system.

### **2.1 Functional Requirements:**

Functional Requirements defines the internal working of the software, i.e., the calculations, technical details, data manipulation and processing and other specific functionality that show how the cases are to be satisfied and how they are supported by non-functional requirements, which impose constraints on the design or the implementation.

The following are the Functional requirements:

1. The ability to perform correct operation when the corresponding keys are pressed.

2. The ability to display the correct information as stored in database.

3. When the corresponding key is selected, the corresponding operation should be performed.

4. Ability to register students on the procurement of correct information.

### **2.2 Non-Functional Requirements:**

Nonfunctional requirements are requirements which specify criteria that can be used to judge the operation of the system, rather than specific behaviors. This should be contrasted with functional requirements that specify specific behavior or functions. Typical nonfunctional requirements are reliability and scalability. Nonfunctional requirements are “constraints”, “quality attributes” and “quality of service requirements”.

It has the following attributes:

1. Reliability

2. Safety

3. Security

4. Availability

5. Dependability

### **2.3 HARDWARE AND SOFTWARE REQUIREMENTS:**

#### Hardware Requirements:

* 128 MB of RAM, 256 MB recommended.
* 110 MB of hard disk space required, 40 MB additional hard disk space required for installation (150 MB total).

#### Software Requirements:

The software requirements are as follows: -

**Development Platform:** WINDOWS 10

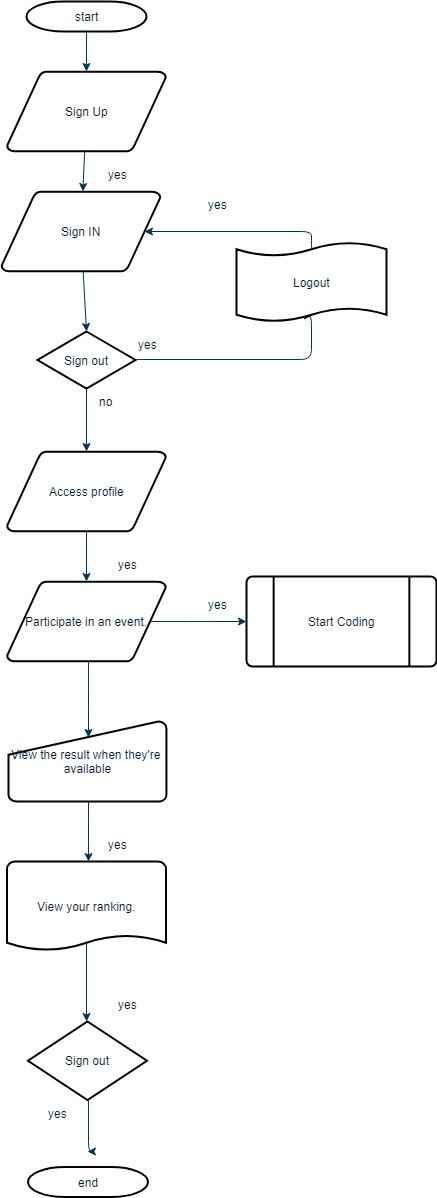
**Language :** HTML, CSS, JAVASCRIPT, PHP

**Tool :** Web Browser such as Chrome or Firefox

# **CHAPTER 3**

## **DESIGN**

Data flow design is as shown below - covering the flow of the data in the system. It describes the relation between user input and the system behavior.



**Figure 3.1: Flow chart for Code Arena**

# **CHAPTER 4**

## **IMPLEMENTATION**

### **Aggregation and segregation of data:**

* The data must be stored in an efficient way.
* Mismatching of has been avoided. For example, the email of a user cannot store his/her name.
* Data has been found to be compatible with respect to the fields of information.

### **Entry into the database:**

* The data has been entered into the database properly.
* Insertions made through the front end were reflected in the database.
* The data in the form of a solution to a problem can be submitted by the user and the result can be viewed instantly.

### **Data Security:**

* Data has been properly validated. For example, an empty field has to be filled first and an incorrect information does not yield the necessary result.
* Incorrect data must not be entered into the database. For example, the Email ID provided by the user should be a valid one.
* The data stored on our databases undergo a hashing process thereby ensuring security of user credentials.

The main actions are performed through as follows:

Module Name:

**I. Sign Up**: **Objective**: - Allows to register a user to the programming platform.

**Input**:- Details related to the user such as the Email ID.

**Output**:- Record successfully inserted and allows to sign in to the system if successful and email verification is done.

**Description**: - The registration page is basically a formal introduction of the user. Contains details such as the Email ID. Users can then use this information to identify themselves and start programming using their account.

**II. Sign In**: **Objective**: - Allows the user to access his account and start coding.

**Input**:-Email ID and Password.

**Output**:- User successfully authenticated and is allowed to access his account.

**Description**: - This module makes sure that the user is who he claims to be as there is a possibility of data being compromised by phishing attacks and snooping. Therefore we have made sure that not even the database administrators have user’s valuable information within their reach. This is made possible by using an SHA-256 hashing algorithm.

**III. Upcoming and Ongoing Battles**: **Objective**: -To display the vast array of competitions (hiring and pleasure driven) that are meant to indulge the user in the world of programming.

**Input**:- Authentication of the sign page.

**Output**:- List of competitions successfully rendered.

**Description**: - This page provides the details with respect to the competitions that are available to the user to participate in. It helps the user to be aware of all the scope of opportunities that the user can grab. It shows both types of competitions, i.e., hiring challenges and pleasure driven competitions.

**IV. Battleground**: **Objective**: -To provide the user with an enriching experience that facilitates one to come up with an efficient solution to the problems given.

**Input**:- User needs to submit his solution in order to verify if he/she has solved the problem.

**Output**:- Shows the result of the submitted solution by indicating if the user has passed all the necessary test cases to assert his dominance over the problem.

**Description**: This is where the programmer is allowed to analyze the problem and come up with an efficient solution that passes all the necessary test cases.

**V. Results**: **Objective**: -To inform the user whether he has left all the other competitors in the dust, emerging victorious in the world of data structures and algorithms.

**Input: -** User needs to submit the solution in order to obtain the result.

**Output: -** Displays the ranks of all the participants along with their scores

**Description: -** This is where the user gets an opportunity to know where he/she stands in the competition. The user can get an insight on the areas he/she needs to improve on.

# **CHAPTER 5**

## **TESTING**

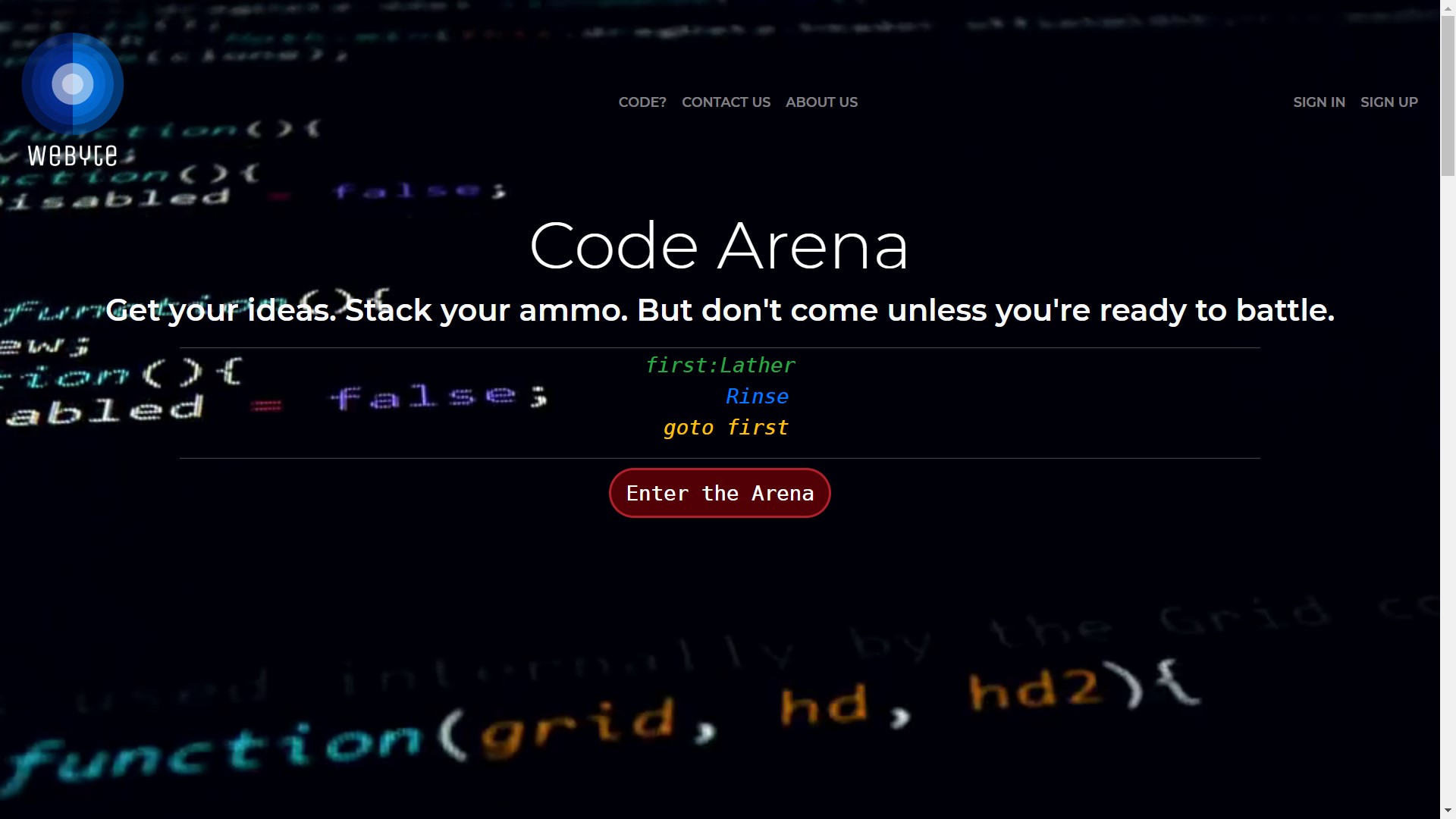
Testing in general means validation and verification. It shows that the system conforms to its specifications and system meets all expectation of the user.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Sl. No** | **Functions with**  **parameters under**  **test** | **Expected Result** | **Actual Result** | **Remarks** |
| 1. | When correct credentials are entered by user. | The user must be able to access his account and start coding. | The user was able to access his account and start coding. | Pass |
| 2. | When the user logs in. | The user should be redirected | The user is redirected to the upcoming battle page. | Pass |
| 3. | When an unspecified data is pressed. | Neither insertion nor deletion should occur. | None of them occurred. | Pass |
| 4. | When user checks the competition results. | A ranking table has to be displayed along with the scores. | A ranking table was displayed along with the scores. | Pass |
| 5 | When user submits a solution that passes all the test cases. | He should be informed that the solution is accepted. | He should be informed that the solution is accepted. | Pass |
| 6 | When user tries to just print the output. | Solution should not be accepted. | Solution is accepted | Fail |
| 7 | When user tries to logout and login within 60 secs. | User is redirected to login page. | User is logged in. | Fail |
| 8 | User enters infinite loop in code. | Server should detect infinite loop and break. | Server stuck in infinite loop and unable to serve further requests. Possible Denial of Service to other users. | Fail |

**Tab 5.1 Testing Table**

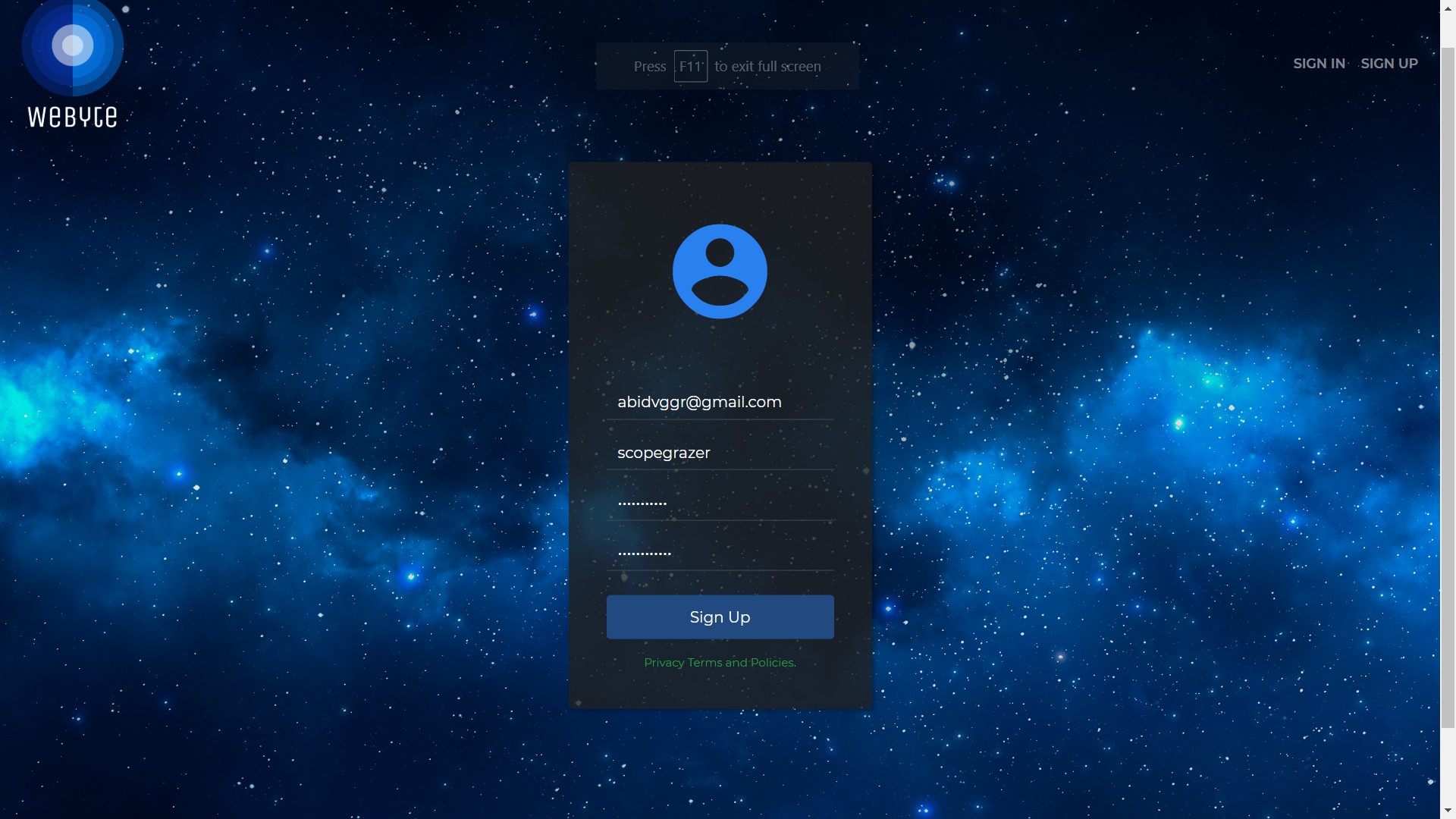
# **CHAPTER 6**

## **SNAPSHOTS**



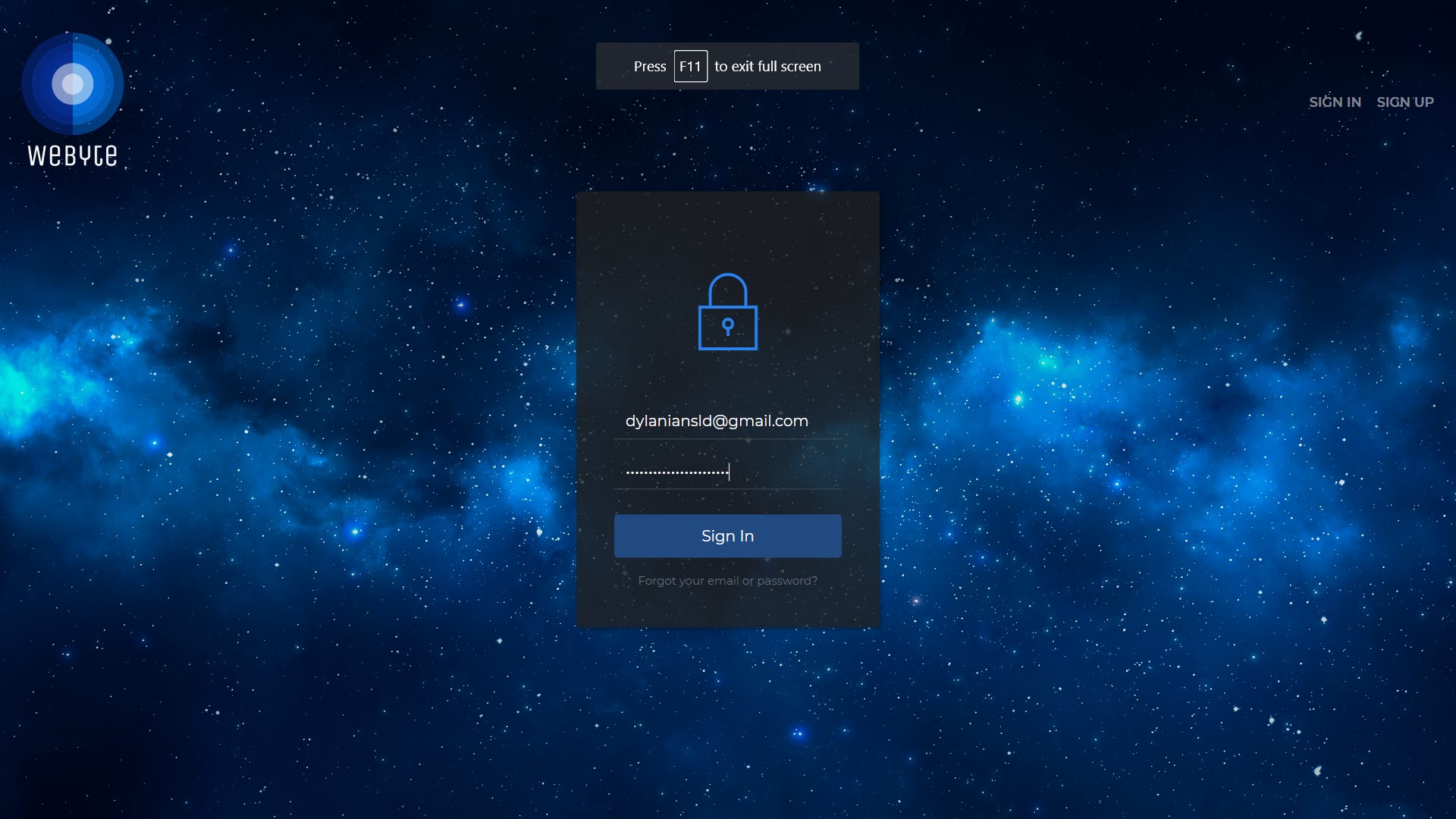
**Fig 6.1: Home Page**

This is the Home Page of the coding platform. It gives the user a good feel about our mission, goal and motives that we have in mind for our users.



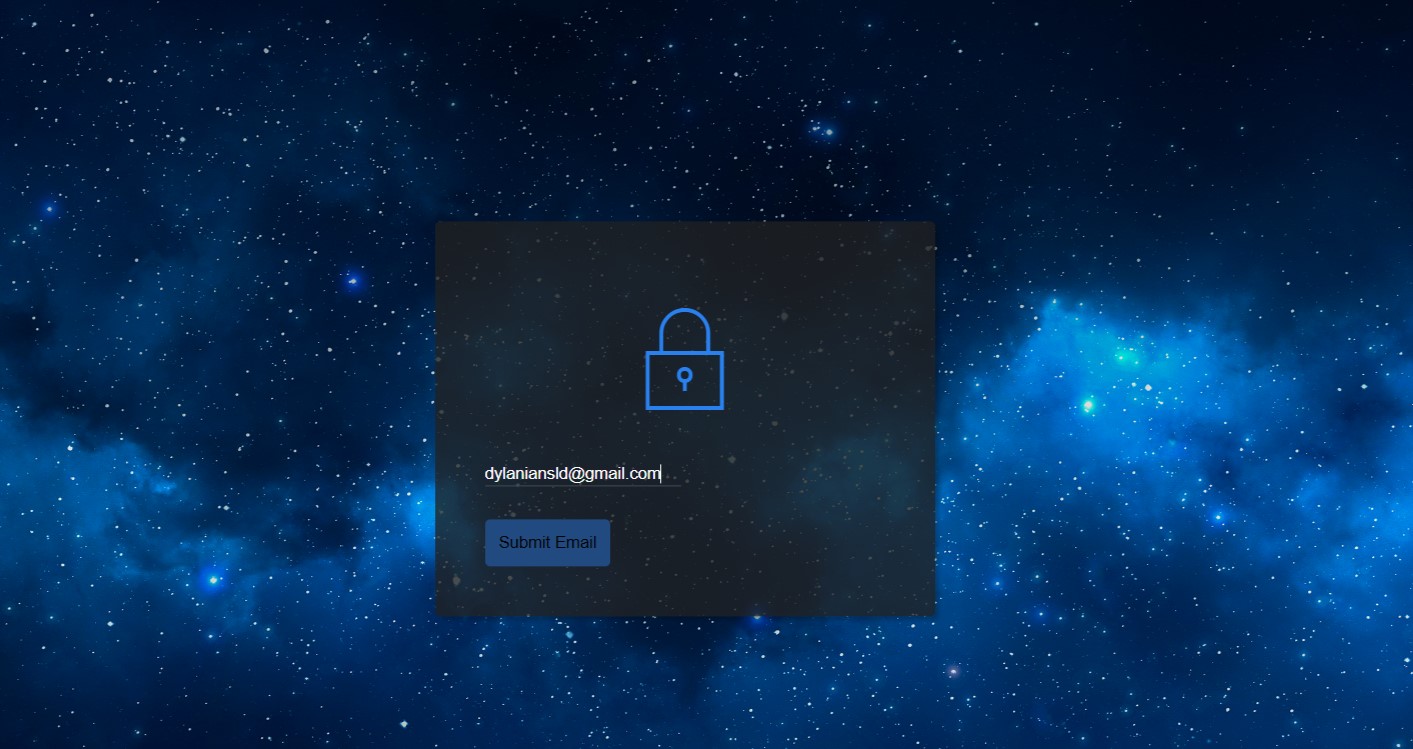
**Fig 6.2: Sign up Page**

The registration page is basically a formal introduction of the user. Contains details such as the Email ID. Users can then use this information to identify themselves and start programming using their account.



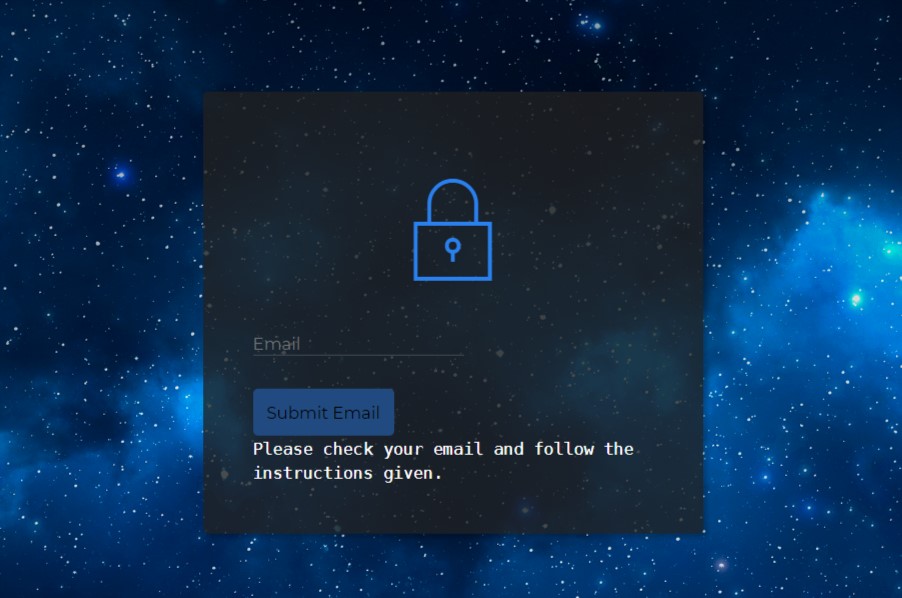
**Fig 6.3: User Login**

This module makes sure that the user is who he claims to be as there is a possibility of data being compromised by phishing attacks and snooping.



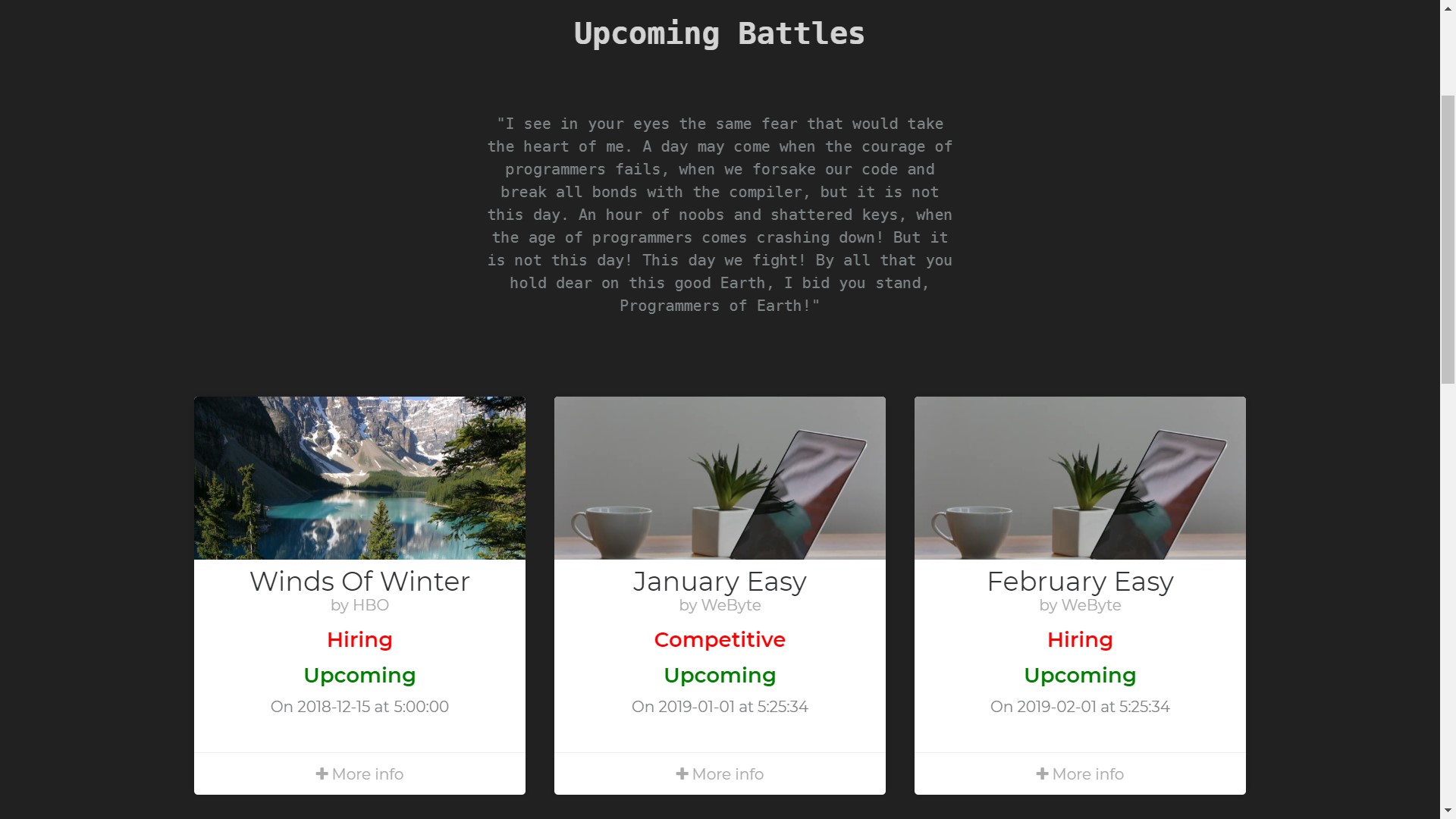
**Fig 6.4: Forgot Password Page**

Should the user ever happen to forget his/her password, he/she can access our Forget Password Page to get access to his/her account.



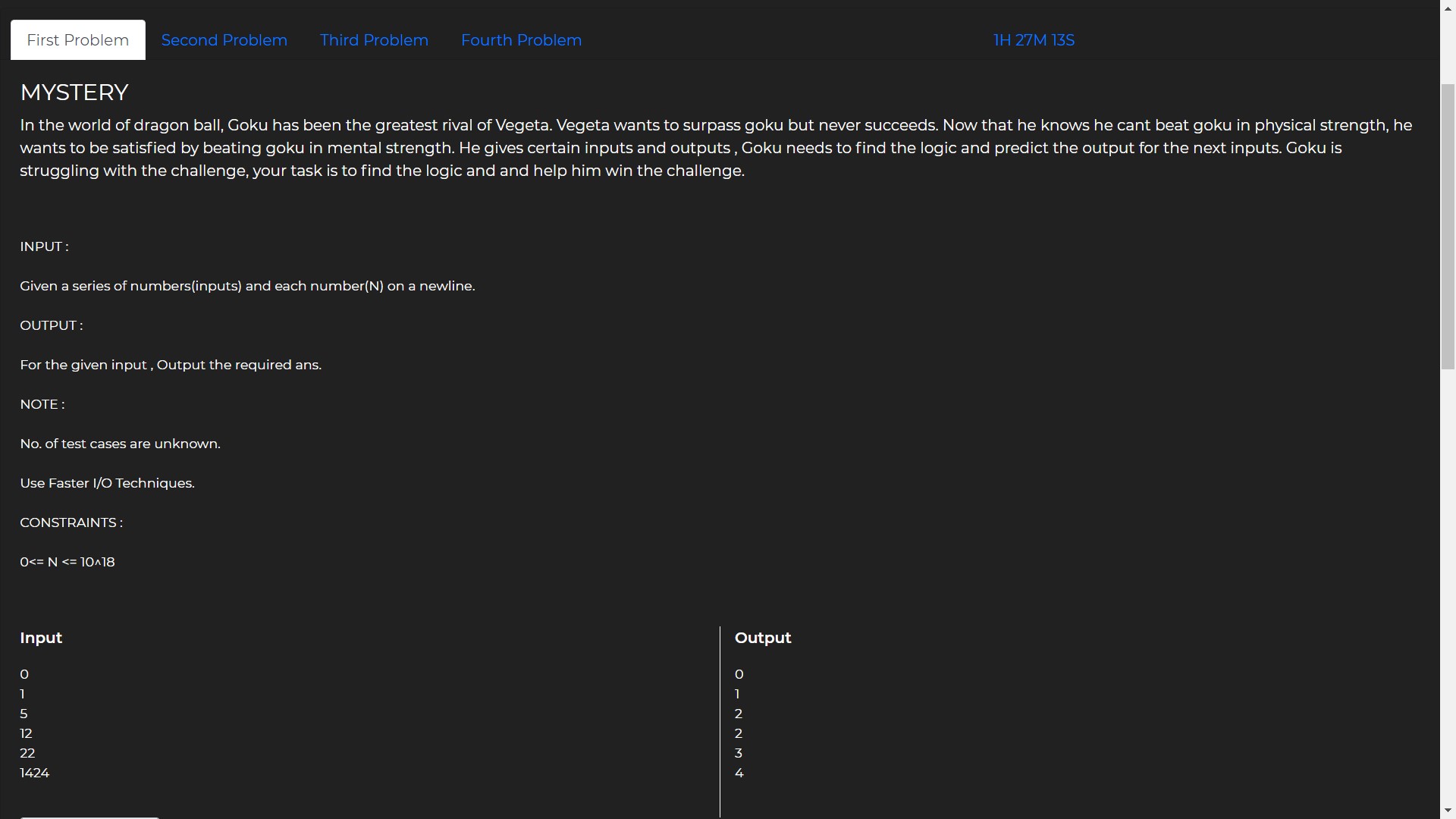
**Fig 6.5: Forgot Password Confirmation**

This page informs the user to check their registered email and get the new password.



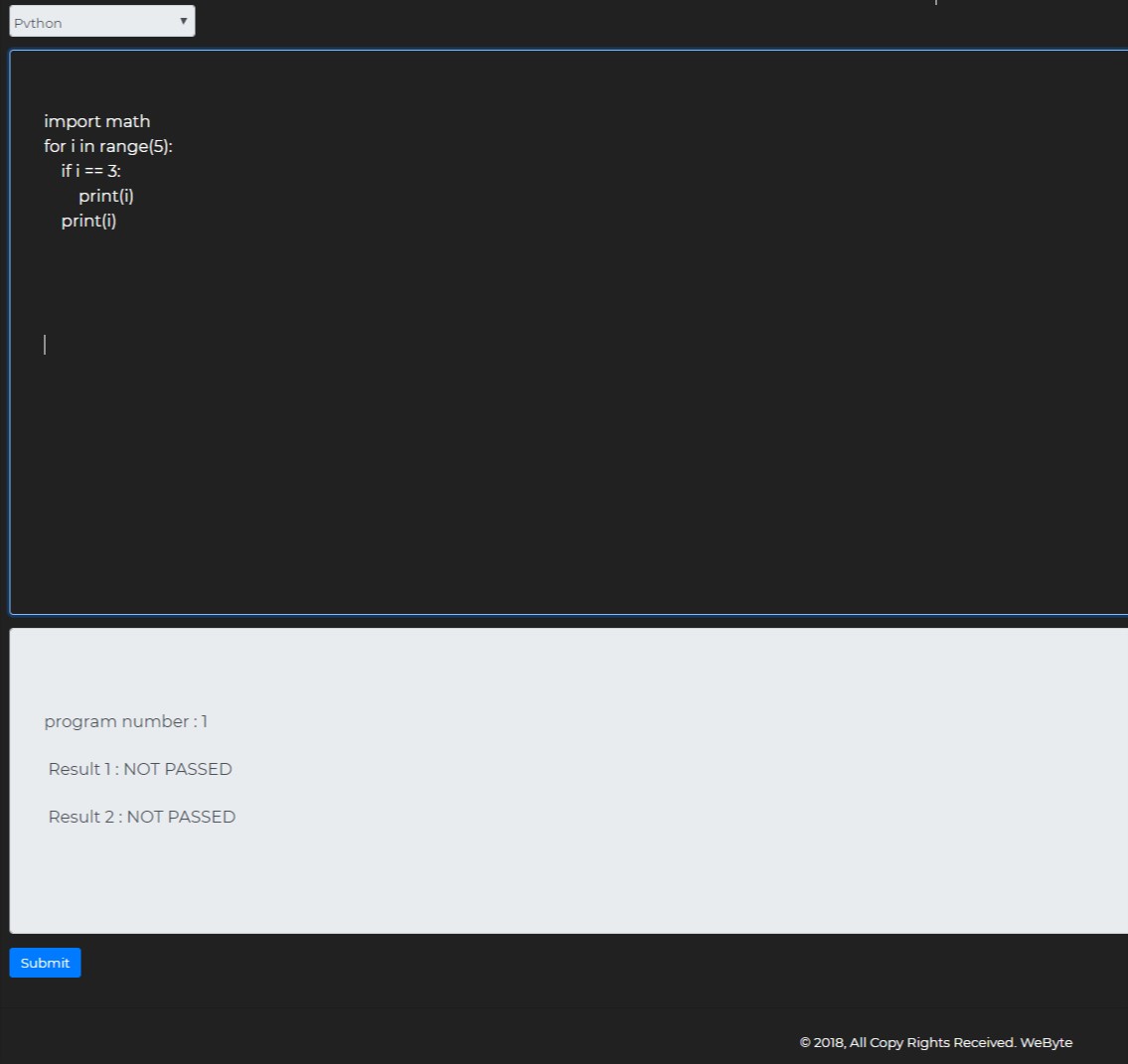
**Fig 6.6: Upcoming Battles Page**

This page provides the details with respect to the competitions that are available to the user to participate in.



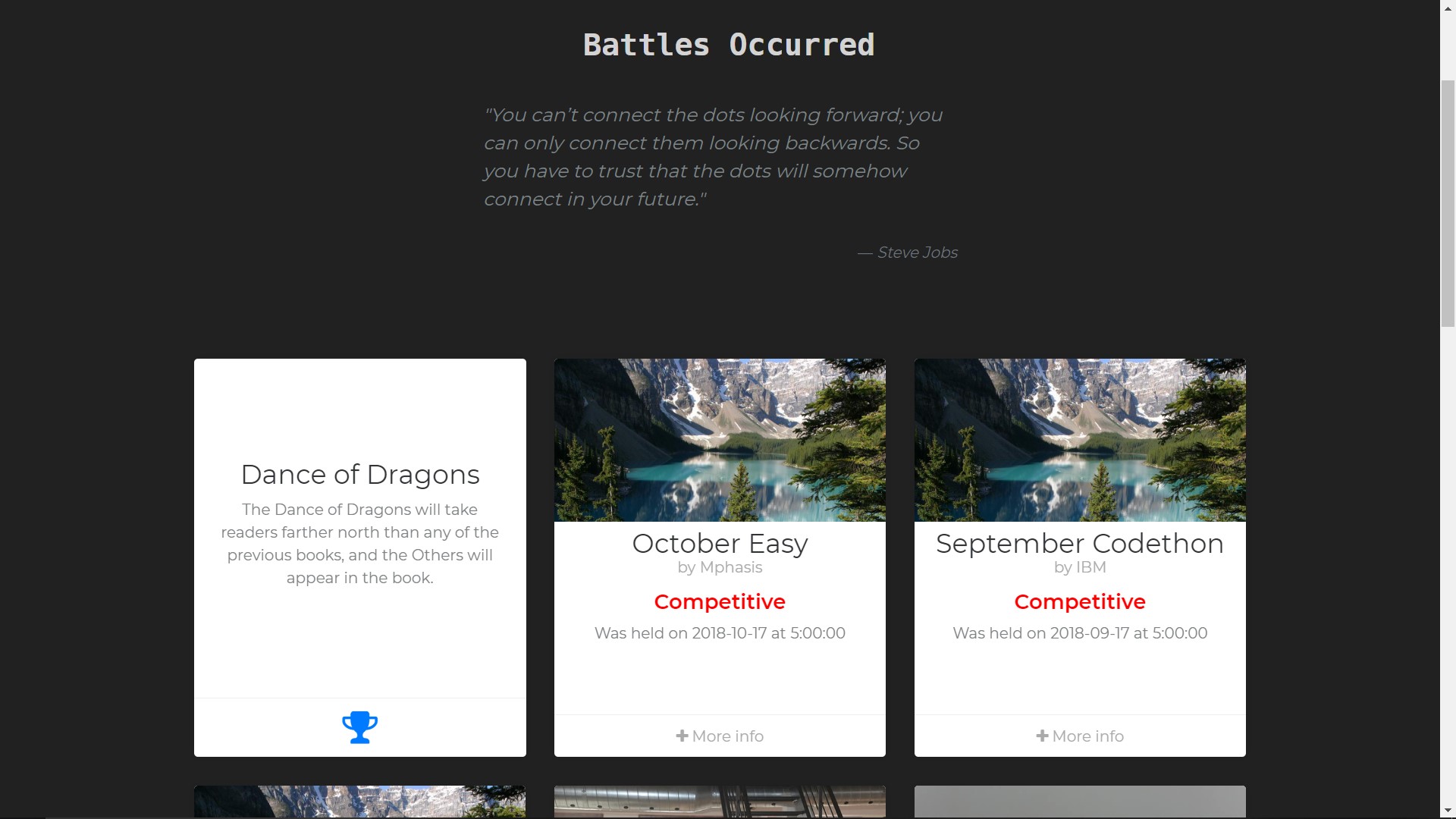
**Fig 6.7: Battleground Page**

This is where the programmer is allowed to analyze the problem and come up with an efficient solution that passes all the necessary test cases.



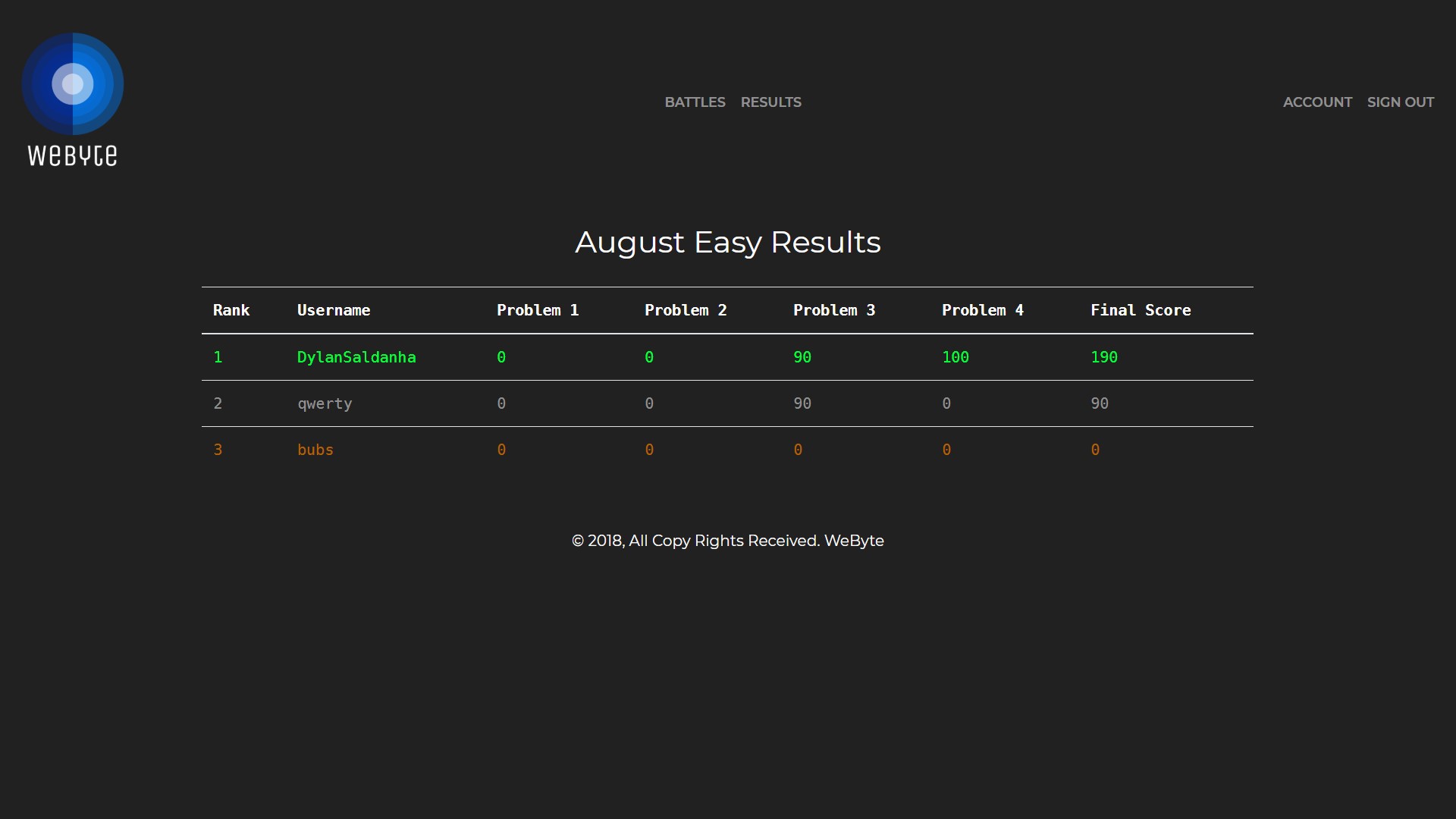
**Fig 6.8: Battleground Editor**

This is the part of the previous page that allows the user to enter one’s solution to the above mentioned problem.



**Fig 6.9: Battle Occurred Page**

This page allows the user to see the list of competitions for which the results are out.



**Fig 6.10: Results Page**

This is where the user gets an opportunity to know where he/she stands in the competition. The user can get an insight on the areas he/she needs to improve on.

# **CHAPTER 7**

## **CONCLUSION AND FUTURE WORKS**

### **CONCLUSION**

The main motive of this Mini Project was to bring out that inner hacker and problem solver out of the users. Also we have made sure that organizations can easily host this website on their server without any further costs. This is done by making the code open source.

### **FUTURE WORKS**

By application of programming tools such as HTML, CSS, JavaScript and Python, we have designed and implemented a coding platform. The system is a simple and user friendly. In future, we plan to add support for programming in multiple languages. As of now Python is the only language supported. We also plan on implementing features like sending the results to the competitors via email. So that they don’t need to always check when the results are out. This should offload some traffic from the servers. Finally, we have also decided to add a feature that allows the programmer to see the output to one’s code and not just if it has passed or failed the test cases.

## **BIBLIOGRAPHY**

1. HTML and CSS: Design and Build Websites, by Jon Duckett
2. JavaScript and JQuery: Interactive Front-End Web Development, by Jon Duckett
3. Think Python: The most basic of this list, Think Python provides a comprehensive Python reference.
4. Flask Web Development, 2nd Edition, by Miguel Grinberg and Felix Kjellberg.
5. http://flask.pocoo.org
6. https://www.w3schools.com/
7. https://docs.python.org/3/