5 Testing

Testing is an **extremely** important component of most projects, whether it involves a circuit, a process, power system, or software.

The testing plan should connect the requirements and the design to the adopted test strategy and instruments. In this overarching introduction, given an overview of the testing strategy and your team's overall testing philosophy. Emphasize any unique challenges to testing for your system/design.

In the sections below, describe specific methods for testing. You may include additional types of testing, if applicable to your design. If a particular type of testing is not applicable to your project, you must justify why you are not including it.

When writing your testing planning, consider a few guidelines:

- Is our testing plan unique to our project? (It should be)
- Are you testing related to all requirements? For requirements, you're not testing (e.g., cost related requirements) can you justify their exclusion?
- Is your testing plan comprehensive?
- When should you be testing? (In most cases, it's early and often, not at the end of the project)

5.1 Unit Testing

What units are being tested? How? Tools?

For unit testing, we have a couple of units that we think fit best here:

The first unit we would test would be the backend code:

Testing the functionality of the CySim scoreboard, to do this, we plan to implement a variety of different test cases throughout the code in order to check if each specific piece is working the way that it should be. Doing this will allow us to confirm that it is working properly and is outputting the correct values as need be. An example of this is that we should have a live tracker constantly checking the percentage finished for each team, if we go in and implement test cases to print out the values of the percentage we will be able to compare it with our actual thoughts that we had for this and determine if it's all functioning correctly.

The other unit we would be testing is the database behind the web application:

Testing the database behind the web application is crucial because this is very important to helping our application work properly. For example, we will use "mock emails" to create new users and input them into the database, from there we can have them go through and use the web application and we should hopefully be able to track their status and progress in the database as well as information that they may provide. The mock emails would be a nice tool that we can use in order to accurately test the database unit of the web application.

5.2 Interface Testing

What are the interfaces in your design? Discuss how the composition of two or more units (interfaces) are being tested. Tools?

Our design will incorporate at least two interfaces, the scoreboard and the web app interface. These interfaces will be tested for understandability and accuracy, with the scoreboard being tested to ensure up-to-date information is being shown to participants in a timely manner. Testing the web app will involve lots of test cases and input handling to ensure that users can interact with the simulation's backend smoothly, likely incorporating unit testing tools like Mockito.

5.3 Integration Testing

What are the critical integration paths in your design? Justification for criticality may come from your requirements. How will they be tested? Tools?

For our groups critical integration paths we have multiple, these are, connecting the different portions of the score calculation in the backend to produce a status percentage, combining all the different pieces of the scoreboard together to display everything needed to be displayed, combining the backend logic to have the game function as it should, develop the database features for the web application so that it can be added to the web app and used properly and also implement an add and create user section of the web application that allows us to collect information about the users accessing and using the web application.

Below is how they will be tested:

Connecting the different portions of the score calculation in the backend to produce a status percentage: For this, we will implement markers and test cases within the code in order to track outputs and make sure that everything is adding up overtime.

Combining all of the different pieces of the scoreboard together to display everything needed to be displayed: Doing this we will output all of the pieces together in order to have a mock display that shows how it all is displayed on the screen, from there we will be able to see if it looks the way we intended it to.

Combining the backend logic to have the game function as it should: Similar to the score calculation, we will be using test cases in order to track outputs for this portion and make sure that it all looks correct.

Develop the database features for the web application so that it can be added to the web app and used properly: We will use "mock emails" to create new users and input them into the database, from there we can have them go through and use the web application, and we should hopefully be

able to track their status and progress in the database as well as information that they may provide.

Also implement an add and create user section of the web application that allows us to collect information about the users accessing and using the web application: We will be able to use the database to track this and make sure that information is being collected and distributed properly so that there are no issues and the add and create users works properly.

5.4 System Testing

Describe system level testing strategy. What set of unit tests, interface tests, and integration tests suffice for system level testing? This should be closely tied to the requirements. Tools?

Some critical features include:

- **Web Application Frontend Login, User, and Features (submit flags and view information)**
- **Web Application CySim Backend Integration (scoreboard and scenario control)**
- **Nagios Web Application Integration**
- **Networking and Access to Specified Machines**

As said before, we will be creating mock users in our database of multiple teams/colors. These can be utilized by C# / dotnet unit tests to ensure users have appropriate team view of the website. We can then further utilize these users as other features of the website are operating correctly. They can be utilized to submit flags, view team specific information, and test controls.

To test our system's backend integrations, such as the scoreboard system, we can utilize unit tests to ensure the scoring of each measured part is calculated correctly. Then we can create a series of mock values for all scoring fields and compare what they are calculated to and weighted as. Additionally, we can test white scoring influence by allowing white team users to modify how scoring is weighted.

To test the Nagios integration, we will be creating a test network/scenario consisting of mock blue team systems with Nagios system monitoring. The server will connect to the Nagios instance for service uptime information. With this information, we can test the system networking capabilities and Nagios integration functionality by confirming information received is what Nagios has stored. We can control the status of blue team services and test the blue team uptime and scoring systems by seeing if they match the appropriate outputs for a series of cases.

5.5 Regression Testing

How are you ensuring that any new additions do not break the old functionality? What implemented critical features do you need to ensure do not break? Is it driven by requirements? Tools?

In order to ensure that new features don't disturb previously added ones, we will employ the use of git version control, allowing us to develop in independent branches for testing and implementation of new features. We will require that any new features to be implemented and fully tested in a separate branch before, and for another team to merge the branch if ready. Along

with this, we can utilize git to integrate with various testing frameworks to automatically test code upon updates to a branch.

In the case of virtual machines systems we are building, we can utilize snapshots between each major update, ensuring that at all times there is at least one working backup of the machine being configured for CySim. As well, during the time of updates/development to machine, we can have multiple instances of a VM running to ensure that other members are not interrupted if a server update does break some functionality.

5.6 Acceptance Testing

How will you demonstrate that the design requirements, both functional and non-functional are being met? How would you involve your client in the acceptance testing?

To demonstrate that the laid out requirements have been met we have been tasked with not just setting up the necessary systems but also building a test scenario which will properly test each element of the project. This scenario is to encompass all elements of the project and be usable by our client. While the test may not be exactly what a final scenario would look like it will be geared towards stressing systems and features that have been implemented. During this phase we will also be in constant contact with our client to ensure that the project meets their expectations and requirements.

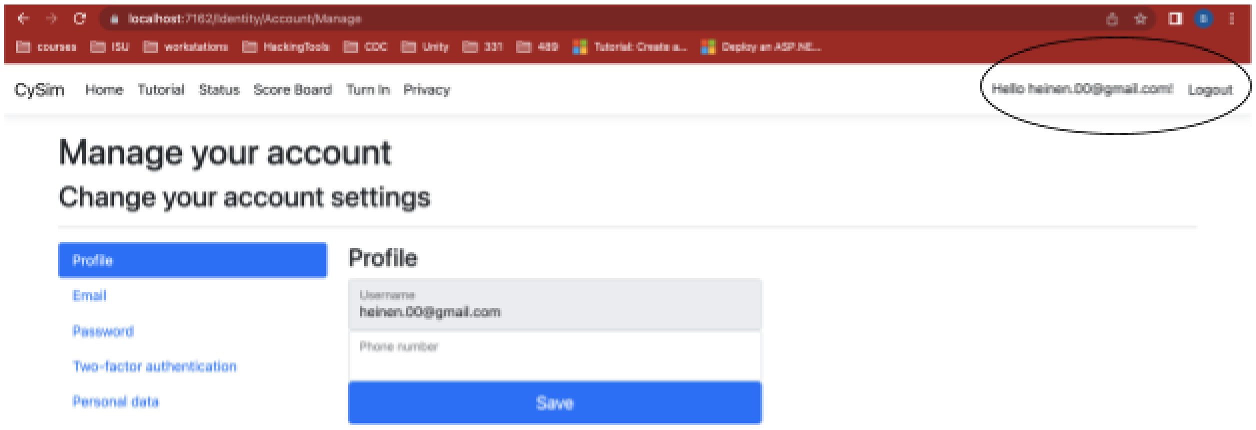
5.7 Security Testing (if applicable)

Not applicable to this project.

5.8 Results

What are the results of your testing? How do they ensure compliance with the requirements? Include figures and tables to explain your testing process better. A summary narrative concluding that your design is as intended is useful.

For testing, we decided the best place for us to start would be to get the skeleton functionality of the website going. The first thing that we were able to do was create a login method mechanism for our website, we were able to use the .Net login. This gave us the ability to create a user, login as the user, and then we were able to view account settings. In account settings we were able to change password, change email, setup two factor authentication, and download personal data.



Manage your account

Change your account settings

Profile

Email

Password

Two-factor authentication

Personal data

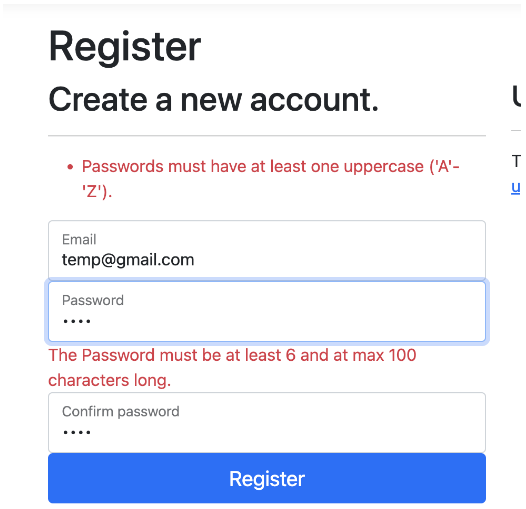
Profile

Username
heinen.00@gmail.com

Phone number

Save

One neat feature that we tested completely is when you are creating a new user, the users password needs to be 6-100 characters long and you need to have one capital letter



Register

Create a new account.

- Passwords must have at least one uppercase ('A'-'Z').

Email
temp@gmail.com

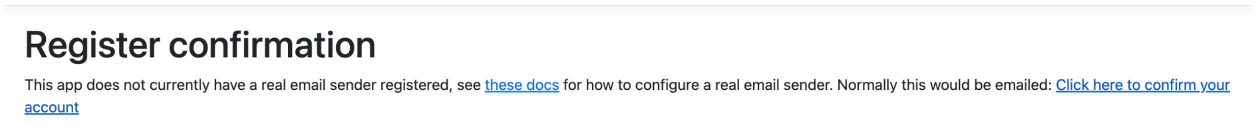
Password
••••

The Password must be at least 6 and at max 100 characters long.

Confirm password
••••

Register

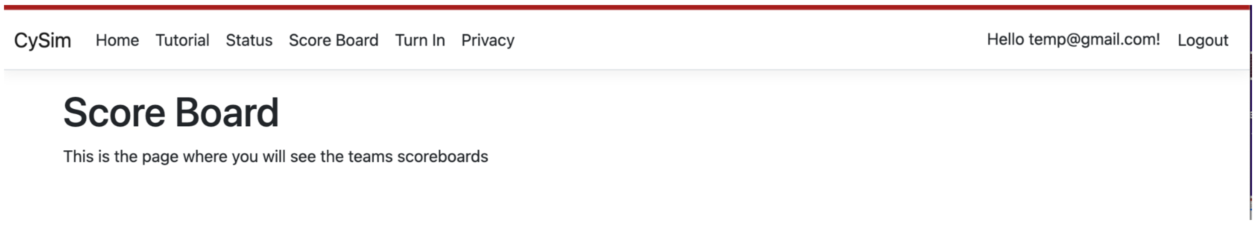
You will also be required to confirm your email account, right now you just click I accept but in the future you will need to go to your own email and confirm.



Register confirmation

This app does not currently have a real email sender registered, see these docs for how to configure a real email sender. Normally this would be emailed: Click here to confirm your account

Another thing that we created and tested was the tab bar and a user is able to go to the following tabs home, tutorial, status, score board, turn in, and privacy. And currently each of these tabs will just load dummy pages.



Score Board

This is the page where you will see the teams scoreboards

I also created a working scoreboard but can not get a picture at the moment. The scoreboard logic will work off of query logic where you search your data and get the data and descending order with the highest score being the first element. And you can select the number of entries that you wish to show like 10 or 15 for example.

This is useful for our design because this is essentially the skeleton of our project. Our entire project will be driven by the web app. This is how participants will be able to access their machines, see their status in the game, and much more. We now will just have to focus on more technical aspects and can focus less on the route.