# SCORN Developer’s Guide

**Laboratory for Advanced Construction Technology**

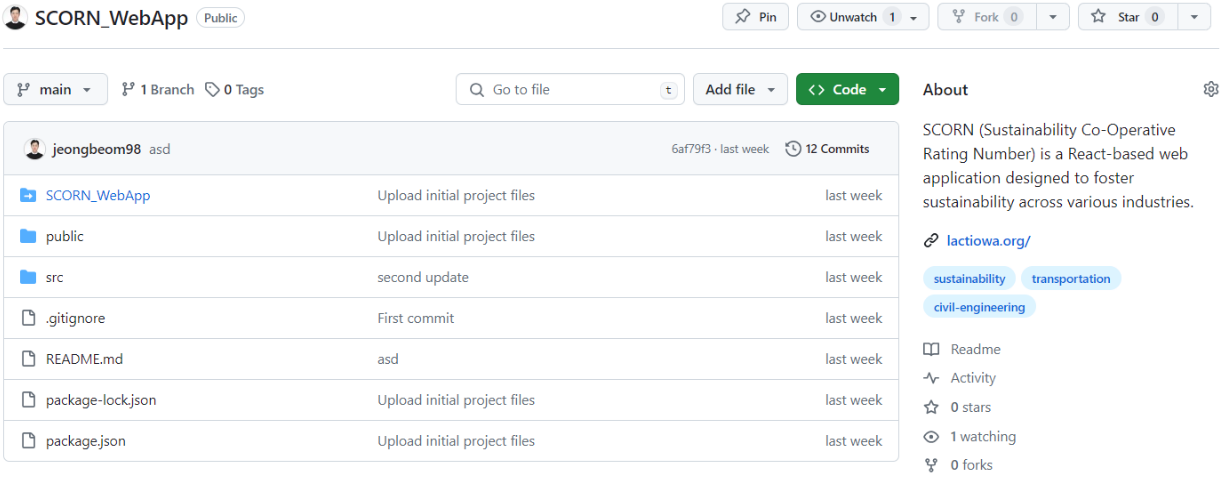
*Prof. David Lee*

*Jeongbeom Lee*

## Table of Contents

## 1. Environment Setup

To start development with the SCORN web application, setting up the right environment is essential. The foundation of this setup includes having the latest version of Node.js installed on your system, which is pivotal for running and developing React applications like SCORN. Node.js comes bundled with Node Package Manager, an indispensable tool for managing packages and dependencies within the project. Equally crucial is Git, a version control system that facilitates code management, allowing developers to clone the project repository, track changes, and collaborate efficiently.



jeongbeom98 asd — 6af79f3 · last week — 12 Commits

| | | |
|---|---|---|
| SCORN_WebApp | Upload initial project files | last week |
| public | Upload initial project files | last week |
| src | second update | last week |
| .gitignore | First commit | last week |
| README.md | asd | last week |
| package-lock.json | Upload initial project files | last week |
| package.json | Upload initial project files | last week |

SCORN (Sustainability Co-Operative Rating Number) is a React-based web application designed to foster sustainability across various industries.

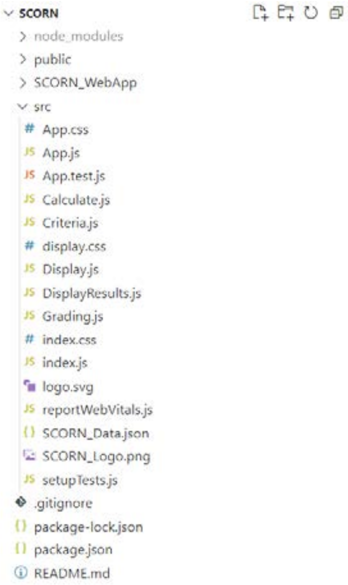
lactiowa.org/

sustainability transportation civil-engineering

Once the software prerequisites are in place, the next step involves downloading the SCORN project repository. This can be achieved through Git by executing a clone command that copies the entire project structure to your local machine. This process ensures that you have a personal copy of the project's codebase, making it possible to embark on development, perform tests, and contribute to the project.

## 2. Project Structure and Key Files Explanation

The SCORN web application is structured around several key JavaScript (JS) files that collectively define its functionality, interface, and the dynamic interactions users experience. Understanding the role of each file is crucial for developers looking to contribute to the project, as it enables efficient navigation and modification of the codebase. Here’s an overview of the core files:



```
SCORN
  node_modules
  public
  SCORN_WebApp
  src
    App.css
    App.js
    App.test.js
    Calculate.js
    Criteria.js
    display.css
    Display.js
    DisplayResults.js
    Grading.js
    index.css
    index.js
    logo.svg
    reportWebVitals.js
    SCORN_Data.json
    SCORN_Logo.png
    setupTests.js
  .gitignore
  package-lock.json
  package.json
  README.md
```

### 2.1 App.js

This is the heart of the SCORN application, serving as the main entry point. `App.js` orchestrates the rendering of the entire application, managing state transitions, routing, and the display of major components such as `Criteria`, `Display`, and `DisplayResults`. It integrates the different parts of the application into a cohesive user experience.

### 2.2 Criteria.js

This file defines the `Criteria` component, which is responsible for presenting the sustainability assessment criteria to the user. It lays out the evaluation standards and domains in an informative manner, guiding users on how to assess their sustainability practices effectively. The `Criteria` component is crucial for the educational aspect of SCORN, ensuring users understand the basis of their sustainability evaluation.

### 2.3 Display.js

The `Display` component, defined in this file, handles the presentation and interaction of the questionnaire. It dynamically generates questions based on the criteria selected by the user, captures responses, and manages the navigation through different parts of the questionnaire. This file is key to the interactive nature of SCORN, making the assessment process engaging and user-friendly.

### 2.4 DisplayResults.js

After users complete the questionnaire, the `DisplayResults` component takes over to show the calculated sustainability grade. Defined in this file, it processes the grading results, displays them in a clear and concise manner, and offers functionality to export the results for further analysis or record-keeping. This component brings closure to the assessment process by providing users with tangible feedback on their sustainability efforts.

### 2.5 CalculateGrade.js

The logic for calculating the sustainability grade based on user responses is encapsulated in this file. `CalculateGrade.js` is a critical component of SCORN, as it directly impacts the outcome of the assessment. It takes into account the points associated with each response, applies the grading logic, and determines the final grade. This file is essential for ensuring the accuracy and reliability of the assessment results.

Together, these files form the backbone of the SCORN application, each playing a distinct role in delivering a comprehensive sustainability assessment tool. For developers contributing to SCORN, familiarity with these files is the first step towards effective collaboration and enhancement of the project.

## 3. Getting Started

The commencement of development activities for the SCORN web application is predicated upon a methodical installation process, followed by procedural steps to operationalize the project on a local computational environment. This segment elucidates a structured guide aimed at facilitating developers in establishing their development milieu efficiently and initiating contributions to the project.

The foundational step in preparing for development on the SCORN project involves the installation of Node.js, inclusive of npm (Node Package Manager). Given that SCORN is constructed utilizing the React framework, the presence of Node.js is imperative for the activation of the development server, whilst npm is utilized for the management of packages and dependencies within the project framework.

Subsequent to the installation of Node.js, it is requisite to clone the SCORN repository to a local directory. This can be achieved by executing the following command in a terminal or command prompt, thereby navigating to the desired directory where the project will reside:

```
>>> git clone https://github.com/jeongbeom98/SCORN_WebApp.git
```

Post-cloning, one must transition into the SCORN project directory via:

```
>>> cd SCORN_WebApp
```

Within the project directory, the installation of necessary npm packages is accomplished by executing:

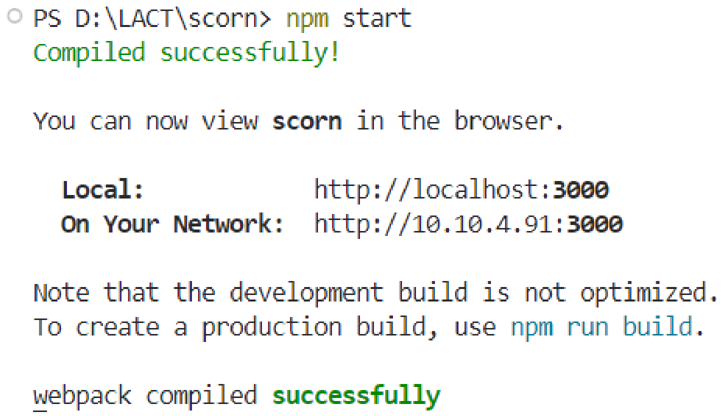
```
>>> npm install
```

This command interprets the `package.json` file and procures all requisite dependencies for the project.

Upon completion of the installation, the project is ready to be executed locally. Within the project's directory, the development server is initiated by executing:

```
>>> npm start
```

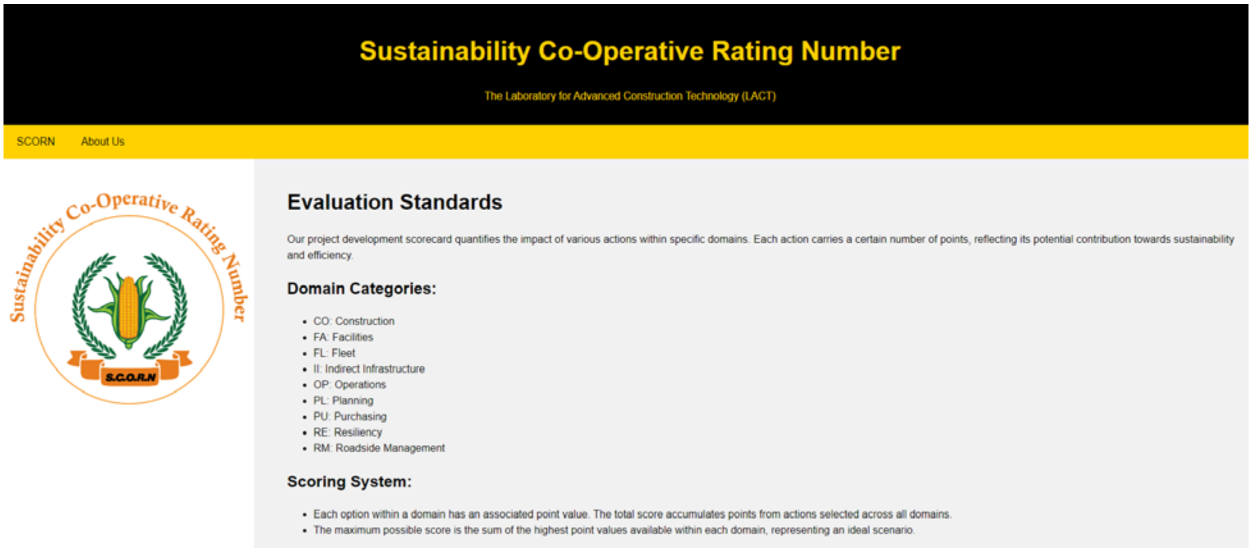
This action activates the React development server and automatically renders the SCORN application within the default web browser. Typically, the application is accessible at `http://localhost:3000`, offering an interactive platform for engaging with the SCORN questionnaire, implementing new functionalities, or ameliorating extant code.



```
○ PS D:\LACT\scorn> npm start
  Compiled successfully!

  You can now view scorn in the browser.

    Local:            http://localhost:3000
    On Your Network:  http://10.10.4.91:3000

  Note that the development build is not optimized.
  To create a production build, use npm run build.

  webpack compiled successfully
```

The local execution of SCORN permits developers to witness real-time reflections of their modifications, facilitating an immediate feedback loop essential for development and diagnostic processes. As modifications are made to the codebase, the development server seamlessly reloads the application within the browser, instantaneously showcasing the implemented alterations.

Through adherence to these installation and execution directives, developers are equipped to commence with contributions towards the SCORN project, thus augmenting the tool's functionality and further endorsing its objective of propagating sustainability practices across varied domains.



Sustainability Co-Operative Rating Number

The Laboratory for Advanced Construction Technology (LACT)

Evaluation Standards

Our project development scorecard quantifies the impact of various actions within specific domains. Each action carries a certain number of points, reflecting its potential contribution towards sustainability and efficiency.

Domain Categories:

- CO: Construction
- FA: Facilities
- FL: Fleet
- II: Indirect Infrastructure
- OP: Operations
- PL: Planning
- PU: Purchasing
- RE: Resiliency
- RM: Roadside Management

Scoring System:

- Each option within a domain has an associated point value. The total score accumulates points from actions selected across all domains.
- The maximum possible score is the sum of the highest point values available within each domain, representing an ideal scenario.

## 4. Contributing to SCORN

Engagement with the SCORN project, by means of contributing to its development, is encouraged under a structured framework designed to facilitate meaningful and coherent additions. This section delineates the protocol for engaging with the project, encompassing the initial steps of forking the repository, the intricacies of integrating modifications—specifically the addition of new questions—and the procedural approach to submitting these changes for review through pull requests.

The initial phase in the contribution process entails creating a personal copy of the SCORN repository. This is accomplished by navigating to the SCORN GitHub page and utilizing the 'Fork' feature, thereby generating a replica of the repository under your GitHub account. This forked repository serves as a private platform where modifications can be freely implemented without affecting the original codebase.

To integrate new questions into the SCORN web app, developers need to update the `SCORN_Data.json` file, which serves as the database for all sustainability-related questions presented within the application. Here's a detailed guide and template you can include in the SCORN Developer's Guide:

### 4.1. Prepare the Data:

- Ensure your new questionnaire data is organized with the necessary attributes: `Category`, `Abbreviation`, `Number`, `Title`, `Option`, and `Points`.

- For new questions, assign a unique `Number` within each `Category`. The `Option` field should detail the possible answers, and `Points` assign a score to each option.

### 4.2. Format the Data:

- Convert your data into a JSON array format. Each question and its options should be a separate object within the array, similar to the example provided.

### 4.3. Update the `SCORN_Data.json` File:

- Replace the existing content of `SCORN_Data.json` with your newly formatted JSON data. Ensure the format is correct to avoid errors in the application.

### 4.4. Test Your Changes:

- After updating the JSON file, run the SCORN web app locally to ensure the new questions are displayed correctly and that the grading logic functions as expected.

- Template for `SCORN_Data.json`:

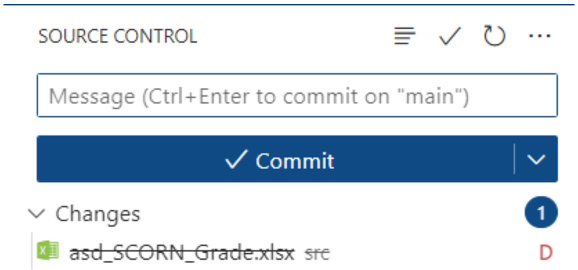
```
[{ "Category": "Your Category",
   "Abbreviation": "Abbreviation",
   "Number": 1,
   "Title": "Question Title",
   "Option": "Answer Option",
   "Points": 1.0},
  {"Category": "Your Category",
   "Abbreviation": "Abbreviation",
   "Number": 1,
   "Title": "Question Title",
   "Option": "Another Answer Option",
   "Points": 2.0},
  // Add more questions and options here]
```

- Example Addition: If you're adding a new question about "Reducing Plastic Use" within the "Facilities" category, your entry might look like this:

```
 [{"Category": "Facilities",
   "Abbreviation": "FA",
   "Number": 3, // Assuming 1 and 2 are already taken
   "Title": "Reduce Plastic Use",
   "Option": "Reduce single-use plastic by 10%",
   "Points": 1.0},
  {"Category": "Facilities",
   "Abbreviation": "FA",
   "Number": 3,
   "Title": "Reduce Plastic Use",
   "Option": "Not Applicable",
   "Points": -1.0}
  // Continue adding other options as needed]
```

Upon the completion of modifications, contributors are required to initiate a pull request—a formal proposal to merge their changes into the original SCORN repository. This is achieved by navigating to the 'Pull Requests' section of the original SCORN GitHub repository and selecting 'New Pull Request'. Contributors must then choose their forked repository as the 'compare' branch and outline the nature and rationale of their contributions in the provided description field.

The submission of a pull request triggers a review process, during which the proposed changes are evaluated by the project's maintainers. This review ensures that contributions align with the project's objectives, standards, and quality requirements. Upon approval, the changes are merged into the SCORN repository, marking the successful contribution of new content or features to the project.


SOURCE CONTROL
Message (Ctrl+Enter to commit on "main")
✓ Commit
Changes
1
~~asd_SCORN_Grade.xlsx~~ ~~src~~
D

## 6. Deploying SCORN to AWS Amplify

To make SCORN accessible as a fully functional web application, deployment through AWS Amplify is recommended. AWS Amplify streamlines hosting both the backend and frontend of web applications, ensuring scalability and reliability. Follow these steps to deploy SCORN to AWS:

### 6.1 Setting up AWS Amplify

Before you begin, ensure you have the AWS Amplify CLI installed and configured as outlined in Part II of the initial setup guide. The CLI will interface with your AWS account to provision cloud resources.

### 6.2 Initializing Your Project in AWS Amplify

In your project's root directory, initiate the Amplify project by running:

```
>>> amplify init
```

Follow the prompts to:

- Name your project and environment.
- Specify the default code editor.
- Choose 'javascript' for the type of app and 'react' for the framework.
- Accept default configurations for paths.
- Select the AWS profile you've set up during the Amplify CLI configuration.

### 6.3 Deploying the Backend

To deploy the backend resources to your Amplify environment, execute:

```
>>> amplify push
```

Confirm when prompted to proceed with the deployment.

### 6.4 Hosting Your Application

For hosting the frontend, run:

```
>>> amplify add hosting
```

Choose 'Amplify Console' and 'Manual deployment'. Once hosting is set up, publish your application by running:

```
>>> amplify publish
```

Confirm with 'yes' when prompted.

### 6.5 Verifying the Deployment

After the deployment, the Amplify Console will provide a URL to your hosted application. Visit this link to ensure that the SCORN web application is live and functional.

### 6 .6 Removing Your Application

Should you need to remove your application from AWS, avoid deleting directly from AWS services. Instead, use the Amplify CLI with the command:

```
>>> amplify delete
```

This ensures a clean removal of services provisioned by AWS Amplify for your project.