생산성 혁신을 위한 프로세스 기반 시뮬레이션 어낼리틱스 인터페이스 명세서

사업명	생산성 혁신을 위한 프로세스 기반 시뮬레이션 어낼리틱스
문서명	인터페이스 명세서
문서번호	io-chdsr-ifs-001

생산성 혁신을 위한 프로세스 기반 시뮬레이션 어낼리틱스 인터페이스 명세서

Document Information

Project Name:	생산성 혁신을 위한 프로세스 기반 시뮬레이션 어낼리틱스		
Project Manager:	Iochord	**Version Number:**	-
		Version Date:	-
Author:	Iochord	**Written Date:**	-
Reviewer:	Iochord	**Reviewed Date:**	-

Document Distribution

Copy Number	Name(Role, Title)	Date	Contact (Phone/email)

Revision History

Version	Version Date	Author	Description
0.0.1	2020/01/31	야베스	

Table of Contents

1. Development Setup

The user interface application was developed using Visual Studio Code and are independent from the simulation engine and web server application. We suggest developers who would like to extend the user interface capabilities to use the editor. Visual Studio Code provides a set of features that enable user to develop applications. Intellisense, the main feature of Visual Studio Code provides "an auto-completions based on variable types, function definitions, and imported modules"[1]. It works with various programming language including TypeScript, the programming language used to develop the UI application. A debugger enables user to debug code right from the editor by setting breakpoints within the code, launch or attach running apps to the debugger, step into the code using the debugger panel, and inspect variables within interactive console. Developers use Source Control Management (SCM) interface built as a part of the editor to manage software revisions towards opened or active project. Developers may extend the capabilities of the editor by installing third-party extensions developed by the open-source community.

Microsoft has developed a comprehensive guide to setup the editor supporting several operating systems[1]. We assume that the Visual Studio Code editor is installed and available within the operating system environment. To develop the user interface application, open the Visual Studio Code. Import the project by locating the root project of the UI application.

```
$ tips-simulation2019_development/ips/ips-simulator-web/src/main/ts
```

Listing 1: The default directory for user interface application for TIPS project.

Running the application for the first time results into an error. Proper installations of project dependencies will resolve such problem. Project dependencies includes third party modules are installed through Node Package Manager (NPM)[1]. To install project dependencies, navigate through the integrated terminal of Visual Studio Code and issue the following command.

```
$ ~/ts> npm install
```

Listing 2: Command to install project dependencies.

To run the user interface application, a development server is needed to be started and hosted on the APP_PORT of the running server domain (configurable by the user). Any amendement of the source code will trigger the server to restart and update the user interface displayed in the browser. To spawn the development server, issues the following command in the terminal.

```
$ ~/ts> npm run serve
```

Listing 3: Command to run the user interface application.

When terminal command from Listing 3 evaluated to success, navigate to a browser and paste the following URL in the search box: http://localhost:<APP_PORT>. This URL will redirect user to the login page of the application as seen in the Figure 1.

[1] npm, Inc. "Node Package Manager." *Npm*, www.npmjs.com/

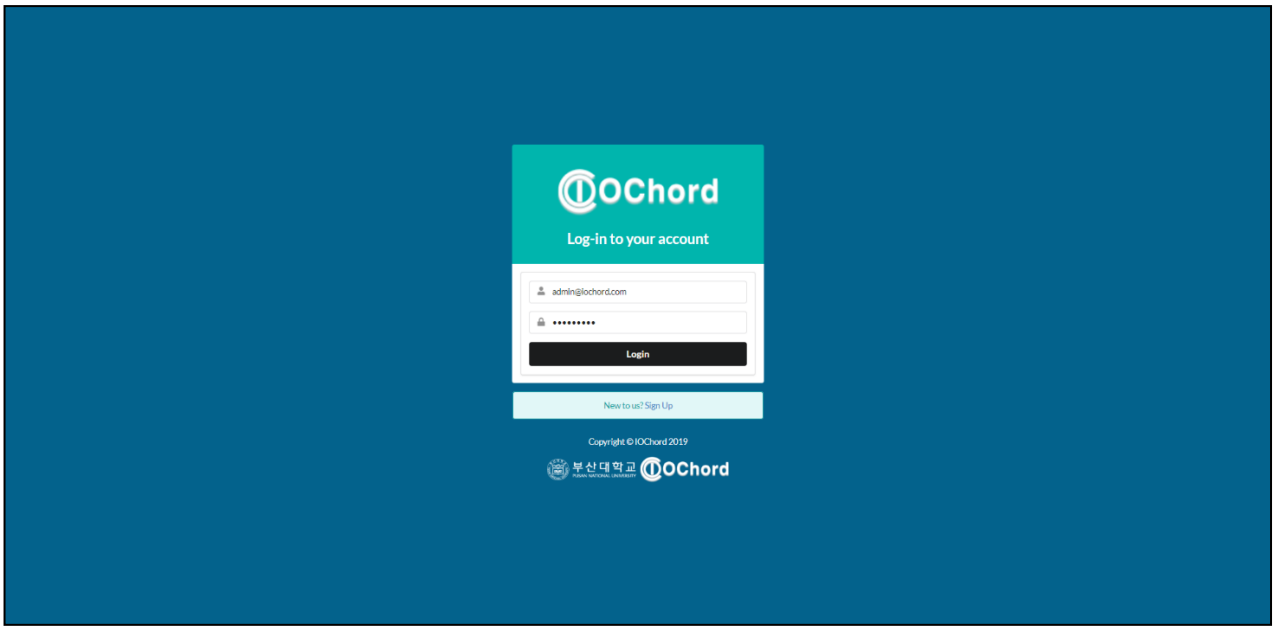

Figure 1 Login Page for the user interface application

2. List of Development Setup Requirement

Since the user interface application is developed using Visual Studio Code editor, a set of extensions is used to make the process of capabilities development and extension for the user interface application becomes convenient. Table 1 lists several important extensions to support such process.

Table 1 List of editor extensions needed to develop and extend the user interface application

Extensions	Description
Debugger for Chrome	This extension enables developers to debug JavaScript code in the Chrome browser, or any other targets that support the Dev-Tools Protocol[1].
Debugger for Firefox	*Debugger for Firefox* extension was made for debugging JavaScript application ran on Firefox browser[1].
Trailing Spaces	Spaces are part of the source code and it consumes a significant amount of local storage. This extension clears up any excessive white spaces e.g. whitespace, tabs, etc. for an opened files.
TypeScript Extension Pack	This extension provides additional tools for TypeScript development including linter, import organizer or sorter, path auto-completion, TypeScript definitions importer, and a code formatter.
Vetur	Vue is a JavaScript framework for building user interfaces and single-page applications based on the Model-View-ViewModel

Extensions	Description
	paradigm. This extension extends the capabilities of Visual Studio Code for developing Vue application. Supported features including syntax highlighting, snippets, linting, formatting, auto-completion, debugging, and Command Line Interface (CLI)

[1] Debugging *client-side* JavaScript code, owing to *Debugger for Chrome/Firefox* extensions, allows developer to examine the state of a program that is currently executing. It is useful when rudimentary techniques are no longer adequate to promote the adherence of software to its testing requirement. While the user interface application is developed using TypeScript, a compiler is utilized to transpile the code into JavaScript, hence clarify the possibility of debugging TypeScript code using the extension.