**CREATE: Documentation**

**1. Introduction**

**General Description**

The project supports the urban transition to a circular economy by inventorying construction material stocks, developing scenarios for material flows, and providing governance arrangements. It focuses on major urban infrastructures, such as buildings and municipal roads. A transdisciplinary consortium combines quantitative modeling with qualitative study. The project enhances existing tools and integrates new digitalization technologies, involving various stakeholders through urban labs and partner cities in five countries. The analysis of best practices and new governance interventions will lead to concrete proposals for the participating cities and a scaling strategy for Europe.

**Application Purpose**

The project aims to quantify and visualize the stocks and flows of materials in the built urban environment, such as concrete and steel, both in accumulated stocks and in the flows required for new constructions and renovations. The KPI-based evaluation will identify the potential for recycling and reuse, comparing the impact of using primary and secondary resources. A stakeholder-centered approach will be adopted to visualize material stocks and flows. Urban labs will experiment with new governance arrangements. The sub-objectives include defining the common methodology, developing innovative visualization solutions, co-creating governance arrangements, assessing the impact in urban labs, and co-diffusing best practices for scaling up governance capacity across Europe.

**2. Installation and Configuration**

**System Requirements**

* **Node.js**
* **npm**
* **Browser:** Chrome, Firefox, Edge, or Safari.

**Installation Steps**

1. **Download Source Code:** Download and extract the source code archive on your local machine.
2. **Install Dependencies:** Open a terminal and navigate to the main application directory, then run the following command:

npm install

This command will install all the modules and packages required to run the application.

1. **Run the Application:** After installing all the dependencies, start the application with the following command:

npm start

The application will be available at http://localhost:3000 (possibly at http://localhost:5000).

**3. Application Architecture**

**Directory and File Structure**

The application is organized into a well-defined directory structure that includes source files, static resources, and configuration files. Here's an overview of the main structure:

* **public/:** Contains static files and public resources.
  + index.html: The HTML entry point of the application.
  + city-images/: Images used in various parts of the application.
* **src/:** The main source code directory.
  + App.js: The main application component.
  + index.js: The entry point for the React application.
  + Create\_project.js: Manages the creation of new projects.
  + Dashboard.js: Displays the dashboard view.
  + Forgot-password.js: Handles password recovery.
  + Login.js: User login.
  + Projects.js: Project management.
  + Select\_perimeter.js: Perimeter selection.
  + TutorialUserManual.js: User manual tutorial.
  + **building-info-tabs/:** Components for displaying detailed building information.
  + **map/:** Components for maps and geospatial data visualization.
* **styles/:** SCSS files for styling the application.

**Main Component Descriptions**

* **App.js:** The base component that structures the application and manages the main routes.
* **Dashboard.js:** Provides an overview of current projects and activities.
* **Login.js:** Allows users to log into the application.
* **Create\_project.js:** Form and functionality for adding new projects.

**4. User Guide**

**Main Usage Flows**

1. **Authentication**
   * Users can log in using their credentials to access the application's features.
2. **Creating a Project**
   * Navigate to the "Create Project" section.
   * Fill in the project details and save it.
3. **Viewing Projects**
   * Access the "Projects" section to see a complete list of projects.
4. **Building Information**
   * Select a project to access details about buildings and the materials used.

**Detailed Features**

* **Dashboard:** A quick view of all projects and their status.
* **Project Section:** Managing existing projects and adding new projects.
* **Building Information:** Specific details about each building, including the materials used.

**5. Customization and Extensibility**

**Modifying Styles and Themes**

To customize the application's appearance, edit the .scss files in the styles/ directory. Here you can change variables and CSS classes to alter theme colors, button styles, and other visual aspects.

**Adding New Features**

1. **Creating New Components:** Add new component files in src/. Each new component should have its own .js file and a CSS file for styles.
2. **Updating Routes:** Modify App.js to include the new components in the routing system.
3. **Adding Styles:** Add styles for the new components in the relevant .scss files.

**6. Troubleshooting and Support**

**Common Issues and Solutions**

* **Missing Dependencies:** Make sure you have run npm install.
* **Running Issues:** Check if port 3000/5000 is free, or specify another port in the configuration.
* **Compilation Errors:** Make sure you're using updated versions of Node.js and npm. Check error messages for details.

**Additional Resources**

* Official React Documentation
* Node.js Documentation
* npm Documentation

**Create\_projects.js / Create\_project.module.scss:**

* **Imports:**
  + React and useState from React for state management.
  + styles from a CSS module for component styling.
  + Various Material-UI components (Radio, RadioGroup, FormControlLabel, FormControl, Button) for form controls and buttons.
  + useNavigate from react-router-dom for navigation.
* **Component Definition:**
  + SelectCity is a functional component that accepts multiple props, among which only setWizardStep, setStep, and wizardStep are explicitly destructured. Additional props can be passed through ...props.
* **State Management:**
  + selectedValue is a state variable initialized with "gothenburg" to keep track of the selected city.
* **Event Handling Functions:**
  + handleChange updates the selectedValue state when a radio button is selected.
  + handleBack uses navigate to redirect the user to the /projects route.
  + handleNext uses navigate to redirect the user to the /select\_perim route.
* **JSX Structure:**
  + The component's JSX includes a title, a description, and a form with radio buttons for city selection.
  + Each city option is represented by an image and a FormControlLabel with a radio button.
  + Two buttons ("Back" and "Next") at the bottom allow navigation between steps.

The SCSS component ensures that the interface is well-organized, with each element described for a smoother user experience.

**Dashboard.js / Dashboard.module.scss:**

* **Imports:**
  + **React and useState from React for state management.**
  + **styles from a CSS module for styling.**
  + **Various Material-UI components for building the user interface.**
  + **useNavigate from react-router-dom for navigation between pages.**
* **Component Definition:**
  + **Dashboard is a functional component that manages multiple states to store the values entered in the form.**
* **State Management:**
  + **useState hooks are used to manage the state of form fields, password visibility, and the acceptance of terms and conditions.**
* **Event Handling Functions:**
  + **handleSubmit prevents the default form behavior and may include logic for submitting data.**
  + **handleClickShowPassword toggles the visibility state of the password.**
  + **handleSwitchToLogin redirects the user to the login page using navigate.**
* **JSX Structure:**
  + **The component's JSX includes a title, a description, and a form with text fields for user data entry.**
  + **The text fields are accompanied by icons and buttons to enhance user interaction.**
  + **Two buttons at the bottom allow form submission and redirection to the login page.**

**Forgot-password.js / Forgot-password.module.scss:**

* **Imports: The component imports various functionalities and components from React, React Router, and Material-UI to build the interface and manage state.**
* **Component State:**
  + **email: Stores the email address entered by the user.**
  + **isEmailSent: Indicates whether the password reset email has been sent.**
  + **isLoading: Indicates whether the email-sending process is ongoing.**
* **Navigation: useNavigate is used to enable user redirection to other pages.**
* **handleSubmit Function:**
  + **Prevents the default form behavior (page reload).**
  + **Simulates a loading process by setting isLoading to true.**
  + **After 2 seconds, it simulates sending a password reset request and sets isEmailSent to true and isLoading to false.**
* **handleBack Function: Redirects the user to the login page when the "Back to Login" button is pressed.**
* **Rendering:**
  + **If isEmailSent is false, the password reset form is displayed with a field for entering the email address and a button for submitting the request.**
  + **If isEmailSent is true, a message is displayed informing the user that an email with password reset instructions has been sent, along with a button to return to the login page.**

**Login.js / Login.module.scss:**

* **Imports:**
  + Imports various functionalities and components from React, React Router, and Material-UI to build the interface and manage state.
  + Imports CSS modules for component styling.
* **Component State:**
  + username: Stores the username or email entered by the user.
  + password: Stores the password entered by the user.
  + showPassword: Indicates whether the password is visible or hidden.
  + isSignUp: Indicates if the user is in sign-up mode.
  + isForgotPassword: Indicates if the user is in password reset mode.
* **Navigation:** useNavigate is used to enable user redirection to other pages.
* **handleClickShowPassword Function:** Toggles between showing and hiding the password.
* **handleSubmit Function:**
  + Prevents the default form behavior (page reload).
  + If the user is in sign-up mode, logic for registration should be added (currently commented out).
  + Checks if the username and password are correct (in this case, just a simple example with "1" and "2").
  + If correct, sets authentication in localStorage and redirects to the "Projects" page.
  + If incorrect, displays an error message.
* **handleSwitchToSignUp Function:**
  + Switches to sign-up mode and sets authentication in localStorage to simulate authentication after registration.
  + Redirects to the "Dashboard" page.
* **handleForgotPassword Function:** Redirects the user to the password reset page.
* **handleSwitchToLogin Function:** Switches to login mode.
* **Rendering:**
  + Displays a title and description.
  + Displays a form with fields for entering the username/email and password.
  + Includes a button to show/hide the password.
  + Includes a link for password reset and buttons for login and registration.

**Projects.js / Projects.module.scss:**

* **Imports:**
  + Imports React and hooks such as useState for state management and useNavigate for navigation.
  + Imports Material-UI components for the user interface, such as Button, IconButton, Menu, MenuItem, etc.
  + Imports CSS modules for component styling.
* **ActionsDropdown Component:**
  + Manages the display and hiding of a dropdown menu for each project.
  + anchorEl stores the reference to the element that triggers the menu.
  + handleClick sets the anchorEl reference to the current element and opens the menu.
  + handleClose resets anchorEl and closes the menu.
  + Displays a button for the dropdown and the menu with options: Overview/Edit, Duplicate, Unpin, Rename, Delete.
* **ProjectItem Component:**
  + Receives props (city, name, creationDate) to display the information for each project.
  + getBackgroundImage returns the path to the background image based on the specified city.
  + Renders a card for each project with a background image and the project's information, including the actions dropdown.
* **Main Projects Component:**
  + Uses useNavigate to enable navigation to other pages.
  + handleBack redirects the user to the login page.
  + handleCreateNewProject redirects the user to the new project creation page.
  + handleTutorial redirects the user to the tutorial/user manual page.
* **Rendering:**
  + Displays a title and description.
  + Displays a section with the user's projects, each project being a ProjectItem.
  + Displays additional information about the CREATE tool and how it can be used.
  + Displays buttons for navigation: Back, Create a new project, and Tutorial/User manual.

**SelectPerimeter.js / SelectPerimeter.module.js:**

* **Imports:**
  + Imports React and hooks like useState for state management and useNavigate for navigation.
  + Imports Material-UI components for the user interface, such as Button, Tooltip, IconButton, and HelpOutlineIcon.
  + Imports specific styles for this component from SelectPerimeter.module.scss.
  + Imports the WMSMap component for displaying the map.
* **Component State:**
  + smallMapBounds is an array used to store the boundaries of the selected map.
* **Navigation:**
  + Utilizes useNavigate to enable navigation to other pages.
  + handleBack redirects the user to the project creation page (/create-project).
  + handleNext redirects the user to the buildingTabs page (/buildingTabs).
* **Component Rendering:**
  + Displays a title and description using styles defined in SelectPerimeter.module.scss.
  + Shows general information about inventory data and uses a tooltip to provide additional details. The tooltip is associated with an IconButton that displays a help icon (HelpOutlineIcon).
  + Includes a map (WMSMap) that allows the user to select a perimeter. The map is resized to a height of 400px and uses the setSmallMapBounds function to update the selected boundaries.
  + Displays navigation buttons (Back and Next) that trigger the handleBack and handleNext functions.
* **Export:**
  + Exports the SelectPerimeter component to be used in other parts of the application.

**TutorialUserManual.js / TutorialUserManual.module.scss:**

* Work in progress!

**BuildingInfoTabs.js / BuildingInfoTabs.module.scss:**

* **Imports:**
  + React, useState for state management in React.
  + Tabs, Tab from Material-UI to create tabs.
  + Specific styles from BuildingInfoTabs.module.scss.
  + Components BuildingInformation, MaterialContent, and MaterialQuality, which will be displayed in the tabs.
* **Component State:**
  + value (state) holds the index of the currently active tab. Initialized to 0 to show the first tab.
* **handleChange Function:**
  + Updates the value state when the user changes tabs, allowing switching between different sections.
* **Tabs:**
  + Tabs is used to create a group of tabs. value is set to the current state, and onChange is set to the handleChange function to respond to tab changes.
  + Tab creates each individual tab. The label specifies the text of the tab, and value defines the index of the tab.
* **Tab Content:**
  + buildingInfoTabs is an array that contains the components BuildingInformation, MaterialContent, and MaterialQuality.
  + Depending on the value of the value state, the corresponding component from the array is displayed.
* **Rendering:**
  + Displays the tabs at the top using Tabs and Tab.
  + Below the tabs, the component associated with the active tab is displayed using the index from value.

**BuildingInformation.js / BuildingInformation.module.scss:**

* **Imports:**
  + Imports React and hooks like useState for state management.
  + Imports Material-UI components for the user interface, such as Button, Select, MenuItem, Slider, Tooltip, and IconButton.
  + Imports specific styles for this component from BuildingInformation.module.scss.
  + Imports the WMSMap component for displaying the map.
  + Imports useNavigate for navigating between pages.
* **Component State:**
  + value for the range of construction years.
  + buildingUse, buildingType, buildingStructure, buildingSize, ownershipType for various filters applied to the buildings.
  + demolitionYear for the range of demolition years.
* **Navigation:**
  + Uses useNavigate to enable navigation to other pages.
  + handleBack redirects the user to the perimeter selection page (/select\_perim).
* **Change Handling Methods:**
  + handleBuildingUseChange, handleBuildingTypeChange, handleBuildingStructureChange, handleBuildingSizeChange, and handleOwnershipTypeChange update the respective component states based on the values selected by the user.
  + handleConstructionYearChange and handleDemolitionYearChange update the year ranges for construction and demolition.
  + handleApplyFilters collects all the filters and logs them to the console (you can add logic to send the data to a server or to update the map).
* **Component Rendering:**
  + Displays filters for users, including building use, building type, building structure, building size, ownership type, construction year, and demolition year.
  + Uses Select and Slider to allow users to select and adjust filters.
  + Includes buttons for resetting and applying filters.
  + Displays the map (WMSMap) with a height of 550px.
  + Displays summary information about the number of selected buildings and the total gross area.
  + Includes buttons for navigation (Back and Save and continue).
* **Export:**
  + Exports the BuildingInformation component to be used in other parts of the application.

**MaterialContent.js / MaterialContent.module.scss:**

* **Imports:**
  + React, useState for state management.
  + Components and styles from Material-UI, including Button, Select, MenuItem, Slider, Tooltip, IconButton, TableContainer, Table, TableHead, TableBody, TableRow, TableCell, and Paper.
  + WMSMap for displaying a map.
  + useNavigate from react-router-dom for navigating between pages.
* **MockData:**
  + MockData is an array that contains test data for the table. It is used to populate the table with information about different types of materials.
* **Components:**
  + **DataTable:**
    - Creates a table using Material-UI. The table displays data from MockData in a tabular format, with columns for Skin, Structure, Space, and Total.
  + **MaterialContent:**
    - Uses state to manage filters and displays various sections and components.
* **Functionality:**
  + **Filters:**
    - Includes selectors for Category, Layer, and Element that allow the user to filter data. These selectors are currently non-functional (they use empty onChange functions).
  + **Map:**
    - Includes a WMSMap component that displays a map with the specified height.
  + **Information:**
    - Displays information such as Total mass and Total volume, and includes additional selectors for material classification.
  + **Data Table:**
    - DataTable is used to display data in a tabular format.
  + **Navigation Button:**
    - handleBack navigates back to the previous page (/select\_perim).
* **Styling:**
  + Styles are imported from MaterialContent.module.scss and applied to arrange and style the elements of the component.

**MaterialQuality.js / MaterialQuality.module.scss:**

* **Imports:**
  + React, useState for state management.
  + Components and styles from Material-UI, including Button, Select, MenuItem, Slider, Tooltip, IconButton.
  + WMSMap for displaying a map.
  + MaterialScenariosModal for displaying a modal with material scenarios.
  + useNavigate from react-router-dom for navigation between pages.
* **State:**
  + openScenarios: State to control whether the material scenarios modal is open or closed.
* **Functionality:**
  + **handleBack:**
    - Function for navigating back to the previous page (/select\_perim).
* **Markup:**
  + **Filters:**
    - Includes selectors for Category, Layer, and Element that allow the user to filter information. These selectors are currently non-functional (using empty onChange functions).
  + **Map:**
    - Includes a WMSMap component that displays a map with the specified height.
  + **Information:**
    - Displays messages and buttons to access a material quality form and to open the material scenarios modal.
  + **Modal:**
    - MaterialScenariosModal is used to display material scenarios and is controlled by the openScenarios state.
* **Buttons:**
  + **Material Quality Form:** Button to access a material quality form (functionality not implemented in this code).
  + **Chosen Material:** Button to open the material scenarios modal. When pressed, it sets openScenarios to true.
  + **Save and continue:** Button to save and continue (functionality not implemented in this code).
  + **Export results:** Button to export the results (functionality not implemented in this code).
  + **Back:** Button to navigate back to the previous page.
* **Styling:**
  + Styles are imported from MaterialQuality.module.scss and applied to arrange and style the component's elements.

**Map.js:**

* **Imports:**
  + React, useState, useRef for state management and references.
  + Required components and modules from react-leaflet and leaflet.
  + proj4 for transforming coordinates between different reference systems.
  + data from a JSON file to load GeoJSON data.
  + Necessary styles for Leaflet.
* **Coordinate Transformation:**
  + proj4.defs defines a custom coordinate system (EPSG:3948).
  + convertToGPS transforms coordinates from EPSG:3948 to GPS coordinates (EPSG:4326).
* **Rectangle Size Limitation:**
  + MAX\_RECTANGLE\_WIDTH and MAX\_RECTANGLE\_HEIGHT define the maximum dimensions of the rectangle that can be drawn on the map.
  + Extends the default behavior of drawing rectangles (L.Draw.Rectangle) to respect these limits.
* **Component Map:**
  + **States and References:**
    - editableFG: State for managing the group of editable features.
    - editRef: Reference to the edit control.
  + **Functions:**
    - **onCreated:** Called when a new object is created. Sets the map bounds and ensures there is only one object in the group of editable features.
    - **parsedGeoJson:** Transforms the GeoJSON data using convertToGPS to adapt it to the GPS coordinate system.
    - **onFeatureGroupReady:** Called when the feature group is ready and sets the editableFG state.
* **Markup:**
  + **MapContainer:**
    - Contains a map centered at specified coordinates with a defined zoom level.
  + **TileLayer:**
    - The map source used (OpenStreetMap in this case).
  + **FeatureGroup:**
    - Group for adding and editing drawn objects.
  + **EditControl:**
    - Control to enable/disable drawing and editing objects on the map.
  + **GeoJSON:**
    - Displays GeoJSON data on the map.

**WMSMap.js:**

* **Imports:**
  + React: Imports the necessary React library for creating components.
  + MapContainer, TileLayer, WMSTileLayer from react-leaflet: These components are used to build and customize the map.
  + leaflet/dist/leaflet.css: Includes the necessary CSS styles for Leaflet.
* **WMSMap Component:**
  + **Props:**
    - height: The map height will be set through inline styling of MapContainer.
  + **Markup:**
    - **MapContainer:** The main component that frames the map. Receives center, zoom, and style (height and width) as props.
      * center: Sets the initial center of the map (in GPS coordinates).
      * zoom: The initial zoom level.
      * style: Applies CSS styles to set the map's height and width.
    - **TileLayer:** Adds a background layer for the map using TileLayer from OpenStreetMap.
      * url: The URL for downloading the map's background images.
      * attribution: The necessary attribution for OpenStreetMap.
    - **WMSTileLayer:** Adds a WMS layer from a GeoServer.
      * url: The WMS server URL.
      * layers: The name of the data layer on the WMS server.
      * format: The image format for WMS requests.
      * transparent: Sets the WMS layer's transparency, useful for overlaying it on other layers.