ReactJS Project

Eazy-Admin by Sanele Manyela & Samkelwe Ngalimane

Executive Summary

The React application project aims to make easy activities associated with viewing and editing supplier and contract data. The application uses an API to read and store the data used.

System investigation

- **Software Purpose**

  The software application is being designed to view supplier and contract data as required and be able to access API data. The application will also allow for the contract and supplier data to be updated.

  The application has the following objectives:

  - Aid an administrative user in viewing data of suppliers and that of contractual agreements.
  - Simplify manipulation of data used by the application.

- **Software system capabilities**

  The software application will access API data and display supplier data and contract data as required. The application will also be able to update both the supplier and the contract data.

- **Software system actors**

  The application will be used by administration personnel.

- **Feasibility Study**
  - **Statement of the problem**

    Design and build a platform to view Supplier and Contract Data, using React for the frontend. The platform should be able to access the API Data and display it as required.

      - **Summary of findings**
        - **Technical feasibility**

          React (also known as **React.js** or **ReactJS**) is a free and open-source front-end JavaScript for building user-interface based on UI components. React will be used for creating the platform. React allows working with and making API calls.

        - **Legal feasibility**

          The application will be subject to Protection of Personal Information Act (POPIA) to protect supplier data and contract data. POIPA ensures customer's personal details are treated in a legitimate and respectful manner. The Protection of Personal Information Bill states that personal information must be used explicitly defined and lawful purposes related to a function or activity of your business. It is the business' responsibility to ensure a customer's information is complete, accurate, truthful, and up to date.

- **Project Planning**
  - **Project background**

    Description: This project is an endeavour to design and develop a platform using React as a choice of technology for the frontend development. The platform will be able to access API data and display it as required. The users of the final deliverable will be administrators who will view supplier and contract data.

Project start date: 07 December 2021

Project end date: 21 January 2022

Scheduling strategy: Schedule from project start date

Calendar: The project will use the standard Gregorian calendar. The project working days are Monday to Friday.

- **Project Team**

  Team members

  - Sanele Manyela
  - Samkelwe Ngalimane

  Roles

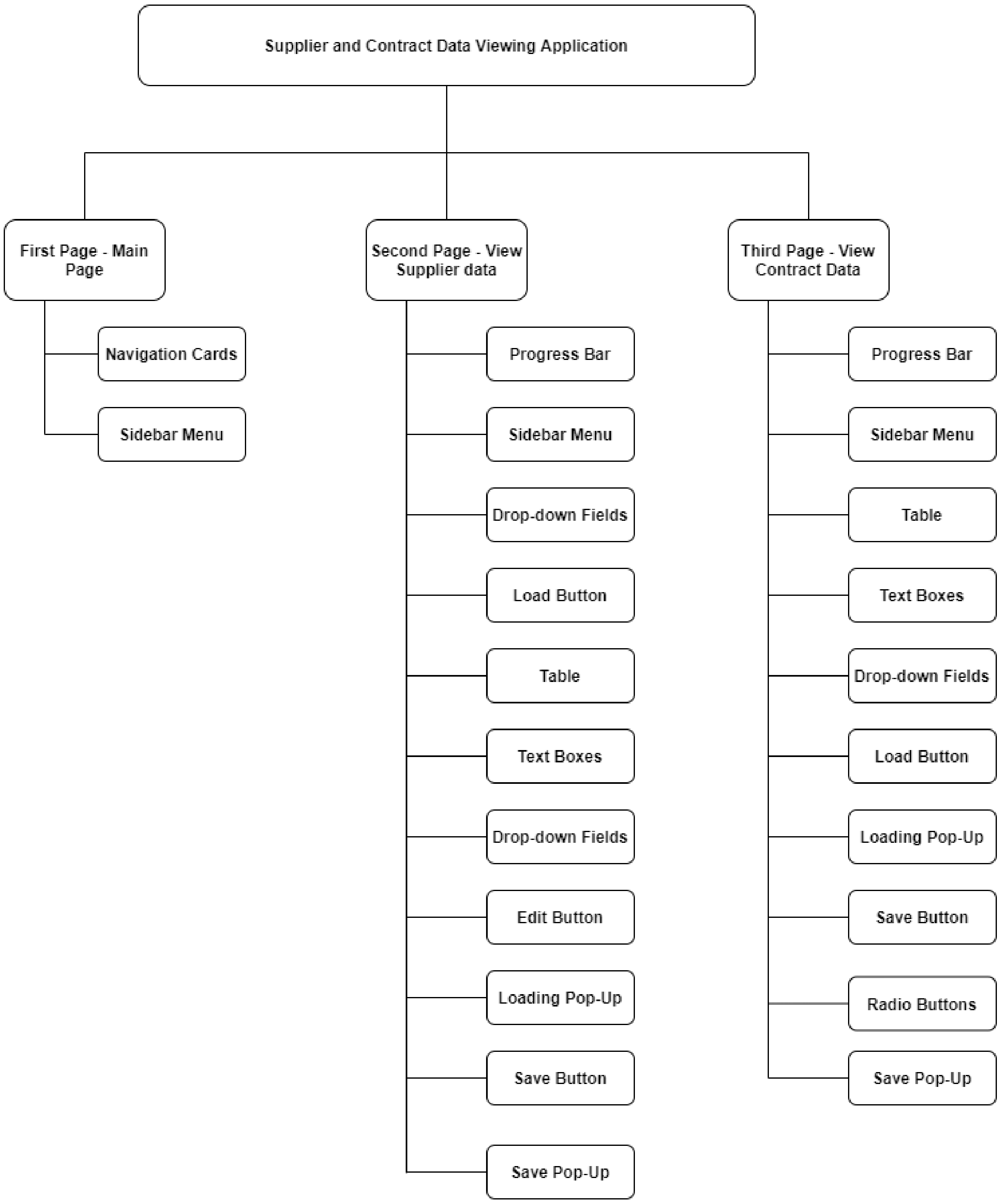
  The team roles for this project are dynamic in their nature, both team members are free to take on the roles of project manager, systems/software analysts, software developer, software tester, and software architecture.

- **Project Scope**

  The scope of the project specifies a requirement to design and build a platform to view supplier and contract data, using React for the frontend. The platform must be able to access API data and display it as required.

  The platform must be comprised of three pages: the front and main page with navigation to other two pages, the second page will display supplier data that can also be updated, and the third page will display contract data that can also be updated.

- **Product Breakdown Structure**

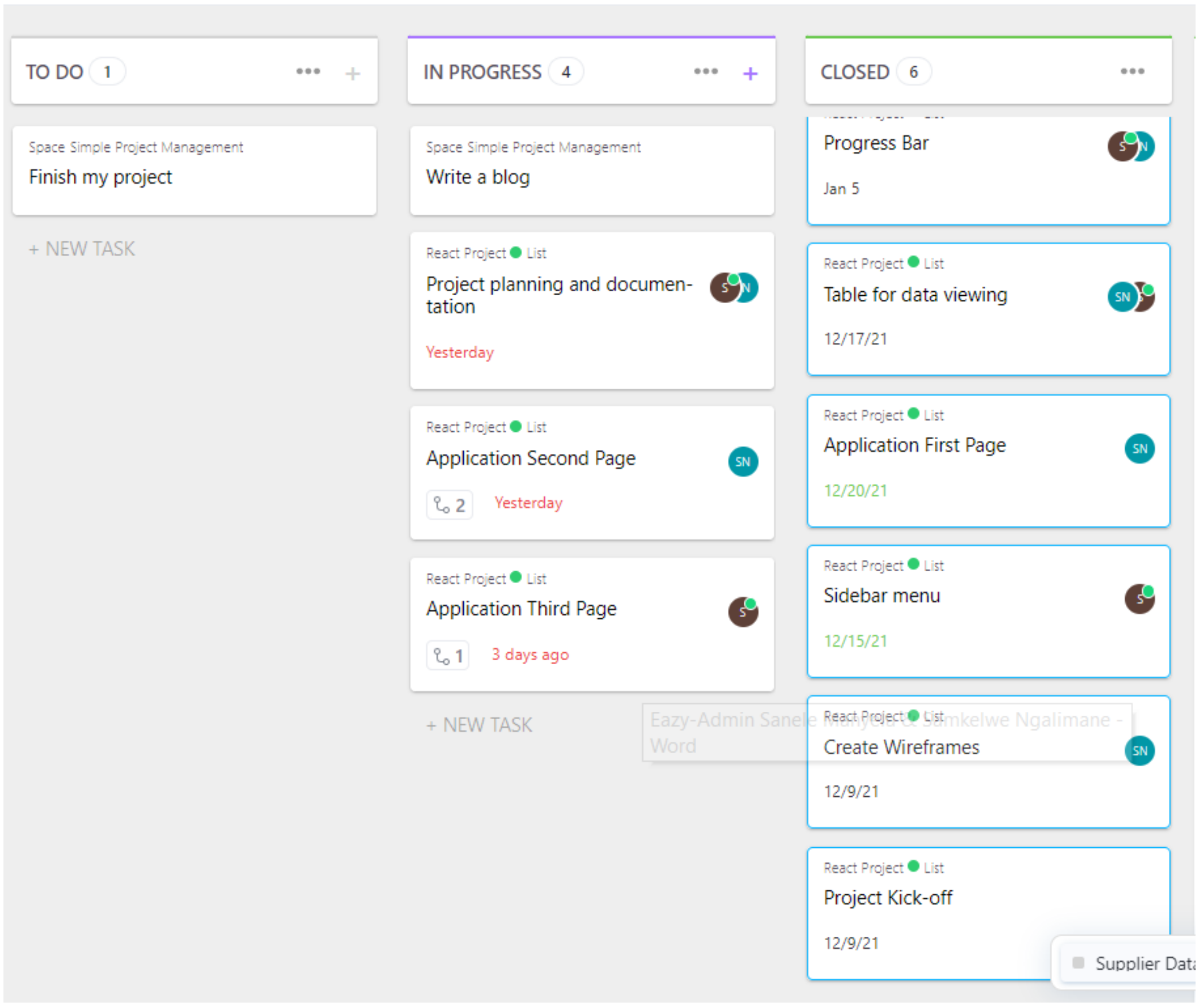


The illustrated figure above depicts the project's product breakdown structure. The project deliverable will consist of five main components: the landing page component, the page component for viewing supplier data, the page component for viewing contracts data, the page component for updating supplier data, and the page component for updating contract data. Each of these components is further divided to self-contained components as shown in the illustration.

- **Assignment of Activities**

S = Sanele Manyela

SN = Samkelwe Ngalimane


TO DO 1

Space Simple Project Management
Finish my project

+ NEW TASK

IN PROGRESS 4

Space Simple Project Management
Write a blog

React Project List
Project planning and documen-tation
Yesterday

React Project List
Application Second Page
2 Yesterday

React Project List
Application Third Page
1 3 days ago

+ NEW TASK

CLOSED 6

Progress Bar
Jan 5

React Project List
Table for data viewing
12/17/21

React Project List
Application First Page
12/20/21

React Project List
Sidebar menu
12/15/21

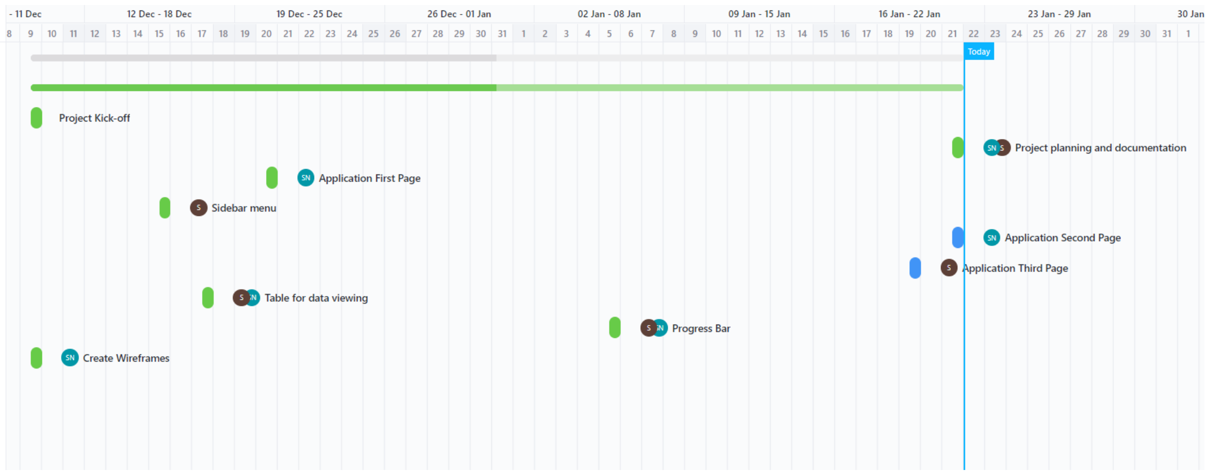
React Project List
Create Wireframes
12/9/21

React Project List
Project Kick-off
12/9/21

Eazy-Admin Sanele Manyela & Samkelwe Ngalimane - Word

Supplier Data

- **Timeline**



- ## Software System Analysis
  - **Information and requirements gathering**

    Information and requirements were gathered using group meeting techniques and the scope document.
    - **Definition of requirements**
      - The application should contain three pages
        - Front Page, with cards
        - Second Page: With Supplier Data
        - Third Page with Contract Data
      - When each page loads, a progress bar should appear, notifying the user the data is loading.
      - Front Page Tile Menu should have three cards.
        - ▪ There should be a card that navigates to each of the following:
          - ○ Navigate to the following URL: www.google.com
          - ○ Second Page: With Supplier Data
          - ○ Third Page with Contract Data
      - ▪ The application requires a sidebar menu, with drawer functionality, the sidebar menu should have a selection for Home, Supplier Reporting, Contract Reporting.

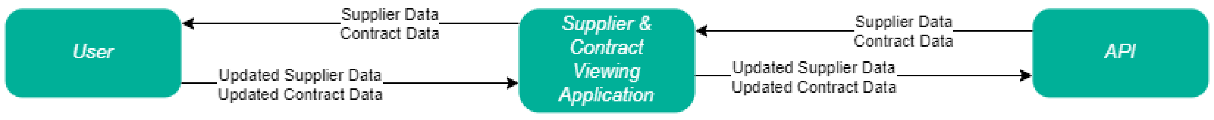
- The second page should have two dropdown fields and a button, as well as a table, which should be filtered by the dropdown selection.
  - Drop Down 1- Year
  - Drop Down 2 – Month
  - Button – Load
  - Table – Data from the Supplier Master Data API.

  - OnClick of Table Row, Supplier Captured Data should be displayed in Editable Field (Either Drop Down or Textbox) and a button to Save.
    - The Save button should be greyed out and only useable when data is changed in the fields
    - A loading popup should be displayed
    - A Saved popup should be displayed
- Third Page should have only a table, which should load when the page is selected.
  - Table – Data from the Contract Master Data API.
  - OnClick of Table Row, Contract Capture Data should be displayed in Editable Field (Either Drop Down or Textbox) and a button to Save.
    - The Save button should be greyed out and only useable when data is changed in the fields
    - A loading popup should be displayed
    - A Saved popup should be displayed
    - The Following fields should be drop down menu
      - LEDSplit
        - MBSA
        - DTBSA
        - MBFS

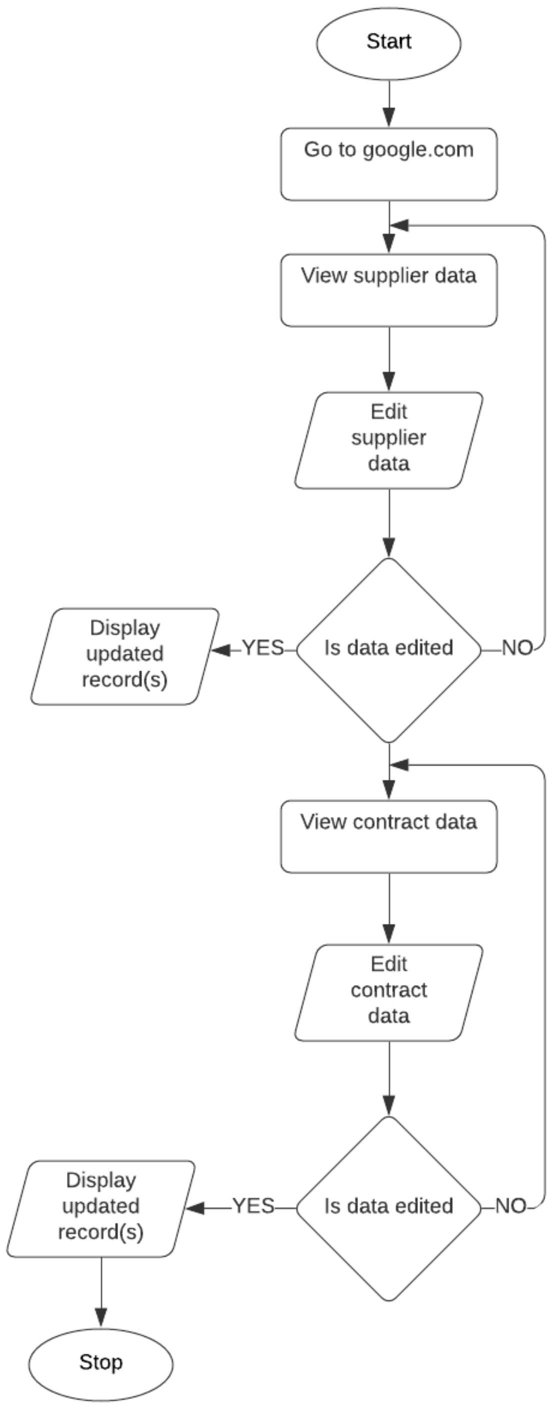
            - Trucks
            - Vans
            - SMH
        - Classification
            - A
            - B
    - The following fields are radio buttons
        - Plant Relevant
            - Yes
            - No
        - Saved to Server
            - Yes
            - No
    - All date fields should display today's date

- **Prioritisation of Requirements**
    1. Sidebar menu
    2. Table for data viewing
    3. Application First Page
    4. Progress Bar
    5. Application Second Page
    6. Application Third Page
    7. Update Supplier Data
    8. Update Contract Data

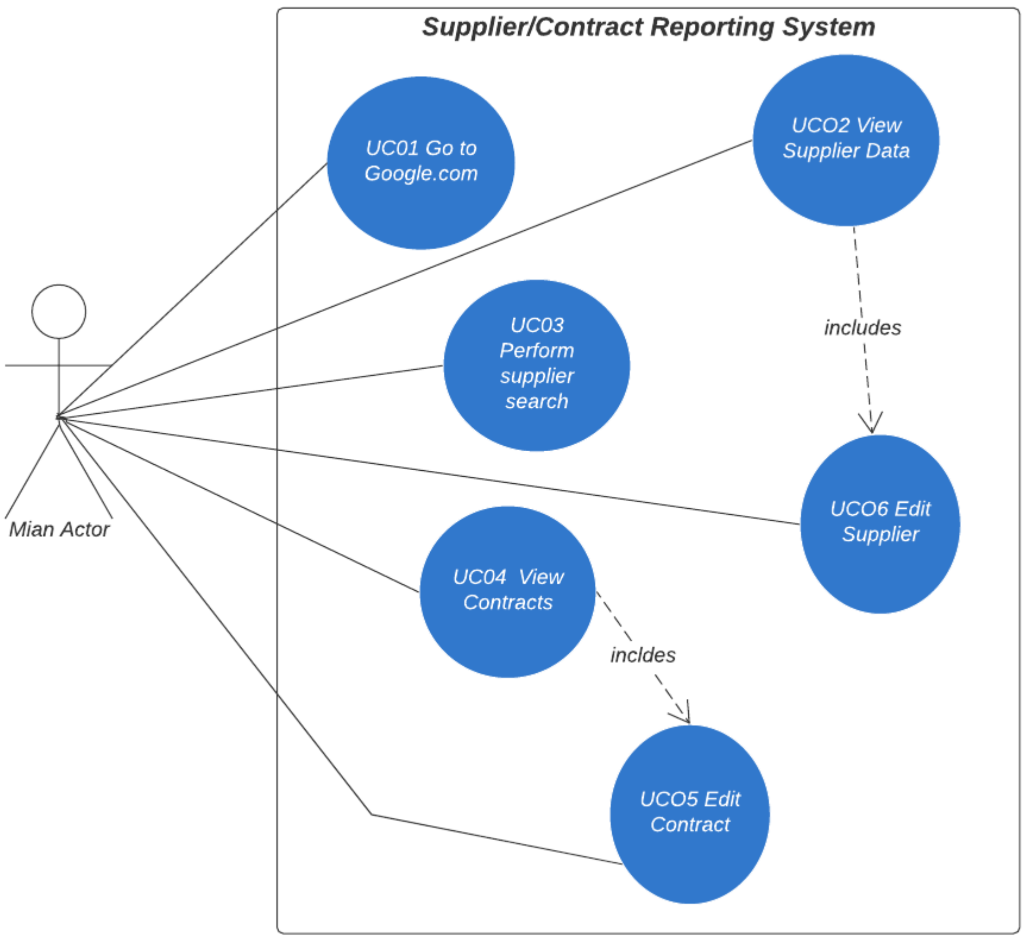
- ## Software System Design
    - ### Context Diagram



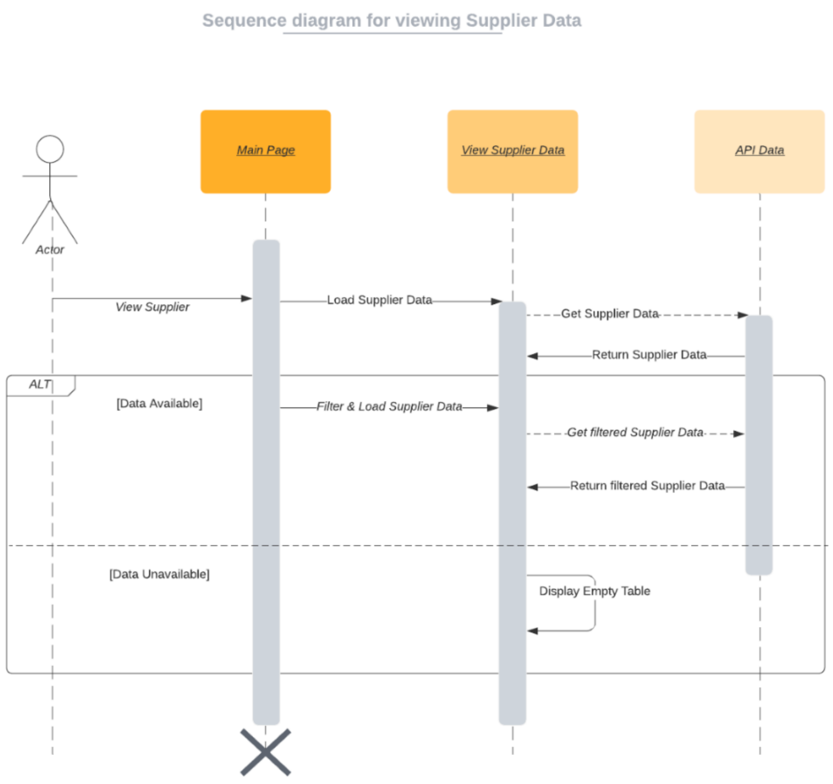
- ## Flowchart Diagram



- ## Use Case Diagram

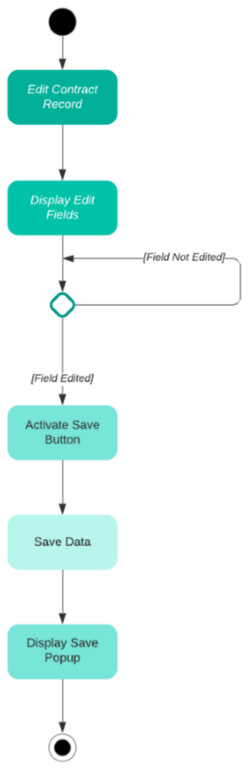


○ Sequence Diagram



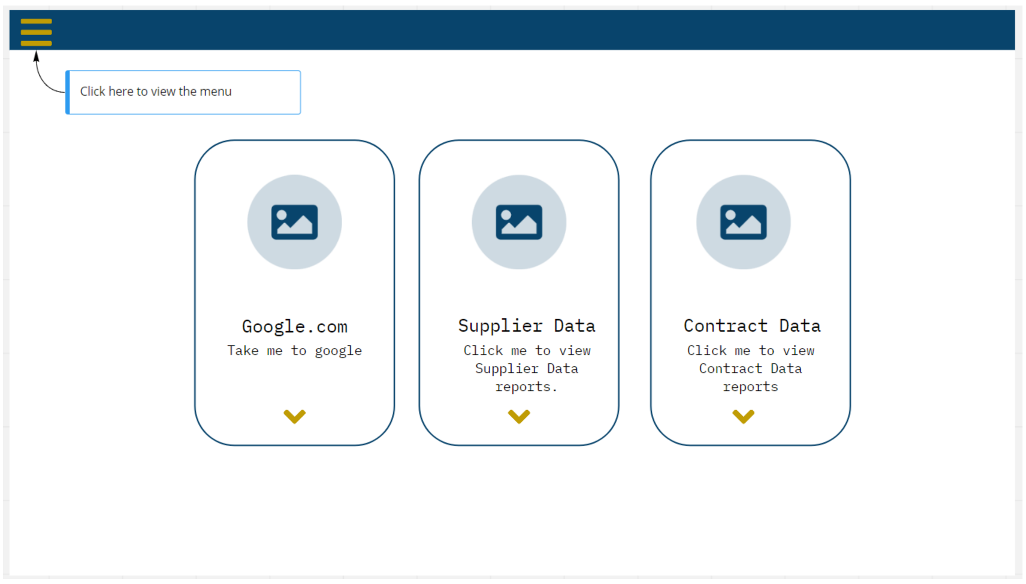
- ## Activity Diagram

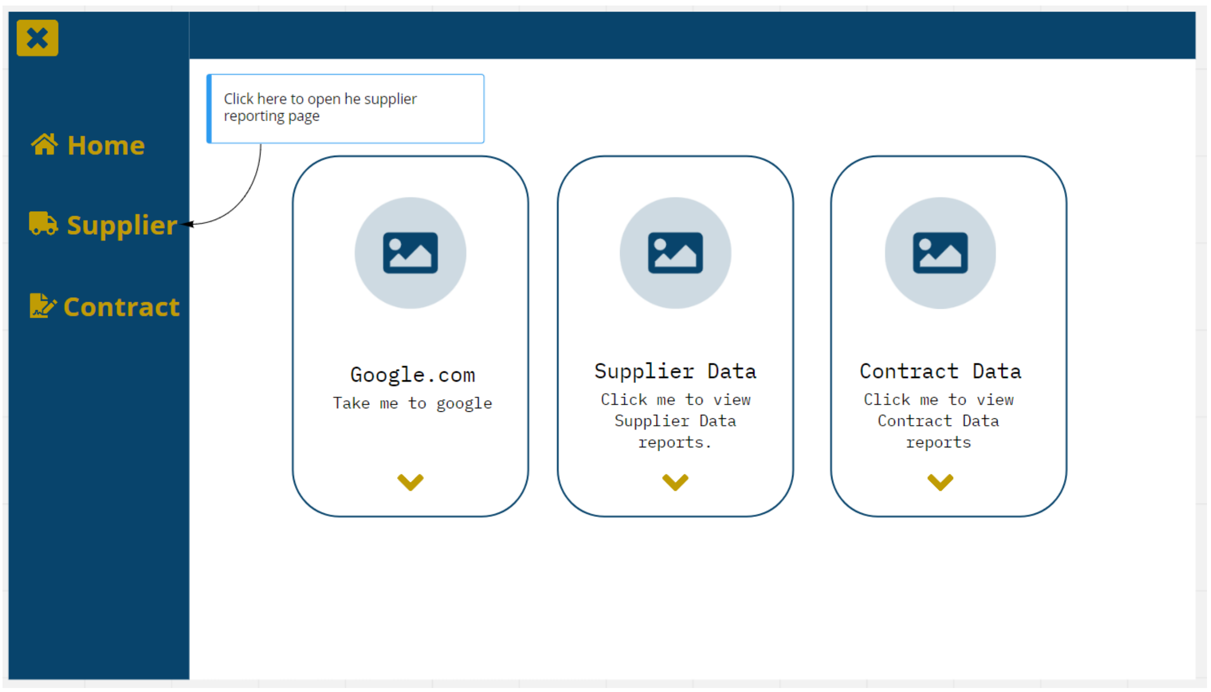
Activity diagram for editing contract data



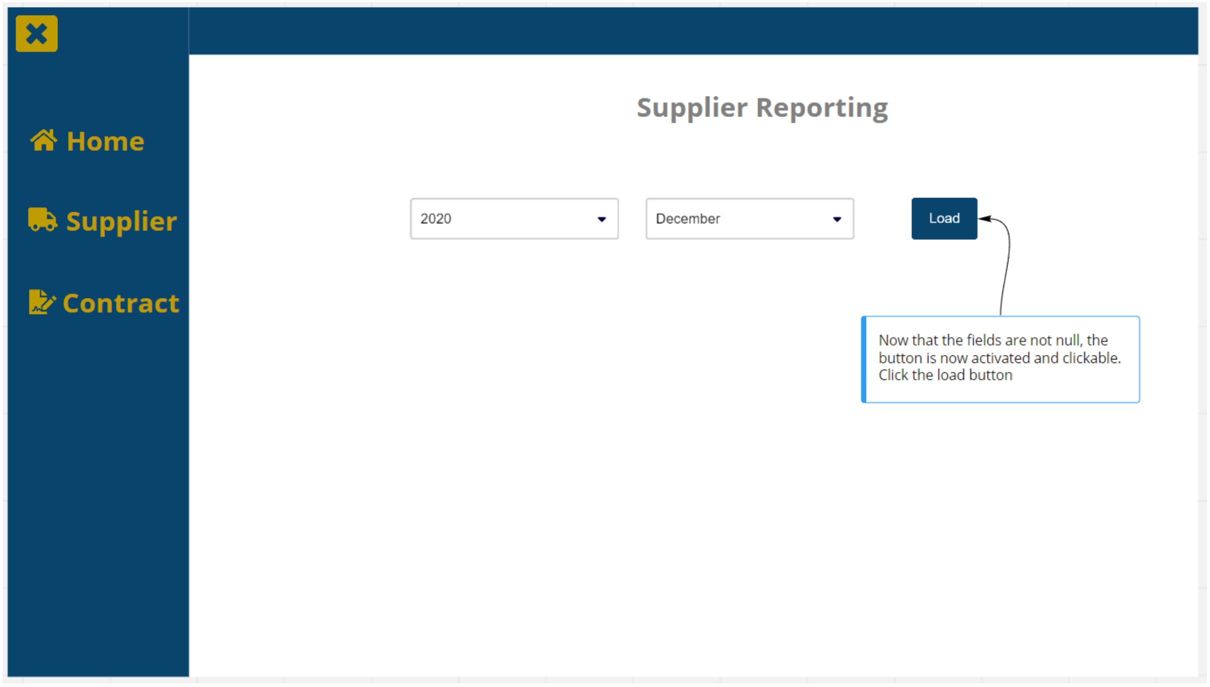
- ## User Interface Design and consideration

Home Page

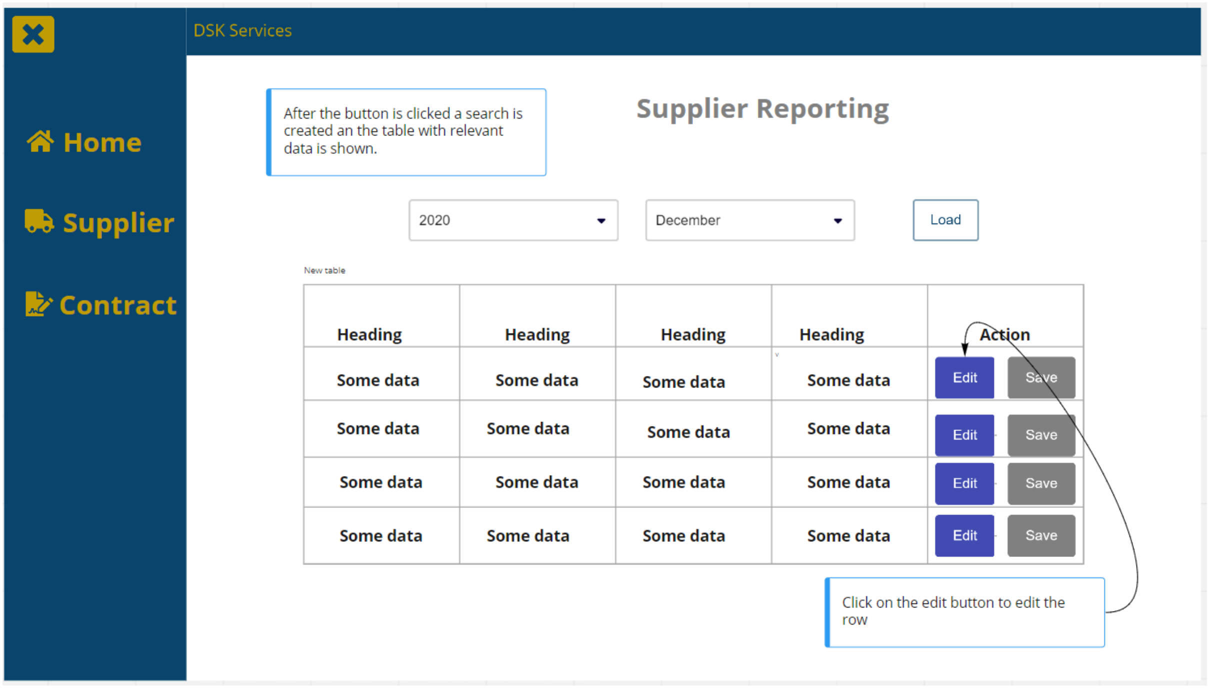

Click here to view the menu
Google.com
Take me to google
Supplier Data
Click me to view Supplier Data reports.
Contract Data
Click me to view Contract Data reports


Home
Supplier
Contract
Click here to open he supplier reporting page
Google.com
Take me to google
Supplier Data
Click me to view Supplier Data reports.
Contract Data
Click me to view Contract Data reports

Supplier Page



Now that the fields are not null, the button is now activated and clickable. Click the load button

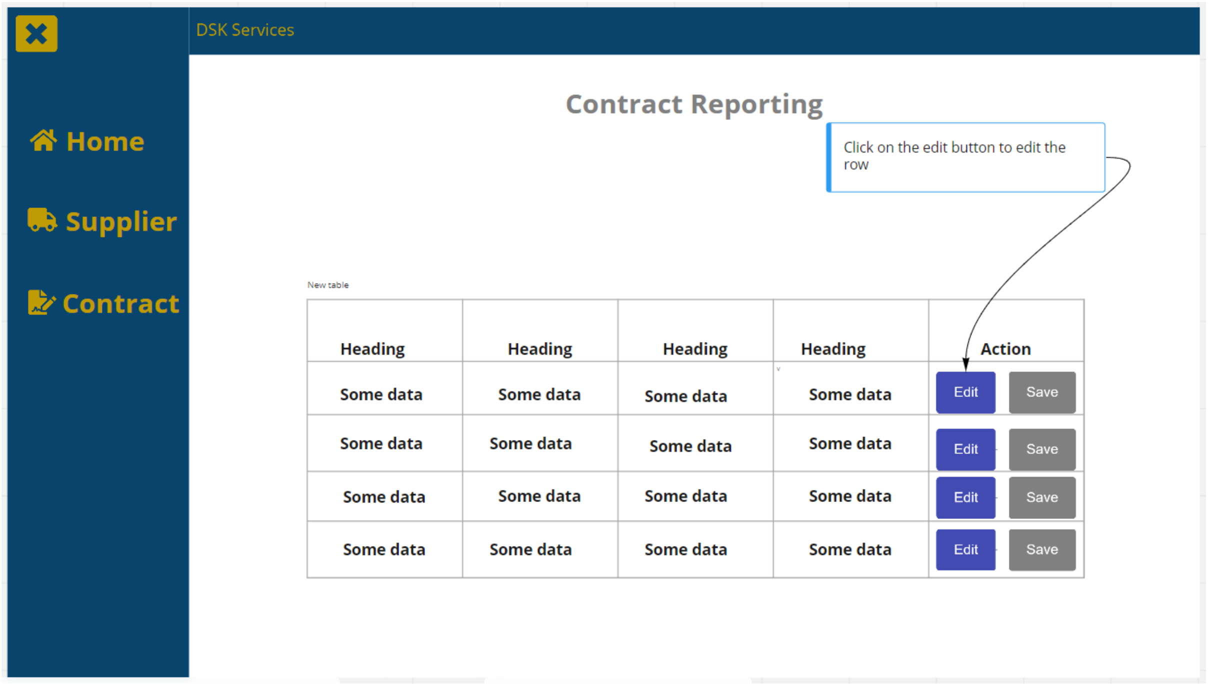


After the button is clicked a search is created an the table with relevant data is shown.

Heading	Heading	Heading	Heading	Action
Some data	Some data	Some data	Some data	Edit Save
Some data	Some data	Some data	Some data	Edit Save
Some data	Some data	Some data	Some data	Edit Save
Some data	Some data	Some data	Some data	Edit Save

Click on the edit button to edit the row

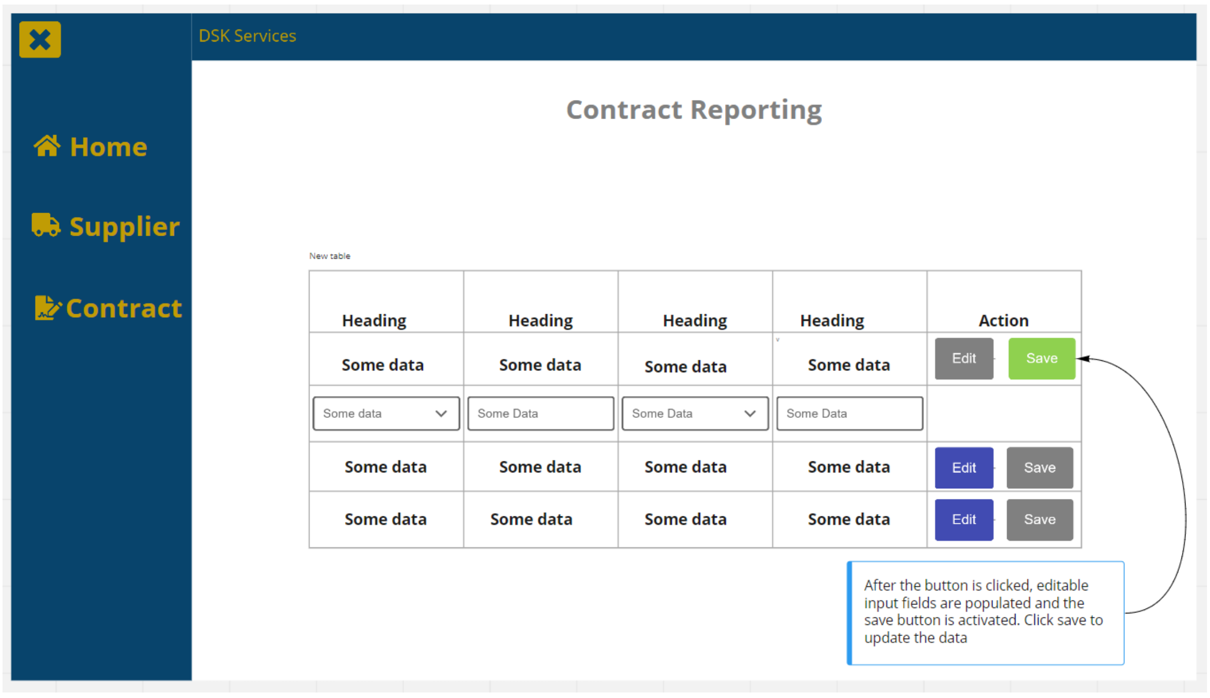
Contract Page



Contract Reporting

Click on the edit button to edit the row

Heading	Heading	Heading	Heading	Action
Some data	Some data	Some data	Some data	Edit Save
Some data	Some data	Some data	Some data	Edit Save
Some data	Some data	Some data	Some data	Edit Save
Some data	Some data	Some data	Some data	Edit Save



Contract Reporting

Heading	Heading	Heading	Heading	Action
Some data	Some data	Some data	Some data	Edit Save
Some data	Some Data	Some Data	Some Data	
Some data	Some data	Some data	Some data	Edit Save
Some data	Some data	Some data	Some data	Edit Save

After the button is clicked, editable input fields are populated and the save button is activated. Click save to update the data

- ## Implementation

  This application documented here is developed using ReactJS, a free and open-source front-end JavaScript library for building user-interface based on UI components. Inside some of these components is imbedded code that connects to an API endpoint to either extract or write data to.
  The API endpoints are tested on Postman, an online collaboration platform for API development.

  The application code is developed in Visual Studio Code, this makes it easy for code to be uploaded and download from a version control application such as GitHub.

  The **testing strategy** is Development Driven Testing.

  - First Page - **Home** Tests

    Test Case 1: [

    IDENTIFIER: Sliding bar navigation
    TEST CASE: Test if the sliding bar menu item can navigate to the Supplier Page.
    PRECONDITIONS: None
    INPUT VALUES: None
    EXECUTION STEPS: On the application's landing page, click the three-line icon at the left top corner of the interface. On display of a slide bar menu, select Supplier.
    OUTPUT VALUES: N/A
    POSTCONDITION: The application navigates to the Supplier interface.

    ]

Test Case 2: [

IDENTIFIER: Sliding bar navigation

TEST CASE: Test if the sliding bar menu item can navigate to the Contract Page.

PRECONDITIONS: None

INPUT VALUES: None

EXECUTION STEPS: On the application’s landing page, click the three-line icon at the left top corner of the interface, on display of a slide bar menu select Contract.

OUTPUT VALUES: N/A

POSTCONDITION: The application navigates to the Contract interface.

]

Test Case 3: [

IDENTIFIER: Sliding bar navigation

TEST CASE: Test if the sliding bar menu item can navigate to the Home Page.

PRECONDITIONS: The application must be in the Contract Page interface.

INPUT VALUES: None

EXECUTION STEPS: On the application’s landing page, click the three-line icon at the left top corner of the interface. On display of a slide bar menu,
select Contract. When inside the Contract Page interface, click in the slide bar again and navigate to Home.

OUTPUT VALUES: N/A

POSTCONDITION: The application navigates to the Home interface.

]

Test Case 4: [

IDENTIFIER: Navigation with Cards

TEST CASE: Test if the Google navigation card navigates to Google.com

PRECONDITIONS: None

INPUT VALUES: None

EXECUTION STEPS: On the application landing page, click the Google.com card, the first card from the left of the three navigation cards.

OUTPUT VALUES: N/A

POSTCONDITION: The application redirects to a Google.com search interface.

]

Test Case 5: [

IDENTIFIER: Navigation with Cards

TEST CASE: Test if the Supplier navigation card navigates to Supplier interface.

PRECONDITIONS: None

INPUT VALUES: None

EXECUTION STEPS: On the application’s landing page, click the Supplier Data card.

OUTPUT VALUES: N/A

POSTCONDITION: The application redirects to Supplier Page interface.

]

Test Case 6: [

IDENTIFIER: Navigation with Cards
TEST CASE: Test if the Contract navigation card navigates to the application’s Contract Page
PRECONDITIONS: None
INPUT VALUES: None
EXECUTION STEPS: On the application’s landing page, click the Contract card
OUTPUT VALUES: N/A
POSTCONDITION: The application redirects to Contract Page interface.

]

- Third Page Tests – **Contract**

Test Case 1: [

IDENTIFIER: Contract Data Retrieval
TEST CASE: Test if the application retrieves the data that will be displayed on the Contract Page table
PRECONDITIONS: Implement a standard method of writing to the console.
INPUT VALUES: None
EXECUTION STEPS: On the application’s landing page, click the Contract card.
OUTPUT VALUES: N/A
POSTCONDITION: The application navigates to the Contract Page interface and the data that is retrieved from the Contract Master Data API is displayed

]

Test Case 2: [

IDENTIFIER: Display Contract Data on table with some fields omitted.

TEST CASE: Test if the application can navigate to the Contract Page and view a table displaying data.

PRECONDITIONS: None

INPUT VALUES: None

EXECUTION STEPS: On the application's landing page, click the Contract card at the centre of the interface.

OUTPUT VALUES: N/A

POSTCONDITION: The application navigates to the Contract interface and displays a table with contract master data.

]

Test Case 3: [

IDENTIFIER: Contract Page table's capability to handle onClick events.

TEST CASE: Test if the Contract Page table is capable of handling onClick events.

PRECONDITIONS: Implement a standard method of writing to the console.

INPUT VALUES: None

EXECUTION STEPS: On the application's landing page, click the Contract card and navigate to the Contract Page. Inside the Contract Page interface, click rows at random.

OUTPUT VALUES: N/A

POSTCONDITION: The Contract Page's table can handle onClick events.

]

Test Case 4: [

IDENTIFIER: Contract Page table’s capability to get data from an API.

TEST CASE: Test if the Contract Page table can get contract capture date by ID from an API.

PRECONDITIONS: Implement a standard method of writing to the console.

INPUT VALUES: Contract capture data ID.

EXECUTION STEPS: On the application’s landing page, click the Contract card and navigate to the Contract Page. Inside the Contract Page interface, click a row, then open browser console.

OUTPUT VALUES: N/A

POSTCONDITION: The Contract Page’s table can extract data from an API by ID and display the data in the console.

]

Test Case 5: [

IDENTIFIER: Contract Page table row renders a modal.

TEST CASE: Test if the table rows of the Contract Page table render a modal at an onClick event.

PRECONDITIONS: None

INPUT VALUES: a data array, a Boolean value for whether to show the modal, another Boolean value to handle the close event.

EXECUTION STEPS: On the application’s landing page, click the Contract card and navigate to the Contract Page. Inside the Contract Page interface, click any row and bring up a modal containing values of a certain contract data in the data reserve.

OUTPUT VALUES: N/A

POSTCONDITION: A modal is brought up with values of a particular contract data.

]

Test Case 6: [

IDENTIFIER: Contract Page table row renders a modal.
TEST CASE: Test if the table rows of the Contract Page table render a modal at an onClick event
PRECONDITIONS: Data to pass to the modal
INPUT VALUES: a data array, a Boolean value for whether to show the modal, another Boolean value to handle the close event.
EXECUTION STEPS: On the application’s landing page, click the Contract card and navigate to the Contract Page. inside the Contract Page interface, click any row and bring up a modal containing values of a certain contract data in the data reserve.
OUTPUT VALUES: N/A
POSTCONDITION: A modal is brought up with values of a particular contract data.

]

Test Case 7: [

IDENTIFIER: Contract Page modal state rendering.
TEST CASE: Test if the Contract Page modal renders a different state each time the modal is brought up.
PRECONDITIONS: Data to pass to the modal
INPUT VALUES: a data array, a Boolean value for whether to show the modal, another Boolean value to handle the close event.
EXECUTION STEPS: On the application’s landing page, click the Contract card and navigate to the Contract Page. Inside the Contract Page interface, click rows at random and bring up a modal containing values of a different contract data each time the modal is brought up.
OUTPUT VALUES: N/A
POSTCONDITION: A modal is brought up with values of a different contract data each time.

COMMENTS & OBSERVATIONS: When the modal is rendered and brought up the first time, it displays the correct data and intended states. The second or n-time the modal is brought up, the modal displays state 1. On debugging, the modal does receive new data each time brought up, the data if for setting states, states are set but the state that gets saved is state 1.

]

Test Case 8: [

IDENTIFIER: Contract Page modal save-button toggle.

TEST CASE: Test if the save button of the Contract Page modal is disabled when the modal is brought up.

PRECONDITIONS: Data to pass to the modal

INPUT VALUES: a data array, a Boolean value for whether to show the modal, another Boolean value to handle the close event.

EXECUTION STEPS: On the application's landing page, click the Contract card and navigate to the Contract Page. Inside the Contract Page interface, click a row to bring up a modal containing values of a certain contract data. Click the save button.

OUTPUT VALUES: N/A

POSTCONDITION: The save button will not trigger any event.

]

Test Case 9: [

IDENTIFIER: Contract Page modal save-button toggle.
TEST CASE: Test if the save button of the Contract Page modal is enabled when data in the modal input fields is edited.
PRECONDITIONS: Data to pass to the modal
INPUT VALUES: a data array, a Boolean value for whether to show the modal, another Boolean value to handle the close event.
EXECUTION STEPS: On the application’s landing page, click the Contract card and navigate to the Contract Page. Inside the Contract Page interface, click a row to bring up a modal containing values of a certain contract data. Edit any of the modal fields, then click the save button.
OUTPUT VALUES: N/A
POSTCONDITION: The save button will trigger a submit event that updates data.

]

Test Case 10: [

IDENTIFIER: Contract Page modal save-button toggle.
TEST CASE: Test if the save button of the Contract Page modal is enabled when data in the modal input fields is edited.
PRECONDITIONS: Data to pass to the modal
INPUT VALUES: a data array, a Boolean value for whether to show the modal, another Boolean value to handle the close event.
EXECUTION STEPS: On the application’s landing page, click the Contract card and navigate to the Contract Page. Inside the Contract Page interface, click a row to bring up a modal containing values of a certain contract

data. Edit any of the modal fields, then click the save button.

OUTPUT VALUES: N/A

POSTCONDITION: The save button will trigger a submit event that updates data.

]

Test Case 11: [

IDENTIFIER: Contract Page modal close button.

TEST CASE: Test if the close button of the Contract Page modal closes the modal.

PRECONDITIONS: Data to pass to the modal

INPUT VALUES: a data array, a Boolean value for whether to show the modal, another Boolean value to handle the close event.

EXECUTION STEPS: On the application’s landing page, click the Contract card and navigate to the Contract Page. Inside the Contract Page interface, click a row to bring up a modal containing values of a certain contract data. Edit any of the modal fields, then click the save button.

OUTPUT VALUES: N/A

POSTCONDITION: The close button toggles the modal display to false, thereby closing the modal.

]

Test Case 12: [

IDENTIFIER: Contract Page modal save-button.

TEST CASE: Test if the logic of the save button can update and write to the API.

PRECONDITIONS: Data to pass to the modal

INPUT VALUES: a data array, a Boolean value for whether to show the modal, another Boolean value to handle the close event.

EXECUTION STEPS: On the application’s landing page, click the Contract card and navigate to the Contract Page. Inside the Contract Page interface, click a row to bring up a modal containing values of a certain contract data. Edit any of the modal fields, then click the save button.

OUTPUT VALUES: N/A

POSTCONDITION: The save button will write to the API the user-updated values.

]

- Second Page Tests – **Supplier**

  Test Case 1: [

  IDENTIFIER: Supplier-Page month-year-load interface.

  TEST CASE: Test if the values of the month and year input fields can be selected and set as states.

  PRECONDITIONS: The month input field must be populated with the standard twelve months, and the year input field must be populated with years 2016 to present.

  INPUT VALUES: month and year.

  EXECUTION STEPS: On the application's landing page, click the Supplier card and navigate to the Supplier Page. Inside the Supplier Page interface, select a month, then a year, and click the load button.

  OUTPUT VALUES: N/A

  POSTCONDITION: Write to the console the values of the month and year selected.

  ]

Test Case 2: [

IDENTIFIER: Supplier-Page month-year-load interface.

TEST CASE: Test if on the event of clicking the load button, data is retrieved from the Supplier-Master-Data API.

PRECONDITIONS: The month input field must be populated with the standard twelve months, and the year input field must be populated with years 2016 to present.

INPUT VALUES: month and year.

EXECUTION STEPS: On the application's landing page, click the Supplier card and navigate to the Supplier Page. Inside the Supplier Page interface, select a month, then a year, and click the load button.

OUTPUT VALUES: N/A

POSTCONDITION: Write to the console the data from the API given the month and year as parameters.

]

Test Case 3: [

IDENTIFIER: Supplier-Page table-interface.

TEST CASE: Test if a table component, passed the data, renders, and displays the data.

PRECONDITIONS: The Supplier Master Data must be already fetched from an API endpoint.

INPUT VALUES: month and year.

EXECUTION STEPS: On the application’s landing page, click the Supplier card and navigate to the Supplier Page. Inside the Supplier Page interface, select a month, then a year, and click load button.

OUTPUT VALUES: N/A

POSTCONDITION: Supplier table interface is displayed.

]

Test Case 4: [

IDENTIFIER: Supplier-Page table-modal-interface.

TEST CASE: Test if a modal component of the Supplier-Page table can display, even without data passed.

PRECONDITIONS: Be inside the Supplier-Page table-interface.

INPUT VALUES: None.

EXECUTION STEPS: Inside the Supplier-Page table interface, click one of the table rows.

OUTPUT VALUES: N/A

POSTCONDITION: A modal is brought up.

]

Test Case 5: [

IDENTIFIER: Supplier-Page table interface.

TEST CASE: Test if the Supplier-Page table component can fetch Supplier-Capture-Data by ID from the API.

PRECONDITIONS: Be inside the Supplier-Page table-interface. Implement a standard console output to display the capture data.

INPUT VALUES: supplier-Capture-Data-ID.

EXECUTION STEPS: Inside the Supplier-Page table interface, click one of the table rows.

OUTPUT VALUES: N/A

POSTCONDITION: Supplier Capture Data is displayed in the console.

]

Test Case 6: [

IDENTIFIER: Supplier-Page table interface.

TEST CASE: Test if a modal component of the Supplier-Page table can display the data it is passed.

PRECONDITIONS: Be inside the Supplier-Page table-interface.

INPUT VALUES: Supplier Capture Data.

EXECUTION STEPS: Inside the Supplier-Page table interface, click one of the table rows.

OUTPUT VALUES: N/A

POSTCONDITION: A modal is brought up displaying the data it was passed.

]

- **User Manual**
  - **Description of the software system.**
    The software application is designed to view supplier data by passing date and contract data as required and be able to access API data. The application will also allow for the contract and supplier data to be updated.
    The application is capable of:
    - Smooth and easy navigation.

    - Navigates to Google.com, navigates to Supplier Data, navigates to Contract Data.

    - Supplier Data – displays a Supplier data table depending on the selected month and year.
      The table has clickable rows that bring up a
      modal filled with values related to each individual row.
      The values in the modal may be updated.

    - Contract Data – displays a Contract data table.
      The table has clickable rows that bring up a modal
      filled with values related to each individual row.
      The values in the modal may be updated.

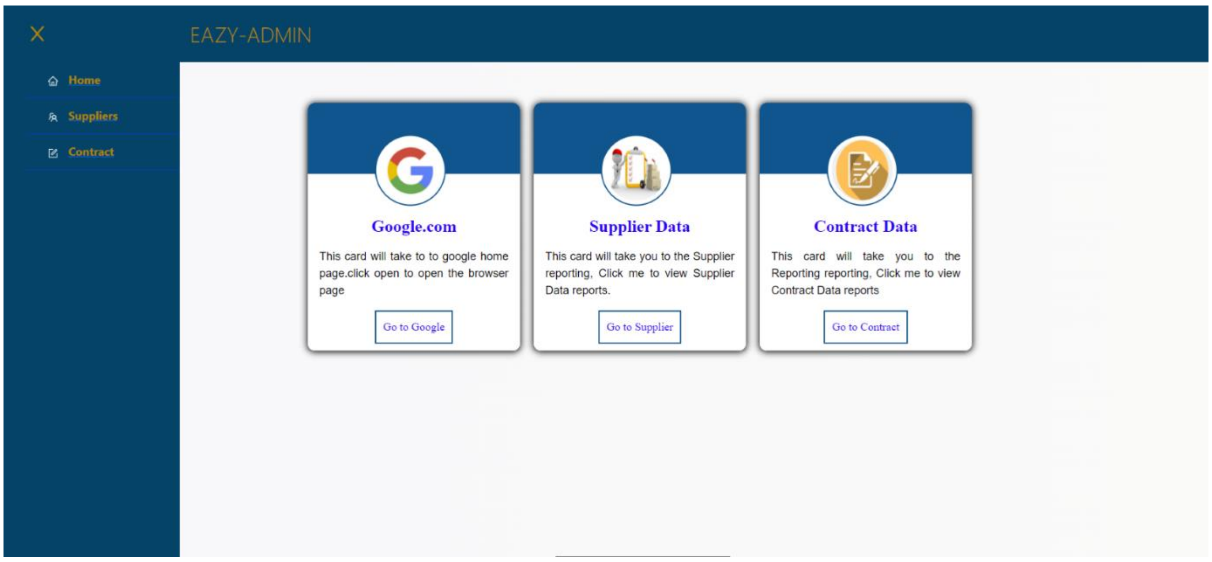
  - **Environment to run the system in.**
    The application is run on Node.js and uses most of the standard browsers.

- **User Interfaces**

1. **Application Landing Page – Home:**

   This is the first interface of the application. Its purpose is navigation, it allows a user to navigate to different interfaces. The home interface uses two components for routing capabilities: the sliding bar and the navigation cards.

   - In the sliding bar, the home item routes to the landing page, the Supplier item routes to Supplier Page interface, the Contract item routes to the Contract Page interface.
   - In the navigation card, the Google.com card routes to Google.com, the Supplier Data card routes to Supplier Page, the Contract Data card routes to Contract Page.



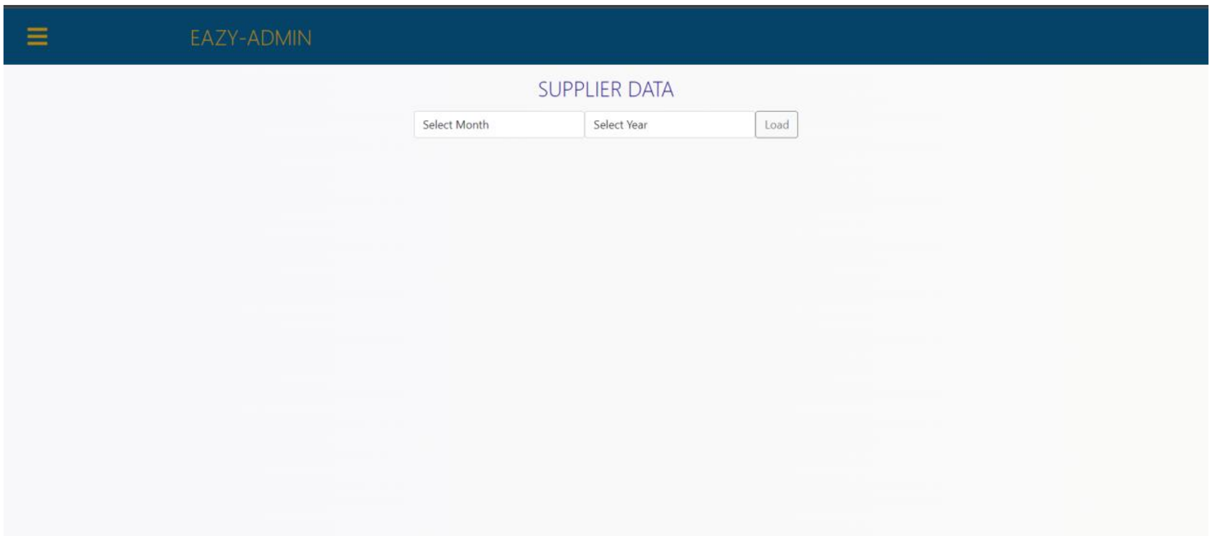
This card will take to to google home page.click open to open the browser page

This card will take you to the Supplier reporting, Click me to view Supplier Data reports.

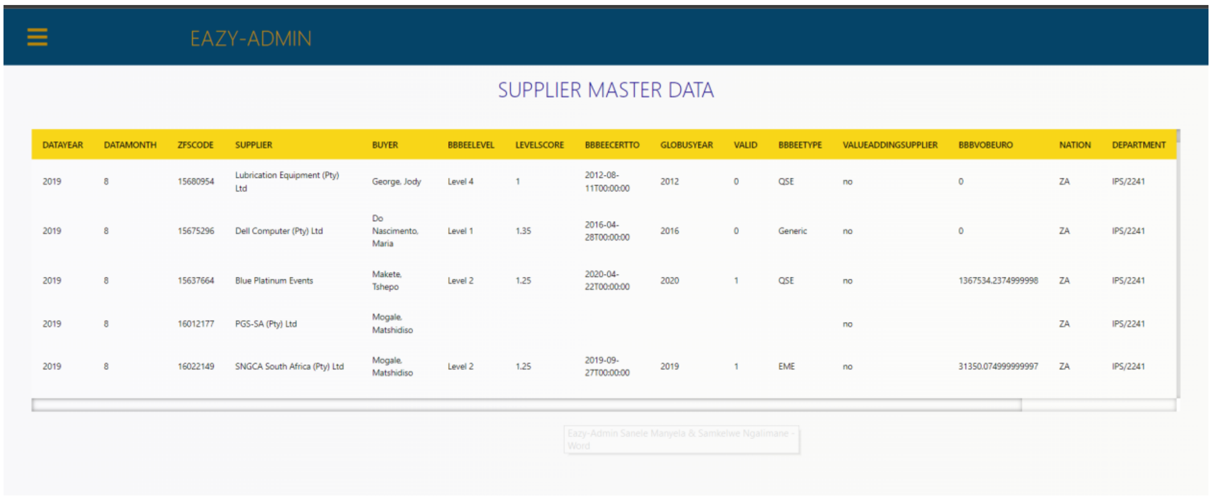
This card will take you to the Reporting reporting, Click me to view Contract Data reports

2. Application second page – Supplier

The Supplier page displays an interface with two input controls and a button. A user selects the month and year the click load button to load supplier data of the that perio



After loading the data, it is displayed in the table of Supplier Page. On row click, the table component retrieves data from an API. This data is passed to a modal component.



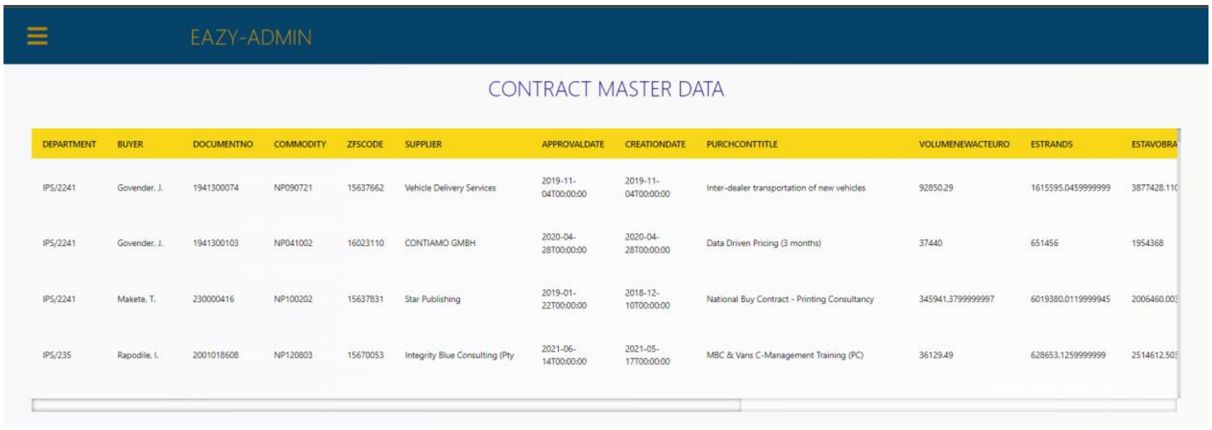
SUPPLIER MASTER DATA

DATAYEAR	DATAMONTH	ZFSCODE	SUPPLIER	BUYER	BBBEELEVEL	LEVELSCORE	BBBEECERTTO	GLOBUSYEAR	VALID	BBBEETYPE	VALUEADDINGSUPPLIER	BBBVOBEURO	NATION	DEPARTMENT
2019	8	15680954	Lubrication Equipment (Pty) Ltd	George, Jody	Level 4	1	2012-08-11T00:00:00	2012	0	QSE	no	0	ZA	IPS/2241
2019	8	15675296	Dell Computer (Pty) Ltd	Do Nascimento, Maria	Level 1	1.35	2016-04-28T00:00:00	2016	0	Generic	no	0	ZA	IPS/2241
2019	8	15637664	Blue Platinum Events	Makete, Tshepo	Level 2	1.25	2020-04-22T00:00:00	2020	1	QSE	no	1367534.2374999998	ZA	IPS/2241
2019	8	16012177	PGS-SA (Pty) Ltd	Mogale, Matshidiso							no		ZA	IPS/2241
2019	8	16022149	SNGCA South Africa (Pty) Ltd	Mogale, Matshidiso	Level 2	1.25	2019-09-27T00:00:00	2019	1	EME	no	31350.074999999997	ZA	IPS/2241

Eazy-Admin Sanele Manyela & Samkelwe Ngalimane - Word

3. Application Third Page – Contract

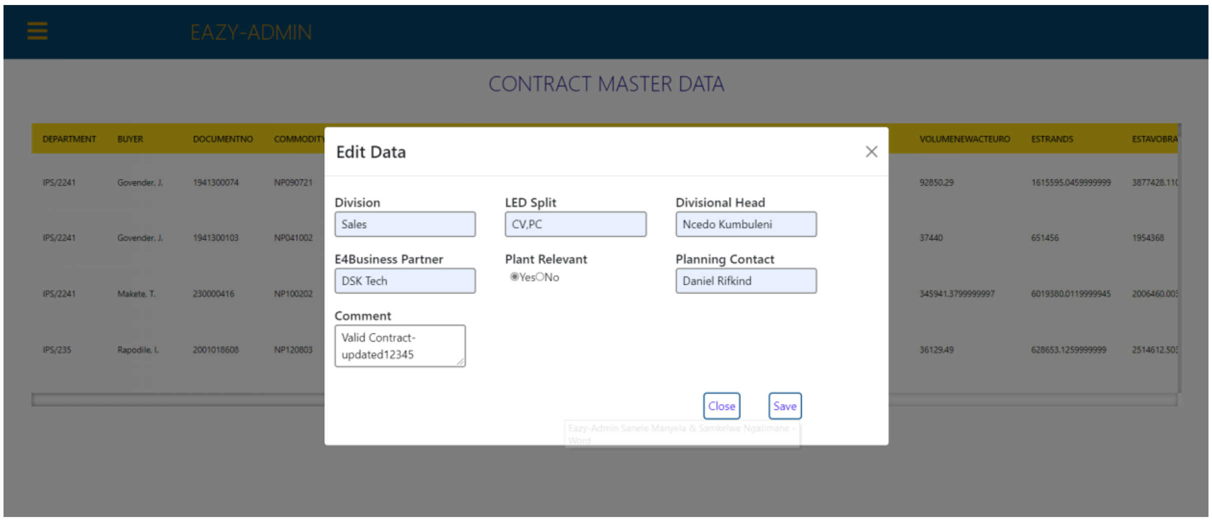
The Contract table displays Contract Master Data. The rows of the table support onClick events, when a row is clicked a modal to update data is brought up.



CONTRACT MASTER DATA

DEPARTMENT	BUYER	DOCUMENTNO	COMMODITY	ZFSCODE	SUPPLIER	APPROVALDATE	CREATIONDATE	PURCHCONTTITLE	VOLUMENEWACTEURO	ESTRANDS	ESTAVOBRA
IPS/2241	Govender, J.	1941300074	NP090721	15637662	Vehicle Delivery Services	2019-11-04T00:00:00	2019-11-04T00:00:00	Inter-dealer transportation of new vehicles	92850.29	1615595.0459999999	3877428.110
IPS/2241	Govender, J.	1941300103	NP041002	16023110	CONTIAMO GMBH	2020-04-28T00:00:00	2020-04-28T00:00:00	Data Driven Pricing (3 months)	37440	651456	1954368
IPS/2241	Makete, T.	230000416	NP100202	15637831	Star Publishing	2019-01-22T00:00:00	2018-12-10T00:00:00	National Buy Contract - Printing Consultancy	345941.3799999997	6019380.0119999945	2006460.003
IPS/235	Rapodile, I.	2001018608	NP120803	15670053	Integrity Blue Consulting (Pty	2021-06-14T00:00:00	2021-05-17T00:00:00	MBC & Vans C-Management Training (PC)	36129.49	628653.1259999999	2514612.503

The modal has two buttons: close – for hiding the modal, and a save button – to save updated data to the API



CONTRACT MASTER DATA

Edit Data

Division: Sales	LED Split: CV,PC	Divisional Head: Ncedo Kumbuleni
E4Business Partner: DSK Tech	Plant Relevant: [✓ Yes] No	Planning Contact: Daniel Rifkind
Comment: Valid Contract-updated12345		