3 Specific Requirements

3.1 Functional Requirements

3.1.1 User Interface

3.1.1.1 Main RIDE GUI - Josh

3.1.1.2 File Creation Wizard - Josh

3.1.1.3 Visualization (Rviz & Node Graph) - Justin

3.1.1.4 ROS Topic Monitor - Dom

3.1.1.5 ROS Topic Publisher - Dom

3.1.1.6 ROS Bag manager - Josh

3.1.2 Algorithms

3.1.2.1 Auto-Update CMakeLists.txt - Dan

3.1.2.2 Snippets - Gavin

3.2 Performance Requirements

3.2.1 Application Performance - Dan

3.3 Assumptions and Constraints

3.3.1 Internet Access - Justin

3.4 Non-Functional Requirements

3.4.1 Database - Dom

Ease of use:

3.4.2 Reliability - Justin

3.4.3 Security - Gavin

3.4.4 Maintainability - Dan

/ Example**

3.#>#>#>. Title (author name)

Description purpose and intent.

3.#.#.#.#. Paragraph. *Clause:*

1.

2/

3/

3.1.1.1. Main RIDE GUI. The system shall provide an icon button on VSCode's activity bar that opens the system's main GUI in VSCode's primary sidebar. *The following functional requirements shall be met:*

1. Nested webviews

2. expandable views

3. contractable views

4. Primary Sidebar with the following clickable options:

a. wizards

b. ROS bags

c. ROS topics

d. Visualization Tools

**/

3 Specific Requirements

3.1 Functional Requirements

Insert Text

3.1.1 User Interface

Insert Text

3.1.1.1. Main RIDE GUI (Joshua Peterson)

The system shall provide an icon button on VSCode's activity bar that opens the system's main GUI in VSCode's primary sidebar.

The following functional requirements shall be met:

1. RIDE's main GUI shall contain nested webviews
2. RIDE's nested webviews shall be expandable/contractible
3. RIDE's main GUI shall be within VSCode's primary sidebar containing the following clickable options:
    A. Wizards
    B. ROS Bags
    C. ROS Topics
    D. Visualization Tools

3.1.1.2 Wizard View (Joshua Peterson)

The system shall provide the ability to create ROS nodes, ROS messages, and ROS services containing code that is automatically generated via information gathered from the user. The creation wizard shall provide a “Create new” button with a drop down menu that has the following options:

- ROS Node
- ROS Msg
- ROS Srv

Upon making a selection from the list, the user shall be navigated to the creation wizard.

3.1.1.2.1 Creation Wizard

The creation wizard GUI will require the following input from the user:

- File Type (C++ or Python) - Dropdown selection
- Node Name (File Name) - String
- Node Location (File Path) - String
- Publisher (Checkbox) - Boolean
- Subscriber (Checkbox) - Boolean

The creation wizard GUI will also provide a cancel button and a submission button. Upon entering the required input and clicking the submission button, the creation wizard shall generate a file in the selected location with the selected name and file type. The contents of the file shall be generated automatically from snippets depending on the file type, publisher selection, and subscriber selection, respectively.

3.1.1.3 Visualization (Rviz & Node Graph) (Justin Tymkin)

The extension shall provide the user tools that visualize information from their current ROS workspace. The user will be able to select

- Rviz
- Node Graph

via a button on the sidebar.

3.1.1.3.1 Rviz

Rviz shall visualize the current ROS package the user is working on.

The following functional requirements shall be met:

1. Automatically read from a ROS package
2. Create a webview inside of VSCode
3. Display the ROS package inside the webview

- **. paragraph. Clause:**
  - **Navigates the users to a pop-up window**
  - **Provide the avility to Select a subset of the following global options**
  - **Provide the avility to Select one of the following global options:**
    - **a/**
  - **Select topic for visualization**
-
- **Node Graph**
  - **Select publisher to see what subscriber its communicating with**

- **What topics publisher and subscriber using**

3.1.1.4 ROS Topic Monitor (Dominik Soos)

The system shall provide access to a mechanism to subscribe to a specified ROS Topic.

The following functional requirements shall be met:

1. The system shall allow subscribers to choose the ROS Topic they would like to monitor, and the current values of the topic's message shall also be displayed.
2. The system shall provide functionality to listen to messages through specific subscribers.
3. The system shall provide the option to choose the message format such as raw, or any custom format.

3.1.1.5 ROS Topic Publisher (Dominik Soos)

The following functional requirements shall be met:

1. The system shall provide an intuitive interface to publish messages to ROS Topics.
2. The system should have a user interface that makes it seamless to enter message data and it should support multiple messages types.
3. The system shall also allow the specification of the frequency at which messages are being published. A higher update rate on the right will imply that the monitor will retrieve the values more frequently when the left side of the setting will retrieve and publish with less frequency.

3.1.1.6 ROS Bag manager (Joshua Peterson)

- Start recording
- Manage bags

- Select Bag
  - Play bag
  - Clone bag
    - Name, Location, Trim, Filter topics

3.1.2 Algorithms

Insert Paragraph

3.1.2.1 Update CMakeLists.txt (Dan)

- The system shall update the CMakeLists.txt file when a ROS node, srv, or msg file is updated

3.1.2.2 Snippets (Gavin St. Clair)

The system shall provide the ability to use snippets by typing the description name and selecting the code snippet.

3.2 Performance Requirements

Insert Paragraph

3.2.1 Application Performance (Dan)

- The application shall update all files within one second of a request being made.
- The application shall update visualization tools within one second of the request being made

3.3 Assumptions and Constraints

Insert Paragraph

3.3.1 Internet Access (Justin Tymkin)

- Assumed user has internet access to install extension

3.4 Non-Functional Requirements

Insert Paragraph

3.4.1 Database

The system shall provide a Usage Analytics database through which the developers can analyze user behavior and improve based on those.

3.4.2 Reliability (Justin Tymkin)

- Reliability within VSCode and keeping extension updated

3.4.3 Security (Gavin St. Clair)

A VS Code extension is secure because a third party attacker has no easy way to modify an existing one without compromising the real developer.

3.4.4 Maintainability (Dan)

-