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 Update CMakeLists.txt - Dan

3.1.2.2 Snippets - Gavin

3.1.2.3 ROS Topic Publisher - 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

3.4.2 Reliability - Justin

3.4.3 Security - Gavin

3.4.4 Maintainability - Dan

3 Specific Requirements

3.1 Functional Requirements

3.1.1 User Interface

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 system's main GUI shall consist of the following webviews nested within expandable and contractible views in VSCode’s primary sidebar:

- Wizards
- ROS Bags
- ROS Topics
- Visualization Tools

3.1.1.2 Wizard View

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)

- **Rviz**
  - **Select global options**
  - **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

The system shall provide access to a mechanism to subscribe to a specified ROS Topic. It should 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. The system shall also provide the option to choose the message format such as raw, or any custom format. The update rate of the ROS Topic

being monitored shall also be configurable. This update rate refers to the number at which the monitor retrieves and displays the latest values of the ROS Topic's message.

3.1.1.5 ROS Topic Publisher

The system shall provide an intuitive interface to publish messages to ROS Topics. The system should have a user interface that makes it seamless to enter message data and it should support multiple messages types. The system shall also allow the specification of the frequency at which messages are being published. The system may provide a range bar at which the frequency can be specified. 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

- Start recording
- Manage bags
    - Select Bag
        - Play bag
        - Clone bag
            - Name, Location, Trim, Filter topics

3.1.2 Algorithms

3.1.2.1 Update CMakeLists.txt

3.1.2.2 Snippets

The system shall provide the ability to use snippets by typing def and clicking from the drop down list.

3.1.2.3 ROS Topic Publisher

3.2 Performance Requirements

3.2.1 Application Performance

3.3 Assumptions and Constraints

3.3.1 Internet Access

- Assumed user has internet access to install extension

3.4 Non-Functional Requirements

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

- Reliability within VSCode and keeping extension updated

3.4.3 Security

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