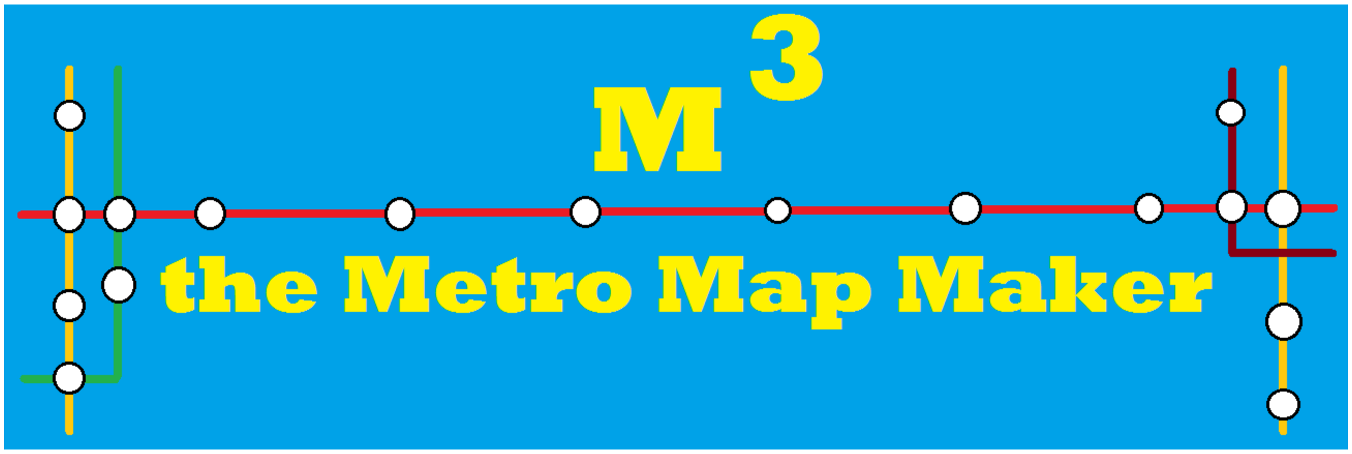
The *Metro Map Maker* TM0
Software Requirements Specification



Author: Richard McKenna
Debugging Enterprises^{TM}

Based on IEEE Std 830TM-1998 (R2009) document format

Copyright © 2017 Debugging Enterprises

No part of this publication may be reproduced in any form, in an electronic retrieval system or otherwise, without the prior written permission of the publisher.

1 Introduction

Finding your way around a new city can be challenging so many people look to the Internet for help. Cities with subway systems typically provide maps to help one navigate from one stop to another across a number of intersecting lines. These maps let one chart which lines to take and how many stops it will be before one arrives and the user can typically choose between multiple routes.

The ***Metro Map Maker (i.e. M^3)*** application will provide the user with a set of tools to build graphical representations of city subway systems with named lines and named stops and intersecting lines and landmarks. It will also provide a means for calculating the best route to take to journey from one particular station to another. Finally, it will provide an export feature such that it may export a generated map and associated metro system information to a format that can be used by a corresponding Web application that will be able to make use of it.

1.1 **Purpose**

The purpose of this document is to specify how our ***Metro Map Maker*** program should look and operate. The intended audience for this document is all the members of the development team, those who will design the maps for use with the Web application, and the potential users of such an application. This document serves as an agreement among all parties and as a reference for how the map creation tool should ultimately be constructed. Upon completing the reading of this document, one should clearly visualize how the application will look and operate.

1.2 Scope

For this project the goal is for users to easily make and edit subway maps. There will be an emphasis on ease of use. Note that there will be a common export format that will be provided for exported subway system data such that all maps can be used by a uniform application.

1.3 **Definitions, acronyms, and abbreviations**

Framework – In an object-oriented language, a collection of classes and interfaces that collectively provide a service for building applications or additional frameworks all with a common need.

GUI – Graphical User Interface, visual controls like buttons inside a window in a software application that collectively allow the user to operate the program.

IEEE – Institute of Electrical and Electronics Engineers, the "world's largest professional association for the advancement of technology".

JavaScript – the default scripting language of the Web, JavaScript is provided to pages in the form of text files with code that can be loaded and executed when a page loads so as to dynamically generate page content in the DOM.

Stylesheet – a static text file employed by HTML pages that can control the colors, fonts, layout and other style components in a Web page.

UML – Unified Modeling Language, a standard set of document formats for designing software graphically.

Use Case Descriptions – A formal format for specifying how a user will interact with a system.

1.4 References

IEEE Std 830^{TM}-1998 (R2009) – IEEE Recommended Practice for Software Requirements Specification

1.5 Overview

This SRS will clearly define how the ***Metro Map Maker*** application should look and operate. Note that this is not a software design description (SDD), which would design how to construct the software using UML. This document does not specify how to build the appropriate technologies, it is simply an agreement concerning what to build. Section 2 of this document will provide the context for the project and specify all the conceptual design. Section 3 will present how the user interface should be laid out. Section 4 provides a Table of Contents, an Index, and References.

2 Overall description

The ***Metro Map Maker*** hopes to be a site where one can go to make and exchange real world subway maps for metro systems in cities around the world and then plot routes through these systems. Why use a site like this? Well, because one may be interested in making their own map of a given metro system that looks completely different from existing standard maps. The site itself is under construction and it will provide the service of letting one choose an existing map created by a user and so let the entire community view the map and plot routes. What the site is being created the hope is to develop a desktop Java application that can be used for creating the maps to be used by the site. Maps means map images with corresponding JSON files describing the map contents.

2.1 Product perspective

Our map making application will let the user make subway maps complete with all station and metro line data such that they can be drawn in great detail and routes can be plotted. Note that the Web site's details are not described in this document as that is being constructed by a separate development team.

2.1.1 System Interfaces

The ***Metro Map Maker*** will be a traditional workspace-type application that will let the user create a new metro map and export it as an image with corresponding JSON file, which the site will be able to load and use to do things like pathing calculations as well as highlighting of map content. Note that the JSON format will have to list complete information about all Metro Lines and Stops such that a Web application could use the data to do its own pathfinding. In addition, the precise pixel perfect (x, y) location of each stop circle in the generated image must be provided such that the Web application can do additional highlighting of content on top of the map image.

2.1.2 User Interfaces

Our program will be a desktop application and so will make use of a mouse and keyboard for user input. Figure 2.4 below summarizes the ways with which the user will interact with our ***Metro Map Maker*** application, which will be further detailed using formal UML Use Case descriptions. These Use-Case descriptions should be fed as input directly into Section 3.1, external interfaces, which is where the design of the user interface is specified. Here is the full list of UML Use-Case Descriptions:

Use Case	UI Context	Use Case
2.1	Welcome Dialog	Create New Map
2.2	Welcome Dialog	Select Recent Map to Load
2.3	Welcome Dialog	Close Welcome Dialog
2.4	File Toolbar	Create New Map
2.5	File Toolbar	Load Map
2.6	File Toolbar	Save Map

2.7	File Toolbar	Save As Map
2.8	File Toolbar	Export Map
2.9	Undo/Redo Toolbar	Undo Edit
2.10	Undo/Redo Toolbar	Redo Edit
2.11	About Toolbar	Learn About Application
2.12	Metro Lines Toolbar	Add Line
2.13	Metro Lines Toolbar	Remove Line
2.14	Metro Lines Toolbar	Edit Line
2.15	Metro Lines Toolbar	Move Line End
2.16	Metro Lines Toolbar	Add Stations to Line
2.17	Metro Lines Toolbar	Remove Stations from Line
2.18	Metro Lines Toolbar	List All Stations in Line
2.19	Metro Lines Toolbar	Change Line Thickness
2.20	Metro Stations Toolbar	Add New Station
2.21	Metro Stations Toolbar	Remove Station
2.22	Metro Stations Toolbar	Snap to Grid
2.23	Metro Stations Toolbar	Move Station Label
2.24	Metro Stations Toolbar	Rotate Station Label
2.25	Metro Stations Toolbar	Change Station Fill Color
2.26	Metro Stations Toolbar	Change Station Circle Radius
2.27	Station Router Toolbar	Find Route
2.28	Décor Toolbar	Set Background Color
2.29	Décor Toolbar	Set Image Background
2.20	Décor Toolbar	Add Image Overlay
2.31	Décor Toolbar	Add Label
2.32	Décor Toolbar	Move Map Element
2.33	Décor Toolbar	Remove Map Element
2.34	Font Toolbar	Change Text Color
2.35	Font Toolbar	Change Text Font
2.36	Navigation Toolbar	Zoom In and Out
2.37	Navigation Toolbar	Increase and Decrease Map Size
2.38	Navigation Toolbar	Navigate Map Up, Down, Left, and Right
2.38	Navigation Toolbar	Toggle Snap Grid
2.39	Title Bar	Exit Application

Figure 2.4: Overview of Use-Case Descriptions

Use Case 2.1: Create New Map

Use-Case:	Create New Map
Primary Actor:	Map Editor
Goal in Context:	The user wishes to make a new metro map
Preconditions:	The application has been started and the user is viewing the Welcome Dialog
Scenario:	1. User is viewing the Metro Map Maker application's Welcome Dialog 2. User clicks on the Create New Metro Map link/button. A text dialog prompts the user for the name of the map to create. 3. User enters a unique map name. The app will then create a new file for that map as well as a directory with that name in the application's export directory.
Exceptions:	Should the user enter a map name that already exists the user should be notified via a warning dialog and prompted for a name again. Note that the name prompt dialog should also have a Cancel option.
Priority:	Essential, must be implemented
When available:	First Benchmark
Frequency of use:	Can only at maximum be used once per session
Open Issues:	Size, location, and style of UI should be finalized by UI designer

Use Case 2.2: Select Recent Map to Load

Use-Case:	Select Recent Map to Load
Primary Actor:	Map Editor
Goal in Context:	The user wishes to load a recently edited map that has already been created
Preconditions:	The application has been started and the user is viewing the Welcome Dialog
Scenario:	1. User is viewing the Metro Map Maker application's Welcome Dialog 2. User clicks on one of the links to a recently edited map. The Welcome Dialog is closed. The Main UI is opened and data is initialized and the user interface is setup with values loaded from the selected map.
Exceptions:	Should any error occur during the loading of an existing map the user should be notified via a warning dialog
Priority:	Essential, must be implemented
When available:	First Benchmark
Frequency of use:	Can only at maximum be used once per session
Open Issues:	Size, location, and style of UI should be finalized by UI designer

Use Case 2.3: Close Welcome Dialog

Use-Case:	Close Welcome Dialog
Primary Actor:	Map Editor
Goal in Context:	The user wishes to close the Welcome Dialog in order to bring forth the Main UI
Preconditions:	The application has been started and the user is viewing the Welcome Dialog
Scenario:	1. User is viewing the Metro Map Maker application's Welcome Dialog 2. User clicks on the dialog's X in the window's title bar. Welcome Dialog is closed. Main UI is opened without the workspace being loaded
Exceptions:	This is only available at the start of using the application as once the Welcome Dialog is closed it is never opened again
Priority:	Essential, must be implemented
When available:	First Benchmark

Frequency of use:	Can only at maximum be used once per session
Open Issues:	Size, location, and style of UI should be finalized by UI designer

Use Case 2.4: Create New Map

Use-Case:	Create New Map
Primary Actor:	Map Editor
Goal in Context:	The user wishes to make a new code check
Preconditions:	The application has been started and the Main UI has been loaded
Scenario:	1. User is viewing the Metro Map Maker application's Main UI 2. User clicks on the Create New Map button. A text dialog prompts the user for the name of the map to create. 3. User enters a unique map name. The app will then create a new file for that map as well as a directory with that name in the application's export directory.
Exceptions:	Should the user enter a map name that already exists the user should be notified via a warning dialog and prompted for a name again. Note that the name prompt dialog should also have a Cancel option.
Priority:	Essential, must be implemented
When available:	First Benchmark
Frequency of use:	Many times per session
Open Issues:	Size, location, and style of UI should be finalized by UI designer

Use Case 2.5: Load Map

Use-Case:	Load Map
Primary Actor:	Map Editor
Goal in Context:	The user wishes to load an existing map that has already been created
Preconditions:	The map to load must already exist
Scenario:	1. User is viewing the Map application's Main UI 2. User clicks on the Load Map button. The user will then be prompted via a File Chooser dialog to select a Map file. 3. The user selects an existing Map file and presses the File Chooser dialog's Open button. The File Chooser dialog is closed. The file's data is initialized and the user interface is setup with values loaded from the selected map
Exceptions:	Should any error occur during the loading of an existing map the user should be notified via a warning dialog
Priority:	Essential, must be implemented
When available:	Second Benchmark
Frequency of use:	Many times per session
Open Issues:	Size, location, and style of UI should be finalized by UI designer

Use Case 2.6: Save Map

Use-Case:	Save Map
Primary Actor:	Map Editor
Goal in Context:	The user wants to save the map being edited
Preconditions:	The application is running and a map is in the process of being edited
Scenario:	1. User is working on a map in any step of the editing process

	2. User clicks on Save button. Map will be saved with latest edits. Save button will then be disabled
Exceptions:	N/A
Priority:	Essential, must be implemented
When available:	Second Benchmark
Frequency of use:	Many times per session
Open Issues:	Size, location, and style of UI should be finalized by UI designer

Use Case 2.7: Save As Map

Use-Case:	Save As Map
Primary Actor:	Map Editor
Goal in Context:	The user wants to save the map currently being edited under another name
Preconditions:	The application is running and the Main UI is loaded and a map is in the process of being edited.
Scenario:	1. User is working on a map in any step of the editing process 2. User clicks on Save As button. 3. A dialog prompts the user for the new map name. The user must then enter a unique name for the new map name. 4. If a unique map name is provided the map is saved as the new name and a directory is created with that name in the export directory. Save button will then be disabled
Exceptions:	Should the user enter a map name that already exists the user should be notified via a warning dialog and prompted for a name again. Note that the name prompt dialog should also have a Cancel option.
Priority:	Essential, must be implemented
When available:	Second Benchmark
Frequency of use:	Many times per session
Open Issues:	Size, location, and style of UI should be finalized by UI designer

Use Case 2.8: Export Map

Use-Case:	Export Map
Primary Actor:	Map Editor
Goal in Context:	The user wishes to export the map for use with a Web site
Preconditions:	The application is running and the Main UI is loaded.
Scenario:	1. User clicks the Export Map button. The application then exports the map image and map data to the proper JSON format. Once this process is complete 2. The user clicks Ok button to close the verification dialog.
Exceptions:	N/A
Priority:	Essential, must be implemented
When available:	First Benchmark
Frequency of use:	Many times per session
Open Issues:	Size, location, and style of UI should be finalized by UI designer

Use Case 2.9: Undo Edit

Use-Case:	Undo Edit
Primary Actor:	Map Editor

Goal in Context:	The user wishes to undo the most recent edit
Preconditions:	The user has made at least one change to the file being edited since it was either originally created (if created during this session) or loaded.
Scenario:	1. User makes edits to the current map 2. User presses Undo button
Exceptions:	Note that there will occasions where the Transaction stack is empty either because the map has not been edited since being loaded or because all edits have already been undone. In such a case the Undo button should be disabled.
Priority:	Essential, must be implemented
When available:	Third Benchmark
Frequency of use:	Many times per session
Open Issues:	Size, location, and style of UI should be finalized by UI designer

Use Case 2.10: Redo Edit

Use-Case:	Redo Edit
Primary Actor:	Map Editor
Goal in Context:	The user wishes to redo the most recently undone edit
Preconditions:	The user has made at least one change to the file being edited since it was either originally created (if created during this session) or loaded and at least one edit has been undone and is available to be redone.
Scenario:	1. User makes edits to the current map 2. User presses Undo button 3. User presses Redo button
Exceptions:	Note that there will occasions where the Transaction stack is empty or the Undo/Redo stack is at the top of the stack with nothing to Redo. In either cases the Redo button should be disabled.
Priority:	Essential, must be implemented
When available:	Third Benchmark
Frequency of use:	Many times per session
Open Issues:	Size, location, and style of UI should be finalized by UI designer

Use Case 2.11: Learn About Application

Use-Case:	Learn About Application
Primary Actor:	Map Editor
Goal in Context:	User wishes to learn a little about the application and its creator
Preconditions:	User has started the application and is viewing the main UI
Scenario:	1. User is viewing the main UI 2. User presses the About button in the top toolbar 3. A dialog opens listing the name of application, the frameworks used, the names of the developers at Debugging Enterprises and year of development
Exceptions:	N/A
Priority:	Essential, must be implemented
When available:	First Benchmark
Frequency of use:	Many times per session
Open Issues:	Size, location, and style of UI should be finalized by UI designer

Use Case 2.12: Add Line

Use-Case:	Add Line
Primary Actor:	Map Editor
Goal in Context:	User wishes to add a subway line to the metro map
Preconditions:	User is in the process of editing a map
Scenario:	1. User is editing a metro map in the workspace 2. User presses Add Line button in the Metro Lines toolbar. This should open a dialog box that prompts the user for the name and color of the line. 3. User enters the name and selects the line color and presses Ok (cancel to close). A line with the name of the line on two ends is then added to the map without any stops. The line will be in the prescribed color. The line is then selected in the Metro Lines combo box and its color is loaded as the background color in the Line Edit button.
Exceptions:	Should the user cancel the add line process in the dialog stage it will not be added.
Priority:	Essential, must be implemented
When available:	Second Benchmark
Frequency of use:	Many times per session
Open Issues:	Size, location, and style of UI should be finalized by UI designer

Use Case 2.13: Remove Line

Use-Case:	Remove Line
Primary Actor:	Map Editor
Goal in Context:	User wishes to remove a subway line from the metro map
Preconditions:	User is in the process of editing a map with at least one line on it
Scenario:	1. User is editing a metro map in the workspace 2. User selects the line to remove in the Metro Lines toolbar’s combo box. 3. User presses the Remove Line button. This is a significant edit so it should verify this action with an Ok/Cancel dialog box. 4. User presses Ok to remove the line (Cancel to cancel). Note that when the line is removed the stops should NOT be removed. Afterwards another Line should be selected and loaded into the combo box (if one exists).
Exceptions:	N/A
Priority:	Essential, must be implemented
When available:	Second Benchmark
Frequency of use:	Many times per session
Open Issues:	Size, location, and style of UI should be finalized by UI designer

Use Case 2.14: Edit Line

Use-Case:	Edit Line
Primary Actor:	Map Editor
Goal in Context:	User wishes to change either the Line name or color
Preconditions:	User is in the process of editing a map with at least one line on it
Scenario:	1. User is editing a metro map in the workspace 2. User selects the line to edit in the Metro Lines toolbar’s combo box 3. User presses the Edit Line button. This will open the Edit Line dialog, which has a text field loaded with the current line name and a color picker loaded with the current line color. 4. User enters any line name changes desired in the text field

	5. User selects color in the color picker 6. User presses Ok to complete changes. Line Edit dialog is closed and the line name change is then reflected in the Metro Lines' combo box and the color in the background of the Edit Line button.
Exceptions:	N/A. Note that we will allow the user to create the maps they like, which may include two lines with the same names or colors.
Priority:	Essential, must be implemented
When available:	Second Benchmark
Frequency of use:	Many times per session
Open Issues:	Size, location, and style of UI should be finalized by UI designer

Use Case 2.15: Move Line End

Use-Case:	Move Line End
Primary Actor:	Map Editor
Goal in Context:	User wishes to move one of the two ends of a Metro Line. Note that every metro line will always have 2 ends that extend beyond the last stop on both ends and the display of the line ends with a textual label with the name of the line. But note that it is useful to use a precise point much like a stop to denote the location of this text. So, during the line end positioning process we will use a point for locating it rather than just text.
Preconditions:	User is editing a map with at least one line.
Scenario:	1. User is editing a metro map in the workspace. 2. User mouse presses on a metro line end text in the map (like the text "Suin" for the Suin Line). This should turn the text at the end into a bright circle the same size as the stops circles, but a brighter, highlighted color. This would denote that it can be moved. 3. User drags the metro line end circle to desired destination. 4. User unselects the metro line end by clicking anywhere else on the map, meaning another component or on nothing at all.
Exceptions:	N/A
Priority:	Essential, must be implemented
When available:	Second Benchmark
Frequency of use:	Many times per session
Open Issues:	Size, location, and style of UI should be finalized by UI designer

Use Case 2.16: Add Stations to Line

Use-Case:	Add Stations to Line
Primary Actor:	Map Editor
Goal in Context:	User wishes to add a station to a subway line
Preconditions:	User is editing a map with at least one line and one station
Scenario:	1. User is editing a metro map in the workspace 2. User selects a subway line in the combo box in the Metro Lines toolbar 3. User presses the Add Station button. This should then put the application in "Add Station Mode", meaning the next click on the workspace will be for adding a station to a line. Note that the cursor should change to denote that a different mode is in use. 4. User clicks on an existing station in the map. If the station is not already in the line it will be added. Note that the order of the station (i.e. where on the line, between which stations) should be automatically determined by what other line stations it is

	closest to. If it is closest or second closest to the end of a line then it would automatically become the last station in the line. Note that the order of line stations must be automatically resorted after each station is added or after stations or ends of lines are moved. 5. User may continue to add multiple stations via one click after another on stations without having the re-click the Add Station button until the user clicks elsewhere on the map (i.e. not on a station to add).
Exceptions:	Should the user enter add station mode and then click on the map elsewhere (i.e. not on a station), then the app should leave this mode and change the cursor back.
Priority:	Essential, must be implemented
When available:	Second Benchmark
Frequency of use:	Many times per session
Open Issues:	Size, location, and style of UI should be finalized by UI designer

Use Case 2.17: Remove Stations from Line

Use-Case:	Remove Stations from Line
Primary Actor:	Map Editor
Goal in Context:	User wishes to remove a station from a subway line
Preconditions:	User is viewing a map with at least one line with a station on it
Scenario:	1. User is editing a metro map in the workspace 2. User selects a subway line in the combo box in the Metro Lines toolbar 3. User presses the Remove Station button. This should then put the application in "Remove Station Mode", meaning the next click on the workspace will be for removing a station from the line. Note that the cursor should change to denote that a different mode is in use. 4. User clicks on a station currently on the selected line. The line should then be removed from the line (but not removed from the map) and the user may then click on additional stations to remove (without having to re-click on the Remove Station button). This remove station mode is ended when the user clicks elsewhere on the map (i.e. not on a station to add).
Exceptions:	N/A
Priority:	Essential, must be implemented
When available:	Second Benchmark
Frequency of use:	Many times per session
Open Issues:	Size, location, and style of UI should be finalized by UI designer

Use Case 2.18: List All Stations in Line

Use-Case:	List All Stations in Line
Primary Actor:	Map Editor
Goal in Context:	User wishes to view a definitive list of all stations in the line in an itemized fashion
Preconditions:	User is viewing a map with at least one line
Scenario:	1. User is editing a metro map in the workspace 2. User selects a subway line in the combo box in the Metro Lines toolbar 3. User clicks the List Stations button in the Metro Lines toolbar. This opens the Line Stations Dialog window which itemizes the stops carefully in proper order from top to bottom. The motivation here is that the map may look deceiving depending on

	how it is drawn and this list definitively lists which stops are in a given line. 4. User presses Ok to close dialog.
Exceptions:	N/A
Priority:	Essential, must be implemented
When available:	Second Benchmark
Frequency of use:	Many times per session
Open Issues:	Size, location, and style of UI should be finalized by UI designer

Use Case 2.19: Change Line Thickness

Use-Case:	Change Line Thickness
Primary Actor:	Map Editor
Goal in Context:	User wishes to change the line thickness for a particular line
Preconditions:	User is viewing a map with at least one line
Scenario:	1. User is editing a metro map in the workspace 2. User selects a subway line in the combo box in the Metro Lines toolbar 3. User drags the line thickness slider to the desired setting
Exceptions:	N/A
Priority:	Essential, must be implemented
When available:	Second Benchmark
Frequency of use:	Many times per session.
Open Issues:	Size, location, and style of UI should be finalized by UI designer

Use Case 2.20: Add New Station

Use-Case:	Add New Station
Primary Actor:	Map Editor
Goal in Context:	User wishes to add a new station to the map
Preconditions:	User is viewing a map
Scenario:	1. User is editing a metro map in the workspace 2. User clicks on the New Station button in the Metro Stations toolbar. A dialog is then opened prompting the user to enter a name for the station. 3. User enters the name of the station and clicks Ok.
Exceptions:	Should the user click Cancel in the New Station dialog then no station will be added.
Priority:	Essential, must be implemented
When available:	Second Benchmark
Frequency of use:	Many times per session
Open Issues:	Size, location, and style of UI should be finalized by UI designer

Use Case 2.21: Remove Station

Use-Case:	Remove Station
Primary Actor:	Map Editor
Goal in Context:	User wishes to remove a station from the map altogether
Preconditions:	User is viewing a map with at least one stop
Scenario:	1. User is editing a metro map in the workspace

	2. User selects a station either by choosing it in the Metro Station toolbar's combo box or by clicking on it in the map. Note that clicking on it in the map should load it into the combo box. 3. User clicks on Remove Station button. The user should then be prompted to verify this action. 4. User clicks ok and the station should be removed from the map. This means it must also be removed from all lines it is associated with.
Exceptions:	Should the user press Cancel in the verification dialog then the station will not be removed.
Priority:	Essential, must be implemented
When available:	Second Benchmark
Frequency of use:	Many times per session
Open Issues:	Size, location, and style of UI should be finalized by UI designer

Use Case 2.22: Snap to Grid

Use-Case:	Snap Map Element
Primary Actor:	Map Editor
Goal in Context:	User wishes to snap the selected map element (station or line end) to a grid point. This helps with aligning things more neatly.
Preconditions:	User is viewing a map with at least one line or station
Scenario:	1. User is editing a metro map in the workspace 2. User selects either a line end or a station 3. User clicks the Snap to Grid button, which will move the center of either the station circle or the line end circle to the nearest grid point.
Exceptions:	N/A
Priority:	Essential, must be implemented
When available:	Third Benchmark
Frequency of use:	Many times per session
Open Issues:	Size, location, and style of UI should be finalized by UI designer

Use Case 2.23: Move Station Label

Use-Case:	Move Station Label
Primary Actor:	Map Editor
Goal in Context:	User wishes to move a station's textual label on the map to one of the four prescribed locations, either top left, top right, bottom left, or bottom right. That makes this a toggle button of sorts with four setttings.
Preconditions:	User is viewing a map with at least one station
Scenario:	1. User is editing a metro map in the workspace 2. User selects a station either in the combo box or by clicking on it in the map, which should load it into the station combo box. 3. User presses the Move Station Label button repeatedly to view the station label for that station in each of the 4 prescribed locations.
Exceptions:	N/A
Priority:	Essential, must be implemented
When available:	Second Benchmark
Frequency of use:	Many times per session
Open Issues:	Size, location, and style of UI should be finalized by UI designer

Use Case 2.24: Rotate Station Label

Use-Case:	Rotate Station Label
Primary Actor:	Map Editor
Goal in Context:	User wishes to rotate the station's textual label on the map to either 0 degrees or 90 degress, which makes this a toggle button with 2 settings. Note that depending on whether the line is displayed horizontally or vertically it can be convenient in rendering the map to display some labels one way and some the other.
Preconditions:	User is viewing a map with at least one station
Scenario:	1. User is editing a metro map in the workspace 2. User selects a station either in the combo box or by clicking on it in the map, which should load it into the station combo box. 3. User presses the Rotate Station button to toggle it and view the station label rotated either to 90 degrees or not at all (0 degrees).
Exceptions:	N/A
Priority:	Essential, must be implemented
When available:	Second Benchmark
Frequency of use:	Many times per session
Open Issues:	Size, location, and style of UI should be finalized by UI designer

Use Case 2.25: Change Station Fill Color

Use-Case:	Change Station Fill Color
Primary Actor:	Map Editor
Goal in Context:	User wishes to change the fill color of a given station's circle
Preconditions:	User is viewing a map with at least one station
Scenario:	1. User is editing a metro map in the workspace 2. User selects a station either in the combo box or by clicking on it in the map, which should load it into the station combo box. 3. User presses the Change Station Fill Color button and then selects the desired color from the color picker. Note that the color should then immediately be reflected in the circle in the map as well as in the color picker itself.
Exceptions:	N/A
Priority:	Essential, must be implemented
When available:	Second Benchmark
Frequency of use:	Many times per session
Open Issues:	Size, location, and style of UI should be finalized by UI designer

Use Case 2.26: Change Station Circle Radius

Use-Case:	Change Station Circle Radius
Primary Actor:	Map Editor
Goal in Context:	User wishes to change the circle radius of a given station's circle
Preconditions:	User is viewing a map with at least one station
Scenario:	1. User is editing a metro map in the workspace 2. User selects a station either in the combo box or by clicking on it in the map, which should load it into the station combo box. 3. User drags the Change Station Circle Radius slider, which would increase or decrease the station circle radius and immediately reflect it in the map.

Exceptions:	N/A
Priority:	Essential, must be implemented
When available:	Second Benchmark
Frequency of use:	Many times per session
Open Issues:	Size, location, and style of UI should be finalized by UI designer

Use Case 2.27: Find Route

Use-Case:	Find Route
Primary Actor:	Map Editor
Goal in Context:	User wishes to verify that given the current map a route can be found between two stations. Note that we'll estimate trips between stops to take on average 3 minutes.
Preconditions:	User is viewing a map with at least one stop
Scenario:	1. User is editing a metro map in the workspace 2. User selects two existing stations from any lines in the two combo boxes in the Find Routes toolbar 3. User clicks Find Route button. This will then find a route from one station to the other and open a Route Dialog that will display an itemized list of trains and transfers needed in order to go from the first station to the second. Note that the route should be the legal one with the minimal number of stops between the two. 4. User should click Close button to close dialog
Exceptions:	Should there be no route from one station to the other (i.e. their lines never cross) then a message should be shown in the dialog stating as such.
Priority:	Essential, must be implemented
When available:	Third Dialog Benchmark
Frequency of use:	Many times per session
Open Issues:	Size, location, and style of UI should be finalized by UI designer

Use Case 2.28: Set Background Color

Use-Case:	Set Background Color
Primary Actor:	Map Editor
Goal in Context:	User wishes to choose the map background color, which will be the color behind all map elements.
Preconditions:	User is viewing a map
Scenario:	1. User is editing a metro map in the workspace 2. User clicks the Set Background Color button which will open the Color Picker. 3. User selects the desired color. This will immediately change the background color of the map.
Exceptions:	Note that the background color would be behind the background image, so it is possible that sometimes the background color is not visible, like in the case where the background image is fully opaque and is at least as large as the size of the map.
Priority:	Essential, must be implemented
When available:	Third Benchmark
Frequency of use:	Many times per session
Open Issues:	Size, location, and style of UI should be finalized by UI designer. UI designer should consider making this a Radial Gradient such that background colors look more appealing.

Use Case 2.29: Set Image Background

Use-Case:	Set Image Background
Primary Actor:	Map Editor
Goal in Context:	User wishes to set a background image to use as the background for the map or to get rid of one altogether
Preconditions:	User is viewing a map
Scenario:	1. User is editing a metro map in the workspace 2. User clicks the Set Image Background button. This will open a dialog and prompt the user as to whether they want a dialog (Yes/No) 3. If the user clicks No, the dialog will close and no background image will be used 4. If the user clicks yes, a file chooser dialog will open where the user can navigate to an image to use. 5. The user selects the image to use as the background image and clicks ok. The map is then drawn on top of the image, i.e. the background image is under everything else. Note that the background image should be centered in the workspace. Note that the background image can have transparency so the background gradient can show through and around the background image (if it's smaller than the map). If the image is larger than the map it should still be centered behind the map.
Exceptions:	Note that there can only be at most one background image. If no image is selected then the background color gradient provides the background.
Priority:	Essential, must be implemented
When available:	Third Benchmark
Frequency of use:	Many times per session
Open Issues:	Size, location, and style of UI should be finalized by UI designer

Use Case 2.30: Add Image Overlay

Use-Case:	Add Image Overlay
Primary Actor:	Map Editor
Goal in Context:	User wishes to add an image overlay on top of the map, which can be useful for adding landmarks and other notations (like a legend) to the map.
Preconditions:	User is viewing a map
Scenario:	1. User is editing a metro map in the workspace 2. User clicks the Add Image Overlay button. This will open a file choosing dialog where the user can navigate to an image and select it. 3. User selects an image and clicks Ok, which will add it to the map.
Exceptions:	Should the user hit cancel when viewing the select image dialog no image is added.
Priority:	Essential, must be implemented
When available:	Third Benchmark
Frequency of use:	Many times per session
Open Issues:	Size, location, and style of UI should be finalized by UI designer

Use Case 2.31: Add Label

Use-Case:	Add Label
Primary Actor:	Map Editor
Goal in Context:	User wishes to add a textual label to the map for the purpose of a map title or note of some sort
Preconditions:	User is viewing a map
Scenario:	1. User is editing a metro map in the workspace 2. User clicks the Add Label button and a dialog prompts the user for text to enter for the label. 3. User enters text in the dialog and clicks Ok, which then adds the label to a default location in the map.
Exceptions:	Should the user click Cancel in the dialog nothing will be added to the map
Priority:	Essential, must be implemented
When available:	Third Benchmark
Frequency of use:	Many times per session
Open Issues:	Size, location, and style of UI should be finalized by UI designer

Use Case 2.32: Remove Map Element

Use-Case:	Remove Label/Image Overlay
Primary Actor:	Map Editor
Goal in Context:	User wishes to remove a map element like a label or image overlay
Preconditions:	User is viewing a map
Scenario:	1. User is editing a metro map in the workspace 2. User clicks on a map element like an overlay image or text label 3. User clicks on Remove Map Element button, which will cause the element to be removed from the map.
Exceptions:	N/A
Priority:	Essential, must be implemented
When available:	Third Benchmark
Frequency of use:	Many times per session
Open Issues:	Size, location, and style of UI should be finalized by UI designer

Use Case 2.33 Move Map Element

Use-Case:	Move Map Element
Primary Actor:	Map Editor
Goal in Context:	User wishes to move a map element like a station, line end, label or image overlay. Note that this is similar in function to Use Case 2.15 which described some of the issues within selecting, displaying, and moving Line Ends.
Preconditions:	User is viewing a map and there is at least one map element on the map
Scenario:	1. User is editing a metro map in the workspace 2. User clicks on a map element like a line end, station, image overlay, or label. This will highlight the map element 3. User drags the map element to the desired location
Exceptions:	Map element cannot be dragged off of map

Priority:	Essential, must be implemented
When available:	Third Benchmark
Frequency of use:	Many times per session
Open Issues:	Size, location, and style of UI should be finalized by UI designer

Use Case 2.34: Change Text Color

Use-Case:	Change Text Color
Primary Actor:	Map Editor
Goal in Context:	User wishes to change the color used to render the selected map element's text, which can be either Line End text, Station text, or Text Label text
Preconditions:	User is viewing a map
Scenario:	1. User is editing a metro map in the workspace 2. User selects a map element, either by clicking on a station (or selecting it via the combo box in the Metro Stations toolbar) or clicking on a Line End, or by clicking on a Map Label in the map. Each of these actions should highlight the selected item and load their current text color into the background of the Text Color Picker 3. User clicks on the Text Color Picker in the Font toolbar to change text color
Exceptions:	N/A
Priority:	Essential, must be implemented
When available:	Third Benchmark
Frequency of use:	Many times per session
Open Issues:	Size, location, and style of UI should be finalized by UI designer

Use Case 2.35: Change Text Font

Use-Case:	Change Text Font
Primary Actor:	Map Editor
Goal in Context:	User wishes to change the font used to render the selected map element, which can be either Line End text, Station text, or Text Label text
Preconditions:	User is viewing a map
Scenario:	1. User is editing a metro map in the workspace 2. User selects a map element, either by clicking on a station (or selecting it via the combo box in the Metro Stations toolbar) or clicking on a Line End, or by clicking on a Map Label in the map. Each of these actions should highlight the selected item and load their current font values into the font controls. 3. User selects the desired font settings, meaning the font size and family and whether it is bold or italicized.
Exceptions:	N/A
Priority:	Essential, must be implemented
When available:	Third Benchmark
Frequency of use:	Many times per session
Open Issues:	Size, location, and style of UI should be finalized by UI designer

Use Case 2.36 Zoom In/Out

Use-Case:	Zoom In/Out
Primary Actor:	Map Editor
Goal in Context:	User wishes to zoom in and out on the map in order to view details better for the purpose of editing.
Preconditions:	User is viewing a map
Scenario:	1. User is editing a metro map in the workspace 2. User presses the Zoom In button to increase the zoom by 10% 3. User presses the Zoom Out button to decrease the zoom by 10%
Exceptions:	N/A
Priority:	Essential, must be implemented
When available:	Second Benchmark
Frequency of use:	Many times per session
Open Issues:	Size, location, and style of UI should be finalized by UI designer

Use Case 2.37: Increase/Decrease Map Size

Use-Case:	Increase/Decrease Map Size
Primary Actor:	Map Editor
Goal in Context:	User wishes to increase or decrease the size of the map by 10 %. Note that the size of the map means the total number of pixels (width and height) that make up the true dimensions of the image that will be exported.
Preconditions:	User is viewing a map
Scenario:	1. User is editing a metro map in the workspace 2. User presses the Increase Map Size to make the map size 10% bigger, which should instantly apply. 3. User presses the Decrease Map Size to make the map size 10% smaller, which should instantly apply. Note that no map elements should ever be moved or removed as a result of this, though some things may no longer be visible after a map is made smaller.
Exceptions:	A map cannot be made too small. The minimum width is 200 pixels wide and minimum height is 200 pixels.
Priority:	Essential, must be implemented
When available:	Second Benchmark
Frequency of use:	Many times per session
Open Issues:	Size, location, and style of UI should be finalized by UI designer

Use Case 2.38: Navigate Viewport Up/Down/Left/Right

Use-Case:	Navigate Viewport Up/Down/Left/Right
Primary Actor:	Map Editor

Goal in Context:	User wishes to navigate the viewport around the map. Note that we do this because we will have a Zoom feature which will help with editing.
Preconditions:	User is viewing a map where the zoom level and map size means the entire map is not visible inside the viewport
Scenario:	1. User is editing a metro map in the workspace 2. User presses the WASD keys to navigate around the map. W (up), A (left), S (down), and D (right). Pressing these keys should move the viewport in a uniform speed.
Exceptions:	Note that the viewport should not be permitted to navigate off of the end of the map.
Priority:	Essential, must be implemented
When available:	Second Benchmark
Frequency of use:	Many times per session
Open Issues:	Size, location, and style of UI should be finalized by UI designer

Use Case 2.37: Toggle Snap Grid

Use-Case:	Toggle Snap Grid
Primary Actor:	Map Editor
Goal in Context:	User wishes to see the map's snap grid, which would appear over the background color and any background image but behind all other map elements
Preconditions:	User is viewing a map with at least one stop
Scenario:	1. User is editing a metro map in the workspace 2. User clicks the Toggle Snap Grid checkbox. This will either hide or show the snap grid, which is a line grid that would be displayed behind everything except the background image. This grid can help line up elements into neat rows and columns. Note that all grid lines must extend being the borders of the map.
Exceptions:	N/A
Priority:	Essential, must be implemented
When available:	Third Benchmark
Frequency of use:	Many times per session
Open Issues:	Size, location, and style of UI should be finalized by UI designer

Use Case 2.39: Exit Application

Use-Case:	Exit Application
Primary Actor:	Map Editor
Goal in Context:	User wishes to exit the application
Preconditions:	User is viewing the main UI
Scenario:	1. User is viewing the main UI 2. User clicks on the X button in the top-right corner of the window. If there is unsaved work the user will be prompted to save their work (Yes/No). 3. Application closes.
Exceptions:	N/A

Priority:	Essential, must be implemented
When available:	First Benchmark
Frequency of use:	Many times per session
Open Issues:	Size, location, and style of UI should be finalized by UI designer

i. Hardware Interfaces

The application should be runnable on any platform that supports Java, but would require a keyboard and mouse.

ii. Software Interfaces

Metro Map Maker will be developed using the Java language. Note that since it is a traditional workspace-type application, it may be best to use the Desktop Java Framework.

iii. Communications Interfaces

Note that this editing application will operate locally. There will be no networking requirements.

iv. Memory Constraints

This application uses a manageable amount of user provided data so this should not be a concern.

v. Operations

We must make sure the data exported to JSON files use the proper format according to the site requirements.

vi. Site Adaptation Requirements

N/A

b. Product functions

N/A.

c. User characteristics

The editor should aim to be as user friendly as possible, using the principles of foolproof design as well as sound UI design principles.

d. **Constraints**

N/A

e. **Assumptions and dependencies**

N/A

f. Apportioning of the Requirements

N/A

3 Specific requirements

The Metro Map Maker application will require a number of user interface contexts and components including:

- Welcome Dialog
- Main UI
    - Top Pane
        - File Toolbar
        - Undo/Redo Toolbar
        - About Toolbar
    - Left Pane
        - Metro Lines Toolbar
        - Metro Stations Toolbar
        - Station Router Toolbar
        - Décor Toolbar
        - Font Toolbar
        - Navigation Toolbar
    - Center Workspace Pane (where map appears)
- Metro Line Edit Dialog
- Metro Line Station Listing Dialog
- Route Display Dialog

3.1 **External interfaces**

The following wireframe mockups provide a look at the types of controls and layout to be used for the User Interface. Note that the User Interface designer should select the appropriate icons for all buttons and should carefully choose color and font combinations that provide good contrast and attract the eye. There are five UI diagrams, one for the welcome dialog, one for the main UI, and then three additional dialogs used during editing. The User Interface designer should also consider additional dialogs for providing adequate feedback to the user as well as for navigation through the file system to select files as part of certain use cases.

Figure 3.1 Welcome Dialog

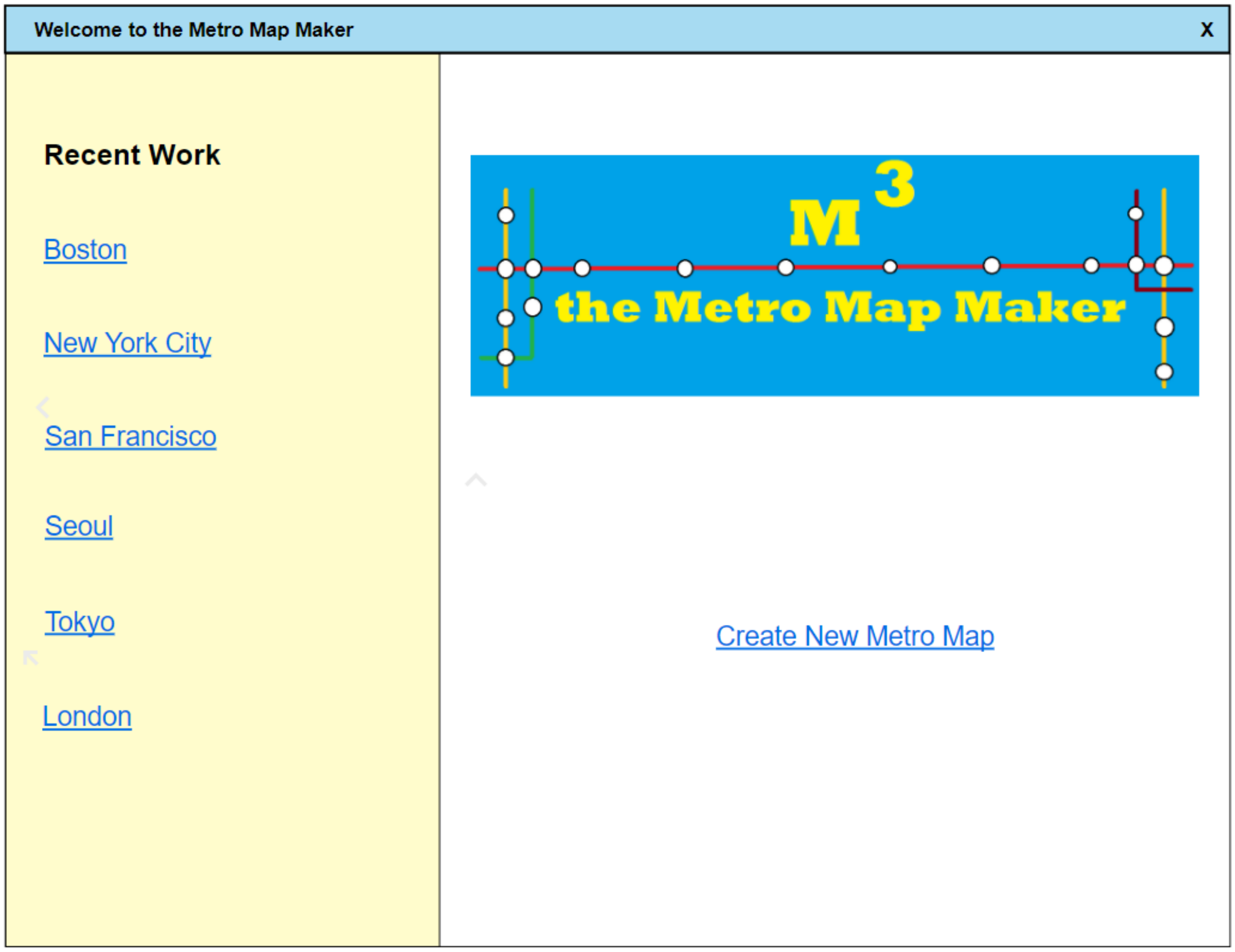


Figure 3.2 Metro Map Maker Workspace

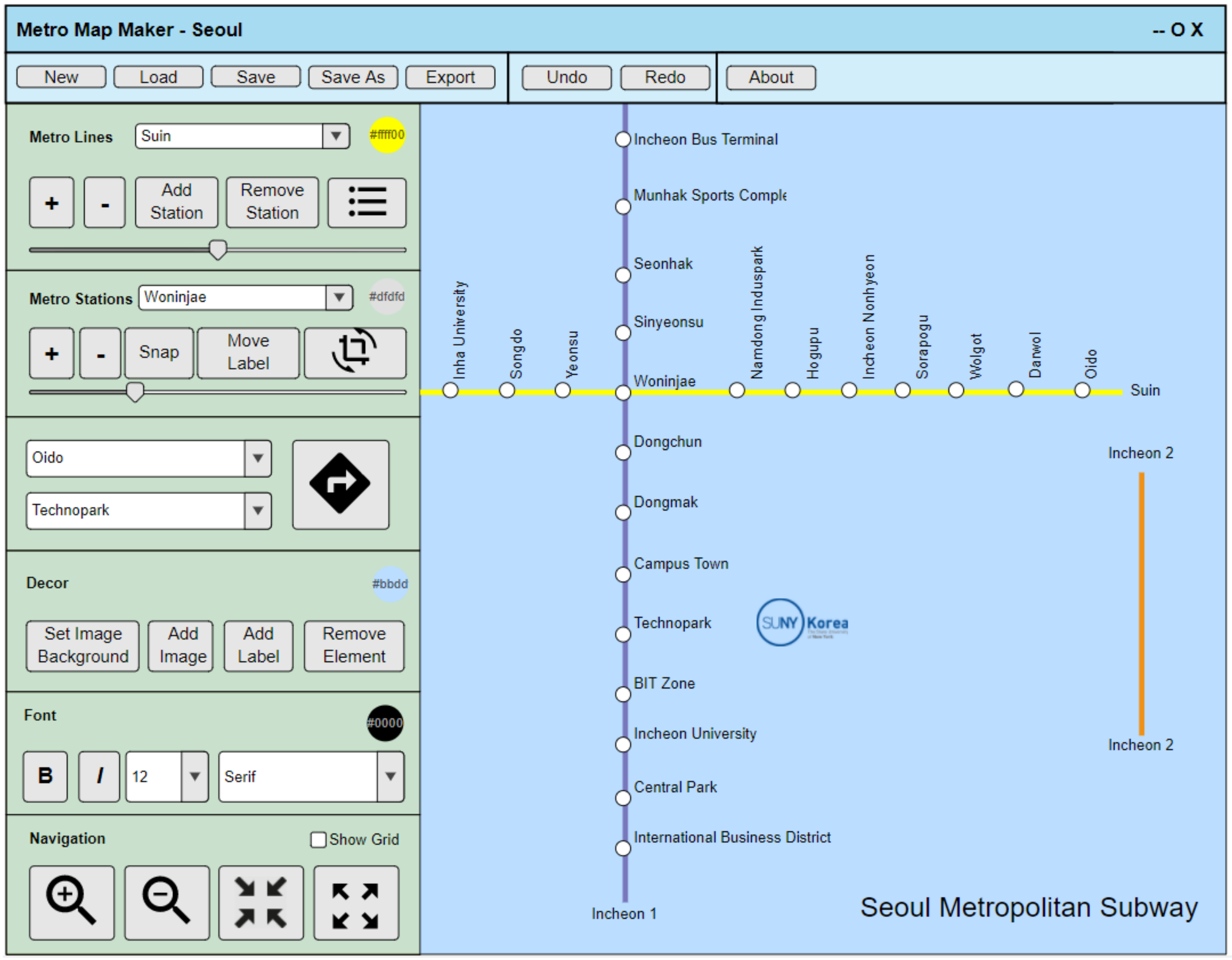


Figure 3.3 Metro Line Edit Dialog

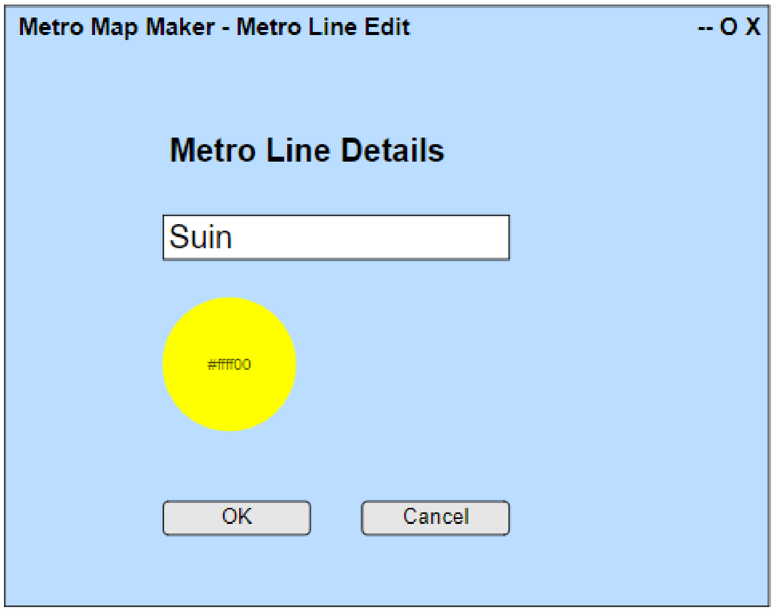
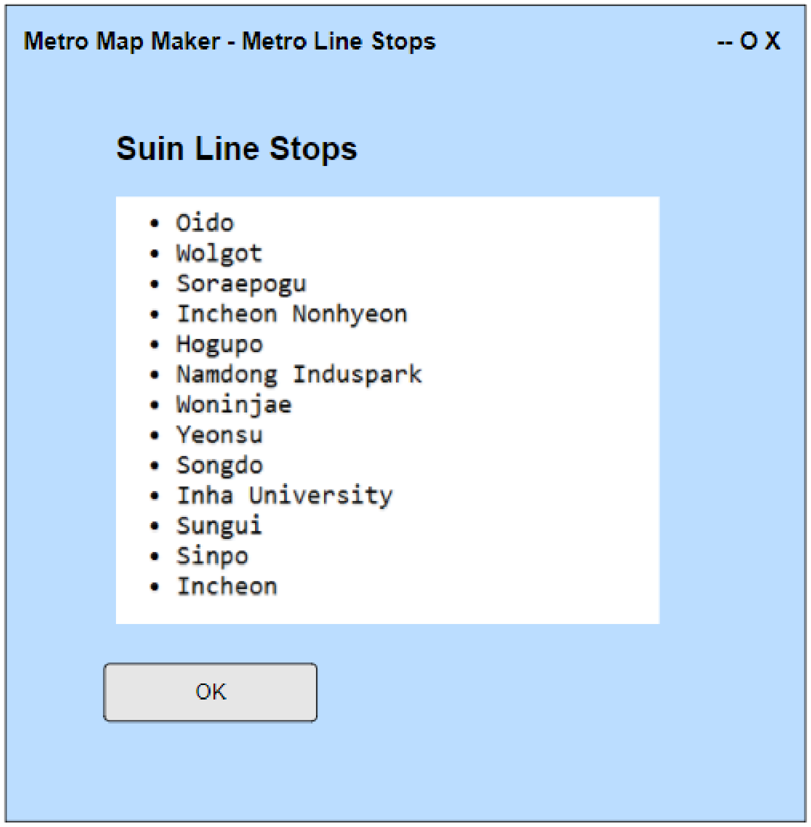


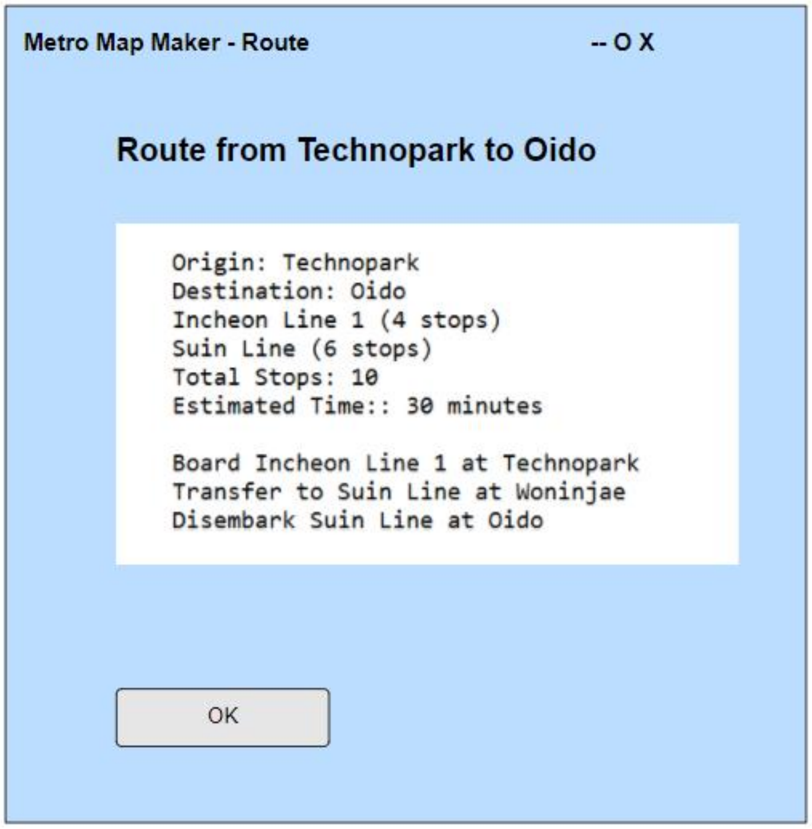
Figure 3.4 Metro Line Station Listing Dialog



Suin Line Stops

- Oido
- Wolgot
- Soraepogu
- Incheon Nonhyeon
- Hogupo
- Namdong Induspark
- Woninjae
- Yeonsu
- Songdo
- Inha University
- Sungui
- Sinpo
- Incheon

Figure 3.5 Route Display Dialog



Route from Technopark to Oido

```
Origin: Technopark
Destination: Oido
Incheon Line 1 (4 stops)
Suin Line (6 stops)
Total Stops: 10
Estimated Time:: 30 minutes

Board Incheon Line 1 at Technopark
Transfer to Suin Line at Woninjae
Disembark Suin Line at Oido
```

3.2 **Functions**

One of the important things to consider in our application is providing the appropriate feedback to the user. Users need feedback to enjoy their experience. This is typically done with visual cues like dialog boxes.

3.3 **Performance requirements**

N/A

3.4 **Logical database requirements**

N/A

3.5 **Design constraints**

JavaFX will be used because it effectively leverages each system's available rendering technologies and provides platform independence for personal computers.

3.6 **Software system attributes**

As professionals, all members of this project must take this project seriously. We are dedicated to producing robust software that exceeds the expectations of our customers. In order to achieve this level of quality, we should build a product with the following properties in mind:

3.6.1 Reliability – The program should be carefully planned, constructed and tested such that it behaves flawlessly for the end user. Bugs, including rendering problems, are unacceptable. In order to minimize these problems, all software will be carefully designed using UML diagrams and a Design to Test approach should be used for the Implementation Stage.

3.6.2 Availability –Customers may download and install the application for free.

3.6.3 Security – All security mechanisms will be addressed by future revisions

3.6.4 Extensibility – It is possible that more JavaScript/JSON widgets might be added to course sites in the future, so by providing an additional tab with suitable data and making changes to exporting methods, this should be considered during this deaign.

1.6.5 Portability – To start with, the app will target desktop Java applications.

3.6.6 Maintainability – Update mechanisms will be addressed by future revisions.

3.7 **Organizing the specific requirements**

Note that the application is simple enough that we need not worry about using an alternative arrangement of the content of this document. The specific requirements for this application already fit neatly into the sections listed in the IEEE's recommended SRS format.

3.8 Additional comments

It is important to keep in mind that the UI designers and instructors should make updates to the themes and content as need to make something that looks great. It will be to their discretion to design all the interface controls in an effective, interactive style.

4 Supporting Information

Note that this document should serve as a reference for the designers and coders in the future stages of the development process, so we'll provide a table of contents to help quickly find important sections.

4.1 Table of contents

4.2 **Appendixes**

N/A

Debugging Enterprises