MICHIGAN STATE UNIVERSITY

# Project Plan Presentation

## General RAte Calculation Environment IDE

## The Capstone Experience

### Team Delta Dental Knowledge Science 1

Hyunmin Kim
Joseph Nagy
Anthony Rodeman
Qinghao Shen
Justin Swinehart

Department of Computer Science and Engineering
Michigan State University

Spring 2022

*From Students…*
*…to Professionals*

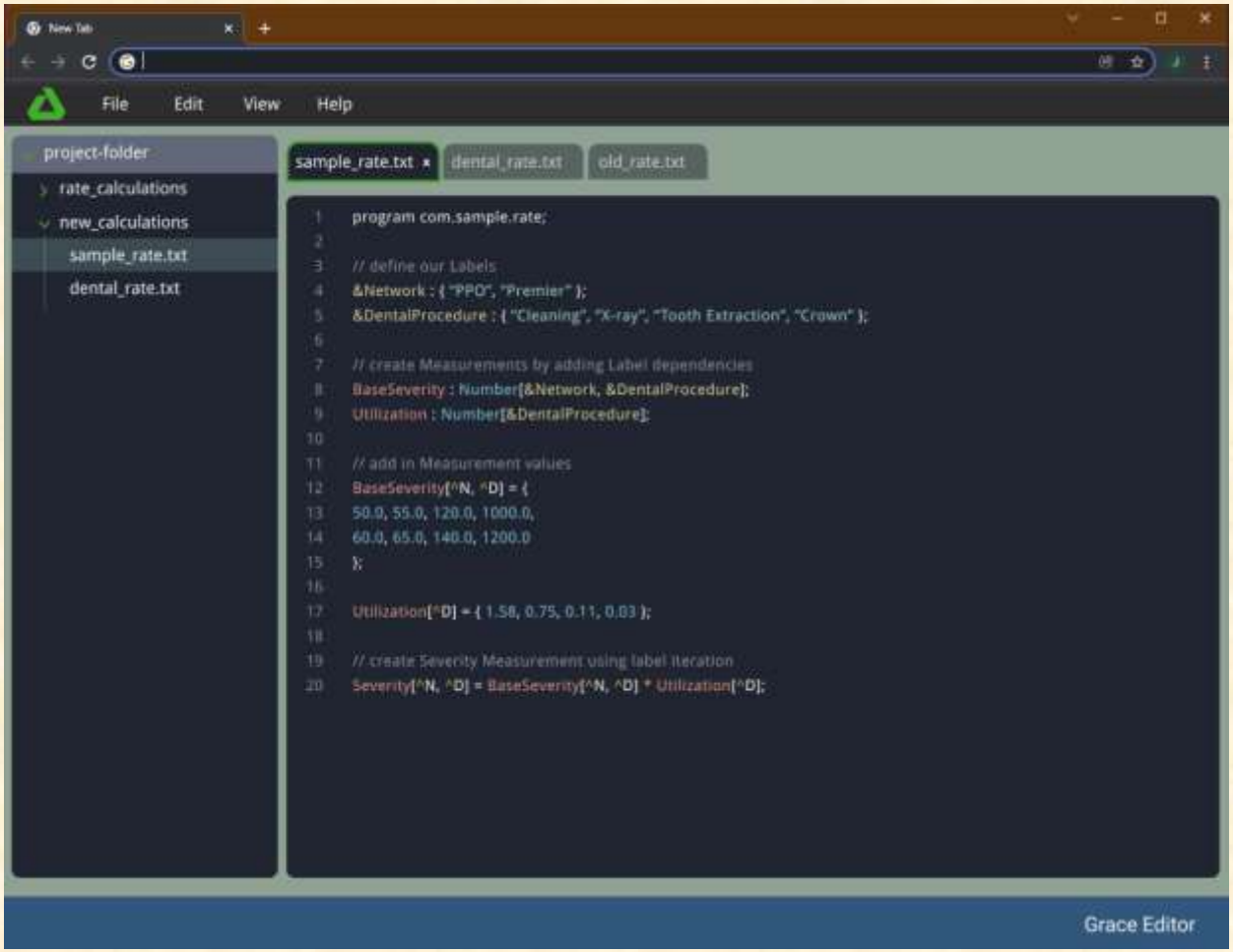
# Functional Specifications

- Web-based editor for General Rate Calculation Environment (GRACE) language
- Helpful code features like code coloring and error highlighting
- Visual Studio Code plugin with same features for more complex development environments

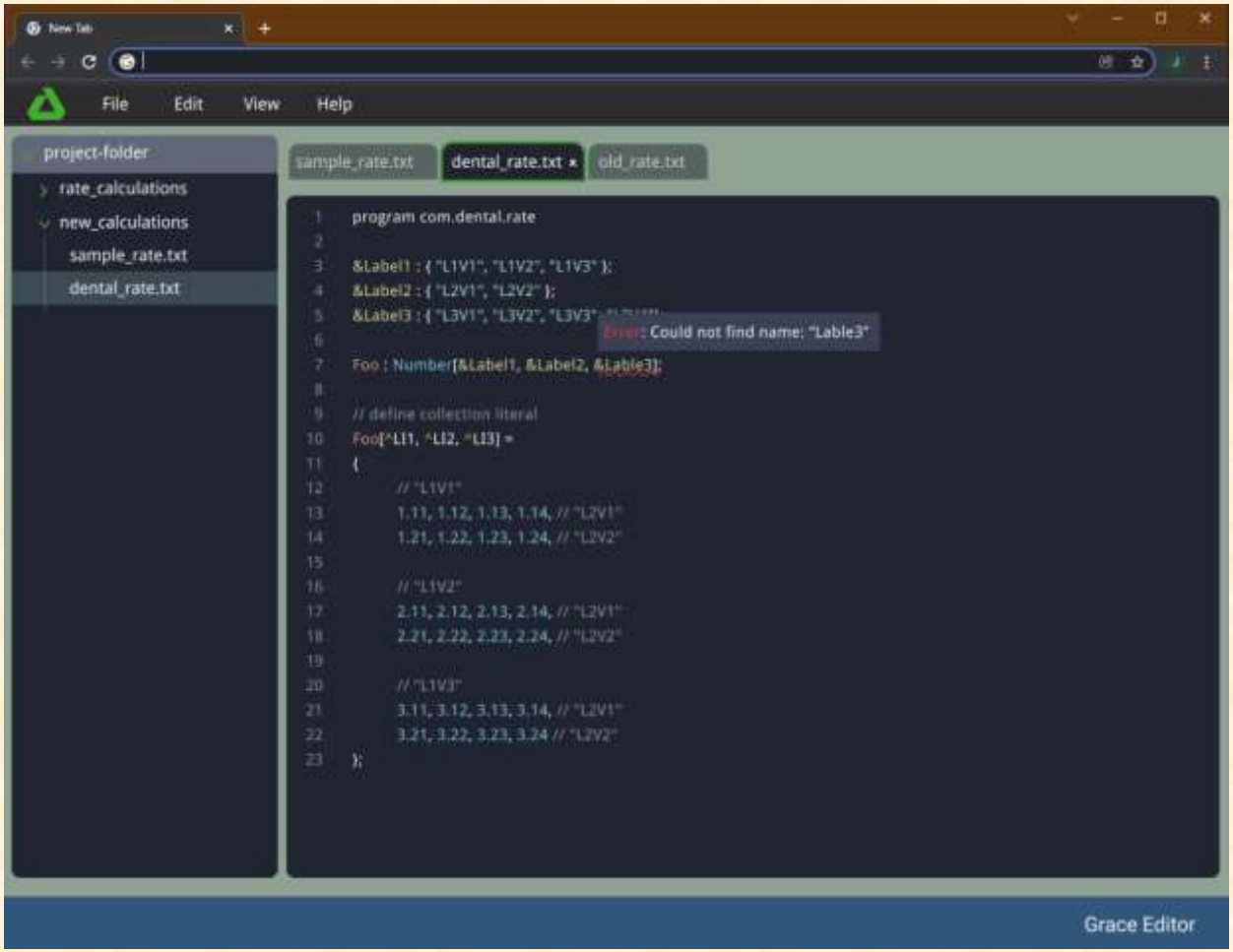
# Design Specifications

- Syntactic Highlighting
- Error Checking
- Code Navigation
- IntelliSense Completion
- Label Navigator

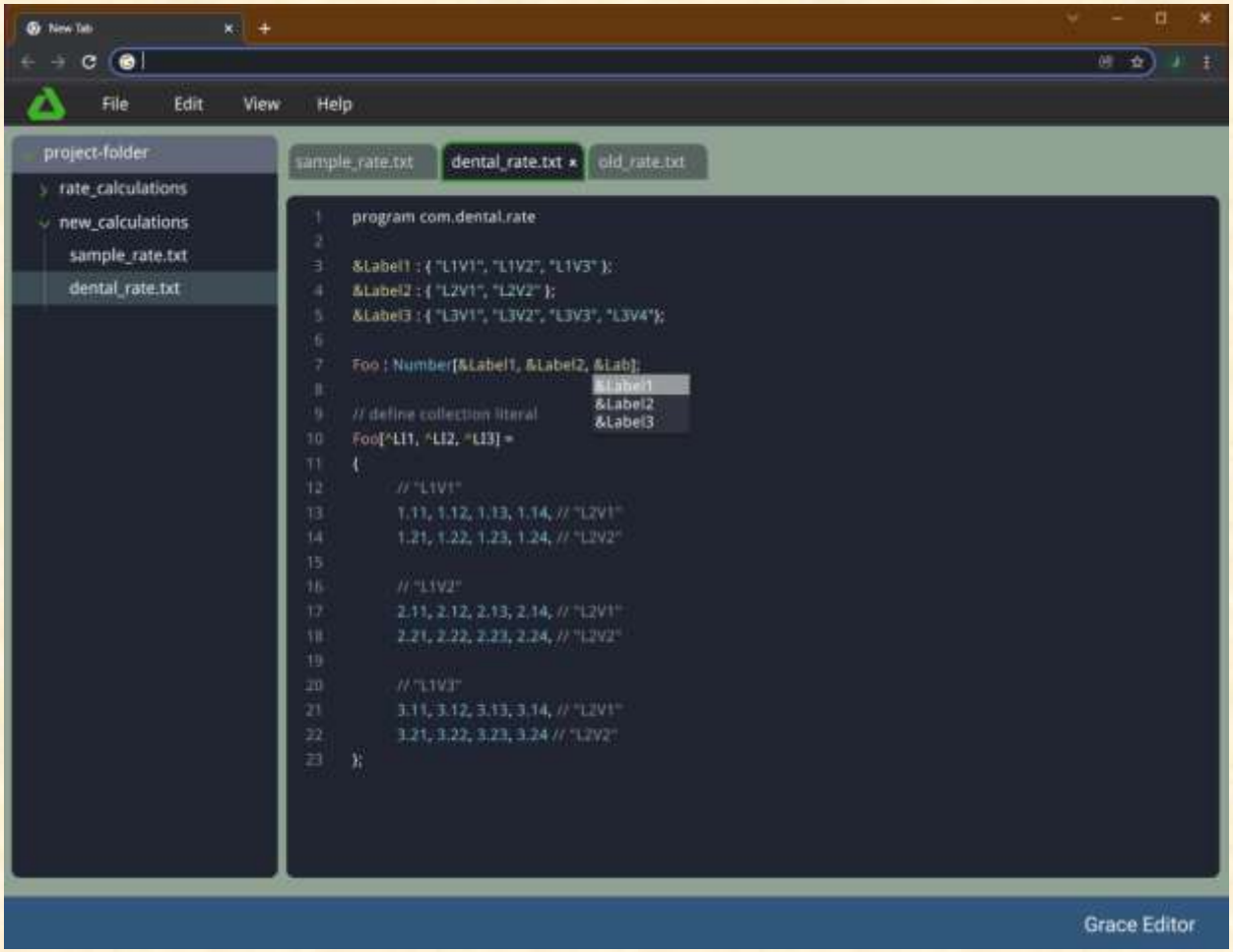
# Screen Mockup: User Interface



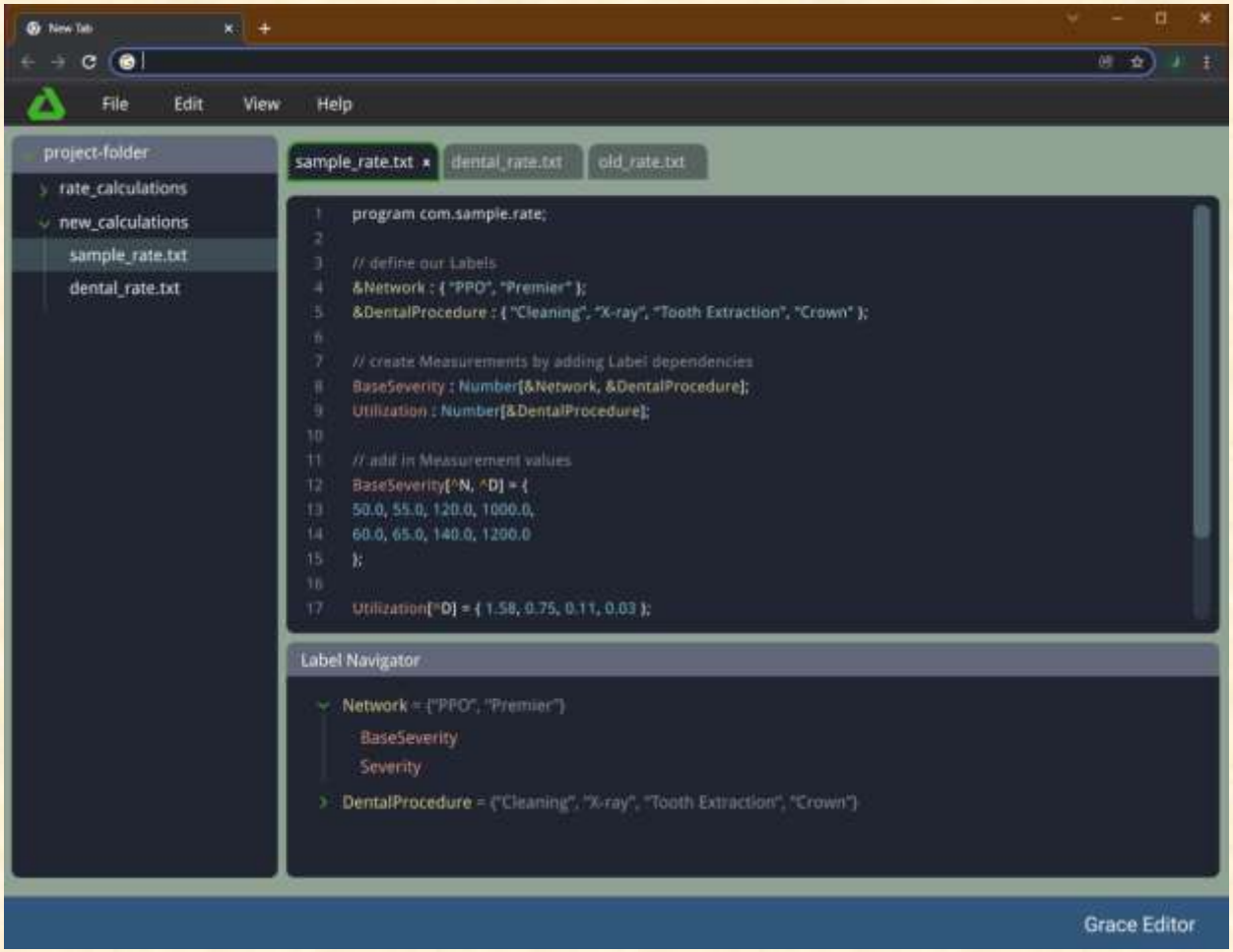
# Screen Mockup: Syntactic Highlighting and Error Checking



# Screen Mockup: IntelliSense Completion



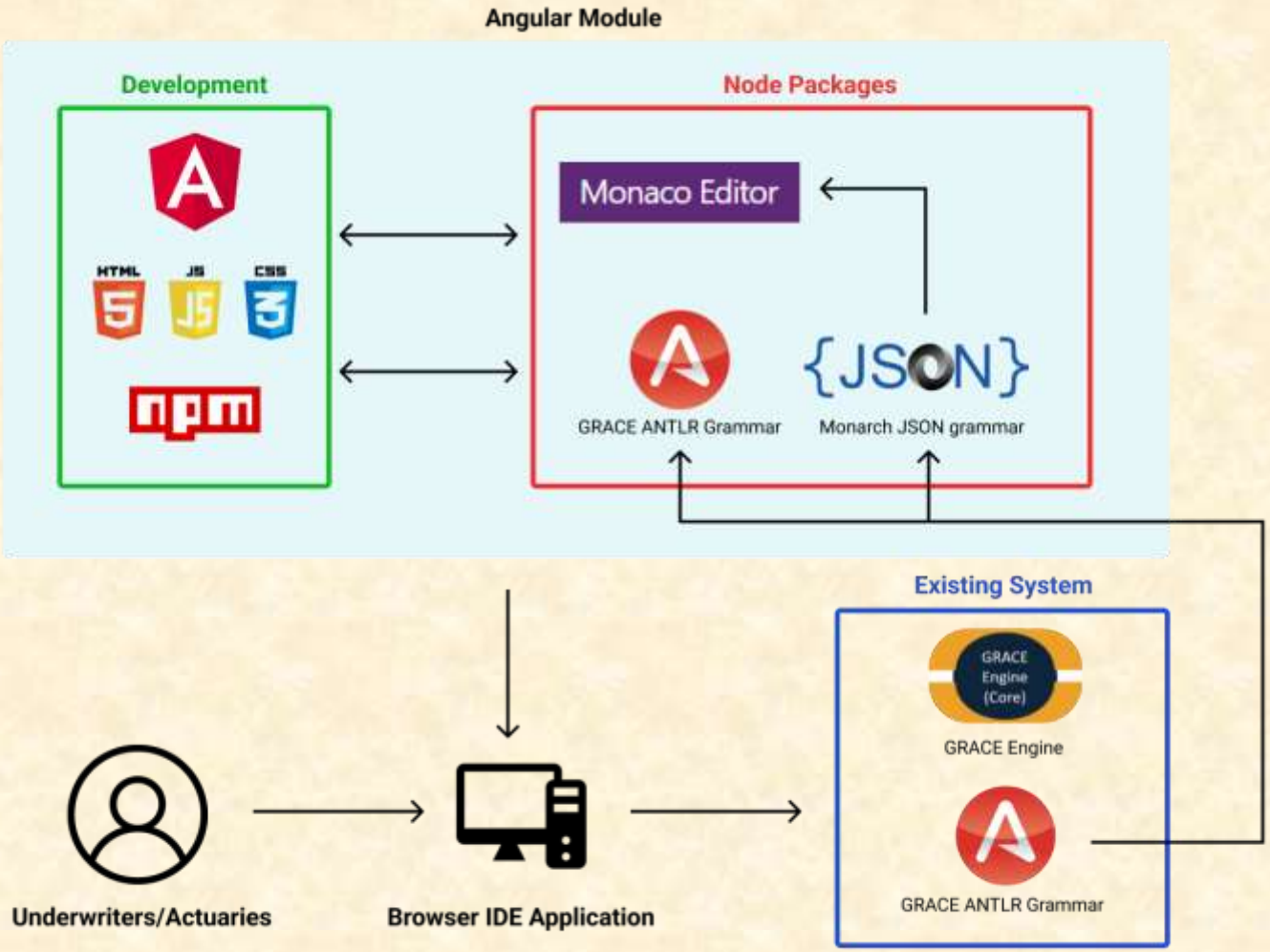
# Screen Mockup: Label Navigator



# Technical Specifications

- Developed as an Angular Module using HTML/ CSS/ JS and managed by NPM.
- Monaco Editor API will provide custom the editor interface as well as language support for typical IDE features.
- ANTLR is a parser generator that will interpret code from within the Monaco Editor to provide error handling.

# System Architecture



# System Components

- Hardware Platforms
  - N/A
- Software Platforms / Technologies
  - Angular Web Application Framework.
  - ANTLR4 JavaScript Runtime Library.
  - Monaco Editor API.

# Risks

- Integration of Different Systems
  - In the development of the project, we will use Angular, Monaco Editor, and ANTLR to develop this IDE. We need to make sure that these three systems can be combined into one.
  - We will divide core tasks for these three systems first, then we can consider how to merge them.
- Grammar-Derived Code Analysis
  - The language features are meant to be computed from the GRACE grammar so that changes to the language syntax can be immediately reflected in the language's tools.
  - We will incrementally upgrade the code analysis features. To establish the front-end experience, some of the components can be produced by hand. The first major step is to make tools with ANTLR to generate those hand-produced assets. Finally, the tools can be made to compute those assets at runtime in memory.
- Performant Code Analysis
  - The code analysis will be running client-side using the JavaScript build of the ANTLR engine. As a web application, GRACE IDE needs to be responsive which means the routines must be efficient while they continuously process new input.
  - ANTLR is already a well-designed system, so performance bottlenecks will be within our control. Incrementing on our abstract syntax tree handlers and testing on a variety of hardware and browsers will let us gauge and improve the overall performance impact of code analysis.

# Questions?