

MICHIGAN STATE

UNIVERSITY

Project Plan Presentation

General Rate Calculation Environment Shell

The Capstone Experience

Team Delta Dental Knowledge Science 2

Dylan Boyd
Kyle Ernster
Huy Nguyen
Justin Park
David Robbins
Yang Zhao

Department of Computer Science and Engineering
Michigan State University

Spring 2022



*From Students...
...to Professionals*

Functional Specifications

- GRACE can be difficult to use for people who are unfamiliar with the technology it involves.
- We aim to provide GRACE program developers with the same valuable console support you would find in mainstream programming languages.
- Our interactive shell will allow for interactive development, real-time testing, and provide traceability of the development process.

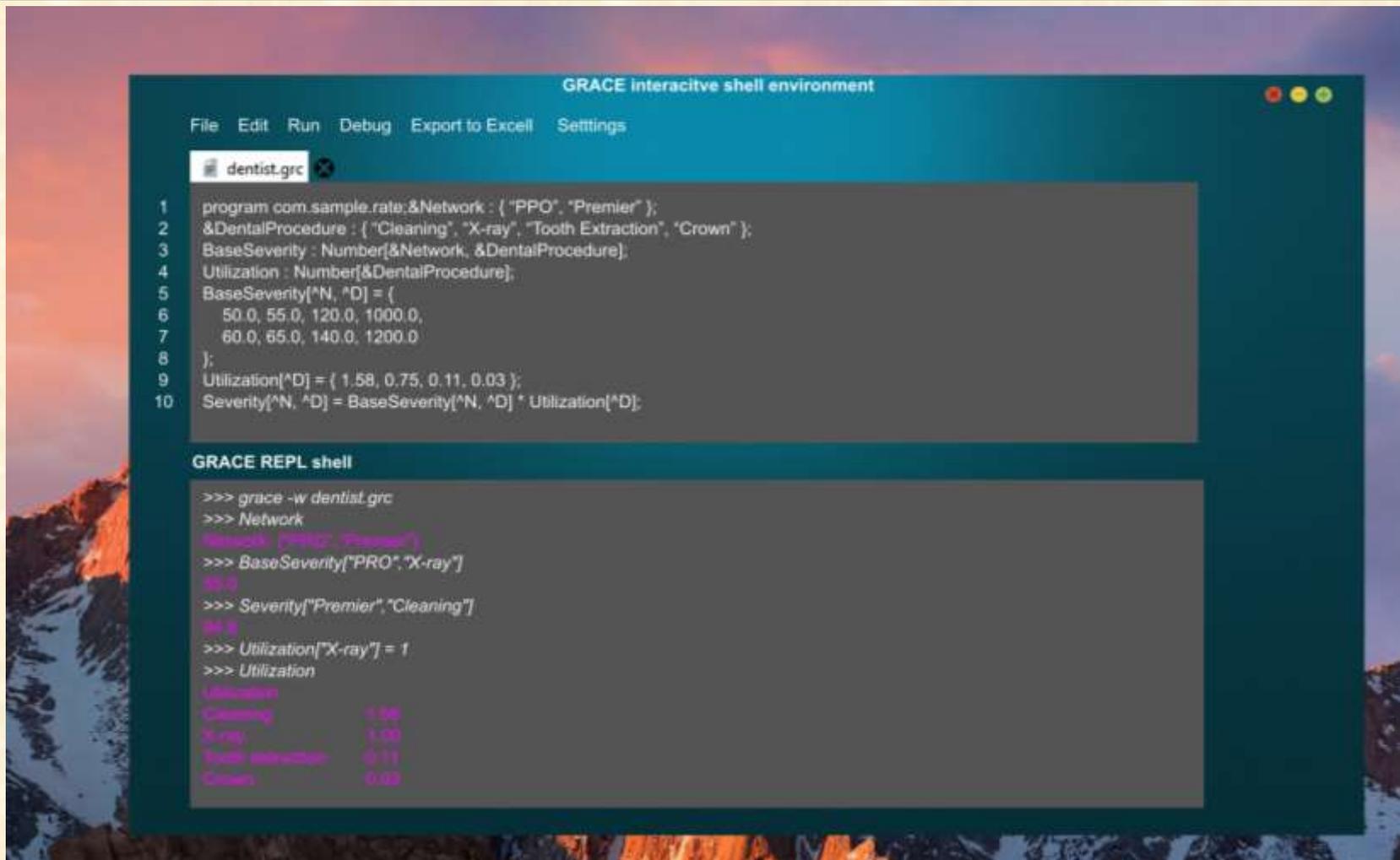


Design Specifications

- Meant for both technical and non-technical users
- User side terminal that looks like an OS shell
- Prompts user for command
- Can save/load data files



Screen Mockup: GRACE interactive shell

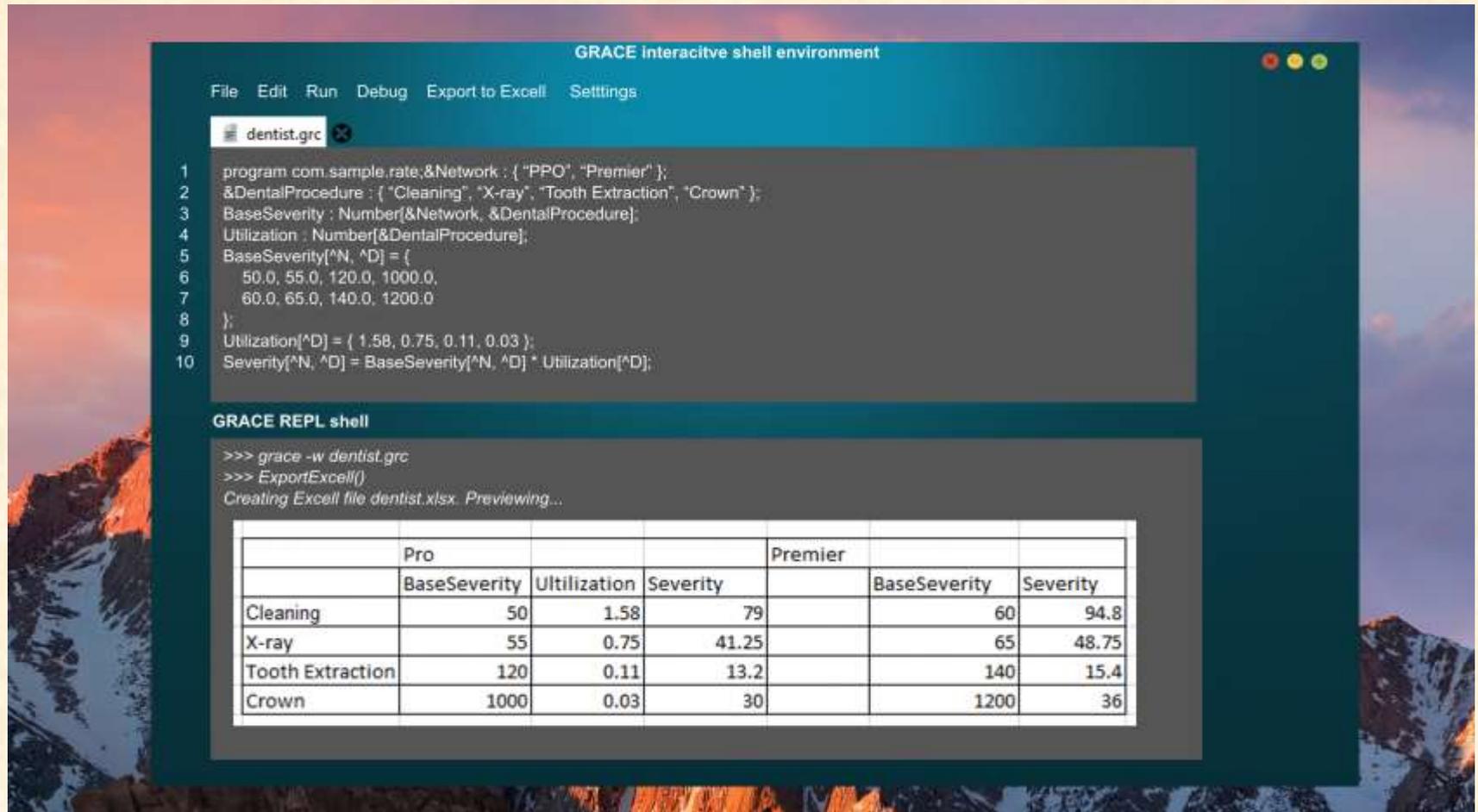


Screen Mockup: GRACE shell only

```
GRACE REPL shell
>>> a = 11
... b = 11;
>>> a * b
121
>>> students = {"Bob", "Alice", "James", "Nicholas"};
>>> exams = {"Midterm", "Final"};
>>> Table grade[ exams, students ] {
... 3 4 4 3
... 2 1 2 0};
>>> grade["Midterm", "Alice"]
4
>>> grade["Final"]
Bob Alice James Nicholas
2 1 2 0
>>> students.add("Kevin")
>>> grade["Midterm", "Kevin"] = 4;
>>> grade["Final", "Kevin"] = 4;
>>> grade
      Bob Alice James Nicholas Kevin
Midterm 3 4 4 3 4
Final 2 1 2 0 4
```



Screen Mockup: Export to Excel



The screenshot displays the GRACE interactive shell environment. The window title is "GRACE interactive shell environment". The menu bar includes "File", "Edit", "Run", "Debug", "Export to Excell", and "Settings". A tab labeled "dentist.grc" is active. The code editor shows the following code:

```
1 program com.sample.rate.&Network : { "PPO", "Premier" };
2 &DentalProcedure : { "Cleaning", "X-ray", "Tooth Extraction", "Crown" };
3 BaseSeverity : Number(&Network, &DentalProcedure);
4 Utilization : Number(&DentalProcedure);
5 BaseSeverity[^N, ^D] = {
6   50.0, 55.0, 120.0, 1000.0,
7   60.0, 65.0, 140.0, 1200.0
8 };
9 Utilization[^D] = { 1.58, 0.75, 0.11, 0.03 };
10 Severity[^N, ^D] = BaseSeverity[^N, ^D] * Utilization[^D];
```

Below the code editor, the "GRACE REPL shell" shows the following commands and output:

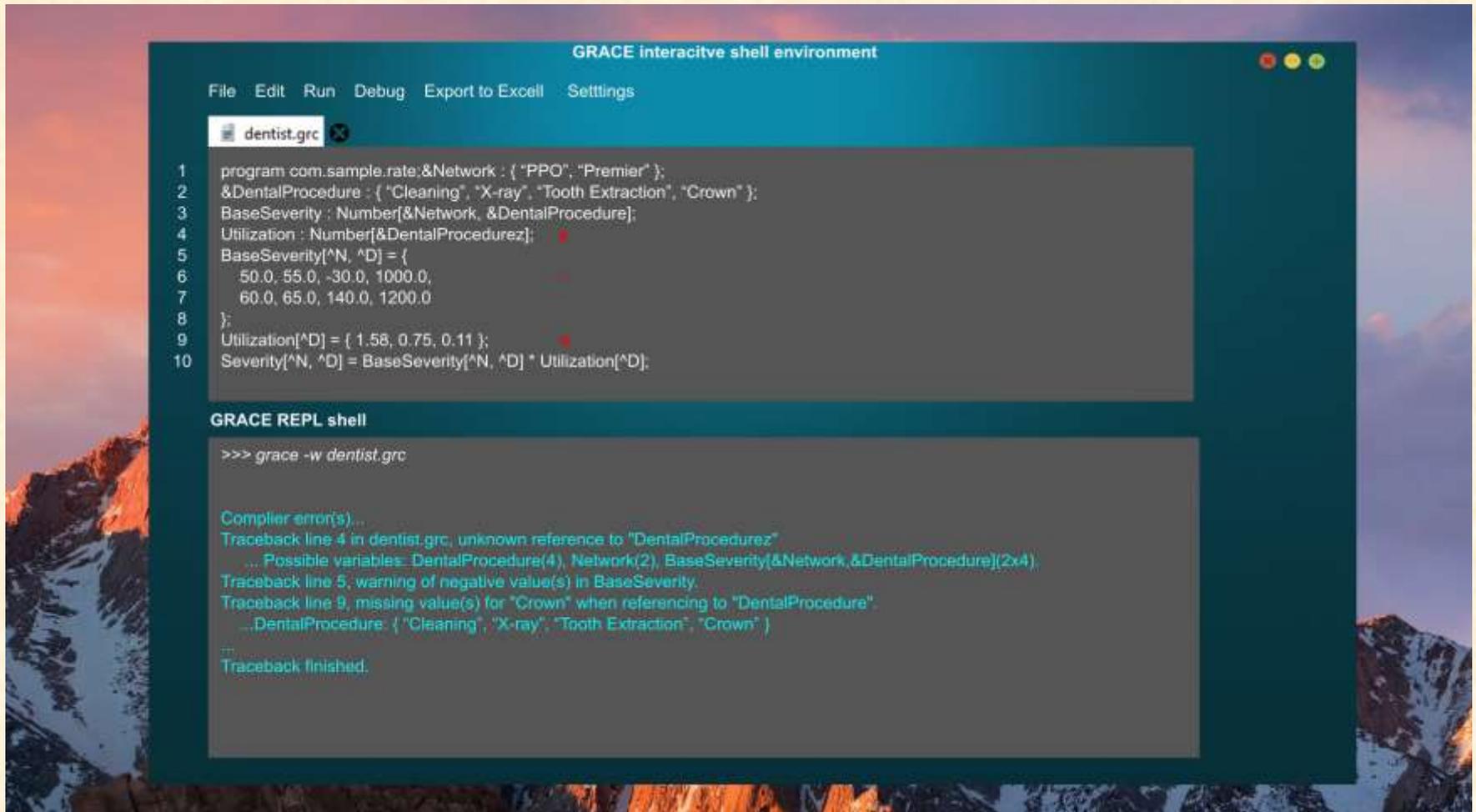
```
>>> grace -w dentist.grc
>>> ExportExcell()
Creating Excell file dentist.xlsx. Previewing...
```

The output is a table with the following data:

	Pro			Premier		
	BaseSeverity	Utilization	Severity	BaseSeverity	Severity	
Cleaning	50	1.58	79	60	94.8	
X-ray	55	0.75	41.25	65	48.75	
Tooth Extraction	120	0.11	13.2	140	15.4	
Crown	1000	0.03	30	1200	36	



Screen Mockup: Error Checking



```
GRACE interactive shell environment
File Edit Run Debug Export to Excell Settings
dentist.grc
1 program com.sample.rate:&Network : { "PPO", "Premier" };
2 &DentalProcedure : { "Cleaning", "X-ray", "Tooth Extraction", "Crown" };
3 BaseSeverity : Number[&Network, &DentalProcedure];
4 Utilization : Number[&DentalProcedurez];
5 BaseSeverity[^N, ^D] = {
6   50.0, 55.0, -30.0, 1000.0,
7   60.0, 65.0, 140.0, 1200.0
8 };
9 Utilization[^D] = { 1.58, 0.75, 0.11 };
10 Severity[^N, ^D] = BaseSeverity[^N, ^D] * Utilization[^D];

GRACE REPL shell
>>> grace -w dentist.grc

Compiler error(s)...
Traceback line 4 in dentist.grc, unknown reference to "DentalProcedurez"
... Possible variables: DentalProcedure(4), Network(2), BaseSeverity(&Network, &DentalProcedure)(2x4).
Traceback line 5, warning of negative value(s) in BaseSeverity.
Traceback line 9, missing value(s) for "Crown" when referencing to "DentalProcedure".
...DentalProcedure: { "Cleaning", "X-ray", "Tooth Extraction", "Crown" }
...
Traceback finished.
```

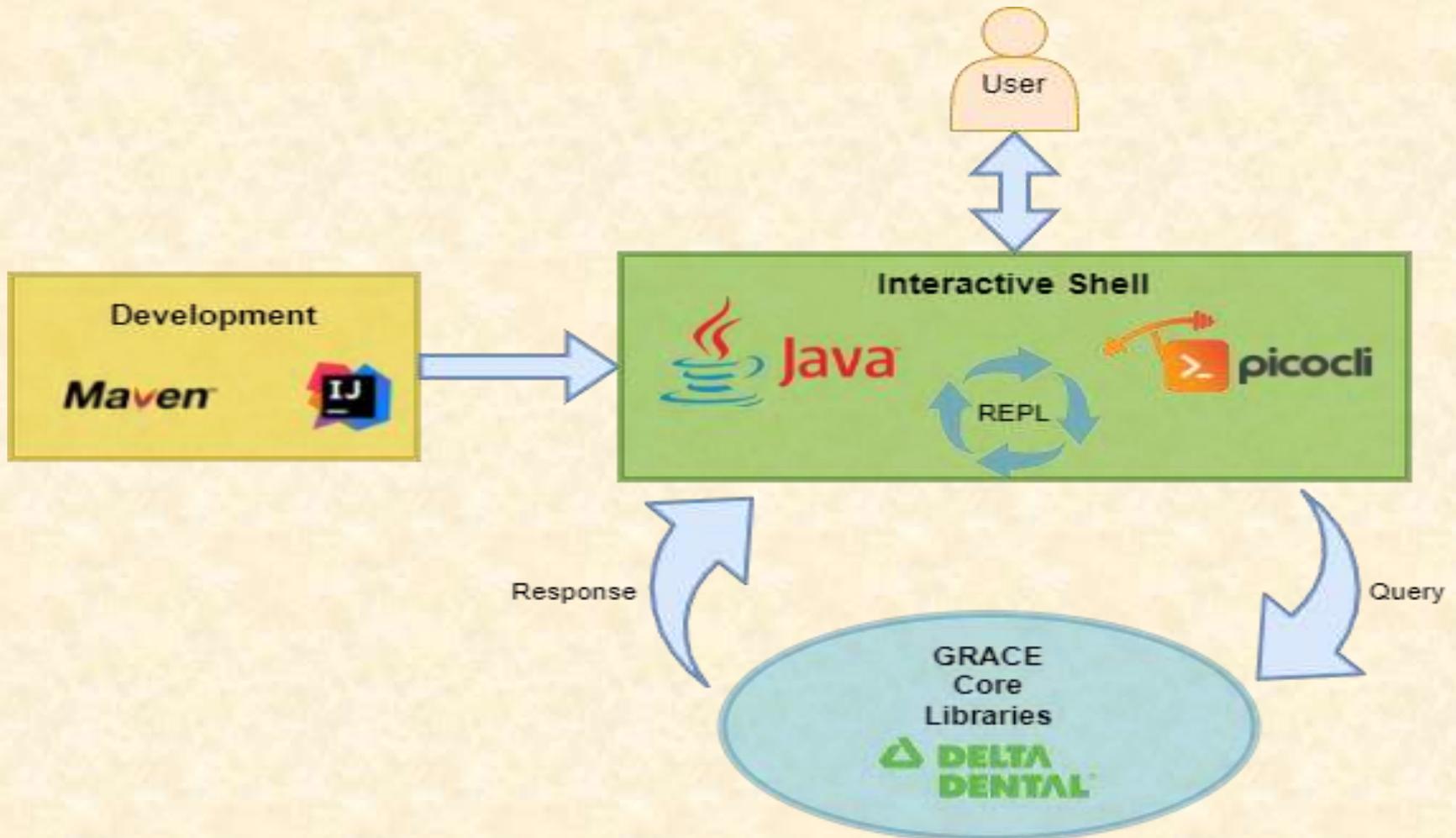


Technical Specifications

- Bundled, single .jar, Java desktop application
- Runs GRACE commands in a REPL Shell session.
- Commands will build models and compute rate calculations that access GRACE libraries.
- Stores commands in the shell session state.
- Can save the session's commands as a file and load it in a different session, restoring the state.



System Architecture



System Components

- Software Platforms / Technologies
 - JDK – version 17.0.2
 - Picocli – framework from which command line interface (CLI) will be built.
 - Maven – manages build pipeline for the project including dependency control, testing, deploying, etc.
 - IntelliJ – IDE used for the project development and Maven integration.
 - GRACE Core – Utilized by CLI for algorithms, data structures and other command handling elements.
 - Github – Version control and deployment



Risks

- Risk 1: Command Line Interface Frameworks and Features
 - Our team is still unfamiliar with advanced CLI features and which CLI features would specifically benefit our product.
 - Research various CLIs and consider additional features.
- Risk 2: Readable and Intuitive Syntax
 - The CLI must be easy for a user to learn and be relatively readable.
 - Researching common syntaxes and discussing with client features that would make the system easier to use.
- Risk 3: GRACE Libraries Integration
 - The GRACE libraries are not currently accessible and likely will not be accessible until around the completion of the Alpha version of the product.
 - Picocli was picked for its natural language processing.
 - Alpha version will be developed with integration in mind.



Questions?

?

?

?

?

?

?

?

?

?

