Stage 3: Prototype

|  |  |  |  |
| --- | --- | --- | --- |
| Project Abbrev & Name | **GFM: An enhanced Grammatical Framework minibar** | Client + email | **Zola Mahlaza** [**zola.mahlaza@uct.ac.za**](mailto:zola.mahlaza@uct.ac.za) |
| Date | **17 August 2023** | Tutor + email | **Hamza Amir** [**amrham001@myuct.ac.za**](mailto:amrham001@myuct.ac.za) |
| Team Members | **KPPDEA001, Dean Kopping,** [**kppdea001@myuct.ac.za**](mailto:kppdea001@myuct.ac.za) | | |
|  | **FRNMOD001, Modjadji Francis,** [**frnmod001@myuct.ac.za**](mailto:frnmod001@myuct.ac.za) | | |
|  | **TSHCAS005, Cassandra Tshabalala,** [**tshcas005@myuct.ac.za**](mailto:tshcas005@myuct.ac.za) | | |

**Table of Contents**

[Prototype choice justification: 1](#_Toc1617218836)

[Prototype Designs 2](#_Toc1842759286)

[Design 1: 3](#_Toc1651368329)

[Design 2: 3](#_Toc1073527139)

[Design 3: 4](#_Toc1861348399)

[Critical Analysis and Final Prototype Choice 5](#_Toc1991427469)

[Class definitions: 6](#_Toc696799308)

[Class member functions: 8](#_Toc1546251083)

[Type Class: 12](#_Toc890693969)

[Class Inheritance 13](#_Toc220744983)

[Scope 14](#_Toc1227573853)

# **Prototype choice justification:**

**Evolutionary**:

The aim of our project is to produce an enhanced version of the minibar application, an already existing and functional system. As a result, we have chosen an evolutionary prototype to utilize the provided foundation as a starting point which should kickstart the team’s productivity rather than starting from scratch.

In addition, our team is small and only consists of three members who will be handling all stages of development. For this reason, throw-away prototyping may be unproductive and will not allow the team to make reasonable progress in terms of developing all the required features to a high standard and having them fully functional.

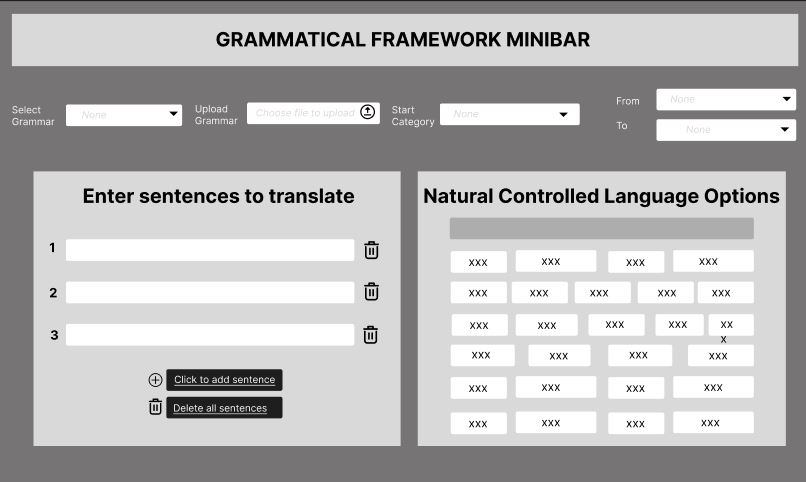
**Vertical**:

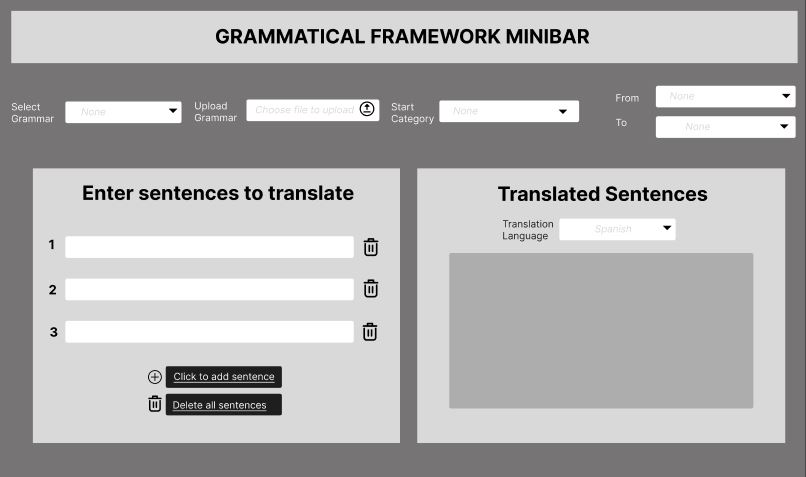
For this project, the team is only required to complete a small set of functional requirements that will be built into the grammatical framework minibar. Therefore, we have selected to utilize the vertical prototyping method as it will allow us to dive deeper into each requirement and develop them to a higher standard. This is beneficial to realizing our goals for the project as opposed to creating a larger range of functionality with less detail.

The vertical prototyping method works hand in hand with the evolutionary prototyping method which we have opted for. This is because we will be building on top of a foundation that already operates as intended and is unlikely to need much editing. Due to this, we can continue to build on the prototype while maintaining the applications functionality.

**Prototype Designs**:

# Design 1:



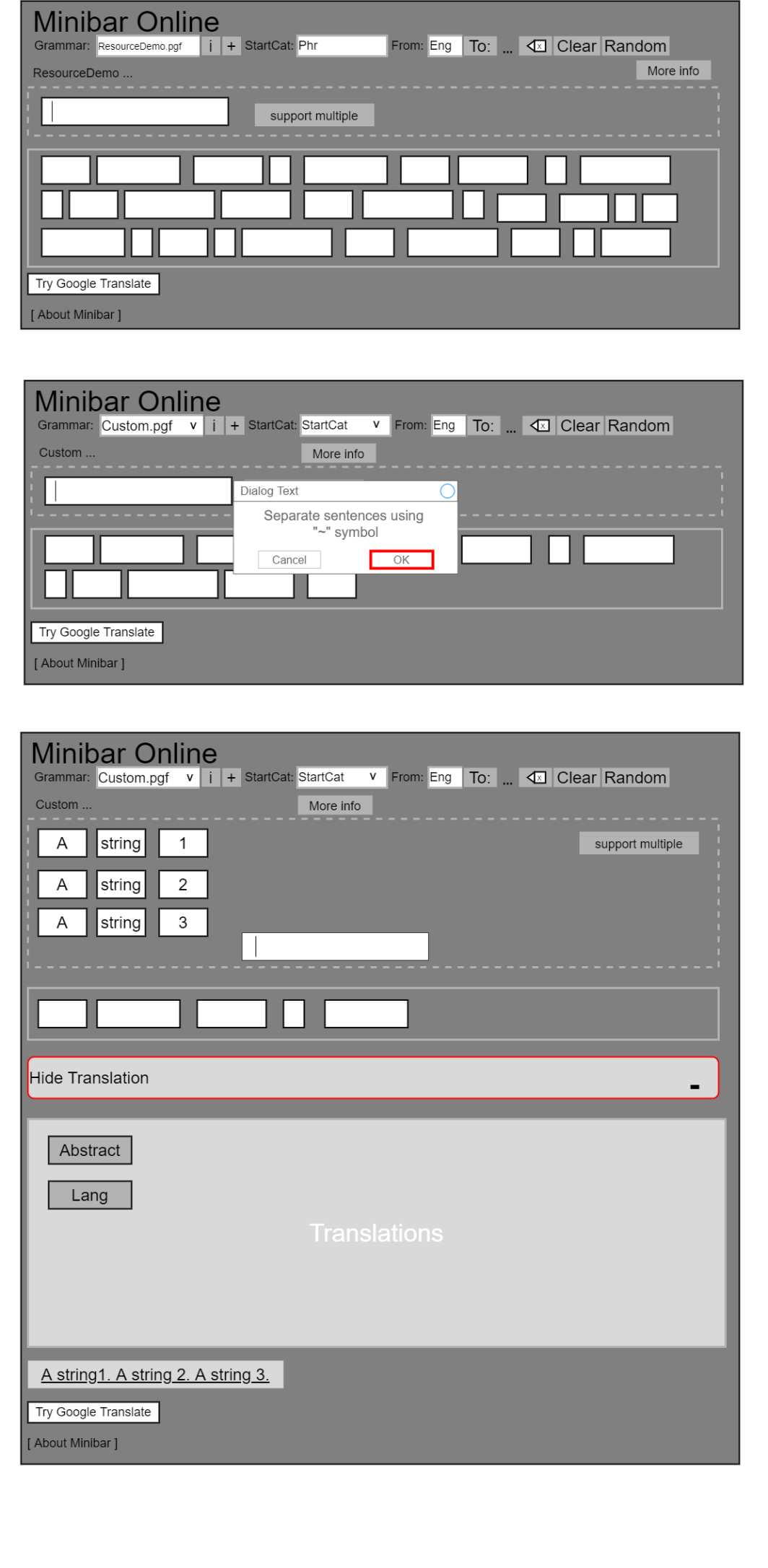


## **Design 2**

# 

## 

# Design 3

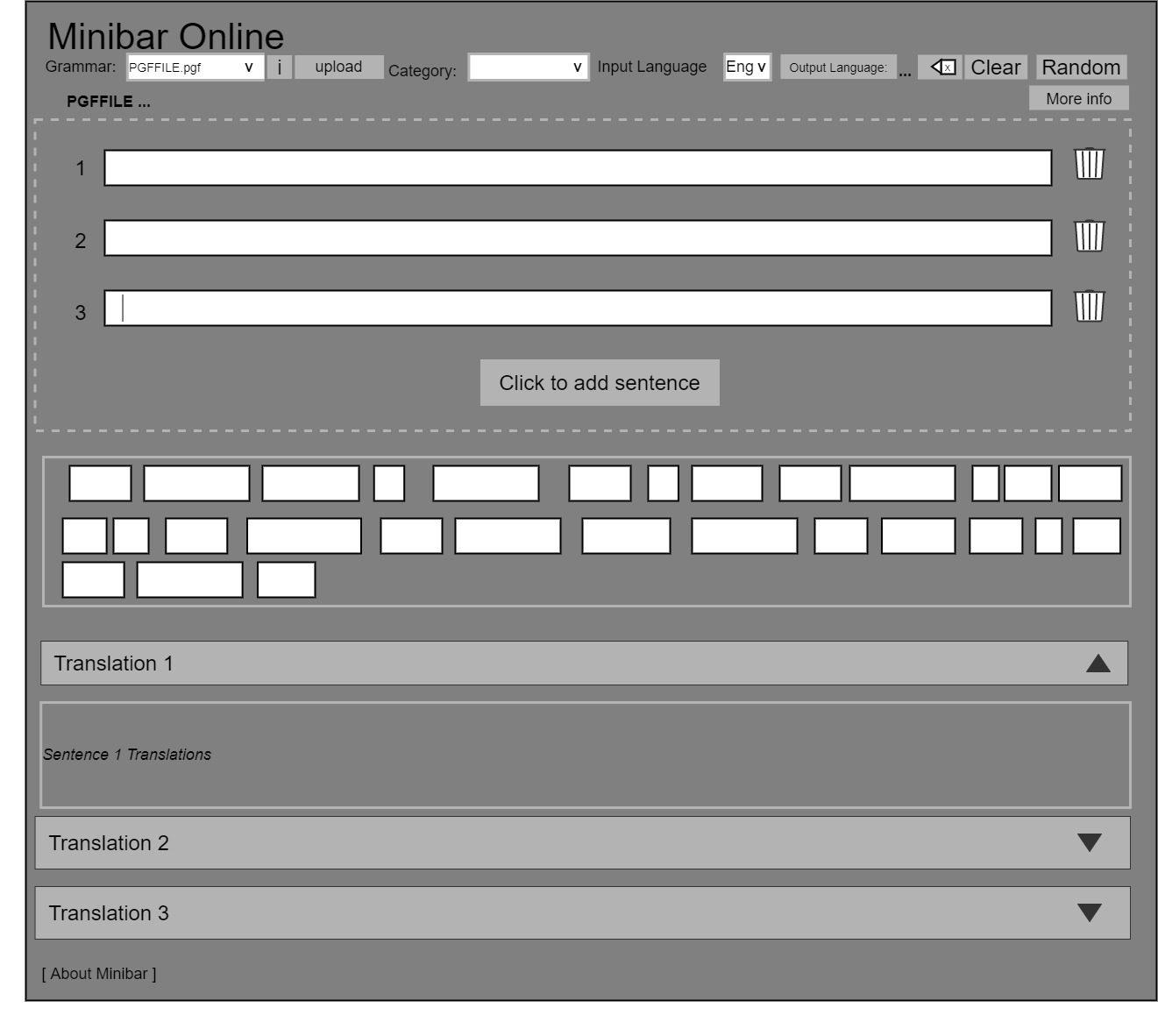


# Critical Analysis and Final Prototype Choice

We have formulated a metric table that we will use to compare and contrast the above prototypes

|  |  |  |  |
| --- | --- | --- | --- |
| **Metric** | **Design 1** | **Design 2** | **Design 3** |
| **Usability** | Clear labels and instructions that indicate to the user what actions to take in order to interact with the platform.  The use of affordances makes the design intuitive and allows ease of use for the system user. This can be seen in the form of clear icons (e.g. bin icon) that indicate functionality.  The design incorporates clear labels and buttons, improving the overall usability. | The design is very simple and clear which makes it intuitive to the user.  The adjustable settings are large and clearly labelled making them accessible and their functionality obvious. | Design implements the existing design with additional features sitting close to the existing ones. The use of buttons and collapse buttons makes it easier to get actions done. |
| **Ability to add files** | The design includes the functionality for the user to upload their own grammar files. This action will be achieved mainly through the upload class. | The user can add a file by selecting the upload button which is located next to the drop-down menu for grammar files.  Once the user has selected their chosen file in their computer file explorer the readFile() class will be called with the file path as a parameter. | The design has an additional button that allows user to upload their own file. The functionality runs similar to the functionality of the readPGF method of the existing code, with the only difference being that it accesses the user’s file system to upload the file. |
| **Supporting multiple sentences** | This design allows the user to enter multiple sentences through the use of multiple text boxes. When a user wants to add more than the already stipulated number of sentences, they are able to click the button to add another sentence.   * This will add another textbox for sentence input   As a user enters a sentence, a corresponding translation of the sentence will be displayed on the left panel. | The user can input multiple sentences in the form of a paragraph.  The design of this input is clean and simple however it leaves room for user error and additional validation and checks would need to be implemented to ensure that the user’s input is valid.  It cannot include symbols and numbers, empty spaces etc. | Design allows users to enter multiple sentences in one textbox. They can do so through separating the strings by a symbol ‘ ~ ’.  It will be difficult for user to modify sentences once entered because everything lies in one sentence. Suggested to separate the sentences into separate textboxes for easier mapping of sentence to translations. |
| **Deletion of words** | Deletion of words is linked to each sentence inputted by the user. Deletion of all the sentences can be achieved through the delete all button. | Through the paragraph text input, the user can delete words easily as well as edit the sentence for re-translation. | Deleting of words is done through using the backspace/delete key of user keyboard. Unclear whether the use can use other methods of deletion. Other keys may be used for deletion. |
| General drawbacks | It strays from the original interface meaning there would be a need to implement a new design which takes away from the time needed to implement the required functionalities. | While the design is simple, intuitive, and aesthetically pleasing, it strays very far from the original GF minibar application.  This would make the implementation far more complicated and call for editing the pre-existing features which are already functional.  The design also does not allow for text prediction which is a major drawback. | Some of the features on the app have not been fully updated to satisfy the 7 design principles (congested wording) and this is due to trying to keep the design as original as possible.  Implementation of the “~” symbol to separate text can be tricky to implement since a user may want to have the symbol in their translation. |

# Final Design and Justification:

**Justification:**

The final selection of our prototype is a combination of all the individual prototypes that were presented. This is because there were features/functionalities that were better in some prototypes (splitting sentences in design 1/2) and some features that would add unnecessary functionality to the app and would need for us to go beyond our scope. In our critical analysis we focused a lot on the general drawbacks of each prototype and tried to find ways in which we can work around them.

The final prototype is an improved design that does not stray too far from the original but includes designs that implement the requirements of the client. It is a clear and intuitive version of the existing interface that includes all the updated functionality. and adheres to the 7 design principles by ensuring that it is usable, it is standardized (not too far from original), provides feedback, makes use of good affordances (I.e. bin icon to indicate deletion) and makes good use of mapping controls to their actions.

Hellodarling,Iloveyou.SincerelyyoursMom

Hello darling , I love you . Sincerely yours Mom

that cheese is boring

# **Class definitions:**

**PGF class:**

**Definition**

This class is responsible for reading the PGF files which are uploaded by the user and representing the grammar in the specified file as its own instance of the class. The class handles all the file’s attributes such as the languages, categories, and functions which are handled by the specific grammar file, as well as provides methods to perform calculation which will aid in the translation and text prediction within different grammars.

**Relationships with other classes:**

The PGF class is primarily utilized by the main class. The main class instantiates a PGF object by calling the readFile() method and passing it the file path.

The main class also needs to call the PGF class’s accessor methods to access the relevant PGF file attributes before being able to translate the user’s text or make text predictions. This includes fetching the language, categories, and functions that are handled by the grammar.

Functions:

+ readPGF(String): PGF

+ readPGF(InputStream): PGF

+ getAbstractName(): String

+ getLanguages(): Map<String, Concr>

+ getCategories(): List<String>

+ getStartCat(): String

+ getFunctions(): List<String>

+ getFunctionsByCat(String): List<String>

+ getFunctionType(String): Type

+ getFunctionProb(String): double

+ generateAll(String): Iterable<ExprProb>

+ compute(Expr): Expr

+ inferExpr(Expr): TypedExpr

+ checkExpr(Expr, Type): Expr

+ graphvizAbstractTree(Expr): String

+ readPGF(String): PGF

+ readPGF(InputStream): PGF

Variables:

- pool : Pool

- ref : long

**Concr Class:**

**Definition**

The methods that are in the class are useful for obtaining various information with regards to a category in the pgf class. It is used by several classes in the PGF file. One of the main uses of this class is the getName method that is used in the main method. More on the method will be discussed in class member functions.

**Relationship with other classes.**

An instance of the Concr class contains the following parameters: PGF gr and long ref. The instance is primary used to get the languages that are associated with a particular category. Instances of this class are created in other

Functions

+ getName(): String

+ parse(String startCat, String s) : Iterable <ExprProb>

+parseWithHeuristics(String startCat,String s, double heuristics, Map< String,LiteralCallback > callbacks): Iterable<ExprProb>

+ complete(String startCat, String s, String prefix) Iterable<TokenProb>

+ lookupSentence(String startCat, String s) : Iterable<ExprProb>

+ linearizeAll(Expr expr): Iterable<String>

+ tabularLinearize(Expr expr): Map<String, String>

+ bracketedLinearize(Expr expr): Object[]

+ lookupMorpho(String sentence): List<MorphoAnalysis>

+ fullFormLexicon(): Iterable<FullFormEntry>

+ lookupWordPrefix(String prefix): Iterable<FullFormEntry>

+ hasLinearization(String fun): boolean

+ graphvizParseTree(Expr expr): String

+ getPrintName(String id): String

+ load(String path);

+load(InputStreamstream)  
+ unload();

Variables

* PGF gr
* long ref

**PGFUpload class**

**Definition**

The pgfupload class inherits from the PGF class and has the additional functionality of checking that the file that the user uploads is correct [correctness being a measure of the using the correct file format]

**Relationship with other classes**.

The class is a subtype of the PGF class and thus all subtypes of a type will have the same behaviour and properties of the type that they are inheriting from therefore the relationship that the PGFUpload file has are the same as those from the PGF class.

**Functions**

* Extends PGF class
* verifyPGF(String filename): Boolean

# **Class member functions:**

|  |  |  |  |
| --- | --- | --- | --- |
| **PGF Class** | | | |
| **Class Member** | **Description** | **Parameters** | **Return** |
| readPGF | Reads a grammar with the specified file path. | String | PGF |
| readPGF | Reads a grammar from an input stream | InputStream | PGF |
| getAbstractName | Returns the name of the abstract syntax for the grammar. | none | String |
| getLanguages | Returns a map from a name of a concrete syntax to a `Concr` object for the syntax. | none | Map<String, Concr> |
| getCategories | Returns a list of all categories in the grammar. | none | List<String> |
| getStartCat | Returns the name of the start category for the grammar. | none | String |
| getStartCat | Returns the name of the start category for the grammar. | none | List<String> |
| getFunctionsByCat | Returns a list of all functions with a given return category | String | List<String> |
| getFunctionType | Returns the type of the function with the given name | String | Type |
| getFunctionProb | Returns the negative logarithmic probability of the function with the given name | String | double |
| generateAll | Returns an inerrable over the set of all expressions in the given category, enumerated in decreasing probability order. | String | Iterable<ExprProb> |
| Compute | Normalizes an expression to its normal form by using the 'def' rules in the grammar. | Expr | Expr |
| inferExpr | Takes an expression and returns a refined version of the expression together with its type | Expr | TypedExpr |
| checkExpr | Takes an expression and checks it against a type. The returned expression is a possibly refined version of the original | Expr,Type | Expr |
| graphvizAbstractTree | Generates a Graphviz representation of the abstract syntax tree for the given expression. | Expr | String |

|  |  |  |  |
| --- | --- | --- | --- |
| **Concr Class** | | | |
| **Class Member** | **Description** | **Parameters** | **Return** |
| getName | Returns the string of a | none | String |
| parse |  | String, String | Parser |
| parsewithHeuristics | Parse a string with a given start category and specific setup of other parameters. | String, String, double, Map<String,LiteralCallback> | Parser |
| complete |  | String, String, String | Completer |
| lookupSentence |  | String, String, | SentenceExtractor |
| linearize | Computes the linearization of the abstract expression | Expr | String |
| linearizeAll | Computes all linearizations of the abstract expression and returns an iterator over the alternatives. | Expr | Iterable<String> |
| tabularLinearize | Linearizes the expression as an inflection table. | Expr | Map<String,String> |
| brackedtedLinearize | Computes the bracketed string for the linearization of the expression. | Expr | Object[] |
| lookupMorpho | Takes a word form or a multilingual expression and returns a list of its possible analyses according to the lexicon in the grammar | String | List<MorphoAnalysis> |
| fullFormLexicon | Creates an iterable over the full form lexicon in the grammar | none | Lexicon |
| lookupWordPrefix | Returns an iterable enumerating all words in the lexicon  starting with a given prefix. | String | Lexicon |
| hasLinearization | returns true if a given function has linearization in this concrete syntax. | String | boolean |
| graphvizParseTree |  | Expr | String |
| getPrintName | returns the print name for that function or category. | String | String |
| load | load the current syntax in memory. | String | none |
| load | When the syntax is no longer needed then this method can be used to unload it. | InputStream | none |
| unload | Unloads the syntax when it is no longer needed. | none | none |

|  |  |  |  |
| --- | --- | --- | --- |
| **PGFUpload Class** | | | |
| **Class Member** | **Description** | **Parameters** | **Return** |
| verifyPGF() | This method is used to verify that the file that the user uploads is a pfg file before it is passed on to readPGF() | String | Boolean |

**Class Inheritance**

In the main method, an instance of the PGF class is created and the instance is thus used with several other classes to obtain the information required by the app. Our PGF class acts as a super class to the PGFUpload class which inherits the properties of the PGF class.

# **Scope**

The aim of the software is to provide users with an enhanced version of the Minibar Online application through additional requirements and functionality. A vertical prototype of each of the functionality was generated to provide an in-depth understanding of the app functionality and design. This approach is discussed below:

1. **Offline functionality**
   1. The application is a web app implemented using HTML, CSS and JS. The backend implementation is achieved using Java. To be functional off
2. **The ability to upload user’s own grammar file** 
   1. To fulfil this functionality, the user will be able to press the upload button which will direct them to the local device storage where they will be prompted to select a grammar file to upload onto the system.
   2. The main functions to be utilized for this functionality include the above defined subclass inheriting from the PGF class. This class includes the necessary methods required to upload a user’s own file.
3. **Multiple Sentences** 
   1. To achieve this functionality, a user will be able to input multiple sentences using multiple text boxes. The standard number of sentences to be entered now is three but the user can click a button (**Click to add a sentence**) allowing them to input another sentence.
4. **Intuitive Deletion of words**
   1. To make deletion easier and intuitive, each sentence input box has a delete button/icon tied to it. This means that the user will be able to delete a specific sentence by clicking on the delete icon. This prototype will make use of the already defined function of the delete/backspace button (delete button on the Minibar online with an x on it) in the implementation of deletion. This means that each textbox representing each of the user’s inputted sentences will have an associated delete button.
5. **Collapsible Panels**
   1. This functionality will allow the user to modify their view of the translated sentences. This includes the use of two methods, one for rendering the translation of a single sentence and one for multiple sentences. The multiple sentences translation method will receive an array of sentences to be translated as a parameter. Each of these sentences will be parsed and translated and then assigned to each of their panels as per stipulated by the prototype design about.