Software Design Document

Decide

**Table of contents**

1. Introduction

1.1 Purpose

1.2 Document Conventions

1.3 Intended Audience and Reading Suggestions

1.4 Scope

1.5 Definitions, acronyms, and abbreviations

1.6 References

1.7 Overview

2. Overall description

2.1 Product Perspective

2.1 Product Functions

2.3 User Characteristics

2.4 Constraints

2.5 Assumptions and dependencies

2.6 Apportioning of requirements

3. Specific requirements

3.1.1 User Interfaces

3.1.2 Hardware Interfaces

3.1.3 Software Interfaces

3.2 Functional requirements

3.3 Performance requirements

3.4 Design constraints

3.5 Software system attributes

4. Release plane

Appendix I

Appendix II

1. Introduction

The System Design Document (SDD) describes how the functional and nonfunctional requirements recorded in the Requirements Document, the preliminary user-oriented functional design recorded in the High Level Technical Design Concept/Alternatives document, and the preliminary data design documented in the Logical Data Model (LDM) transform into more technical system design specifications from which the system is built. The SDD documents the high-level system design and the low-level detailed design specifications.

The SDD describes design goals and considerations, provides a high-level overview of the system architecture, and describes the data design associated with the system, as well as the human-machine interface and operational scenarios. The high-level system design is further decomposed into low-level detailed design specifications for each system component, including hardware, internal communications, software, system integrity controls, and external interfaces. The high-level system design serves as primary input to the Preliminary Design Review (PDR). The low-level detailed design serves as input to the Detailed Design Review (DDR).

1.1 Purpose

The SDD documents and tracks the necessary information required to effectively define architecture and system design in order to give the development team guidance on the architecture of the system to be developed. Design documents are incrementally and iteratively produced during the system development life cycle, based on the particular circumstances of the information technology (IT) project and the system development methodology used for developing the system. Its intended audience is the project manager, project team, and development team. Some portions of this document, such as the user interface (UI), may be shared with the client/user, and other stakeholders whose input/approval into the UI is needed.

1.2 Scope

“Decide” is a desktop application that aims to help users specially food business owners to analyze the customer reviews quickly and efficiently through computer without any human interference.

**Decide** is a tool aims to help users to build data driven business by facilitating the analyzing of data through fast accurate classification and visualization.

1.3 Reference Material

IEEE Template for Software Design Documents:

<https://sovannarith.files.wordpress.com/2012/07/sdd_template.pdf>

1.4 Definitions and Acronyms

|  |  |
| --- | --- |
| **Work** | **Definition** |
| NLP | Natural language processing |
| Sentiment-analysis | Process of classification of texts to different rating classes |
| Training | Core process to create a learning model |
| Pre-trained | Any model that is not loaded as default in official version |
| Prediction | Process of using an already created model to classify input to desired classes |
| NLTK | Open-source library for natural language processing |
| Prediction set | Set of input reviews that are desired to be classified using a trained model |
| SRS | Software Requirements Specification, documents that contains the needed requirements by the customer. |
| Use Case | Diagram that consists of a series of actions that a user must initiate with the system to carry out some useful work and to achieve his/her goal. |
| DEP | Dependency |
| RAT | Rational |
| DESC | Description |

**2 System overview**

Decide is a software that enable users to classify their own reviews to a 1:5 classes, it provides users with the ability to choose among different classifying models which could better describe their own dataset, Moreover it get users’ understanding enhanced by set of available charts for both input and output data.

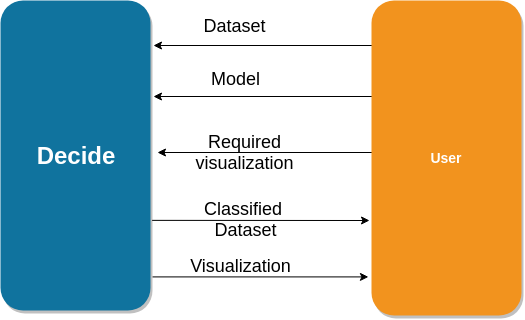
Decide has single external actor who is the user who can input

* Data to be classified
* Model to classify data according to or use the default model

and take as an output:

* Classified data
* Set of charts that describe desired data

**Context diagram**

****

In order to prepare the desired software the following phases and components must be developed:

* Train a Model that can classify text that is fed by users.
* Pre training components to extract data from input file and put it in suitable format.
* Interface in order for users to deal with software through.
* Prediction component that can use the parameters of the model them outputs the results.
* Visualization components.

One important constraint is the modularity of the classifying model.

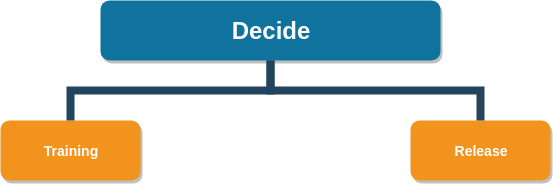
Design is describing 2 phases:

* Training phase:
  + Where we describe the needed components to create the default model.
* Release phase:
  + Where we describe the permanent design and architecture of the software that the user will normally deal with.

**3 System Architecture**

The software has two main component

1. Training component
2. Release component







|  |  |
| --- | --- |
| 1. Training component: | |
| DESC | It’s the component that is responsible for producing the predicting model with the intended accuracy on the desired distribution  (Food reviews in Decide software)  User has no access to this module |
| Input | Dataset |
| Output | Model parameters |
| DEP | None |

|  |  |
| --- | --- |
| 1. Release component: | |
| DESC | It’s the component that is responsible for producing the prediction results for user.  Represent the user side of the software, User can access this module. |
| Input | Dataset.  default or Pre-trained model . |
| Output | Predictions.  Visualizations. |
| DEP | Existence of model.  Existence of a dataset. |

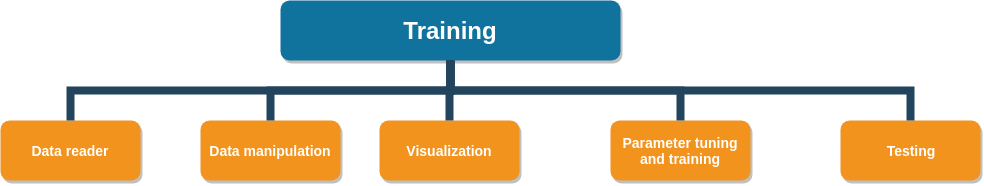
**Overall description**

Training component is used once by the development team to provide the release component with the model, which implies that developers are not permanent users.

There will be no need for the user to train the model each time a prediction is needed.

**Decomposition architecture**

For each of the main two component there is a structure break down to several simpler component that provide one and only one task.



1. Training component

|  |  |
| --- | --- |
| 1.1. Data reader | |
| DESC | It’s responsible for reading the input dataset with a specific extension from the desired path |
| Input | Address Path |
| Output | Object that represent the dataset |
| DEP | Existence of file in the input path |

|  |  |
| --- | --- |
| 1.2. Data manipulation | |
| DESC | It’s responsible for extracting text column and its label columns from the read dataset.  Perform some statistics.  Put data in suitable format for further processing. |
| Input | Object that represent a file |
| Output | Data in certain desire format |
| DEP | File object |

|  |  |
| --- | --- |
| 1.3. Visualization | |
| DESC | It’s responsible for producing charts for data before and after training.  This phase is not a must but it enhance the developer understanding . |
| Input | Dataset  File objects |
| Output | Charts |
| DEP | Imported libraries |

|  |  |
| --- | --- |
| 1.4. Parameter tuning and training | |
| DESC | It’s responsible for producing the intended model with the desired accuracy through different algorithms and by tuning the algorithm’s parameters.  Parameter tuning is based on feature extracted by the aids of visualization. |
| Input | Dataset with suitable format |
| Output | Set of weights to be input in a model |
| DEP | Desired accuracy  Available computation resources |

|  |  |
| --- | --- |
| 1.5. Testing | |
| DESC | It’s responsible for ensuring the **generalization** concept of the trained model on different example.  Act as the meter for reaching the intended accuracy |
| Input | Dataset  Set of weights(trained model) |
| Output | Predictions  Accuracy result |
| DEP | Parameter tuning and training |

1. Release Component



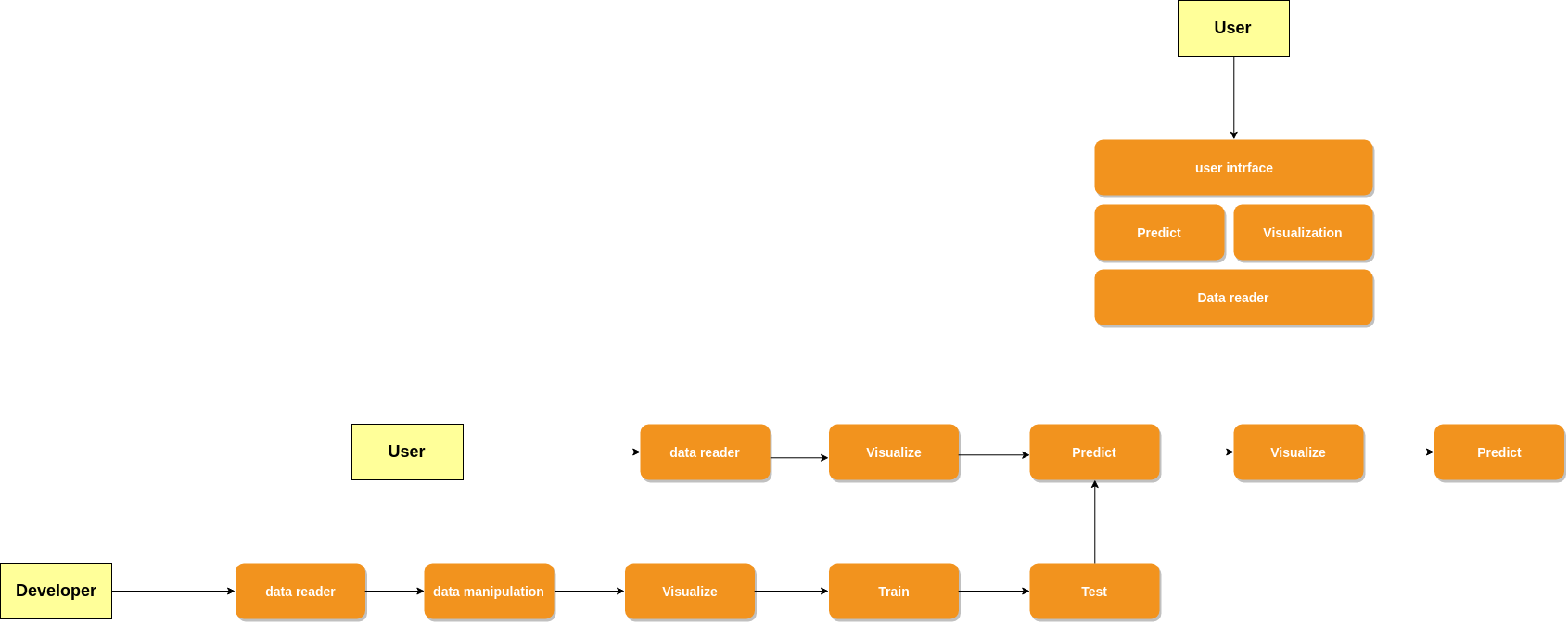
|  |  |
| --- | --- |
| 2.1. Data reader | |
| DESC | It’s responsible for reading the input dataset with a specific extension from the desire path |
| Input | Address Path |
| Output | Object that represent the dataset |
| DEP | Existence of file in the input path |

|  |  |
| --- | --- |
| 2.2. Visualization | |
| DESC | It’s responsible for producing charts for data before and after prediction.  This phase is not a must but it enhance the user understanding. |
| Input | Dataset  File objects |
| Output | Charts |
| DEP | Imported libraries |

|  |  |
| --- | --- |
| 2.3. Predict | |
| DESC | It’s responsible for producing the classification results. |
| Input | Dataset of texts |
| Output | Labeled dataset |
| DEP | Default model  Pre-trained model |

|  |  |
| --- | --- |
| 2.4. User Interface | |
| DESC | It’s responsible for facilitating the user interaction with all components  It has 2 modes:  CMD mode: Where user start the program from CMD.  GUI mode:  Where user can deal with buttons and check buttons for facility. |
| Input | Certain command in case of CMD model contains path and required visualization.  User clicks in case of GUI |
| Output | Order for lower components to operate with given inputs, |
| DEP | None |

**Decide** Architecture could be described by two architectural patterns :



**Layered architecture**

Where at the base layer the data reader where all other processes depend on this stage.

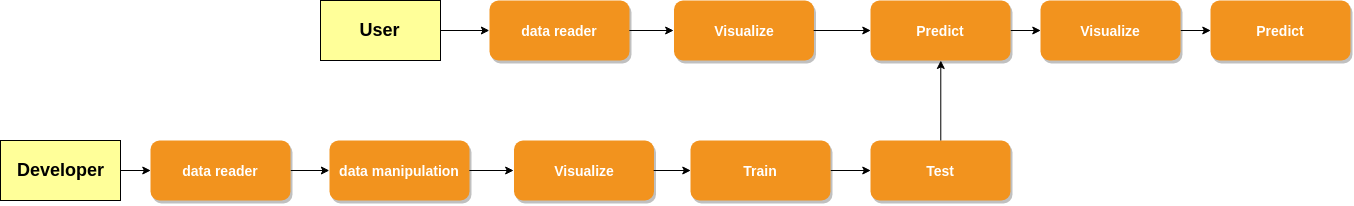
Second layer contains both the optional visualization techniques and the predicting module which gives the output.

The upper layer represent the user interface where user can control lower layers through.

**Data flow: Batch sequence**

Where the flow of data is describe as a sequence of changes that are made on data









The Bottom sequence represent the Training component which is used once for developing the model and no need for user to deal with that module or operate it each classification process.

An external pre-trained model could also be supplied by user to predict module instead of the default loaded one

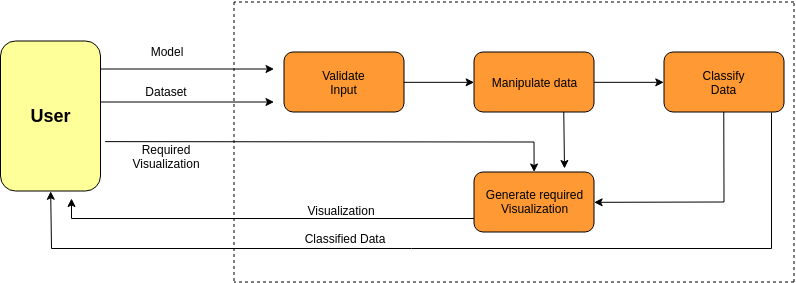
Data flow diagram.

Note: Developers are not only the official developers of the software, Developers can be any one who trained or developed a model that can be used in the system.

**Data Flow Diagram**

User input the model or can use the default loaded model, User also inputs the dataset and select the desired visualization for both input and output after data classification process.

User receives classified data and chosen visualization

****

**Data Design:**

**Data Description:**

- we use two types of data inputs **model** and **reviews**

**model**:

- is a serialized object of machine learning model

- its format must be \*.pkl

- it is not mandatory to use the default model provided with the software,

instead users can use pre-trained models of their choosing

**reviews**:

- is a file that contains reviews which will be classified

- its format must be \*.csv

- all reviews must be labeled with "text" other data will be ignored

- the program will read the file and transform its content to a data-frame

represents all the reviews

**Design Rationale**

The main goal of the designed architectures is to achieve the non-functional requirements in Software Requirement Specification

Layered Architecture : It efficiently describe how components rely on each other, starting from low core components up to the user-Interface

It supports the required non-functional requirement **QR1** which require the ease of usage if User Interface, It apparently describes how the UI component communicate with all lower components to provide user with few choices that reduce the overhead of using the software

It also supports **QR7( Portability )** as this architecture could easily accept lower layers as virtual machine or different operating system layers, so it’s descriptive architecture for such needs.

Data Flow – Batch Sequence : The Architecture is very convenient to learning algorithms which represent a strong candidate for implementing the **Train** component.

It provides high level of modularity where each component could easily be replaced which satisfies **QR6 ( Modularity )** which is crucial for such system in order to enable the user to reuse pre-trained models other than the one loaded by default in the software.

The architecture has no definite length so it can be extended however the user needs are, This supports **QR5 ( Extendability ),** as an example visualization could be done once twice of zero times according to user need.

Also extra phases could be added.

Batch Sequence provides simple efficient method to calculate response time based on the details of implementation of each module or stage which facilitates the achievement of **QR2(Response Time)**

and monitor the **Design constraints.**

**Collectively** these two architectures provides the ability to track the efficiency of each module in the system and replace it easily.

It is based on simple independent components which support the reuse of ready open-source components which can easily fit independently from the system that it will work in as long as suitable interface will be designed.

**Other Candidates**

**Client-Server :** No need for two sides architecture as all computation is performed on the client side and the software owners has no cloud services to provide.

**Main Return:** It will add nothing to the implementation more than the overhead of calling and returning form the single thread which will put an extra issue on achieving the Response time constraint.

It could make the software more testable as it will be consisted of set of independent functions, on the other hand batch sequence could provide the same feature without the time overhead on the main thread.

**4. Component Design**

As stated earlier in the decomposition architecture, system components can be broadly seen as a Training component and a Testing component, each of which consists from smaller components. The goal of this section is to provide a summary of each algorithm for each component listed in the decomposition architecture in pseudocode.

**Training component**:

All components will be developed in Python 3.

**Data Reader**:

This component takes a csv file directory, reads the file, then stores the data in a “Data frame” making it ready for further analysis. The csv file is assumed to be the file with the labeled data that will be used in the future to help train the model.

**Data Manipulation**:

This component functions by taking the DataFrame, extracts the reviews, normalize its text, and returns it for further analysis.

**Data Visualization**:

In such component we take a Dataframe from the “**Data Manipulation**” component, generate histograms and box and whisker plot to see whether the text length per review is a distinctive feature between different classes.

**Training**:

This component is given the normalized text from the “**Data Manipulation**” component plus all good features (if existed) found from the “**Visualization**” component, inputs is the the model (Support Vector Classifier) to train it.

**Testing** :

Once the model finishes training, it is tested with a data that we already know its expected outcome, and compare it with the actual output from the model, generate a confusion matrix, and finally generate a classification report to find the overall precision of the model.

**Tuning**:

Such component might not be used at all, but in the case of a bad precision from the **SVC** model we use these components to tune its (gamma and c) parameters using gridSearch, once we find the best parameters we re-ues the training and testing components until a good precision is achieved.

**Testing component**:

**Data Reader**:

This component takes a csv file directory, reads the file, then stores the data in a “Data frame” making it ready for further analysis. The csv file is assumed contain the data that need to be classified and also the data should be under a label called “text”.

**Prediction**:

This component takes a Dataframe from the “**Data Reader**” component, feed it into the pre-trained model, and save the output in a new csv file contains the reviews, and its corresponding stars (rating).

**Data Visualization**:

In such component we take a Dataframe from the “**Prediction**” component, and generate whatever (if any) visualizations the user has requested.

**5. Human Interface Design**

**5.1 Overview of User Interface**

When a user need to classify some reviews then he should possess 2 things, First: The classifying model (We provide a default one) of their choosing, Second: Reviews should be grouped in a single csv file and should also be under a label called “**text**”.

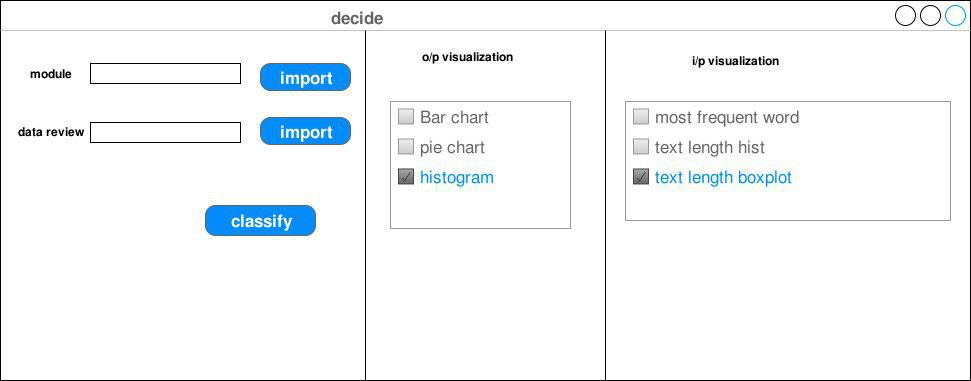
Users can click on the “**Import**” buttons in the interface to specify the location of both the model and the dataset.

If users desire they can specify some visualizatoins to be given out with the classified data. Visualizations can be given to describe the input data or the output one or both.

Then to finalize the process the user hit the “**Classify**” button to start processing.

“**Decide**” alerts the user upon finishing.

**5.2 Screen Images**

****

**5.3 Screen Objects and Actions**

Screen objects can be broadly classified into Buttons, Checkboxes, some guiding labels, and textboxes.

**Buttons**:

**Import**: either one of the import buttons when clicked opens a separate window from which users can select the required data.

**Classify**: When clicked this button activates a background script that handles data validation, manipulation, and classification.

**Checkboxes**:

All check boxes have the same end-goal which is to specify which kind of visualizations (if any) does a user requires.

**Labels**:

They are used to help users understand how to use the software in a quick and easy way.

**Textboxes**:

They provide a place to present an input’s directory specified by the user.

**6. Requirement Matrix**

|  |  |  |  |
| --- | --- | --- | --- |
| FR ID | FR name | Component ID | Component name |
| FR1 | Import Reviews | 2.1 | Data reader. |
| FR2 | Import Model | 1  2.3 | Train component.  Predict. |
| FR3 | Choose visualization tool | 2.2 | Visualization. |
| FR4 | Classify data | 2.3 | Predict. |