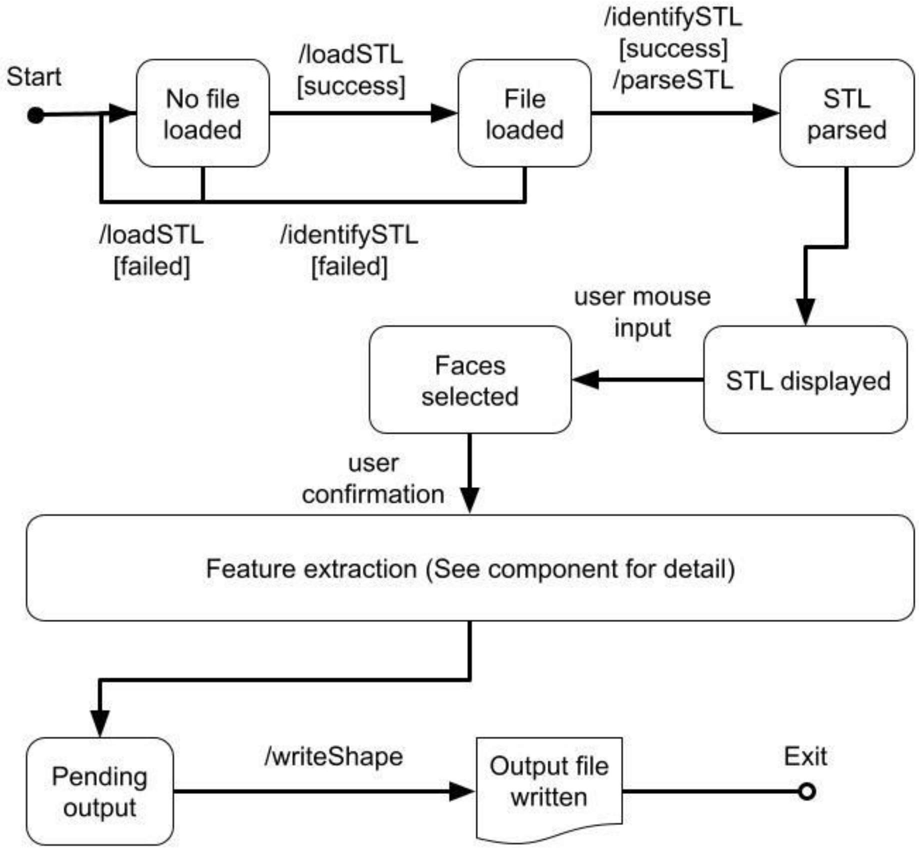
Component Design

Below is a list of components and their respective owners. Additionally a brief description of the component and their corresponding test is provided for each item.



Component: Geometry Input / Output (Calvin Boyle)

- *loadSTL*
    - Description:

      A very basic function to open and validate the file integrity of a .STL file using fstream. It is kept as a separate function in case we desire to load multiple or repeatedly load single files in the future.
    - Input: string filename
    - Output: fstream object
    - Test:
    -
        - Load a known good ASCII STL file
        - Load a known good binary STL file
        - Load a known bad STL file, indicate error

- *identifySTL*

- ◆ Description:

  Determines the STL type. The possible types are Binary or ASCII. The determination is made by matching the expected identifiers in the file headings. The file type will determine the parser behavior. This function will report an error if the file is not an acceptable STL file.
- ◆ Input: fstream object
- ◆ Output: flag variable indicating STL type
- ◆ Test:
  - Identify a known good ASCII STL file
  - Identify a known good binary STL file
  - Identify a known bad STL file, indicate error

➔ *parseSTL*

- ◆ Description

  Translates the STL geometry format into a vector object as a list of vertex locations. Behavior depends on the flaggin of *identifySTL,* though the output will always be of the same format. The output of this function is used for all later visualization and feature extraction components.
- ◆ Input: fstream object & type flag
- ◆ Output: parsed stl data in vector format
- ◆ Test:
  - Translate a known good ASCII STL file
  - Translate a known good binary STL file
  - Translate a empty stl file, indicate error

➔ *writeShape*

- ◆ Description:

  Converts the extracted feature geometry into an ASCII formatted text file that can be read by the Matlab end-user. If multiple shapes are to be extracted, the text file will store all shapes in the same file and delineate the geometry data with named identifiers. The formatting and Matlab processing of this output file is to be developed alongside this program.
- ◆ Input: selected vertex list in vector form
- ◆ Output: .txt file of translated shapes
- ◆ Test:
  - Write a known good shape file
  - Write a known good multi shape file
  - Write a known empty shape file, indicate error

Component: UI (Jae seok Oh)

➔ *Display, Select and Convert*

- ◆ Description:

  Draws the STL data in 3D and lets the user choose the desired faces to extract. When a mouse cursor hovers on the surface of the object, it will highlight all the polygons that have the same normal vectors. The user will be able to move/rotate/zoom the model with mouse and/or keyboard. After the selection is complete the function shows extracted 2-D view point and gives the option to store it into other file formats (.txt, .log, etc.). The GUI will have a button “Convert” available when selection is complete.

- Input: Parsed STL (ASCII / Binary) data along with the calculated normal for every polygon.
- Output: 2-D view of extracted planes and corresponding extracted files.
- Test:
    - Load a random STL file and compare with the actual CAD model.
    - Visual Inspection of rotation / translation / Zoom.
    - Check the normal vectors of the extracted plane and validate. Returns true if all the vectors are identical.
    - Convert the final selection, and plot the saved file (e.g) .txt) and check with corresponding view on CAD model.

Component: Feature Extraction (Kyshalee)

- Calculate triangle normal
    - Description:

      Calculates the normal from the plane of the polygon selected in “Select Face for projection.” The normal can be obtained by the cross product between two vectors. One vector is defined by the subtraction of vertex 1 and vertex 2. The second is defined by the subtraction of vertex 1 and vertex 3. This function is to be called for each surface selection.
    - Input: Data type “vector” that contains the selected surface vertices
    - Output: Data type “vector” contains the unit vector of the normal of the selected triangle.
    - Test:
        - Select a triangle on a part’s top view
        - Select a triangle on a part’s side view, compare top and side view and make sure the relative angle of the surface is expressed by the changes in normal value.
        - Select a triangle on a part’s bottom view, compare the top and bottom view and make sure they are opposite.
- Compare normals
    - Description:

      Iterates through all the triangles in the STL file, and compares the normal of each triangle to the normal specified in “Calculate triangle normal”. All triangles found to have a normal less than **ε** are considered to be coplanar. All coplanar triangles are stored in a vector of vectors containing each triangle’s vertices.   This function is to be called for each surface selection.

  - Input: Data type “vector” contains the unit vector of the normal of the selected triangle.
  - Output: Data type “vector of vectors” which contains every coplanar triangle as a vector of its vertices.
  - Test:
    - Draw all coplanar triangles identified to a selected triangle on a part’s top view.
    - Draw all coplanar triangles identified to a selected triangle on a part’s side view.
    - Draw all coplanar triangles identified to a selected triangle on a part’s bottom view.
- Get Shape
  - Description:

    Extracts outer edges of all coplanar triangles which describe a surface. Can be achieved by finding all vectors that are not used by more than one triangle on a surface. This function is to be called for each surface selection. The output will describe the outer shape of the parts. This description is used for collision detection, as well as to specify welding path targets.
  - Input: Data type “vector of vectors” which contains every coplanar triangle as a vector of its vertices.
  - Output: Data type “vectors of vectors” containing the (X,Y) coordinates of the selected surface outer edges.
  - Test:
    - Draw all outer edges of the surface of the selected triangle on a part’s top view.
    - Draw all outer edges of the surface of the selected triangle on a part’s side view.
    - Draw all outer edges of the surface of the selected triangle on a part’s bottom view.