“Implementation Generator”

...

Sprint Report #1

Vincent Cote
Jace Riehl
Nathan Tipper

Introduction

- Gets rid of tedious work
- Convenient for developers
- Implementation files are something that every C++ developer needs to make

The Idea

- Generates a .cpp or .cc file based upon the the header file of a class
- All functions with their signatures are generated and the header file is included.
- Default parameter names are given, unless they are given in the header file.
- Default return values are returned from each function.

Sprint #1 Plan

- Research the Code::Blocks API
- Generate a .cpp file based on a .h file
- Have it included in the project
- Register with the UI
- Open the newly created file in Code::Blocks
- Include the header file it was generated from

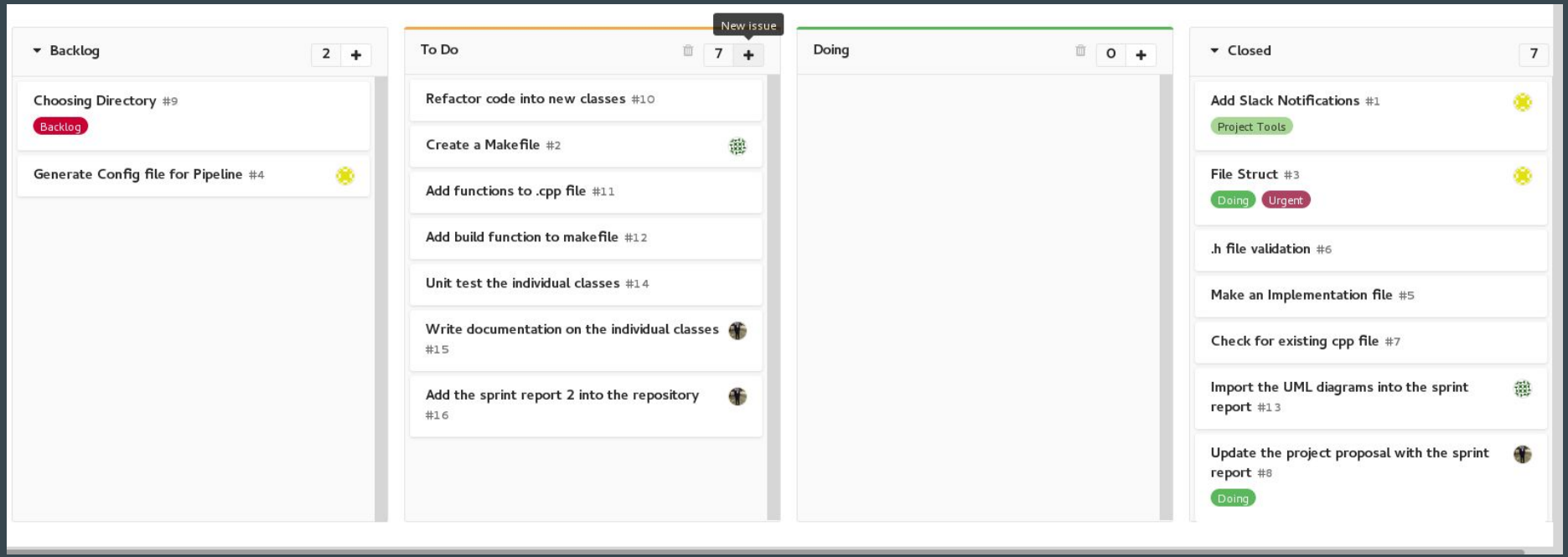
Example:

```
#ifndef CALCULATOR_H
#define CALCULATOR_H


class Calculator
{
    public:
        Calculator();
        virtual ~Calculator();

        int sum(const int, const int) const;
        int mult(const int, const int) const;
        int sub(const int, const int) const;
        int div(const int, const int) const;

    protected:

    private:
};

#endif // CALCULATOR_H
```

```
Calculator::Calculator()
{
    //ctor
}

Calculator::~Calculator()
{
    //dtor
}

int Calculator::sum(const int a, const int b) const {
    return -1;
}

int Calculator::mult(const int a, const int b) const {
    return -1;
}

int Calculator::div(const int a, const int b) const {
    return -1;
}

int Calculator::sub(const int a, const int b) const {
    return -1;
}
```

DEMO

Our board


Backlog
2
Choosing Directory #9
Backlog
Generate Config file for Pipeline #4
New issue
To Do
7
Refactor code into new classes #10
Create a Makefile #2
Add functions to .cpp file #11
Add build function to makefile #12
Unit test the individual classes #14
Write documentation on the individual classes #15
Add the sprint report 2 into the repository #16
Doing
0
Closed
7
Add Slack Notifications #1
Project Tools
File Struct #3
Doing
Urgent
.h file validation #6
Make an Implementation file #5
Check for existing cpp file #7
Import the UML diagrams into the sprint report #13
Update the project proposal with the sprint report #8
Doing

Sprint #1 Process

- Iterative process, trying to get used to the CodeBlocks API
- Started small, and built on that
- Scrapped the idea of “Clean Code” for “Functional Code”

Sprint #1 Process: Things that went well

- Research was quick, which allowed us to get to implementation faster
- Two researchers, one coder
- Able to get implement what our scope was for Sprint #1

Sprint #1 Process: Things to Improve

- Refactor the code so we can start our testing for separate classes
- Have multiple developers contributing to the code
- Make better use of Scrum meetings (better scheduling)

How these improvements will be made?

- Refactor code
- Have a detailed schedule for tasks and meetings

The Next Step

- Refactor and start unit testing
- Validation of the header file and its functions
- Actually writing the signatures of functions (start small)
- Ease of use (Keyboard shortcut)
- Allow for different file structures