SSW567 Final Project

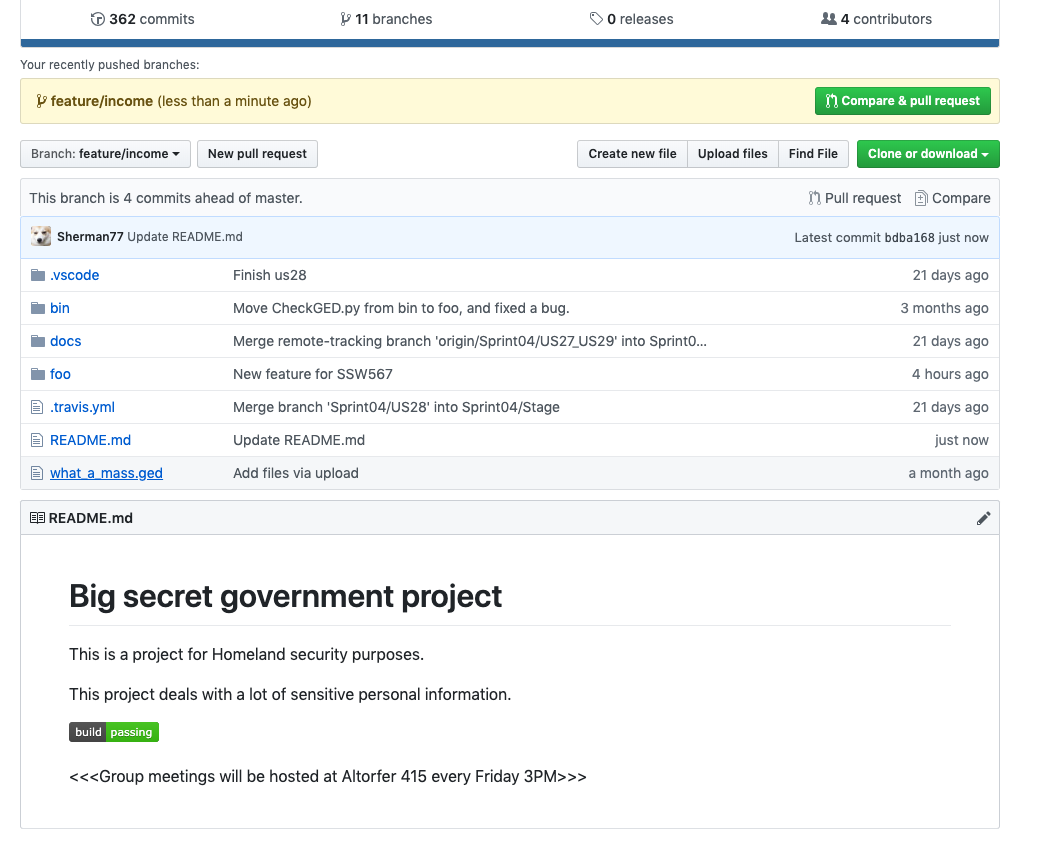
The United States government is starting a new social research project provide better social benefit towards low household income families. We got a government contract to developed a prototype of a software that will help determine benefits based on family household income and family size and children’s age. The requirements has been provided to us by the government, as well as a small portion of sampling data. This prototype will be running and tested on local machine until we determine it is ready to be released to the government.

Introduction

* Scope
  + Configuration management
  + Model-based testing
  + Requirement testing
  + Unit testing
  + Integration testing
  + System testing
  + Usability testing
  + Security testing
  + Performance/Reliability
  + Concurrency
  + Conclusion

Configuration Management

For configuration management, we used Github, all source code and artifacts has been stored and controlled and shared with Github.



Model-based testing

The key function of our software is to determine the proper benefit package based on the household income, number of children and children’s age. Based on the values of different variables we developed a decision table to make sure we cover all possible outcomes of this function.

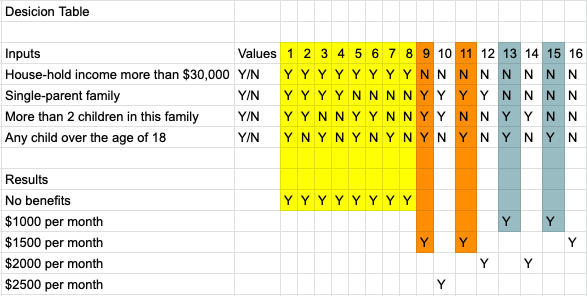
Rules:

1. Only families with annual household income lower than $30,000 are eligible for the benefits;
2. For families with one parent/guardian:
   1. If any of the children is over age 18, families would be eligible for a $1500/month benefit;
   2. If all children are below age 18 and the number of children is less than 3, families would be eligible for a $2000/moth benefit;
   3. If all children are below age 18 and the number of children more or equal 3, families would be eligible for a $2500/moth benefit;
3. For families with more than one parent/guardian:
   1. If any of the children is over age 18, families would be eligible for a $1000/month benefit;
   2. If all children are below age 18 and the number of children is less than 3, families would be eligible for a $1500/moth benefit;
   3. If all children are below age 18 and the number of children more or equal 3, families would be eligible for a $2000/moth benefit;

Input:

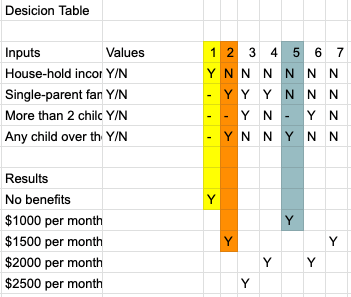
|  |  |
| --- | --- |
| Inputs | values |
| House-hold income more than $30,000 | Y/N |
| Single-parent family | Y/N |
| More than 2 children in this family | Y/N |
| Any child over the age of 18 | Y/N |

Decision Table(Original)



Decision Table(Simplified)

We combined all cases with household income over $30,000 and cases with children’s age over 18 for both single and multiple parent/guardian scenarios



Requirements Testing

All requirements was provided to us from government, and has been carefully reviewed by our development team. From those requirements we developed our user stories.

|  |  |  |
| --- | --- | --- |
| **Story ID** | **Story Name** | **Story Description** |
| **US01** | **Read the file** | **Program can read the GED file properly** |
| **US02** | **Family summary** | **Present a summary of family information for all families in the GED file** |
| **US03** | **Less than 150 years old** | **Death should be less than 150 years after birth for dead people, and current date should be less than 150 years after birth for all living people** |
| **US04** | **No bigamy** | **Marriage should not occur during marriage to another spouse** |
| **US12** | **Parents not too old** | **Mother should be less than 60 years older than her children and father should be less than 80 years older than his children** |
| **US15** | **Fewer than 15 siblings** | **There should be fewer than 15 siblings in a family** |
| **US16** | **Male last names** | **All male members of a family should have the same last name** |
| **US24** | **Unique families by spouses** | **No more than one family with the same spouses by name and the same marriage date should appear in a GEDCOM file** |
| **US32** | **List multiple births** | **List all multiple births in a GEDCOM file** |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

Unit Testing

For each of the user stories we implemented, we have a unit test script developed for it to make sure it works properly

(Don’t forget about negative Testing - Sherman) coverage.py, pylint - add screen shots too.

Integration testing

We adopted agile practice of continuous integration with TravisCI.

(Raw Log - Travis - Sherman)

(Will add Orthogonal Array - Katya)

**System Testing**

Scenarios:

1. John Doe is a single parent with 3 underage children. He is struggling trying to take care of all his children with the low income of $20,000 a year. He came to get assistance from the government and they have to decide what and how much he is eligible for.
2. Mary Poppins decided to retire. Her income is above $30,000 a year; however, she still claims that she is eligible for benefits as she has 3 underage children to take care of.
3. Bonnie and Clyde are a married couple who have 1 child who is 10 years old. They want to move in to a new house; however, they don’t have enough income. Both of their income combined is still below $30,000. They are inquiring to see if they can get any benefits.

SOAP OPERA Scenario:

J. K. Rowling is married to an abusive husband. She has 3 children with him and one day decides to leave him along with her children. She doesn’t have income and does not live with him; however, they have not filed for divorce so she is still considered to be married to him. Even though she receives some benefits now, she claims she deserves more as all the children are underage and she is a single parent with no income.

Negative testing:

1. A single parent family includes 4 children and gets $2,500 in benefits in error.
2. An information is missing about the age of the children.
3. The program is tested on the platform with python 2 interpreter.
4. The file with data is missing.
5. Provided data is in the wrong format.

Please see Unit Testing Section for a more thorough examination of negative scenarios unit testing.

**Operations Testing**

Through operations testing, our development team will make sure that all the specified requirements have been met and that it’ll be used easily within its intended operating

environment.

We will test against minimum configurations that were provided by the client:

Mac X

Windows 10

1GB RAM

We will include testing of backup/restore, maintenance tasks and perioding check of security vulnerabilities.

**Usability Testing**

For usability testing, our team will meet with one of the government representatives who will test our systems for usability. We will follow Concurrent Probing where one of our moderators will ask probing questions while the representative uses the system.

Furthermore, we will employ Retrospective Think Aloud technique where we will ask the representative to retrace their steps after they are done with the session. We will also record the session so it will be easier for us to review it at a later time.

By implementing these techniques we will get a better understanding of the subject’s thought process through and after the session.

We will record such things as how many errors the representative made, total error-rate, how long it took him/her to complete it and overall feedback from the subject.

In addition to aforementioned techniques, we will also ask one or two government representatives to complete the System Usability Scale. If the score will be above 68, then we will consider the usability of the system as good.

We will perform the usability testing before the delivery of the final product.

**Security Testing**

(Add a little about security - Ashish)

Database/file has to be checked for special characters

SQL Injection attacks

**Performance Testing**

Performance requirement from government is this software should be able process a file with at least 50 families within 2 seconds. We will be using decorator to examine the actual execution time.

(Sherman Sam - fix code, manually change ged file)

Stress Testing

System’s behavior at and beyond the system’s limits- increase the size of the file to be read to observe the point where the system takes longer than 2 seconds to respond.

Endurance Testing

Continuous Expected Load - keep the system going for 100 cycles.

Additionally, we will generate a steady state workload where we will increase the file size until we reach the expected load and test it continuously under that expected file size to see how the system is performing; hence testing the overall stability of the system.

**Testing Progress**

defect status: total open, new defects discovered that week, closed that week

historical data for last few releases

Your ship criteria is:

Reliability:

* <= 2 failures per day expected in operation

System Test Complete:

* >= 98% tests executed
* >=99% tests passed
* <= 5 open defects

System testing Status:

|  |  |  |  |
| --- | --- | --- | --- |
| **Week** | **Planned** | **Executed** | **Passed** |
| Week 1 | 25 | 18 | 12 |
| Week 2 | 50 | 76 | 19 |
| Week 3 | 100 | 102 | 75 |
| Week 4 | 200 | 175 | 100 |

(Sherman Sam)

Defects

|  |  |  |
| --- | --- | --- |
| Backlog | New Defects | Closed Defects |
| 0 | 11 | 6 |
| 5 | 22 | 18 |
| 9 | 8 | 12 |
| 5 | 0 | 3 |
| 2 |  |  |

operational profile testing status:

|  |  |  |
| --- | --- | --- |
| **Week** | **Failures per execution day** | **FI/FIO** |
| 1 | 10 | 5 |
| 2 | 12 | 6 |
| 3 | 4 | 2 |
| 4 | 2 | 1 |

(Katya will draw graphs)

Conclusion