**Final Report**

**CMSC 495 6381 Current Trends and Projects in Computer Science (2208)**

**Group Charlie**

James Lisle

Aniebiet Jacob

Mark Tasker

Catherine Wingfield

Sam Shanzhong Yuan

***University of Maryland University College***

**Loan Amortization Calculator**

Revision: 1.0

Date: 10/12/2020

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Revision** | **Date** | **Name(s) of Editor** | **Description** |
| 1.0 | 10/09/2020 | Lisle | Original Document |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

[Overview 4](#_Toc53411944)

[Project Plan 4](#_Toc53411945)

[Project Description 7](#_Toc53411946)

[**Loan Amortization Schedule** 7](#_Toc53411947)

[**Definitions** 8](#_Toc53411948)

[Requirement Specifications 8](#_Toc53411949)

[System Specification 9](#_Toc53411950)

[Test Plan & Results 9](#_Toc53411951)

[**Testing Scenarios** 9](#_Toc53411952)

[**Application Validation** 9](#_Toc53411953)

[**Application Usability** 9](#_Toc53411954)

[**Service Operation Validation** 9](#_Toc53411955)

[**Client Test Cases** 10](#_Toc53411956)

[**Service Test Cases** 34](#_Toc53411957)

[**Testing Tools** 39](#_Toc53411958)

[Resource Requirements 39](#_Toc53411959)

[System Design 40](#_Toc53411960)

[**Class Design** 40](#_Toc53411961)

[**Client** 41](#_Toc53411962)

[**Service** 42](#_Toc53411963)

[**WSDL Specification** 44](#_Toc53411964)

[**Sequence Diagram** 46](#_Toc53411965)

[**Business Logic** 46](#_Toc53411966)

[Alternate Design 48](#_Toc53411967)

[Development History 49](#_Toc53411968)

[Conclusions 51](#_Toc53411969)

[References 52](#_Toc53411970)

[Addendum 53](#_Toc53411971)

[**User Guide** 53](#_Toc53411972)

# Overview

The project under taken by Group Charlie consisted of creating a loan amortization calculator which given three of four variables calculates the forth and optionally emails an amortization schedule to the user’s designated email address. This project offered a slight twist in that the calculation was performed by a WSDL-based SOAP service in the Heroku Cloud environment, and verified the user’s email address using SendGrid’s cloud-based email service. The technologies used to create the system included Java, Axis2, Tomcat, SendGrid, and the Heroku Cloud environment.

The project was challenging but enjoyable with everyone in Group Charlie participating in multiple facets of the endeavor. A user forum was created for general communication amongst group members as well as email usage, weekly conferences were held for status, collaboration, and general consensus on project details. An impromptu training session was also held to demonstrate Axis2 SOAP technology for the project. All project artifacts were stored in a GitHub repository. This group brought a variety of experiences to the table and offered a flexible, detailed, and pleasant atmosphere which created a productive environment to accomplish the goal of delivering a functional loan calculator.

**Group Charlie Contributors:**

|  |  |
| --- | --- |
| Member | Contributions |
| James Lisle | design, development, documentation, testing |
| Aniebiet Jacob | design, development, documentation, testing |
| Mark Tasker | design, development, documentation, testing |
| Catherine Wingfield | design, development, documentation, testing |
| Sam Shanzhong Yuan | design, development, documentation, testing |

# Project Plan

The timeline and deliverables for this project which was to create a Loan Amortization Calculator, (see Project Description and figures 1a-c), was dictated by the course schedule so we adapted our project to it, see table 1. As detailed in table 1, there were four distinct phases and each had a specific set of tasks and deliverables for creating the calculator. The first phase included the creation and finalization of the WSDL file used to define the calculator’s interface for the calling client, the client classes implemented using the MVC pattern, the calculator service that provided the four calculations, the creation of the Heroku based service environment, and the GitHub repository creation and check-in of the first iteration of the code base. The second phase included the connection of the client to the service, the creation of a Ping class for ensuring the availability of the service, the creation of JUnit tests for the client classes, the creation of an IText PDF helper class to create an amortization schedule in PDF format which ended up not being used due to time limitations to fix formatting issues and was replaced using a plain HTML wrapper to generate an amortization schedule, the modification of the WSDL to add email verification functionality to the interface, JUnit tests for the service, SoapUI tests for the service, and updating documentation to reflect the current state of the project. Phase three involved adding the functionality to the client to allow the user to verify their email address using a 2-step verification process, the associated code additions to the service to support 2-step verification and amortization schedule creation as well as adding the SendGrid service interface to the Heroku environment which provided email capabilities to the service. Calculator functional testing, amortization schedule creation testing, email creation testing as well as end to end testing was performed throughout phase three. Phase four included end to end testing as well as final documentation.

|  |  |  |
| --- | --- | --- |
| **Task** | **Date** | **Responsibility** |
| **Phase 1** | | |
| Finalize Loan Calculator  WSDL | 9/16-9/22 | Group Charlie |
| Create Client Project | 9/16-9/22 | Group Charlie |
| Create Client Controller | 9/16-9/22 | Group Charlie |
| Create Client View Class | 9/16-9/22 | Group Charlie |
| Create Client Model Class | 9/16-9/22 | Group Charlie |
| Create Client Stub Classes | 9/16-9/22 | Group Charlie |
| Create Service Project | 9/16-9/22 | Group Charlie |
| Generate Service Classes from WSDL | 9/16-9/22 | Group Charlie |
| Generate Service Classes | 9/16-9/22 | Group Charlie |
| Create LoanCalcEngine Class | 9/16-9/22 | Group Charlie |
| Check-in code to Github | 9/16-9/22 | Group Charlie |
| Create Heroku Environment for Service | 9/16-9/22 | Group Charlie |
| **Phase 2** | | |
| Connect Client to Service | 9/23-9/29 | Group Charlie |
| Review and enhance validation. | 9/23-9/29 | Group Charlie |
| Enhance JUnit tests for unit tests | 9/23-9/29 | Group Charlie |
| Test ping Timer class over a long duration (4 hours) | 9/23-9/29 | Group Charlie |
| Add field for email address & verification code to GUI if amortization table is selected. Include validation that sends email account a verification code to prove user controls account | 9/23-9/29 | Deferred to Phase 3 |
| Review Service Calculation class | 9/23-9/29 | Group Charlie |
| Review and Enhance JUnit tests for unit tests | 9/23-9/29 | Group Charlie |
| Update WSDL with email address. | 9/23-9/29 | Deferred to Phase 3 |
| Service Testing using SoapUI, JMeter | 9/23-9/29 | Group Charlie |
| Create Itext PDF helper class for creating pdf documents | 9/23-9/29 | Group Charlie |
| Updated Project Plan | 9/23-9/29 | Group Charlie |
| Update User Guide | 9/23-9/29 | Group Charlie |
| Update Testing Doc | 9/23-9/29 | Group Charlie |
| Update Technical Doc | 9/23-9/29 | Group Charlie |
| Create Phase 2 Report | 9/23-9/29 | Group Charlie |
| **Phase 3** | | |
| Add 2-step email verification dialog to client | 9/30-10/06 | Group Charlie |
| Update amortization schedule displayed to include loan info with interest paid | 9/30-10/06 | Group Charlie |
| Client Security Review | 9/30-10/06 | Group Charlie |
| Defer final PDF generation modifications until Phase 4 | 9/30-10/06 | Group Charlie |
| Update WSDL with email address and pin verification. | 9/30-10/06 | Group Charlie |
| Generate Web Service artifacts for client and service for 2-step verification for email address. | 9/30-10/6 | Group Charlie |
| Create email class for SendGrid – EmailManager – that allows emails with and without attachements. | 9/30-10/6 | Group Charlie |
| Service Testing of Email Functionality | 9/30-10/6 | Group Charlie |
| Add property file for SendGrid API access | 9/30-10/6 | Group Charlie |
| Add amortization schedule generation to service. | 9/30-10/6 | Group Charlie |
| Service Security Review | 9/30-10/6 | Group Charlie |
| End to End testing | 9/30-10/6 | Group Charlie |
| Assorted bug fixes | 9/30-10/6 | Group Charlie |
| Documentation updates | 9/30-10/6 | Group Charlie |
| **Phase 4** | | |
| System Testing | 10/7-10/13 | Group Charlie |
| System Documentation Completion | 10/7/10/13 | Group Charlie |
|  |  |  |

**Table 1 – Project Timeline for the Loan Amortization Calculator**

# Project Description

The Loan Amortization Calculator provides a user the ability to determine any of the four variables used in the formula shown in figure 1 dependent on the other three variables being provided. The user will also has the option of emailing an amortization schedule showing the number of payments over the life of the loan and the breakdown of interest and principal for each.

where

* A = loan payment per term
* P = loan principal
* r = interest rate per term
* n = total number of payments

**Figure 1a – Amortization Equation**

## **Loan Amortization Schedule**

Loan Amount ......: 15000.00

Loan Payment .....: 1278.96

Interest Rate ....: 4.25

Loan Term ........: 12

Interest Paid ....: 347.55

Month Payment Principal Interest Balance

1 1278.96 1225.84 53.13 13774.16

2 1278.96 1230.18 48.78 12543.98

3 1278.96 1234.54 44.43 11309.45

4 1278.96 1238.91 40.05 10070.54

5 1278.96 1243.30 35.67 8827.24

6 1278.96 1247.70 31.26 7579.54

7 1278.96 1252.12 26.84 6327.43

8 1278.96 1256.55 22.41 5070.87

9 1278.96 1261.00 17.96 3809.87

10 1278.96 1265.47 13.49 2544.40

11 1278.96 1269.95 9.01 1274.45

12 1278.96 1274.45 4.51 0.00

**Figure 1b – Amortization Schedule**

## **Definitions**

* Amortization is a process of reducing a debt by applying regular payments over some time period that typically includes both principle and interest.
* Loan Amount represents the magnitude of the loan.
* Loan Payment represents the amount of money to be paid on a regular basis that will reduce the balance of the loan and includes an interest payment.
* Loan Interest Rate is the amount charged as a percentage of the loan amount over each term.
* Loan Term represents the number of payment periods for the loan.
* Amortization Schedule is a report showing the characteristics of a loan (amount, payment, interest rate, and term) and a table that has the payment of each term of the loan consisting of payment, principal, interest, and the remaining loan balance.

**Figure 1c – Definition of terms**

# Requirement Specifications

|  |  |  |
| --- | --- | --- |
| **Functional Requirements** | | |
| **Requirement #** |  | **Description** |
| 0 | The System will provide a GUI interface. | |
| 1 | The System will provide a cloud-based Loan Calculation Service | |
| 2 | The System will provide a user with a form to enter 3 of the 4 required variables | |
| 3 | The System will validate the data values at entry and calculation. | |
| 4 | The System will provide the user with an error message if validation fails. | |
|  | The System will provide the user with an error message if connectivity fails | |
| 5 | The System will calculate and display the missing variable. | |
| 6 | The System will synchronously communicate with the cloud-based Loan Calculation Service via a SOAP API call over HTTPS | |
| 7 | The System will optionally allow the user to create an amortization schedule. | |
| 8 | The System will receive, calculate, and display the calculated variable and if selected an amortization schedule. | |
| 9 | The GUI will timeout after n seconds of initiating a call to the Loan Amortization Service | |
| 10 | The GUI will ping the service periodically to ensure that it is available and set the <Calculate> button according to availability (enabled/disabled) | |
| 11 | The GUI will optionally email an amortization schedule after a 2-step email validation process. | |
| **Non-Functional Requirements** | | |
| **Requirement #** |  | **Description** |
| 1 | The System will not provide authentication or authorization. | |
| 2 | The System will not be guaranteed to be available. | |
| 3 | The System will not be guaranteed to be scalable. | |
| 4 | The System will not permanently persist any user data. | |

**Table 2 - Requirement Specifications**

# System Specification

**Client Side**

* The Amortization Calculator requires a computer with at least 4 GB memory, multicore CPU(s), and connectivity to the internet.
* The Java Runtime Environment (JRE) version 8 or greater must be installed.
* LoanCalcClient.jar must be installed.

**Server Side**

* Installation of the Loan Calculation Service into the Heroku Cloud environment that include the SendGrid service interface.

# Test Plan & Results

The testing scope for the Loan Amortization Calculator included four test case scenarios with and without the email option selected, table 3, the expected application functionality, performance testing of the service, environmental dependencies (connectivity), JRE version compatibility, and minimum hardware configurations scenarios.

## **Testing Scenarios**

## **Application Validation**

* Test case for calculating loan amount with and without email option selection
* Test case for calculating loan payment with and without email option selection
* Test case for calculating loan interest rate with and without email option selection
* Test case for calculating loan term with and without email option selection
* GUI validates data as numeric and greater than 0

## **Application Usability**

* GUI allows the user to select the variable to calculate and provides the required fields for entering the three required values to complete the calculation
* GUI gives the user the option of emailing an amortization schedule to email with the entered loan scenario after 2-step email verification is satisfied
* GUI reports connectivity issues
* GUI provides an enabled Calculate button for submitting the loan scenario to the Loan Calc Service
* GUI provides a Clear button for resetting all values displayed on the screen
* GUI provides the user with the ability to exit the application

## **Service Operation Validation**

* SOAP request and response artifacts created in SoapUI for service operation testing Performance, Scalability, and Average Round Trip Testing

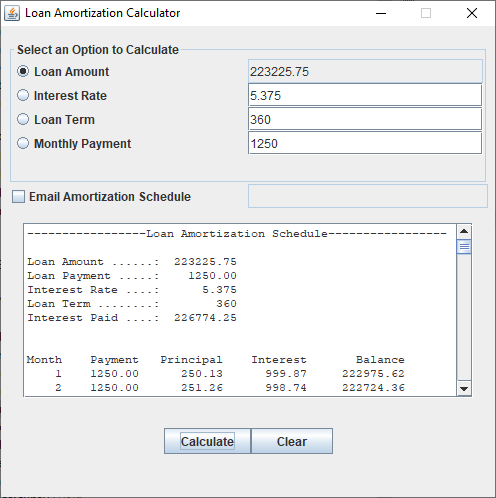
## **Client Test Cases**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **TC #** | **Variables To Be Calculated** | **Loan Amount** | **Loan Payment** | **Loan Rate** | **Loan Term** |
| **TC 1** | **Loan amount** | **223225.75** | **1250.00** | **5.375%** | **360 months** |
| **TC 2** | **Loan payment** | **45000** | **844.06** | **4.75%** | **60 months** |
| **TC 3** | **Loan Rate** | **60000** | **600** | **3.736%** | **120 months** |
| **TC 4** | **Loan Term** | **24950** | **699.50** | **7.25%** | **40 months** |

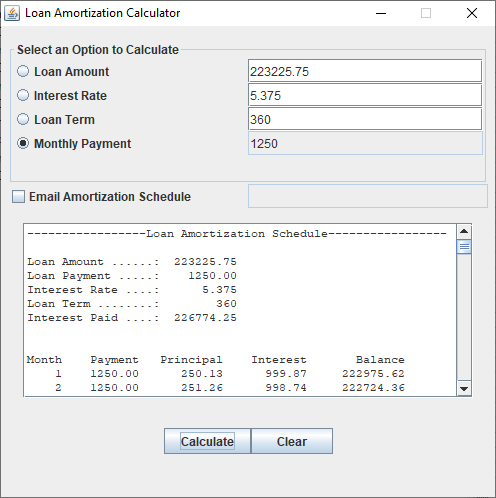
**Table 3 – Test Cases used for both Loan Amortization Client and Service**

**Testing Scenarios – Application Validation**

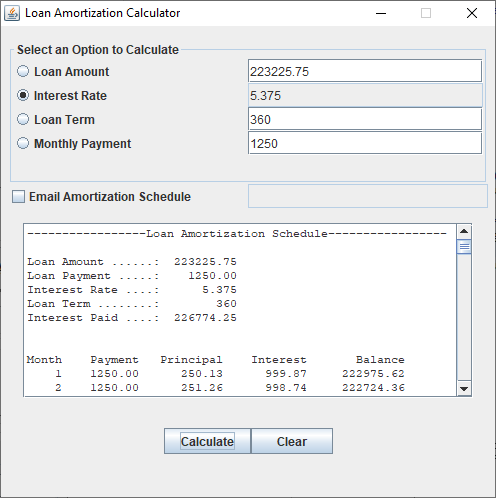
* Utilizing the first test case in Table 3 each loan amortization calculation scenario was performed to show the consistency of the resulting calculation (see figures 2a-d).



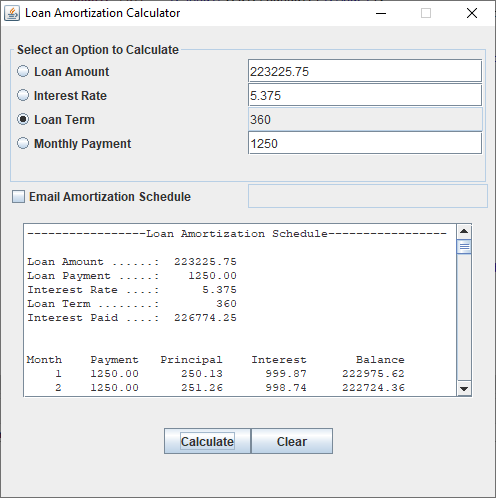
**Figure 2a – Loan Amount Calculation without Email Option**



**Figure 2b – Loan Payment Calculation without Email Option**



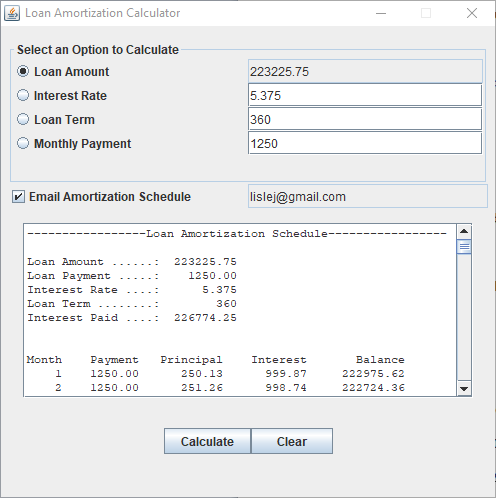
**Figure 2c – Loan Interest Calculation without Email Option**



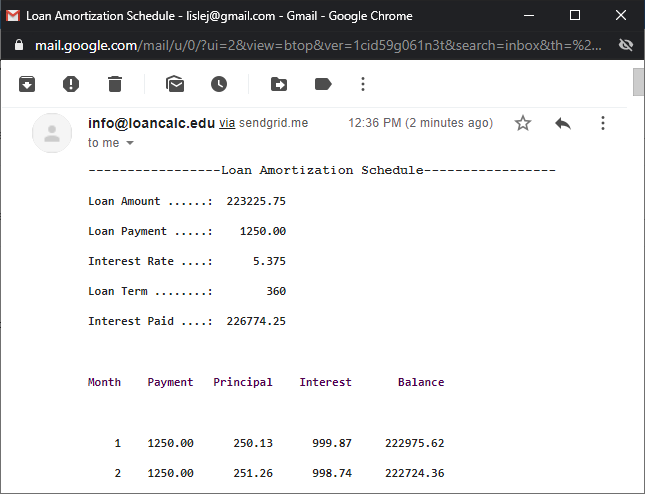
**Figure 2d – Loan Term Calculation without Email Option**

**Testing Scenarios - Application Validation**

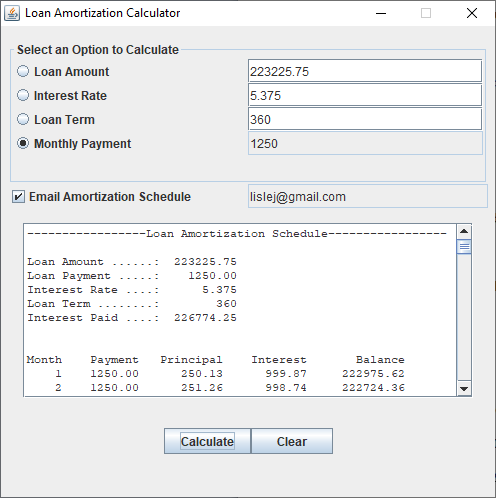
* Utilizing the first test case in Table 3 each loan amortization calculation scenario was performed to show the consistency of the resulting calculation (see figures 3a-h) as well as the amortization schedule on screen and in email.



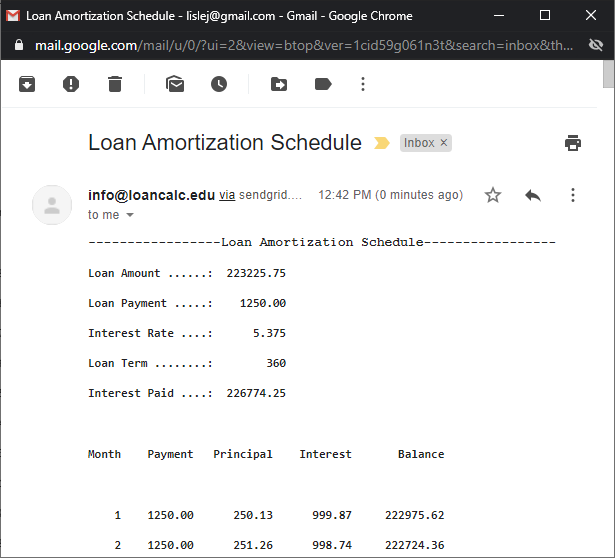
**Figure 3a - Loan Amount Calculation with Email Option**



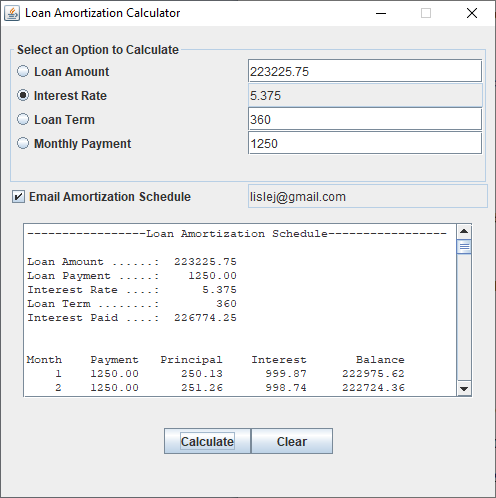
**Figure 3b – Associated Loan Amortization Schedule**



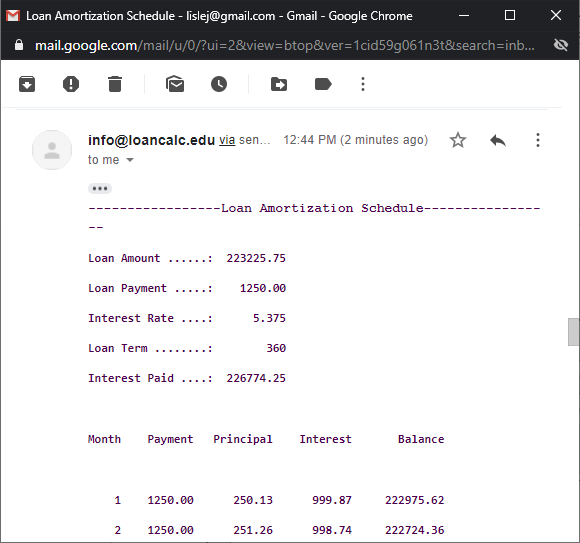
**Figure 3c - Loan Payment Calculation with Email Option**



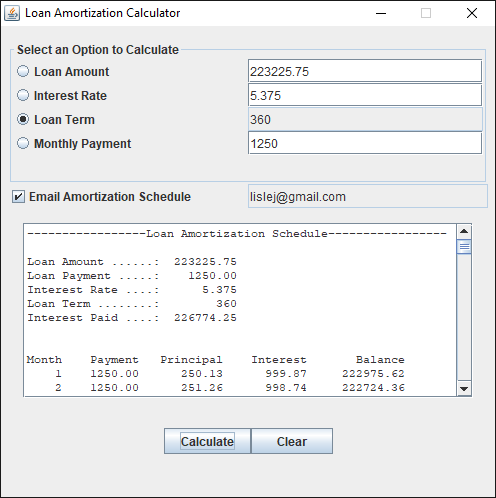
**Figure 3d – Associated Loan Amortization Schedule**



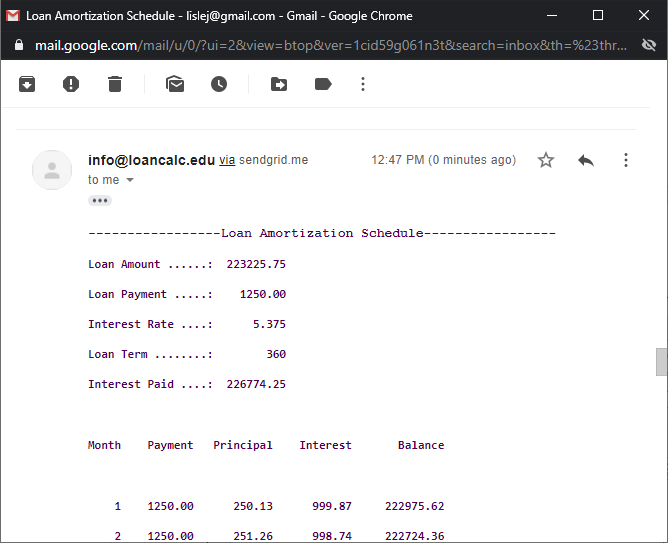
**Figure 3e - Loan Interest Rate Calculation with Email Option**



**Figure 3f - Associated Loan Amortization Schedule**



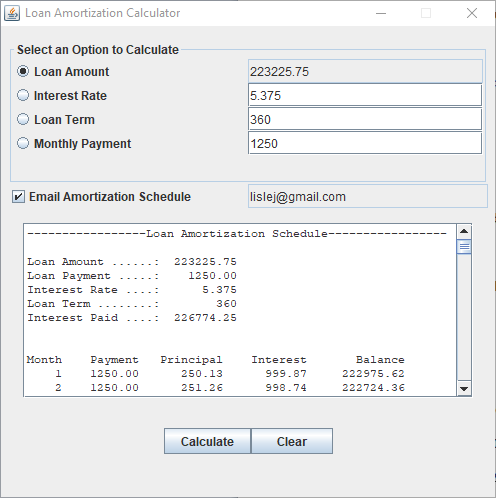
**Figure 3g - Loan Term Calculation with Email Option**



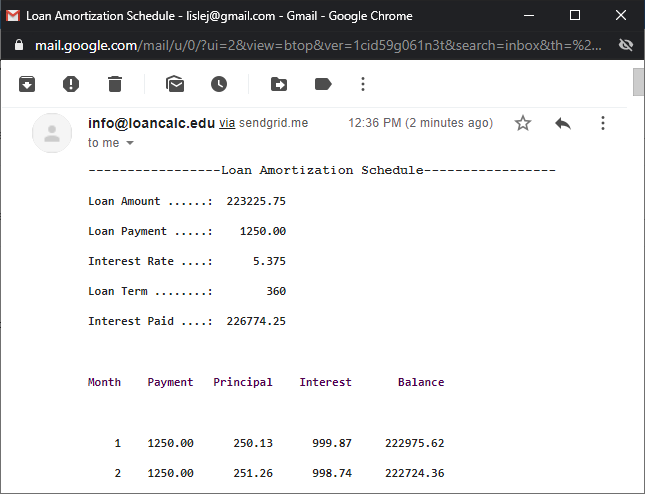
**Figure 3h – Associated Loan Amortization Schedule**

**Testing Scenarios - Application Validation**

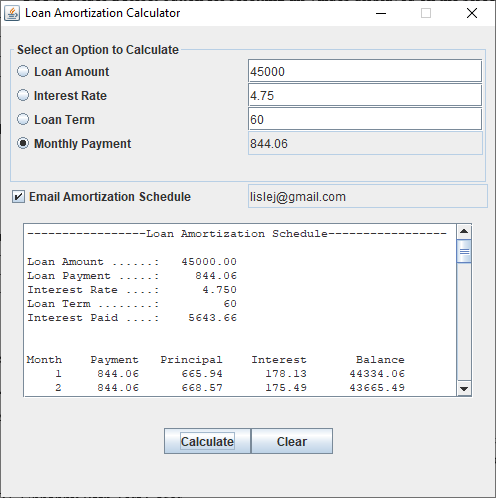
* Utilizing the four test cases in Table 3 each loan amortization calculation scenario was performed with the email option selected to show the consistency of the resulting calculation (see figures 3a-h) as well as the amortization schedule on screen and in email.



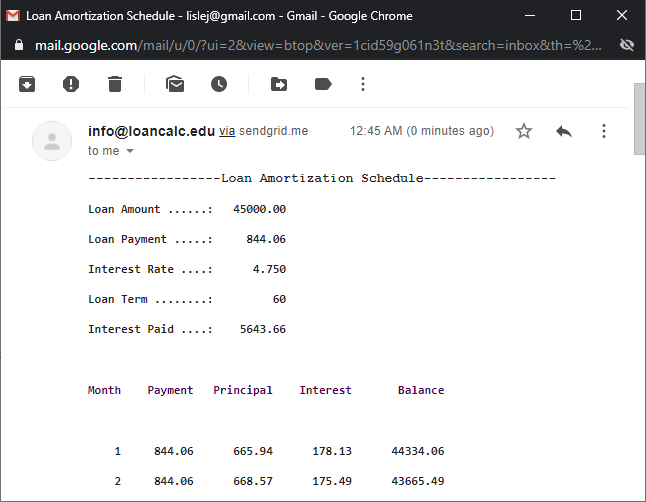
**Figure 4a – Loan Amount Calculation with Email Option**



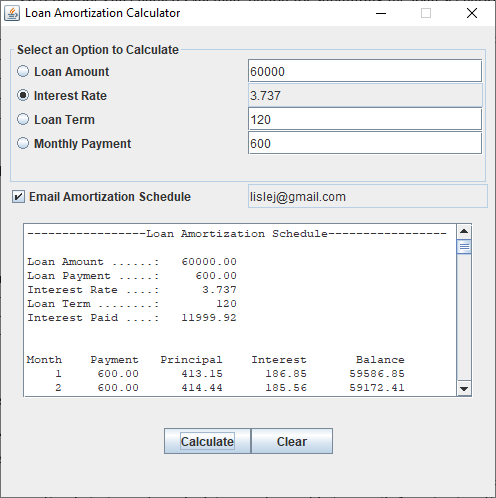
**Figure 4b – Associated Loan Amortization Schedule**



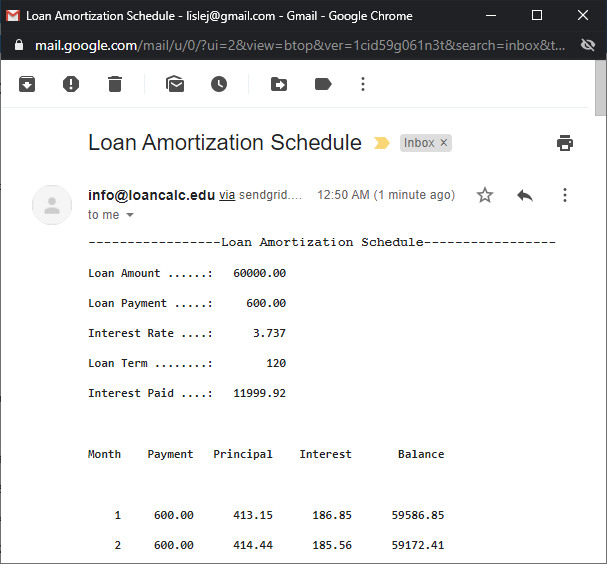
**Figure 4c – Loan Payment Calculation with Email Option**



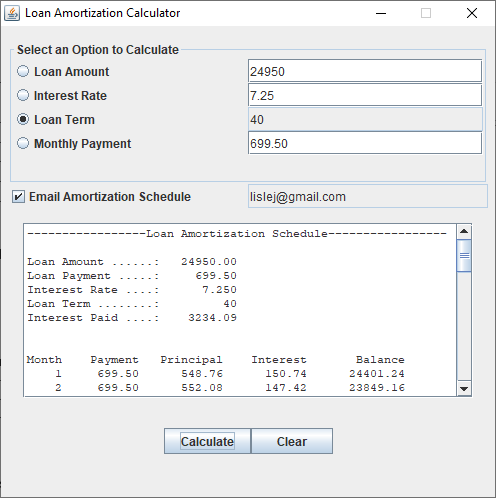
**Figure 4d – Associated Loan Amortization Schedule**



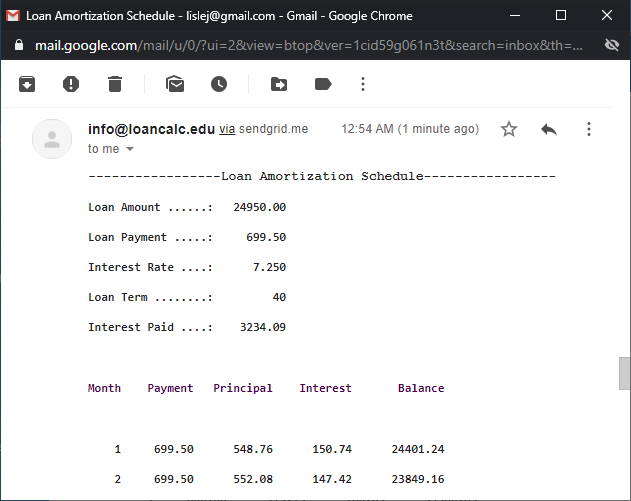
**Figure 4e – Loan Interest Calculation with Email Option**



**Figure 4f – Associated Loan Amortization Schedule**



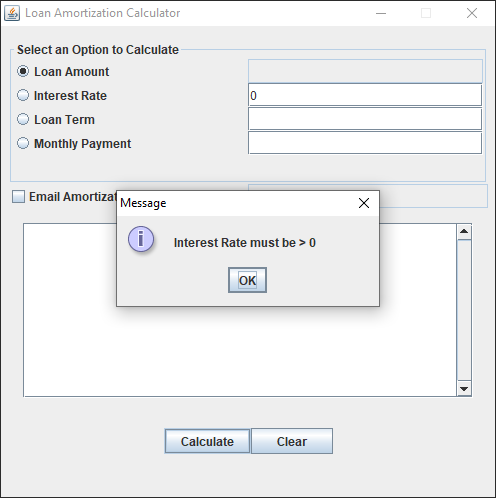
**Figure 4g – Loan Term Calculation with Email Option**



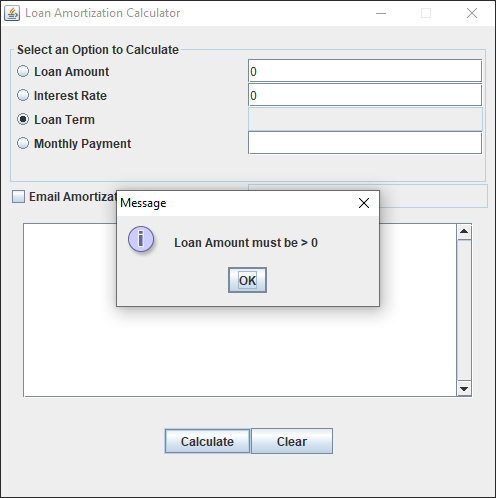
**Figure 4h – Associated Loan Amortization Schedule**

**Testing Scenarios - Application Validation**

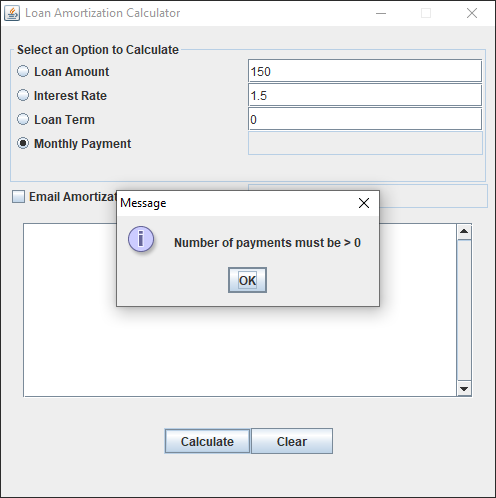
* GUI validates data as numeric and greater than 0



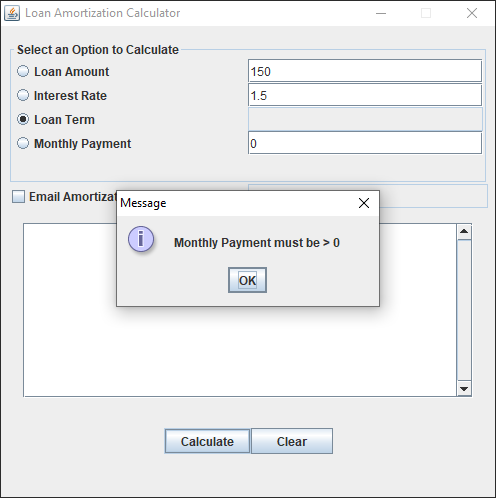
**Figure 5a - Interest Rate validated as numeric and greater than 0**



**Figure 5b – Loan Amount validated as numeric and greater than 0**



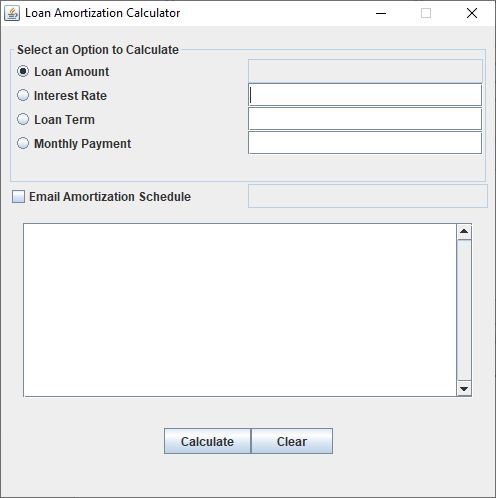
**Figure 5c – Loan Term validated as numeric and greater than 0present**



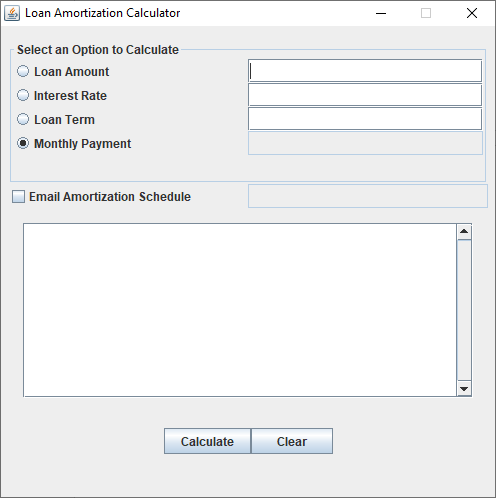
**Figure 5d – Loan Payment validated as numeric and greater than 0**

**Testing Scenarios - Application Usability**

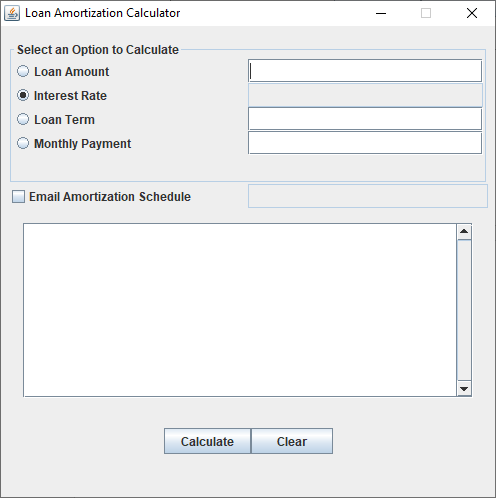
* GUI allows the user to select the variable to calculate and provides the required fields for entering the three required values to complete the calculation



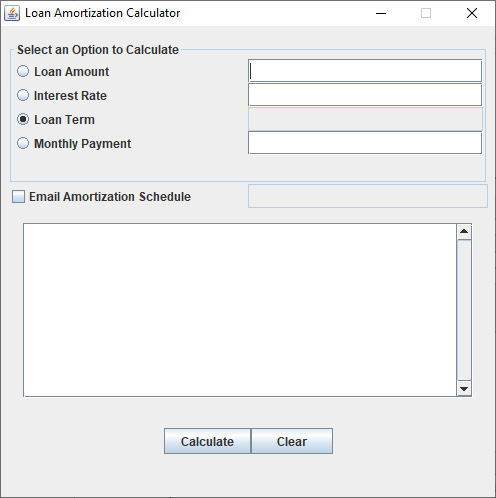
**Figure 6a - Calculating Loan Amount**



**Figure 6b - Calculating Loan Payment**



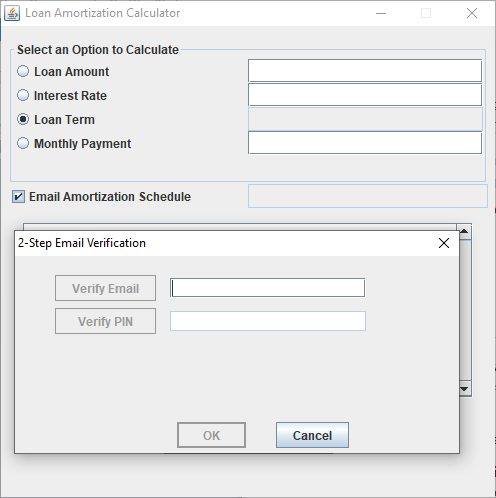
**Figure 6c - Calculating Interest Rate**



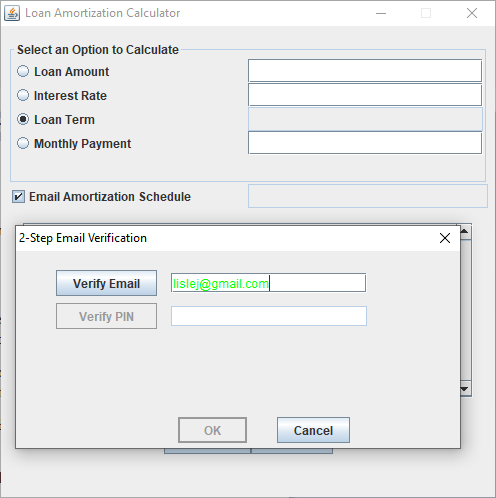
**Figure 6d - Calculating Loan Term**

**Testing Scenarios - Application Usability**

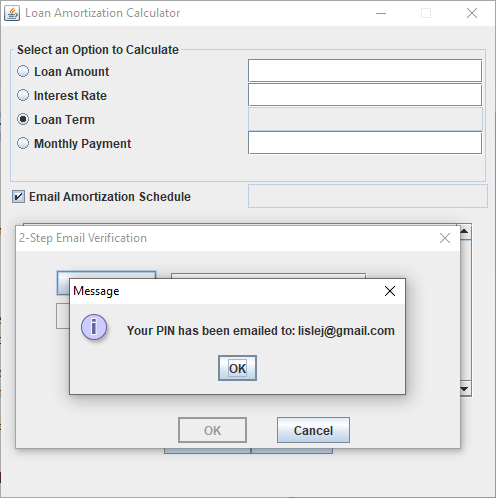
* GUI gives the user the option of emailing an amortization schedule with the entered loan scenario after 2-step email verification is satisfied



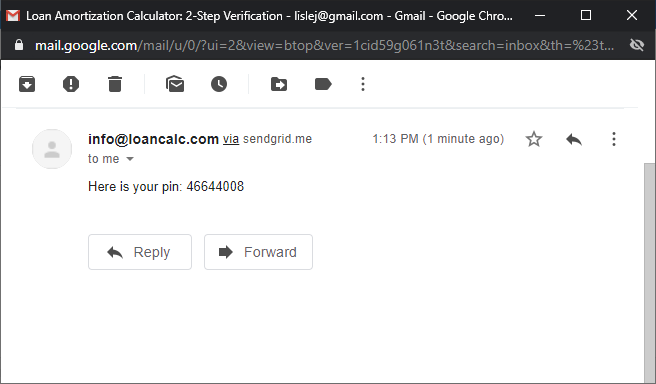
**Figure 7a - 2-Step Email Verification**



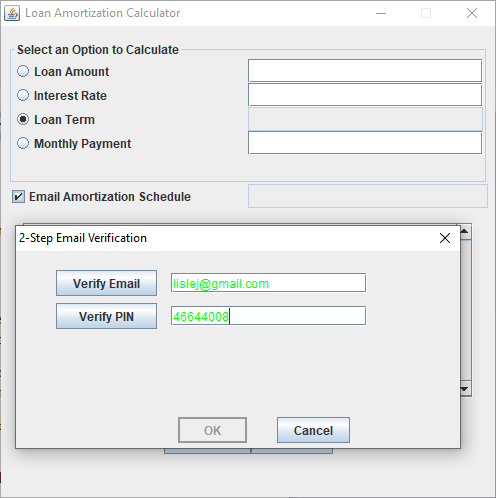
**Figure 7b - 2-Step Email Verification with valid email address**



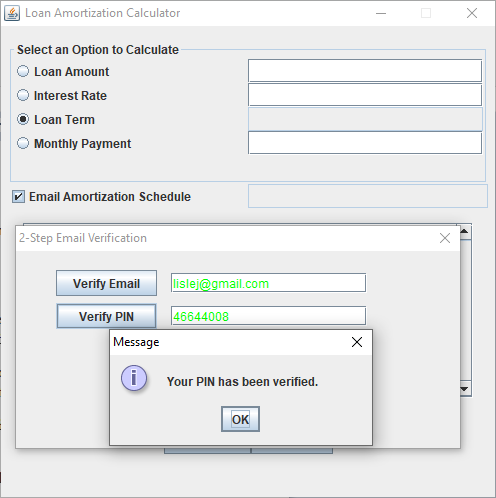
**Figure 7c - 2-Step Email Verification with PIN sent to email address being verified**



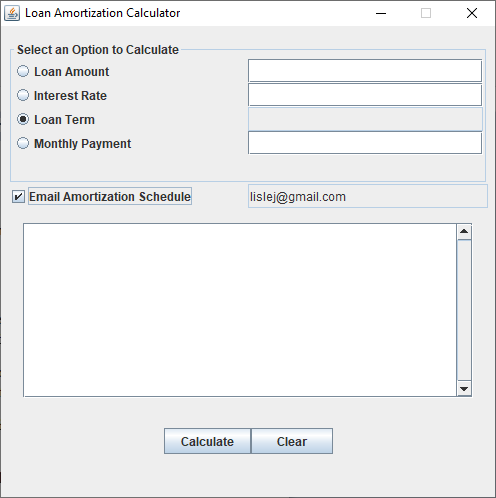
**Figure 7d - 2-Step Email Verification with retrieving PIN from email**



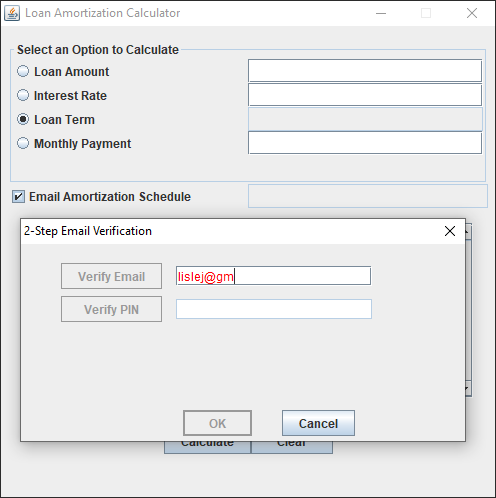
**Figure 7e - 2-Step Email Verification with PIN being verified**



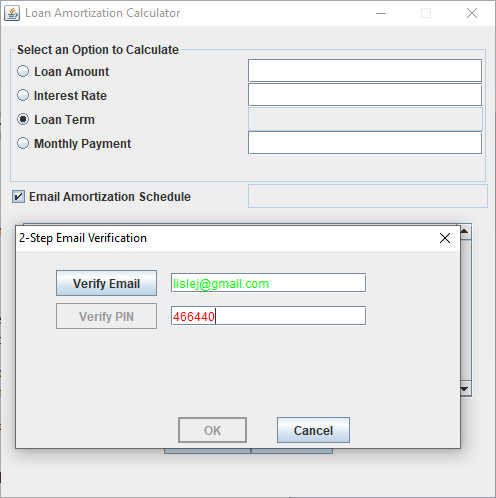
**Figure 7f - 2-Step Email Verification with verified PIN**



**Figure 7g - 2-Step Email Verification with verified email address**



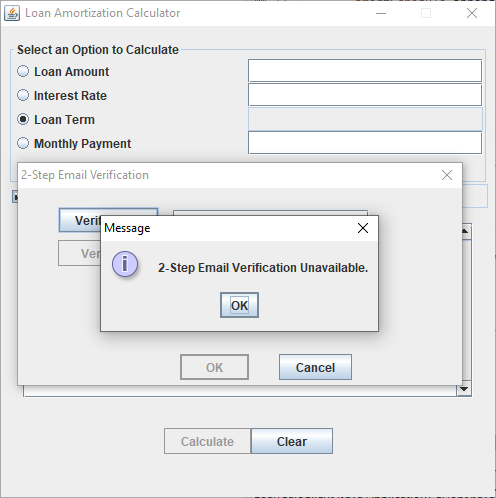
**Figure 7h - 2-Step Email Verification with incomplete email address**



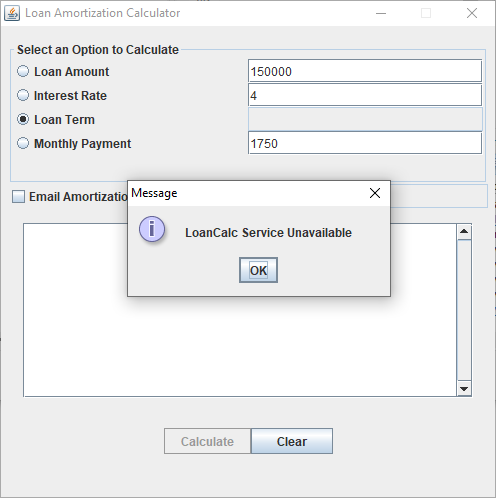
**Figure 7i - 2-Step Email Verification with incomplete PIN**

**Testing Scenarios - Application Usability**

* GUI reports connectivity issues



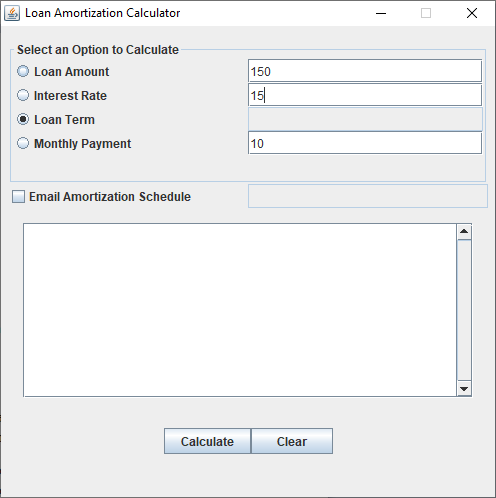
**Figure 8a - 2-Step Email Verification Unavailable**



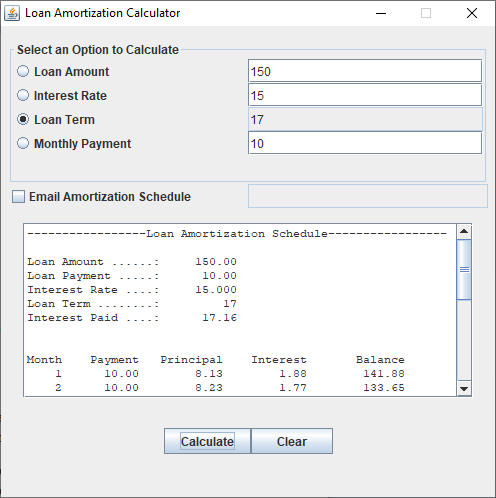
**Figure 8b - LoanCalc Service Unavailable with grayed out Calculate Button**

**Testing Scenarios – Application Usability**

* GUI provides an enabled Calculate button for submitting the loan scenario to the Loan Amortization Service



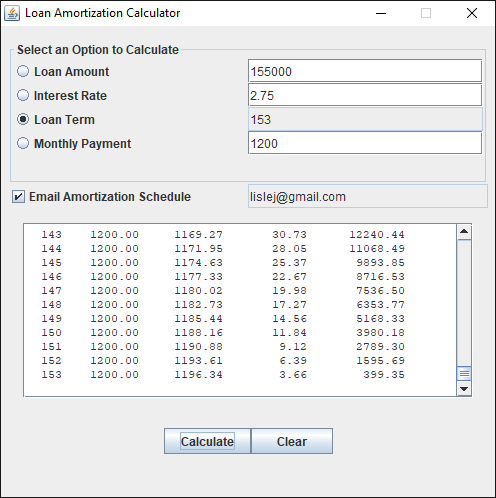
**Figure 9a – Calculator data staged for submission**



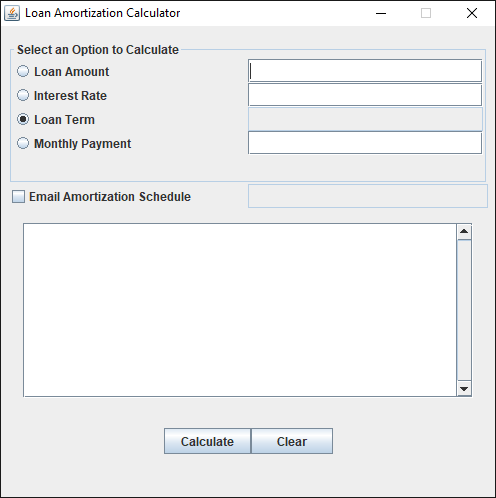
**Figure 9b – Calculator results after submission**

**Testing Scenarios - Application Usability**

* GUI provides a Clear button for resetting all values displayed on the screen



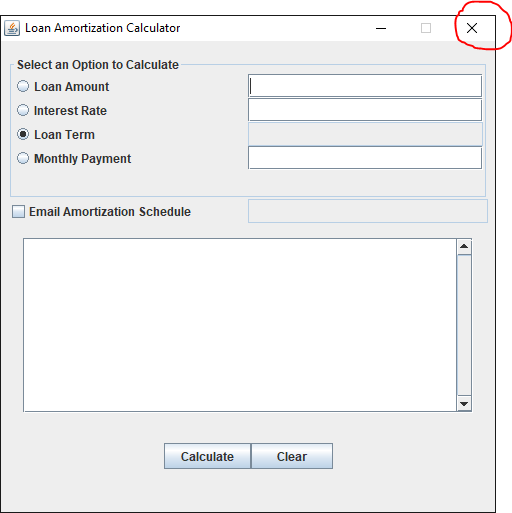
**Figure 10a – Calculator view before Clear button press**



**Figure 10a – Calculator view after Clear button press**

**Testing Scenarios - Application Usability**

* GUI provides the user with the ability to exit the application



**Figure 11 – Red Circled X shows the control to be pressed to Exit Application**

**Testing Scenario – Service Operations Validation**

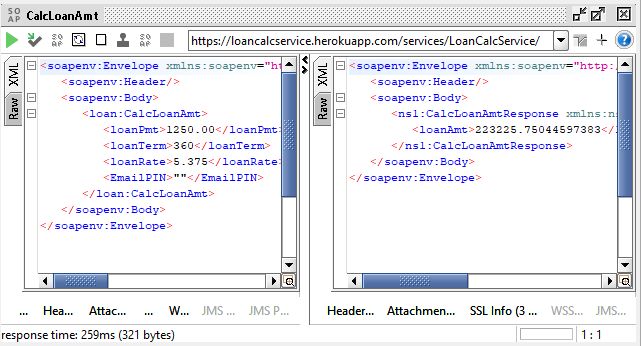
## **Service Test Cases**

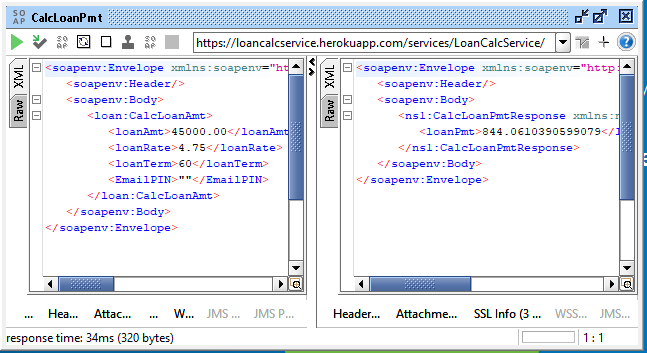
|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **TC #** | **Loan Amount** | **Loan Payment** | **Loan Rate** | **Loan Term** | **Data**  **Inputs** | **Expected**  **Results** | **Actual**  **Results** |
| **1** | **P**  **223225.75** | **1250.00** | **5.375%** | **360 months** |  |  |  |
| **2** | **45000** | **A** | **4.75%** | **60 months** |  |  |  |
| **3** | **60000** | **600** | **R** | **120 months** |  |  |  |
| **4** | **24950** | **699.50** | **7.25%** | **n** |  |  |  |

**Table 3 – Service Operation Test Cases**

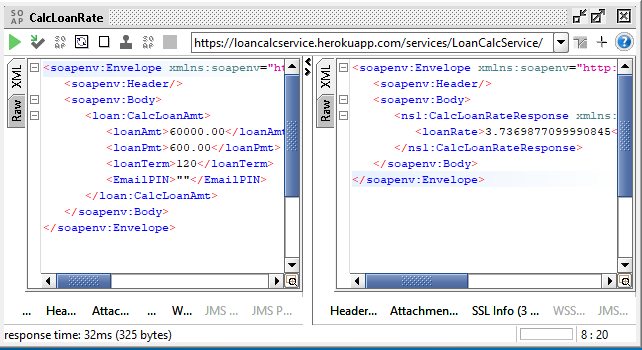
**Testing Scenarios - Service Operation Validation**

* SOAP request and response artifacts created in SoapUI for service operation testing

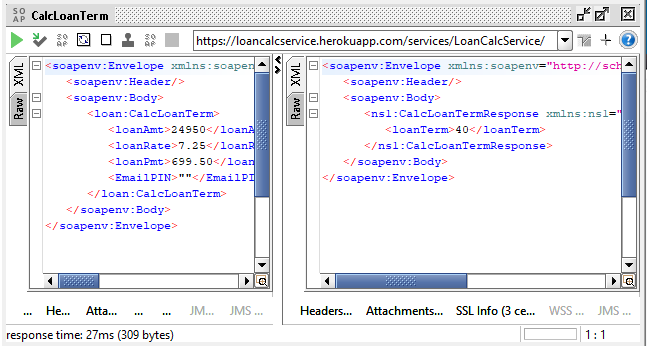
**Figure 12 – CalcLoanAmt Request and Response in SoapUI**



**Figure 13 – CalcLoanPmt Request and Response in SoapUI**



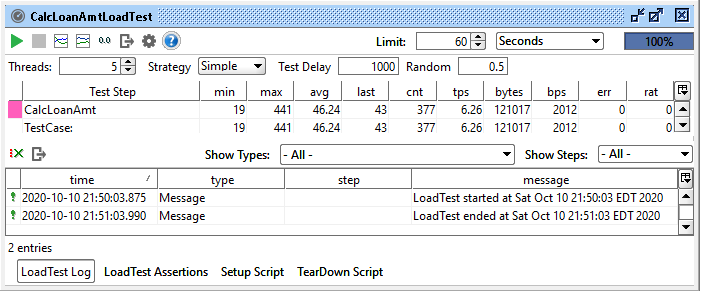
**Figure 14 – CalcLoanAmt Request and Response in SoapUI**



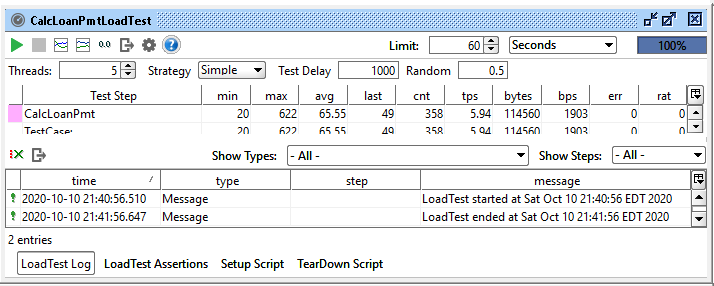
**Figure 15 – CalcLoanTerm – Request and Response in SoapUI**

**Testing Scenarios - Service Operation Validation**

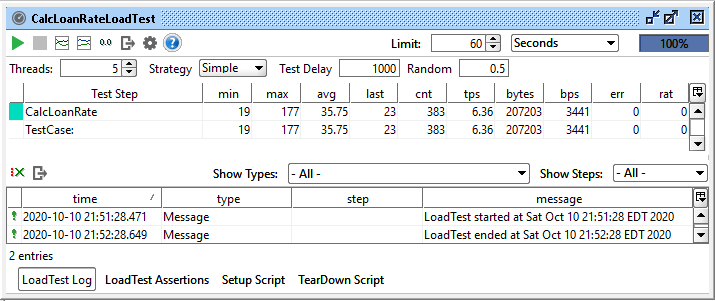
* Rudimentary performance and scalability testing was performed across the four calculation operations and is shown in figures … determined that given 5 concurrent connections submitting approximately 6 transaction per second yielded average round trip timings of 31.08ms to 65.55ms with throughputs ranging from 358 to 388 total transactions in 60s.



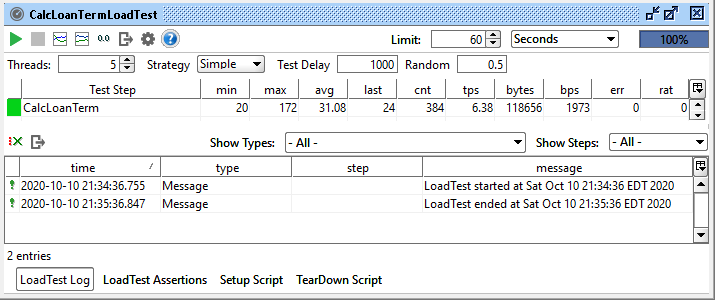
**Figure 16a - Performance and scalability results for CalcLoanAmt**



**Figure 16b - Performance and scalability results for CalcLoanPmt**



**Figure 16c - Performance and scalability results for CalcLoanRate**



**Figure 16d - Performance and scalability results for CalcLoanTerm**

## **Testing Tools**

**Client**

* Loan Amortization Calculator Client

**Service**

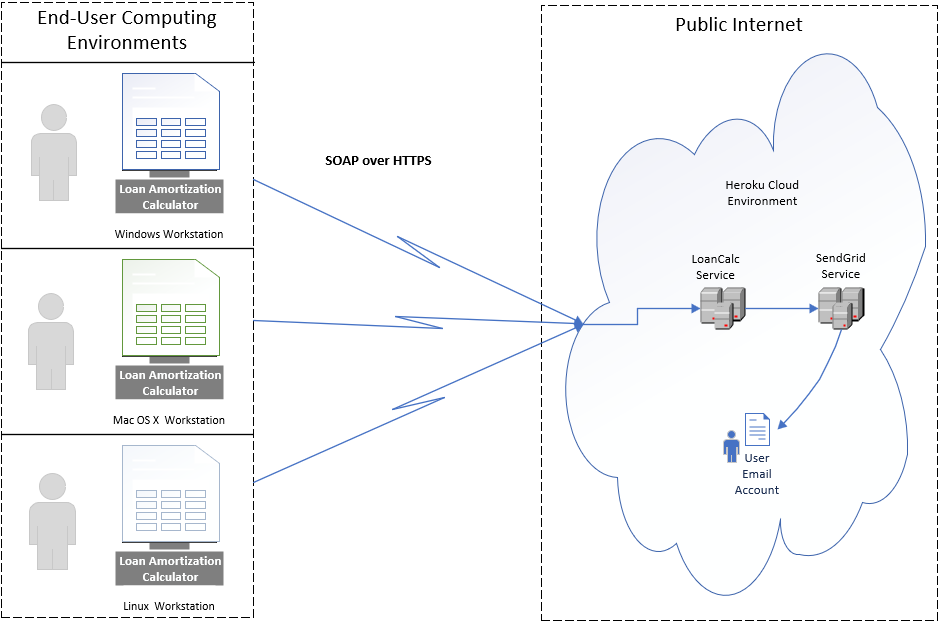
* Loan Amortization Calculator Client
* SoapUI Client – scenario testing
* SoapUI Client – performance testing

# Resource Requirements

* Computer running with minimum hardware requirements of 4GM memory, 1GB free disc space, connectivity to the internet with one of the following OSs - Windows 10, MacOS 10.12+, any popular Linux distro
* Client application installed
* Java JRE 8+ installed
* Loan Calc Service installed on Heroku Cloud Environment

# System Design

The proposed solution for the Loan Amortization Calculator includes a Java Swing Interface that can run on any modern operating system such as Windows, Mac OS X, or Linux with Java version 8 or above installed, and an internet connection to a cloud-based service that provides the loan amortization calculation engine, see figure 17.



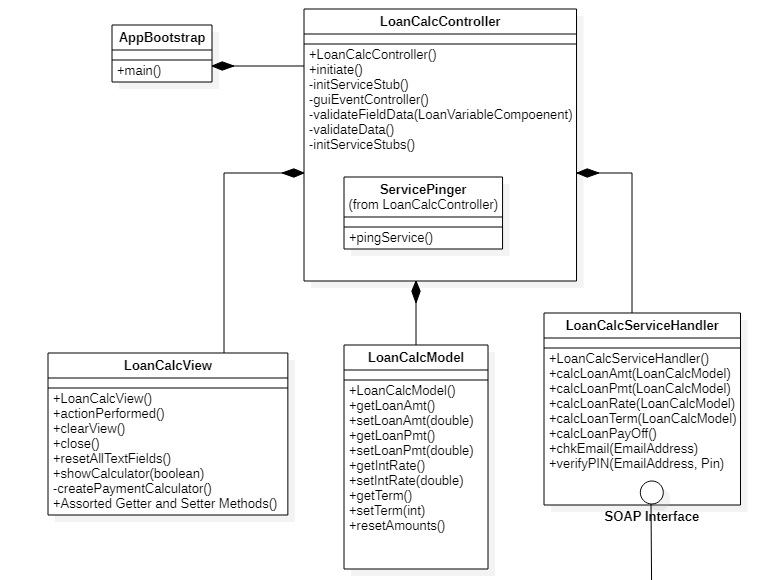
**Figure 17 – System Architecture for the Loan Amortization Calculator**

## **Class Design**

The Loan Amortization Calculator design consists of a Java Swing GUI for selecting the type of amortization calculation desired and a remotely located service running on Tomcat 9.x in the Heroku Cloud to perform the actual calculation. The components communicate securely using the SOAP 1.2 protocol over HTTPS in a synchronous call.

## **Client**

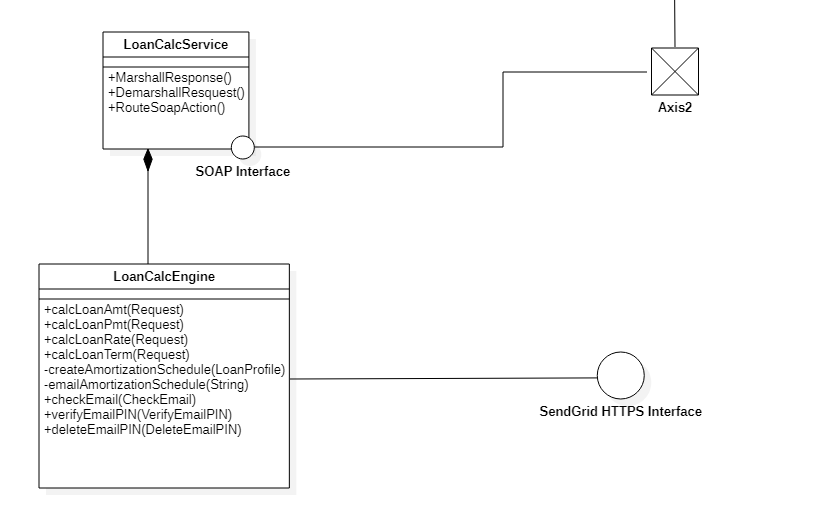
The client component implements an MVC design pattern using Java Swing. A high level view of the classes used to implement the GUI are shown in figure 18.



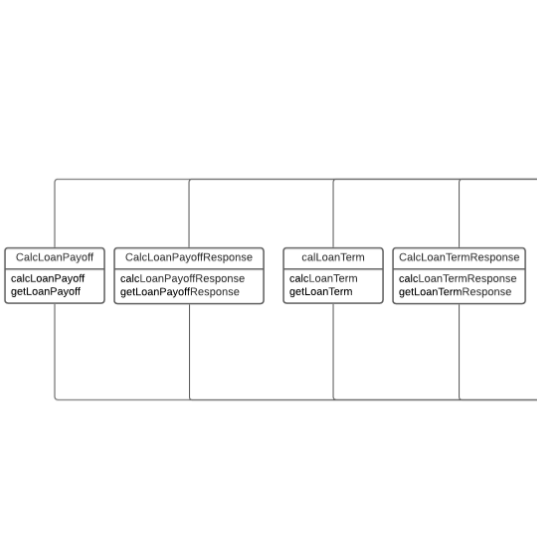
**Figure 18 - Loan Amortization Calculator Client Classes**

## **Service**

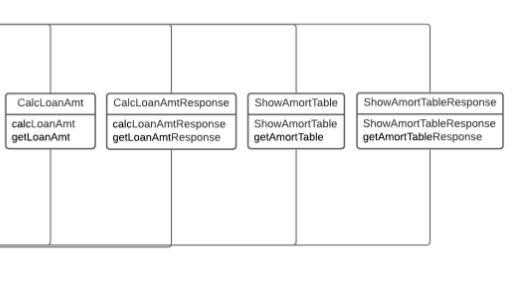
The service component consists of a calculation class wrapped with a set of auto-generated classes by Axis2 to marshal/de-marshal the SOAP messages and route them to the corresponding operation. A high level view of the classes used to implement the Service are shown in figures 19, 20a, 20b, and 20c.



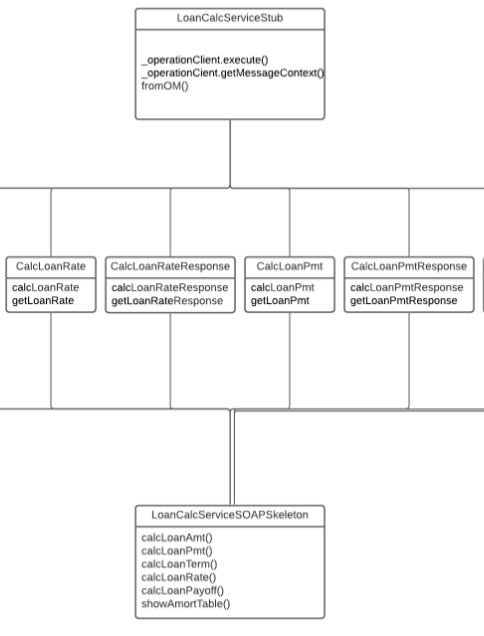
**Figure 19 - Loan Amortization Calculator Service Classes**



**Figure 20a – Axis2 Request and Response Classes**



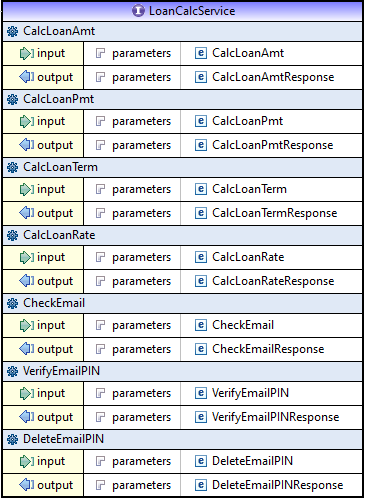
**Figure 20b – Axis2 Request and Response Classes**



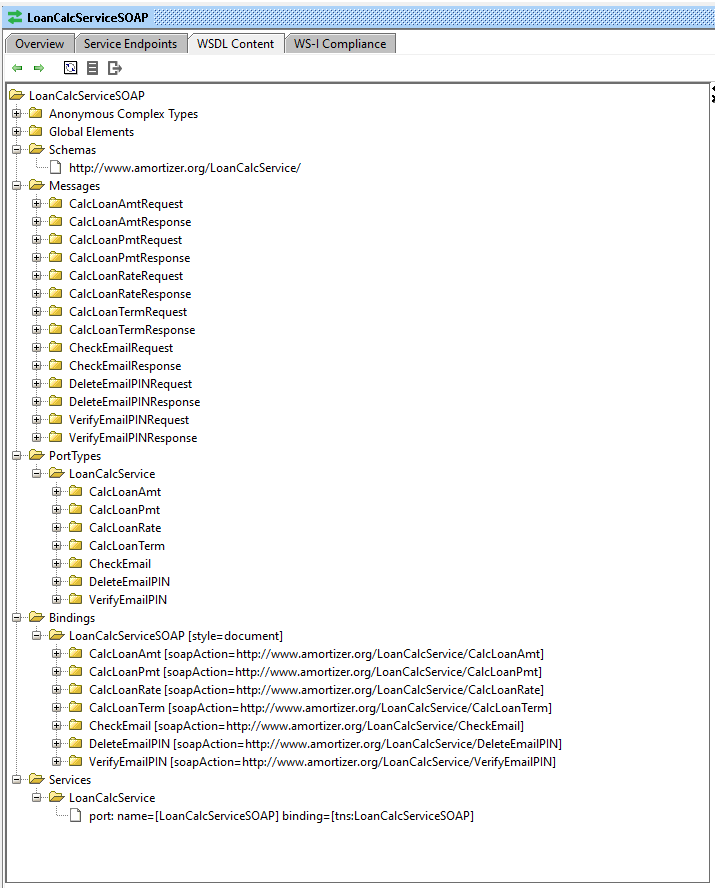
**Figure 20c – Axis2 Request, Response, Stub, and Skeleton Classes**

## **WSDL Specification**

The service is defined by a WSDL specification shown in Figure 5 In brief this file specifies the schema of the service, the associated messages, a port type that defines an operation and the associated request and response messages, a binding which defines the ‘style’ and transport of the message which generally are either RPC or Document and HTTPS or SOAP, respectively, and the service.



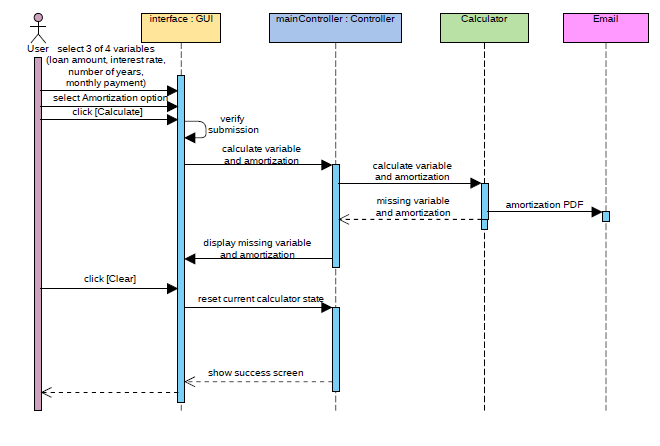
**Figure 21 - WSDL specification for the Loan Amortization Calculator**



**Figure 22 – LoanCalcService Components of SOAP Interface**

## **Sequence Diagram**

End to end call for passing a SOAP request from the Loan Amortization Client to the Service with a SOAP response returned with the calculated result.



**Figure 23 – Loan Amortization Calculation Sequence Diagram**

## **Business Logic**

Loan Payment Calculation:

Loan Amount Calculation:

Loan Term Calculation:

where

* A = loan payment per term
* P = loan principal
* r = interest rate per term
* n = total number of payments

Loan Rate Calculation:

**double** maxRate = 100;

**double** minRate = 0;

**double** midRate = 0;

**double** guessLoanAmt = 0;

**double** monthlyRate = 0;

While (minRate < maxRate – DELTA) {

midRate = (minRate + maxRate)/2

monthlyRate = midrate / (12\*100)

loanAmtGuess = calcLoanAmt(…)

if ( loanAmtGuess > loanAmt)

maxRate = midRate;

else

minRate = midRate;

}

this.rate = midRate

**Data Model**

There is no persistent data model used in this application.

**Performance**

The Loan Amortization Calculator has been shown to perform in sub-second throughput in performance testing.

**Risk**

The biggest risk to this application is that the service is unavailable due to connectivity issues or outage.

**Assumptions**

The end user will be able to set up the application with a valid JRE and a necessary internet connection.

# Alternate Design

The only alternate design per say that was utilized in this project an HTML wrapper to create the loam amortization schedule in place of creating a PDF file due to formatting complications.

# Development History

The development history for this project is outline in the four phases of the project plan listed in

|  |  |
| --- | --- |
| **Task** | **Date** |
| **Phase I** | |
| Finalize Loan Calculator  WSDL | 9/16-9/22 |
| Create Client Project | 9/16-9/22 |
| Create Client Controller | 9/16-9/22 |
| Create Client View Class | 9/16-9/22 |
| Create Client Model Class | 9/16-9/22 |
| Create Client Stub Classes | 9/16-9/22 |
| Create Service Project | 9/16-9/22 |
| Generate Service Classes from WSDL | 9/16-9/22 |
| Generate Service Classes | 9/16-9/22 |
| Create LoanCalcEngine Class | 9/16-9/22 |
| Check-in code to Github | 9/16-9/22 |
| Create Heroku Environment for Service | 9/16-9/22 |
| **Phase II** | |
| Connect Client to Service | 9/23-9/29 |
| Review and enhance validation. | 9/23-9/29 |
| Enhance JUnit tests for unit tests | 9/23-9/29 |
| Test ping Timer class over a long duration (4 hours) | 9/23-9/29 |
| Add field for email address & verification code to GUI if amortization table is selected. Include validation that sends email account a verification code to prove user controls account | 9/23-9/29 |
| Review Service Calculation class | 9/23-9/29 |
| Review and Enhance JUnit tests for unit tests | 9/23-9/29 |
| Update WSDL with email address. | 9/23-9/29 |
| Service Testing using SoapUI, JMeter | 9/23-9/29 |
| Create Itext PDF helper class for creating pdf documents | 9/23-9/29 |
| Updated Project Plan | 9/23-9/29 |
| Update User Guide | 9/23-9/29 |
| Update Testing Doc | 9/23-9/29 |
| Update Technical Doc | 9/23-9/29 |
| Create Phase 2 Report | 9/23-9/29 |
| **Phase III** | |
| Add 2-step email verification dialog to client | 9/30-10/06 |
| Update amortization schedule displayed to include loan info with interest paid | 9/30-10/06 |
| Client Security Review | 9/30-10/06 |
| Defer final PDF generation modifications until Phase 4 | 9/30-10/06 |
| Update WSDL with email address and pin verification. | 9/30-10/06 |
| Generate Web Service artifacts for client and service for 2-step verification for email address. | 9/30-10/6 |
| Create email class for SendGrid – EmailManager – that allows emails with and without attachements. | 9/30-10/6 |
| Service Testing of Email Functionality | 9/30-10/6 |
| Add property file for SendGrid API access | 9/30-10/6 |
| Add amortization schedule generation to service. | 9/30-10/6 |
| Service Security Review | 9/30-10/6 |
| End to End testing | 9/30-10/6 |
| Assorted bug fixes | 9/30-10/6 |
| Documentation updates | 9/30-10/6 |
| **Phase IV** | |
| System Testing | 10/7-10/13 |
| System Documentation Completion | 10/7/10/13 |

**Table 4 – Loan Amortization Calculator Project History**

# Conclusions

This project was an exercise in creativity, collaboration, time management, execution, and delivery. At the end of the day a reasonable product was created, tested, and delivered which fulfilled the goals laid out in the design document. The project was successfully delivered based on teamwork and common goals. While the Loan Amortization Calculator created by Group Charlie is functional many more elements could be added to give it greater usability such as prepayment options, loan life payoff scenarios, what if scenarios, as well as graphics to show the principle and interest curves of the life of a loan and of course a PDF-based amortization schedule. All in all this project was a lot of fun and served the purpose of providing a work product that was achieved based on executing a set of choreographed steps that are essential to delivering a quality software application in a predictable period of time.

# References

Sullivan, M. (2008). Algebra & Trigonometry. pg 952. Pearson. New Jersey

n.d.(2020). Working with WSDLs. Retrieved from <https://www.soapui.org/docs/soap-and-wsdl/working-with-wsdls/> on September 11, 2020

n.d. (2020). Build apps for free on Heroku. Retrieved from [https://www.heroku.com/free on September 11](https://www.heroku.com/free%20on%20September%2011), 2020.

n.d. (2020). How to Send an SMTP Email. Retrieved from [https://sendgrid.com/docs/API\_Reference/SMTP\_API/getting\_started\_smtp.html on October 1](https://sendgrid.com/docs/API_Reference/SMTP_API/getting_started_smtp.html%20on%20October%2012), 2020.

# Addendum

## **User Guide**

