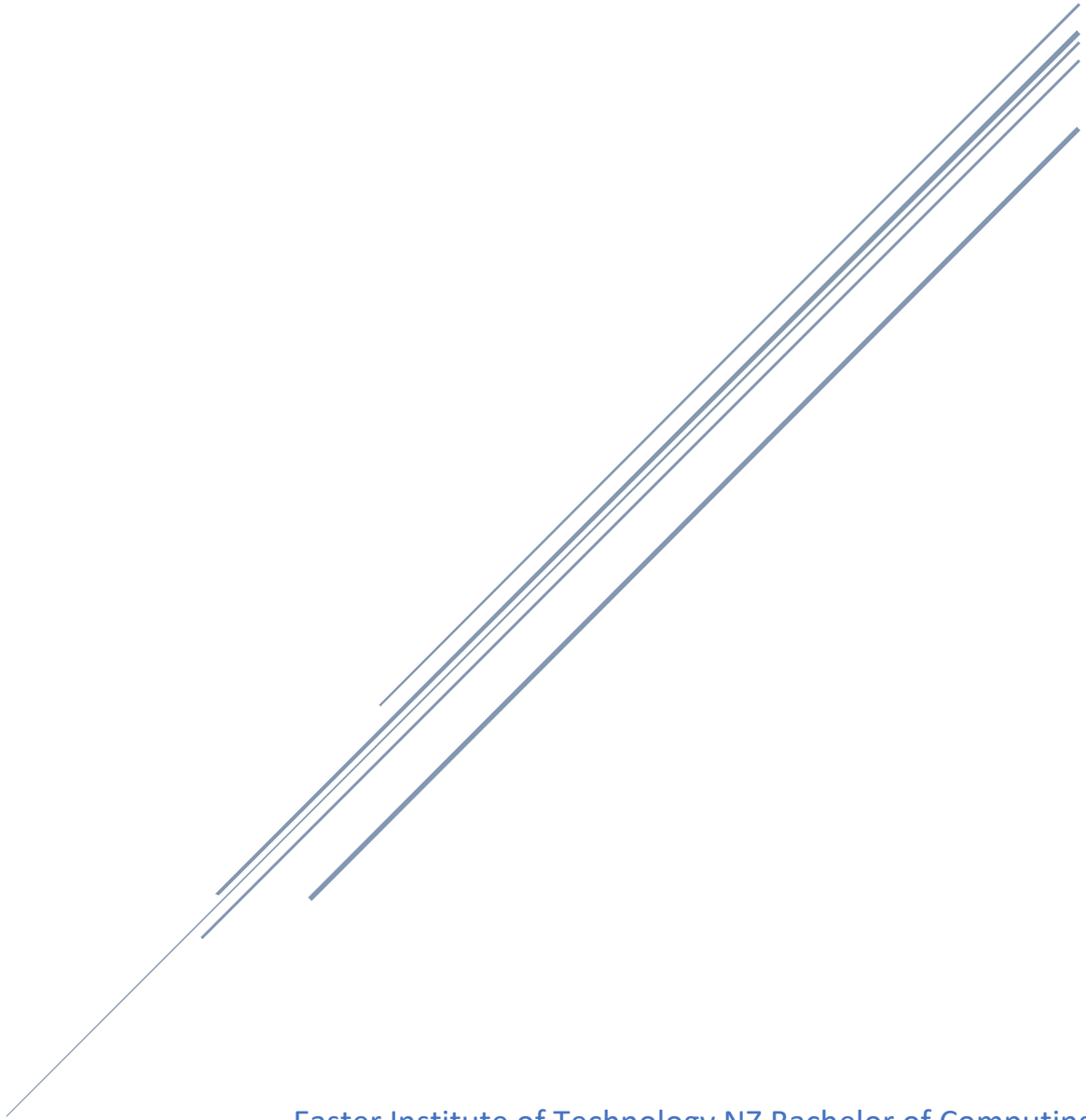


SOFTWARE DESIGN DOCUMENTATION

Mortgage Calculator



Easter Institute of Technology NZ Bachelor of Computing Systems
ITPR7.508 Business Application Programming

John Jamieson

Katherine Mulder & Alex Borawski

Revision History

Name	Date	Reason For Changes	Version

Contents

Revision History.....	i
Section 1. Introduction	1
1.1 Document Purpose	1
Section 2. Subject Scope	1
2.1 Project Includes.....	1
2.2 Project Excludes	2
Section 3. Analysis/ Reflection of Business Issues	2
3.1 The first stage of the initial set-up	3
3.2 Update mortgage	3
3.3 Transaction reporting	3
3.4 Mortgage editing	3
3.5 Visualization chart.....	3
3.6 Authentication	3
3.7 Personalization.....	4
3.8 Intuitive interface	4
Section 4. Analysis/ Reflection of the Program Specifications	4
4.1 Graphical user interface	4
4.2 User Accounts.....	5
4.3 Data entry structure	5
4.4 Analytical functionality.....	5
4.5 Data removal processes	6
Section 5. Key potential software application issues	6
5.1 Reliability	6
5.2 Scalability	7
5.3 Performance.....	7
5.4 Maintainability	7
5.5 Security	7
5.6 Usability.....	8
5.7 Compatibility.....	8
Section 6. System Overview.....	8
6.1 Software Development Technologies and Tools	9
Section 7. Data Design	9
7.1 Entity Relationship Diagram.....	9
7.2 Data Flow Diagram	10
Section 8. System Design	11
8.1 User Case Diagram	11
8.2 Component Diagram	12
8.3 Activity Diagram	13

8.4	Class Diagram.....	18
Section 9.	Interface Design	18
9.1	The user login page	18
9.2	The signup page	19
9.3	The index (home) page	20
9.4	The add mortgage page.....	21
9.5	The updated mortgage page.....	22
9.6	Deleting Data Page	23
9.7	User account icon	24
9.8	Update Account.....	25
9.9	User account settings	26
Section 10.	Test Framework.....	28
10.1	Unit Testing	28
10.2	Integration Testing	31
10.3	System Testing	32
10.4	User Acceptance Testing.....	32
Conclusion		35

Section 1. Introduction

This document outlines the architecture and design of the proposed Mortgage Calculator (calculator). This calculator aims to simplify mortgage management processes for individuals by offering a user-friendly interface and comprehensive features. It is important to note that this calculator is intended for personal use only and is not designed for financial advisers or professional consultation. This Software Design Document serves as a detailed guide for developers, stakeholders, and users, providing insights into the software's functionality, structure, and behaviour. This includes data design, system design, interface design, and testing framework.

1.1 Document Purpose

This document explains how the Mortgage Calculator works and how it is built. This is designed to make it easy to understand how the software functions and what it will look like. We have also included a user-required specifications document that defines what users can expect from this project.

- **Users/ Project Managers:** This document serves as a guide for both project managers and users of the Mortgage Calculator. It provides the software's functionality, design rationale, usability aspects, and system behaviour. By understanding these details, users and project managers can make informed decisions about the project, while users can use the software to meet their needs.
- **Developers:** This guide helps developers understand the software design, important decisions made about its structure, and how it is built. It provides a roadmap for developers to turn requirements into a working program.

Section 2. Subject Scope

The Scope for the Mortgage Calculator is as follows:

2.1 Project Includes

- A Mortgage Calculator will be able to take mortgage information given by a user and return an analysis of the mortgage in the form of a summary, graph /and table.
- Users will be able to adjust to their mortgage's information in the form of a transaction, which when completed, will update the analysis given the new information from the given date.
- Users will be able to adjust the period the analysis is showing.

-
- Users will be able to create an account within the Calculator application, allowing them to save their mortgage(s) information and transaction information to a separate database.
 - Establishing a database to contain data from the Mortgage Calculator.
 - A file to run to initialize the database to be run before the first use of the Mortgage Calculator application alongside documentation to help this process.
 - Testing the software to ensure that any major issues are dealt with.
 - All test files will be included in the final delivery of the project.
 - User documentation containing instructions on how to use the Mortgage Calculator

2.2 Project Excludes

- Complex mortgage types: • Adjustable-rate mortgages (ARMs) with various index rates and margins will be excluded.
- The project will focus on developing the website and its functionalities but will not involve the deployment process onto live servers.
- The project will not include SEO services beyond basic considerations for website visibility.
- Post-project maintenance and updates will not be covered under this project. This includes ongoing software support and updates.
- The project will focus solely on basic mortgage calculations and will not include advanced financial analysis features.
- The project does not include any legal advice related to mortgages. Users will need to seek independent legal advice.
- The project does not include tax advice. Users will need to consult with tax professionals or refer to relevant tax regulations for any tax-related matters.
- The project will not include integration with external systems beyond the scope of basic mortgage functionalities.

Section 3. Analysis/ Reflection of Business Issues

In this section, we delve into the process of reflection and analysis regarding key business issues we have encountered, and we will examine the rationale behind the decisions we have agreed upon and provide clear justifications grounded in various considerations.

3.1 The first stage of the initial set-up

Analysis: Getting started is important because it's the base of the whole mortgage management system. Users have to enter the right info to make sure their mortgage gets handled right.

Reflection: To deal with this problem, the system has to make it easy and simple to set up. It should give clear instructions and check that everything's correct.

3.2 Update mortgage

Analysis: It's normal for people's finances or loan terms to change when they have a mortgage. Users should be able to easily and accurately update their mortgage info in the system.

Reflection: The system should let users make these updates themselves. It needs to make sure the changes show up correctly.

3.3 Transaction reporting

Analysis: Transaction reports help users see their payment history and how much they still owe.

Reflection: Users need to have detailed and easy-to-understand transaction reports so they can keep track of their history and understand what's going on.

3.4 Mortgage editing

Analysis: Sometimes, mortgage agreements have to change because of different situations or what users want. Users need to be able to change certain details of their mortgage while making sure everything stays accurate and consistent.

Reflection: Adding editing features is important to make sure users can do what they need to with their mortgage.

3.5 Visualization chart

Analysis: Having pictures of mortgage data can help users understand their money situation and how they've been paying. Users need charts that show their mortgage info clearly.

Reflection: Adding charts can make it easier for users to understand and get interested in their mortgage.

3.6 Authentication

Analysis: Users need a process to create and manage their accounts.

Reflection: Building an authentication system that allows users to log in, log out, change passwords, and delete accounts is essential.

3.7 Personalization

Analysis: Users should have the flexibility to edit or delete historical transactions to ensure the accuracy of their financial records.

Reflection: Users with full control over their dashboard and mortgage details and implement user-friendly editing and deletion options for historical transactions, allowing users to easily make changes as needed.

3.8 Intuitive interface

Analysis: An intuitive interface is important for ensuring that users can easily navigate the mortgage management system and access its features without confusion.

Reflection: We should test how easy it is to use and ask users for feedback to find any problems or confusing parts. Then, we can make changes to the design to make sure it's what users want and expect. Doing these tests regularly and updating the system is important to keep it easy to use.

Section 4. Analysis/ Reflection of the Program Specifications

In reflecting on the program specifications, we'll analyze the final software application and consider any necessary changes or modifications. Also, identify any potential gaps or inconsistencies and propose adjustments to ensure the successful implementation of the software.

4.1 Graphical user interface

Users will be greeted with a login page to access their accounts. There will also be a separate page to create an account. Alongside this, once logged into the application, there will be a user settings page to allow for password changes or account deletion. The program will have a simple interface designed to quickly provide analysis of a mortgage. To be able to do this, the main page of the application will contain all the information relating to the analysis of mortgages, primarily a table of summarized analysis, a graph showing the change in equity over time, and an amortization table showing specific timestamped information.

There will be a separate interface to allow for the creation of a mortgage and transactions. Due to the amount of information required for each of these sections, they will have their page. The user will then be able to see some analysis of the information they enter before saving it permanently and returning to the home page.

Lastly, there will be a page to allow users to remove mortgages and transactions that they no longer want, listing them out and showing identifying information, so they can correctly select the ones they want to remove.

4.2 User Accounts

User Accounts will be implemented in the software to separate mortgages (and subsequently transactions) from other users, with no access whatsoever being allowed between users. These accounts will require a password to secure the account. Primarily, due to the scale of the project, there is no plan to include an admin account to access and manage users from within the application itself, but there will be a way to access the information from the database end, allowing changes to be made there if necessary.

4.3 Data entry structure

With the calculator requiring a lot of specific information for both mortgages and transactions, these pages will be kept simple and straightforward to help users correctly enter information into the right field.

All entry fields will have a description next to their titles outlining the correct formatting. Additionally, the entry fields in the transaction creation screen will include automatically input data from the mortgage that is selected so the users only need to adjust the information in the fields that need changes. Lastly, if any information is entered incorrectly, the entry fields will be highlighted red, and an error message will be displayed to let the user know that something in this field is wrong.

4.4 Analytical functionality

The application will need to handle a lot of analysis, and most of it will be done as a user creates a mortgage or transaction to show on the page before a user confirms this is correct.

This analysis will include the Estimated Repayment, a payment breakdown of interest and principal and information regarding the mortgage maturity (including payments over the full term, full term to amortize, interest over the full term and principal + interest).

It is important to understand that in the transaction creation screen, there will be added information to showcase any extra payments, the payments over the reduced term, estimated reduced term to amortize, interest over the reduced term, interest saved over the reduced term and principal + interest over the reduced term.

This information will then be used to generate a graph (viewable on both the main page and creation pages) and an Amortization table (main page only) based on the state of the mortgage.

The state will be determined by combining the initial details of the mortgage and applying any transactional changes that have occurred to it at the correct time. Users will also be able to apply a starting date to the analysis on the main page, only seeing the start of the mortgage from that date.

4.5 Data removal processes

Another important function of the application is the ability to remove data from the system. Users will be able to remove any mortgages or transactions they want through a single page which will show a table for each type of data, containing all of their entries. Transactions will show the date they were applied for, a comment on the purpose of the transaction, and the mortgage that it relates to.

For mortgages, it will show the start date of the mortgage, the name given to the mortgage, as well as the current balance remaining on the mortgage. Then, next to all of these entries will be a deleted button that if clicked will show a popup confirming the deletion of the selected entry, with a yes button to confirm, and if selected, the data will be removed.

There will be a way to completely delete the account if the user decides to. This option will be accessible under the user settings page, and if selected and confirmed, it will delete the account and return to the login page.

For all these forms of data removal, if data that is reliant on that deleted entry exists, it will also be removed, e.g., if a user deletes a mortgage, all connected transactions will be removed as well.

Section 5. Key potential software application issues

There are some important areas the software will need to handle/work with to deliver a complete solution to the client.

5.1 Reliability

Because we are working with financial information, we need to ensure the application runs reliably. With this in mind, we will make sure our application correctly handles any mathematical data entered to ensure accurate results, as this is the core functionality of the Mortgage Calculator.

The most common areas that could impede this are rounding errors or inputting data out of the range of the expected field.

The best way to overcome this is to include robust error checking in our code to ensure that this is mitigated as much as possible. Robust error checking will be extended throughout the software to make sure that the program is as reliable as

possible. More information about this will be available in our Test framework, which will show how we intend to ensure that our application catches these errors.

5.2 Scalability

Although this software isn't intended to be scaled up to a larger audience, designing the code in a way that allows added features to be included is important to allow for future flexibility of the program.

To allow for this, the code will be designed in a more modular way, with set definitions for classes and different aspects of the code, this will allow future developers to more efficiently make adjustments to the code to suit their needs. Another way to allow for future scalability is through the design of the application as a whole, creating an application environment that can be easily integrated into a wider-reaching framework.

5.3 Performance

The performance of the application is important because the analysis should be readily available to the user when they log into the application.

This is the biggest bottleneck in the application and will require deciding the code to load analyses as quickly as possible. Multi-threading is a potential solution, but most certainly we will be loading all the analyses of a user when they log into the application to speed up load times.

5.4 Maintainability

Although our involvement in the application will cease once the development is complete, we will still develop the application to be easily maintained well into the future. The main way we will deliver on this is by designing the code to be modular, with clear signs of what each section of the code is responsible for.

Comments within the code will be in-depth, covering the functionality of what each section does, how it does it, and what is not included in the launch. This will help future developers understand what is happening within the software and how they can quickly edit, add or remove sections to suit their needs.

5.5 Security

It will be important to handle data securely within our application, especially as we are dealing with users' sensitive financial information. The most likely vulnerability is that

users might be able to access the other user's data without authorization, which means we need to include robust methods to prevent this from happening.

Another potential vulnerability is entering malicious data into the data entry fields throughout the application. To prevent this, we will include a lot of data validations, ranging from type checking, length checking and value checking.

We will also want to secure sensitive data that even administrators should not directly access. To do this, we will use a form of encryption, making the data much harder to access.

5.6 Usability

We will make sure the application is easy to understand and navigate through. To do so, we will design the pages, and the connections between them, to be easily identifiable. Clearly showing the user what is present on a page and what is to be expected on the next page will make the application as a whole easier to understand.

Another aspect of making the software easy to use is giving clear feedback to any errors inputted by the user, as well as highlighting what each part of the interface is doing. To achieve this, we will provide clear feedback to the user after they try and enter incorrect data, explaining what they did wrong. By including descriptions alongside any data input or display areas, we will make it much clearer what is expected or shown to the user.

5.7 Compatibility

Our software will be designed in a manner that makes it more compatible with potential future changes to the environment in which it is operating. With this in mind, we will develop the software to easily support changes in the way the application is used. By isolating how the software accesses a database and how it displays itself on a web browser, we can make it easier to swap out or augment how the application is used.

Section 6. System Overview

Factoring in everything discussed in the previous sections of this document, we have decided to develop an application that connects to an external database to save user's data. This would then connect to our software to create analyses of the user's mortgage(s) and then display this information to the user via a web browser.

This design approach allows us to keep the application small and lightweight, extremely portable and modular, allowing for different usages to be added or removed in the

future, whilst also providing flexibility in how the program is presented to the user via a Graphical User Interface, especially as the default browser of the hardware will be used to render the interface.

Below, we have chosen the specific software we will use to develop this application and why it was chosen over other similar options.

6.1 Software Development Technologies and Tools

For the backend logic and data processing, we have decided to use Python as the programming language. This decision was reached as Python is a language easily readable by most developers, whilst also providing simple ways to import potential packages that might be necessary for the project.

By working with a web browser, we have elected to use HTML, CSS, and JavaScript to create and design our web pages for the project. We will also use Bootstrap to access pre-designed features that make the software more standardized.

We decided to follow a Web framework to make the routing and delivery of web pages more streamlined and easier to understand and modify in the future. With this in mind, we chose Flask as our framework, as it is straightforward to set up and configure as well as understand what each part of the software does.

To handle our data storage, we have decided to use Postgres. The main reason we decided to use Postgres was because of how simple it is to set up and that it can handle small to large data extremely well, meaning that if the software were to be scaled to meet a larger audience, it would still handle the data with ease.

To test our software, we will make use of Pytest as it automates the unit testing of our software, ensuring that the code can handle all predicted problems correctly.

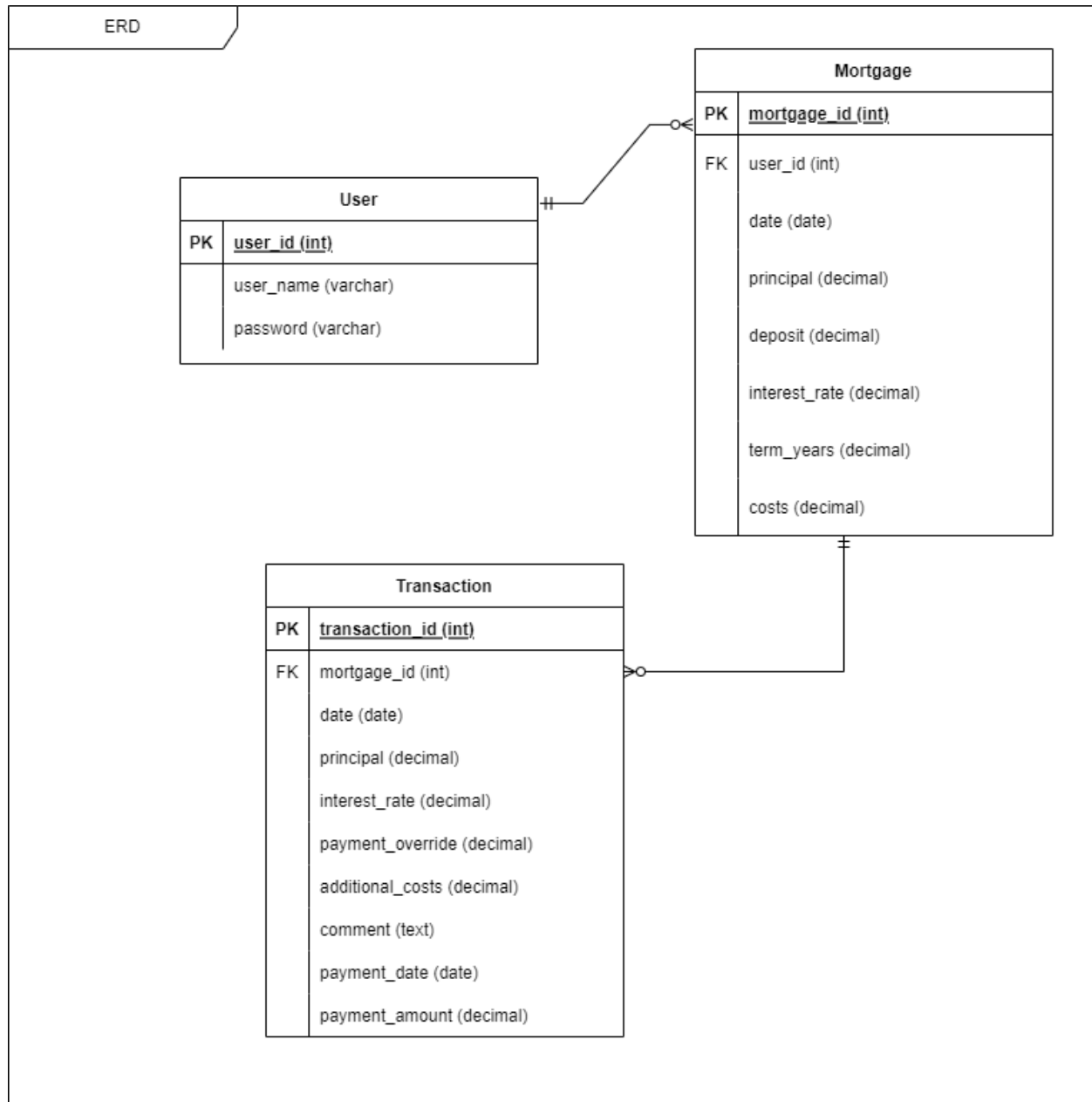
Section 7. Data Design

Data design is about organizing data so we can store, find, and use it easily. In our project, good data design is important because it helps keep our data accurate, makes things run faster, and ensures everything works as it should. We'll use diagrams like ERDs and DFDs to show how our data is set up and how it moves around in our system.

7.1 Entity Relationship Diagram

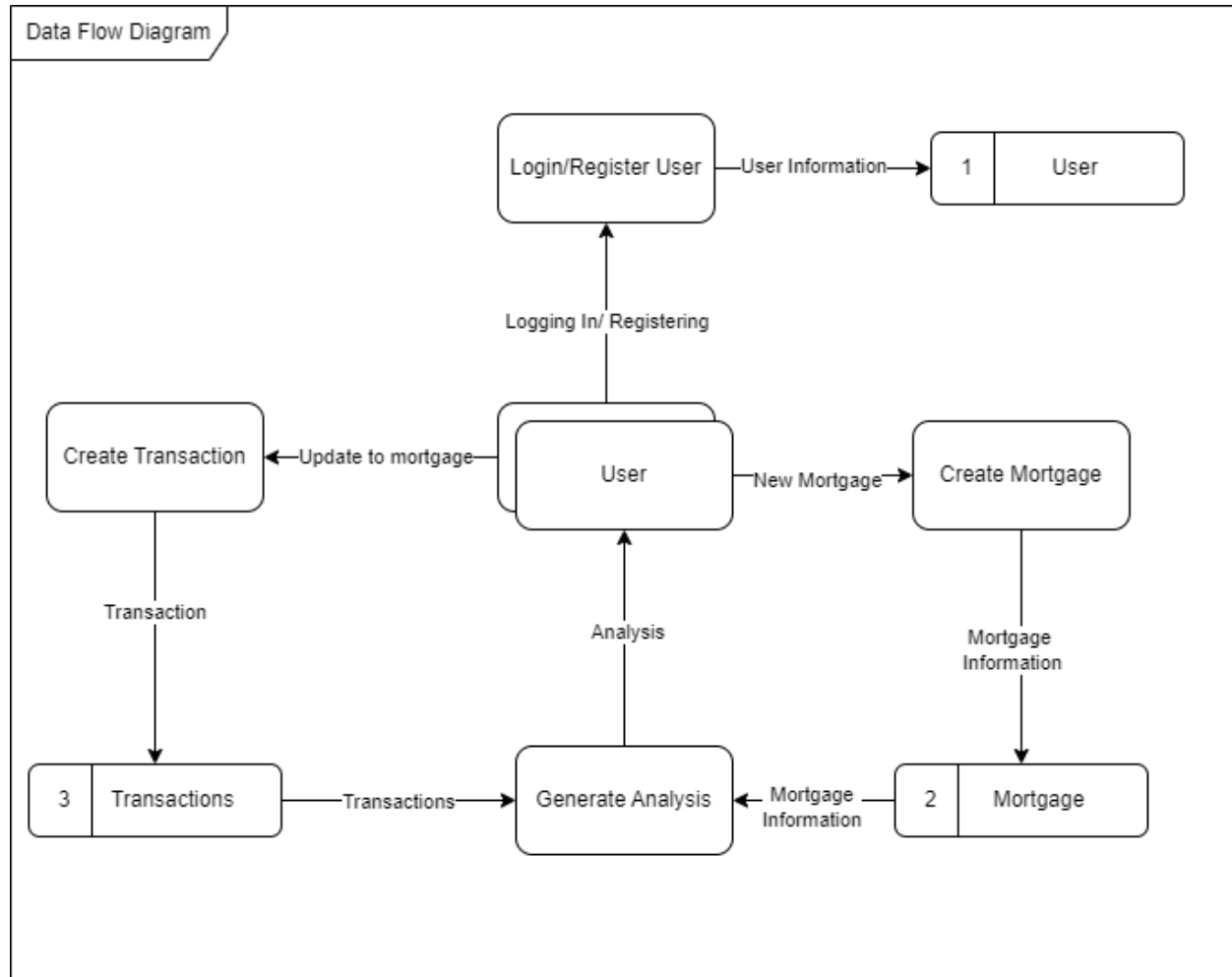
The purpose of the Entity-Relationship Diagram (ERD) is to visually represent the structure of the database for the mortgage calculator. By mapping out entities like "User," "mortgage", "transaction", along with their attributes and relationships.

The ERD will provide a clear understanding of how data is organized within the system.



7.2 Data Flow Diagram

The Data Flow Diagram (DFD) serves to illustrate the flow of data within the system, showing how data is input, processed, and output. It helps to understand the flow of information and the interactions between different components within the software.

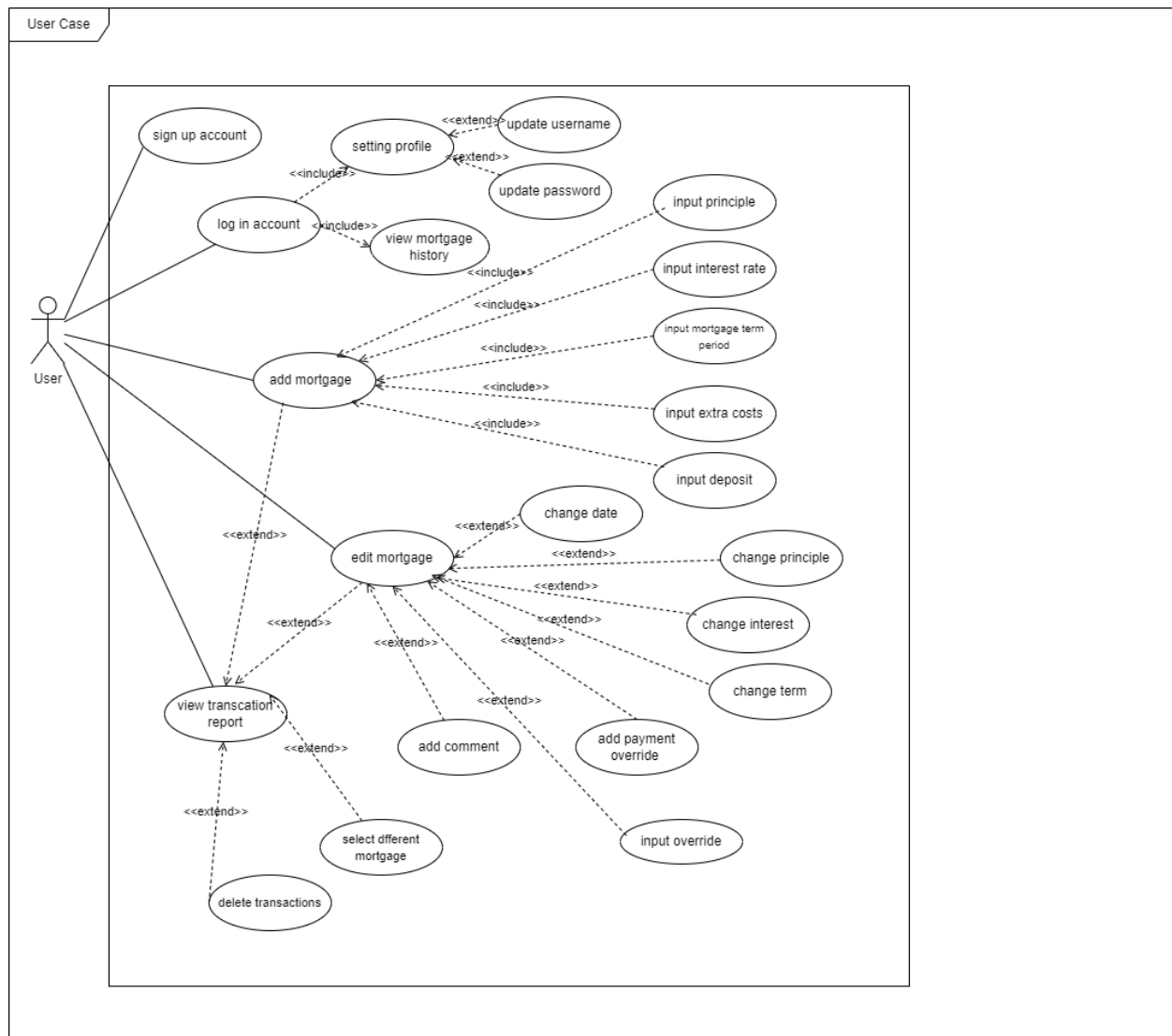


Section 8. System Design

We will provide comprehensive user case, component, activity, and class diagrams to illustrate the structure and implementation of our project system.

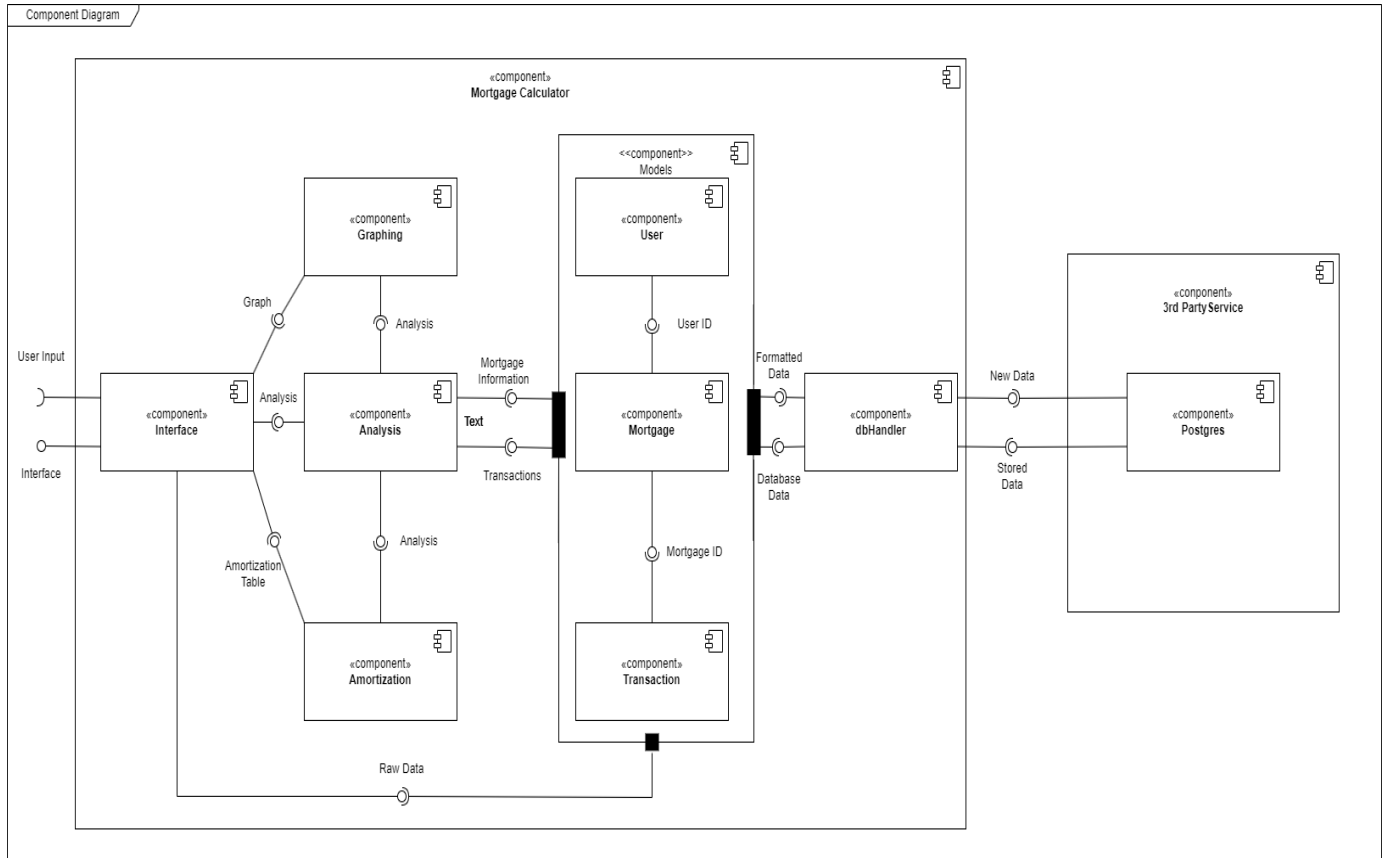
8.1 User Case Diagram

The use case diagram outlines the interactions between users and the system, including the primary functions such as calculating mortgage payments, viewing amortization schedules, adjusting parameters, and accessing mortgage features. user case diagram.



8.2 Component Diagram

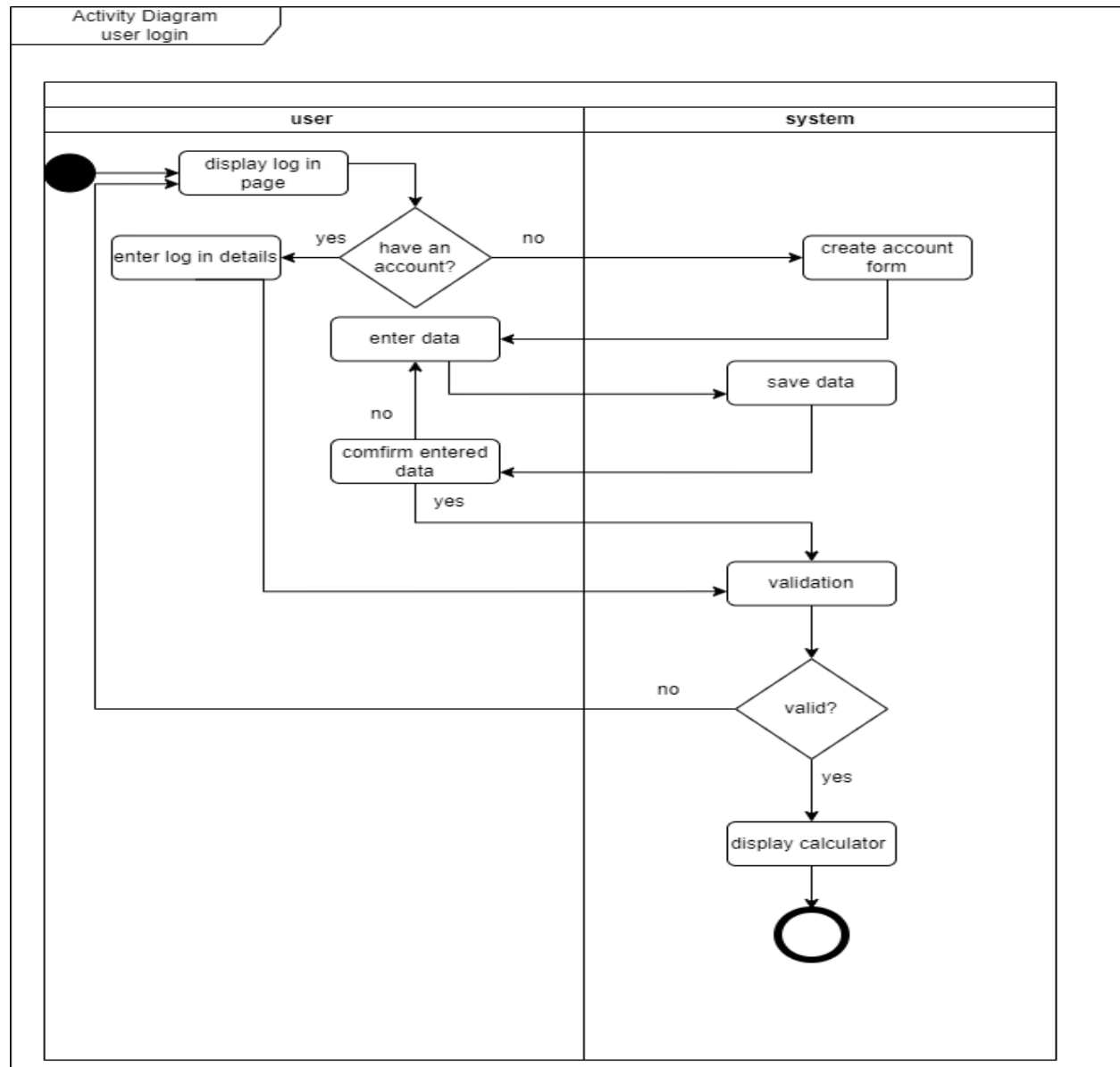
The component diagram shows an overview of how the components of our software will be designed and communicate with each other. As shown in the diagram below, the software will need to communicate externally with Postgres to store the data, which is then handled by the Data Models, run through the analysis, amortization, and graphing components to then produce the interface. component diagram.



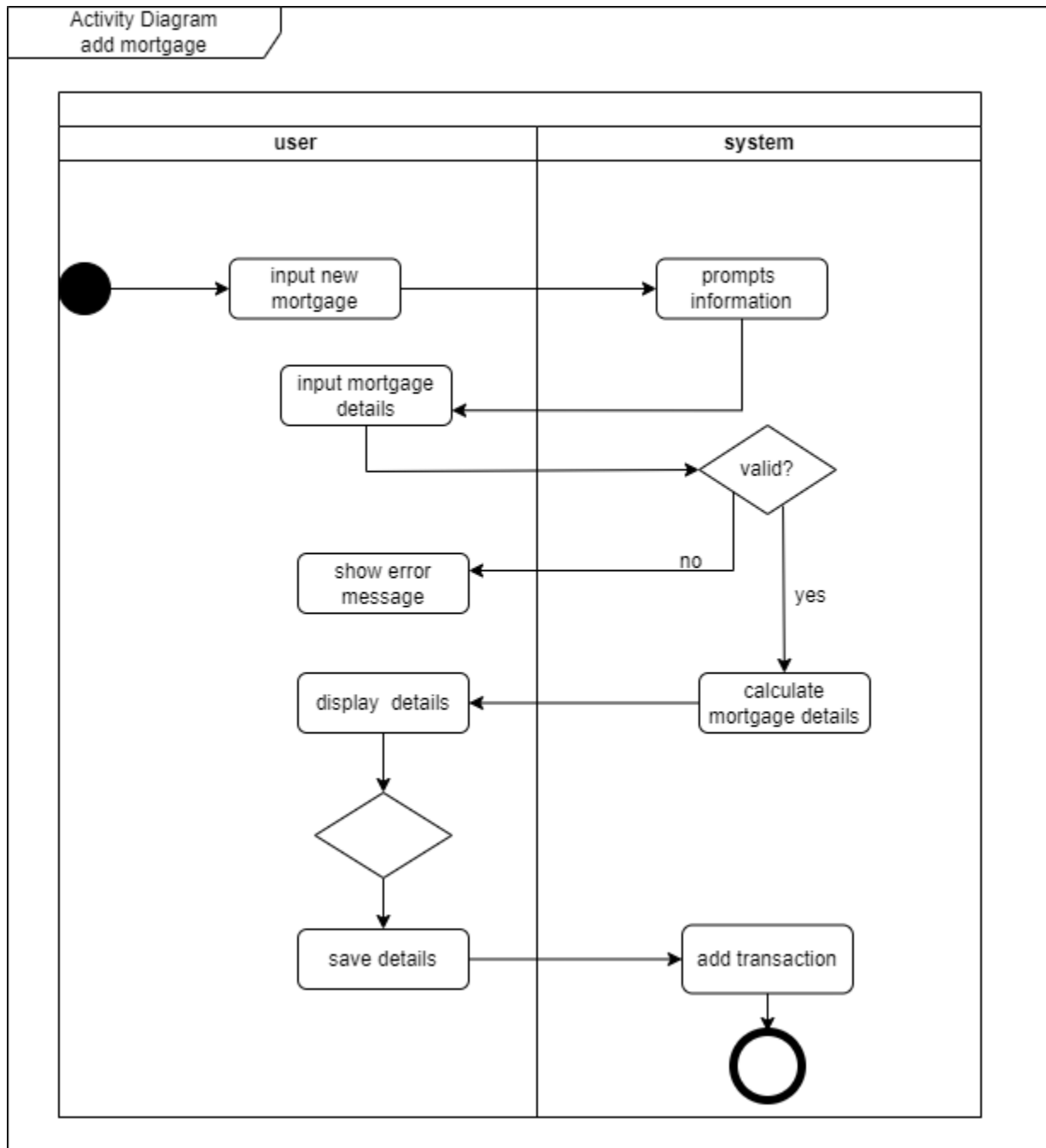
8.3 Activity Diagram

The activity diagram is the sequence of actions involved in performing tasks such as inputting data, calculating mortgage payments, viewing amortization schedules, adjusting parameters, and so on. Each activity is connected through transitions, showing the flow of control within the system.

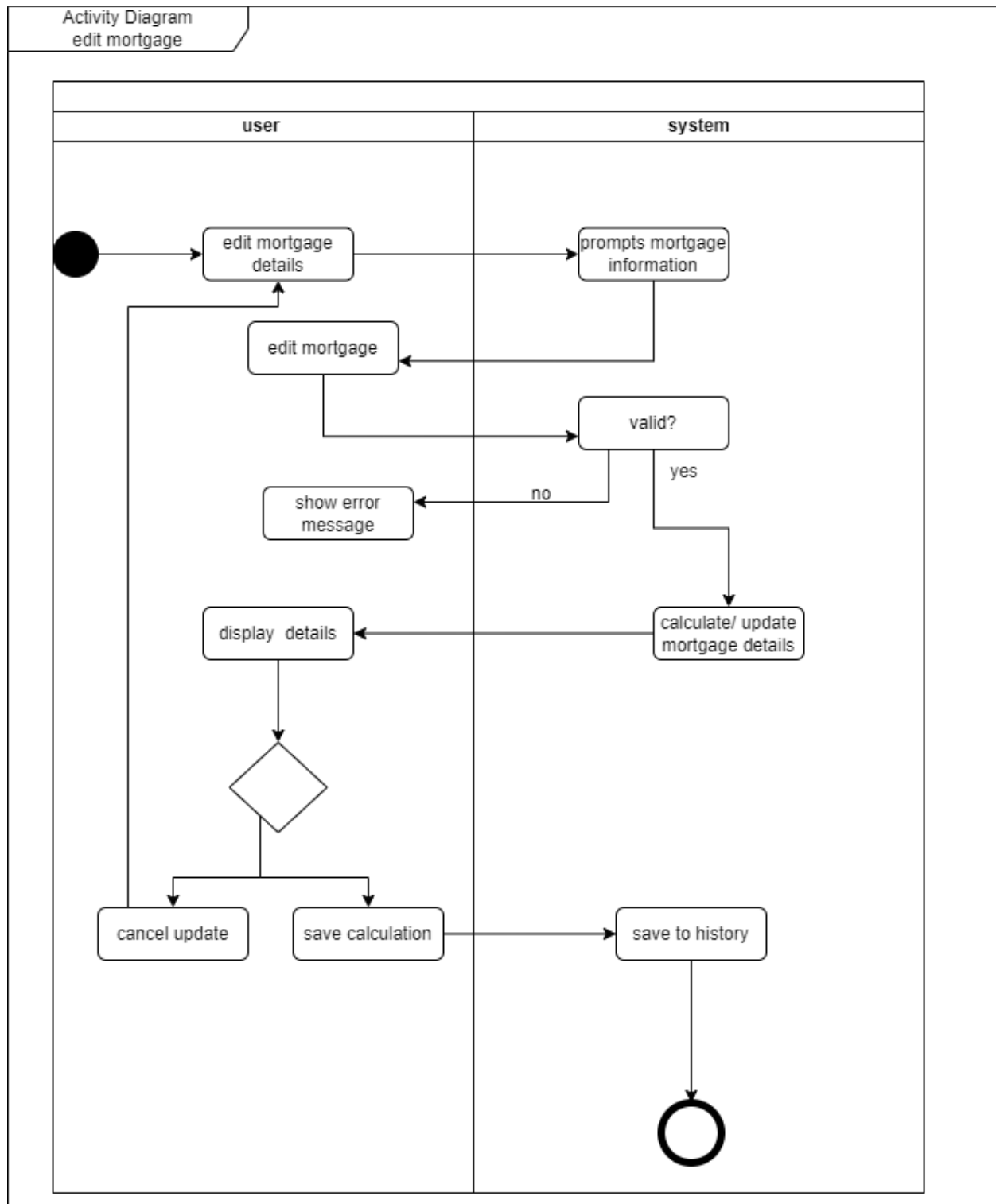
1. The user login activity diagram:



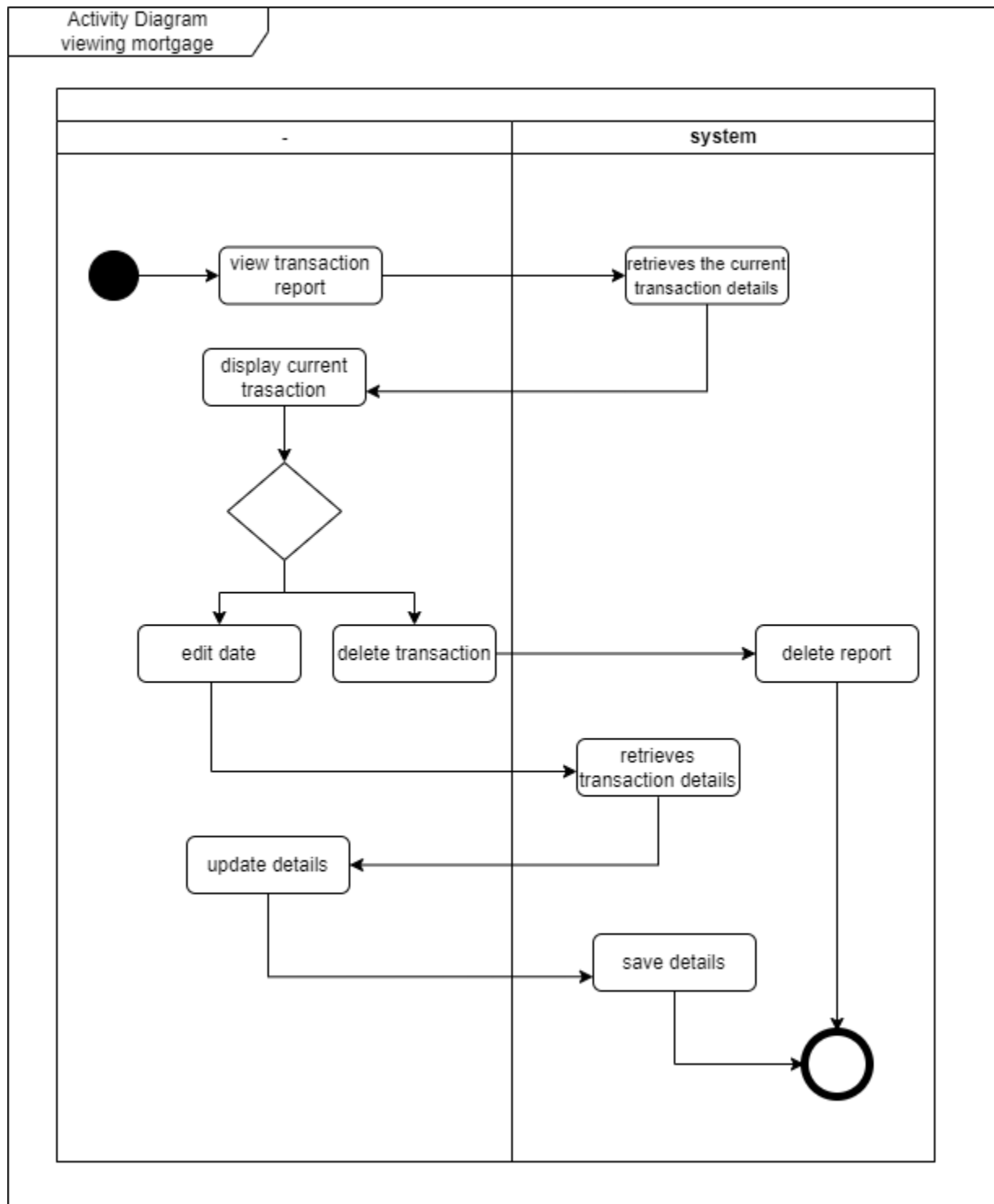
2. The user adds a mortgage activity diagram



3. The user edits the mortgage activity diagram

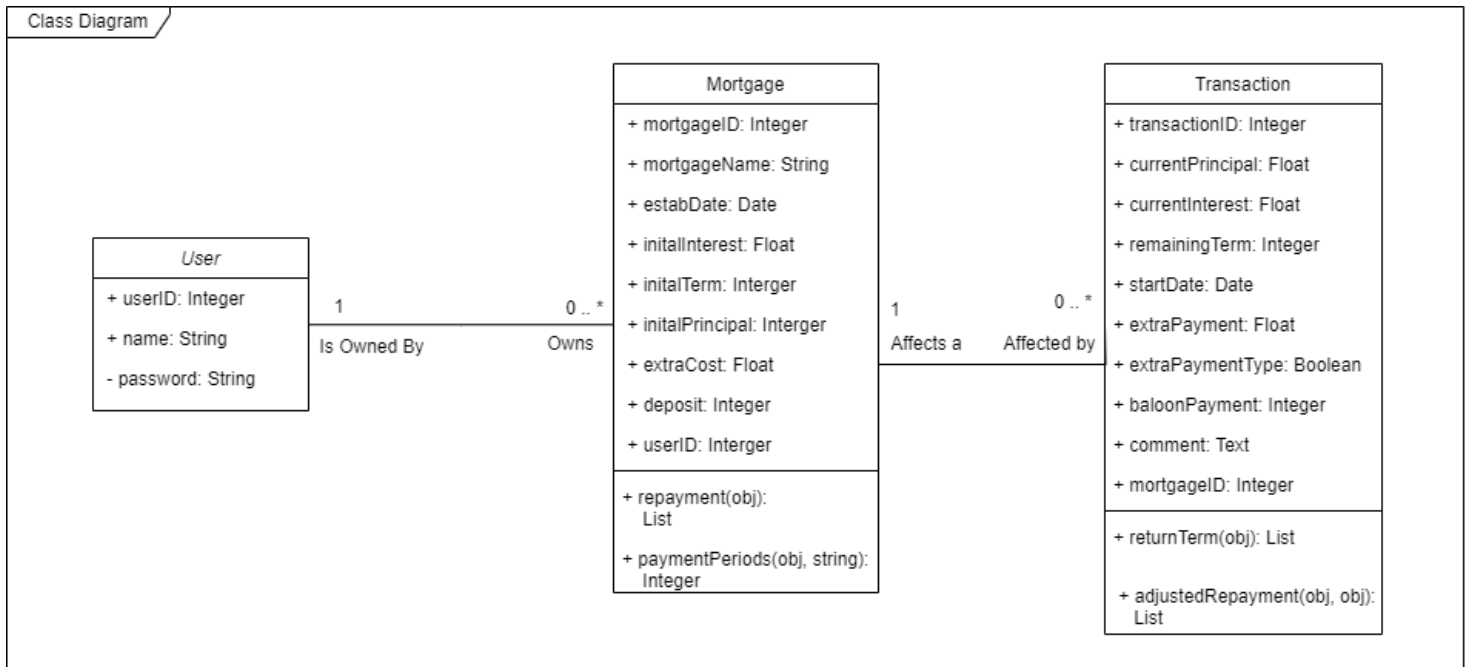


4. User' viewing mortgage activity diagram



8.4 Class Diagram

The class diagram shows the structures of the various classes used throughout the application as well as the methods that are inherent to those classes. This helps drive our object-orientated programming to work smoothly. class diagram.

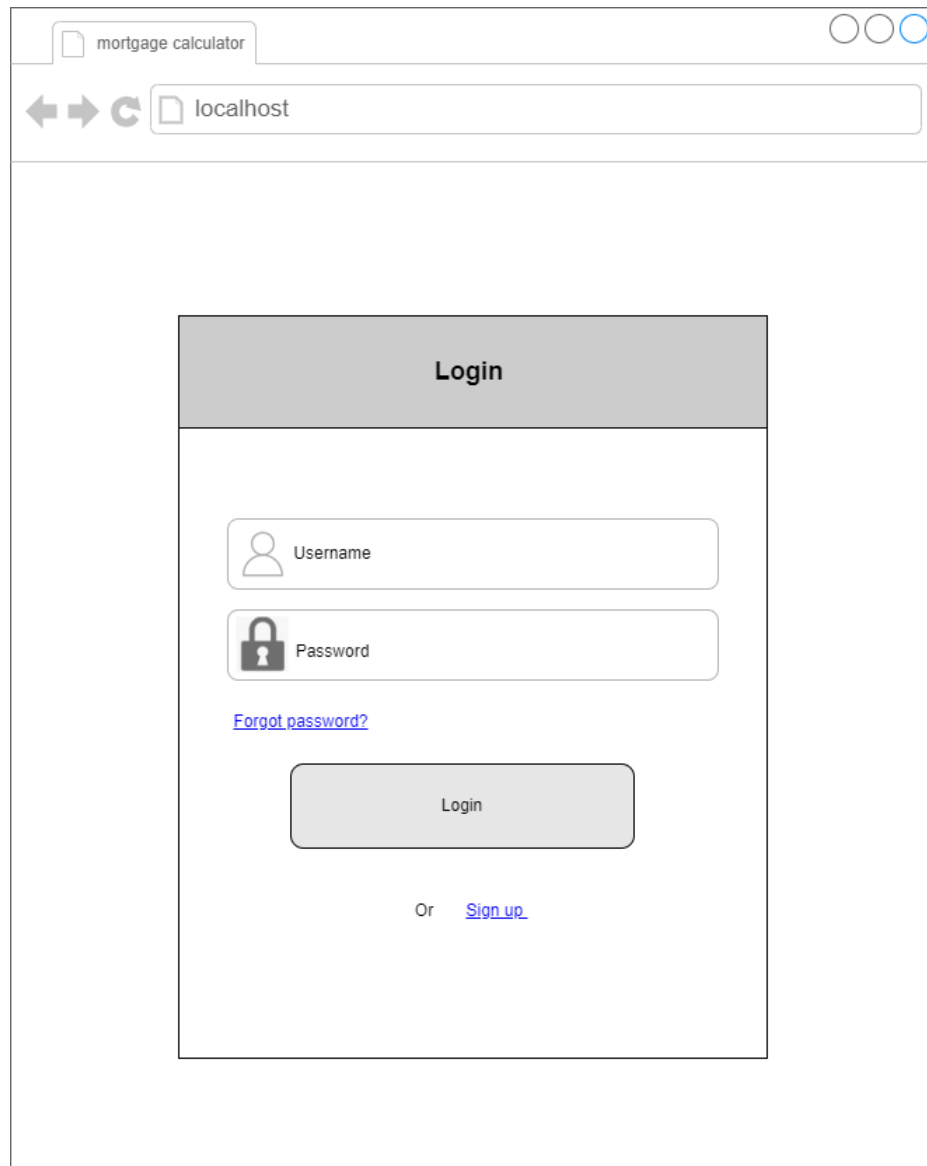


Section 9. Interface Design

We will be shaping how users interact with our system by using wireframes. These wireframes serve as skeletal outlines of our web pages, showcasing the structure and placement of elements. By leveraging wireframes, we can plan and visualize the layout of our interface.

9.1 The user login page

This login page will display before entering the calculator. This page gives users options for logging and signing up for an account.



9.2 The signup page

On this page, users can sign up for an account by providing a username and password.

The image shows a wireframe of a web browser window. The browser has a single tab labeled 'mortgage calculator' and the address bar shows 'localhost'. The main content area displays a 'Sign up' form. The form is centered and consists of a title bar 'Sign up', a back button labeled '< Back', and three input fields: 'Enter your user name', 'Create a password', and 'Confirm your password'. A 'sign up' button is located at the bottom of the form.

9.3 The index (home) page

Users will be able to view their transaction reports on this page. Where users can add new mortgages, update mortgages, and delete data. This page gives the users of overall of their summary data and mortgage maturity.

Home

←

→

↺

📄

Mortgage 1

Mortgage 2

Combined

Add a new Mortgage

Update Mortgage

Remove Data

👤

Analysis Summary

From:

Today

Fortnightly

☐ Monthly

Estimated Repayment:		Full Term to Amortize:	
Interest:		Estimated Reduced Term to Amortize:	
Principal:		Interest Over Full Term:	
Extra:		Principal + Interest:	
Repayment:		Interest over Reduced Term:	
Payments Over Full Term:		Interest Saved Over Reduced Term:	
Payments Over Reduced Term:		Principal + Interest Over Reduced Term:	

Amortization Table

From:

Fortnightly

☒ Monthly

Date	Balance	Interest	Principal	Extra	Payment	Amount of Principal	New Balance	Accum Interest	Accum Principal payment

9.4 The add mortgage page

This page allows users to input a new mortgage.

mortgage calculator

localhost

< Home

Add mortgage

Mortgage:

Principal (\$):

Interest (%):

Deposit (\$):

Extra Costs (\$):

Term (Years):

9.5 The updated mortgage page

After a new mortgage is established, the user will be able to edit the mortgage on this page or when they click the update mortgage button on the index page. This page will display current mortgage details and allow users to update the information. Users can analyse override payments and add extra costs if there are any. A comment text area for comment on the costs.

mortgage calculator

localhost

< Home

Update Mortgage

Mortgage:

Select mortgage

Date:

18/04/2024

Principal (\$):

750000

Interest (%):

7%

Term: years:

28

 months:

2

Payment override (\$):

500

Monthly ☒ Fortnightly ☐

Additional costs:

200

Comment:

extra cost for administration fees and house content insurance.

Update Mortgage

Save Changes

Cancel

9.6 Deleting Data Page


If users wish to delete existing transactions or mortgages, they can access this page through the removing data button on the index page. They will see a table of both mortgages and transactions with identifying information so that they can remove exactly what they want to remove.

Remove Data

https://www.default.com

< Back

Removing Data



Transactions:

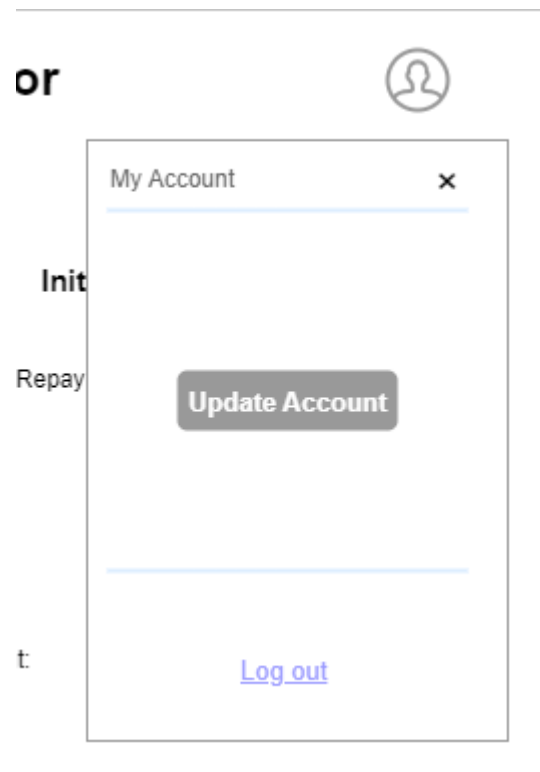
Date:	Comment	Related Mortgage	
Value	Value	Value	Delete
Value	Value	Value	Delete
Value	Value	Value	Delete
Value	Value	Value	Delete

Mortgage(s):

Mortgage Name:	Current Balance	Remaining Term:	
Value	Value	Value	Delete
Value	Value	Value	
Value	Value	Value	
Value	Value	Value	

9.7 User account icon

When a user clicks the user icon on the top right page, then a pop-up window will appear for the user to update their account information.



9.8 Update Account

After the user clicks update account from the pop-up window, a new page will appear for the user to choose whether they want to log out, delete their account or change their password by clicking the account setting button.

mortgage calculator

localhost

Update Account

User name:
Johnjamieson123

Password:

Account settings

Log out

Delete account

9.9 User account settings

When users click account settings, this page will appear for the user to change their password or if they decide not to then they can click log back to the home page.

mortgage calculator

← → ↺

localhost

Account Settings

User name:

Johnjamieson123

Old password:

Enter previous password

New password:

Enter new password

Cofirm password:

Confirm new password

Update

Go back

Section 10. Test Framework

We will outline the test framework that will be employed to ensure the quality of our project. For our project, we will cover the four levels of testing: unit testing, integration testing, system testing and user acceptance testing.

10.1 Unit Testing

The main objective is to isolate written code and determine if it works as it should, so we can detect early flaws in code. We will focus on creating unit test cases before developing the actual code by using automated unit testing to validate the functionality.

We will provide a code example here. This code defines a class Mortgage with properties and setters for the attribute principle, as well as some other values.

```
1  class Mortgage:
2      def __init__(self, mortgageID, mortgageName, estabDate, initialInterest,
3                  initialTerm, initialPrincipal, extraCost, deposit, UserID):
4          self.mortgageID = mortgageID
5          self.mortgageName = mortgageName
6          self.estabDate = estabDate
7          self.initialInterest = initialInterest
8          self.initialTerm = initialTerm
9          self.initialPrincipal = initialPrincipal
10         self.extraCost = extraCost
11         self.deposit = deposit
12         self.UserID = UserID
13
14         1 usage
15         @property
16         def principal(self):
17             return self.initialPrincipal
18
19         1 usage
20         @principal.setter
21         def principal(self, value):
22             if not isinstance(value, int):
23                 raise ValueError("Principal must be an integer")
24             self.initialPrincipal = value
25             return
```


We will use Pytest for the Mortgage class. It checks if the initialized mortgage object has the correct principal value and whether it raises a ValueError when non-numeric values are passed for initialization parameters.

```
1 import pytest
2 from mortgage import Mortgage
3
4
5 class TestStrings:
6     mortgage = Mortgage(mortgageID: 1, mortgageName: "Test Mortgage", estabDate: "16-04-2024", initialInterest: 0.05,
7                         initialTerm: 30, initialPrincipal: 100000, extraCost: 0, deposit: 20000, UserID: 1)
8
9     def test_mortgage_principal_setter(self):
10         self.mortgage = Mortgage(mortgageID: 1, mortgageName: "Test Mortgage", estabDate: "16-04-2024",
11                                 initialInterest: 0.05, initialTerm: 30, initialPrincipal: 100000, extraCost: 0, deposit: 20000, UserID: 1)
12         assert self.mortgage.initialPrincipal == 100000
13
14     def test_mortgage_principal_setter_with_string(self):
15         mortgage = Mortgage(mortgageID: 1, mortgageName: "Test Mortgage", estabDate: "16-04-2024",
16                             initialInterest: 0.05, initialTerm: 30, initialPrincipal: 100000, extraCost: 0, deposit: 20000, UserID: 1)
17         with pytest.raises(ValueError):
18             mortgage.principal = "test"
19
```

An example in the Pytest will show you the results for the class.

```
===== test session starts =====
collecting ... collected 1 item

test_mortgage.py::TestStrings::test_mortgage_principal_setter PASSED [100%]

===== 1 passed in 0.01s =====
```

```
===== test session starts =====
collecting ... collected 1 item

test_mortgage.py::TestStrings::test_mortgage_principal_setter_with_string PASSED [100%]

===== 1 passed in 0.02s =====
```

Test scenarios for each test area:

In the test scenario here we have seven test areas that cover the unit's functionality, behaviour, and interactions to ensure its correctness and reliability within our project.

1. Mortgage

Testing Type	Test Area	Test scenario	Test ID	Status	Cause of test failure
Unit Testing					
	Mortgage				
		mortgageName is too long	UT_mortgage_01		
		estabDate is in the wrong format	UT_mortgage_02		
		estabDate contains invalid values	UT_mortgage_03		
		initialInterest is not a float	UT_mortgage_04		
		initialInterest is less than 0	UT_mortgage_05		
		initialTerm is not an Integer	UT_mortgage_06		
		initialTerm less than or equal to 0	UT_mortgage_07		
		initialPrincipal is not an Integer	UT_mortgage_08		
		initialPrincipal is less than or equal to 0	UT_mortgage_09		
		extraCost is not a Float	UT_mortgage_10		
		desposit is not an Integer	UT_mortgage_11		
		userID is not a Integer	UT_mortgage_12		
		userID is equal or less than 0	UT_mortgage_13		
		paymentPeriods string argument isnt a string	UT_mortgage_14		
		paymentPeriods string argument isnt "monthly" or "fornightly"	UT_mortgage_15		

2. User

User				
	name is not a string	UT_user_01		
	name is too long	UT_user_02		
	password is too short	UT_user_03		
	password is too long	UT_user_04		

3. Analysis

Analysis				
	analysis isnt given a mortgage object	UT_analysis_01		
	transaction check returns nothing	UT_analysis_02		
	there are no transactions	UT_analysis_03		

4. Amortization

Amortization				
	amortization isnt given an analysis	UT_amortization_01		
	amortization isnt given a mortgage	UT_amortization_02		
	transaction check returns nothing	UT_amortization_03		
	there are no transactions	UT_amortization_04		

5. Transaction

Transaction			
	currentPrincipal is not a float	UT_transaction_01	
	currentPrincipal is less than or equal to 0	UT_transaction_02	
	currentInterest is not a float	UT_transaction_03	
	currentInterest is less than 0	UT_transaction_04	
	remainingTerm is not an Integer	UT_transaction_05	
	remainingTerm is less than 0	UT_transaction_06	
	startDate is in the wrong format	UT_transaction_07	
	startdate contains invalid values	UT_transaction_08	
	extraPayment is not a float	UT_transaction_09	
	extraPayment is less than 0	UT_transaction_10	
	extraPaymentType is not boolean	UT_transaction_11	
	comment is not a string	UT_transaction_12	
	comment is too long	UT_transaction_13	
	mortgageID is not an integer	UT_transaction_14	
	mortgageID is less than or equal to 0	UT_transaction_15	
	adjustedRepayment is not given a mortgage	UT_transaction_16	

6. Graphing

Graphing			
	graphing isn't given an analysis	UT_graphing_01	
	graphing isn't given a mortgage	UT_graphing_02	
	transaction check returns nothing	UT_graphing_03	
	there are no transactions	UT_graphing_04	

7. dbHandler

dbHandler			
	database returns an error	UT_dbhandler_01	
	database check doesn't return the expected result	UT_dbhandler_02	

10.2 Integration Testing

Where individual components or modules of a system are combined and tested as a group. The purpose of integration testing is to ensure that the interactions between these components work as expected and that the integrated system functions correctly as a whole.

For a small and straightforward project like a Mortgage Calculator, we will integrate all the components/modules and test the entire system, using the big bang integration testing approach. This approach is straightforward, as the interactions between components are relatively simple and easy to manage. Our project will use manual testing through the Pytest framework.

Test scenarios:

We have eight test areas that need to be tested.

Integration Testing				
	ModelsdbHandler intergration	dbHandler can access models	IT_ModelsdbHandler	
	ModelsAnalysis intergration	Analysis can access models	IT_ModelsAnalysis	
	AnalysisAmortization intergration	Amortization can access Analysis	IT_AnalysisAmortization	
	AnalysisGraphing intergration	Graphing can access Analysis	IT_AnalysisGraphing	
	AnalysisInterface intergration	Interface can access Analysis	IT_AnalysisInterface	
	AmortizationInterface intergration	Interface can access Amortization	IT_AmortizationInterface	
	GraphingInterface intergration	Interface can access Graphing	IT_GraphingInterface	
	InterfaceModels intergration	Models can access Interface	IT_InterfaceModels	

10.3 System Testing

Testing will be done to see if our software can be integrated correctly with external systems that are used to make our software function properly. In this case, our application won't be working with too many external systems and will be easy to test if they connect properly.

Test scenarios:

We have two test areas that need to be tested.

System Testing				
	Calculator Database connection	Mortgage Calculator can connect to the Database	ST_CalculatorDatabase	
	Calculator Web Browser connection	Mortgage Calculator can load on a Web Browser	ST_CalculatorBrowser	

10.4 User Acceptance Testing

User Acceptance Testing will be done to ensure that when a potential user of our software does an action in our application the correct response is shown to the user. These responses could be errors, new pages, confirmation windows and more.

Test scenarios:

In the UAT test scenario here we have seven pages that need to be tested.

1. The log-in page

Login Page			
	User enter username but not a password	UAT_Login_01	
	User enters username but wrong passord	UAT_Login_02	
	User enters a correct password but no username	UAT_Login_03	
	User enters a correct password but the wrong username	UAT_Login_04	
	User enters nothing	UAT_Login_05	
	User wants to go to the register page	UAT_Login_06	

2. The register page

Register Page			
	User enters too long of a username	UAT_Register_01	
	User enters no name	UAT_Register_02	
	User enters too short of a password	UAT_Register_03	
	User enters too long of a password	UAT_Register_04	
	Passwords do not match	UAT_Register_05	
	User doesn't enter a password	UAT_Register_06	
	User wants to return to the login page	UAT_Register_07	

3. The index page

Index page			
	User wants to see a different Mortgage	UAT_Index_01	
	User wants to adjust the analysis start date	UAT_Index_02	
	User wants to go to create a new mortgage	UAT_Index_03	
	User wants to go to update a mortgage	UAT_Index_04	
	User wants to go to remove data	UAT_Index_05	
	User wants to go to their account settings page	UAT_Index_06	
	User wants to logout	UAT_Index_07	

4. The removing data page

Removing Data Page			
	User wants to delete a transaction	UAT_RemovingData_01	
	User wants to delete a mortgage	UAT_RemovingData_02	
	User wants to go to their account settings page	UAT_RemovingData_03	
	User wants to logout	UAT_RemovingData_04	
	User wants to return to the index page	UAT_RemovingData_05	

5. The add mortgage page

Add Mortgage Page			
User doesn't enter a name	UAT_AddMortgage_01		
User enters invlaid data in the name	UAT_AddMortgage_02		
User doesn't enter a principal	UAT_AddMortgage_03		
User enters invalid data in the principal	UAT_AddMortgage_04		
User doesn't enter an interest	UAT_AddMortgage_05		
User enters an interest in the incorrect format	UAT_AddMortgage_06		
User enters invalid data in the interest	UAT_AddMortgage_07		
User doesn't enter a deposit	UAT_AddMortgage_08		
User enters invalid data in the deposit	UAT_AddMortgage_09		
User doesn't enter an extra cost	UAT_AddMortgage_10		
User enters invalid data in the extra costs	UAT_AddMortgage_11		
User doesn't enter a term	UAT_AddMortgage_12		
User enters invalid data into the term	UAT_AddMortgage_13		
User clicks calculate mortgage without all needed data being valid	UAT_AddMortgage_14		
User clicks calculate mortgage with all needed data being valid	UAT_AddMortgage_15		
User clicks save mortgage without all needed data being valid	UAT_AddMortgage_16		
User clicks save mortgage with all needed data being valid	UAT_AddMortgage_17		
User wants to go to their account settings page	UAT_AddMortgage_18		
User wants to logout	UAT_AddMortgage_19		
User wants to return to the index page	UAT_AddMortgage_20		

6. The updated mortgage page

Update Mortgage Page			
User doesn't enter a date	UAT_UpdateMortgage_01		
User enters invalid data in the date	UAT_UpdateMortgage_02		
User doesn't enter a principal	UAT_UpdateMortgage_03		
User enters invalid data in the principal	UAT_UpdateMortgage_04		
User doesn't enter an interest	UAT_UpdateMortgage_05		
User enters invalid data in the interest	UAT_UpdateMortgage_06		
User doesn't enter a year or month in the term	UAT_UpdateMortgage_07		
User doesn't enter a year in term but enters data in the months	UAT_UpdateMortgage_08		
User doesn't enter a year in term but enters invalid data in the months	UAT_UpdateMortgage_09		
User enters data in the year but doesn't in the months	UAT_UpdateMortgage_10		
User enters invalid data in the year but doesn't in the months	UAT_UpdateMortgage_11		
User doesn't enter a payment override	UAT_UpdateMortgage_12		
User enters a payment override that is less than the current repayment	UAT_UpdateMortgage_13		
User doesn't enter additional costs	UAT_UpdateMortgage_14		
User enters invalid data into additional costs	UAT_UpdateMortgage_15		
User doesn't enter a comment	UAT_UpdateMortgage_16		
User enters a comment that is too long	UAT_UpdateMortgage_17		
User clicks update mortgage without all needed data being valid	UAT_UpdateMortgage_18		
User clicks update mortgage with all needed data being valid	UAT_UpdateMortgage_19		
User clicks save changes without all needed data being valid	UAT_UpdateMortgage_20		
User clicks save changes with all needed data being valid	UAT_UpdateMortgage_21		
User wants to go to their account settings page	UAT_UpdateMortgage_22		
User wants to logout	UAT_UpdateMortgage_23		
User wants to return to the index page	UAT_UpdateMortgage_24		

7. The account setting page.

Account Settings Page			
	User incorrectly enters old password	UAT_AccountSettings_01	
	User's new password is too short	UAT_AccountSettings_02	
	User's new password is too long	UAT_AccountSettings_03	
	Confirm password doesn't match with the new password	UAT_AccountSettings_04	
	User clicks update account without all needed data being valid	UAT_AccountSettings_05	
	User clicks update account with all needed data being valid	UAT_AccountSettings_06	
	User wants to return to the previous page	UAT_AccountSettings_07	
	User wants to logout	UAT_AccountSettings_08	

Conclusion

This document outlines our plan for building the mortgage calculator, which comes from feedback we got in a recent meeting with our client. We carefully looked at the business and program details, thinking about the problems we faced and the solutions we suggested. By really digging into the program details, we made sure we understood exactly what the project needed. When we were designing it, we focused on three main things: how the system works, how the data is organized, and how people will use it.

We included diagrams and sketches to show how the software will work, what it's made of, and who will use it. These pictures help explain what the mortgage calculator does, what it can do, and who it's for. We set up a way to test the project to make sure it's good quality and dependable. We've got different tests to check that the mortgage calculator works right. By testing it a lot, we hope to find and fix any problems or mistakes, making the software better overall.

This document is like a roadmap for making the mortgage calculator. We used feedback from our client and what we learned from the program details to plan it out. With clear designs, user-focused needs, and thorough testing, we want to create a tool that makes managing mortgages easier for our client. On top of the design stuff, we wrote a document about what users want. It looks at things from their point of view, spelling out what they expect and need. By listening to users and including their thoughts in our work, we want to make a mortgage calculator that works well.