|  |  |
| --- | --- |
|  | Documentation of LoanAPound Design Demo |

Contents

[Outline 1](#_Toc500067763)

[Scope 1](#_Toc500067764)

[Data Model 2](#_Toc500067765)

[Implementation Overview 3](#_Toc500067766)

[Implementation Details 3](#_Toc500067767)

[Process A Loan Explained 4](#_Toc500067768)

[Testing 4](#_Toc500067769)

[Running the Demo 4](#_Toc500067770)

# Outline

The purpose of this document is to provide an overview of a draft solution to the LoanAPound Loan Approval System. It should be used for discussion and to clarify the client’s requirements. The current specification is still vague, and no development would normally be undertaken until clarity has been provided on the requirements. However, I’ve provided a sample implementation to meet the user stories, as this is a test.

# Scope

This will look at how how Stories 1- 8 of the Fairsail Technical Test document can be met.

Some suggested functionality for the Loan Engine has been implemented in order to focus the discussion.

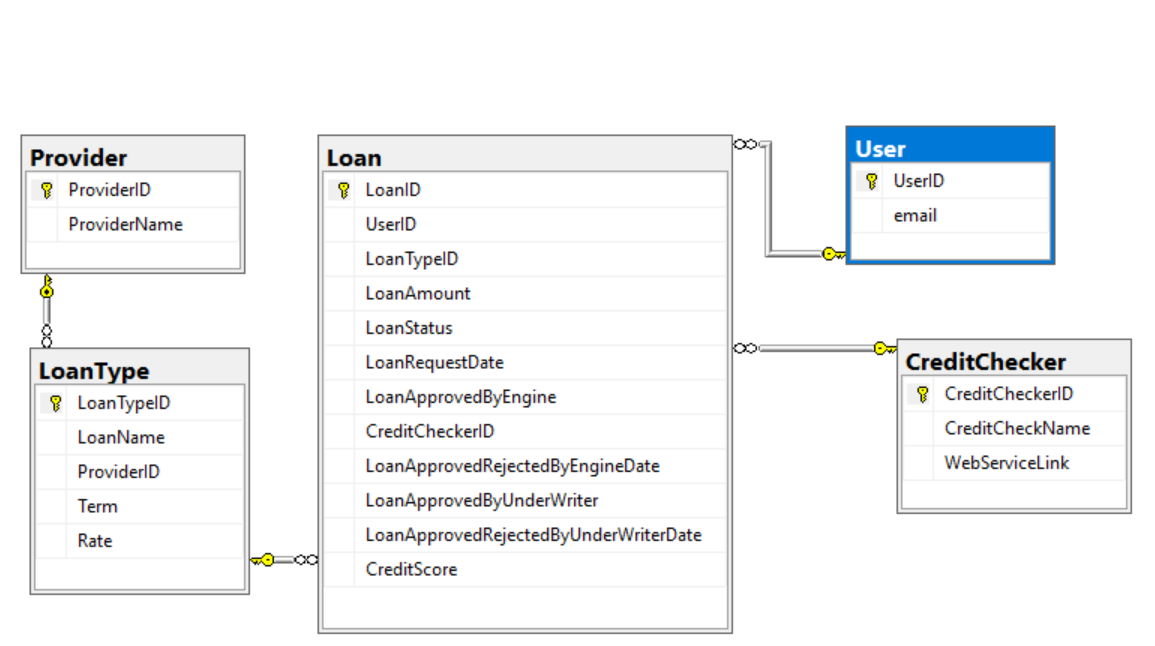
In order to maintain this focus, the Frontend and backend storage have largely been ignored. A data model is provided (main objects only), but for the implemented code only an in-memory, non-persistent data repository is provided.

As the stories are unclear about UI requirements, the functionality can only be run thru unit tests.

Tests are used to prove basic functionality works as intended. There is no attempt at high test coverage, as validation and object properties are far from being ascertained.

# Data Model

Below is a suggested data model to meet the primary requirements of the stories. Other support tables will be required, and it is expected that the fields and relationships will change on better understanding of the requirements.



The model will be discussed below in relation to the User Stories:

1. **As a LoanAPound administrator, I want to be able to setup different types of loans with criteria such as (but not limited to) term/years, borrowing rate, provider…**

The LoanType table holds this data. It is envisaged that loan types will vary by provider, term and rate.

1. **As a loan applicant, I want to be able to apply for a loan online so that I get a decision on money being lent to me or not**

The system needs to be able to create users and user roles. (user roles and other supporting tables have not been included above).

A logged-in user will be able to select a LoanType, and an application will result in a record in the Loan table with the Users ID, the LoanType ID, the date of the request, and the amount requested.

1. **As a loan engine, I want to get a credit score from a third party system, so that it can be used to inform whether or not the applicant will get his loan accepted or rejected**

The Loan Engine can retrieve a list of unprocessed loans by looking at the status in the LoanTable.

A set of rules will determine which CreditChecker should be used for the check.

(These rules are currently held within the application. If these vary frequently then perhaps they could be held in the database instead).

Once a Credit Checker has been chosen, a call out to the 3rd Party is required. For now, this has been assumed to be some sort of web service request, but this may not be correct. It may be that each CreditChecker has a different set of fields and this may have to be catered for.

There will be further rules to determine from the returned credit score whether this is pass or fail. Each CreditChecker may have different ways of representing a Credit Score, and determining whether this is pass or fail may also be dependent on credit agency.

The credit score, along with the approval/rejection and date are added to the Loan table, and the status updated.

1. **As a loan engine, I want to be able to implement multiple credit score third party system, so that I can chose to use any of them based on various creiteria**

As above.

1. **As loan engine, I want to be able to approve the loan or not based on the credit score&loan amount so that I can refer it to the underwriter decision or not**

As above. The underwriter can review any loans that have reached a certain status.

1. **As an applicant, I want to be able to view online the progress of my application**

From the Loan Table the applicant can view status of any applications he/she has made.

1. **As an underwriter, I want to see a queue of pending applications so that I can approve or reject them**

The underwriter can review any loans that have reached a certain status.

1. **As a loan engine, I want to notify by email the applicant of the progress of its application, so that he is aware of what is happening**

This can either be done as soon as the credit check is done, or a poll of the database of records where status has changed can be used to send emails.

# Implementation Overview

Wherever possible, interfaces are used so that concrete classes can be swapped in/out as required. This aids with testing. It is also essential to be able to provide concrete instances of specific rules and specific 3rd parties, where the logic varies.

Factories have been provided for injecting classes.

# Implementation Details

Three ‘controller’ classes have been implemented (Another may be required for Underwriter). Specific Authorisation attributes can be added to these to control access.

Any UI would make requests via the ‘controllers’.

The controllers currently merely call in to the Business Logic Layer (BLL). However, as mentioned above, some authorisation could take place here.

There is a Data Access Layer (DAL) which holds the data model and, for this demo, a TestRepository.

## Process A Loan Explained

(See ProcessALoan in BLLLoanAPound)

Given a loan, a credit check needs to take place. Different sets of rules may apply to determining which Credit Checker to use.

CheckCreditRuleCreator.Create will have some way of determining which rule applies.

(Currently, this is hardcoded to be based on loan amount. Such a process to swap rules in or out may or not be required)

The chosen rule will determine which credit checker to use.

The demo provides 2 hardcoded sample rules: one based on Term, the other based on rate.

Once the CreditChecker has been selected, a way of sending a credit check request to that 3rd party needs to be provided.

It is assumed that this is likely to be via a call to a 3rd party web service, but other methods may be necessary.

Each of these will have their own authentication/authorisation demands. Requests via SOAP or REST, for example, may require different information and different processs. For example, a request may come back with just ‘Ok, We got your request’ or ‘Here is your score’.

It may be necessary to make a different call, or to wait for the 3rd party to call into our web service before we get a score.

All these variations need to be catered for.

Currently, in the demo, these are just hardcoded to return a score.

Once the credit score has been retrieved, the loan engine needs to apply some rules to determine whether the loan. should be approved.

Currently a rule has been hard coded.

The Loan Table can then be updated with the status, approved/rejected and date. The applicant can then be alerted by email.

# Testing

A few tests have been added to exhibit and test the basic behaviour of each call to the ‘controller’ classes.

No attempt has been made to validate input as this needs to be determined.

# Running the Demo

No attempt has been made to provide an installer.

The easiest way to run the method calls is via the test.

In Visual Studio, on the Test Menu, click Run All Test.

Any problems with this, please email the author at: [ged.marsden@hotmail.co.uk](mailto:ged.marsden@hotmail.co.uk)