Project Proposal

Income Verification Application for Corporate Premises Rental

Daniel O’Driscoll, 19219318, x19219318@student.ncirl.ie

Higher Diploma in Science in Computing

Web Development

Date : 7/6/2022

**Objectives**

The objective of this project is to develop an income verification application for use by rental agencies focusing towards business premises rental. A number of key features will be implemented in order to achieve this objective:

1. Allow the rental agencies to advertise currently available properties.
2. Allow the potential customer to build a user profile.
3. Allow for the customer to perform an income verification step which will act as proof of income for the rental agency.
4. Allow for the customer to submit their user profile and apply for advertised properties.

**Background**

Currently proof of ability-to-pay is typically provided in the form of bank statements in PDF form, provided by the potential customer, to the landlord/rental agency upon seeking to rent/buy .Here, the customer’s monthly income will be listed which is then compared by the landlord rental agency against the cost of the monthly rent (account balance at the end of the month may also be taken into account – though typically not specified directly on rental advertisements).

There are a number of issues with this, both from the perspective of the rental agency as well as the customer. From the customer’s perspective, this represents a potential breach of privacy as while monthly income will be listed (proof of which is a valid requirement), all additional transactions are also listed – the nature of which is irrelevant to the rental requirements.

While letting agencies will typically adhere to GDPR regulations there is little oversight as to how this information is stored, how long it is stored for, and who will have access to it after it has been sent to the letting agencies. While larger organization may have a more robust strategy in place, it is likely the smaller agencies do not have as strict protocols when it comes to handling customer data. This becomes additionally problematic for landlords who do not wish to let through rental agencies and act as sole-traders – over which there is comparatively less oversight.

Due to the move towards a cashless society, the number of monthly card transactions can be significant. This represents a potential hurdle for customers who value privacy to removing transaction information by redacting unnecessary information. The volume of transactions listed on customer bank statements can also be an issue for rental agencies. As statements are typically provided as PDFs, agency employees are required to manually work through the statements in order to validate that the potential tenant is capable of making the required rental payments. This is an inefficiency which becomes particularly apparent where there are a large number of equally desirable tenants. As a result of the increased workload, due diligence may be lacking or only performed on a selective basis.

**Technical Approach**

Requirements of the application

There will be a central login page where a user can login either as a rental agency or as a renter. Depending on the choice of login, the user will be brought to one of two distinct pages. Login details (email and password) will be listed in a database with separate “agency” and “user” tables – ensuring that only registered users in each case can access the appropriate information.

Main Landing Page

* Key Functionalities

1. Ability to search through different adds listed on the site.
2. Linked to different property categories.
3. Login link for both rental agencies as well as renters.

Renter Landing Page (Renter has already logged in)

* Key Functionalities

Layout similar to the main landing page with the addition of

1. Link to user profile.

User Profile Page

* Key Functionalities

1. Ability to Upload Documents (references, bio etc.).
2. Ability to Upload Images (photo ID).
3. Link to financial verification page (in order to obtain verification cert).
4. Ability to perform the above for multiple people in a single application.

Rental Ad Browsing Page

* Key functionalities

1. Ability to scroll through listed adds.
2. Ability to filter adds based on a number of different criteria (price, location etc.).

Individual Ad Page

* Key functionalities

1. Ability to scroll to view all listed information regarding property.
2. Ability to view photos of property both on screen and as part of carousel.
3. Listed requirements.
4. Ability to apply for apartment (onclick – confirmation page about wishing to apply for property).

Renter Application Testing

* Key Functionalities

1. Test whether the income verification cert is valid based on the income specified in the individual add page.
2. If the test is passed – the user will be prompted that their application is successful. The user will be asked whether they want to return to the home page or continue looking at property adds.
3. If the test is failed, a message is displayed saying that the application was not successful due to a cert validation failure.

Agency Landing Page (Rental Agency has logged in)

* Key functionalities

1. Link to creating a new property ad.
2. View currently listed ads.
3. Ability to remove specific ads.

Creating New Property Ad Page

* Key functionalities

1. Ability to upload photos.
2. Ability to upload description.
3. Set minimum monthly income requirement + buffer (this will be the value that the renter income verification cert will be tested against).
4. Set requirements for the user profile i.e Bio, references required etc. The user will be able to submit an application will these requirements missing.

Review Current Listed Ad Page

* Key functionalities

1. Ability to view to the number of total applications for an apartment.
2. Ability to filter applications based on the number of requirements met.
3. Ability to view individual applications.

Graphical user interface

Description automatically generated

Figure 1: Site Map. Users are brought to one of two distinct sub-trees of pages depending on login information.

**Special Resources Required**

In order to link safely and securing to individual bank accounts. The Plaid API will be used in order to retrieve financial information from the user user’s account. Plaid is a company specialising in the secure retrieval of financial information from linked accounts and is compatible with bank accounts from 11,500 institutions.

**Project Plan**

**Work Package 1: Data acquisition**

Tasks

T1.1 – Test development for financial record pull. Develop a unit test to ensure that the API call is successful and json data returned.

T1.2 – Pulling financial records using Plaid API (refactoring if necessary, depending on test outcome).

T1.3 – Test development for record sorting. Develop a unit test to ensure that records are correctly sorted into income and expenditure categories.

T1.4 – Sorting financial records into income and expenditure categories (refactoring if necessary, depending on test outcome).

T.5 – Test development for token creation. Develop a unit test to ensure that a token is created reflecting the net monthly income of the customer.

T.6 – Creation of a financial verification token reflecting net monthly income obtained from sorted financial records (refactoring if necessary, depending on test outcome).

Milestones

M1.1 – Successfully pulled full financial records from bank account.

M1.2 – Records successfully sorted.

M1.3 – Financial verification token successfully created reflecting results of verification step.

**Work Package 2: Database Creation**

Tasks

T2.1 – Create relational database with “User”, “Agency”, “Property\_Ad” tables.

Milestones

M2.1 – Relational database table created for application.

**Work Package 3: Site Architecture**

Tasks

T3.1 – Page creation according to listed pages above.

T3.1 – Establish routing between related pages.

Milestones

M3.1 – Skeleton site created with successful routing established between related pages.

**Work Package 4: Database Management**

Tasks

T4.1 – Test development. Develop a unit test to ensure that each crud operation has the intended effect on database tables.

T4.2 – CRUD operations – user table.

T4.3 – CRUD operations – agency table.

T4.4 - CRUD operations – property ad table.

Milestones

M4.1 – CRUD operations for manipulating database tables are implemented.

**Work package 5 – Login Functionality**

Tasks

T5.1 – Implement a login system with redirection to correct pages with successful login.

Milestones

M5.1 Successful login system implemented.

**Work package 6 – Site Design**

Tasks

T6.1 – Design wireframes for each of the site’s pages.

T6.2 – Apply wireframe designs to each of site’s pages.

Milestones

M6.1 – Successful implementation of wireframe designs.

**Work Package 7 – Report Writing**

Tasks

T7.1 – Continuous documentation of progress.

Milestones

M7.1 – Completed report.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **WP** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **11** | **12** | **13** | **14** | **15** | **16** | **17** | **18** | **19** | **20** | **21** | **22** | **23** | **23** | **25** | **26** | **27** | **28** | **29** | **30** | **31** | **32** | **33** | **34** | **35** | **36** | **37** | **38** | **39** | **40** |
| **Data Acquisition** | T1.1 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | T1.2 |  | M1.1 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  | T1.3 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  | T1.4 |  |  | M1.2 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  | T1.5 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  | T1.6 |  |  | M1.3 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **Database Creation** |  |  |  |  |  |  |  |  |  |  |  |  |  |  | T2.1 | M2.1 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **Site Architecture** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | T3.1 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | T3.2 M3.1 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **Database Management** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | T4.1 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | T4.1 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | T 4.2 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | T4.3 |  |  |  | M4.1 |  |  |  |  |  |  |  |  |  |  |
| **Login Functionality** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | T5.1 |  | M5.1 |  |  |  |  |  |  |  |
| **Site Design** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | T6.1 |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | T6.2 |  |  |  |  | M6.1 |
| **Report Writing** |  |  |  |  | T7.1 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | M7.1 |

Figure 2: Project Plan Gantt Chart. 40 days are allocated for the completion of the project (including report writing) and additional week is left as a contingency in the case of unexpected delays during the completion of any of the listed tasks.

**Technical Details**

The key functionality of this application is the income verification step used to determine the customer’s ability to meet the monthly repayments set by the rental agency for the property. In order to successfully integrate this functionality into the site, the Plaid API is used. The Plaid API specializes in allowing a client side application to pull financial records from a linked bank account. In order to ensure that this is done securely the client-side application does not communicate with to the plaid service directly as Plaid cannot verify the authenticity of the client-side application without the API secret key provided to the client. Standard security protocols dictate that the secret API key is not allowed to be stored on a client-side application. Instead, rather than the client side application making a request to the Plaid service, the application server will make the initial contact. In this server to server call, both the client ID and the secret API key can be passed securely – allowing the Plaid Service to verify that the request from the client is genuine.

This process takes place in a number of steps.

Step 1 - The client slide application sends a request to the application server for something known in the Plaid environment as a “Link Token” - a one-time password that is valid for about 4 hours.

Step 2 – The application server requests a link token from the Plaid service, sending across the unique client ID and secret API key in the process. This link token is then provided by the Plaid service once the information has been verified and passes that token back to the client application via the client server.

Step 3 – The client application then passes the link token to the Plaid service in order to verify authenticity and sends a request to link a financial account. The user then signs into their bank account (taking the user to the bank’s login page directly) in order to share data with the client application.

Step 4 – Specific requests are then made to the financial institution via Plaid in order to pull recent transactions. In this case of this project, this will consist of a GET request for the previous 12 months transactions. By default, the plaid service will hold this information as long as the account is linked. In order for the client-side application to be able to use this data an additional step is implemented in order to facilitate this without the need for the customer to login to the application every time a request is a made.

Step 5 – Plaid generates a short-lived 30 mins token known as a “Public Token” and returns it to the client application. Once the client application receives the “Public Token” it is sent to the client server. The client server then sends this “Public Token” to the Plaid Service. The plaid service then verifies that the public token is valid and was designated to be used with the client application and has not been used before. The plaid service will then generate an ”Access token” uniquely associated with that linked bank account and send that back to the application server so that the client will be able to access the financial information in the future. Once an Access token for a linked account has been generated, the client application will be able to access the pulled financial records directly without requiring the user to login in repeatedly.

In this case of the application to developed here, repeated access to customer’s financial data will not be required. As an additional step, once the financial records have been pulled and analysed for net-monthly income, an “Income Token” will be generated within the application which corresponds to the user’s payment capacity for a monthly rental. The “Access Token” generated by Plaid will then be deleted, resulting in the bank account no longer being linked and requiring the user to re-enter their login details if they wish to re-verify their income at a later stage.

This “Income Token” will then be stored as part of the user profile in the application server and used as part of the income verification step when applying to an property advertisement.

**Development Languages and Frameworks – MERN Stack**

MySQL – Database Development Language

As listed in the project plan above, a database with tables reflecting the “user”, “agency” and “property\_ad” will be required along with the CRUD operations required for managing the database tables. In the case of this project, the database will be created using MySQL. MySQL is table-based system for designing relational databases. This was chosen due to the developers familiarity with the language and experience in creating relational databases using the schema structure.

Express – Backend Framework

Express.js is back-end application framework used to simply the process of creating request end-points with the Node.js back-end platform. This will be used to streamline the development process .

React – Frontend Framework

For this project the react framework will be used to develop the front end of the website. React was chosen for a number of reasons. From a practical perspective, react is the most commonly used front end development framework which results in the framework being stable and well maintained. The use of reusable pieces of code in the form of components will be essential for saving time during the course of development as there are a number of overlapping aspects in this project where components can be used multiple times. The virtual DOM provided by react means that a faster, more interactive page is capable of being developed.

Additionally, the documentation for the Plaid API start-up guide is centered around React. Due to the features of the Plaid API and the necessity for security while handling sensitive financial data, integration of the API can be complicated. By using the React framework, this reduces the time required for integration which will allow for considerably quicker integration with the rest of the site.

Node – Backend Platform

For this project Node will be used for backend development. This was done for a number of reasons. Firstly, Node is the most commonly used backend development framework. As such, there are many well maintained libraries available for implementing different functionalities to the site. Additionally, as React is also javascript based – it provides a smoother development process by disregarding the needs to switch between languages. The Plaid API server documentation is also specifically focused on Node.JS. As this is an essential part of the website functionality, it will be necessary to make this integration process as smooth as possible.

**Evaluation**

The income verification application will be tested using the bank account and financial transactions of the developer. Two property ads, one within the price range of the developer and one outside, will be listed and applied for using the developer as an example business. The developer’s financial transactions for the previous 12 months will be pulled and an “Income Token” generated corresponding to his monthly net income. The developer will then apply for both ads ( one of which will fail and one will pass) demonstrating the income verification step.

**A pair of glasses

Description automatically generated with low confidence**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Signature of student

Date: 7/6/2022