**Capstone Project 2**

**Name: John H Morgan**

**Course: Software Engineering**

**Proposal: An estate/building management app.**

Problem

There is no system available that helps a residents’ association manage their property in conjunction with the managing agent. So much important information is held in various files in different locations or not recorded at all e.g. a list of suppliers, their contact details and maintenance contracts.

The other issue is not being aware of upcoming events such as the expiry of insurance, maintenance visits etc.

A third problem is the lack of visibility provided to apartment owners of issues they have raised.

Solution

Provide a system that allows managing agent, directors of the association and apartment owners to enter and view information. The data will be held in a database and will initially be populated manually from existing documents. The goal is that the app will be used in a live environment and is not just a project.

Tech stack

React and Node will be the chosen tech stack. React provides an attractive, flexible front-end, it is widely used in industry and so the development practice will be useful. Node and express will serve as the back-end where the routes are defined and the connections created to a PostgreSQL database.

Tech focus

It is difficult to determine whether the focus will be more front or back-end; the objective is a working, functional application that provides an impressive user experience. The work will be put in wherever it is required to deliver the end goal. Libraries will be used where it is considered they add value.

The application will be a website. The coding requirements necessary for going from website to mobile app are not fully understood. If it is a simple task of adapting the html to produce an effective mobile-sized display this will be attempted.

Objective

The goal is the provision of information; an objective which can probably be applied to the majority of apps. In this case, the focus is on the management status of a residential apartment block. A secondary goal is a means of communication. In this sense it allows apartment owners to raise issues that need a resolution.

Target users

There will be three types of users; apartment owners who have no managerial function, the directors of the company that owns the apartment block, and the employees of the company (the managing agent) appointed to manage the building on behalf of the ownership company.

Data details and functionality

Due to the nature of the application, I will be creating my own API and starting data will be entered manually. A high-level summary of the data/functionality that will be available is as follows:

* Issues raised by apartment owners – these will be tracked with dates, responses and the current status. It is also planned that filtered reports can be produced on the open issues which can be printed or downloaded.
* Statements for service charge payments – owners will be able to view the status of invoices received for shared running costs and also their payments made against these invoices.
* Payment gateway – a really nice feature to have, but likely to be impractical; the technology will still be studied.
* Company finance information such as annual accounts, annual budget, debtor reports, and actual expenditure vs. budget.
* Current expenditure showing details of paid invoices from suppliers and contractors.
* A calendar showing upcoming important events such as the expiry of insurance or maintenance visits.
* Details of suppliers and contracts.
* Admin functionality – authorise new users; remove access for those who have left.

Note: some owners have more than one apartment. The functionality will be coded to allow such a user to see details for all owned properties.

The above is the ideal scenario. Some data will require access to a third-party system which may not have an API. An example is the statement showing invoices and payments for each individual owner; something of genuine value, but which depends on accessing the finance system used by the managing agent.

Authentication and authorisation will also be key in such an application. Certain information will be for the eyes of the individual apartment owner only. Other data for the directors and managing agent.

Database schema

Diagram

Description automatically generated

API issues

The API will be self-generated. Additionally, it is hoped that information can be drawn across from a third-party financial system. The complications here start with whether there is an API available to permit this and, even if this first hurdle can be overcome, continue to confidentiality and trust issues. This is something to be addressed with the managing agent and the supplier of their finance system.

Security

This has been touched on. Authentication and authorisation will be important. Users will need to create an account. A username and password will be defined by the user when registering. Some form of additional verification will be necessary to ensure a newly registered user is linked with the correct finance account for the display of invoices and payments.

User flow

Managing issues

Diagram

Description automatically generated

Viewing company payments

Diagram

Description automatically generated

Viewing apartment owner’s financial situation

Diagram

Description automatically generated

Issues/risks

* The plan to present finance information as a React component page may not be feasible and the best alternative may be pdf files that can be downloaded.
* The company that controls the building is on the point of changing managing agents. This will mean moving to another agent that will have their own finance system. This is likely to take place at the end of June 2021. Thus any interface with the current system will be short-lived and cooperation to achieve it limited. Interfacing with the system of the new agent may not be possible immediately.