# INTERIM REPORT

Housing Management System

PROJECT 17/18

Musab Adam ec15210@qmul.ac.uk
SUPERVISOR: DR REBECCA LYNNE STEWART

## Contents

## Background Literature

My project will be aimed at housing accommodation companies that leases properties and then rents it out to tenants. These companies must pay monthly rents to the landlords that they have leased with the possibility of rents being deducted (rent being reduced for the month) to cover maintenance costs. There will also be times where properties must be returned to its landlord therefore companies need to move tenants to another property that they currently lease. Companies will also manage maintenance issues where an issue at a property is reported by a staff member or a tenant themselves and is then dealt with, using in house builders or external maintenance companies.

Currently I work with a housing accommodation company and whilst they do have something that does each of these tasks. That something are multiple spreadsheets where each employee is responsible for a specific spreadsheet. The issue with spreadsheets is that spreadsheets cannot read data with other spreadsheets seamlessly. The only way to get passed this is to use a formula that points a "cell" to another "cell" in another spreadsheet which is required to be running as well, which leads to another problem which is no centralised data. This means that data that is needed is fragmented into different spreadsheet for example, if the accounts team wanted to know the profit and loss margins of the month they would have to manually gather all the data from the expense spreadsheet and the rents spreadsheet thus making it an inconvenience for the accounts team. Another problem with spreadsheets is the reliability and robustness of the files. This means that spreadsheets tend to be corrupted over time and once they were corrupted, there was no recovery option for them.

I also did some research into programs that are already on the market and found a student accommodation program called "occam" that manages student's accommodation. Based on the brochure of the application (since there was no demo to use), it had the ability to track tenants (students) and their payments for instance, the program would track when the students stay at the accommodation would end and what their balance would be. However, there was no feature related to maintenance and expenses therefore if this was used in the company I work with, they would have to resort to spreadsheets to compensate for the missing feature and find a way of exporting valuable data from the student accommodation program thus making it more of an inconvenience for employers looking to gain something out of the data. (Occam, n.d.)

Another program I found online was called "localpad". Localpad is an online solution that deals with social housing accommodation rather than student accommodation like occam. From reading their solutions and benefits, there were features that manages tenants and tracks how many days tenants were living at the property. Another feature that I found very useful was that it allowed users to see what properties are vacant for incoming new tenants. However, a feature still missing was property maintenance therefore has the same issue as the occam software. Also since its a web application, there is a high chance that the software will be sluggish and compatibility issues with different browsers. (solution, n.d.)

My project would address these issues. It would have a storage that allows data to be centralised into one place and be accessible by different programs at the same time by using MySQL DBMS on a dedicated server. Also since MySQL will be in use, therefore the chance of data being corrupted is very low and if there was some sort of corruption in the DBMS, the server can restore its state to a time thus allowing data to be recovered. My program will also have the standard features that the two currently existing which are tenant management and property management but will not be an online application, rather it will be a java application thus keeping performance intact. Finally, with property maintenance issues, users using the program will be able to track issues using the java application and dispatch builders whilst tenants and staff members can use a web application to report issues thus allowing tenants currently living at the accommodation to report their issues without having access to the java application.

## Resources used

### AWS tutorials

In my program, it consists of having the ability to store and retrieve data from an external source which in this case was using a MySQL. However, this involves using a server that holds the database which can be connected to. For my server I started a one-year trial with amazon's web service that hosts a variety of applications. Using their RDS (relational database service), I've created an instance which will hold 20gb worth of data which will be more than enough the project and will create an empty MySQL type database. Whilst setting up the RDS instance, I had to resort to the AWS RDS's 'getting started with Amazon RDS' in order to make sure that I am using the most optimum setting for the tier I have used (free tier). (Amazon, n.d.)

### MySQL

As the database will be in MySQL, a driver is needed for java to communicate with the DBMS. This was obtainable through MySQL website in a form of a JAR file which is then imported into the java project. Once imported into a class, the driver needed to configure so that it can connect to the AWS server. For this, I had resorted back to amazon's AWS tutorials specifically how to connect JDBC to AWS instance which demonstrated on what the URL would be when connecting to my AWS instance and how to specify the username and password when connecting to the instance. (How to connect to mysql with jdbc driver java, 2009)

### Jfoenix Library

My program consisted of a custom JavaFX library called Jfoenix that takes a normal javafx library and applied a material look and feel to it. It is used using the scene builder software and has a variety of GUI elements that has an already defined CSS applied to it thus adding a hint of animation to the elements and a vibrant look to the elements. The library is open source and is available on GitHub (JFoenix, n.d.) thus not requiring any permission to use and modify again for other purposes. The website also consists of a full documentation on how to utilise the library correctly as the coding between the elements and the controller is done differently therefore it will require me to constantly be referring to the documentation for the elements to have an action. (Documentation, n.d.)

### Java Library

Another resource I will be constantly referring to is the default java library documentation. As majority of my program will be using method and classes from the original java library, there is a high chance that I will hit a barrier where a method may not work as I expect thus leaving to look back at the documentation which will consist of all the available methods, methods and arguments and what the method returns. (Oracle, n.d.)

### DB Schema

Since the project would be partnered with a database system. The database must be planned and designed carefully to not create abundant data and making it simper to handle data. As I have never handled real estate's data before, I did some research to see what schemas existed for real estates. Specifically, how real estates handled rent payments on databases and how it handled tenants moving between properties. I referred to a website consisting of many template models that was used in different types of business (Databaseanswers, n.d.)

## Requirements

### Functional requirements

- General
  - Log in using Username and Password
  - Correct log in will take user to home screen
  - Incorrect log in will deny access to home screen and will notify user of the combination being wrong
- Home screen menu
  - Be able to go to different part of the program example tenant management will open a new window that allows managing of tenants
  - Be able to come back to the home screen to navigate to another part of the program
- Manage service users/tenants (SUs)
  - Remove SUs from properties
  - Add SUs from properties
  - Move SUs from properties to another property
  - Record amount of nights/days stayed
  - Edit SUs (Name, day moved in)
- Manage properties
  - Add New property
  - Close off a property
  - Edit a property
  - Calculate rent that needs to be paid
  - Record the type of property (House/flat/studio/room)
  - Notify user as to what rent is due
  - Calculate the amount of rent that needs to be paid
  - Record the rent payed
- Manage expenses paid towards a property
  - Remove expenses
  - Record expenses paid towards a property and categorise it
  - Record utilities paid as well as keeping track of account numbers for the utilities
- Manage company's finance
  - Track profit/loss per property
  - Track amount of expense paid per property
  - Track amount of income gained from each property
  - Possibly import bank statements from online banking and automatically place each transaction into its correct category
- Manage house maintenance issues
  - Allow tenants to log into a website using email and password
  - Allow tenants to report a problem through an online system
  - Allow tenants to upload images of the issue
  - Track an issue from the application
  - View images uploaded by the tenant
  - Assign an engineer/builder that will attend to the reported issue

### Non-Functional Requirements

- Clear GUI
- Easy to use and easy to learn GUI

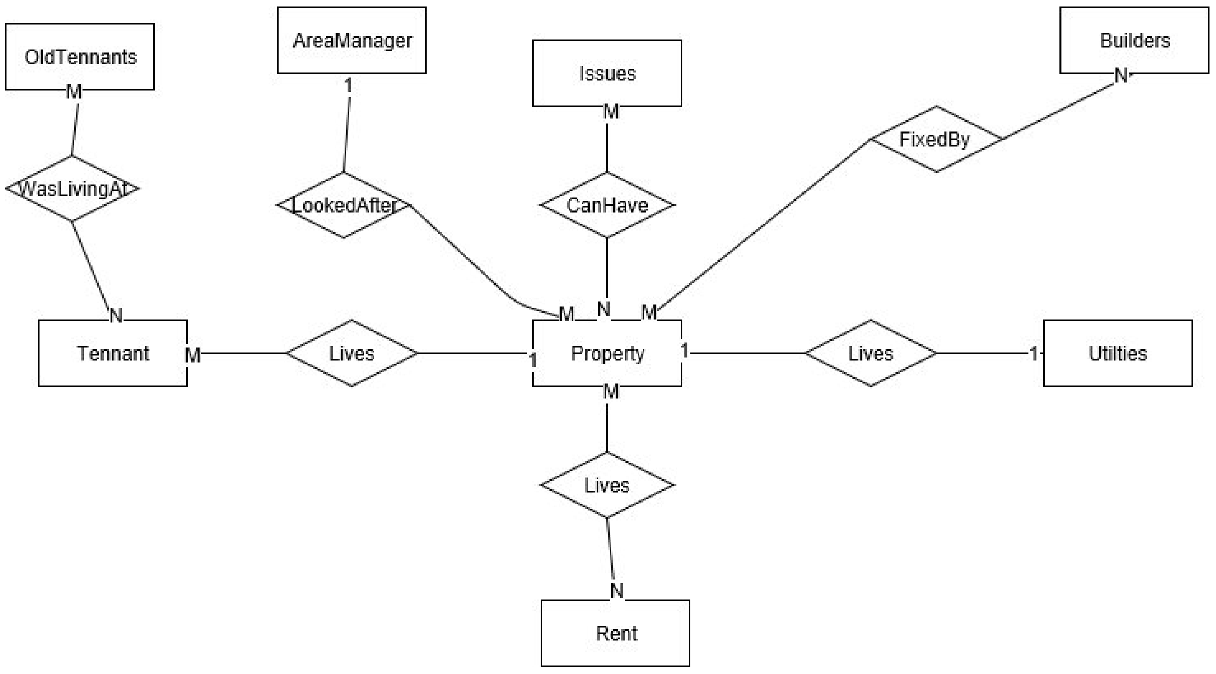
- Include validation
    - Check if tenants already exists when adding new tenant
    - Check if property already exists when adding new property
    - Check if username and password is not empty when logging in
- Must be responsive
- Must not crash or freeze due to a bug
- Have a stable connection to the DBMS

## Design so far

So far, my java project consists of a fully working login that takes the user's username and password and check the combination with the database on the cloud. The database is currently hosted by amazon's web application website that allows developers to host their web applications and DBMS. The database is secured with a master username and password which has been hard-coded into the program in a separate package. So, whenever a class requires a connection to the database, it will need to import the class from the package containing the method to connect to the database making it easier and to code and more efficient due to less coding.

### Entity Relationship Model

I have also designed a database schema which I have now used to generate the models in my database. I initially wrote out the schema on a word document and carefully added relations to each model with a brief description on how it is related. What's left is to normalise the schema, preferably up to 3NF to ensure that there is no abundant data in the database. I then visually designed the schema using a flow chart on the website called draw.io which is a web application that allows users to draw flowcharts and UML diagrams.



### Relational Schema

Tennant(<u>*tennantID*</u>, FirstName, LastName, DOB, *LivesAt,* LivingFrom )

LivesAt is a foreign key to PropertyID

Property(<u>*PropertyID*</u>, FlatNo, DoorNo, Firstline, PostCode, Town, Borough, *AreaManager, Ended*)

Rent(<u>*rentID*</u>, Cost, dayOfMonth, LandlordName, Account No, SortCode)

PaidRent(*PropertyID, rentID*, DatePaid, Deduction)

PropertyID is a foreign key to Property AND rentID is a foreign key to PaidRent

Utilities(*utilID*, *PropertyID*, CTaxBorough, CTaxAccntNo, CTaxCost, EnegComp,EnegAcntNo, EnegCost, GasComp, GasAcntNo, GasCost, WaterComp, Water AcntNo, WaterCost,DirectDebit)

PropertyID is a foreign Key to Property

Issues(*IssueID*, Description, Type, *reportedBy*, Date)

reportedBy is a foreign Key to Tennant

PropIssues(*PropertyID, IssueID*, *BuilderAssigned*,JobStarted, JobCompleted,Comments)

PropertyID is a foreign key to property AND IssueID is a foreign key to Issues AND BuilderAssigned is foreign key to Builder

Builder(*BuilderID*, FirstName, LastName, Specialty)

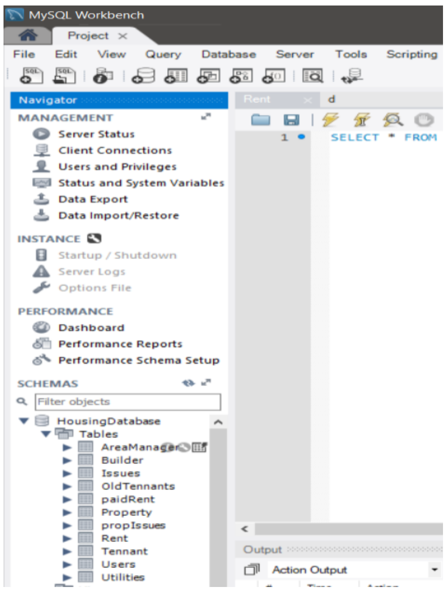
Area Manager(*staffID*, FirstName, LastName)

OldTennants(*tennantID*, FirstName, LastName, DOB, *LivesAt*, LivingFrom,LeftOn)

## Implementation

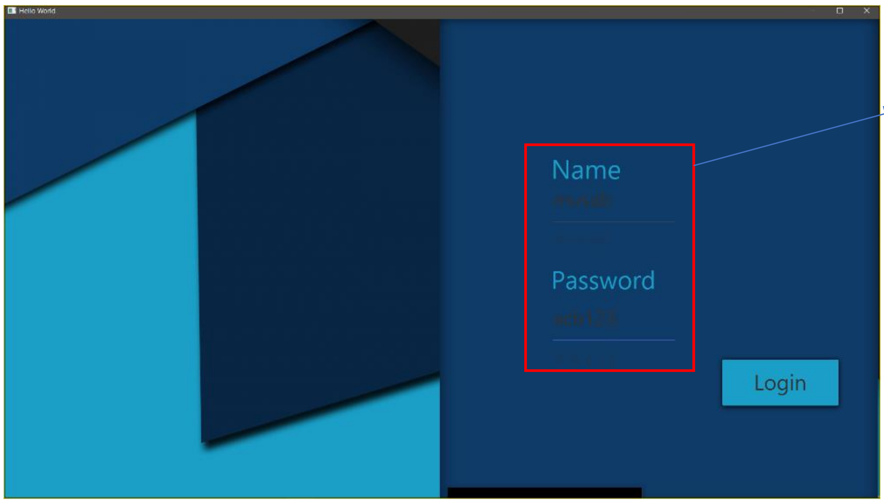
### Database

To create the table for the database, I used MySQL's workbench which is a graphical interface for managing the database. The interface contains predefined SQL statements that simplify managing the database. For instance, if I wanted to create a table, all it takes would be to right click the database and "create table"



## Styling

I have also decided to follow a material theme in my program which to aid me in this, I have researched a custom javafx library called Jfoenix which was designed with a material theme in mind. The library had to be downloaded their site and imported manually into the java project library. Since javafx uses scene-builder (a GUI editor) to allow developer to drag and drop elements onto a blank stage. Scene-builder also required the custom library to be imported into the GUI editor for the custom elements to be used in the editor. These custom elements have been styled to look like button used on an android application and have a predefined CSS with animations that makes it feel like an android application. To apply functions on the custom elements, I had to resort to Jfoenix documentation which teaches you how each element work with the controller and an example of how it used.



Here is an example of a field that is from the Jfoenix library

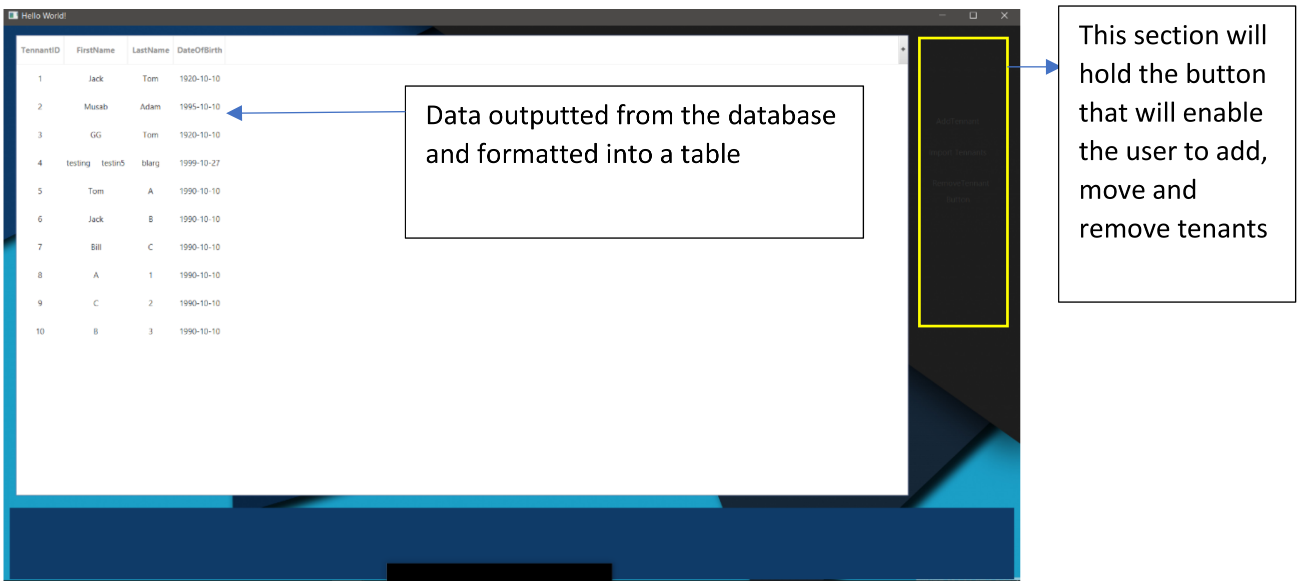
Since the custom library is coded differently, I will be constantly referring to the documentation so that the elements are working as intended.

## Login Screen

The login system has also been completed and can successfully communicate with the AWS server on the cloud with queries and results from the queries. The login class, once the login button is clicked, it takes the values entered and send a query to the DBMS on the AWS server. If a result set is returned, it means that a record was found thus the combination was correct and can be granted access to the home screen. However, if the login was incorrect, an error message would pop up asking the user to "retype their username and password correctly".

## Tennant Management

I have also started working on the tenant management part of the program and currently it now retrieves tenants from the database and outputs onto the custom table view from the Jfoenix library. However, it only gets the basics attributes in the "Tennant" table and for now does not handle any foreign key values which is something that will be simple to implement once I start the other major features. I have also implemented a feature that I have never done in the past which is to import data from a spreadsheet into java and then use data to insert into the database. Initially I wanted to use MySQL's built in feature that automatically takes data from a specified CSV file however as began testing it, it became clear that it wouldn't work as the Db engine could not find the specified file since the engine was on cloud and the file was stored locally. To get pass this, I manually went through each column using ".readnext" method which went through each column, taking out data from each cell in the column and then placing into an array. The array would be used in the SQL statement and then executed. This would be happening iteratively per column.


Hello World!
AddTennant
Import Tennants
RemoveTennant Button
Data outputted from the database and formatted into a table
This section will hold the button that will enable the user to add, move and remove tenants

| TennantID | FirstName | LastName | DateOfBirth |
|---|---|---|---|
| 1 | Jack | Tom | 1920-10-10 |
| 2 | Musab | Adam | 1995-10-10 |
| 3 | GG | Tom | 1920-10-10 |
| 4 | testing testin5 | blarg | 1999-10-27 |
| 5 | Tom | A | 1990-10-10 |
| 6 | Jack | B | 1990-10-10 |
| 7 | Bill | C | 1990-10-10 |
| 8 | A | 1 | 1990-10-10 |
| 9 | C | 2 | 1990-10-10 |
| 10 | B | 3 | 1990-10-10 |

My next step is to assign and un-assign tenants to the property whilst also getting the number of nights stayed at the property is tracked by the database correctly which would finalise the tenant part of the program. Then it will be moving onto the property part of the program.

## Risk Assessment

| Risk Description | How it will impact the project | Chances of occurring 1=low, 5=high | Preventative steps |
|---|---|---|---|
| Loss of data due to corruption or device being broken or lost | The impact would vary based on the progress of the project. If the risk occurred early in the project, then the impact would be low as I can quickly get back on track and redo everything that was loss due to minimal progress made. However, if the loss happens near the end of the project then the impact will be extremely high as it would be near impossible to redo everything that was lost in time remaining in completing the project. | 3 | Use GitHub and regularly make commits to the git to have "checkpoints" in the project so if a loss happened during the project, I can simply pull the latest checkpoint and work from the checkpoint. |
| AWS free tier subscription ended | The impact won't affect the project as much, but it will make it an inconvenience due to recreating the database and adding the data back into the database | 1 | Make monthly back ups of the data and if the subscription ends then I will migrate to Google's firebase servers to host my DBMS. |
| Being significantly off track | I will have less time to complete features and may cause me to rush implementing features thus increasing the risk of bugs occurring in my program | 4 | Implement features that are non-essential towards the end and focus on implementing features that are vital for the project to succeed |

| | | | |
|---|---|---|---|
| Feature being too difficult to implement | Will cause me to spend more time trying to implement the difficult feature thus increasing the risk of being off track | 4 | Research the feature and how to implement it using java documentation and examples on stack overflow website |
| Too busy working on other modules and leaving the project to the side | Will cause me to become more off track thus increasing the risk of the project not finishing in time | 3 | Plan out my week on what tasks I'm going to do for certain modules then allocate a time dedicated to progressing the project |
| Database data becoming corrupted and causing program to become unstable. | Will cause me to waste time remaking the database and adding the data back into the database thus making me off track with the project | 2 | Enable daily back ups on AWS server which allows the admin to rollback to a snapshot of the DBMS if a problem arises |

## TIMEPLAN

| *Features to complete* | *Start* | *End* |
|---|---|---|
| · Manage service users (SUs) | | |
| o Remove SUs from properties | 27/11/2017 | 20/12/2017 |
| o Add SUs from properties | done | Done |
| o Move SUs from properties to another property | 27/11/2017 | 20/12/2017 |
| o Record amount of nights/days stayed | 27/11/2017 | 20/12/2017 |
| · Manage properties | | |
| o Calculate rent that needs to be paid | 25/11/2017 | 01/01/2018 |
| o Record the type of property (House/flat/studio/room) | 25/11/2017 | 01/01/2018 |
| o Notify user as to what rent is due | 25/11/2017 | 01/01/2018 |
| o Calculate the amount of rent that needs to be paid | 25/11/2017 | 01/01/2018 |
| o Record the rent payed | 25/11/2017 | 01/01/2018 |
| · Manage expenses paid towards a property | | |
| o Record expenses paid towards a property and categorise it | 02/01/2018 | 09/01/2018 |

| | | |
|---|---|---|
| o Record utilities paid as well as keeping track of account numbers for the utilities | 02/01/2018 | 09/01/2018 |
| · Manage company’s finance | | |
| o Track profit/loss per property | 10/01/18 | 25/01/2018 |
| o Track amount of expense paid per property | 10/01/18 | 25/01/2018 |
| o Track amount of income gained from each property | 10/01/18 | 25/01/2018 |
| o Possibly import bank statements from online banking and automatically place each transaction into its correct category | 10/01/18 | 25/01/2018 |
| · Manage house maintenance issues | | |
| o Allow tenants to report a problem through an online system | 26/01/2018 | 15/02/2018 |
| o Allow tenants to upload images of the issue | 26/01/2018 | 15/02/2018 |
| o Track an issue from the application | 26/01/2018 | 15/02/2018 |
| o View images uploaded by the tenant | 26/01/2018 | 15/02/2018 |
| o Assign an engineer/builder that will attend to the reported issue | 26/01/2018 | 15/02/2018 |

## References

Amazon. (n.d.). *Tutorial: Create a Web Server and an Amazon RDS Database*. Retrieved from aws.amazon.com: http://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/TUT_WebAppWithRDS.html

Databaseanswers. (n.d.). *Kick-Start Conceptual Data Model for a Real Estate Business* . Retrieved from Databaseanswers: http://www.databaseanswers.org/data_models/real_estate_agent/index.htm

*Documentation*. (n.d.). Retrieved from jfoenix: http://www.jfoenix.com/documentation.html

*How to connect to mysql with jdbc driver java*. (2009, August 19). Retrieved from mkyong: https://www.mkyong.com/jdbc/how-to-connect-to-mysql-with-jdbc-driver-java/

*JFoenix*. (n.d.). Retrieved from Github: https://github.com/jfoenixadmin/JFoenix

Occam. (n.d.). *Features benefits*. Retrieved from Occam: http://www.occam.co.uk/features-benefits/

Oracle. (n.d.). *Java™ Platform, Standard Edition 8*. Retrieved from docs.oracle: https://docs.oracle.com/javase/8/docs/api/

solution. (n.d.). *solution*. Retrieved from localpad: http://www.localpad.co.uk/solution