GMIT
INSTITIÚID TEICNEOLAÍOCHTA NA GAILLIMHE-MAIGH EO
GALWAY-MAYO INSTITUTE OF TECHNOLOGY

UniLodge

An Accommodation Sharing Service for Students

Final Year Project
B.Sc.(Hons) in Software Development

by
Faris Nassif
Aaron Burns

May 10, 2020

Advised by Dr. John French & Dr. Martin Kenirons
Department of Computer Science and Applied Physics
Galway-Mayo Institute of Technology (GMIT)

Contents

List of Figures

About this Project

Abstract Housing, and the lack of affordable accommodation has become a hot topic in recent times, especially in relation to students having to endure undesirable living conditions for even more so undesirable rates. Daily, students are commuting great distances to avoid having to endure the financial burden of living at a local level. Students being unable to bear this burden leads to lower admission and attendance rates, an undesirable outcome for both the educational institutions and those looking to attend.

The proposed solution to help bridge this issue will be a web application, providing an easily accessible platform for both students searching for accommodation and for students living locally or living at a reasonable distance, who may not have the outlet to advertise their spare room or may be looking to room-share.

Authors This was developed as a 15 credit project by Faris Nassif and Aaron Burns, final year students of Galway-Mayo Institute of Technology.

Acknowledgements The authors would like to acknowledge the project supervisors Dr. John French and Dr. Martin Kenirons for the time and advice they dispensed during the course of the project.

Chapter 1

Introduction

This chapter will serve as a prelude to the project. The context, objectives and metrics for success and failure will be defined, followed by a summary of each chapter of the dissertation and its relevance.

1.1 Context

During the decision making process it was decided that the project must be relevant not only to the team but also to peers. The project must also hone existing skills and allow for the natural development of new techniques and processes while also being worthy in scope.

The team felt the issue of affordable student housing, or the lack thereof to be both a familiar and worthy problem to attempt to address. Being students, the team had access to first-hand continuous feedback and input from fellow students on the subject, allowing a more specialized and niche approach, an advantage others who attempt to approach the topic may be in absence of. Following the establishment of the problem area, the team held internal and supervisory discussions on how best to approach the task.

UniLodge was the result of much deliberation. UniLodge would serve students and home owners in Galway, allowing for both a practical and streamlined avenue of accommodation advertisement while existing as a simple to use platform for identifying listings that fit the standards and requirements of the student.

1.2 Objectives of the Project

As previously mentioned, the end goal of the project is to create an application that would help bridge the gap between home-owners and students by providing both parties with a platform that would allow for the organization of accommodative housing services specifically for students in the Galway area. To ensure this end goal was reached, objectives were outlined and followed by the team, they are as follows:

- Investigate the field by gathering the opinions and ideas of students in relation to the problem area.
- Evaluate and investigate the frameworks and tools available for creating a platform independent application.
- Create and develop an application based on student feedback that will allow users to arrange or offer lodging services for students.
- Incorporate state of the art technologies, frameworks and tools where applicable to ensure the developed application is both easy and simple to navigate while being of applicable standard.
- The application will, at a minimum, allow users to register an account, securely login, post listings and communicate with other users via a commenting system.

In regards to specific investigative objectives, a brief survey [33] was issued to fellow students early in the development cycle to help provide an idea of what the target audience would feel to be important in an application of this nature. The results helped shape the objectives and metrics for success or failure. The exact resulting actions taken in response to the outcome of the survey will be discussed throughout the dissertation.

1.3 Metrics for Success and Failure

The outlining of criteria for success or failure was imperative in ensuring the project remained on track regarding avenues that would enable it achieve its outlined goals. The defined metrics are of similar nature to that of the underlying objectives in *Section 1.2*, with the biggest difference being the metrics will be defined at a higher level.

The metrics for success and failure will be addressed and discussed in the conclusion following the completion of the applied project. These metrics were broken in two sections.

1.3.1 Dissertation

A selection of metrics were outlined specifically for the dissertation, they are as follows:

- *The dissertation will be easily understood, allowing readers without knowledge or familiarity of the subject area to form a suitable understanding of the concepts and ideas discussed.* In order to accurately measure this, periodic input was received from fellow students and friends, who would kindly read newly integrated sections and provide feedback.

- *The investigative aspect of the dissertation must thoroughly seek out and employ the feedback of end users, ensuring a ground-level foundation for the direction of the dissertation.* A basis for the dissertation will be established through means of investigative methods, ensuring a credible and concrete direction for the overall project.

1.3.2 Applied Project

A selection of metrics were outlined specifically for the applied project, they are as follows:

- *The developed student specific web-application will be easy to use, navigate and attempt to address the defined problem domain.* To ensure this was adhered to, at varying stages of development feedback from fellow students and friends was observed and documented relating to the application.

- *Collaboration between team members will be maintained to ensure a smooth and fair flow of work throughout the project.* Disagreements stemming from difference in vision and opinion are common within group-based projects, however it's important for the health of the project to ensure internal communication is maintained and practiced. Breakdown in communication could result in sudden change in scope or unmet development deadlines.

1.4 Dissertation Summary

This section will contain a brief overview of the dissertation structure including a description of the objectives of each chapter.

1.4.1 Methodology

In this chapter, the processes undertaken during the life cycle of the project regarding planning and development will be outlined. Investigative methods that were employed during the research phase, their results and the effect they had on the direction of the project will be brought to the attention of the reader. Additionally, the decisions, thought processes and influential factors leading up to those processes and design implementations will also be described.

1.4.2 Technology Review

A technological review will attempt to encapsulate the technical aspect of the project. This includes the different technologies incorporated, their implementation, why they were implemented and why they were chosen. The benefits of the chosen implementations will be critically analysed and compared with alternatives.

1.4.3 System Design

A detailed explanation of the overall architecture of the project will be provided. Code-snippets and diagrams will be included to help illustrate the inner workings of the application at a high level. Improvements to the system will be identified and potential competitive alternatives will be discussed.

1.4.4 System Evaluation

An evaluation of the software developed in the project will be carried out with the initial project objectives in mind. The final results of the project will be reviewed, including an analysis of areas for improvement and potential changes applicable to the overall system.

1.4.5 Conclusion

To conclude, a brief review will encapsulate the overall system. Key insights will be identified and reflected upon. A final analysis will describe the overall experience and what was learned from the development life-cycle of the project.

Chapter 2

Methodology

2.1 Project Management and Research

The first project meeting began in the final week of September, briefly after the project requirements had been defined and outlined. An early start was agreed to be something that would greatly benefit the overall development of the project and it was concluded that an idea should be finalized as soon as possible to allow for necessary pre-development research.

This section will further explore the aforementioned pre-development process, the influence of the supervisory meetings on this stage and the overall methodical conclusion and it's influence on the initial project direction.

2.1.1 Brainstorming and Initial Supervisory Meeting

In the weeks before development began, after the project idea had been finalized, technologies, concepts and potential inclusions were explored and discussed between the team members. A brainstorming phase was conducted on what to incorporate into the project.

Brainstorming

The team members met prior to the initial supervisory meeting to discuss potential avenues of exploration during the development phase. To produce effective ideas, questions had to be asked relating to the ultimate goals and objectives of the project, these included:

- ***What research areas should be prioritized before the development phase is initialized?***
- ***What type of Methodology would best fit our approach?***

- ***What benefits would different methodologies have when compared to others?***

Initial Supervisory Meeting

During the initial supervisory meeting following the project decision, potential research and developmental approaches were discussed and the team were advised to spend the next week heavily considering the nature of how the project will be implemented. All parties agreed external considerations and feedback were important for the nature of the project considering students will be the end users.

Following the supervisory meeting the team decided to research avenues for allowing other students a platform to contribute suggestions and ideas. This was felt to be an important inclusion since, while the team consists of students, it would be highly valuable and beneficial for the health and development of the project employ the ideas and feedback of multiple potential end users.

2.1.2 Research Methodology Consideration

The team felt there was a clear direction to take when it came to gathering opinions and information from fellow college students, meaning a comparative analysis of methodical approach wasn't necessary. Face to face discussions with students along with a brief survey on student accommodation and related accommodation applications would be necessary mediums for gathering information.

2.1.3 Development Methodology Consideration

There were numerous possible methodologies to consider, and following discussions internally between the team members and talks with supervisors, the list of potential methodologies were shortlisted to both **Waterfall** and **Agile**.

2.2 Gathering Information

Following the decision to employ both surveying and face to face methods for gathering information, a short online survey was constructed with the following questions:

1. ***How many students do you know that are currently renting accommodation local to the College?***
2. ***Out of those students, how many do you know of that are unhappy with their current living situation?***
3. ***What applications are you aware of that you (or those you know) would use to find accommodation local to your college?***

4. ***If an application was developed specifically to help students in Galway find local accommodation, would you or students you know be interested?***

5. ***What features would you like to see in the application?***

The team felt the choice to have as few questions as possible would be a valuable decision in terms of encouraging more students to partake. The goal was to avoid students feeling intimidated by an otherwise overwhelming barrage of questions.

Distributing the Survey

The survey was sent to various friends and students within the course. Recipients of the survey were encouraged to distribute the survey to friends of theirs who were also students. Responses were accepted for three weeks following the distribution.

2.2.1 Survey Results

Following the closure of responses the team were very happy with the higher than expected overall response to the survey [33]. Twenty-one students participated in the survey altogether, the great response rate was likely due to the simplicity and briefness of the survey.

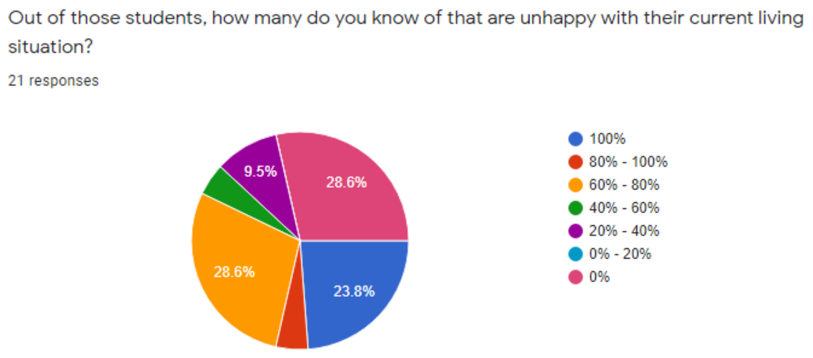
This subsection will act as a brief analysis of the responses given for the major questions in the survey.

Initially the recipients were asked how many students were they aware of that were renting locally to the college and based on that answer, were prompted to answer how many of those same students were unhappy with their current living situation.

Student Accommodative Unhappiness

This part of the survey was important in gauging the rough amount of students that are unhappy with their current living situation. Based on this result, it would be possible to see if the necessity of an accommodation application targeted at students would even be warranted.

Figure 2.1: Student Unhappiness



Out of those students, how many do you know of that are unhappy with their current living situation?

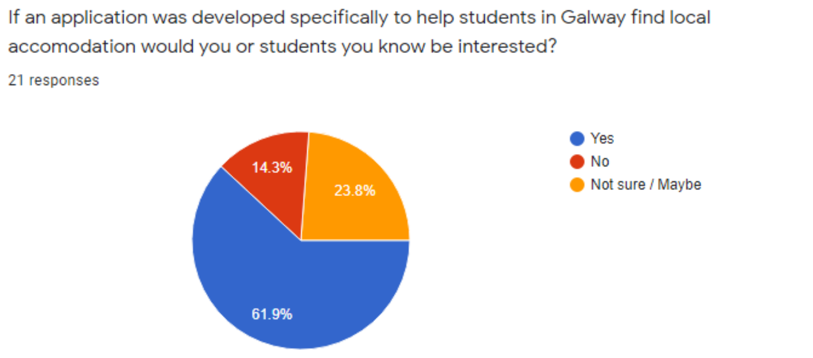
21 responses

Following the closure of the survey, it can be seen from *Figure.2.1* that just over a quarter of students are happy with their current living situation, a definitive result. Based on this response, it can be taken as a given that there is definitely a gap for a student-specific accommodation related product. Looking at the result, it can be wise to assume there isn't a similar application that is considered viable currently, and if there is it isn't effective.

Student Interest

Students were asked about their interest in a student specific accommodation application, this question was included to remove any ambiguity from the previous question if there happened to be any.

Figure 2.2: Student Interest



If an application was developed specifically to help students in Galway find local accomodation would you or students you know be interested?

21 responses

Following the overwhelming positive response illustrated in *Figure.2.1*, the results displayed in *Figure.2.3* only confirmed what was already known, that there is definitely a gap for a product of this nature.

Similar Applications

In order to provide the team with an idea of what similar in nature applications are currently available to students, survey recipients were asked to list some applications they feel themselves or others they know would use when searching for accommodation local to their educational institute in Galway.

Application Occurrence	
Application	Frequency
Adverts	10
Daft	6
Done Deal	5
AirBnB	5
Facebook	4
MyHome	2
Don't Know	3

Table 2.1: Frequency of Application Occurrence

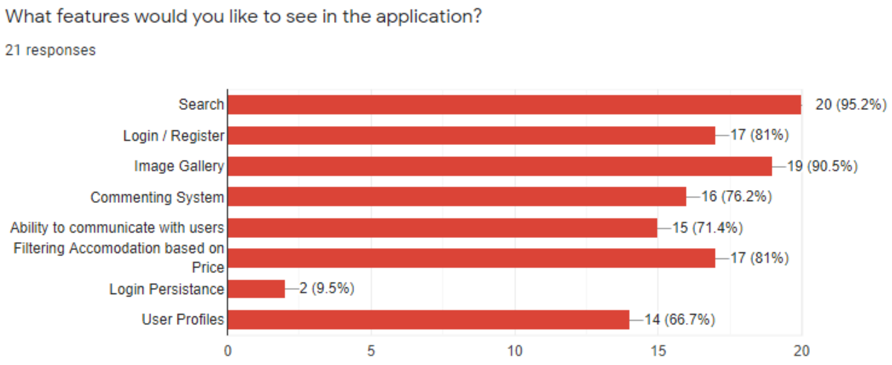
Following the analysis of the results illustrated in *Table.2.1*, it became clear there wasn't an absolute front runner that all recipients would be able to name on the spot. The results were scattered, with Adverts being the most known related application. The most known application in this field being one that doesn't actually specialize in finding accommodation speaks volumes about the lack of relevant student services of this nature.

This feedback would also prove useful in the sense that the listed applications could be analysed during the development process. Key strengths of the applications could be brought on board while weaknesses and unnecessary features excluded, allowing for a more specialized student experience.

Application Functionality

Finally, recipients were asked about the features they felt would be necessary in an application of this nature. This was asked mainly to gauge the main application features that end users felt would take priority over others.

Figure 2.3: Application Feature Priority



This response turned out to be highly valuable to the development phase of the project. The team could see first-hand the key features end users felt to be a necessity to an application of this nature. During the development phase this chart was highly referenced by the team when deciding the order of features to implement.

2.2.2 Face to Face Interviews

Following the overwhelmingly positive response of the survey, the team felt personal interviews with students would not be necessary and chose to occupy their time with working on key features mentioned by recipients of the survey.

That isn't to say students weren't consulted during the development of the project. Over the course of development students would periodically be asked their opinion on a new feature or to test a piece of functionality. This was felt to be an important process in ensuring the project remained healthy during the development phase.

2.2.3 Conclusion

The team felt very positive following the valuable feedback provided by fellow students. The need for an application of this nature was confirmed by the participants and applications that are considered similar in nature by potential end-users were provided, allowing the comparison and analysis of strengths and weaknesses of these applications, helping the team to build a platform combining the strengths and key features of similar natured applications.

Without the input, the development of the application may have gone off course, features may have been implemented that wouldn't have been considered a high priority or functionality may have been added that may make sense to the developers but end users may struggle to get a grasp of.

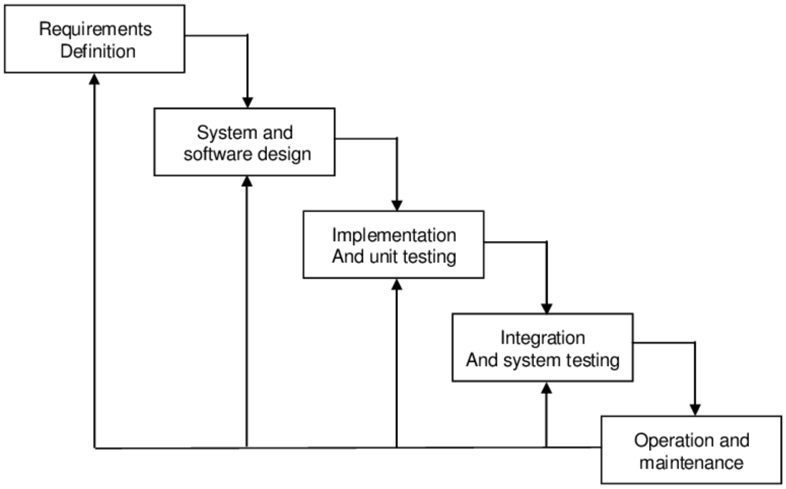
2.3 Determining the Development Methodology

Following the decision to shortlist both Waterfall and Agile, research began on which route would be best to take when considering the scope and overall goals of the project and comparisons between the two were drawn. A brief high-level overview of both methodologies will be included followed by comparisons and a practical analysis of each methodology in relation to the project.

2.3.1 Waterfall

Like most traditional software development models and methodologies, the Waterfall model is based on a series of phases or steps, illustrated in *Figure 2.1.* Waterfall allows progress to be easily measured, the complete scope of the project is known in advance which can be preferred based on the project being undertaken. Since the overall design is finished early in the development cycle, the Waterfall approach is especially effective in projects where various software components need to be designed in parallel [20]

Figure 2.4: Waterfall Model Cycle

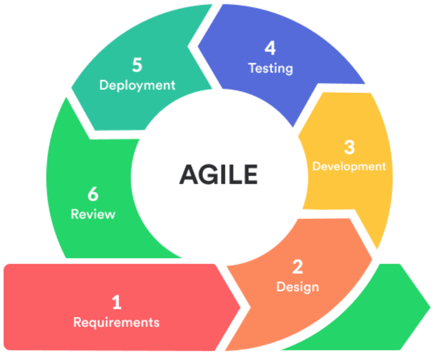

Requirements Definition
System and software design
Implementation And unit testing
Integration And system testing
Operation and maintenance

The Waterfall model is arguably the most well known development model, which is likely attributed to how long the model has been around, not to mention it's overall simplicity. Waterfall being an easily understood model naturally means it isn't difficult to manage which is mainly attributed to it's strict requirement definitions that ensures requirements are clearly defined, well received and understood [30]. Each phase is processed and undertaken uniformly, meaning phases are completed sequentially and don't overlap [38].

2.3.2 Agile

Initially described in the Manifesto for Agile Software Development, Agile is a an array of principles and methods for project management [29]. It is characterized by an iterate approach that allows software to be delivered to the end-user in periodical releases while also employing flexibility, allowing previously defined requirements to shift and project scope to change without significant damage or obstruction on the task currently being undertaken [39].

Figure 2.5: Agile Cycle



The popularity of Agile development methodologies among the software development industry has increased since their introduction in the mid-nineties [24], with almost 86% of surveyed international software developers using agile methodologies in their work [16].

2.3.3 Comparing Agile and Waterfall

Following supervisory and team meetings, research and analysis, both Agile and Waterfall were compared under the following headings:

- Applicability and Compatibility with the Project
- Requirement Delivery
- Flexibility

With the establishment of the aforementioned headings, comparisons were drawn between both approaches.

Agile	**Waterfall**
+ Strong ability to respond to the changing project requirements. + Issues and roadblocks can be detected and addressed rapidly. + Feedback is immediate and helps drive development. + Attractive methodologies that would fit the nature and goal of the project, namely **Kanban** and **Scrum**. + Encourages frequent end-user involvement, considering college students would be the target audience for the application this Agile benefit is especially attractive. – Lack of emphasis on necessary designing and documentation. – Project may grow ever-larger since there isn't a clearly defined end point.	+ Clearly defined and formalized requirements. + The software development structure is carefully planned and detailed, minimizing the number of potential issues and roadblocks. + Progress is easily measured, the beginning and end points for each phase are fixed allowing easier progress tracking. – Analysis, planning and requirements definition phase can end up taking time out of actually developing the project. – Low flexibility level meaning it may be difficult if not impossible to make major changes to the project while in the middle of development, considering the ever changing nature of the defined application this would be troublesome.

Table 2.2: Comparisons drawn between Agile & Waterfall

Following comparisons being drawn it became clear to the team an **Agile** approach would best suit the outlined project. This approach would enable many benefits but those that were the most attractive to bring to the project included:

1. **Interactive user involvement and feedback -** Having the target audience of the application be students while still being in college surrounded by potential users meant this was hugely beneficial.

2. **The ability to develop via small periodic releases -** Based on feedback from supervisors and other students the priorities may shift and sway, small incremental changes mean there would be increased flexibility and versatility when it came to adapting to change.

3. **Kanban and Scrum Methodologies -** The idea of incremental releases combined with periodic sprints based on workflow defined via the Kanban method was something that the team agreed would improve the overall flow of development following supervisory discussions.

2.4 Agile Approach

Given the ever changing requirements and nature of the project and for additional reasons outlined in the analysis section, the integration of Agile methodologies would be crucial to ensure an effective development cycle.

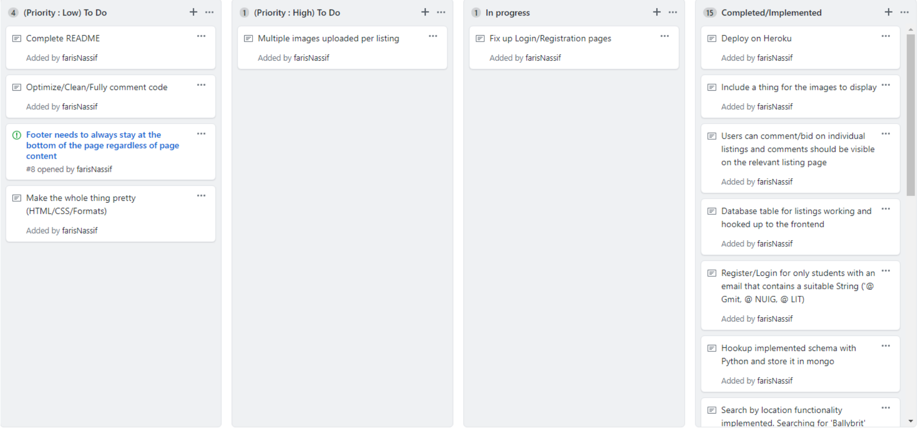
Deciding what Agile methodologies to incorporate into the project was the next step, Kanban and Scrum were two that had already been identified and deemed highly beneficial. Feature Driven Development and Extreme Programming were avenues that would merit future exploration.

2.4.1 Kanban

Since development began, the Kanban methodology was adhered to. The Kanban method is essentially a lean method to manage and improve flow systems. Like Scrum, Kanban is a process intended to help teams work together more effectively and efficiently [22].

Kanban allows a team to easily prioritize and visualize the main elements of the project in-progress and also easily delegate tasks based on what work needs to be started, what work is in progress and what has been completed.

Figure 2.6: Github Kanban



4 (Priority : Low) To Do

Complete README
Added by farisNassif

Optimize/Clean/Fully comment code
Added by farisNassif

Footer needs to always stay at the bottom of the page regardless of page content
#8 opened by farisNassif

Make the whole thing pretty (HTML/CSS/Formats)
Added by farisNassif

1 (Priority : High) To Do

Multiple images uploaded per listing
Added by farisNassif

1 In progress

Fix up Login/Registration pages
Added by farisNassif

15 Completed/Implemented

Deploy on Heroku
Added by farisNassif

Include a thing for the images to display
Added by farisNassif

Users can comment/bid on individual listings and comments should be visible on the relevant listing page
Added by farisNassif

Database table for listings working and hooked up to the frontend
Added by farisNassif

Register/Login for only students with an email that contains a suitable String ('@ Gmit, @ NUIG, @ LIT)
Added by farisNassif

Hookup implemented schema with Python and store it in mongo
Added by farisNassif

Search by location functionality implemented. Searching for 'Ballybrit'

Figure 2.6 above consists of the Kanban board used during the development of the project. The Kanban board consists of four main columns:

1. **To Do - Low Priority**
2. **To Do - High Priority**
3. **In Progress**

4. **Completed**

Initially the team had planned to use Trello [15] for project tracking, however following discussions with other students and finally supervisory discussions it was found that Github had a built in Kanban function.

Having the ability to track the progress of the project on the same platform as where the project is actually being collaborated on would be a huge benefit, it meant issues and commits could be assigned to tasks listed on the Kanban board meaning team members could work with more efficiency than if an external tracker like Trello was used.

2.4.2 Scrum

In addition to Kanban, the Scrum methodology was incorporated and employed. The Scrum methodology is very efficient when paired with the Kanban methodology. Sprints were employed and carried out by the team. A Sprint is a time-limited iteration of the continuous development cycle [14]. Sprints were generally one week in length. At the end of each week, a supervisory meeting would be conducted where the team would communicate their progress throughout the week and also any issues or blockers were brought to the attention of the supervisor.

At the end of the meeting, goals for the next Sprint were defined and potential solutions and avenues of exploration were discussed. Any new goal was included on the Kanban board and assigned to a team member. Based on previous information gathering, these tasks were undertaken in the determined priority order.

2.5 Version Control

Initially Github [5] was the only platform considered for version control coverage, however following discussions with other students, Gitlab [6] was spoken highly of, so this was another potential consideration.

Github had the home advantage of being a platform that had been used extensively by the team in previous projects, however Gitlab isn't too different and is likely only less popular since it's a newer platform. Both platforms are git based software-development platforms that provide access control and several collaboration features for developers.

The main advantage of GitLab for the team was its open source nature, which allows the hosting of your repositories on your own server free of charge. Gitlab's continuous integration pipelines were also very attractive, Github requires the employment of third-party tools tools like Travis-CI to avail of features like this.

The advantages of Gitlab were nice, however following internal discussions it was decided Github would remain the main source of Version Control for the project. The main reason for this being the advantages of Gitlab wouldn't be able to be as utilized as the team would like. Github being familiar and not too different from Gitlab meant there wasn't any deal-breaker that would merit the switch.

2.5.1 Github

A Github repository was setup remotely and used during development to allow for collaboration, code security and to track the progress of the project as well as providing the functionality of managing dependencies and providing alerts if new versions should arise.

The team took advantage of many of the additional features Github has to offer including it's Kanban board, the ability to branch and merge was utilized but could have been used more and the option to open and assign issues to name a few.

Chapter 3

Technology Review

Over the course of the project life cycle a plethora of frameworks, tools and development applications were available for integration or use with our application. This section aims to discuss the tools and technologies that were heavily considered and those that were ultimately used, why they were chosen and what alternatives were available.

3.1 Initial Considerations

During the discussion and planning phase goals were outlined and proposed however, how to reach the end point was still very ambiguous. For this reason a lot of time was spent considering different approaches and uncovering the benefits and drawbacks of venturing down a chosen route. This brief section will outline those initially considered approaches.

3.1.1 MEAN Stack

The MEAN Stack combines the best of JavaScript based technologies. The Stack is essentially a collection of open source components that provide a streamlined environment for building dynamic web applications.

The MEAN Stack consists of:

- MongoDB
- ExpressJS
- Angular
- Node.js

Perhaps the greatest attribute of the MEAN Stack for developers is that it's essentially a single language development stack, which can also be one of it's most undesirable attributes depending on the developer's JavaScript competency [19]. Other attributes that the development team considered attractive were the vast array of libraries and modules exposed via Node, it's speed, usability and flexible structure.

MongoDB

MongoDB is essentially a document-oriented NoSQL database. It's mainly used for high volume data storage. MongoDB uses collections and documents in place of tables like classical relational databases would. Documents consist of key-value pairs which are the basic unit of data in MongoDB.

The database is considered to be ideal for projects that need to be scalable or deal with vast amounts of data [35]. An attractive benefit MongoDB yields is its user friendly nature. If an entry is made into a collection that doesn't fit a defined schema, nothing halts and instead the entry is inserted into the collection without issue allowing the developer full control.

ExpressJS

ExpressJS, or simply Express, is a web application framework for NodeJS specialized for building web applications and APIs [3]. Express forms the backend cluster of the MEAN stack, handling all interactions between the frontend and the database, making sure data is seamlessly propagated to the user.

Express is completely reliant on API's exposed by NodeJS, acting as a thin layer over NodeJS, allowing for streamlined routing and server development with very little coding. Serving an application with Express can be accomplished with half a dozen lines of code, making it very lightweight.

```javascript
const express = require('express');
const path = require('path');

/* Static path definition */
app.use(express.static(__dirname + '/dist/client'));

app.get('/*', function(req,res) {

/* Serving static files */
res.sendFile(path.join(__dirname+'/dist/client/index.html'));
});

/* Exposing a port to listen for requests on (Served with NodeJS,
    Discussed below!) */
app.listen(process.env.PORT || 8081);
```

Listing 3.1: Serving with Express

Angular

Angular is a front-end framework built on top of JavaScript and developed by Google that specializes in enabling the development of single-page applications [1]. Single-page applications have become increasingly popular since their breakthrough, providing an array of benefits and advantages over the traditional multi-page web application, some of those advantages being:

- The possibility to run the developed application on almost all platforms and devices.
- Transference of the business logic from the server to the client.
- Attractive user interface that utilizes application-like UI components and aesthetically pleasing transitional animations.
- Can support the building of applications for offline usage or testing, easing the development experience and allowing the loading of the complete application at once.

While the benefits listed above [34] are some of the more attractive features of single-page applications there are of course drawbacks. The performance of single-page applications on mobile devices has been criticized and the 'all-in' approach may be an unattractive feature for larger applications, if one element of the overall page logic causes an issue it's likely the whole page will by extension will suffer.

NodeJS

NodeJS, or simply Node, considered the backbone of the MEAN stack, is an open source, cross-platform, JavaScript run-time environment that executes JavaScipt code outside of a web browser [23]. Using asynchronous events, Node allows the processing of multiple connections simultaneously, ideal for cloud-based applications that require scalability on demand.

Node comes with a complete integrated web server, allowing seamless application and database deployment to the cloud with the ability to support millions of simultaneous connections [36]. Node can be weak to larger processes, while a single thread may protect against process deadlocks, on the same token large scale freezes may occur with frequent high-resource requests.

Comparing MEAN and MERN

Another technology stack that piqued the attention of the developers was the MERN Stack, which is essentially the MEAN Stack excluding Angular and including React. Research was conducted on comparing the two [21] and the following was found:

Angular	**React**
+ Testing tools like Jasmine and Karma are well documented Angular frameworks that allow for seamless human-readable Unit Tests or browser/platform based test cases. + Application logic is a lot clearer and less convoluted than React due to it's declarative nature. + Enforces MVC-like design, giving developers an underlying structure to adhere to. React applications can be harder to maintain considering the overall design can be ambiguous and more unstructured. + Unidirectional data flow in applications allow data to flow to more seamlessly check for a change of state. − Weak ability to debug code. Debugging can be ambiguous without manual inclusion of libraries.	+ Mastering React is a lot less punishing than delving into Angular, Angular being a complete framework that incorporates associated knowledge of concepts like MVC or familiarity with Typescript. + Unidirectional data flow in applications allow data to flow to more seamlessly check for a change of state. + Very lightweight and less cumbersome than Angular for setup and collaboration. Dependency control is managed automatically. − Relies heavily on third-party libraries for actions and tasks that Angular could perform by on the fly due to it's built in service wrappers like for example Angular's built in wrappers for HTTP calls to the backend.

Table 3.1: Advantages and Disadvantages of React & Angular

Following the comparison of both React and Angular, the team spoke both internally and at supervisory meetings about the best route to take. Conclusively, the team felt Angular had the edge over React, the main factors being:

- Jasmine and Karma were two both very attractive testing tools that the team felt would enhance the development of the project.

- Libraries like Angular-material and Angular-animations to name few of many are very compatible with a project of this nature, not to mention very well documented.

- The team previously had exposure to Angular, which was a small bonus, however it was clear following the aforementioned research that previous trials with the framework hadn't even scraped the surface of the available features.

3.1.2 VueJS

VueJS is a JavaScript based framework used for building user interfaces and single page applications that can integrate seamlessly into a project at any stage. VueJS is marketed as an approachable, versatile and performant framework, boasting an incrementally adoptable system that's scaleable between a fully featured framework and a library [18]. However a major downfall of the framework is considered to be it's steep learning curve. VueJS was trialed for two weeks by the team, and following supervisory meetings and discussions between team members, the team ultimately decided due to the steep learning curve, the intimidating documentation for beginners and lack of tutorials that it wouldn't see a place in the development stack.

3.1.3 Redis

Redis, meaning REmote DIctionary Server is an in-memory distributed key-value data-store. Redis supports multiple types of data structures including, streams, bitmaps, sets and spatial indexes to name a few [28]. Key value databases excel at providing rapid access to information that has a corresponding function and following research and discussions with supervisors Redis was initially a very attractive inclusion into the project. The use of in-memory storage offers a number of advantages, namely data retrieval which as mentioned previously is extremely fast as well as memory writing being performed in mono-thread allows the write to be isolated, avoiding data loss [17].

At early stages in the project, various machine learning implementations were being considered. Redis would have been a perfect inclusion should the project have adopted artificial intelligence in any form, allowing for rapid access and storage of short-lived large scale machine learning data.

3.2 Chosen Technologies

Following research, input from supervisors and trials of the aforementioned technologies via a developed prototype, a stack was constructed that the developers felt would fit both the objective and scope of the project. This section will outline the technologies, tools, languages, frameworks and concepts that were ultimately implemented and descriptions of relevant implementations will be illustrated at a conceptual level.

3.2.1 Angular

After the elimination of the possibility of VueJS from the prototype and the ruling out of React, Angular was the strongest contender remaining. Following trials and integration with the prototype, the team found Angular to be a very nice fit. While a stack with Flask, Python and Angular isn't traditionally common, the Angular documentation allowed for a very smooth integration [2].

Within just over a week of integration into the prototype, the team were able to serve up static Angular files with Flask and establish connections and routes for accessing various defined endpoints.

3.2.2 The Flask Micro-framework

The Flask Micro-framework is designed for building simple and robust web-applications with a Python backend [4]. The Flask backend looks similar to a traditional Python file, functions are defined with an **@app.route()** decorator, declaring the function a handler for the endpoint which may be defined via parameters.
The performance of Flask is held in high regard when compared with frameworks of a similar nature [37]. The team found Flask to be lightweight and easy to use while still yielding attractive results. Additionally, another big advantage of using Flask is having the choice of an array of Python or Flask sub-modules that enhance the functionality and development experience of working with Flask.

```
@app.route('/delete/<string:_id>')
def delete(_id):
    try:
        collection.delete_one( {'_id': ObjectId(_id) } )
        return redirect('/')
    except:
        return 'Error - Cant delete the object!'
```

Listing 3.2: Sample Route Definition with Flask

3.2.3 MongoDB Atlas

Trialed during the development of the prototype, MongoDB Atlas proved to be worthy of inclusion into the project stack. MongoDB Atlas is a DBaaS (Database as a Service) platform designed by the same team that develops MongoDB.

Atlas provides all the features of its database counterpart without any complexities of security or maintenance [10]. Some of these features include:

- Automated Security Features.
- Built-In Replication.
- Backups and Point-In-Time Recovery.
- Monitoring and Traffic Tools.

Integration is also simplified, a URI is provided based on the development language being used that references the cluster, libraries like PyMongo allow for the dissection of the URI, providing an array of functionality relational to the URI, allowing for the interaction and control of individual collections within the cluster associated with the URI.

```
import pymongo
from pymongo import MongoClient

cluster = MongoClient(”MONGO_URI”)
database = cluster[”example_database”]
collection = database[”example_collection”]

collection.insert_one(postForCollection)
```

Listing 3.3: Defining and Accessing a Mongo Collection with Python

3.3 Deployment

An array of deployment platforms and tools for easing deployment were available to the team. Following experimentation with the prototype, a platform was decided and during development the tools to help ensure smooth deployment were identified. This section will provide a conceptual level overview of the aforementioned deployment tools and chosen platform.

3.3.1 Heroku

Heroku is an open source platform as a service cloud platform, providing extensive and well developed services for many aspects of the deployment life-cycle [7]. Heroku boasts deployment simplicity, allowing the deployment of almost every type of application. Should the type of application not be supported, there are various official and third-party build-packs to allow for seamless deployment. Additionally, Heroku also supports deployment via Git, a very attractive feature considering the Version Control setup of the project.

Initially AWS had been the likely deployment platform for the application. While hands-on experience with AWS would be beneficial, the team ultimately decided to deploy the application via Heroku. Supervisory discussions and research into Heroku's different build-pack and deployment options made Heroku the clear front runner considering the nature of the developed project. The documentation is also extensive and very well written, allowing the team to save valuable time with deployment while still getting hands-on experience with a widely used cloud application platform.

Generally speaking, it's highly advantageous to employ a cloud-hosted service as an array of benefits can be made available, these include:

- **Cost Savings** - Perhaps the biggest benefit of cloud-based services. Physical hardware investments aren't necessary, personnel isn't required to maintain hardware and the buying and management of equipment is handled via the service provider.
- **High Speeds** - Allowing deployment of a service in a matter of clicks, cloud computing enables the gathering of resources required for the system faster than conventional means.
- **Reliability** - Being one of the more attractive advantages of cloud services, reliability is measured by mainly by performance, connectivity, and security, all of which are more often than not guaranteed at a high level by cloud services .

The aforementioned advantages [25] are but a few of the more major advantages of cloud computing, though it would be disingenuous to ignore some of the disadvantages of cloud-based services which can include varying performance, potential technical issues or downtime.

3.3.2 Honcho

Honcho is a Python port of Foreman, a tool for managing procfile-based applications [8]. Honcho's purpose is to abstract away any complications of the procfile format and allow the application to be either deployed directly or export it to some other process management format. Another benefit of Honcho in this context and the main reason it was employed is down to it's utility of allowing multiple processes to run in unison.

```
python: gunicorn runner:app -b 0.0.0.0:8087
node: npm install && npm start -b 0.0.0.0:8081
```

Figure 3.1: Honcho Procfile

Honcho allows a procfile to be defined (*As outlined in* ***Fig.4.15***) in such a way that permits the specification of multiple processes to be ran on the cloud without employing additional Dyno workers, in this case, both the Python server for the API and the Express server to serve the static files can be ran in unison without any additional workers. The port must also be specified otherwise Heroku will assign a random port number to run on.

```
web: honcho -f ProcfileHoncho start
```

Figure 3.2: Heroku Procfile

The Heroku procfile (*As outlined in* ***Fig.3.2***) consists of a single line that starts the Honcho procfile, it acts as a type of necessary procfile proxy, since the contents of the Honcho procfile wouldn't run correctly in this procfile, it points Heroku to the Honcho procfile.

3.4 Languages

Working with a diverse stack naturally means an array of languages have to be employed, especially languages that wouldn't regularly see collaboration. This section will serve as an overview of the programming languages incorporated and provide a high-level review of each.

3.4.1 Python

Python is a high-level language, allowing developers to very efficiently express large ideas and prototype out complex functionality fast. Python is suitable for programmers of any skill-level, however has a relatively steep learning curve but is still very welcoming for beginners, described as an easy to learn but hard to master programming language.

Python has become one of the most widely used languages globally in the last decade, a result of multiple factors, not limited to but including it's extensive range of libraries, detailed and straightforward documentation [13] and the overall versatility of the language.

3.4.2 JavaScript

JavaScript is a high-level, just-in-time compiled multi-paradigm programming language. JavaScript uses curly-bracket syntax, first-class functions, prototype-based object-orientation and dynamic typing [9].

JavaScript is the backbone behind the creation and functionality of dynamic website content. If something moves, refreshes or otherwise changes on the screen, it's made possible with JavaScript, so naturally with the technology stack being used to develop the defined web-application, JavaScript will undoubtedly play a big role.

3.4.3 TypeScript

TypeScript is a super-set of JavaScript, as a result, JavaScript is still permitted within a TypeScript environment. TypeScript is considered a lot more developer-friendly than JavaScript, less declaration is needed and functional errors generally won't forbid compilation, making it a lot more manageable within larger systems.

One of the bigger changes TypeScript brings is the way in which it can declare variables. In JavaScript the variable is interpreted by how the developer manages and manipulates the variable, TypeScript allows for the specific declaration, for example, 'string', 'number' and 'boolean' may be unambiguously declared.

3.4.4 HTML

Hypertext Markup Language is the standard markup language for displaying documents in a web browser. Generally HTML is assisted by technologies like Cascading Style Sheets and scripting languages like the aforementioned JavaScript. Naturally, since the developed project is a web-application, HTML is prevalent throughout the frontend.

3.4.5 CSS

Cascading Style Sheets, or CSS, is a style-sheet language mainly used for the description of the presentation of a document written in a markup language, like HTML. CSS is considered a pillar technology of the web alongside HTML and JavaScript.

CSS is generally described as the 'skin' of an application or website, as it can control the scale, position and aesthetic of an overall system. CSS is considered an easy to use tool but is still very hard to master, so naturally those experienced and with an artistic eye can easy manipulate an initially dilapidated website into something a bit more aesthetically pleasing.

3.5 Development Environment Tools

This section will discuss the tools and features employed during the development cycle of the project that helped ease the overall development experience and help make it as smooth as it could be.

3.5.1 Visual Studio Code

The primary source-code editor used throughout the project was Microsoft's Visual Studio Code [32]. There were multiple reasons the team chose Visual Studio Code as the primary code editor during the project development cycle.

Debugging

Employing Python, JavaScript, TypeScript and other minor languages like HTML & CSS in the project meant having a code editor that could support the debugging of all relevant languages would save a great deal of time and effort during the development cycle, not to mention improve the overall quality of the project by extension.

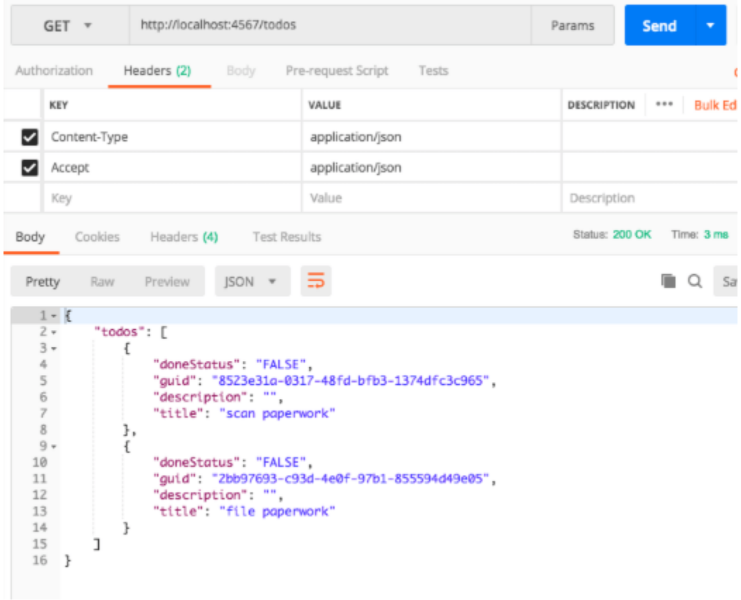
Extensions

A wide range of extensions are available for use with Visual Studio Code, these can include extensions to help with debugging, testing, or just language-specific extensions. During the development phase not only were all necessary language-specific extensions employed, but other quality-of-life extensions like the *Angular Essentials* and *Python Library Version Manager* extensions [31] were immensely beneficial.

3.5.2 Postman

Postman is a specialized tool intended to help test or dissect RESTful API's. Postman allows testing of API's via a very user-friendly user interface without the chore of writing blocks of code to test the functionality.

Figure 3.3: Postman API Requests



GET http://localhost:4567/todos

	KEY	VALUE	DESCRIPTION
☑	Content-Type	application/json	
☑	Accept	application/json	
	Key	Value	Description

Status: 200 OK Time: 3 ms

```
{
    "todos": [
        {
            "doneStatus": "FALSE",
            "guid": "8523e31a-0317-48fd-bfb3-1374dfc3c965",
            "description": "",
            "title": "scan paperwork"
        },
        {
            "doneStatus": "FALSE",
            "guid": "2bb97693-c93d-4e0f-97b1-855594d49e05",
            "description": "",
            "title": "file paperwork"
        }
    ]
}
```

Instead of writing up various tests in Flask to test each route and it's expected responses, a simple JSON body can be created with Postman and used send requests to any available endpoint. While Postman use did fade away as the project became slightly more convoluted, during the initial and prototyping stages it was very valuable.

3.5.3 Ngrok

Ngrok is a tool that can expose local servers behind network address translations and firewalls to the public internet over secure tunnels. Ngrok was very attractive, it would in theory allow the team to work off the same externally hosted server / API simultaneously, unfortunately it didn't see much use in this regard, however there were other uses the tool would be applied to.

PyNgrok, a Python module that wraps the functionality of Ngrok into an easy to use library was discovered and used during development to test deployment of the application. The API was the most troublesome cluster of the application to deploy to the cloud, Ngrok allowed the deployment of the application while having requests being directed externally to the Ngrok address, allowing the gradual deployment of the application which enabled thorough troubleshooting.

3.6 Representational State Transfer

Representational State Transfer, or simply REST, outlines a set of constraints to be used for creating Web services and is defined as a software architectural style. Services that conform to the REST architectural style are considered RESTful services. These RESTful services provide interoperability between systems over the internet [26].

In similar fashion to HTTP, RESTful services use request methods like GET, PUT, POST, DELETE, ensuring portability. REST is stateless, meaning every request happens in absolute isolation. When the client makes a request, it will include all necessary information required for the surver to fulfill the request. The server will never rely on information from previous requests [27].

3.7 Testing

After defining the technology stack for the application, complimentary testing tools were researched that would enable continuous and efficient testing of the main attributes of the project.

3.7.1 Karma

Karma is a command-line unit testing tool that can spawn a web-server and run source code against test code for each browser connected. The results of each unit test against each connected browser are examined and displayed via the command-line, highlighting which browser specific tests passed or failed.

Karma also monitors all the specified files defined within the configuration file, and checks to see if any file changes. If changes are detected, it can trigger the test to re-execute by sending a signal to the testing server to inform all of the highlighted browsers to re-run the test code. Each browser re-loads the source files, executes the defined tests and reports the results back to the server.

3.7.2 Jasmine

Jasmine is a behavior driven development framework for JavaScript. Jasmine provides functions that can help with test structuring and assertions.

```
describe(’sorting the list of users’, function() {
  it(’sorts in descending order by default’, function() {
    var users = [’faris’, ’bob’, ’jeff’];
    var sorted = sortUsers(users);
    expect(sorted).toEqual([’bob’, ’faris’, ’jeff’]);
  });
});
```

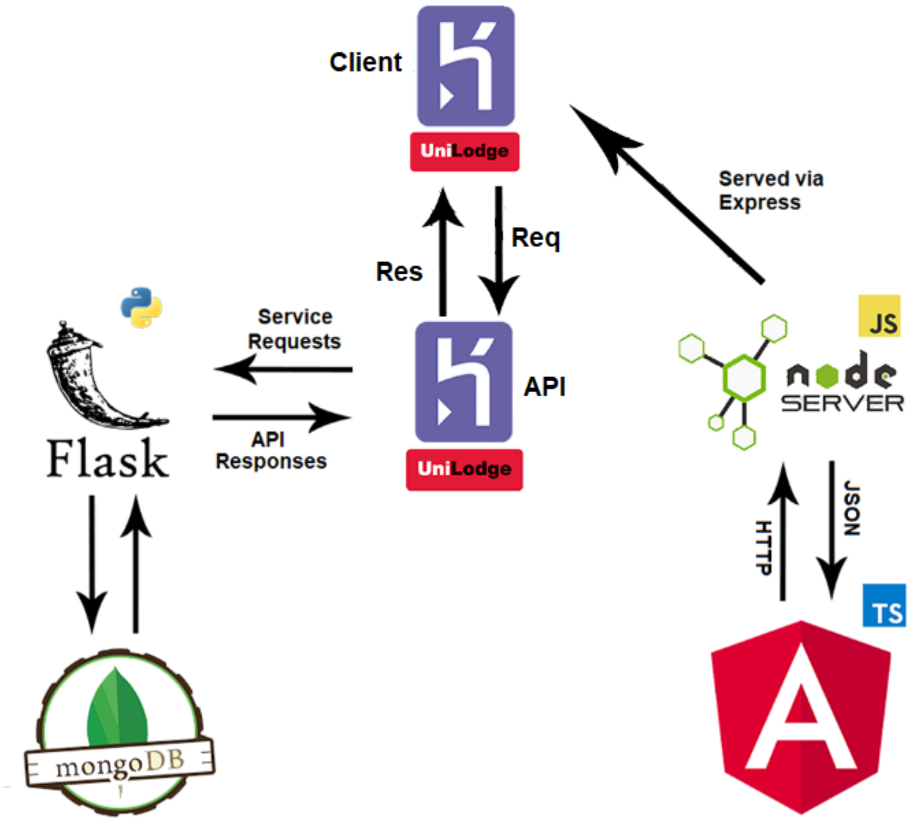
Listing 3.4: Test Definition with Jasmine

As the amount of tests grow, keeping them structured, in order and documented is vital, and Jasmine helps achieve this. The *describe* function is used to group tests together while multiple individual tests, defined with the *it* function can be contained within the overall describe function.

Chapter 4

System Design

Figure 4.1: Final System Architecture



4.1 Architectural Overview

The overall design of the system changed at multiple points throughout the development cycle before arriving at the conclusive design that can be seen in **Fig.4.1**. This section will give a brief high-level overview of how the final architectural design came to be.

4.1.1 Major Architectural Changes

Initially, the project during development consisted of Angular for the presentation layer, Flask for the server and MongoDB Atlas for the database. Routes would be defined with Angular and API calls would be directed towards Flask which would in turn interact with MongoDB Atlas and return a response to the frontend.

During deployment however a major shift in design occurred. Major issues were encountered whenever Heorku used Flask to serve static Angular build files, resulting in pages and components not working as they would locally. To address this, different methods were tested and verified.

Firstly, Node and Express were used to serve the Angular build files. A server proxy would redirect Express API calls to the Flask server address, allowing Flask to retain it's role as the sole API while using the Express server to serve the Angular build files. This action, while somewhat convoluted, did address the issue which had caused a blocker for multiple days.

Following testing of the aforementioned method, the team found the application to be uncomfortably slow and choppy. The decision was then made to deploy the Client and API on separate domains. This method, while not being completely efficient, does allow the overall system to become a lot more scaleable and following trials, worked flawlessly with very comfortable speeds.

4.1.2 Final Architectural Design

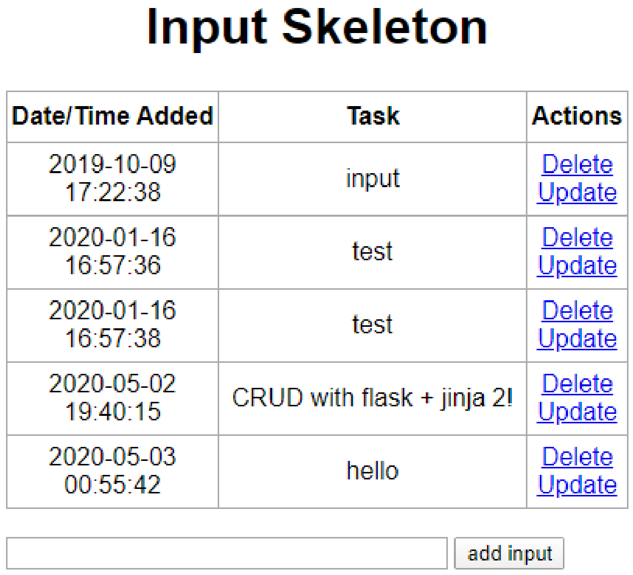
The conclusive design (*Which can be seen in **Fig.4.1***) allows the overall system to be delivered to users efficiently via cloud services, enabling a software as a service distribution model.

Two separate procfiles are used to enable the execution of both the Flask API server and the Express/Node server on different domains. This allows Heroku to use Express/Node to serve the Angular build files without issue while still allowing the application efficient access to API resources.

4.2 Prototyping

Prior to commencing development on the project, an initial prototype was built, incorporating technologies and frameworks that the team felt would compliment the scope and nature of the established project [11]. The goal of constructing the prototype was mainly to become familiar with integrating the technologies that were heavily considered and to see if they complimented each other in practice as well as they did in theory.

Figure 4.2: Basic CRUD Prototype



Input Skeleton

Date/Time Added	Task	Actions
2019-10-09 17:22:38	input	Delete Update
2020-01-16 16:57:36	test	Delete Update
2020-01-16 16:57:38	test	Delete Update
2020-05-02 19:40:15	CRUD with flask + jinja 2!	Delete Update
2020-05-03 00:55:42	hello	Delete Update

The resulting prototype as seen in **Fig.4.2** consisted of a very basic CRUD web-hosted application that would serve as a skeleton foundation for which to build the final project onto [12].

4.2.1 Prototype Technologies

As mentioned, the prototype was built with tools and technologies the team saw to be very likely inclusions in the final project.

The application was served with **Flask**, all application logic and endpoints were defined with **Python**. Initially, **VueJS** was incorporated into early stages of the prototype, however the decision to made to remove the framework following undesirable performance, documentation and integration. Instead of serving Vue static files, Flask would serve **HTML** and **Jinja2** templates, since no frontend was incorporated into the prototype, the design naturally was very basic, with very little CSS, which was mainly used to style and format the input table.

The Python module **PyMongo** was used to connect to **MongoDB Atlas**, which handled the storage of data. Finally the application was (*And is still*) hosted on **Heroku** [11].

4.2.2 Prototype Conclusion

Creating and deploying the prototype turned out to be beneficial for the team when initial development of the final project began. The use of some technologies was completely ruled out and a lot of time was saved and put to use on research or development.

The head start on the final project enabled by the decision to develop the prototype can't be understated, from creating local environments, generating requirements, creating a procfile for which to deploy the application and routing in flask to mention a few newly learned methods from a list of many, the team were glad for to have taken the prototyping route.

4.3 Initial Architecture

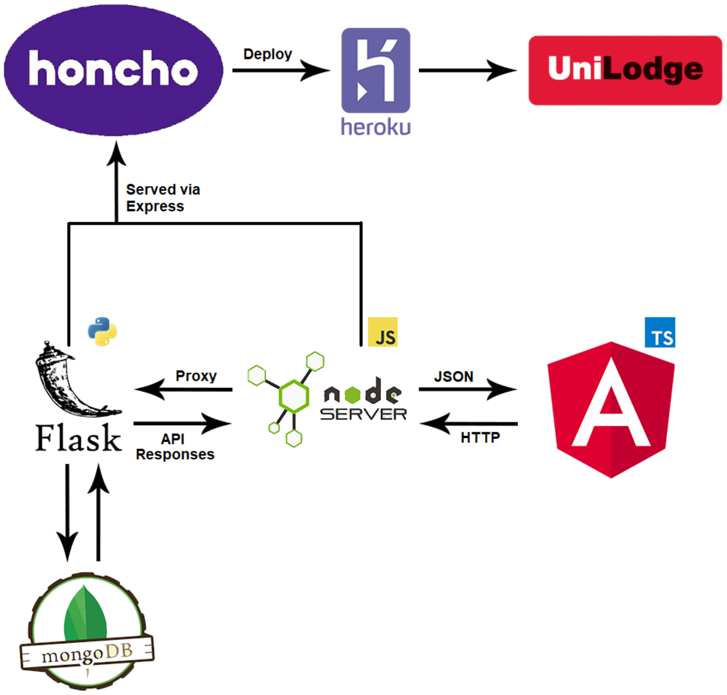
This section will briefly discuss the initial overall design before the decision was made to decouple the API and Client.

Following trial and error, a temporary design was constructed that allowed the deployment of the overall application, however it was far from ideal.

The design consisted of a Honcho procfile, which was used to enable the execution of both the Flask API and the Express/Node server while using just one cloud worker. Node/Express was used by Heroku to serve the Angular build files while API requests made to the express server would be redirected to flask with the use of a server proxy, allowing interactions between primary components of the system to remain unhindered.

This initial approach did pose issues, requests made via the proxy were a lot slower than they should ideally be, prompting the team to seek other methods of deployment.

Figure 4.3: Initial Design



4.3.1 Express Proxy

As mentioned previously in the chapter, the integration of Node/Express was required to serve the Angular static build files rather than Flask, so the use of a proxy was required to still access and interact with the Flask API without using it to serve the application.

```
{
    "/api": {
        "target": "http://0.0.0.0:5000",
        "secure": false,
        "changeOrigin": true
    }
}
```

Listing 4.1: API Proxy

The proxy file defined in **Listing 4.1** is simply a JSON file that is referenced in the **package.json**, where additional compilation settings are configured. The proxy itself is defined in a local file named **proxy-conf.json**, this file must be referenced as a script. When Heroku builds and starts the application, the proxy must be defined within the start script.

```
"scripts": {
  "ng": "ng",
  "start": "node server.js --proxy -p proxy.conf.json",
  "build": "ng build",
  "test": "ng test",
  "lint": "ng lint",
  "e2e": "ng e2e",
  "heroku-postbuild": "ng build --prod"
},
```

Listing 4.2: Proxy Reference

Once established, the start script will execute **node server.js**, starting the Express server and configuring the proxy path. Since the purpose of the server is just to serve static files, the server script doesn't need to be lengthy since any heavy work will be delegated to Flask.

```
const express = require('express');
const path = require('path');
const cors = require('cors')
const apiProxy = proxy('/api', { target: 'proxy.conf.json' });

const app = express();

app.use(express.static(__dirname + '/dist/client'), apiProxy);
app.use(cors())

app.get('/*', function(req,res) {

res.sendFile(path.join(__dirname+'/dist/client/index.html'));
});

app.listen(process.env.PORT || 8081);
```

Listing 4.3: Express Server Runner

Further insight on the decisions made during the development cycle and how the overall system is structured, including how individual components interact with each other will be discussed throughout the chapter.

The remainder of this chapter will cover and discuss the overall Final Design.

4.4 Data Tier

MongoDB Atlas was employed to manage all the data-related needs of the project. Database operations are all executed by Python, based on requests made to the API by the frontend. In this section, the interactions between the aforementioned data related components will be conveyed and their inner-workings explained.

4.4.1 Database Models

Models had to be defined to represent intractable entities within the system. These models can be interacted with allowing an array of CRUD functionality. Key values were used to associate associative models, enabling the simplification of the overall cluster and allow for the employment of functions like cascading operations.

```
export class Listing {
    public Unique_Id: any,
    public Title: string,
    public Seller: string,
    public Description: any,
    public Location: any,
    public Price: number,
    public ContactNumber: string,
    public Image?: string,
}
export class User {
  _id: number;
  Username: string;
  Password: string;
  Image?: any;
}
export class Comment {
    Listing_ID: string;
    Comment_ID: any;
    Poster: string;
    Content: string;
    Timestamp?: any;
 }
```

Listing 4.4: Definiton of Database Models

The models were defined at a class level, permitting more control and allowing the ability to define an attribute as optional or required. With TypeScript, this was accomplished by appending a '**?**' to the name of the attribute, defining it as an optional value.

Unique values were used to associate key values to associative models, all content created by a specific user will always be associated with that user's unique ID, making operations like finding all comments or posts by that user a simplified task.

4.4.2 MongoDB Atlas Interactions

The Python library **PyMongo** was utilized to expose and allow for the use of the full suite of Mongo utility methods and crucial CRUD commands.

To adhere to the single responsibility principle, a specific Python class was constructed that would encapsulate the accession of specific data collections. For each collection, a method would be defined, permitting invocation of the method which will return the collection object which can be used for manipulation.

```python
from flask_pymongo import MongoClient

# URI for database connection was stored in a local file for security
with open("mongo_uri.txt", "r") as uri:
    MONGO_URI = uri.read()

cluster = MongoClient(MONGO_URI)
database = cluster["UniLodge"]

def Users():
    users = database["Users"]
    return users
```

Listing 4.5: Database Accession Class

Based on the defined class in **Listing.4.5**, the collections could be called externally, enabling encapsulation of the collection logic while still exposing necessary PyMongo functionality. In classes that would externally interact with the collections, the manipulation of collections would still be simplistic, they would need only to import and define the accession class.

```python
# Definition of Database class as d_a
import data.database_accessor as d_a

@temp_users_blueprint.route('/api/users', methods=['GET'])
def list_users():
    # The object d_a allows full control of the collection
    userList = list(d_a.Users().find({}, {'_id': False}))

    return jsonify(userList)
```

Listing 4.6: Interacting with Collections

As can be seen in **Listing.4.6**, interactions with the collection objects still remain practical and simplistic. In this specific scenario, a list of users are returned to the client as a JSON object, enabling further interaction with the collection by the client.

4.4.3 Hashing

To ensure sensitive user information would never be compromised, the Python library Bcrypt was employed in the backend, allowing the encryption of confidential data.

```python
import bcrypt as bc

def generate_hash(plain_text_password):
    pw_hash = bc.bcrypt.hashpw(plain_text_password.encode("utf-8"), bc.gensalt())
    return pw_hash
```

Listing 4.7: Hashing a Password with Bcrypt

The function defined in **Listing.4.7** when provided with a plain text password, generates a salted hash and returns the hashed password to the caller. The function is called whenever a newly registered user submits a registration form with their plain-text password being the sole parameter for the function. Once called, the plain-text password is hashed and salted, before being stored in the database as the designated hash.

```python
import bcrypt as bc

def check_password(plain_text_password, hashed_password):
    if bc.bcrypt.checkpw(plain_text_password, hashed_password):
        result = True
    else:
        result = False # Password is wrong :(
    return result
```

Listing 4.8: Verifing a Password

The hashing algorithm used is one-way, meaning it can never be decrypted back to it's original plain text password. To verify the user in the future if for example they want to login, when their plain text password is entered it will again be hashed, this new hash will be compared against the hash stored in the database from when they previously registered. Should both hashes match, it would prove the passwords are the same considering the same input will always yield the same hash.

4.5 Logic Tier

The middle tier for the system acts as a doorway between the data tier and presentation tier. This tier consists of a mix of Python and TypeScript. API requests defined in TypeScript service files are propagated to the Flask API, depending on the undertaken user action. The API performs performs an action, generally in relation to the database and propagates a response back to the user through the TypeScript service file. This section will break down and further explain the interlinked processes that make up this tier.

4.5.1 Flask API

The API is completely written with Flask and documented with Swagger. TypeScript handles the sending of requests to the API. Simplistic methods contain the API endpoint to invoke as well as the expected object that should be return.

```
getListings(): Observable<Listing[]> {
  return this.http.get<Listing[]>(this.userUrl + '/api/listings');
}
```

Listing 4.9: Basic Request to the API

The TypeScript method in **Listing 4.9** will seek the defined API endpoint and will expect an array of Listings. Assuming the Flask server is running, the endpoint will be accessed and will return a Listings array for use in the frontend.

```
@listings.route('/api/listings', methods=['GET'])
def list_listings():
    # Find all listings within the Database
    listings = list(d_a.Listings().find({}, {'_id': False}))

    # Send the list of listings to the frontend
    return jsonify(listings)
```

Listing 4.10: API Counterpart Route

Blueprint

Using the Python library **Blueprint**, routes could be categorized and coupled together in separate classes depending on their function. This allowed a single class to handle all requests and delegate a task to it's designated route and class.

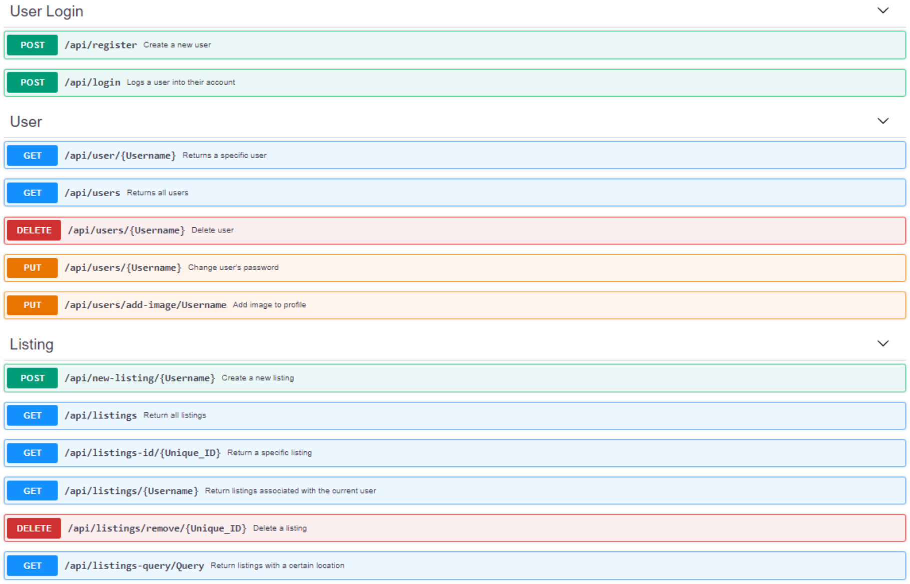
```
from routes.login_route import login_blueprint
from routes.register_route import register_blueprint
from routes.temp_users_route import temp_users_blueprint
from routes.listings_route import listings_blueprint

app.register_blueprint(login_blueprint)
app.register_blueprint(register_blueprint)
app.register_blueprint(users_blueprint)
app.register_blueprint(listings_blueprint)

if __name__ == "__main__":
    app.run()
```

Listing 4.11: Main Flask Runner

API Documentation

The API was documented using Swagger. The API is RESTful, allowing full user application access via the Client.

Figure 4.4: Swagger API



User Login

POST /api/register Create a new user

POST /api/login Logs a user into their account

User

GET /api/user/{Username} Returns a specific user

GET /api/users Returns all users

DELETE /api/users/{Username} Delete user

PUT /api/users/{Username} Change user's password

PUT /api/users/add-image/Username Add image to profile

Listing

POST /api/new-listing/{Username} Create a new listing

GET /api/listings Return all listings

GET /api/listings-id/{Unique_ID} Return a specific listing

GET /api/listings/{Username} Return listings associated with the current user

DELETE /api/listings/remove/{Unique_ID} Delete a listing

GET /api/listings-query/Query Return listings with a certain location

The view the full API documentation via Swagger, see *Appendices*.

4.5.2 Authentication

To further secure and authenticate the overall system, JSON Web Tokens were employed with the help of the Python library Flask_JWT, which would only allow the accession of sensitive routes if the user had a specific access token saved in their local browser storage. The token will be stored locally whenever a login is successful, allowing the navigation of sensitive routes.

```
from flask_jwt_extended import JWTManager

# Load confidential Secret Key from local file
app.config.from_envvar('SECRET_KEY')

jwt = JWTManager(app)

@login.route('/api/login', methods=['POST'])
def login():
    if (p_h.check_password(password, stored_hash)):
        # Map JWT token to the Username
        # Set it to expire in 25 minutes
        result = create_access_token(identity=str(username), expires_delta=(datetime.timedelta(minutes=25)))
    else:
        result = Invalid
    return jsonify(result)
```

Listing 4.12: Issuing a limited JWT Token

Once an access token is issued, it's valid for twenty-five minutes. Until the time expires, the token will be present in local storage, permitting authenticated behaviour, once expired the user must again log in.

```
if (login_result) {
    localStorage.setItem("access_token", login_result);

    this.router.navigate(['/profile/' + username]);
}
```

Listing 4.13: Storing the Authentication Token

Following the issuing and validation of the token, the now authenticated user may access routes that previously locked. This is accomplished with a simple method decorator (**@token_required**) as well as some TypeScript logic.

```
# Create a new Listing
@listings_blueprint.route('/api/new-listing/<string:Username>', methods=['POST'])
@token_required
def new_listing(Username):
    Username = get_jwt_identity()
    listing_data = json.loads(request.get_data().decode())

    try:
        d_a.Listings().insert_one(listing_data)
        result = Success
    except:
        result = Invalid

    return jsonify(result)
```

Listing 4.14: Method with Token Annotation

4.5.3 Angular Services

Angular Services act as the doorway to the API, any query for specific data has a corresponding service method. For example, take the following component method that requests all listings currently in the database so they can be displayed:

```
getListings(): void {
  this.listingService.getListings().subscribe(listings => this.
      listings = listings);
}
```

Listing 4.15: Method for Acquiring all Listings

If correctly executed, the method outlined in **Listing 4.15** will acquire all listings from the backend and store them in the array *listings*. In order for this to happen, it needs to access the API through the service method previously mentioned.

```
private userUrl = 'http://localhost:5000';

constructor(private http: HttpClient) { }

getListings(): Observable<Listing[]> {
  return this.http.get<Listing[]>(this.userUrl + '/api/listings');
}
```

Listing 4.16: Service Method for Retrieving all Listings

The Service method which can be seen in **Listing 4.16** implements an observable interface, which is common for executing asynchronous operations. A get request is fired at the defined API endpoint, returning all listings to the Service method.

Since the component method outlined in **Listing 4.15** is *subscribed* to this Service method, it waits for the call to the API to finish. When complete, the array of listings is returned to the component, it can then be used to display all listings currently in the database.

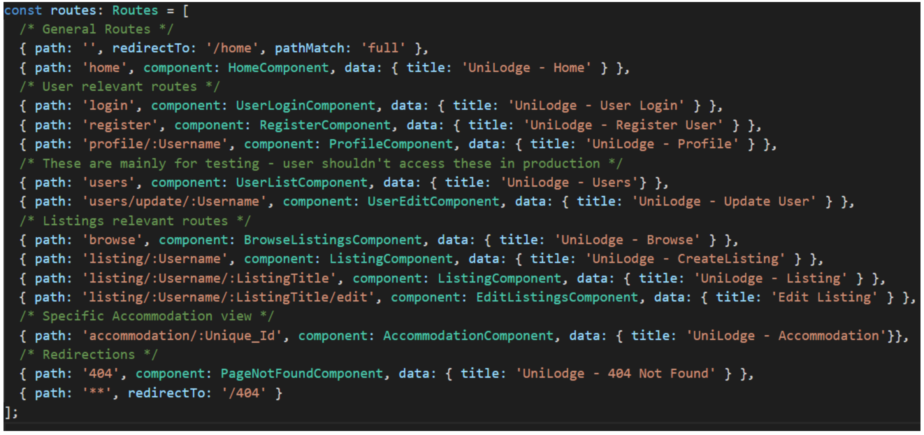
4.6 Application Tier

Angular was used to develop the client side of the application. Using Angular allowed the client to be presented as a single-page application which allows the injection of multiple components as apposed to traditional multi-page applications. This section will cover all information related to the presentation of the application and how it's wired together.

4.6.1 Angular Routing

Angular Routing was employed to define and initialize all in-app navigational routes. Route definition was associated with a specific component, ensuring the navigation to that route displayed the relevant and corresponding component.

Figure 4.5: Angular Routing



```
const routes: Routes = [
  /* General Routes */
  { path: '', redirectTo: '/home', pathMatch: 'full' },
  { path: 'home', component: HomeComponent, data: { title: 'UniLodge - Home' } },
  /* User relevant routes */
  { path: 'login', component: UserLoginComponent, data: { title: 'UniLodge - User Login' } },
  { path: 'register', component: RegisterComponent, data: { title: 'UniLodge - Register User' } },
  { path: 'profile/:Username', component: ProfileComponent, data: { title: 'UniLodge - Profile' } },
  /* These are mainly for testing - user shouldn't access these in production */
  { path: 'users', component: UserListComponent, data: { title: 'UniLodge - Users'} },
  { path: 'users/update/:Username', component: UserEditComponent, data: { title: 'UniLodge - Update User' } },
  /* Listings relevant routes */
  { path: 'browse', component: BrowseListingsComponent, data: { title: 'UniLodge - Browse' } },
  { path: 'listing/:Username', component: ListingComponent, data: { title: 'UniLodge - CreateListing' } },
  { path: 'listing/:Username/:ListingTitle', component: ListingComponent, data: { title: 'UniLodge - Listing' } },
  { path: 'listing/:Username/:ListingTitle/edit', component: EditListingsComponent, data: { title: 'Edit Listing' } },
  /* Specific Accommodation view */
  { path: 'accommodation/:Unique_Id', component: AccommodationComponent, data: { title: 'UniLodge - Accommodation'}},
  /* Redirections */
  { path: '404', component: PageNotFoundComponent, data: { title: 'UniLodge - 404 Not Found' } },
  { path: '**', redirectTo: '/404' }
];
```

For individual routes that display specific listing or user data, a service call to the API is sent and specific data related to that entity is returned, populating the page with the relevant information.

4.6.2 Page Views

This section will contain the various page views for UniLodge and the functionality tied to each individual page.

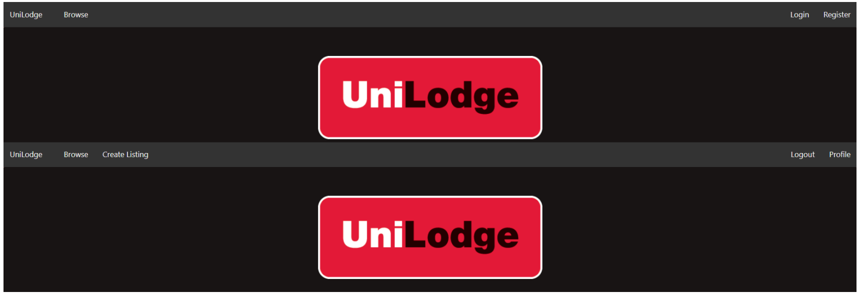
Navigation

At the top of each page, a navigation bar is present. This bar is dynamic and will change depending on the logged in state of the user. Initially, the Navigation bar provides traversal to the following pages:

- Home Page

- Browse Page
- Login Page
- Register Page
- Profile Page **Available after login*
- Create Listing Page **Available after login*

Figure 4.6: Nav Bar Before & After Login

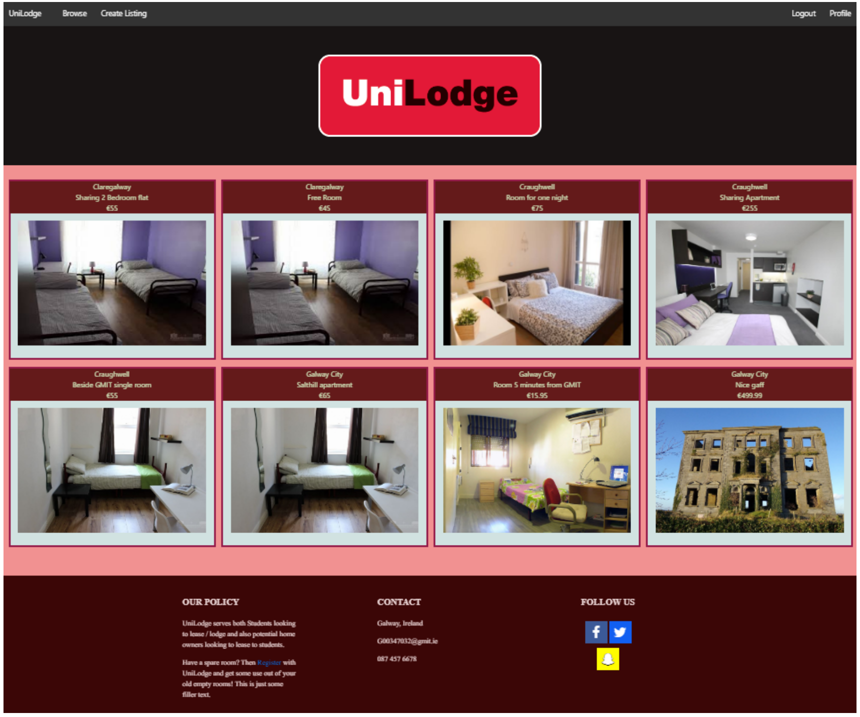


Once logged in, the **Register Page** and **Login Page** buttons are replaced with **Profile** and **Create Listing** navigation buttons.

Home Page

The first page the user will see when visiting UniLodge will be the home page. The home page displays up to 8 random listings to the user. The page itself provides no exclusive functionality and acts solely as an introduction to the application.

Figure 4.7: UniLodge Home Page



Claregalway
Sharing 2 Bedroom flat
€55

Claregalway
Free Room
€45

Craughwell
Room for one night
€75

Craughwell
Sharing Apartment
€255

Craughwell
Beside GMIT single room
€55

Galway City
Salthill apartment
€65

Galway City
Room 5 minutes from GMIT
€15.95

Galway City
Nice gaff
€499.99

OUR POLICY

UniLodge serves both Students looking to lease / lodge and also potential home owners looking to lease to students.

Have a spare room? Then Register with UniLodge and get some use out of your old empty rooms! This is just some filler text.

CONTACT

Galway, Ireland

G00347032@gmit.ie

087 457 6678

FOLLOW US

As mentioned in the previous section, an Angular Service method is invoked to retrieve the listings from the API, which is then stored at a component level once retrieved as an array of JSON objects. NgForOf, an Angular structural directive allows for the iteration over this array in the HTML, and since Angular uses two-way binding, the JSON array can be accessed from the HTML.

```
<div class="container-fluid bg-3 text-center" >
  <div class="row">
    <div *ngFor="let listing of listings; let i=index" class="col-sm-3">
      <ion-slide *ngIf="i<8" >
      <div class="listing-box">
        <p>{{ listing.Location  }} <br> {{ listing.Title }} <br> {{ listing.Price }}</p>
        <img src="{{ listing.Image }}" class="img-responsive" alt="Image" (click)="listingRedirect(listing.Unique_Id, listing.Seller)">
      </div>
    </ion-slide>
    </div>
  </div>
</div>
```

Listing 4.17: NgFor to Display Listings

Each image is also clickable, navigating the user to the specific listing page for that listing.

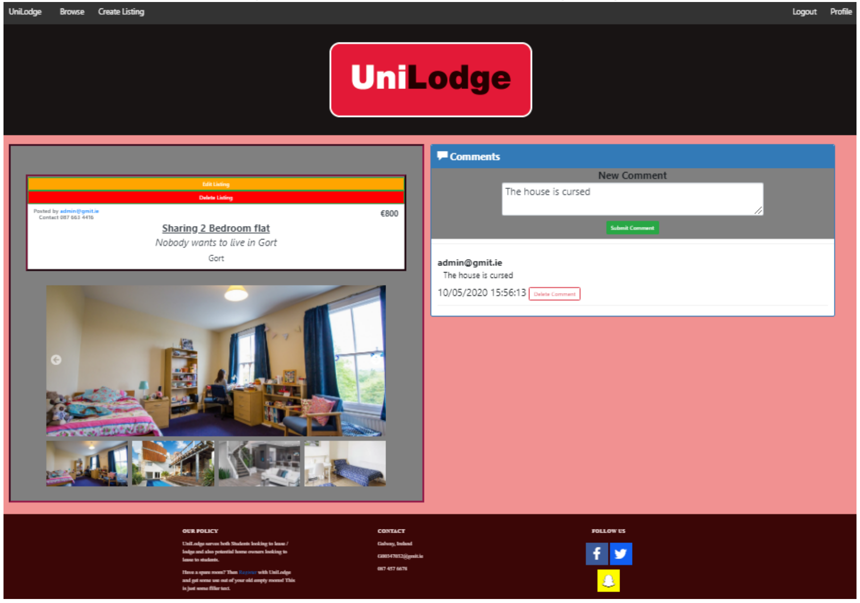
Individual Listing Page

As mentioned, once a listing is clicked the user is navigated to the specific page for that listing. This page will display some additional information about that listing, like the description, the contact information of the poster and all comments associated with that listing.

Based on the currently logged in user, delete and edit buttons may or may not be available. To make sure only the user who issued the listing can interfere with the listing, the currently logged in user is compared against the seller associated with the listing.

Comments may be posted by any logged in user, and are validated so only those who made the comment may delete them. As well as this, the associated images are displayed via an interactive gallery component, allowing a more pleasant viewing experience for the user.

Figure 4.8: Individual Listing Page



Posted by admin@gmit.ie
Contact 087 663 4416

€800

Sharing 2 Bedroom flat

Nobody wants to live in Gort

Gort

New Comment

The house is cursed

admin@gmit.ie

The house is cursed

10/05/2020 15:56:13

OUR POLICY

UniLodge serves both Students looking to lease / lodge and also potential home owners looking to lease to students.

Have a spare room? Then Register with UniLodge and get some use out of your old empty rooms! This is just some filler text.

CONTACT

Galway, Ireland

G00347032@gmit.ie

087 457 6678

Since there isn't any way to privately message posters for now, users are encouraged to communicate through posted comments. Once serious about reaching an agreement, users are encouraged to continue their discussions through alternative means like a phone call or a third party messaging service.

Browse Page

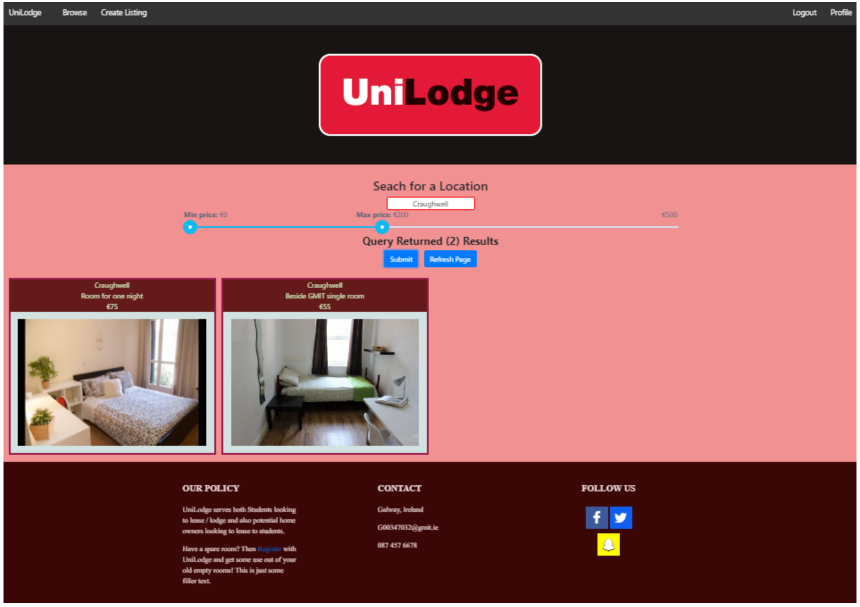
A page was included that would allow the user to specify both their search location and price range. A query object is sent to the API containing details like the chosen location, minimum price and maximum price. Once attained, a specific search is performed on all listings in the database.

```
@listings_blueprint.route('/api/listings-query/<string:Query>', methods=['GET', 'POST'])
def list_listings_by_location(Query):
    query_data = request.get_data().decode()
    query_res = list(d_a.Listings().find({'Location': query_data['location'], 'Price': {"$gt": query_data['minVal'], "$lt": query_data['maxVal']}}, {'_id': False}))

    return jsonify(query_res)
```

Listing 4.18: Querying Specific Listings

The location data may be input via a text box, and both the minimum and maximum price range values are adjustable via a double-edged slider. Queries are fully validated and input is fully responsive, searching for 'Galway' will return both 'Galway City' and 'Claregalway'.

Figure 4.9: Browse Page



Seach for a Location

Craughwell

Min price: €0

Max price: €200

€500

Query Returned (2) Results

Craughwell
Room for one night
€75

Craughwell
Beside GMIT single room
€55

OUR POLICY

UniLodge serves both Students looking to lease / lodge and also potential home owners looking to lease to students.

Have a spare room? Then Register with UniLodge and get some use out of your old empty rooms! This is just some filler text.

CONTACT

Galway, Ireland

G00347032@gmit.ie

087 457 6678

Registration and Login Pages

Both the Registration and Login components are very similar, however are separated into their respective page routes. (*For the sake of simplicity, the components are illustrated side by side in* ***Figure 4.10***).

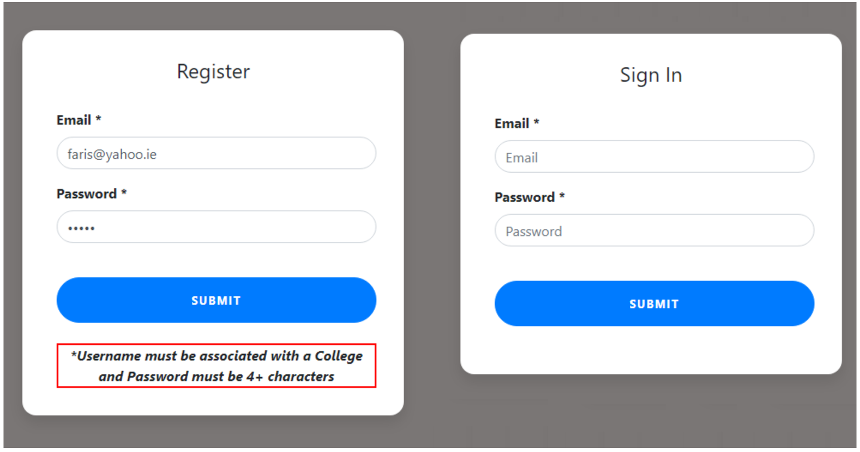
Registration

The registration form only accepts email addresses associated with colleges situated in Galway and input validators are displayed depending on the specified input (*Password must be a specified length, or email must be a valid address for example).* Once successfuly registered, the user will be prompted with the option to Login.

Login

After an account has been made, the user may log in with their registered email address. Like the Registration component, the Login component is fully responsive with Angular Snackbar popups. Once a log in is successful, the user will be redirected to their Profile and a JWT token received from the server will be saved in their browser, allowing authorization to functionality and routes like listing creation.

Figure 4.10: Registration and Login Forms



Email *: faris@yahoo.ie

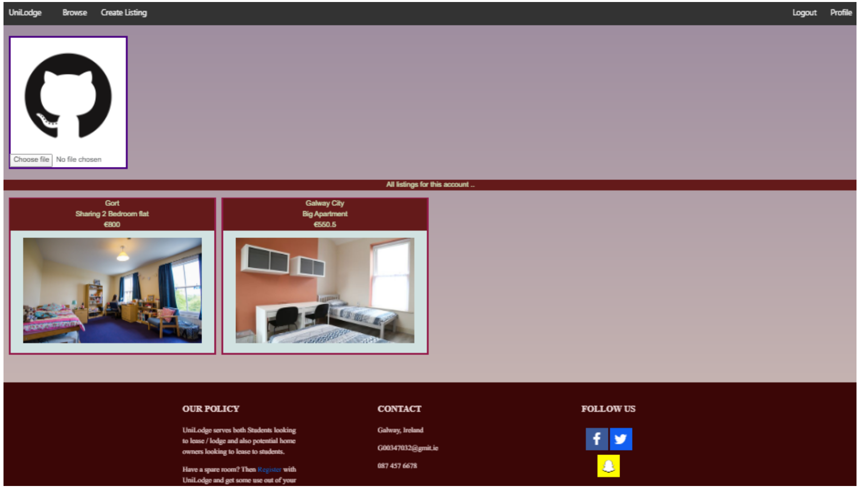
Password *: •••••

**Username must be associated with a College and Password must be 4+ characters*

Profile Page

Once registered, the user may access their profile which shows all listings associated with them. User profiles weren't initially considered but following feedback from students it was ultimately implemented. The idea is to have a space where the user can easily access and manage their listings.

Figure 4.11: User Profile



All listings for this account ..

Gort
Sharing 2 Bedroom flat
€800

Galway City
Big Apartment
€550.5

OUR POLICY

UniLodge serves both Students looking to lease / lodge and also potential home owners looking to lease to students.

Have a spare room? Then Register with UniLodge and get some use out of your

CONTACT

Galway, Ireland

G00347032@gmit.ie

087 457 6678

FOLLOW US

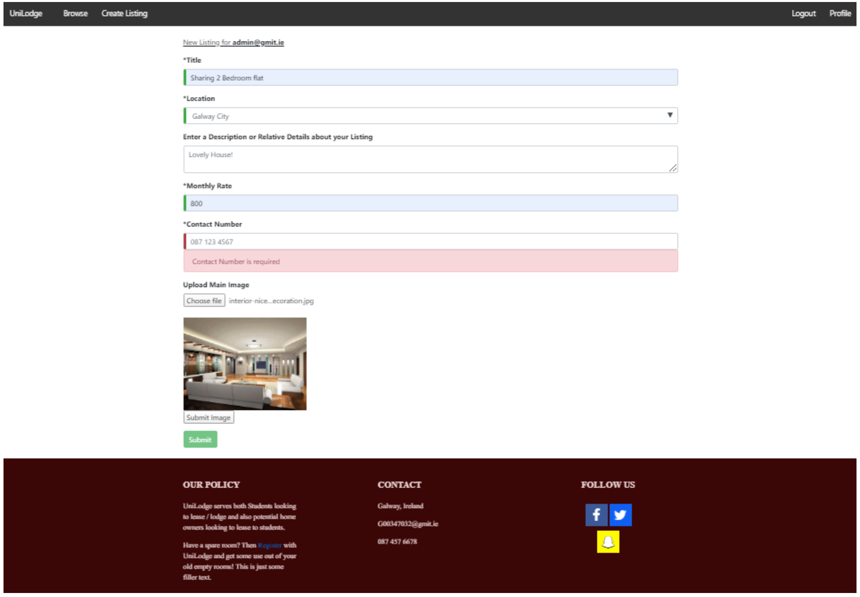
Since the profile was one of the later inclusions and wasn't a priority, it lacks areas for user interaction. Currently the profile page is static depending on listings associated with the user, it allows for traversal to the listing via the listing image and allows for minimal personalization via a profile image.

In future iterations, more personalization via the user profile would be ideal, features like page views, user feedback and a self description section would enhance the overall user experience.

Create Listing Page

Once logged in, the listing creation page becomes available. The page is JWT token authenticated, meaning only the logged in user has access to their specific creation page. The listing form is fully validated and allows for the input relevant seller information and any additional notable listing information.

Figure 4.12: Create Listing Page



New Listing for **admin@gmit.ie**

*Title

Sharing 2 Bedroom flat

*Location

Galway City ▼

Enter a Description or Relative Details about your Listing

Lovely House!

*Monthly Rate

800

*Contact Number

087 123 4567

Contact Number is required

Upload Main Image

Choose file interior-nice...ecoration.jpg

OUR POLICY

UniLodge serves both Students looking to lease / lodge and also potential home owners looking to lease to students.

Have a spare room? Then Register with UniLodge and get some use out of your old empty rooms! This is just some filler text.

CONTACT

Galway, Ireland

G00347032@gmit.ie

087 457 6678

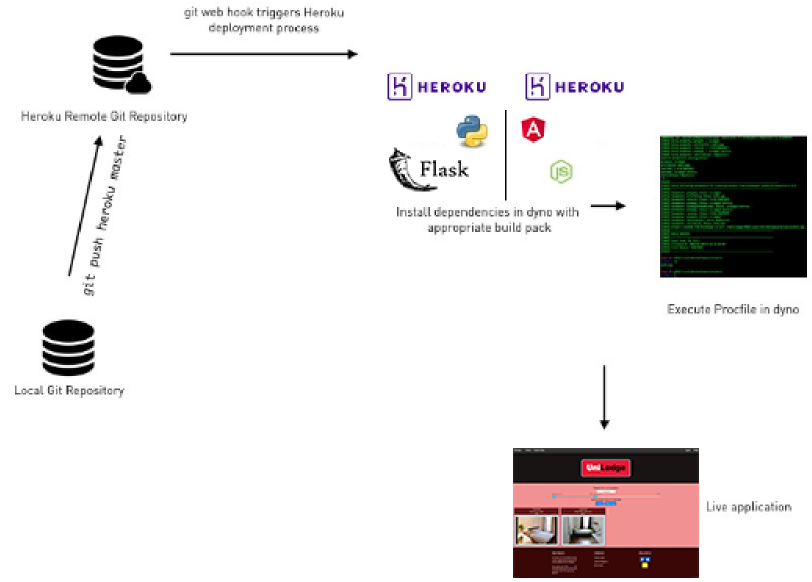
FOLLOW US

On submission, the listing is created and associated with the creator. It can be accessed and viewed by anyone but can only be edited or removed by the original poster.

4.6.3 Deployment

The application was deployed to Heroku with both the Flask API Server and Node/Express Server being hosted on separate domains.

Figure 4.13: Deployment Overview


git web hook triggers Heroku deployment process
Heroku Remote Git Repository
git push heroku master
Local Git Repository
HEROKU
HEROKU
Flask
Install dependencies in dyno with appropriate build pack
Execute Procfile in dyno
Live application

Two separate procfiles were employed, Gunicorn, a pre-fork worker model, was used to run the Flask server while Node was used to handle the client side deployment.

web: gunicorn runner:app -b 0.0.0.0:8087

Figure 4.14: Flask Procfile

web: npm install && npm start -b 0.0.0.0:8080

Figure 4.15: Node/Express Procfile

While initially the team felt it would be ideal to deploy both processes on the same domain, following supervisory meetings felt decoupling processes enhanced the overall scalability and would make the overall application much easier to maintain and test in the future.

Chapter 5

System Evaluation

The underlying goal of this project was to develop an easy to use web-application, providing both students searching for accommodation and those looking to share or advertise their rooms with a platform tailored exclusively for them. More precisely, the project goals could be summarized by the following objectives:

1. Investigate the field by gathering the opinions and ideas of students in relation to the problem area.
2. Evaluate and research potential technological inclusions including frameworks, libraries and tools that may see a place within the development cycle.
3. Based on information gathered and research on various state of the art technologies, Build a full stack system that attempts to address the outlined problem domain while collaborating and communicating as a team throughout development

5.1 Evaluation of Objectives

This brief section will describe how each objective was addressed over the course of the project.

- *Investigate the field by gathering the opinions and ideas of students in relation to the problem area.*

  Before project development began, a survey aimed at students was distributed, allowing the team to gain valuable insight on necessary key features, similar applications and by extension set the foundation for which to commence development.

During development, especially during the earlier stages, students were heavily involved in steering the course of the project, providing key insights. For example, a feature that was intended to be implemented would allow users to 'follow' other users, following testing by students, it was determined that a commenting on system should be a much higher priority.

- *Evaluate and research potential technological inclusions including frameworks, libraries and tools that may see a place within the development cycle.*

  Research on a wide array of technologies was conducted which turned out to be highly beneficial when it came to determining the best course to take with different components of the system. Research and prototyping helped eliminate frameworks and tools that the team felt had no place in the project or wouldn't be as effective in practice as they would be in theory.

  As as example, if the project prototype was never developed and research was never conducted, the team may have ultimately considered VueJS as their frontend framework, initiating the development phase only to realise after time that the chosen technologies aren't as compatible with this specific framework as they would be with another.

  Instead, the team were able to take measures and identify the strengths and weaknesses of different technologies working together in unison and choose a stack that could both enhance the development skill of the team and yield a robust and user friendly application.

- *Based on information gathered and research on various state of the art technologies, Build a full stack system that attempts to address the outlined problem domain while collaborating and communicating as a team throughout development.*

  Not only does the developed platform allow users to exchange services and communicate with each other, achieving the end-goal of the project, but additional functionality was implemented based on previously gathered information and researched technologies, including login authentication, password security, user profiles and a gallery for images to name a few additional features.

  While there is scope for improvement within the overall system, in terms of accomplishing a previously identified goal the application has absolutely accomplished it's objective.

Unfortunately collaboration and communication between team members is something that could have seen improvements during development, this outcome as well as system flaws and improvements will be discussed in the following chapter and further in this chapter respectively.

5.2 Testing

While the overall application was not extensively tested, various methods of testing were integrated in both a development and production environment. This section will briefly cover the methods incorporated to evaluate the robustness of the overall system.

5.2.1 Unit Testing

Unit testing was minimally incorporated using both Karma and Jasmine. These tests were carried out on various functions and components before they were fully integrated into the system.

```
beforeEach(() => {
    TestBed.configureTestingModule({
        providers: [
            {
                provide: Router,
                useValue: {
                   url: ’/accommodation/’
                }
            }
        ]
    });

let route: ActivatedRoute
let userService: UserService
let router: Router
let listingService: ListingService
component = new HomeComponent(route,userService,router,
    listingService);
})

it(’checks that listings are returned’, ()=> {
    spyOn(component, "getListings");
    component.getListings();
    expect(component.listings).toEqual(listings[]);
})
```

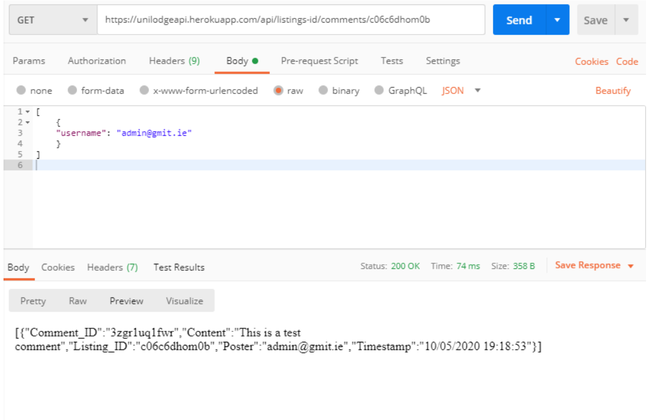
Listing 5.1: Testing with Karma

Ideally, Karma unit tests would be written to test new functions as they were implemented. If this was practised throughout development, a lot of errors would have been caught and addressed much sooner.

Postman

Additionally, Postman was used to test newly developed API endpoints. This allowed issues to be caught before full integration to the client began. This approach allowed the team to make requests to the API and ensure the expected status codes and responses were returned.

Figure 5.1: Postman Finding User Associated Comments



https://unilodgeapi.herokuapp.com/api/listings-id/comments/c06c6dhom0b

```
[
    {
    "username": "admin@gmit.ie"
    }
]
```

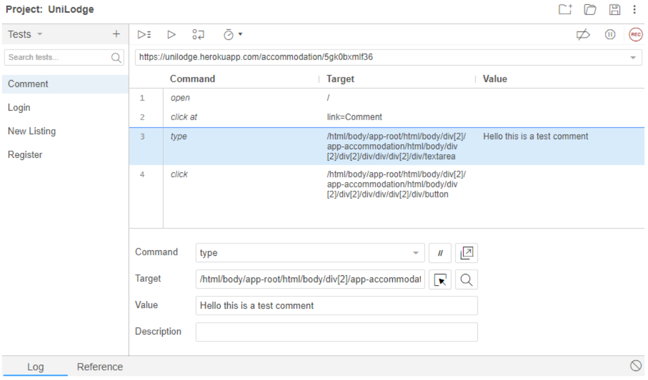
Status: 200 OK Time: 74 ms Size: 358 B

```
[{"Comment_ID":"3zgr1uq1fwr","Content":"This is a test
comment","Listing_ID":"c06c6dhom0b","Poster":"admin@gmit.ie","Timestamp":"10/05/2020 19:18:53"}]
```

5.2.2 System Testing

Selenium was used to test the overall system functionality and to validate the overall application. Selenium tests were ran against both Firefox and Chrome to test the overall functionality.

Figure 5.2: Selenium Testing Functionality



https://unilodge.herokuapp.com/accommodation/5gk0bxmlf36

	Command	Target	Value
1	*open*	/	
2	*click at*	link=Comment	
3	*type*	/html/body/app-root/html/body/div[2]/app-accommodation/html/body/div[2]/div[2]/div/div/div[2]/div/textarea	Hello this is a test comment
4	*click*	/html/body/app-root/html/body/div[2]/app-accommodation/html/body/div[2]/div[2]/div/div/div[2]/div/button	

Command: type
Target: /html/body/app-root/html/body/div[2]/app-accommodat
Value: Hello this is a test comment
Description:

Selenium allowed the testing of vital functionality via automated clicks, input and submissions among other potential options. In **Figure 5.2**, a URL could be specified for Selenium to run the pre-written test against. By defining specific actions and elements to interact with by specifying their xpath, Selenium can automate an isolated test against the specified path to ensure it functions as expected.

5.3 Limitations and Opportunities for Improvement

While the proposed objectives for the project were achieved, there are still multiple areas within the developed system that can be vastly improved.

5.3.1 Testing

Although periodic unit testing with Karma, Jasmine and Selenium was conducted and usability tests were incorporated along with acceptance and verification testing to validate the project objectives, overall testing could have been a lot more prevalent throughout the system.

Unit Testing wasn't as abundant as it should have been, however during the early stages of development Karma and Jasmine were utilized at an acceptable level to ensure functionality like navigation and intractable components worked as intended. With that said, System Testing was an area of testing that really fell short during the project life-cycle.

System testing earlier in the development life cycle would have identified the issue of Flask not correctly serving Angular build files. Angular build files weren't generated until time came to deploy the application, so Flask only ever served development files. Build files were required for Heroku to be able to host the application, meaning the workaround was necessary, but could have been avoided if system tests were incorporated earlier and builds were generated periodically to test deployment. If this was carried out, the issue with Flask would have been able to be addressed much earlier in the development cycle.

5.3.2 Lack of Additional Functionality

The developed system, while adhering to the outlined goals, does lack in the area of additional user functionality that would see it be a serious alternative to applications of a similar nature.

Features like profile enhancements, additional user functionality and the overall aesthetic of the application are on the list of future enhancements that will see implementation in future development cycles.

5.3.3 Device Portability

Following deployment, the application was tested on mobile, and while it works, the overall display of pages and components isn't ideal and can be improved upon. This can be solved in future by including some media and device responsive CSS on the most problematic pages.

This could have been addressed during development by consistently testing the application on alternative devices, however this was never a priority or an objective. In the future it would absolutely correct the cross-device aesthetic of the application.

5.3.4 API Request Speed

Something that the team had attempted to address at multiple points during the development cycle is the stuttering of some API interactions. This delay would result in components loading faster than the API data could be retrieved, an undesirable feature that isn't pleasant to see in an application.

Different methods like lazy loading and progress / loading bars were trialed and implemented to attempt to address the inconvenience, however, while it was mostly corrected with the aforementioned methods, it does still remain a slight problem with the application with some components more than others.

5.4 Overall Evaluation

Overall the system functions as was intended, a platform for both students searching for and offering accommodate services. The requirements outlined in **Section 1.1**. Even though much was accomplished, the importance of testing in a system of this scale was learned, regardless if in terms of industry standard systems the developed application is minuscule in comparison, it only highlights the importance of maintainability during development.

With that being said, not extensively testing did yield a silver lining, the team did learn how to overcome unexpected issues that may arise and find unique solutions that wouldn't traditionally be sought out.

Finally, the overall development stack is considered by the team a worthy choice. The fact it was diverse stack not only enhanced knowledge of the individual components, but also first-hand experience on how to wire technologies together that wouldn't traditionally be coupled is a valuable learning outcome that can be applied to future projects.

Chapter 6

Conclusion

The main goal set out by the team from the beginning was to create a unique platform for students in the Galway area to both advertise and search for accommodation. Concluding the project, as a team we can say the goal has been achieved in the form of a web-application exclusively for students local to Galway.

UniLodge is intended to fill the market gap that was identified during the research phase. The finding showed there was no service for students to exclusively interact and advertise accommodate services. As a team we hope that by developing this service we can help bridge the gap between the two parties that previously had no access to a platform of this nature.

6.1 Key Insights & Learning Outcomes

Along the way, a wide range of learning outcomes and insights were gained, with most of them encountered indirectly and unexpectedly. This section will highlight and discuss the key insights acquired by the team.

- ***The Importance of Testing***.

  The team knew testing was an important part of building a robust system, however had to learn the hard way how vital it was in ensuring unexpected blockers and outcomes didn't outright halt and disrupt development.

  In future, more time and resources will undoubtedly be dedicated towards testing, the team saw first hand the missed investment and the associated result of not incrementally testing.

- ***Working with Diverse Technologies***.

  While Angular, Flask, Python and MongoDB aren't exactly uncommon names to see in development stacks, working with specifically those names was found to be a rocky experience during the initial development phase. As it turned out, there wasn't a whole lot of content to be found integrating those three technologies.

  To get a handle on how to assemble the stack meant reading through base level documentation to try and figure out where and how everything would fit together. Unfortunately, there was no cookie cutter method to get everything set up unlike there would be when working with a traditional MEAN stack.

  However the importance of effective research and documentation analysis was picked up, allowing the inclusion and wiring together of other components (*albeit unintentional and unexpected*) and different methods of integrating them.

- ***The Effect of Communication during Development***.

  Unfortunately, internal communication issues did cause disruption during the project cycle. This would ultimately lead to loose ends, unfulfilled commitments and ambiguity surrounding major elements of the project. As a result, the scope of the project had to frequently be readjusted and redefined, leading to constant backlogs and ultimately features that wouldn't see implementation before the deadline.

  Even though it may seem obvious from an outside view that effective communication is essential within a project, experiencing and having to adapt to it first-hand really did reinforce it's importance for the team.

- ***Consultation with End Users***.

  The team were very fortunate to have access to a wide range of potential end users during development. While the initial base-vision for the overall system was defined at an early stage, constant input from fellow students helped shape the vision into something tailored and specialized towards a specific target audience.

The highlighted conclusive insights consist of a handful of outcomes that the team consider to be some of the more valuable takeaways from the project. As a team, our goal for the future is to take the highlighted outcomes as well as those passively attained and apply them to any future tasks.

6.2 Future Development

While the project objectives and goals were met, there is undoubtedly still room for growth. This section will outline some key areas that will definitely see future development, the features will consist of components that couldn't be implemented due to time constraints or features that the team feel would be practical additions.

6.2.1 Browsing Features

A list of search enhancing features were recommended by others and considered by the team during development, a brief description of each of these features to be developed in the future will be described here.

GPS Functionality

A potential inclusion the team felt excited about was an interactive map that would enable the user to search for nearby listings based on their location, or even compare the distance between the listing and their educational institute.

More Listing Depth

Currently the information available on individual listings is limited. In future iterations more specific content would ideally be included, like how long the location is available for or closest educational institute for example.

More Browse Functionality

When browsing for listings currently, two parameters may be used for searching, price and location. In future, including parameters that allow to sort by date posted, poster and potentially rating would be ideal.

6.2.2 User Features

Features that would enhance the overall experience of the user are considered to be some of the more important potential inclusions for future development, some of the more important features will be described below.

Profile Customization

More details about the user would give more depth to the overall application. Inclusions like positive or negative reputation, previous postings or page views would enhance the overall user experience and by extension the overall service.

Follow Feature

The ability to 'follow' other users and be notified of relevant actions taken by that user was a feature that was heavily considered at early stages. Following discussions with other students, the overwhelming majority of those consulted concluded that other features like browsing or commenting were far more important and should take precedence.

Private Messaging

While users can communicate through comments and organize between themselves their conditions or follow up with more detailed conversation through a phone call or text message, an ideal scenario would allow both parties to discretely message each other when discussing more sensitive topics.

6.2.3 Miscellaneous Features

A number of potential miscellaneous features were identified throughout the project cycle.

Mobile Friendly

Following deployment the application was tested on mobile devices and while the functionality was working as intended, the page views weren't as intended, overall the mobile experience is unpleasant. In an age where the majority of similar applications are accessed via mobile devices, this would be a relatively high priority.

More Overall Functionality

Generally speaking, more pages and features are needed for this application to compete with any similarly natured service. Once more necessary features are included, the team would like to see inclusions like payment authorizations or partnerships with local educational institutes.

6.3 Final Thoughts

Initially the team had set out to help address a real world problem, relevant to not only us, but to a lot of friends and fellow students. The problem itself isn't one that can be fixed with a single application, there isn't a one size fits all solution, especially within the given time frame. With that said, being able to address it in the way the team has achieved is something to very happy with.

We've developed a platform for students to exclusively interact with each other, advertise and search for accommodate services, communicate with each other and hopefully follow-up with each other following positive discussions. A lot of development was driven by the time and enthusiasm of other students who have also played a big role.

In conclusion, our hope is that UniLodge helps alleviate the burden or pressure some students may experience when searching for accommodation local to their educational institute. Along the way, as well as developing the solution, the team gained valuable insights and enhanced their technical ability and knowledge, and while it wasn't always a positive experience, the team found that the majority of knowledge was gained not during the positive sections but during the unsavory phases.

Bibliography

[1] Angular. URL: https://angular.io/.

[2] Angular documentation. URL: https://angular.io/docs.

[3] Expressjs. URL: https://expressjs.com/.

[4] Flask microframework. URL: https://flask.palletsprojects.com/en/1.1.x/.

[5] Github. URL: https://github.com/.

[6] Gitlab. URL: https://gitlab.com/.

[7] Heroku. URL: https://dashboard.heroku.com/.

[8] Honcho. URL: https://honcho.readthedocs.io/en/latest/.

[9] Javascript. URL: https://www.javascript.com/.

[10] Mongo atlas documentation. URL: https://docs.atlas.mongodb.com/.

[11] Prototype heroku. URL: https://prototype-crud.herokuapp.com/.

[12] Prototype skeleton project. URL: https://github.com/farisNassif/FourthYear_FinalProjectSkeleton_Inactive.

[13] Python documentation. URL: https://docs.python.org/3/.

[14] Scrum methodoloy. URL: https://www.scrum.org/.

[15] Trello. URL: https://trello.com/.

[16] Developer practices and habits survey results, 2018. URL: https://insights.stackoverflow.com/survey/2018#development-practices.

[17] Younes Kkhourdifi Alae El Alami, Mohamed Bahaj. Supply of a key value database redis in-memory by data from a relational database. *International Journal of Innovative Research in Computer and Communication Engineering*, pages 49–50, 2019.

[18] Kostas Maniatis Alex Kyriakidis and Evan You. The majesty of vue.js. pages 6–7, 2016.

[19] Dantala O. Oyerinde Bakwa D. Dunka, Edim A. Emmanuel. Simplifying web application development using - mean stack technologies. *International Journal of Latest Research in Engineering and Technology*, 04:70–74, 2018.

[20] K K Baseer. A systematic survey on waterfall vs. agile vs. lean process paradigms. *Journal on Software Engineering*, pages 34–36, 2015.

[21] Pragati Bhardwaj. Analysis of stack technology: a case study of mean vs. mern stack. *International Journal of Innovative Research in Computer and Communication Engineering*, 06:3610–3614, 2018.

[22] Benjamin Haefnera Gisela Lanzaa Constantin Hofmanna, Sebastian Laubera. Development of an agile development method based on kanban for distributed part-time teams and an introduction framework. pages 46–47, 2017.

[23] Ryan Dahl. Visual studio code. URL: `https://nodejs.org/en/`.

[24] Alek Al-Zewairi1 et al. Agile software development methodologies: Survey of surveys, 2017. URL: `https://www.scirp.org/journal/paperinformation.aspx?paperid=75114`.

[25] Anca Apostu et al. Study on advantages and disadvantages of cloud computing – the advantages of telemetry applications in the cloud. pages 119–121, 2017.

[26] Diego Serrano et al. Linked rest apis: A middleware for semantic rest api integration. pages 41–44, 2017.

[27] Roy T. Fielding et al. Reflections on the rest architectural style and "principled design of the modern web architecture". pages 5–7, 2017.

[28] S Sanfilippo et al. Redis. URL: `https://redis.io/`.

[29] Arie van Bennekum Alistair Cockburn Ward Cunningham Martin Fowler et al Kent Beck, Mike Beedle. Manifesto for agile software development. 2001.

[30] Mihai Liviu. Comparative study on software development methodologies. *Database Systems Journal*, pages 41–42, 2012.

[31] Microsoft. https://marketplace.visualstudio.com/. URL: `VisualStudioCodeExtensions`.

[32] Microsoft. Visual studio code. URL: `https://code.visualstudio.com/`.

[33] Faris Nassif. Student survey results. URL: `https://docs.google.com/forms/d/1EvcqNezkSgqm7b2vaaSDztwU-ZzRO3phARMnbxCWbTU/viewanalytics`.

[34] Klaus Nygard. Single page architecture as basis for web applications. pages 41–43, 2015.

[35] Rinkle Rani Aggarwal Rupali Arora. Modeling and querying data in mongodb. *International Journal of Scientific & Engineering Research*, pages 141–142, 2013.

[36] Steve Vinoski Stefan Tilkov. Nodejs: Using javascript to build high-performance network programs. pages 82–83, 2010.

[37] Sverker Söderlund. Performance of rest applications found in four different frameworks. pages 41–44, 2017.

[38] Lucidchart Content Team. Waterfall overview. URL: `https://www.lucidchart.com/blog/pros-and-cons-of-waterfall-methodology`.

[39] Samia Farooq Zahid Ali Masood. The benefits and key challenges of agile project management under recent research opportunities. *International Research Journal of Management Sciences*, pages 20–21, 2017.

Appendices

Source Code

https://github.com/farisNassif/UniLodge

Heroku Web Application

https://unilodge.herokuapp.com/home

Swagger API

https://app.swaggerhub.com/apis-docs/GMIT7/Unilodge-API/0.2-oas3#/

Survey Results

https://docs.google.com/forms/d/1EvcqNezkSgqm7b2vaaSDztwU-ZzRO3phARMnbxCWbTU/viewanalytics

Screencast

https://www.youtube.com/watch?v=VaFmgG3MCNk