UniLodge

An AirBnB Alternative for Students

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

BY
Faris Nassif
Aaron Burns

May 4, 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 students, and for those living locally or living at a reasonable distance, who may not have the outlet to advertise their spare room or inhabited apartment.

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 tenants 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, namely Waterfall, Rapid Application Development and Agile to name a few. 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.

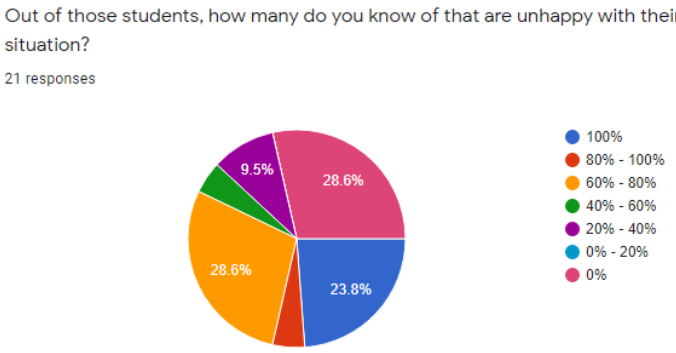
This subsection will act as a brief analysis of the responses given for the major questions in the survey.

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

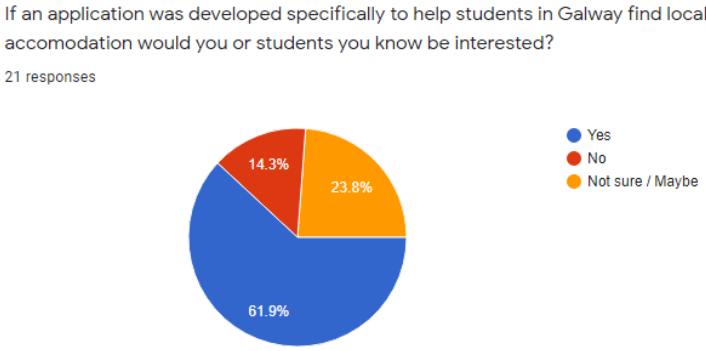


Following the closure of the survey, it can be seen from *Figure.2.1* that just over a quarter of students are happy with their current living situation, a definitive result. Based on this response, it can be taken as a given that there is definitely a gap for a student-specific accommodation related 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



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

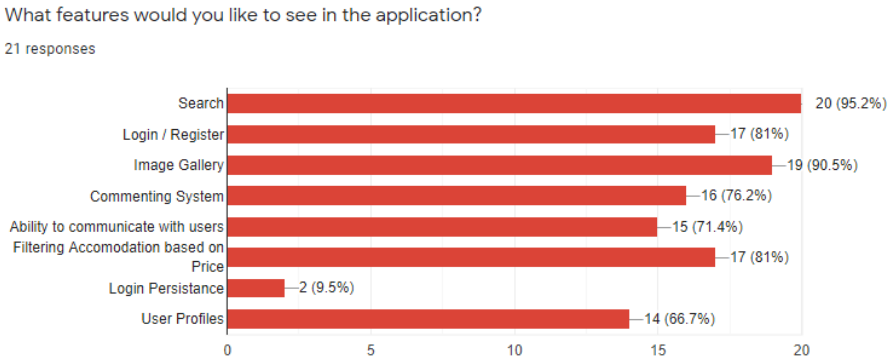
Following the analysis of the results illustrated in *Table.2.1*, it became clear there wasn't an absolute front runner that all recipients would be able to name on the spot. The results were scattered, with Adverts being the most known related application. The most known application in this field being 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.

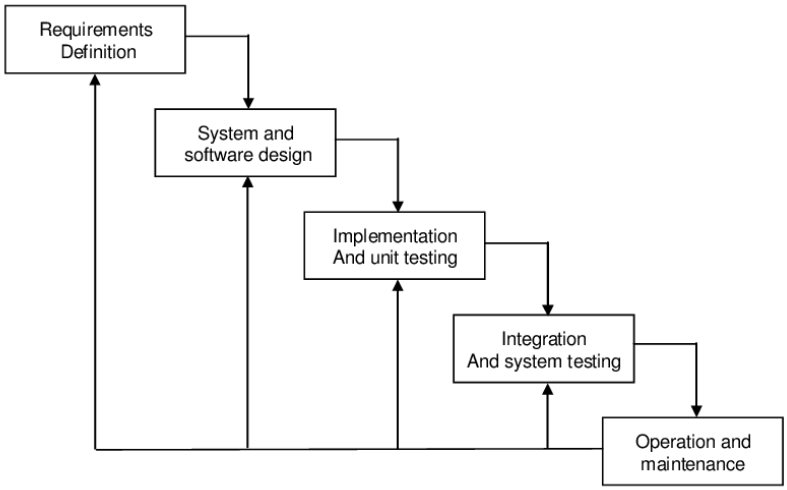
2.3 Determining the Development Methodology

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



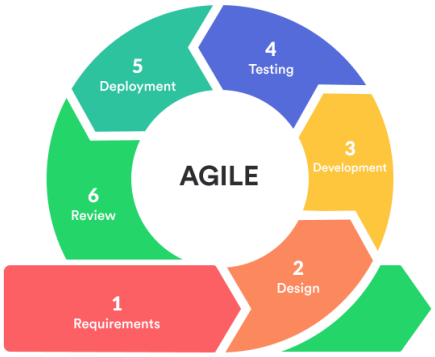
The Waterfall model is arguably the most well known development model, which is likely attributed to how long the model has been around, not to mention it's overall simplicity. Waterfall being an easily understood model naturally means it isn't difficult to manage which is mainly attributed to it's strict 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. Following more research into methodologies including Extreme Programming (XP) and Feature Driven Development (FDD) TODO THE REST OF THIS AT A LATER DATE**

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 (Scaling needs fixing)

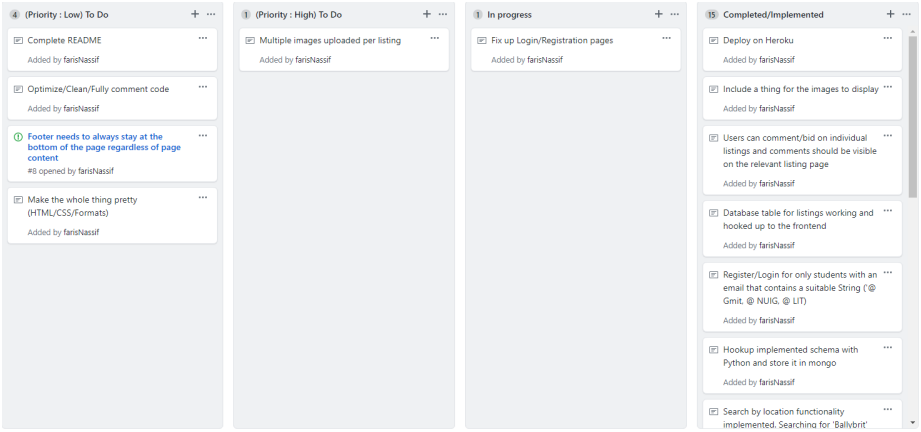


Figure 2.6 above consists of the Kanban board used during the development of the project. The Kanban board consists of four main columns:

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.

```
1  const express = require('express');
2  const path = require('path');
3
4  /* Static path definition */
5  app.use(express.static(__dirname + '/dist/client'));
6
7  app.get('/*', function(req,res) {
8
9  /* Serving static files */
10 res.sendFile(path.join(__dirname+'/dist/client/index.html'));
11 });
12
13 /* Exposing a port to listen for requests on (Served with NodeJS,
       Discussed below!) */
14 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

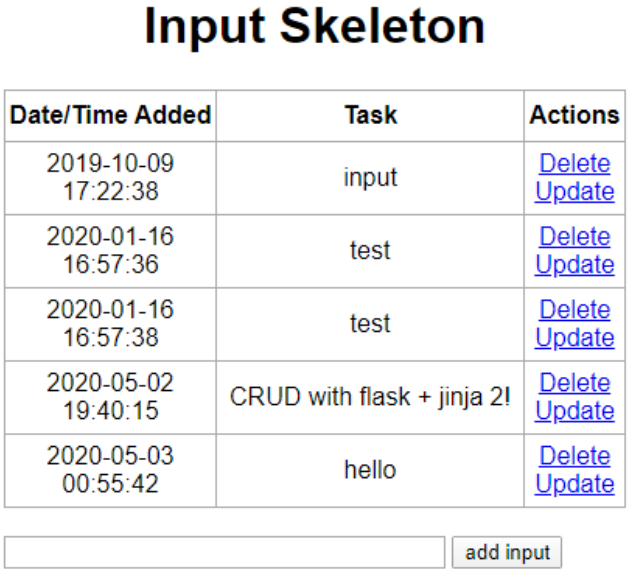
Redis, meaning REmote DIctionary Server is an in-memory distributed key-value data-store. Redis supports multiple types of data structures including, streams, bitmaps, sets and spatial indexes to name a few [28]. Key value databases excel at providing rapid access to information that has a corresponding 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 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. 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 3.1: Basic CRUD Prototype



The resulting prototype as seen in **Fig.3.1** 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].

3.2.1 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].

3.2.2 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.

3.3 Chosen Technologies

Following research, input from supervisors and trials of the aforementioned technologies via the 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.

3.3.1 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.

3.3.2 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

3.3.3 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

3.4 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.

3.4.1 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.

3.4.2 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 3.2: Honcho Procfile

Honcho allows a procfile to be defined (*As outlined in* ***Fig.3.2***) 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 3.3: Heroku Procfile

The Heroku procfile (*As outlined in* ***Fig.3.3***) 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.

3.5 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.

3.5.1 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.

3.5.2 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.

3.5.3 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.

3.5.4 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.

3.5.5 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.

3.6 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.

3.6.1 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.

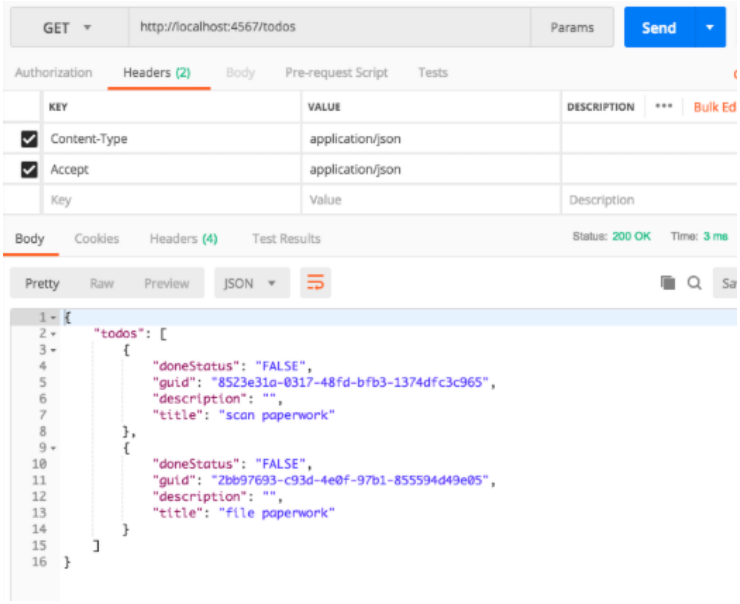
Extensions

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.

3.6.2 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 3.4: Postman API Requests



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.

3.6.3 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.

3.7 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].

3.8 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.

3.8.1 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.

3.8.2 Jasmine

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

```
1 describe('sorting the list of users', function() {
2   it('sorts in descending order by default', function() {
3     var users = ['faris', 'bob', 'jeff'];
4     var sorted = sortUsers(users);
5     expect(sorted).toEqual(['bob', 'faris', 'jeff']);
6   });
7 });
```

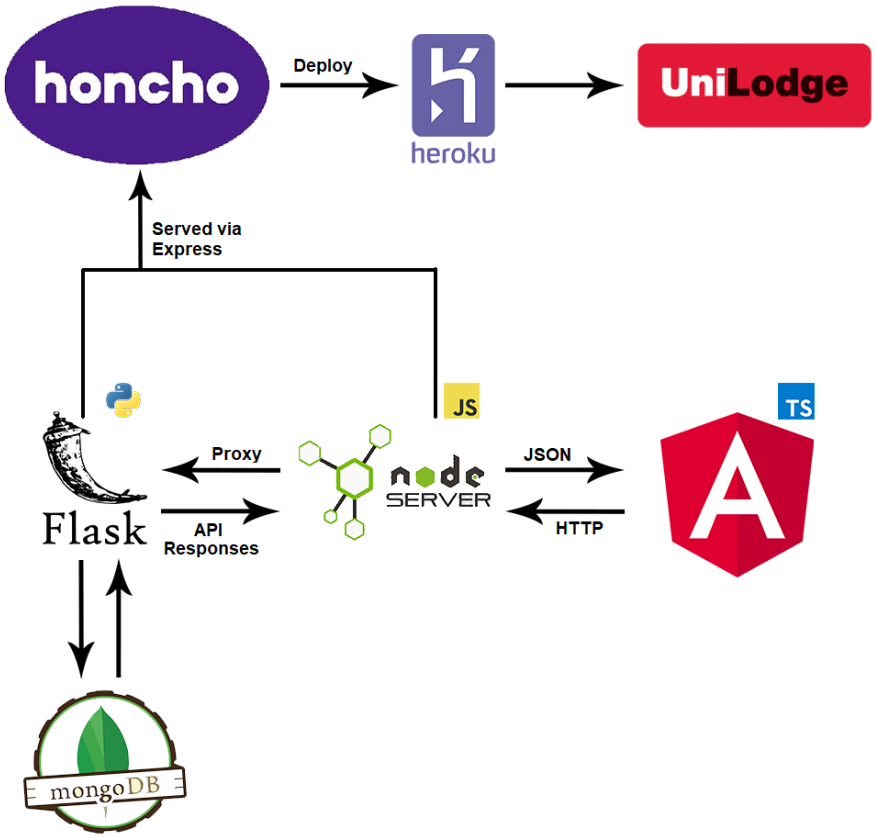
Listing 3.4: Test Definition with Jasmine

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, 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.

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 via cloud services, enabling a software as a service distribution model.

A Honcho procfile is used to enable the execution of both the Flask API server and the Express/Node server using just one worker on the cloud. Heroku will use Express/Node to serve the Angular build files while RESTful requests made to the Express server will be redirected to Flask with the use of a server proxy, allowing interactions between the primary components of the system to remain unhindered.

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.

4.2 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.

4.2.1 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 4.1: 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.

4.2.2 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.

```
1  from flask_pymongo import MongoClient
2
3  # URI for database connection was stored in a local file
       for security
4  with open("mongo_uri.txt", "r") as uri:
5      MONGO_URI = uri.read()
6
7  cluster = MongoClient(MONGO_URI)
8  database = cluster["UniLodge"]
9
10 def Users():
11     users = database["Users"]
12     return users
```

Listing 4.2: Database Accession Class

Based on the defined class in **Listing.4.1**, 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 define the accession class.

```
1  # Definition of Database class as d_a
2  import data.database_accessor as d_a
3
4  @temp_users_blueprint.route('/api/users', methods=['GET'
       ])
5  def list_users():
6      # The object d_a allows full control of the
           collection
7      userList = list(d_a.Users().find({}, {'_id': False}))
8
9      return jsonify(userList)
```

Listing 4.3: Interacting with Collections

As can be seen in **Listing.4.2**, 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.

4.2.3 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.

```
1 import bcrypt as bc
2
3 def generate_hash(plain_text_password):
4     pw_hash = bc.bcrypt.hashpw(plain_text_password.encode
          ("utf-8"), bc.gensalt())
5     return pw_hash
```

Listing 4.4: Hashing a Password with Bcrypt

The function defined in **Listing.4.1** 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.

```
1 import bcrypt as bc
2
3 def check_password(plain_text_password, hashed_password):
4     if bc.bcrypt.checkpw(plain_text_password,
          hashed_password):
5         result = True
6     else:
7         result = False # Password is wrong :(
8     return result
```

Listing 4.5: 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.

4.3 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.

4.3.1 Flask API

The API is completely written with Flask and documented with Swagger. TypeScript handles the sending of requests to the API. Simple methods that 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 4.6: Basic Request to the API

The TypeScript method in **Listing 4.6** will seek the it's 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 4.7: 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
```

```
5
6  app.register_blueprint(login_blueprint)
7  app.register_blueprint(register_blueprint)
8  app.register_blueprint(users_blueprint)
9  app.register_blueprint(listings_blueprint)
10
11 if __name__ == "__main__":
12     app.run()
```

Listing 4.8: Main Flask Runner

API Documentation

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

4.3.2 Login/Authentication

4.3.3 Web Token

4.3.4 Angular Routing

4.4 Application Tier

4.4.1 Pages?

4.4.2 Styling?

4.4.3 Deployment

Chapter 5

System Evaluation

5.1 Testing

Approaches to testing something something

5.1.1 Types of Testing

5.1.2 Deployment Testing

5.1.3 Unit Testing?

5.1.4 System Testing

5.1.5 Some more things here I'm neglecting

5.2 Overall Evaluation

Chapter 6

Conclusion

6.1 Overview

6.2 Learning Outcomes

6.3 Final Thoughts

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/farisNassif/UniLodge/1`

Survey Results

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

Screencast

`TODO`