UniLodge

An AirBnB Alternative for Students

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

BY
Faris Nassif
Aaron Burns

April 30, 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

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.

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.

1.1 Objectives of the Project

As previously mentioned, the main objective 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.
In order to reach that goal, by extension, other objectives had to be set out to enable TODO MIGHT GET RID OF THIS

- Evaluate and investigate the frameworks and tools available for creating a platform independent web application.
- Create and develop an application that will allow users to arrange or offer lodging services for students.
- Identify and compare applications of a similar nature, critically analyze those alternatives and apply any beneficial findings to our project.
- The application will, at a minimum, allow users to register an account, login, post listings and communicate with other users via a commenting system.

1.2 Summary

This section will contain a brief overview of each chapter outlined in this dissertation.

1.2.1 Methodology

In this chapter, the processes undertaken during the life cycle of the project in regards to planning and development will be outlined. The decisions, thought processes and influential factors leading up to those processes and design implementations will also be described.

1.2.2 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.

1.2.3 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.

1.2.4 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.

1.2.5 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 assigned 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, the overall methodical conclusion and it's influence on the initial project architecture.

2.1.1 Initial Meeting and Brainstorming

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 and 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?***

2.1.2 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**.

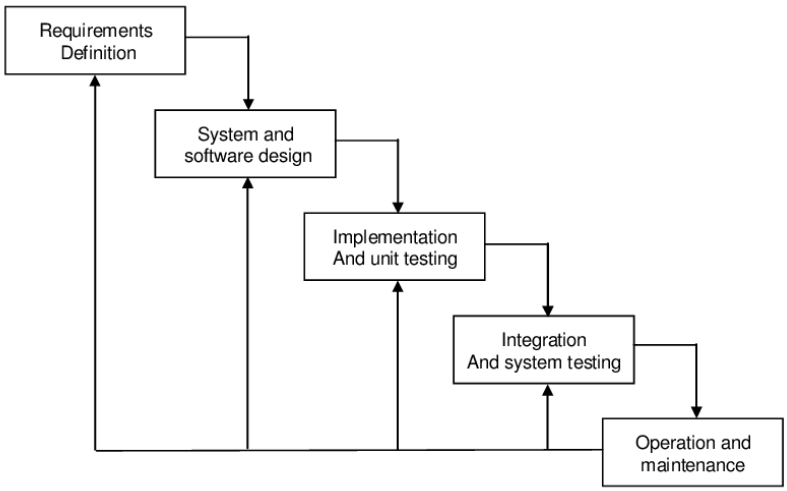
2.2 Determining the Methodology

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.

2.2.1 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 [10]

Figure 2.1: Waterfall Model Cycle

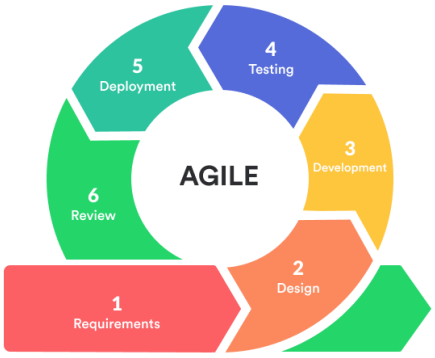


The Waterfall model is arguably the most well known development model, which is likely attributed to how long the model has been around, not to mention it's overall simplicity. Waterfall being an easily understood model naturally means it isn't difficult to manage which is mainly attributed to it's strict requirement definitions that ensures requirements are clearly defined, well received and understood [16]. Each phase is processed and undertaken uniformly, meaning phases are completed sequentially and don't overlap [17].

2.2.2 Agile

Initially described in the Manifesto for Agile Software Development, Agile is a an array of principles and methods for project management [15]. 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 [18].

Figure 2.2: Agile Cycle



The popularity of Agile development methodologies among the software development industry has increased since their introduction in the mid-nineties [13], with almost 86% of surveyed international software developers using agile methodologies in their work [6].

2.2.3 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 2.1: 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.

2.3 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**

2.3.1 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 [12].

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 2.3: Github Kanban (Scaling needs fixing)

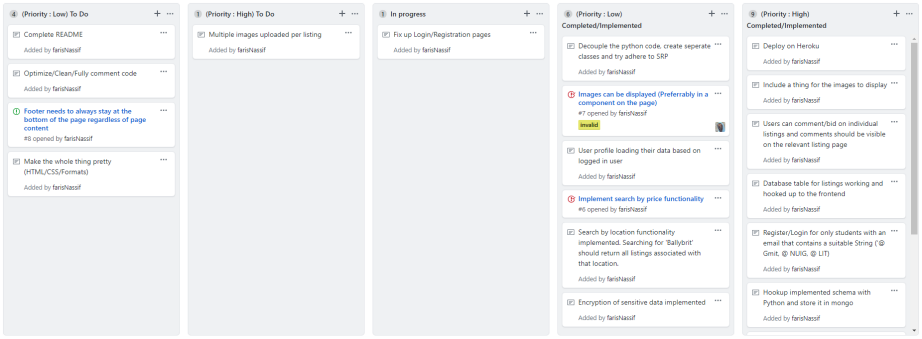


Figure 2.3 above consists of the Kanban board used during the development of the project. Initially the team had planned to use Trello [5] for project tracking, however following discussions with other students and finally supervisory discussions it was found that Github had a built in Kanban function.

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.

2.3.2 SCRUM

2.4 Version Control

Initially Github [1] was the only platform considered for version control coverage, however following discussions with other students, Gitlab [2] 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 in 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.

2.4.1 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.

2.5 Testing

2.5.1 Types of Testing we'll do

2.6 System Architecture

2.6.1 Components of Project

2.6.2 Brief discussion about moving pieces

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 developers JavaScript competency [9]. 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.

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 [11] 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

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 [8]. 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 [14]. 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 [7].

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, 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

3.2.2 The Flask Micro-framework

3.2.3 MongoDB

3.2.4 ExpressJS / Node

3.3 Deployment

3.3.1 Heroku

Heroku is an open source cloud application platform, providing extensive and well developed services for many aspects of the deployment life-cycle [3]. 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.

3.3.2 Honcho

Honcho is a Python port of Foreman, a tool for managing procfile-based applications [4]. Honcho's purpose is to abstract away any complications of the procfile format and allow the application to be either ran 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.3.1***) 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.

3.3.3 Ngrok

Chapter 4

System Design

4.1 Overview

4.2 Data Tier

4.2.1 MongoDB

4.2.2 Schema

4.2.3 Hashing

4.2.4 Another thing i'm forgetting

4.3 Logic Tier

4.3.1 Login/Authentication

4.3.2 Web Token

4.3.3 Flask API

4.4 Application Tier

4.4.1 Angular Stuffs

4.4.2 Routing?

4.4.3 Something else ..

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

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

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

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

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

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

[7] 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.

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

[9] 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.

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

[11] 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.

[12] 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.

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

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

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

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

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

[18] 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