TUTORTINDER©

TECHNICALREPORT

JAVA123

Sophie Walker – 117349696
Olivia Roche – 117477344
Emma Looney – 117326096
Adam O' Ceallaigh – 117320641
Sean O' Sullivan - 117322633

TABLE OF CONTENTS

APPLICATIONPITCH

Of UCC's 21,204 Undergraduate and Post-graduate students, 9,966 of them commute daily to campus. Over 4000 students applied for the total of 1,277 beds available in UCC's student accommodation for the 2018/2019 academic year.

A lack of new student accommodation being built combined with rising student numbers has created a housing crisis around UCC. This has led to an increase in the number of students choosing to commute to college and an increase in the commuting distance.

These students face longer daily commutes and a reduction in free time available for studying, working and other commitments. Longer commuting times create barriers to access student supports such as grinds services.

Commuting students relying on public transport cannot properly take advantage of the student supports offered by UCC. This leaves them at a significant disadvantage if they require grinds as their lack of free time compounded with their distance from UCC makes sourcing suitable grinds tutors almost impossible. This will cause them to suffer come exam time.

Our app, TUTORTINDER aims to solve this problem, by acting as a peer to peer platform facilitating the sourcing of grinds based on location. It utilizes the swipe format that students are already familiar with to provide a seamless experience for the user. The current system offered by UCC, Grinds Centre, only provides the student with the general location of the tutor. Using TUTORTINDER the student can select a radius of up to five kilometers from their current location and the app will then provide them with a list of tutors meeting their requirements. It also provides an inbuilt chat messenger, so all of their communications are stored in one place.

USERSTORY

As a **student**,

I want **a system that allows me to**

connect with tutors in close proximity to me,

So that I may **connect with these tutors for grinds lessons.**

PROJECTPROGRESS

Once our group had successfully brainstormed project ideas and chosen the idea that would be the basis for our project, we began the development process.

At our third meeting we begun creating our paper prototypes. Each member of the group created their own paper prototype for the functionality we believed the application should have. We then presented our ideas to the rest of the group and pitched why we believed our solution was most suited for the application. When it came time to decide on which idea we would pursue, we struggled to come to a consensus as each team member believed their concept was the most suitable. We solved this issue by incorporating an aspect of each team members concept into the final idea. This allowed each team member to feel as if they had a contribution in the finalized application idea. We then created a story board that would be the basis for our application development. We also created a GitHub for our project to allow easy merging of every team members code at the end of the project

During week four, we met as a group and divided up the workload. Once every team member was assigned their tasks, we began the development of the application. We had weekly meetings from this point onwards. The weekly meetings allowed us to view each other's progress and discuss any necessary changes to the design of the application. The weekly 'Progress Update' meetings also functioned as a means to keep each team member accountable and on schedule.

By week six we had the login in page and the two user sign up pages completed. We completed these tasks quite quickly as we found them similar to a project we had completed in first year. We also explored the possibility of adapting the application for IOS so that we could complete a live demo. However, we decided against pursuing this due to time restraints.

During week six we ran into issues regarding the messaging form. In our original plan we had hoped to have a live chat function where the student and tutor could send messages in real time. However, this proved quite difficult to implement as it required the use of a significant amount of technologies and software. We believed this function was outside the scope of our abilities and so we changed this form to allow the student to send a message to the tutor's inbox. We found this email format easier to implement with regards to the time we had remaining before the due date.

At the midway point in the development process, we reflected as a group over the work we had already completed and made necessary changes. At this point we agreed that our Trello board was not properly maintained and could be improved on greatly. We assigned the job of Trello board management to one team member. Once we did this, we found Trello became an asset to the development of the application as the detailed documentation of the assigned tasks gave us a clear view of each members contribution to the application. One of the group members also created a comprehensive business plan for Java123 Technologies in accordance with the Startup School theme of the project.

Week eight was a breakthrough week for our group as we merged all the GitHub branches with the master branch. Once this was completed, we then focused on refining the project by making small edits to functionality and GUI design. We began the creation of the presentation pitch and the technical report this week also.

Final tests on the application and the database were completed during week ten. Each team member had the opportunity to test out the completed application. We each ran tests on the application, searching for bugs and small errors. When we were satisfied with the application, we then began writing our personal statements.

At week eleven we had our presentation and script finalized. We then decided as a group who would make the presentation. This posed an issue for us as the idea of presenting to the entire BIS2 class seemed quite daunting. Eventually two members volunteered to present and the rest of the team helped in coaching and perfecting the presentation.

During the final week of the project we put the finishes touches on our personal reports and began to put the project together. Everyone had the opportunity to read through the project documentation as well as an opportunity to do final tests on the application. Once we were satisfied with every aspect of the report, we submitted it for evaluation.

SOFTWAREARCHITECTURE

To best describe the software architecture process layers, a basic analogy can be used to simplify it. We can take each layer as a different entity in the story.

The first part of the story is the *data persistence layer*. This can be described as the food in kitchen, or the database,

The second part being the *data access layer* can be known as the chef. The data access layer gets the food from the kitchen, and then creates something with it.

This is then brought to the *business logic layer*, also known as the waiter. The waiter brings the data or "food" in a more presentable manner to the presentation layer.

The *presentation layer* is the final layer in the story. It is the guests at the table, waiting on the "food" or the data. The data has been brought from the data persistence layer, to the data access layer where it is accessed and changed. It is then given to the business logic layer, who changes it further to a more presentable form, and presents it to the presentation layer who is essentially the GUI. Just the same as the food in the kitchen is given to the chef, who makes a meal from it, who then hands it to the waiter who may add a few final touches before presenting it to the guest at the table.

Data Persistence Layer

- The data persistence layer can be described as a set of files which is used to communicate between the database and the application being created.
- To start with, we created a class, calling it "Tutor Tinder Controller (TTC)".
- We then loaded the database into the TTC.

Data Access Layer (Chef)

- The data access layer is the second step in the software architecture process. This layer is the layer that actually access the data.
- The purpose of the data access layer is to present the data in such a way that the business logic layer can both access and understand it, and in turn the data into the desired presented GUI design.
- It passes the data in the form of functions to the business logic layer from the database.

Business Logic Layer (Waiter)

- The business logic layer is a set of rules of how to retrieve the data from the database.
- Similarly, it sends information to the GUI, which will be displayed to the user of the application.
- The purpose of the business logic layer is to present and display the data in an organised way, which the presentation layer can easily interpret. It can be described as user specified.
- This layer retrieves data from the data access layer and presents to the presentation layer, and in turn, the GUI.

Presentation Layer

- The presentation layer is one of the most important layers of the software architecture.
- It can also be known as the Graphical User Interface (GUI).
- This is the layer that the user of the application will interact with.
- The sever takes the information from the business access layer to display on the user presentation layer.
- To create our presentation layer, we conducted a bit of research into how to create the perfect GUI.
- The first step we undertook was researching Jesse James Garrett's "Elements of User Experience". This book details how users experience different things due to the way your application or website is laid out through its architecture. The book divides the application into five different layers – the surface, skeleton, structure, scope and strategy layer. We learned about how each plane is interdependent, and how each layer individually affects the user. For example, the surface layer is what the user will see. It is how the buttons and texts are displayed and placed on the screen and will indicate how easily the user will be able to use the application.
- The second step we took was researching the android guidelines. This took us through both material design basics, and quality guidelines. We learned about things such as the style, patterns and animation of the app will affect users. Similarly, we learned about how different technologies such as iPad's, TV's and Laptops will view your app differently.

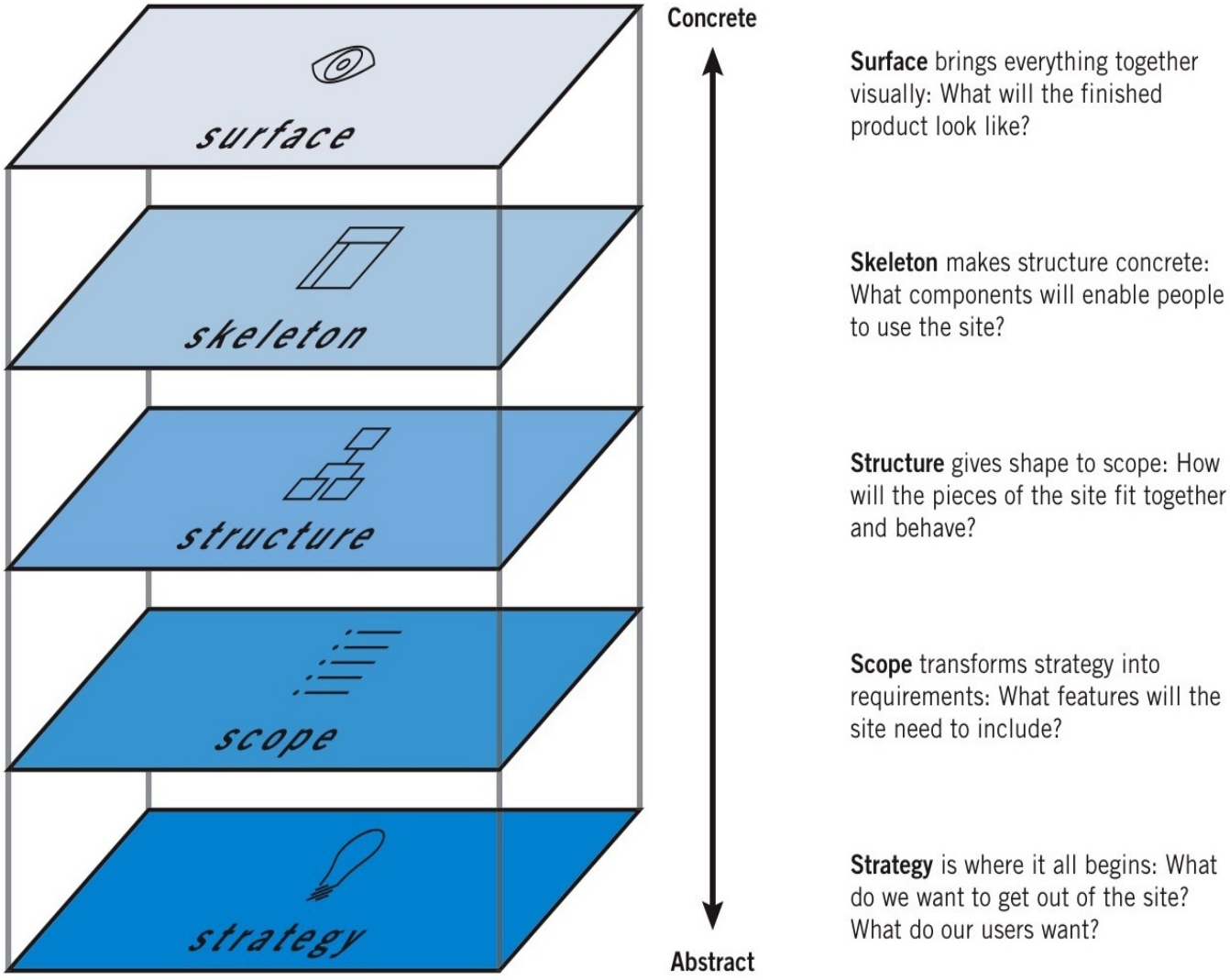
- Finally, we conducted more research on Don Normans 6 principles of interaction design. We partially read a book by Don Norman called "The design of everyday things". This book outlined for us the 6 main principles of design which are –

    1. Visibility (The more visible functions are the more likely users will be able to know what to do next)
    2. Constraints (determining ways of restricting the kind of user interaction that can take place at a given moment.)
    3. Feedback (sending back information about what action has been done and what has been accomplished, allowing the person to continue with the activity.)
    4. Mapping (relationship between controls and their effects in the world)
    5. Consistency (designing interfaces to have similar operations and use similar elements for achieving similar task)
    6. Affordance (attributes of objects that allow people to know how to use them)

- By researching these topics, we were able to design a GUI that would be easily accessible and understood by any user, of any age, with any kind of tech background.
- We decided to use a simple red logo, as we felt it would catch the users eye but not distract from the rest of the app.
- We used a consistent layout of buttons to allow the user the easily navigate their way across the app, and to make the process of using the app extremely easy to encourage user to use the app again.

Jesse James Garrett:

Elements of User Experience [1]



[1] https://www.quora.com/Did-Elements-of-User-Experience-by-Jesse-James-Garrett-specify-Visual-Design-a-part-of-UX-Design-or-as-a-separate-role (14:34,03/04/19)

RETROSPECTIVEREPORT

1. What would you do differently if this was week 1?

Although we have all agreed we are happy with the end product of this project, no assignment is without fault. As this was a massive project, with many group members to include and a large work load keep track of, we definitely feel as though we made a few mistakes along the way, however we also feel they were a learning curve for us all.

The first thing that we would do differently if this were week one would be we would aim to get organized and to establish all team roles and their respective responsibilities in the very first week. Similarly, we would target to have an rough draft / idea of what our project idea actually is, rather than leaving it to the second week like we did, to have a finalised project idea.

We would also aspire to all have an in depth understanding of the brief and of what our role is within the group.

Secondly, we would also aim to set up all developers with a GitHub branch of their own during the first week so that they can individually do their assigned job in the project and do it to the best of their abilities from anywhere that is convenient to them. The main reason we chose to use GitHub branches instead of a google drive account would be so the project would be both easily controlled and accessible. We also thought it would be convenient that we would be able to revert the project to its original state if we made a mistake or made an undesired change(s).

The third aspect of the project approach we took that we would change should it be week one again, would be that instead of using three Trello boards, we would aim to have a few more boards so as to be able to efficiently manage the project and update it accordingly for

all members of the group to contribute and be able to understand what stage the project is at, at all stages

Finally, if we were to redo this project, something small we would change would be communication and time management within the group. We would aim to establish a little better communication between group members , especially around the area of conflict as we feel we would have been able to dissolve conflict faster if we had better communication skills from the get go. Similarly, we all felt if we had better time management skills, so all team members could meet at specific times and not have conflicting work schedules or other project meetings at the same time this would have been beneficial to the overall outcome of the project.

2. **Have you thought of any scenarios that will cause your implementation to change?**

We would envisage the only way that limitations with google drive would be eradicated would be if google implement a developer section in google drive or version control the files. This would ultimately result in us potentially choosing to implement the google drive system rather than using GitHub. This would be the only scenario that we can imagine that might encourage us to change our project collaboration system. However, we are still insistent that GitHub was very efficient in this manner for this project anyway.

In addition to that, we feel since we used JavaFX which enables our application to run on a range of devices, there wouldn't be much that will cause our implementation to change.

3. **Did you learn anything unexpected or useful from implementing the project (technical or otherwise) that you would not have otherwise learned?**

We believe that we have learned a plethora of interesting and unexpected things throughout this project, varying from technical aspects to professional factors.

One such lesson would be that we had always found group projects especially group projects involving code is extremely hard to manage, as you constantly have to update all team members code to the current state of the code. This has to be completed before more coding or implementation of other members codes for their assigned job / task into the current state of the project can be uploaded.

For this reason, we concluded that branches would be the best way to overcome this problem as each member is reminded and notified when changes are made to the current state of the program by another member by way of merging conflict and they are also able to review the changes and depending on whether they want to override the current state of the program or just leave the program in the state it's up to them after review. Although we had all separately used GitHub before for our own personal coding projects, we had never used it for collaborative means. Despite our knowledge of Google drive, we chose to use GitHub to see what it would be like, and we were pleasantly surprised at its accessibility and convenience. We found this was a very worthwhile lesson to learn from this project as we had no clue of how to use GitHub for multiple users before this project and we can confidently say we have a better understanding of how it works and that it is much more easier to use and implement version control then multiple google drive file uploads and this will serve as a valuable resource to use in projects to come.

In contrast to that technical lesson, we learned how to manage our time a lot better as a result of undertaking this project. Our group had five members in it, which was a larger group number than any of us had ever worked with before. Initially, we found it hard to find a time that suited everyone to meet, and a place to meet that was convenient for everyone. We found that this semester we had four other group projects to complete in and around the same time as this deadline. This meant all our members had other commitments to other group projects, as well as social and personal commitments. Similarly, the workload for this project was bigger than any we had had before. There were many areas that needed to be completed, by different people at different stages. It was difficult at the start to see clearly which areas of the project we should start first, and which were the most important. However, through the use of prioritisation and accommodating attitudes, we were able to not only make it work, but also beat our initial deadline set for this project. As a result, we agreed that this project has definitely improved our time management and prioritisation skills.

A third unexpected lesson we learned from completing this project would be how different members of the group had different strengths which added varying degrees of value to the overall project. While we have had group projects before where certain members have had various strengths they could add to the project, we had never had a project this size with such a large scope of different jobs to be completed (from the actual coding and creation of the application, to the software architecture write up and class presentation). This enabled different team members to comfortably add value to the project, in a way that they felt would benefit the project most. While every member helped every aspect of the project in some way or another, members were able to showcase their strengths in areas they were confident in. We found it was interesting to learn from members new skills or tips, and watch how things were done differently to the way you would have approached or implemented a similar method.

We found it really interesting to see how the use of story boards, user stories and Scrum methods enabled us to keep our project on track, and our goals clear. Although we had learned about agile methods last year in a different module, we thought it was cool to see how the implementation of these methods enabled an efficient creation of our application. Similarly, it was interesting to see how our story board allowed us to keep on track with our roles and assigned tasks, but also how after a few new ideas, our story board changed slightly from our initial plans.

We thought it was intriguing to learn about how JavaFX made such a huge difference to our application. As we wanted our application to be run on various devices, we found the use of JavaFX to be vital for the successful creation of TUTORTINDER. This was a technical aspect we feel will definitely benefit us in third year on placement, and even into the working world.

Finally, learning about the different layers really allowed us to understand how the application worked. This was really unexpected as we had never thought about how the application worked in depth before. It was funny how we had assigned different members to complete different layers of the project such as the presentation layer and the data persistence layer, but not one member had taken a moment to realise how each of these layers connected to create the perfect application.