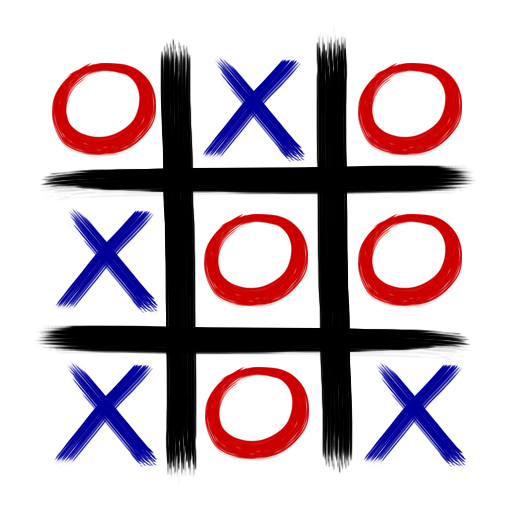
|  |
| --- |
|  |
| **Project Progress Meeting Transcript** |
|  |

UOE GROUP 2



|  |
| --- |
|  |
| PresentersIan J WolloffLukman MohamedSathira PadukkaPavendran WimalendranBabatunde Ahmed |

**University of Essex SEPM Course August 2022**

|  |
| --- |
|  |
| **The following is a transcript from the recent progress meeting held to show the current progress of the child’s game contract and to discuss the road to production.** |
|  |

## Introduction

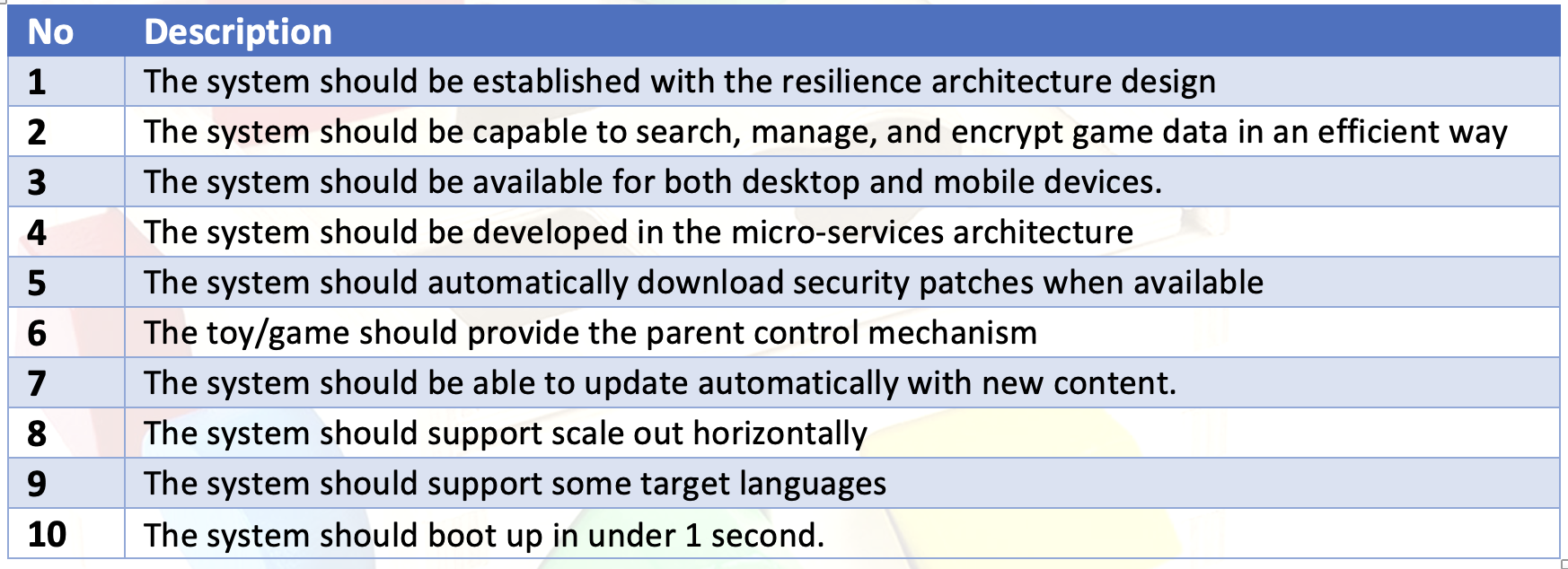


Hello we are happy to welcome you to the first scheduled update meeting of the child toy project that you commissioned us to scope for your organization. Our session today is going to be split into a number of sections where we will go over the discovery work that has been carried out and describe to you the path to production for the project. The areas we will look at today will be a mixture of project management, Proposed Technology and development practices along with a demo of a prototype application.

We hope that you will be happy with the work that has been done so far on the project

**Agenda**

Before we begin a little overview of the structure of today’s session, will first begin by introducing you to the team members who make up your delivery team and the roles they carried out through the design phase and what part of the solution they produced we will also briefly mention key design decisions that were made by the team in the delivery of the project. We will then provide you with an update on the progress that has been made on developing the game. The current state of the game any issues that have been encountered through the design phase. That will need to be addressed before the project can move forward. We will then follow this up by giving you an overview of the project process the stages in the project plan and how these have been executed the effort expended and how this lines up with the predicted cost model. We will then follow up with a demo of the prototype application highlighting the key areas from the requirements gathering phase and how the application has been designed to meet these requirements.



We will follow up the demo with a brief discussion of the application testing that has been employed throughout the application development cycle and how this testing ensures high quality code and finally we will discuss the road to production and how we can enable the Proof of Concept that has been designed to move from design into a fully-fledged production grade system.

**Introductions**

## I’m Ian and I have been acting as the project manager on this project your delivery team was also made up of two developers Lukman and Sathira who coded the prototype you will see in our demonstration they were backed up by our Quality assurance and technical writing team made up of Pavendran and Babatunde.

If we us start with the requirements gathering process. As you are know, we have met with members of your organisation to gather system and operational requirements for your product. Following our meetings and negotiations we were left with 10 requirements to be implemented into the solution that has been produced no hard requirements were discussed over the design of the game allowing us to come up with a concept.

**Design**

I would like to give you an update on the design of the solution. The design of our solution was based on the specifications you provided. We made the decision early on to handle the requirements provided in both code and in infrastructure as how the solution implements a requirement should be based on the best way to realize that requirement and sometimes the best way is to separate requirements that can be done via code and to do this at the infrastructure level.

To Facilitate a smooth development process we produced a **PID (Project Initiation Document)** and a **RAID log** to act as the Single source of truth during the project

The first requirement we designed was the ability to Dynamic Update the game code as this was a key requirement to the solution. As a development team we went through a number of possible solutions to this requirement that would allow you to update the code running for the game the one we finally settled on was one based on containers as the design is based on microservices, all the game code is contained within a container. we used docker as this the industry standard container engine though if required you could also use the podman or LXD container engines if you did not wish to use the docker engine. This is combined with a container from the docker registry called watchtower. This container will check the container registry where the code is stored either a public repository or authenticated private repository and if the container has been updated with new code, then this service will pull the new image and update the running container to the new version of the image. This allows you to always have up to date code running without also having to manually update the system to new versions. This also has the advantage of being able to schedule updates via the use of container labels.

This technology can be combined with your DevOps pipelines such as GitHub actions to provide a fully end to end process

1. New Code is created
2. Pipeline Creates new Docker Image
3. Watchtower Service updates running image to new version

The second requirement we addressed was the ability of the system to scale out for this we prototyped a couple of solutions

1. Round Robin Load Balancing
2. Round Robin Load Balancing with Fail Over
3. Docker Swarm
4. K8s Cluster

After looking at the solutions option Three and Four give the maximum scalability but with a large overhead in required knowledge required to successfully implement the solution. It is because of this we made the decision to implement option 3 as this gives you good scalability while maintaining the ability to be managed without needing to employ additional resources. If, however you do have technical resources then option three or four maybe a more elegant solution to the question of scaling your application to meet potential demand.

The next requirement we addressed was the ability of the application to be presented in a number of languages for this we prototyped two solutions one was to do Realtime translation using the Google Translate API so that game elements could be presented in a number of languages this worked well there were a couple of downsides however to this method that we needed to address. The first was as the service needed to call an external API this resulted in an unacceptable delay in the startup of the application missing the requirement for startup in under one second. There is also the possibility that if the game proved to be popular that the translation API would stop working due to the game exceeding the fair use rate limit that is imposed on the use of the service.

It was because of this that we looked for a locally managed solution and we found the flask babel library while not being able to do real time translation like a API as the elements of the interface do not change between play sessions, we can store a translation matrix so that users with differing primary languages can use the game this method combined with the python request library allows for automatic detection of the users language so that the interface can adjust without requiring user interaction. This also allows for the requirement to start the game in under one second to be met.

We next looked at the requirement to have the game work on both desktop and mobile devices as the game is designed to be delivered via a web interface this was done via a mixture of pure HTML CSS and JavaScript along with the use of a number of frameworks such as bootstrap to provide a web solution that is responsive to different display sizes and different platforms

The next area of design we looked at was the requirement to boot up in under one second as previously mentioned a number of design decisions were made to make this requirement possible first of these was to use a high-performance server to serve out the application as there was a requirement that the system be developed using the python language and due to use of the Flask framework, we made the decision to use Gunicorn as our WSGI platform.

**Progress (Milestones) & Costs**

### Those who plan do better than those who do not plan, even though they rarely stick to their plan.” –  Winston Churchill

We chose the waterfall model to complete this project. Business needs and requirements must be present at the start when using the waterfall methodology. This enables the analytic team to identify the business needs and requirements precisely, improving the output in terms of meeting the needs of the organization. When team members are spread out across the nation or the globe, this strategy can be highly productive. Additionally, a waterfall model requires fewer resources to deploy than alternative approaches.

It seemed logical to choose the waterfall model because the project's criteria were clear and unanimous, and the crew was distributed across several sites.

While sticking to the waterfall approach, we did split the development work into a number of sub development tasks

1. Development of user Logon and Registration
2. Development of Core Gameplay Mechanics
3. Development of REST API

The reason for this was to allow the development work to be done in parallel between the developers with each developer owning a part of the system. As the design was based on microservices this was also the most efficient way to work as each component would be its own microservice container. If the design was a monolithic design, then we may have gone down a different route for the development work required.

**Testing and Testing Methodology**

We employed a mixture of testing techniques while testing the application

* Manual Testing
* Automated Testing
* Hybrid Testing

The Manual testing, we did was done during the development phase where one of the developers creating the solution would test each function as it was created this was to ensure that the function or block of code successfully carried out its function within its design parameters. This however is only the first step in our testing and verification the second step of ensuring that the code quality of your application meets the required standards was by the use of automatic tooling. We used two separate tools for this bandit for to evaluate the attack surface of the code and then SonarQube was used to carry out a more detailed analysis of the code base the results of this were

**Bandit Results**

Code scanned:

Total lines of code: 139

Total lines skipped (#nosec): 0

Run metrics:

Total issues (by severity):

Undefined: 0

Low: 2

Medium: 1

High: 1

Total issues (by confidence):

Undefined: 0

Low: 0

Medium: 3

High: 1

Files skipped (0):

**Issues**

Above a certain complexity any solution may have bugs or undesirable execution pathways the following issues have been identified in the current codebase. The issues identified in the bandit scan were “Issue: [B104:hardcoded\_bind\_all\_interfaces] Possible binding to all interfaces.” This is not an issue as the code is deployed via a container.

**Roadblocks**

While development has not always been smooth some roadblocks that were encountered during the development of the solution were

* Requirements were quite loose well-defined requirements would have helped

**Path to production**

We have registered a Domain for your Game if however, you wish to host under your own domain / sub domain then this can be achieved with a simple change to DNS and update to the Traffik SSL Proxy that is sat Infront of the game docker containers.

**https://play.uoe-game.online/login.html**

While the current solution is a good example of what can be done the road to production for the solution requires a number of steps before the solution is ready for a full production

1. **Security / Monitoring**

The first of these is to strengthen the security position of the solution while hosting and infrastructure security is outside of our remit as developers, we would recommend putting in IDS monitoring and ensuring that there is full auditing of users and configuration events.

We also recommend a regular penetration test of the solution to ensure that the game and underlying supporting services and infrastructure are secure and not vulnerable to attack

1. **Resilience**

The current solution is hosted in a single datacenter before any full scale move to production it is recommended that the solution be distributed across multiple data centers to allow for unforeseen outages and DR with a more to production, we would also recommend a higher availability design on the infrastructure side to allow for failure of components without compromising the ability of users to access the game.

1. **Support and Training**

While we are happy to provide technical support for the application, we have created for your organization for longer term support we would recommend employing someone with the necessary technical skills to fully support the application once it has reached the deployment stage.

**User Documentation**

**Summary Time Estimation**

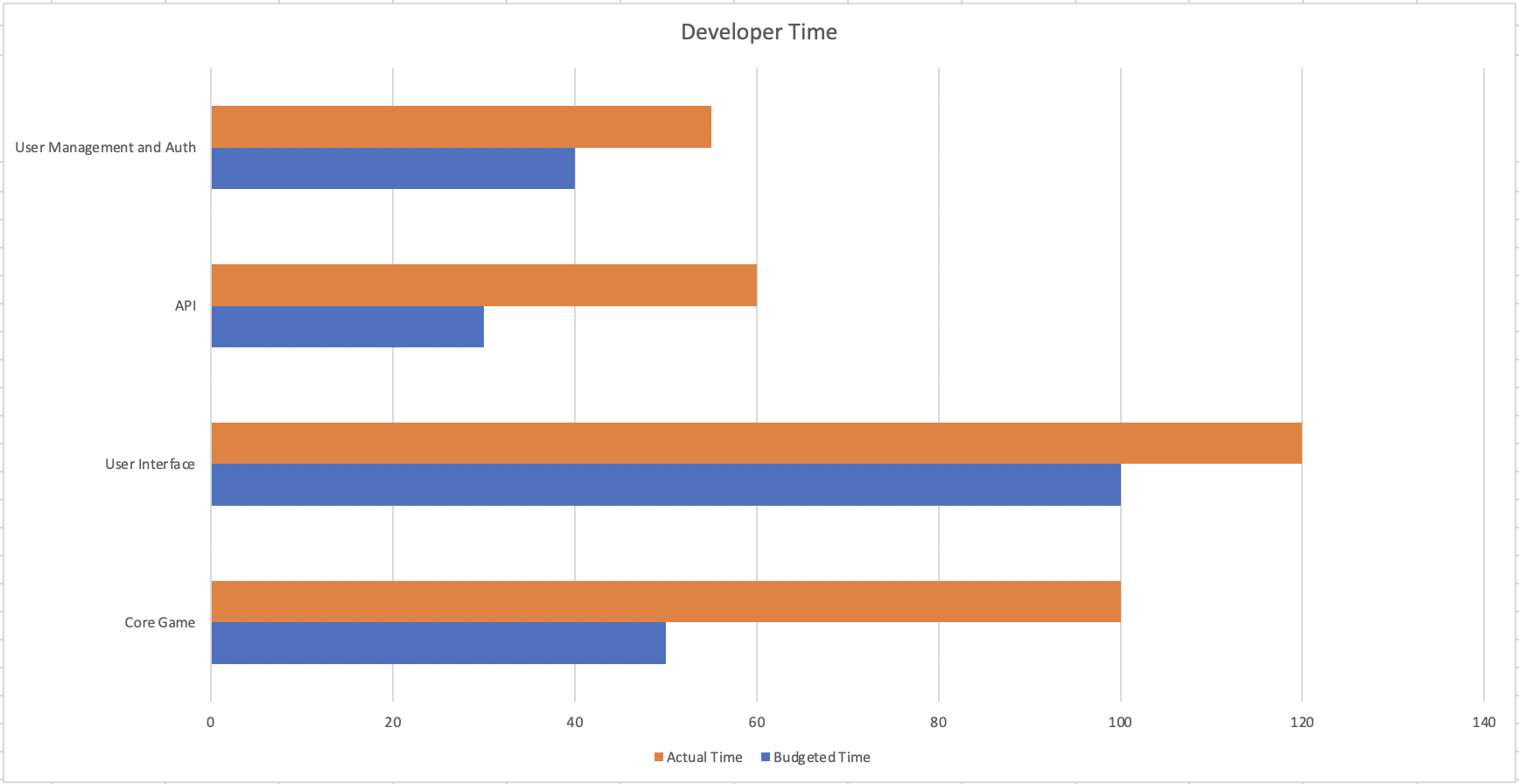
During our prototype development, we carried out a couple of methods for estimating the project complexity and length of the project the main we used was the COCOMO technique once this had been done and the project roles had been assigned within the team we could estimate the amount of time it would take to achieve each of the steps in our waterfall approach.

Having now created the prototype for this meeting this could have been better as the estimations were based on a typical project and your project had some unique challenges that may not be present in a typical project. Humans are quite bad at estimation so providing accurate estimates for the work to be carried out was always going to be a challenge. This proved to be the case

|  |  |  |
| --- | --- | --- |
| **Phase Area** | **Budgeted Time** | **Actual Time** |
| Core Game | 50 | 100 |
| User Interface | 100 | 120 |
| API | 30 | 60 |
| User Management and Auth | 40 | 55 |
| Total | 210 | 325 |

Or the actual time was 54.76% greater than predicted while this is a fairly large increase accurate predictions are a science or

There's no point in making predictions. It's not worth speculating because nothing is set in stone and things change - [Cristiano Ronaldo](https://www.brainyquote.com/authors/cristiano-ronaldo-quotes)

**

If we take a look at costs If to take the development time and convert them to an actual cost we first have to take the average python developer rate this can be seen from the following table

Average Python Developer Salary UK

|  |  |
| --- | --- |
| **Level** | **Salary Per Annum (£)** |
| Junior | 38000 |
| Middle | 57000 |
| Senior | 80000 |

UVIK. (n.d.).

To Get indictive hourly rate we will take the mean of the levels

38000 + 57000 + 80000 / 3 = 58333

From this we can calculate an average daily rate

58333 / 365 = Daily Rate of £159.81

If we then take this value and divide by 8.0 the standard working hours in day in a 40 Hour working week, we get an adjusted hourly rate of £19.97.

From this we can calculate the development cost of the solution is £6492.28 this is higher than predicted due to the time taken being longer then first predicted along with increased inflation pressures driving up developer pay meaning development now costs more than it did just a few weeks ago

**Project Risks**

During the project a number of risks were identified that either did or had the potential to affect the delivery of the project.

The first of these was the requirements for the project being quite open this meant that a fair amount of interpretation was required in that regard to what you would like in your solution I think we would have preferred a more ridged requirements that being said having creative freedom was welcome and we hope you like the solution that has been provided.

The second risk that we encountered was that due to the distributed nature of the development team it was not always as easy to communicate then if the team was centrally

The Third Risk we encountered was the members of the team are not experts therefore some of the choices and decisions made may not be the same as those made by a team with a wider knowledge base or real world this was most apparent in prediction of time as time predicted for a task to take place did not align with what actually happened with some more experience this could be overcome but for this initial phase this proved to be a challenge.

The last risk we faced was the hard requirement that the game had to be produced in python while this was achieved and the core experience was developed in python using the flask framework it is fair to say that this is not the first-choice language for all of the development team.

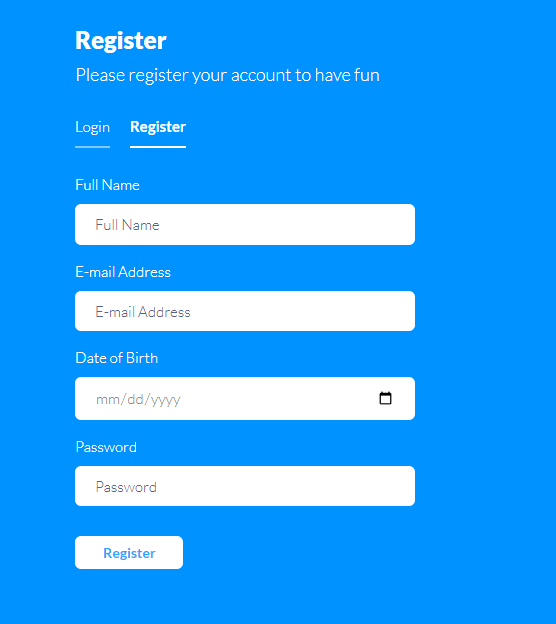
**Authentication**

In order to perform authentication and authorization we used firebase a service provided by google. Firebase allows us not to store user login information in our databases. Firebase is GDPR compliant. Register page and login page are all connected to the firebase service via API connection. Our application is containing data such as user emails and names but no credentials.

For future development we can also implement social media login and also it has detailed analytics.



A user must register and verify their identity at which point Firebase will generate a signed JSON Web Token which will contain users unique User UID for that user.



**References**

RAID Log | IST Project Management Office | University of Waterloo. (n.d.). Retrieved August 25, 2022, from https://uwaterloo.ca/ist-project-management-office/methodologies/project-management/planning/raid-log

PID - Project Initiation Document - purpose & contents. (n.d.). Retrieved August 25, 2022, from https://www.stakeholdermap.com/project-management/pid.html

Project Initiation Documentation :: PRINCE2® wiki. (n.d.). Retrieved August 25, 2022, from https://prince2.wiki/management-products/project-initiation-documentation/

Garcia, D. (n.d.). GamesCrafters: Tic-Tac-Toe. Gamescrafters.Berkeley.Edu. Retrieved August 25, 2022, from http://gamescrafters.berkeley.edu/games.php?game=tictactoe

Tic-Tac-Toe. (2002). Wolfram MathWorld. http://mathworld.wolfram.com/Tic-Tac-Toe.html

Game Theory and Tic Tac Toe | IB Maths Resources from Intermathematics}, url = {https://ibmathsresources.com/2013/11/24/game-theory-and-tic-tac-toe/}, }

File:Hello in different languages word cloud.jpeg - Wikimedia Commons. (n.d.). Retrieved August 25, 2022, from <https://commons.wikimedia.org/wiki/File:Hello_in_different_languages_word_cloud.jpeg>

Radhika D Amlani, 2012 . Advantages and Limitations of Different SDLC Models.

International Journal of Computer Applications & Information Technology Vol. I, Issue III,

November 2012.

Python Developer Salary in 2022 (Worldwide & The US) | UVIK. (n.d.). Retrieved August 27, 2022, from https://uvik.net/blog/all-about-python-developer-salary-cost-to-hire/