Presentation Transcript

GitHub Repository

Something-Fruity/Something-Fruity.github.io

Presentation Video

SEPM Presentation Final.mp4 - Google Drive

Introduction

Good day, everyone; we are incredibly thrilled to share with you the progress of the ongoing development of your exciting and engaging online game, Something Fruity.

Agenda

To kick-off, we will start with the agenda for this presentation.

- We will begin with a brief overview of the team executing your product.
- The project design includes the requirements gathering process, delivery roadmap, and milestones, which comprise an overview of the sprints and current sprint statuses.
- A demonstration of the application showcasing the game's functionality and the requirements for the demonstration.

- The testing methodologies used throughout the project to ensure that we are meeting the requirements throughout the development life cycle.
- And finally, the user documentation that we created for the game allows users to understand the game's features and functionality.

Team Overview

The project would not be possible without a dedicated team of professionals. We want to take a short opportunity to introduce the team members and give you an overview of the team structure.

Members

Firstly I will introduce myself, my name is Anrich, and I am the project manager on this project. The team was further comprised of two developers, namely Aidan and Lukasz, Technical Writers Tony and Richard, and lastly, the customer success team, namely David and Uzayr.

Project Design and Status

Requirements Gathering

Let us start the presentation with the requirements gathering process that took place. As you are aware, we have engaged with members of your organisation throughout the requirements gathering process. In our meeting on 26 November, you provided us with a list of fifteen requirements. Upon further negotiations in the subsequent

weeks, this list was reduced to ten requirements due to anticipated time constraints in meeting the project deadlines. At the meeting, you provided us with a carte blanche regarding the game's outcome and left us to come up with the concept.

The ten requirements that we have agreed on are seen on the following screen.
The five requirements that we have worked on for this presentation are:

- The system should run on L/W/IOS.
- Data must be stored in the most efficient way.
- A player should be able to create a user profile.
- Additional languages should be available as free downloadable packs.
- Sounds should be able to be muted from the UI with a single keypress.

We used GHERKIN syntax that uses unique keywords to describe a given requirement in English, followed up by using the Python library Behave to write a series of acceptance tests to meet each requirement (*Behavior Driven Development — behave 1.2.6 documentation*, no date).

As an example, we will walk through the first of the five requirements, creating a user profile. We have a features directory within the acceptance tests directory that outlines the requirement as a Behave feature. We start by describing the feature at the top of the file; in this case, create a user profile followed by the plain text user story used within Jira and is ultimately segmented into tasks within a sprint. Next, we have scenarios for the given requirement/feature; here, we can see that there are two scenarios, one for a new user with a unique username and another with an existing username. Let us take a look at scenario one. Given the user is on the main landing

page, When the user clicks register in the menu, she enters her details and clicks the 'register' button, Then she should be logged in and redirected to the account page. Now that we have a feature/requirement in plain English, we can move over to the steps directory, where we implement the test in python code and determine whether the requirement has passed. We will dive further into the testing methodologies used later in the presentation.

Delivery Roadmap and Project Milestones

Following on from the requirements gathering, we will now give you an overview of the delivery roadmap that provides a short demonstration of the JIRA project structure, followed by a summary of the planned sprints and the current project status.

Jira Demonstration

As you can see, we are now on the home page of the JIRA project management tool that we used for managing the project.

GitHub Code Integration

Before running through the project design specifics, I want to draw your attention to the GitHub code integration within JIRA. We have connected the Github repository where all of the code is hosted to JIRA using the integration, and we have done this for two primary purposes. This integration allows us to track the branches created when completing a specific user story, task or bug. Secondly, we receive additional

information regarding pull requests once a task is completed and merged into the main branch.

Epic Breakdown

I will now click on the backlog and begin showing you how we have designed and structured the project.

As a starting point, we began by defining a series of epics for the project; an epic represents a larger body of work that is then further broken down into a series of user stories and tasks. As an example, the user profiles epic represents the body of work that will ultimately fulfil the requirements about user profile creation and management, which are critical components of the application.

User Stories

When clicking on the user profiles epic, we filter for user stories, tasks, and bugs within the epic. As you can see, we have various items within the epic, and all items with the green icon depict user stories. As a group, we considered all of the requirements and then determined a series of user stories to fulfil those requirements and those stories ultimately make up the epic. The first on the list is SGN2-61 As a user, I want the ability to edit and update my details when I choose. The first user story forms a part of the requirement “A player should be able to create a user profile.” After the user story was created, this task was assigned to my colleague Aidan, one of our developers. When clicking on the user story, we can see that Aidan is indeed assigned and if we scroll further down, we see that Aidan created two commits and two pull requests to complete the story. Due to the integration with

GitHub, once a branch is created for that story, we can track these branches within the story details, as you can see. If we click on the pull requests, we can see further details about the addition made to the main branch.

An additional example worth drawing your attention to is how we dealt with bugs in the system. By clicking on the bug with the red icon, we can see that I raised a bug as we were experiencing an issue with the application. I described the bug and scrolling further down, and you will see that there is a branch created to solve the problem. If I click on the branch, we are taken to GitHub, where we can see the branch and then return to JIRA. We made four commits to solve the problem, and that bug was then resolved and merged into the main branch.

Sprints

As we created all user stories, they were ingested into various sprints. Removing the User Profiles epic filter, we can now see all of the user stories, tasks and bugs that belong to the current sprint called Development Environment Sprint number two. This leads to the next section, the overview of the planned sprints to complete the project.

Roadmap

I will now navigate to the roadmap section to see an overview of the project with the planned sprints. The roadmap section shows all of the epics we saw before with their beginning and end dates. We can see that the requirements gathering process began on the 20th of November and concluded on the 4th of December. We estimated the time scales that we would complete the epics, and we built sprints that were, on average, two weeks long within these timescales to complete the project tasks.

Overview of Planned Sprints

Looking at the roadmap, we also see an overview of the planned sprints, including completed sprints, current sprints, and future sprints. I will quickly give you an overview of the sprints within the project.

Requirements Sprint

As alluded to earlier in the presentation, the requirements sprint represented the process of gathering the requirements from yourselves. The sprint ran from the 20th of November and concluded on the 4th of December 2021. As mentioned, we have agreed on the ten requirements that you saw previously.

Development Environment Sprint 1

Next, we have the development environment sprint one; I will navigate to the reports page to see the burndown of the sprint along with the planned stories and tasks for the sprint. At the top of the report, we can see that the sprint began on the 3rd of December and ended on the 17th of December. Scrolling down, we can see a list of the stories and tasks that were completed during the sprint. The predominant focus of the sprint was to set up the development environment depicted in the user story “as a developer I want a repo to store my code” and set up the flask container environment depicted by the user story “as a developer, I want a basic flask application as a container for the game” along with a few other user stories that you can see on the screen.

Developments Environment Sprint 2

Then we have the development environment sprint number two, which started on the 17th of January and ended on the 30th of January 2022. The purpose of the sprint was to complete some of the critical requirements that were needed for this presentation. When scrolling down, we can see that the predominant functionality fulfilled requirements about the user profile, such as “as a user I want to be able to delete my account” and “as a user I want to be able to view my account details.”

Presentation Sprint

Navigating back to the roadmap, I will quickly highlight the presentation sprint.
The presentation sprint started on the 29th of January and was completed on Sunday the 13th of February 2022. The sprint was created to keep track of any tasks surrounding this presentation that you are watching or listening to right now.

Gameplay Sprint

Navigating to the backlog, I will take you through the gameplay sprint that forms part of the two remaining future sprints.
The gameplay sprint is likely the most crucial sprint of the application. We have scheduled time to complete this sprint from the 15th of February up to the 1st of March. The gameplay sprint contains user stories such as “as a user I want to be able to play against my friend,” this user story would fulfil the requirement “It should be possible to create a multi-player game.”

Documentation Sprint

The final sprint is the documentation sprint which forms part of the documentation epic. The sprint begins on the 2nd of March and ends on the 16th of March and the sprint aims to complete the product owner user story “as a product owner I want to access all system documentation” and there is an additional task to create deployment guidelines.

Application Demonstration and Code Organisation

Next, I will be handing over Aidan. He will be giving you a demonstration of the application, where he will highlight the five requirements selected for the presentation and how the functionality fulfils each requirement.

Demonstration

A player should be able to create a user profile

Thank you, Anrich; first off, we are going to demonstrate the requirement "A player should be able to create a user profile." When we navigate to the game's home page, the user is asked to log in. If I am a new user, who does not have a username and password, I can click on the register button at the top right-hand side of the screen. Then I am asked for a username, password, first name, last name, email address, and default language. Let me fill these out. While I am filling them out, I will draw your attention to the fact that we use field validation throughout the registration form.

All fields are required. As you can see, an error message is displayed when I fail to complete the form.

Let’s try again. Here you can see if I enter a fake email address, I receive a message indicating that I have entered an invalid email address, and I should start again.

Further backend validation includes checking the complexity of the password. If I enter a password that lacks complexity, I receive an error indicating that I have entered an invalid password and that the password must contain at least six characters. It is worth pointing out that the upper limit on passwords is 255 characters. This ensures that users can use complex passwords if they so wish and reduces the potential for hackers to be able to crack the password using brute force.

As you can see, our application complies with GDPR regulations by conspicuously displaying our privacy policy and by asking users to confirm that they have read and understood it. Now that I have entered valid data in the form, let me register an account by clicking on the register button.

Once registered, the user is taken to the account page. To edit the account details, we click on the edit account button. Once I have clicked the button, I can update my email address by changing the address and clicking on the update button; the changes will be updated on the database. Once again, the same verification is taking place in the background to ensure that the values entered are valid. As you can see here when I enter a non-valid email address, the changes are not persisted to the database, and I receive a notification informing me that this is so.

If I decide that I no longer want to play the game and wish to delete my account, I can click on the delete account button, after which I will be asked whether I am sure that I want to delete the account. Once I am sure, I will click the delete account button, and my account will be removed from the game, and I will be redirected to the home page. This is also in line with GDPR regulations which state that an individual must have access to, and be able to delete their details, from the data stored by an organisation.

Every account has several players associated with the account; in this case, these would most likely be the account holder’s children. As the game is aimed at young children, the parents can create the player accounts for their children. There are two scenarios, firstly we can use the player by clicking on the use player button, and the player will be taken to the gameplay page. Secondly, the parent can add a new player on the previous screen by clicking on the add new player button, then entering the name of the player and clicking on the add player button.

A player should be able to create a persona

Next, I will demonstrate the following requirement: "A player should be able to create a persona". While we are on the account page, we will click on the user button to take us to the gameplay page. Once we are on the gameplay page, we can select a persona that the user has created.

Additional languages should be available as free downloadable packs.

Now, I will demonstrate the requirement "Additional language packs should be available as free downloadable packs." First let me log in as an existing user. As our customer success representatives, David and Uzayr, previously agreed with you, for this the first implementation, we will only be demoing the ability of language switching in English and French.

As we are using the Flask framework to deliver the application, we have the flexibility of allowing the user to switch between various translations of the game within the game rather than requiring the user to download a language pack. As you have seen earlier in the presentation when users register their account for the first time, they can select their default language, and the selection is stored in the database. As an example of language switching, we have included a button on the navigation bar allowing the user to switch between English and French; when on the account page, the text on the page changes to the translated version.

All of the text that is displayed on the website has been annotated to make it discoverable by a widely used localisation tool called Babel. The benefits of using this tool are that it extracts all of the translatable text from the application and stores it in a .pot file. Additional languages can be added by simply providing a translation in the desired language for each of the terms in the file. Here I am showing a screenshot of the French translation file, so you can see how straightforward it is to add further languages. Here are each of the terms from the site in English, and beneath in their French translation. Thank you.

The system should run on L/W/IOS

Next, I will note that the requirement "The system should run on Linux, Windows and IOS" is fulfilled without porting the application to other operating systems as the game is hosted in the cloud and is accessible through any modern web browser.
We do have instructions in the codebase on how to get the application up and running in either Windows, IOS or Linux and the development team have been working and launching the application from various both iOS and Linux machines.
On the web application interface itself, we have carried out extensive testing on the most popular modern browsers to ensure that the web interface remains bug-free regardless of which browser the user is using to access the site. As you can see here in these tabs, I have the application open in Chrome, Firefox and also Safari.
This ensures your users will have a great experience, accessing the site from any operating system, using their preferred browser.

Data Must be Stored in the most efficient way

The requirement "Data must be stored in the most efficient way" is fulfilled as we have a MySQL database that securely stores the user's details. We have used a widely known Python library, SQLAlchemy, to implement an object relational model in our code. This minimises the data manipulation in the code, and effectively allows the database records to be stored directly as objects in the code, with no JSON or String manipulation necessary, while also avoiding multiple database callouts from the code which can be time consuming and make the application run less smoothly.
Most importantly, we have use the werkzeug library to securely hash and salt the

user's password, which makes it impossible for a password to be compromised even if the database were accessed maliciously. Additionally database normalisation and best practises were used to ensure that database performance was as optimised as possible.

Sounds should be able to be muted from the UI with a single keypress.

Lastly the requirement “Sounds should be able to be muted from within the UI with a single keypress” is fulfilled as we have provided a mute button on the navigation bar and when clicking on the sound button the sound is muted or unmuted depending on the state of the button. At the moment, we have put a sample game soundtrack. This, of course, can be updated when you have chosen the music you’d like to use as a theme. When the game is hooked up to the application, the button will also mute and unmute gameplay sounds.

Testing Methodology

We employed a dual testing approach incorporating manual and automated testing (Sawant, Bari and Chawan, 2012; Sánchez-Gordón, Rijal and Colomo-Palacios, 2020). Our manual testing included technical code review by multiple developers and a walk-through with other team members (Itkonen, Mäntylä and Lassenius, 2009). Linters were used to reduce errors and code smells (Theunissen, Hoppenbrouwers and Overbeek, 2021). Automated testing included scripts for correctness, reliability, performance and security (Sawant, Bari and Chawan, 2012). We used the "Bandit" open-source security scanner to ensure security (Konoor, Marathu and Reddy, 2017). No medium or high-security issues were detected.

There are two types of test, tests which verify the code acts as plans, and tests which validate the code, by checking that it meets certain requirements.

We wrote a suite of unit tests which can be run by the development team executing a command in the terminal. Using ‘Coverage’, a Python module which measures how much of your code the tests exercise, we are happy to say that our 47 unit tests offer 87% code coverage. Of course, as we continue to develop the application, we will also continue to add more unit tests, and expect that our coverage will always be above 85%.

The next step in this testing system is user acceptance testing, which includes testing by the quality control team and the customer to ensure quality control, quality assurance and satisfying the requirements proposed by the customer (Suman and Sahibuddin, 2019). The acceptance tests were written in Gherkin, a human-readable language which allows non-technical team members to write acceptance tests. They are executed in Python using the behave framework, and here you can see an example of one of the tests. Our customer success team brought these tests to you a fortnight ago to be signed off, and ensure that the requirements we were testing for, and the ones you wanted.

Manual Testing

We have decided to conduct a functional blackbox testing along the way in order to test the usability of the software. The main focus was to manually check the inputs and outputs of the AUT (the Application Under Test).

The other focus in this testing is to check whether the end-user can understand and operate the application easily or not: we do this without checking the internal code structure and the details of the implementation.
The tester should have the perspective of an end user. We need to check also if all the features are working as mentioned in the requirement document. In this process, testers execute the test cases and generate the reports manually without using any automation tools. (Rajkumar, 2021) I am going to show you now an example of one of the reports of the manual testing.

Documentation

User Documentation

We have gone to great lengths to develop user documentation for your game. We are currently on the GitHub page for your application, and the documentation can be found by clicking on the GitHub pages URL found below the repository description. We can see a brief introduction to the game on the documentation page. Next, we have the minimum hardware requirements for users that wish to play the game. We then have a section outlining the account creation process, logging in to the game, profile and account management dashboard and more. Most importantly the user documentation contains language and privacy policy documentation inspired by the GDPR directive to ensure that we are compliant with the directive and provide a means for all users to protect and control their privacy.

Budget and Summary

During our development stage, we identified three main methods for estimating the project timing: Algorithmic- involving COCOMO or functional point analysis, non-algorithmic- involving expert judgement, price to win, estimation (top down and bottom up) and learning oriented methods- Artificial neural networks, Bayesian models and regression (Chirra and Reza, 2019).

Due to the size of the project being comparatively small, algorithmic and learning oriented methods were not appropriate. We therefore chose to utilise a non-algorithmic model- expert judgement, which is also the most commonly used method in time estimation today (Aizaz *et al.*, 2022).

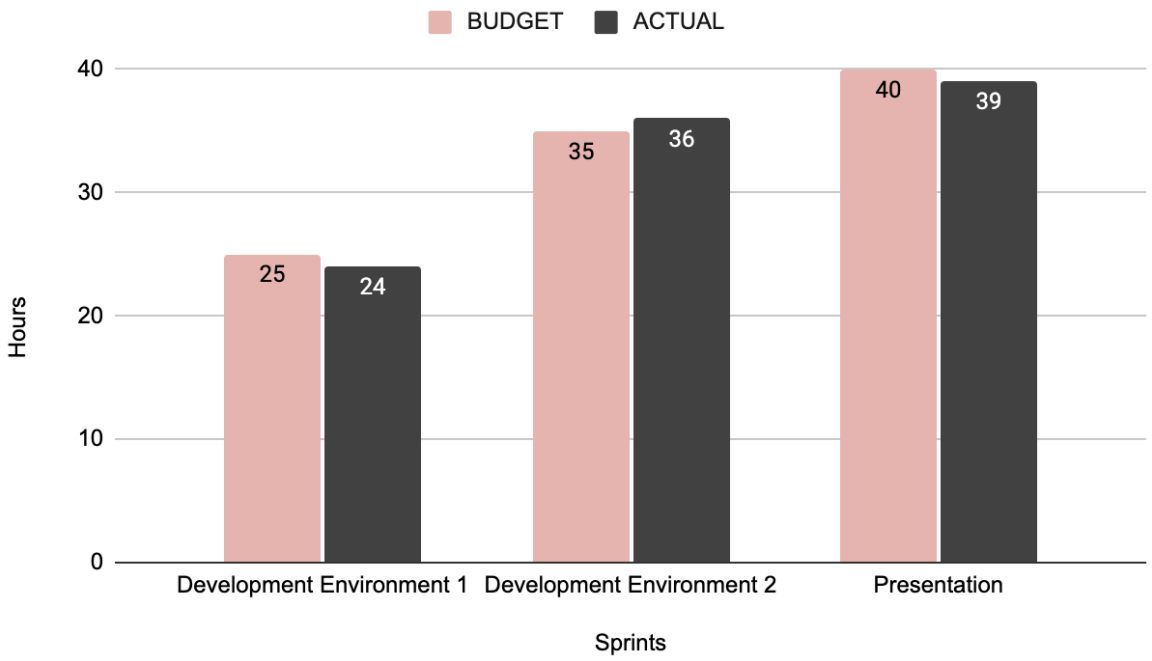
After outlining the different roles, we decided on the different phases of our agile development process. We then each estimated the amount of time it would take to achieve each of our tasks. Once this was done, we compared the estimates with other members of the group, and deliberated until a consensus was reached. The first table indicates our estimated time in terms of man-hours to complete the project.

Sprints	**BUDGETED HOURS**			
	Developer	**Quality control**	**Project lead**	**Total**
Development Environment 1	10	10	5	25
Development Environment 2	20	10	5	35
Presentation	10	10	20	40
Gameplay	50	30	30	110
Documentation	10	20	10	40
Total	**100**	**80**	**70**	**250**

With regards to risk analysis, we found that the following risks posed the greatest threat to a successful project:

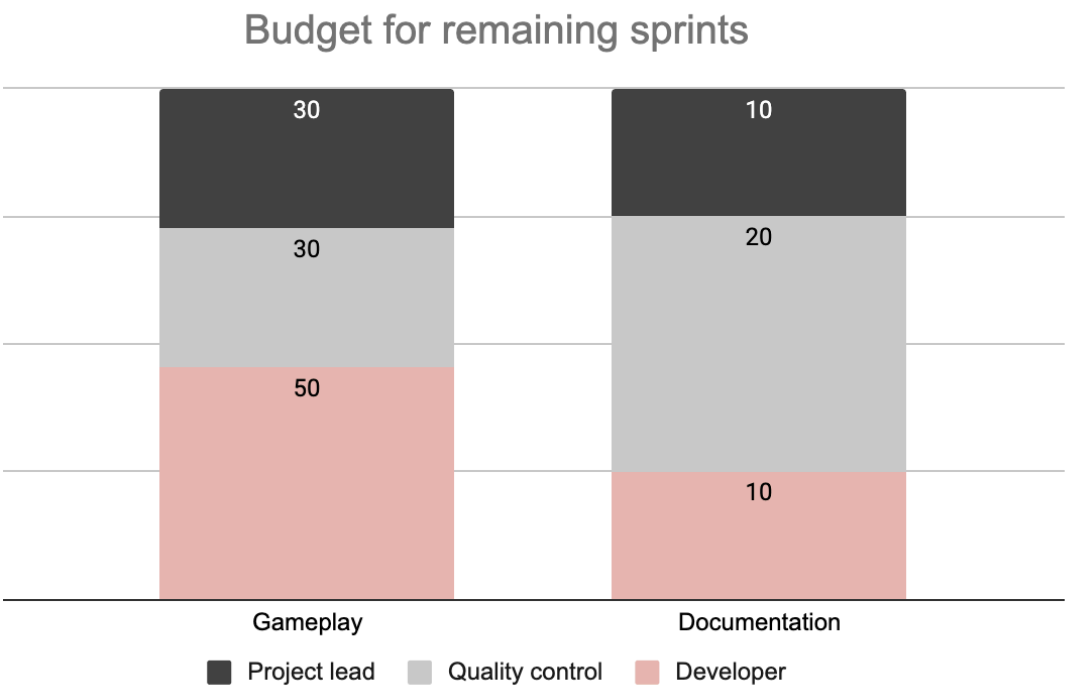
- Geographic and temporal differences meant that we were rarely able to conduct full team meetings, which may impact communication and therefore successful development (Malik *et al.*, 2018).

- Secondly, this is a university assignment, none of the project members are “experts” as yet, and therefore our time estimation may be considered highly subjective. However, as we are at a similar level of education and experience, we found this time estimation to be a worthy exercise (Tam *et al.*, 2020), as noted by comparing actual hours.



We logged our actual hours to ensure that we remained on track with the project plan. Also, to anticipate whether we need to make any adjustments to the budget.

We have noted that the budgeted hours are in line with actual hours spent on the tasks.



In order to appropriately budget for the remaining two sprints, we have applied the same logic to our estimations. For example, for the project lead, we have estimated 30 hours for gameplay and 10 hours for documentation.

We greatly appreciate your time today, and we look forward to the continuous development of your game and application, something Fruity.

References

Aizaz, F. *et al.* (2022) ‘An Empirical Investigation on Software Cost Estimation Techniques and Barriers on Agile Software Development in Software Industry of Pakistan’, pp. 194–199. doi: 10.1109/fit53504.2021.00044.

Behavior Driven development — behave 1.2.6 documentation (no date). Available at: https://behave.readthedocs.io/en/stable/philosophy.html#the-gherkin-language (Accessed: 7 December 2021).

Chirra, S. M. R. and Reza, H. (2019) ‘A Survey on Software Cost Estimation Techniques’, *Journal of Software Engineering and Applications*, 12(06), pp. 226–248. doi: 10.4236/jsea.2019.126014.

Härlin, M. (2016) ‘Institutionen för datavetenskap Testing and Gherkin in agile projects’. Available at: http://www.diva-portal.org/smash/get/diva2:908749/FULLTEXT01.pdf.

Itkonen, J., Mäntylä, M. V. and Lassenius, C. (2009) ‘How do testers do it? An exploratory study on manual testing practices', *2009 3rd International Symposium on Empirical Software Engineering and Measurement, ESEM 2009*. IEEE, pp. 494–497. doi: 10.1109/ESEM.2009.5314240.

Konoor, D. K., Marathu, R. and Reddy, P. (2017) 'Secure OpenStack Cloud with Bandit', *Proceedings - 2016 IEEE International Conference on Cloud Computing in Emerging Markets, CCEM 2016*. IEEE, pp. 178–181. doi: 10.1109/CCEM.2016.044.

Malik, B. H. *et al.* (2018) 'Geographical distance and communication challenges in global software development: A review', *International Journal of Advanced Computer Science and Applications*, 9(5), pp. 406–414. doi: 10.14569/IJACSA.2018.090553.

Ortu, M. *et al.* (2015) 'Measuring and understanding the effectiveness of JIRA developers communities', *International Workshop on Emerging Trends in Software Metrics, WETSoM*. IEEE, 2015-August, pp. 3–10. doi: 10.1109/WETSoM.2015.10.

Özkan, D. and Mishra, A. (2019) 'Agile Project Management Tools: A Brief Comprative View', *Cybernetics and Information Technologies*, 19(4), pp. 17–25. doi: 10.2478/cait-2019-0033.

Sánchez-Gordón, M., Rijal, L. and Colomo-Palacios, R. (2020) 'Beyond Technical Skills in Software Testing: Automated versus Manual Testing', *Proceedings - 2020 IEEE/ACM 42nd International Conference on Software Engineering Workshops, ICSEW 2020*, pp. 161–164. doi: 10.1145/3387940.3392238.

Sawant, A. A., Bari, P. H. and Chawan, P. . (2012) 'Software Testing Techniques and Strategies', *Journal of Engineering Research & Applications*, 2(3), pp. 980–986.

Suman, R. and Sahibuddin, S. (2019) 'User acceptance testing in mobile health applications: An overview and the Challenges', *ACM International Conference Proceeding Series*, Part F148384, pp. 145–149. doi: 10.1145/3322645.3322670.

Taba, S. E. S. *et al.* (2017) 'An exploratory study on the usage of common interface elements in android applications', *Journal of Systems and Software*, 131, pp. 491–504. doi: 10.1016/j.jss.2016.07.010.

Tam, C. *et al.* (2020) ‘The factors influencing the success of on-going agile software development projects’, *International Journal of Project Management*. Elsevier Ltd, 38(3), pp. 165–176. doi: 10.1016/j.ijproman.2020.02.001.

Theunissen, T., Hoppenbrouwers, S. and Overbeek, S. (2021) ‘In Continuous Software Development, Tools Are the Message for Documentation’, 2(Iceis), pp. 153–164. doi: 10.5220/0010367901530164.

Rajkumar SM (2021). *Automation Testing Vs Manual Testing | Important Differences You Must Know*. [online] Software Testing Material. Available at: https://www.softwaretestingmaterial.com/automation-testing-vs-manual-testing/#When-to-use-Manual-Testing [Accessed 13 Feb. 2022].