

Gamified Application Development Resource Site

Yonatan H. Fessehaye



Thesis submitted for the degree of
Master in Informatics: Programming and System Architecture
60 credits

Department of Informatics
The Faculty of Mathematics and Natural Sciences long

UNIVERSITY OF OSLO

Spring 2022

Gamified Application Development Resource Site

Yonatan H. Fessehaye

© 2022 Yonatan H. Fessehaye

Gamified Application Development Resource Site

<http://www.duo.uio.no/>

Printed: Representralen, University of Oslo

Abstract

Open-source testing platforms promote innovations and collaborations and solve real-world problems, yet it is common to encounter incomplete and complex understanding documentation. This thesis paper explores one of the many ways to minimise such confusions and incompletions, i.e. by exploring a documentation guideline with a step-by-step approach in a gamified manner with an engaging and collaborative process to perform a task.

DHIS2 has one of the many scalable open-source platforms; it is leveraged commonly to manage health information related uses-cases in many resource-constrained environments worldwide. Moreover, it provides a collaborative environment for building web applications. Nevertheless, as with all open-source platforms, there is a need for documentation which is easy to use for promoting a feasible development environment of the web- applications.

Design Science Research approach is used to design, analyse and research the implications of the artifact on how: (1) the artifact is built, (2) the artifact is reviewed by users and experts, and (3) data analysis to improve the artifact iteratively.

Keywords — *Open-source platform; web-applications; guidelines; gamification; DHIS2*



ACKNOWLEDGMENTS

This thesis work, the artifact, and the research could not have been possible without the exceptional support of my supervisor, Dr. Johan Ivar Sæbø. In addition to his comments, recommendations, and knowledge of the DHIS2 platform, his honesty and detailed feedback were motivating to keep me working on my thesis project. In the process, he has become my inspiration to complete the thesis work. I will always be grateful for his consistency, honesty, and availability, starting from the early meetings while discussing the project ideations and possible scenarios until the last minute of delivery.

Much gratitude to the members of the DHIS2 Design Lab and Magnus Li for the informative meetings, workshops, and peer review sessions. The resources from Magnus Li and the team were detailed and effective in researching the artifact. I cannot thank the DHIS2 Design Lab team enough; everything in this thesis work has contributions from the entire team, starting from the initial discussions when we (students) were picking thesis projects and to the peer reviews.

It was also an honor to inquire questions regarding the artifact and receive such excellent feedback from the DHIS2 core team; their feedbacks were clear and detailed enough to tackle the challenges with the artifact. Furthermore, the feedbacks were an inspiration to push the artifact's depth and breadth by considering a multitude of directions and set-of users both on usability and efficiency for use.

Lastly, I am grateful for the comments from the DHIS2 community, the beautiful discussions with several individuals, the critics who gave critical comments, the inputs from all the individuals who contributed to the artifact, and family, friends, and team Stingray.

Yonatan H. Fessehaye
University of Oslo
June 2022



ACRONYMS

OS Open Source

DX Developer Experience

PWA Progressive Web-Application

API Application Programming Interface

DHIS District Health Information Software

DHIS2 District Health Information Software 2

DEx Developer Experience



LIST OF FIGURES

- 2.1 App developers, platform owners and end-users. (Figure 2.8, Tiwana 2014) 7
- 2.2 Ecosystem architecture. 8
- 2.3 Developer Experience Framework, (Figure 1, Fagerholm and Münch 2012) 9
- 2.4 Open-source Game building framework (cocos2d-x) 13
- 2.5 Game and entertainment industry by market capitalisation 14

- 3.1 DHIS2 Homepage 15
- 3.2 DHIS2 Developer Portal 16

- 4.1 Design Science Research checklist of evaluation. (A. Hevner and Chatterjee 2010) 19
- 4.2 Focus groups in DSR, (fig 10.2, A. Hevner and Chatterjee 2010) 21

- 5.1 Orignal repository and configuration steps 23
- 5.2 Task: html-js-css level 1 25
- 5.3 Task: React Js level 1 26
- 5.4 DHIS2 Application level 1 27
- 5.5 DHIS2 Application level 2 28
- 5.6 Gamification by design: Implementing game mechanics in web and mobile apps. (Zichermann and Cunningham 2011) 29
- 5.7 Commit history of completed tasks 30
- 5.8 Types of awards (badges) given 30
- 5.9 Artifact architecture diagram 31
- 5.10 Activity Diagram 31

- 6.1 Artifact iteration 1 37
- 6.2 Six components of an information system design theory (A. Hevner and Chatterjee 2010) . . . 38
- 6.3 Artifact forks 40



CONTENTS

- 1 Introduction** **4**
- 1.1 Motivation 5
- 1.2 Research Question 5
- 2 Related Literature** **6**
- 2.1 Information Systems 6
- 2.2 Digital Platforms 6
 - 2.2.1 Open-source 7
 - 2.2.2 Architectures of Platforms 8
- 2.3 Web-Applications (Webapps) 10
 - 2.3.1 Web-Application Development Guidelines 10
 - 2.3.2 Architectures of web applications 11
- 2.4 Gamification 12
 - 2.4.1 Theoretical Underpinnings of gamification 12
 - 2.4.2 Gamification: pros & cons 12
 - 2.4.3 Gamification Design approaches 13
- 3 Background** **15**
- 3.1 DHIS2 15
 - 3.1.1 DHIS2 and Application Development 17
- 4 Methodology** **18**
- 4.0.1 Research process 18
- 4.0.2 Design Science Research 19
- 4.0.3 Data collection 21
- 5 Artifact description and Evaluation** **22**
- 5.0.1 How the artifact works 22
- 5.0.2 Artifact design 29
- 5.0.3 Evaluation of artifact architecture 30
- 5.0.4 Technologies used 34
- 6 Results and Analysis** **36**
- 6.1 Artifact iterations 36
 - 6.1.1 Iteration I: 36
 - 6.1.2 Iteration II 37
 - 6.1.3 Iteration III 38
- 6.2 Technical challenges 38
 - 6.2.1 Artefact design: 38

6.2.2	Challenges and shortages of the research process	40
7	Discussions	42
7.1	Findings	43
7.1.1	Step-by-step guidelines	43
7.1.2	Gamification for motivation to follow guidelines	44
7.1.3	Limitations of the study	45
8	Conclusion	46
8.1	Future works	47
8.1.1	Gamification frameworks (platforms)	47
8.1.2	Exploring design patterns inside step-by-step guidelines	47
8.1.3	Gamifying bottom-up	47
8.1.4	Glimpse into the Metaverse	47

INTRODUCTION

A web-application (web-app) is a software application that can be accessed over the internet (or computer networks) and executed on web browsers (Chrome, Opera, and others) to provide functionalities. Web-based applications have been getting tremendous interest among software designers and engineers to develop responsive and mobile-friendly applications (Shahzad 2017); moreover, with the advancement in hardware and software services, the interest from every industry continues to increase.

Modern Webapps are generally built with many programming languages, frameworks, design tools, and engineering approaches. Moreover, software product owners, stakeholders, developers, end-users (customers), and related conventional architectures and standards play essential roles in producing usable Webapps.

Similarly, the challenges of building Webapps increase with the increase in size and complexity of the Webapps, significantly if the web-app consumes resources from software platforms, cloud infrastructures, or server-less computing resources.

DHIS2 as a software platform provides an environment to build Webapps that are customizable, reusable, modular web components and open APIs that are designed for resource-challenged environments. The platform's use-cases are mainly in the health sector and are used by more than 70 low-middle income countries. In addition, many complementary products are continuously built by consuming resources from the platform.

Having a software platform with clear documentation guidelines about the building process and the Webapps' architecture (integral parts of Webapps) to follow promotes optimal use of resources for building those Webapps. This is even more visible in open-source software platforms, where the complexity of building can be reduced by having well-organized documentation. Furthermore, documentation that is illustrated through several example applications and practical exercises may prevent developers from going side-ways beyond the functionality that is presented therein (Kirk, Roper and Wood 2007), which shows having well-organized documentation can be attained without all exemplary definitions of every function definitions, but by addressing the critical and challenging areas.

Thus, by considering the software platform (DHIS2), developers' and web-app development practices, this thesis work explores an application-building resource-site, which is gamified with a step-by-step guideline documentation approach (Procida 2020). Users of the resource-site use the steps to achieve certain milestones and earn badges upon completion of the milestones. The evaluation of the artifact explores the validity of gamification principles and step-by-step guidelines for a resource site.

1.1 MOTIVATION

Programming can be viewed as a scientific discovery for designing programming environments and training methods, which can be difficult for new programmers (Kim and Lerch 1997). It can be safely stated that programming and creating a web application are still tricky, requiring a broad range of skills and experience (Rode 2004); therefore, many individuals, companies, and online-educational platforms are continuously exploring to improve the learning challenges with different approaches.

DHIS2 is the most effective health management information system in the world, providing various services to several technological solutions. When combined with non-routine data such as household surveys, support population-level decision-making, and data from all sources, health service data enhance a country's ability to monitor, detect and respond appropriately to public health emergencies (Adu-Gyamfi, Nielsen and Sæbø 2019). In this context, such technological solutions differ in their stabilities and usability based on how well they consume the functionalities provided by the services providers, DHIS2. So, this begs the question of having a standardized way of building those technological solutions.

DHIS2 community of developers, along with the developer resources, training, and documentation provided by DHIS2, helps standardize the building processes of several technological solutions like the web-applications. However, Webapps in different use-cases and implementations by the local developers do not usually follow the same building processes (e.g., programming languages and frameworks) which is laid out by DHIS2, and this thesis work looks to explore and find a solution to this common problem with innovative and flexible open-source platforms.

After the initial investigation of online services that teach programming focusing on the web-applications include Codecademy, Codeschool, and Treehouse, by which rewards and badges play essential roles to the developers of the web-applications.

Moreover, games like WarriorJS, Checkio, Codecombat, CodinGame, JS Robot, Screeps, JS Dares, and the book Learn To Code by Playing Games (by Harsh Makadia) are some of the fastly growing environments to learn web-application programming while playing games. They all have several types of incentive mechanisms for learners, and such incentive principles of rewards and badges come from the principles of gamification principles, which are backed by research that finds that gamification does improve the learning experiences of users by abstracting complex concepts while increasing motivation among users and keeping their goal-oriented to achieve a milestone.

1.2 RESEARCH QUESTION

The consideration for starting the research question is to try to explore an approach for incorporating guidelines approach of documentation with gamification principles within the context of application development resource site:

- *How do aspects of gamification and documentation guidelines pertain to the building of web-applications in a software development platform?*

Here, a detailed review of guidelines, gamification, and feasibility is seen along with a web-application building. Finally, a repository is created on Github to explore both the social and technical implications of guidelines and gamification. Results of the artifacts' evaluations and surveys (data collected) from participants give insights into the aspects of guidelines, gamification for a web-application building resource, and further exploration into the app-building resource-site.

RELATED LITERATURE

This chapter presents an overview of the literature. It explores the research that is relevant to my research which has relevance (implications) to the design of the artifact and the findings of the research in general. It has four main sections where the literature around information systems, digital platforms, gamification, and web-applications.

2.1 INFORMATION SYSTEMS

According to Lee (Lee 2004), an information system is not the information technology alone but the system that emerges from the mutually transformational interactions between the information technology and the organization. An organization in this context can be assumed as human (variables politics and socio-technical aspects) and the dynamics of its complexities within the information systems, and Lee also tries to see this from social theory. The information technology aspects, in turn, play roles in the complexity of information systems by providing technical services to the organizations.

2.2 DIGITAL PLATFORMS

A software platform is a software-based product or service that serves as a foundation on which outside parties can build complementary products or services (Tiwana 2014). Based on their purposes, research foundations, material properties, and value creation, there are two categories of platforms: transactional and innovation platforms (Bonina et al., 2021).

Thus, by focusing on the properties of innovation platforms, the following are discussed: architectures, complementary products, or use-cases used by users (end-users) are considered. Innovation platforms thus have the different digital platforms share three standard sides, viz. *App Developers*, *Platform Owner* and the *End-Users*.

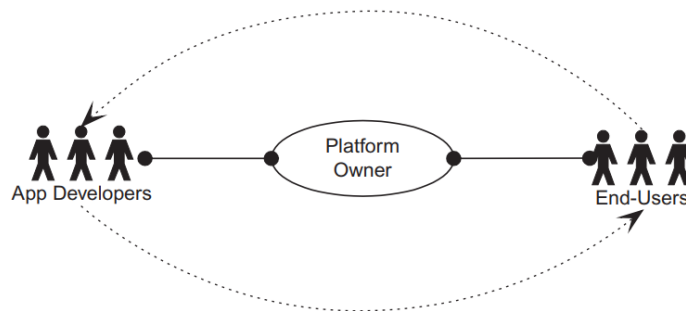


Figure 2.1: App developers, platform owners and end-users. (Figure 2.8, Tiwana 2014)

The application developers build the applications by consuming the resources of the platform-owner. Such applications then provide services intended for the end-users. The platform owner manages the direction, and general goals of the platform, moreover platform owner, can enforce related standards, guidelines, and protocols. In the general digital platform, owners play vital roles in directing the entire ecosystem of the platform since governance of the platform is directly related to platform owners. Moreover, platforms could have sustainable growth and network effect by democratizing the digital platforms' governance mechanism, i.e., sharing the governance between the application developers and end-users.

2.2.1 Open-source

One approach to democratizing the platform's ownership is to open or serve them with licenses as open-source software, where the software package's source code can be inspected and modified with new features. In contrast, when a platform is closed, similar to proprietary (closed) software, the governance mechanism becomes vulnerable to centralization of architecture and limitations from using other open-source software.

The official definition of open-source from OSI-Docs 2007 shows that open-source is not just about the source code, but also it defines the following criteria:

1. **Free Redistribution:** The license allows selling, royalty, or other fees for such sale as a component of aggregate software distribution.
2. **Source Code:** The license regards the source code and should allow distribution of the source code.
3. **Derived Works:** The license allows modifications, but the derived software packages must still be distributed at least with the same license.
4. **The integrity of The Author's Source Code:** The license must explicitly permit the distribution of software built from modified source code. The license may require derived works to carry a different name or version number from the original software.
5. **No Discrimination Against Persons or Groups:** The license prevents discrimination against persons, groups, or demographics.
6. **No Discrimination Against Fields of Endeavour:** The license prevents discrimination against fields of use, use-cases, or line of business.
7. **Distribution of License:** The rights of the license apply to the other software packages. It is redistributed without an additional license by distributors.
8. **License Must Not Be Specific to a Product:** The license goes similar to other software package derivatives of the original software package.
9. **License Must Not Restrict Other Software:** The license has no right to limit the distribution of other derived software packages with the same license.

10. **License Must Be Technology-Neutral:** The license does not limit technological preferences or the usage (mixing) of non-open-source software packages.

For this project context, open-source undertaken as from [OSI-Docs 2007]; however, the ideation of open-source is still debatable on how far the openness should go and to what extent should IPR (intellectual property right) and chain of creation in open-source projects. For instance, the research policy (West 2003) explores different debatable areas such as free software vs. open source, Linux vs. other UNIX-like platforms, and the understanding through times and companies involved in contributing to the open-source packages.

2.2.2 Architectures of Platforms

The platform ecosystem comprises three significant sides: the platform, apps, and interfaces (Tiwana 2014). A platform's governance and architecture go hand in hand; therefore architecture of platforms plays a significant role in the overall ecosystem of platforms.

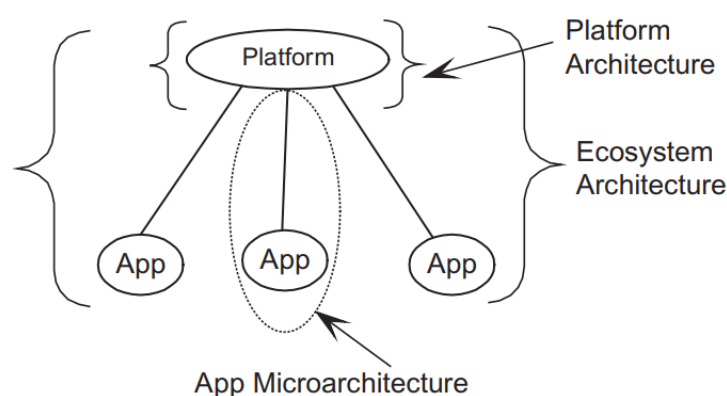


Figure 2.2: Ecosystem architecture.

Platforms deliver their services to applications via boundary resources (APIs, SDKs, ABIs, regulations, and relevant government policies). Similarly, the boundary resources help the applications (and application developers) communicate with the platform and other applications. Depending on how open a digital platform is, boundary resources can have many permissions and services provided for internal or external (third-party applications development).

An open digital platform (ODP) can thus be defined as an extensible digital core open for third parties to contribute improvements or add complements (De Reuver, Sørensen and Basole 2018).

The platform's architecture plays a huge role in keeping existing applications built on the platform, the developers, and the end-users of the applications. Platform architecture imposes constraints on all apps in a platform's ecosystem; therefore, many properties of app architectures are correlated with the platform's architecture (Tiwana 2014); thus, such constraints can be the primary areas of innovation for developers to focus on.

However, if constraints are fundamental, the only option that remains could be to change or move to other platforms. The innovation dynamics of a digital platform often depend on its dependencies with platforms on different levels of the technical architecture (De Reuver, Sørensen and Basole 2018).

Platform Innovations

Innovation platforms are exemplified by mobile operating systems such as Android, and iOS, whose functionality is drawn upon through APIs by a platform ecosystem of third-party developers to build and innovate apps as services (Bonina et al. 2021). Thus, platforms provide ample opportunities for innovations by leveraging their resources via boundary resources.

In addition to that, the boundary resources App developers face two broad types of costs in their ongoing work: (1) app innovation costs and (2) systems integration costs (Tiwana 2014). Such costs can be seen as incentive mechanisms from developers' point of view and as competitive advantage between the different platforms to keep the developers in their platforms.

Developer experiences in platforms play a significant role in innovating and exploring new use-cases, especially in platforms like DHIS2, which engages developers across several geographical and cultural spaces. DEX (Developer Experience) consists of experiences relating to all kinds of artifacts and activities that a developer may encounter as part of their involvement in software development (Fagerholm and Münch 2012).

Involving developers and relevant stakeholders from the ideation to the production stages of a software product enhances the product's success as it can help focus the need of the application's goals with the resources used to build it. To develop software product features that are both desirable and feasible, a great deal of knowledge is needed, both about situated contexts of use and the software architecture (Roland et al. 2017). Several types of participatory design have been explored for involving developers. Several types of research show that PD (participatory design) plays a massive role in involving developers, thus enabling DEX of developers.

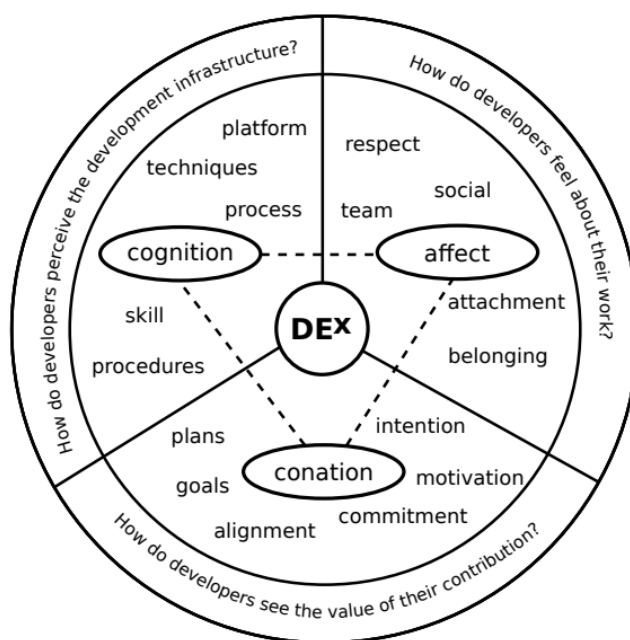


Figure 2.3: Developer Experience Framework, (Figure 1, Fagerholm and Münch 2012)

Moreover, platforms' governance that are put on app architectures do limit or promote innovation of applications and range of use-cases; thus, better platform governance can promote innovations and positive network effects to the platform they own.

By looking into the finance world, specifically the Defi or Decentralised Finance, decentralized governance of platforms is preferred to centralized governance (Chainlink-docs 2022). For instance, many Defi projects propose that innovations on platforms (use-cases) can be increased if the governance of the platforms is decentralized. In many Defi protocols (Chainlink-docs 2022), platform governance and architectural decisions, and incentive mechanisms are community-driven; thus, governance is shared

between the community. Architectural differences can also explain not just the frequency of innovations feasible by app developers but also the types of innovations that do and do not occur in an ecosystem (Tiwana 2014).

Unlike centralized platforms, which are owned and governed by platform owners, decentralized platforms can be owned and governed in more decentralized ways. Some have no platform owners and are governed through community efforts (Chen, Pereira and Patel 2021). Ethereum is one of the few substantial projects (software platforms) in the Blockchain industry. Ethereum's software boundary resources come only as ABIs, reducing the boundary resources developers need to consider to interact with the platform or other projects on Ethereum (Chainlink-docs 2022). The strength of the Ethereum platform is that the decentralized organization and the escrow contract do not need to care about what kind of account each party to the contract is (Buterin et al. 2014).

Keeping the application in a platform also depends on the incentive mechanism the architecture provides to the applications; thus, some developers prefer platforms that promote decentralized application architectures to centralized (traditional client-server) micro-architectures of applications. The level of centralized is many, to mention a few are: governance, storage providers, services provided, and their accessibility.

2.3 WEB-APPLICATIONS (WEBAPPS)

An application initiated by triggering a request from a web server and rendering the result in the browsers (headless or real browsers) is generally called a web application. The modern-day web- applications keep increasing complexity across all industries by providing functionalities in an interactive and accessible manner to the web-application users.

The complexities in web applications keep increasing their building requirements from the primary web programming languages like HTML, CSS, JavaScript, and related libraries to additional requirements of frameworks, SDKs (Software Development Kit), and architectural modifications. Similarly, these constant changes in the web-application building processes require the software builders to adapt to the changes and steadily increase their skills of development.

Teams and projects suffer from delays in delivery or technical debts due to the continuous learning curves of web-application building processes. The research by Cordle 2017, describes how the challenge is continuous, and to have quality Webapps, requires the consideration of many factors. In the research, they describe that the issues with web applications may take time to appear at the start, e.g., initially, AngularJS (a framework) was being used to build an application, yet as developers started to build professional, rich, full-grade applications on the framework with large teams over long periods, its limitations began to show.

2.3.1 Web-Application Development Guidelines

Web applications can be simple interactive, or well-designed, tested applications built with PWA features, and guidelines apply equally to all web applications. Guidelines have massive potential in reducing the overhead of building web applications by saving time and reducing possible technical debts while providing timely practices. Later, following the guidelines remains a vital step to maintaining the quality of the web application, but still, the motivation of application builders can also play a role in the quality of the produced web-application (Graziotin et al. 2018, França, Da Silva and Sharp 2018).

In addition, according to the W3C, guidelines are not testable but provide the framework and overall objectives to help authors understand the success criteria and better implement the techniques (W3C 2018). By exploring a guideline based on a gamified framework, and success criteria for better implementation, the thesis work aspires to contribute to both the industry and academia. Therefore, considering the potential of guidelines, DX (developer experiences), gamification motivates developers to have a notion of constant skill upgrade (a highly required trait among software developers).

2.3.2 Architectures of web applications

App microarchitecture is the same architecture as realized in the implementation of an individual app by its developer (Tiwana 2014). Fig. 2.2 shows an application’s microarchitecture, which constitutes an application’s internal architecture and the interface it uses to communicate with the platform.

Applications can be web-based or native applications (or applications that are required to be installed to be used). Technically, the micro-architectures of the application could be distributed between the client and server-side, e.g., several client-side libraries (as redux for React-based web-applications) or server-side performance-enhancing libraries like GraphQL to increase API usability and performance for the client-side.

In general, the web application architectures in this project are considered for an MVC (Model View Controller) based web applications. MVC is a typical architectural pattern of applications. Model stands for the logical handling of data within the application, e.g., to use the data and generate information or knowledge. In this section, programming languages like java and frameworks like .NET Core and Java Spring Boot are commonly used tools.

View stands for the user interface or the part of the application that the user interacts with. In this part, libraries like ReactJS and Vuejs are used tools. This includes user application design, user interface (UI), and user experiences (UX).

Lastly, the controller is the section that manages the flow of data, including data security and updating of the view with updated data. Controllers communicate as middlemen with both the other parts: View and Model.

Types of applications

In the typical standard approaches of modern-day web applications, the major architectural classifications can be divided into two types: monolithic or microservices. Similarly, the general trend in this time is the migration of existing or legacy monolith applications into the cloud-based microservices-based application to leverage scaling from the cloud services.

Benchmark	Monolithic	Microservice
Scalability	Challenging because the whole application has to scale	Easier because a single microservice can scale independently of other microservices
Speed of delivery	Faster because it is easier to manage, troubleshoot, and develop a single application	Slower delivery because of the different independent components of the application.
Security	Easier because it is concerned with testing and securing a single application	Challenging because of higher attack surfaces on the independently deployed components
Complexity	Less complex because technical requirements are fewer	More complex because it could be distributed across different cloud platforms
Price (cost)	Cheaper because it requires few team members, and simple applications	Pricer because it requires sufficient engineering levels and domain expertise

2.4 GAMIFICATION

Gamification can be defined through many but related definitions: "the process of game-thinking and game mechanics to engage users and solve a problem" (Zichermann and Cunningham 2011), "the use of game concepts in a non-gaming environment which is used in many fields such as businesses, health, and education"(Elshiekh and Butgerit 2017), it is an informal umbrella term for the use of video game elements in non-gaming systems to improve user experience (UX) and user engagement (Deterding et al. 2011). Thus, those definitions show that gamification plays a role in engaging participants in different environments for good. Moreover, its ability to express a complex environment in simple terms helps transfer valuable attributes of a use-case to different use-cases (environments).

Gamification's usability to educate software development can be seen in different forms as seen in web-programming teaching platforms: Codecademy (with short programming instructions and instant feedback), CodeSchool (by using different badges and image rewards to show a status of progress), CodeCombat (by showing a map of a place with interactive messages with progressing in programming), Treehouse, and many others. Gamification has become an effective technique in education in general and is useful in programming courses (Elshiekh and Butgerit 2017; Khaleel et al. 2015).

2.4.1 Theoretical Underpinnings of gamification

The importance of gamification is also constantly rising across many industries. Especially during the pandemic; during the covid-19 days, gamification is seen to improve students' performances (Whiddington 2020), GameFi (gaming use-case and merging of gaming with finances for blockchain), which is a disrupting or challenging scene in the gaming industry.

A practical gamification concept captures and retains learners' attention, engages, entertains, challenges them, and teaches them (Furdu, Tomozei and Kose 2017); therefore, a careful design and architectural implementation of a gamified system are recommended to fit a specific use case.

Gamification architecture must be designed by constituting aesthetics, dynamics, and mechanics (Zichermann and Cunningham 2011). Mechanics make up the functional components of the game, which can be points, badges, levels, challenges, and rewards. Aesthetics of the game is about how the game makes the player feel during an interaction, and dynamics are about the player's interactions with the game's mechanics. This architecture of gamification is preferable when considering the application development resource site for reasons:

2.4.2 Gamification: pros & cons

Pros

- Gamification is a practical approach to positive change in students' behaviour and attitude toward learning to improve their motivation and engagement (Kiryakova, Angelova and Yordanova 2014). Moreover, as a new opportunity to convey and receive information, it promotes an alternative avenue to knowledge which can further enhance the general goal of learning the software development by following the guidelines to complete the tasks.
- When learning is joyful for learners of a new way of software development, it is the preferred option for it reduces the possible negative repercussions that could come from failing to learn it. A better learning experience is obtained by combining fun with instant feedback (Furdu, Tomozei and Kose 2017), which shows that gamification can enforce a positive feel for web-application building.
- The use of gamification not only facilitates work but also assists in learning, provides feedback on work done, and helps compare work productivity over time (Platonova and Bērziša 2017). Feedback cycles in learning web-development are essential to organize tasks and sub-tasks, improve code quality, improve collaboration in teams and enforce agile software development methodology.

Cons

- Too much gaming can have addictive side effects, which can hurt the performance of individuals. Further researches show that it can have adverse effects such as indifference, loss of performance, undesired behavior, and declining effects (Toda, Valle and Isotani 2017).
- The possible short time side-effects of gamification could prompt long-term serious side-effects side-effect of gamification. We found that though students from each course started at the same levels of intrinsic motivation, satisfaction, effort, social comparison, and empowerment, over time, students in the gamified course tended to decrease in motivation, satisfaction, and empowerment relative to the non-gamified course (Hanus and Fox 2015).

Albeit all the shortages of gamification, when designed correctly, it has the potential to maximize the positive effects and improve learning in software development applications. It does this by abstracting the different parts of software tasks into manageable pieces and concentrating on opening developers' minds by teaching how to abstract problems and providing solutions by structuring a set of different web-application building.

2.4.3 Gamification Design approaches

Frameworks

The need for engaging learning games has led to the development of design frameworks and processes rooted in traditional instructional design models (Dimitriadou et al. 2021). However, the frameworks in the paper do not follow the same designs, making it challenging to have a common or standard design of games. Similarly, platforms like Unity, AppGameKit, CryEngine, Unreal Engine, Cocos2d-x (see fig 2.4, next page), Amazon Lumberyard, and Titanium platforms are used to build games.

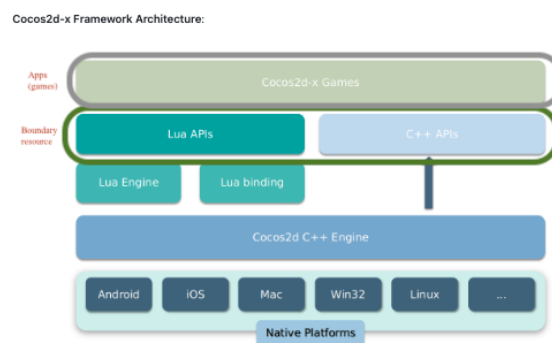


Figure 2.4: Open-source Game building framework (cocos2d-x)

Gamification frameworks have many advantages:

- Designing effective game models to test or prototype,
- Designing software architecture for scalability, game resolution, or better UX (user experiences)
- Designing effective accessible incentive mechanism,
- Designing interoperable software, e.g.:
 - (a) for smooth migration from centralised to decentralised game models or vice-versa,
 - (b) for integration with other technologies as Extended Reality (XR), Augmented Reality (AR), Mixed Reality (MR), or Virtual Reality (VR),

Incentive mechanism

Moreover, from an industry and market valuation point of view, the gaming industry is still by far the biggest in the entertainment industry by market capital (see fig 2.5), and the valuations are related to the games' business models. Those business models do, in turn, include the players, which can be models like play-to-earn, game purchases, game feature purchases, and many other business models.

Implementing game models in use-cases unrelated to finances (e.g., open-source or non-profit projects) is challenging to design because the incentive mechanisms in both environments are different. Well-designed incentive mechanisms are fit enough to keep players engaged and motivated to come again to use a gamified environment.

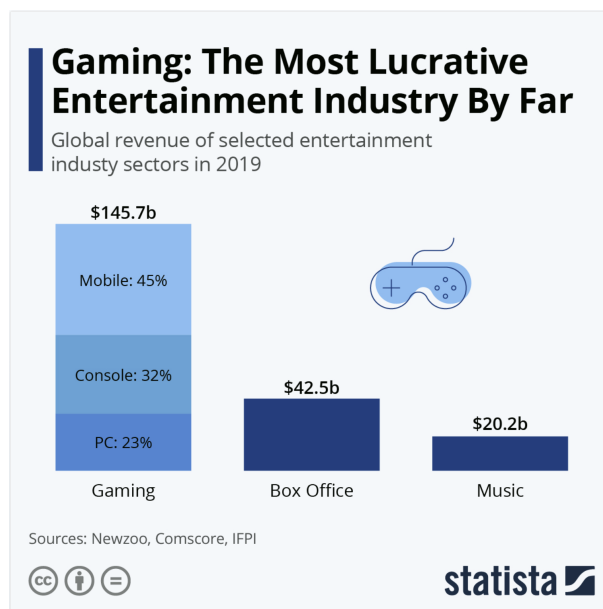


Figure 2.5: Game and entertainment industry by market capitalisation

To avoid any possible negative side-effects, game elements need to be studied as the reward mechanism could have negative impacts as well, it is possible to observe that Leaderboard had a strong influence on almost all of the negative effects, followed by Point and Badge, both with the same influence (Toda, Valle and Isotani 2017).

BACKGROUND

This chapter covers the background of DHIS2 by focusing on software or technical side.

3.1 DHIS2

DHIS2 (District Health Information System 2.0), as a software platform, is a flexible information system for data capture, management, validation, analytics, and visualization (see www.Github.com/dhis2/dhis2-core). Moreover, it provides tools and services to build a multitude of use-cases, mainly within health information-related systems. It is currently the most extensive Health Information System available in the world, and it is managed by the HISP Center at the University of Oslo (UiO).

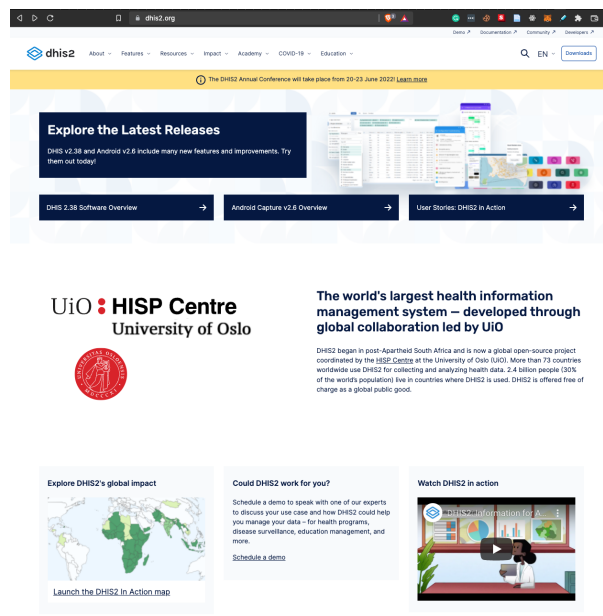


Figure 3.1: DHIS2 Homepage

Historically, it started to solve the necessary solutions to respond to public health emergencies and information management challenges in South Africa. The strategy adopted to achieve this aim was through tools and data standardization, developing essential datasets, and a software application to support its implementation (Adu-Gyamfi, Nielsen and Sæbø 2019).

DHIS2 interfaces with third-party web portals and technologies, including SMS, E-mail, and Geographical Information Systems (GIS), to enhance its functionality. DHIS2 provides a multitude of features and integrations (Adu-Gyamfi, Nielsen and Sæbø 2019). Such integrations enable further innovations and innovative use-cases in education, agriculture, e-government, and logistics management (see www.dhis2.org/user-stories).

The development of DHIS2 software is led by core developers based in Oslo (University of Oslo) in collaboration with research teams, viz. HISP, student (academic) researchers, and other external contributors from all over the world. Moreover, since it is open-source, it is open for software developers to contribute, and it provides contributors guidelines and access to tasks listed in Jira (issue tracking product).

Some of the many projects managed by core developers:

- DHIS2 Developer Portal: Home page of the necessary developers' documentations.
- DHIS2 Core: To provide the data models and services which are exposed through a boundary resource (RESTful Web API),
- DHIS2 Application Platform: Gives ample opportunity to standardise the application development tooling for the web-applications built on DHIS2.
- DHIS2 UI and Design approach: Shows a means to design the application and maintain consistency of applications in the platform.

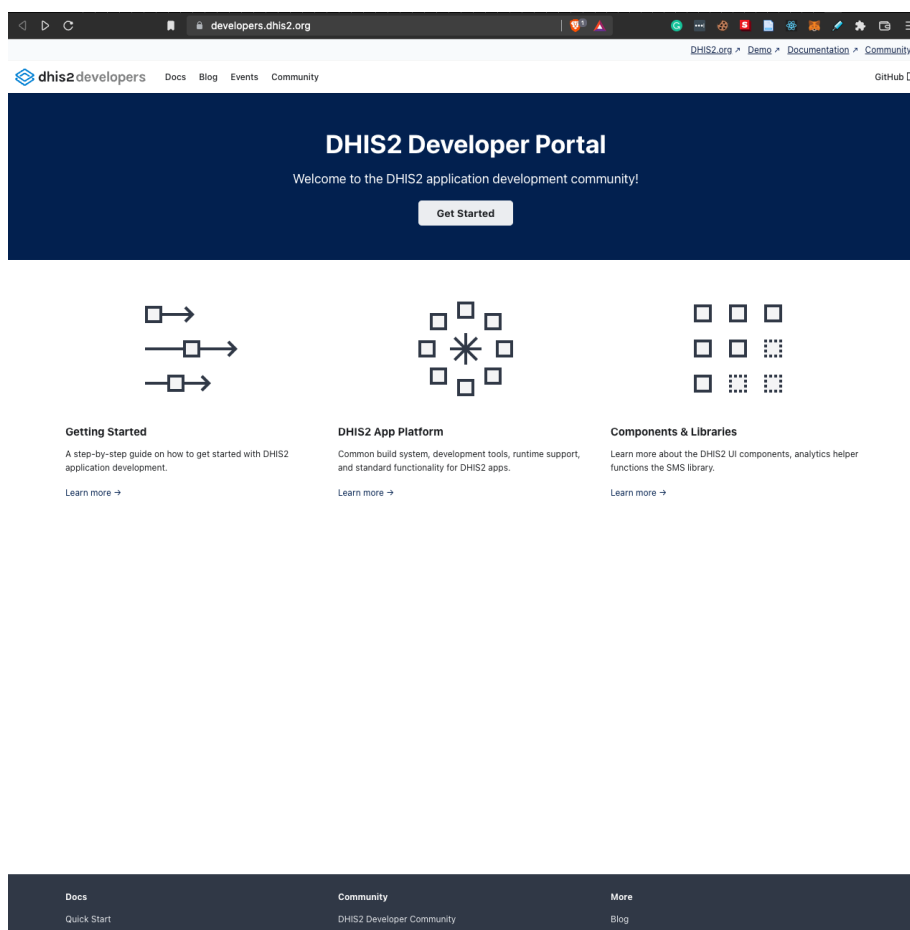


Figure 3.2: DHIS2 Developer Portal

3.1.1 DHIS2 and Application Development

The software has evolved from a custom-made desktop application to a modular web-based platform used globally (Roland et al. 2017). Thus, DHIS2, as a digital web-based platform, leverages several digital resources for complementary web-applications that are built on the DHIS2 platform. These resources, among many, include *programming resources* such as documentation about developer guidelines, technical documentation about UI components, build systems, development tools, runtime support, and standards for software-code linting and testing.

Documentation

Several types of documentation exist for the DHIS2 community, starting from setting up DHIS2 web-application projects, using existing boundary resources, training, hosting services, and getting related support services.

- DHIS2 Developer Community: Here web-application builders (DHIS2 practitioners) ask or discuss questions in multitude digital communication options, viz. DHIS2 Community, by joining the Slack channels or get community updates in Twitter.
- Developer Guidelines: Here, there are set of all the necessary documentations for developers.
- Level 2 Academies: Here one can pick from the list of all the possible use-cases that one can build using DHIS2, for instance: about Web App Development.

DHIS2 Design principles

A set of design principles exist for building web-application provided by DHIS2, which helps produce high-performing web applications that are easy to maintain, upgrade, and high performance.

Standards of Software Code

Using d2 (a tool provided by DHIS2 to bootstrap web-application projects), the web-applications built can have relatively similar tooling. This, for instance, can be with linting (a process that improves code readability), testing (a process to unit test functions), and building (to prepare web-application for deployment or hosting).

UI-Components

DHIS2 has custom-made reusable UI components built with React (the most commonly used JavaScript library for front-end projects). Those UI components are built with DHIS2 design principles and with the ability to integrate with other web-application building support libraries, e.g., Redux, custom data providers, and other libraries.

Development tools

To build web-applications, the bare minimum requirement for tooling includes the following:

- OS (Operating System): MacOS, LinuxOs, or Windows,
- IDE (Integrated Development Environments): VsCode, Atom or Sublime,
- Softwares for development: Git (for version control), nodejs, d2
- Useful libraries for the web-applications: ReactJs (for UI), Redux (for state management), CypressJs (for integrated tests)

METHODOLOGY

This chapter explores the methodology that is the cornerstone of the research: the design process of the artifact and the relevant analysis processes starting from early start of the project.

Initial ideation of the thesis work was conceived from the Design lab thesis work projects, (UiO 2020), then following several discussions with Dr. Johan Ivar Sæbø (the thesis work project supervisor), getting involved in several meetings (discussions and workshops) with the Design lab (UiO 2020) team and email communications with DHIS2 core developers, then finally the project was started.

The project's work plan contains several parts: literature reviews, an artifact design, development, and evaluation, following a research paradigm (design-science research) approach. The critical differentiator between professional design and design research is the clear identification of a contribution to the archival knowledge base of foundations and methodologies and the communication of the contribution to the stakeholder communities (A. Hevner and Chatterjee 2010). Therefore, designing the artifact by following DSR gives ample opportunity to explore means for developers to use the application resource site.

4.0.1 Research process

An IT artifact, implemented in an organizational context, is often the object of study in IS behavioral-science research (A. R. Hevner et al. 2004). The artifact can be assumed as a gateway to exploring a problem for intelligent solutions by following research fit to reach a goal solution. Thus, several iterations of the artifact exist to understand how gamification pertains to learning software development.

A DSR approach can be seen as a complex process with many aspects that need be considered. DSR is an iterative process, starting with identifying a problem in the problem space and evaluating alternative solutions in the solution space (Brocke and Maedche 2019). Thus, after the last iteration and thorough research follow documenting the takeaways from the project; moreover, the challenges and shortages of the project are documented after practical evaluation of the artifact.

The figure (fig 4.1.) below shows a DSR framework from (Fig. 2.2, A. Hevner and Chatterjee 2010); the framework has three cycles: *relevance cycle*, *design cycle* and *rigor cycle*. The design cycle is where the artifact is built and evaluated, and the gamified artifact is the main focus. The research process consists of those three cyclic processes by collecting data from participants, evaluators, and learnings along the overall process of the thesis work. Finally, scientific theories and relevant research results are extracted after the evaluation process (stage 5 in the framework).

4.0.2 Design Science Research

Design science research effectively fits the process of designing and developing a gamified artifact (for application development resource) and then evaluating it (with the goal of an application development resource site) through iterative processes. In order to thoroughly test an artifact, it is recommended for multiple iterations of the design cycle in design science research before contributions are output into the relevance cycle and the rigor cycle (A. Hevner and Chatterjee 2010).

Evaluation of an artifact in DSR can have several ways, and according to (Venable, Pries-Heje and Baskerville 2016), the evaluation processes are done before and after the artifact is built (namely *ex ante* and *ex post* evaluations). During the evaluation, the artifact is questioned if it answers the desired solution. According to (A. Hevner and Chatterjee 2010), one can ask if the design artifact improves the environment and how this improvement can be measured.

The answers to the questions give input for further study of the problem, measurements, and evaluation processes. In addition to that, they mention that to distinguish between formative and summative evaluation. This distinction does not arise in the innate qualities of the evaluation process but instead inhabits the functional purpose of the evaluation (A. Hevner and Chatterjee 2010).

Considering the research paradigm (fig 4.1), the checklist is in the table (table 4.1) on the next page. The answers in the table are taken as evaluations of the respective DSR process, which can be seen in fig 4.1. Since the **research question** 1.2 is to explore how *how gamification pertains to learning software development*, three iterations can be seen in the design cycle.

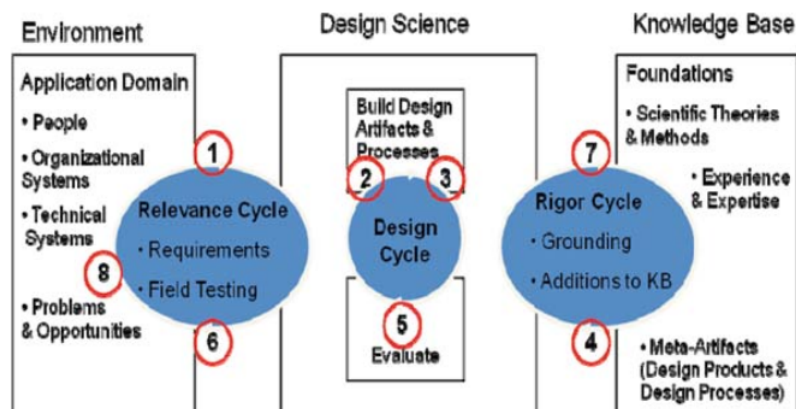


Figure 4.1: Design Science Research checklist of evaluation. (A. Hevner and Chatterjee 2010)

The table below maps the relevant questions (checklists) in each paradigm cycle. The major iterations in the project come in three, and the DSR follow the same structure for the three cycles as seen in the table below:

Question	Iteration 1	Iteration 2	Iteration 3
1. What is the research question (design requirements)?	How aspects of gamification and documentation guidelines pertain to the building of web-applications in a software development platform?		
2. What is the artefact? How is the artefact represented?	Web-page (https://app-resource-dev.vercel.app/)	Github repository (yonatanhf/Application-Development-Resource)	Github repository (dhis2designlab/Application-Development-Resource)
3. What design processes (search heuristics) will be used to build the artifact?	To use tested and stable libraries thus uses DHIS2 documentation library (docsify), relevant content	To automate badge reward and maintain privacy, thus uses Github-actions	To improve accessibility and testing of artifact
4. How are the artifact and the design processes grounded by the knowledge base? What, if any, theories support the artifact design and the design process?	Accessibility (i.e. to be similar to existing products as DHIS2: platform, runtime, cli)	Privacy, automation and gamification rewards	Usability (accessibility)
5. What evaluations are performed during the internal design cycles? What design improvements are identified during each design cycle?	Requests for reviews from peer MSc students and evaluating reviews	Requests for reviews from Supervisor, peer MSc students and evaluating responses	Requests for reviews from Supervisor, DHIS2 core-developers and evaluating responses
6. How is the artifact introduced into the application environment, and how is it field tested? What metrics are used to demonstrate artifact utility and improvement over previous artifacts?	As a website, but misses metrics and SSL security	As a Github page, over the previous version, it improves SSL security, privacy, and automation. The metrics used are a number of forks.	As a Github page, it improves the readability and usability of the page.
7. What new knowledge is added to the knowledge base and in what form (e.g., peer-reviewed literature, meta-artifacts, new theory, new method)?	Importance of website security, metrics, and gamification design	Github actions for automation of tasks, game reward mechanisms	Importance of usability, documentation readability
8. Has the research question 1.2 been satisfactorily addressed?	No, because gamification was not added to the website	Relatively yes: application resource site and gamification	Relatively yes

4.0.3 Data collection

Data collection is the process of getting responses from research participants and the artifact's analysis, including the type of metrics and the type of tools used to collect the data for the artifact.

Data collection in DSR (Design Science Research) using CFGs (Confirmatory Focus Groups) is useful for participants exploring the artifact, which is designed in the application field. When using focus groups for rigorous research, the unit of analysis will be the focus group and not the individual participants. (A. Hevner and Chatterjee 2010)

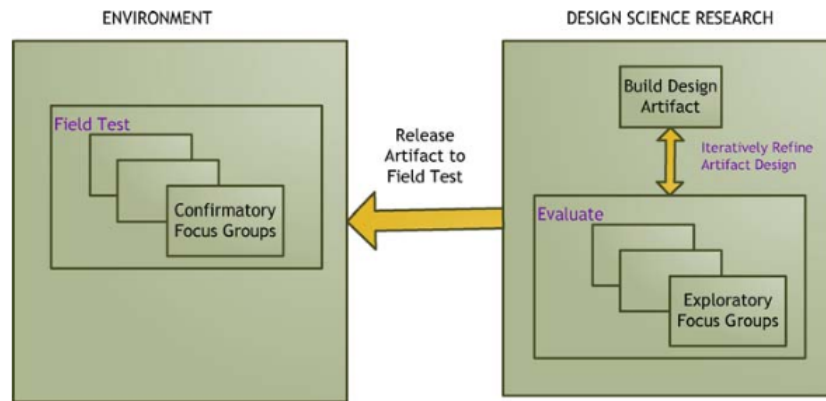


Figure 4.2: Focus groups in DSR, (fig 10.2, A. Hevner and Chatterjee 2010)

ARTIFACT DESCRIPTION AND EVALUATION

This chapter describes the artifact (gamified application development resource site) built to serve as a resource for building web-application by following a step-by-step guideline in a software platform (DHIS2 platform). Moreover, this chapter considers the process model of the artifact, the measures and assessments taken during the design, and the technology choices used in the artifact.

In the DHIS2 software platform, application builders use Github to store and manage software codes. Moreover, DHIS2 core developers use Github to manage projects (can see them at <https://github.com/dhis2>).

5.0.1 How the artifact works

The artifact provides an environment to explore the ideation and organization of the proposed (gamified application resource site) into an application development resource site form. Focusing on the gamification aspects and step-by-step documentation guidelines, it serves as an environment to review the results of the artifact in the end. Then it rewards users with badges for completing the milestones described in the artifact while following the process in an interactive gamified manner.

The artifact is a Github repository and an artifact to be used by itself. Most Github repositories are hosted or deployed in a hosting environment; however, this artifact is already ready to be used by forking it from the original repository (upstream).

Automation

Github actions perform the automation; the activity diagram below explains the flow of control between the parts of the artifact or the transition in states and controls between the different parts of the artifact and the human participant.

The screenshot shows the GitHub repository page for 'dhis2designlab / Application-Development-Resource'. The repository is public and has 44 commits. The file list includes folders like 'github/workflows', 'assets', and files like 'dhis2', 'html-css-js/level-1', 'react', '.gitignore', 'README.md', and 'README.md'. Annotations include:

- Step 1:** A red arrow points to the 'Fork' button in the top right, with the text 'Step 1: Click and fork this repository'.
- Step 2:** A red arrow points to the 'Gamified Application-Development-Resource' section, with the text 'Step 2: Read this step-by-step guides required to configure the personal artefact.'
- Step 3:** A red arrow points to the 'react' file in the file list, with the text 'Step 3: Click on tasks, follow the available to be worked on'.
- Languages:** A green oval highlights the 'Languages' section, which shows a bar chart for JavaScript (96.3%), CSS (1.9%), and HTML (1.8%). A red arrow points to it with the text 'Programming languages that build the artefact'.
- Badges:** A blue arrow points to the 'Web Applications' completion status section, which shows 'HTML-CSS-JS' and 'React' with 'level-1' badges that are 'resource not found'. The text says 'Badges awarded automatically for completed tasks'.

Figure 5.1: Original repository and configuration steps

The artifact works by leveraging Github features (see Fig. 5.1), i.e., Git pushes as inputs from a user, then processes the input (trigger Github actions) and awards badges as incentives to keep the user engaged with the artifact. The artifact's process model refers to the input/output of tasks and the conditions required for each task.

The artifact process models can be classified into two parts: *Artefact's process model* and *User tasks' process model*:

- **Artifact's process model:** The artifact's building process model or SDLC (Software Development Life Cycle Model) follows an agile software development. Such a model is primarily used type of software development process model, by which features are added through the continuous process by taking inputs from contributors (mainly the project supervisor and core developers).
- **User tasks' process model:** These tasks are each set the user needs to complete to earn the badges; similar to the todo section, user tasks in an agile board can be assumed to be user stories from agile methodologies. User stories are traditionally written on note cards, where cards may be annotated with estimates, notes, Etc. (Cohn 2004). A good user story contains metrics on how long a task takes, what conditions need to be met, and the acceptance metrics for defined user stories.

Browsing the game tasks:

a. *html-js-css* stories:

1. level-1:

Requirement: Goal to build - *A registration form using HTML, CSS and JS,*

Details:

The screenshot shows a GitHub repository page for 'Application-Development-Resource/html-js-css/level-1'. The repository is owned by 'yonatan88'. The commit history table lists several commits related to adding and updating files like 'cypress', 'src', 'badge-status.js', 'cypress-cucumber-preprocessor.config.js', 'cypress.env.json', 'cypress.json', 'package.json', 'readme.md', and 'server.js'. The 'readme.md' file is open, showing the task details. The goal is to build a registration form using HTML, CSS, and JS. The README provides completion requirements, a reminder to copy and paste code, and a step-by-step guideline to achieve the goal. The step-by-step guideline includes starting the application, finding the index.html file, and changing the content inside the <body> with the provided HTML code.

Task identifier points to the repository name: `yonatan88/html-js-css/level-1`

Goal points to the text: **A registration form using HTML, CSS and JS.**

Step by step guideline to achieve the goal points to the section: **Steps to follow (4 steps):**

```

<!-- Form -->
<div id="formContainer">
  <h2>Register</h2>
  <p>Please fill in this form</p>
  <hr />
  <label for="username">Username: </label>
  <input
    type="text"
    placeholder="Enter username"
    name="username"
    id="username"
    required
  />
  <br />
  <label for="firstName">First Name: </label>
  <input
    type="text"
    placeholder="First Name"
    name="firstName"
    id="firstName"
    required
  />
  <br />
  <label for="email">Email: </label>
  <input
    type="text"
    placeholder="Enter Email"
    name="email"
    id="email"
    required
  />
  <br />
  <label for="password">Password: </label>
  <input
    type="password"
    placeholder="Enter Password"
    name="password"
    id="password"
  />

```

Figure 5.2: Task: *html-js-css* level 1

b. react stories:

1. level-1:

Requirement: Goal to build - *Calculator WebApp with ReactJS (and JSX)*

Details:

The screenshot shows a GitHub repository page for 'dhis2designlab / Application-Development-Resource'. The task identifier is 'yonatanH update readme: csa update requirement'. The goal is to build a 'Calculator Application using html and jsx (can see jsx docs)'. The step-by-step guide includes:

- Initializing the application by CRA (Create React App):
 - It has two numeric values as inputs and perform mathematical operation,
 - It displays the result.
- Remember:
 - All you need to do is copy and paste the code in the boxes below into their respective files.
 - When following the steps below, save the work and access the application on the browser (<http://localhost:3000>) to see the changes.
 - To test (see [testing](#) at the bottom) before submission.
- Steps to follow (4 steps):
 - First: inside level-1 application directory:


```
cd level-1
```
 - Second: setup application development environment, run the following commands respectively:


```
npm create-react-app app
cd app
```
 - Third: edit the App.js file inside the src directory
 - Import React library (copy & paste the following line into the file App.js):


```
import React, { useState } from "react";
```
 - Initialize state variables and define the necessary functions: (copy and paste the following inside the function App[()...]):


```
// defining state variables
const [values, setValues] = useState({
  firstNumber: 0,
  secondNumber: 0,
  mathOperator: "+",
});
const [result, setResult] = useState(0);

// define input handling function
const handleSubmit = (e) => {
  let name = e.target.name;
  let value = e.target.value;
  values[name] = value;
  setValues(values);
};

// defining math operating function
const handleMathOperation = () => {
```

Figure 5.3: Task: React Js level 1

c. *dhis2* stories:

1. level-1:

Requirement: Goal to build - *Initiate, log-in and display the DHIS2 homepage*

Details:

The screenshot shows a GitHub repository page for 'dhis2designlab / Application-Development-Resource'. The main content is a README file titled 'dhis2-app-init readme update'. The page is annotated with three red arrows and labels:

- Task identifier:** Points to the repository name 'Application-Development-Resource / dhis2 / level-1' in the breadcrumb navigation.
- Goal:** Points to the section 'Initializing a local DHIS2 Application:' under the heading 'Applications To Build:'.
- Step by step guide to reach the goal:** Points to the 'Steps to follow (4 steps):' section, which includes instructions for setting up the application environment, installing dependencies, and starting the application.

The README content includes the following sections:

- Applications To Build:**
 - Initializing a local DHIS2 Application:**
 - Completion requirements:
 - It has a DHIS2 header
 - It displays 'Hello World!'
 - Remember:
 - All you need to do is follow the steps and make required changes in each step.
 - When following the steps below, save the work and you may reload the browser (<http://localhost:3000>) to see the changes.
 - To test (see [testing](#) at the bottom) before submission.
 - Steps to follow (4 steps):
 - First: go to the level-1 application environment:


```
cd level-1
```
 - Second: setup a dhis2 application development environment, run the following commands respectively: `cd app scripts; exit`; if you get an error, you might need to install `d2` (read about `d2`) by running: `yarn add d2`
 - After setting up the application development environment, start to install `node_modules` by:


```
yarn install
```

then you might start the application by:

```
yarn start
```
 - Third: you need also a running DHIS2 server. To start a DHIS2 server instance, run the following commands (in a new terminal window):


```
dhis-portal --serverplay --instance-dev
```

If you get an error or `dhis-portal` is not installed, run the following command and try to run the above command again:

```
yarn global add dhis-portal
```

Figure 5.4: DHIS2 Application level 1

2. level-2:

Requirement: Goal to build - *A DHIS2 application that lists and displays lists' details*

Details:

The screenshot shows a GitHub repository for 'dhis2designlab / Application-Development-Resource'. The main content is a file named 'readme.md' which contains the following information:

Task identifier: Application-Development-Resource / dhis2 / level-2/

Goal: An application that displays lists and details

Completion requirements: This level is based on the application described in this assignment and you may read the requirements there.

Remember:

- All you need to do is follow the steps and make required changes in each step.
- When following the steps below, save the work and you may reload the browser (<http://localhost:3000>) to see the changes.
- To test (see [testing](#) at the bottom) before submission.

Steps to follow (5steps):

- First: go to the application directory called 'level-2' and you could use:


```
cd level-2
```
- Second: in a new terminal, fire up DHIS2 instance by executing the command: `dhis2-portal --server=play --instanceid=`, which can be accessed at <http://localhost:9999>
- Third: in another terminal of the 'level-2' directory:
 - Install the node modules:


```
yarn install
```
 - Start the application by executing the command:


```
yarn start
```

 - access it at <http://localhost:3000> on your browser (Chrome preferable)
- Fourth: the starting source code of the application is found in the `src` directory:
 - To build the application, start editing the file: `App.js` :

So, follow the steps below (copy & paste the code):

 - To start, you might need ReactJS library, so add the following line on the top:


```
import React, { useState } from "react";
```
 - You might use `useTableQuery` to fetch data from the DHIS2 instance to the application, see the example at [DHIS2 Application Runtime](#), so add the following code on the top:


```
import { useTableQuery } from "dhis2-app-runtime";
```
 - The application need to make it look like [design](#), so so add the following code on the top:


```
import {
  Table,
  TableBody,
  TableCell,
  TableHead,
  TableCellHead,
  TableShow,
  TableHeader,
} from "dhis2/ui-core";
```

Figure 5.5: DHIS2 Application level 2

5.0.2 Artifact design

In IS, ISDT (Information Systems Design theory) has several theses on picking a preferable design theory. The book (A. Hevner and Chatterjee 2010) suggests that when building artifacts, design theory in ISDT could provide ways to design the artifact.

The design process had several iterations, which can be assumed to be imagined as several versions (evolutions) of the artifact and can be seen in the section 4.0.2. Since the artifact explores the implementation of gamification, step-by-step guideline, and software development, it is not feasible to use one design theory for all. Therefore, the design theories in the artifact contain theories from those areas, viz., step-by-step guidelines, gamification design, platforms, security, and privacy.

After the design process follows the development of the artifact and the feedback from the project supervisor, the DHIS2 core developers and individuals who test the artifact have given productive inputs to design it better. In such a manner, the final version of the artifact is the result of several iterations.

Gamification design

The gamification design is a continuous work in progress. Nevertheless, the artifact is currently based on three major parts: the Aesthetics, Dynamics, and Mechanics parts of a game from the book (Zichermann and Cunningham 2011). The game design by Zickermann and Cunningham is well represented in many research works and can fit all types of game environments.



Figure 5.6: Gamification by design: Implementing game mechanics in web and mobile apps. (Zichermann and Cunningham 2011)

Completing the user stories and earning rewards:

When the user stories or the tasks that the user need to complete are done by a user, the user commits those changes and sends those changes via git push operation. The figure below (fig 5.7) shows the git commit history of the series of completed tasks.

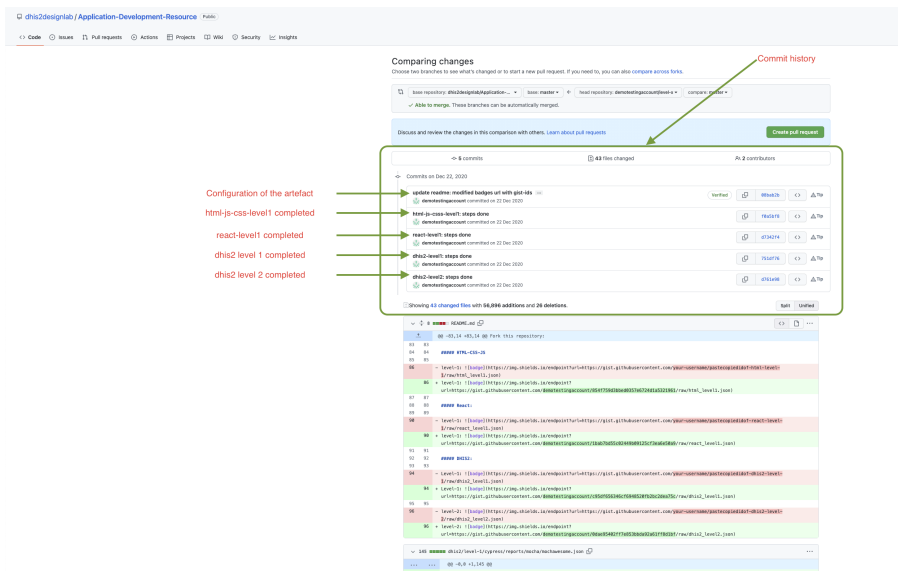


Figure 5.7: Commit history of completed tasks

Interpreting rewards:

Fig 5.1 (in its lower section) (*Web Application's completion status*) is the area where the badges awarded are displayed. Fig 5.8 shows how the awards look when the tasks are completed, and badges are awarded.

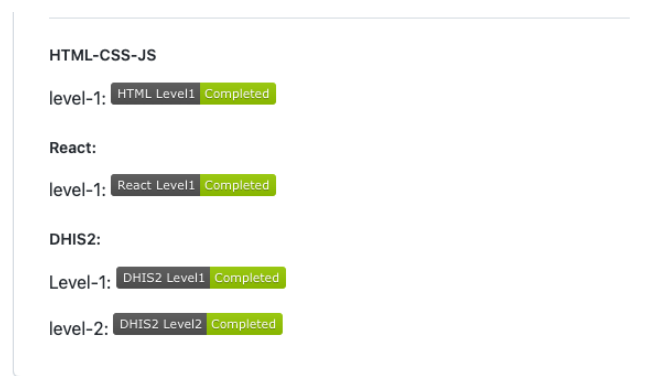


Figure 5.8: Types of awards (badges) given

5.0.3 Evaluation of artifact architecture

The figure above (fig 5.9) shows the arrangement of the major parts of the current artifact architecture:

Github Actions:

Github Actions allow the automation of tasks based on various triggers (e.g., commits, pull requests, issues, comments, etc.) and can be easily shared from one repository to another, making it easier to automate how developers build, test, and deploy software projects (Kinsman et al. 2021). In this project, Github Actions check if the steps defined in the tasks are correctly put and to finally provide Badges for correctly put projects.

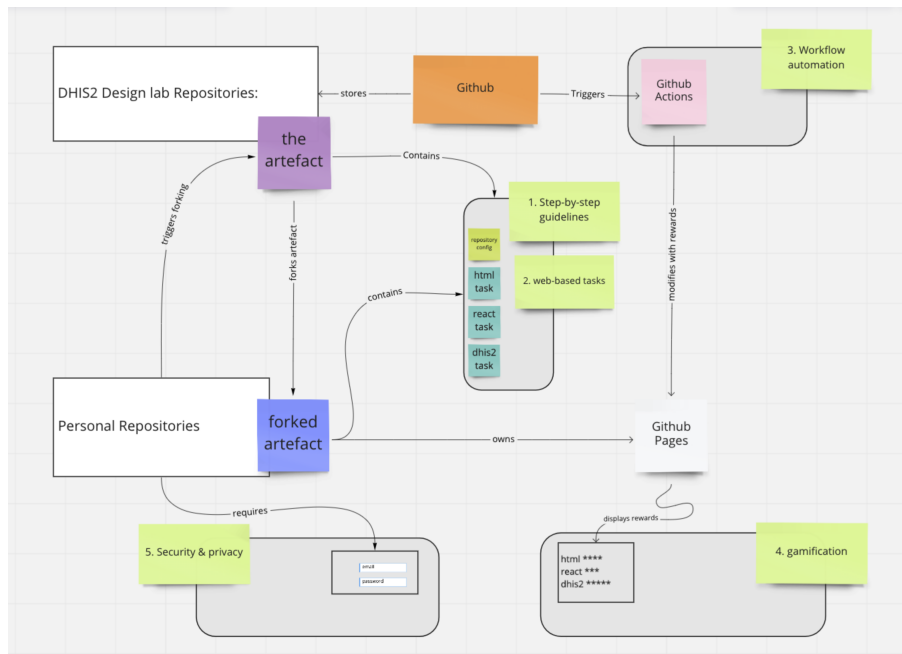


Figure 5.9: Artifact architecture diagram

A workflow can contain one or more Actions. Developers can create their own Actions by writing custom code that interacts with their repository, and use them in their workflows or publish them on the Github Marketplace (Kinsman et al. 2021). The current workflow of Github actions can be seen in fig.s 5.1 & 5.9, by which the Github actions are triggered when a developer (user of the artifacts) pushes code to his/her repository.

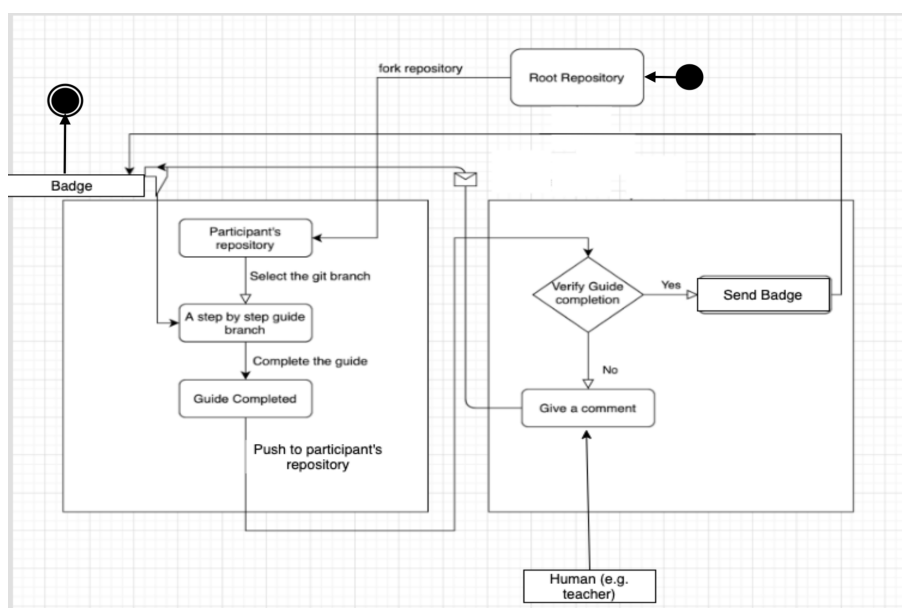


Figure 5.10: Activity Diagram

- **Root Repository (the artefact):**
Github, being one of the most secure software development management platforms, it is being used to secure millions of software applications and projects. For instance: DHIS2 core applications and DHIS2 Design Lab repositories can be seen in Github. This is location of the artifact, which is deployed (found) as one of the DHIS2 Design Lab Github repositories.
- **Forked Repository (the personal artifact of the application builder):**
This is the copy of the root repository (artefact), but owned as personal artifact of the artifact user (participant or application developer). The user can have copy of the root repository by forking the root repository.

- Documentation (step-by-step guideline, web-based tasks):
This section describes the activities to be performed by the user in-order to complete the tasks and earn the rewards.
- Gamification (badges, response mechanism):
This part includes the game parts of the artifact, viz. rewards, task completion incentive mechanisms, response automation and gamification related specifications.
- Security & privacy:
Since the artifact is open-source project, security (specifically privacy) is important part of the artifact's evaluation. Github's security measures are used to secure millions of software projects and leveraging the same security measures from Github the participants do have the same level of security. For more, see at the Github about forks.

OWASP (Open Web Application Security Project) is one of the most used developer guidelines to audit or analyse security risks of web applications. It is equally used with-in the open-source and commercial security risks and respective mitigation. Considering the artifact, the table below describes the top 10 security risk from OWASP (Stock 2021) as measurement of evaluating the security of the artifact.

OWASP Top 10 checklist: 1-7 (Stock 2021)				
Attribute	Description & areas of use	Artifact iteration 1	Artifact iteration 2	Artifact iteration 3
A01:2021-Broken Access Control	Access control management of the user to the artifact, e.g., to manage access privilege, the permission of other users, and access to APIs.	Irrelevant, since users do not need to save or share information in/from an artifact.	Protected by Github security measures	Protected by Github security measures
A02:2021-Cryptographic Failures	About protecting sensitive data, e.g., encrypting transmitted data between software parts	Not required	Not required	Not required
A03:2021-Injection	To mitigate SQL, NoSQL, OS, and LDAP injections that come via query (untrusted data is sent to an interpreter).	Not required, since there is no database	Same as previous iteration	Same as the previous iteration
A04:2021-Insecure Design	About security design concerns	No sensitive data is collected or saved.	Same as iteration 1 and leverages Github security designs	same as the previous iteration
A05:2021-Security Misconfiguration	About security misconfiguration within the architecture of the artifact, e.g., setting default user/password for the artifact.	Not required	Github actions do not require security configuration	Same as the previous iteration
A06:2021-Vulnerable and Outdated Components	About detecting, fixing, and mitigating security issues from outdated components.	Requires constant security auditing of libraries like docsify	requires constant security auditing of libraries	Similar to the previous requirement
A07:2021-Identification and Authentication failures	About authentication via passwords, 2FA (two-factor authentication) and session management	Not required	Handled by Github identity management	Same as the previous iteration

OWASP Top 10 checklist: 8-10 (Stock 2021)				
Attribute	Where to implement	Artifact Iteration 1	Artifact Iteration 2	Artifact Iteration 3
A08:2021-Software and Data Integrity Failures	About the artifact updates or critical data. CI/CD (and Github actions) can be an example	Not required	Uses non-custom & audited Github actions	Similar to previous iteration
A09:2021-Security Logging and Monitoring Failures	About user activity logging	Not required	Not required	Not required
A10:2021-Server-Side Request Forgery	Relevant when data is fetched from servers & data integrity	Not required	Not required	Not required

Table 5.1: OWASP Top 10 for artifact's security evaluation

5.0.4 Technologies used

The technologies or technical tools used in the artifact are based on Github, commonly known as a software version controlling system.

Github

Github is a platform that enables software development teams to collaborate, share code and operate general software development of an artifact. Github enables collaboration and awareness and, as social computing technology, shifts the focus of interaction to individual contributors and their activities with electronic artifacts (Dabbish et al. 2012).

Pros and Cons of Github

Github comes with several pros:

- Enables ease of communicating with other participants and contributors in a transparent manner,
- Enables to provide badges, stars, and related reward mechanisms,
- Enables automation of badge rewards via Github Actions,
- Enables a layer of security for the repositories and integrity of badges,

Github cons:

- Complexity of process to setup and configure participant's environments,
- Poor user interface, when compared to web-based interactive,
- Limited features to add, compared to websites, e.g., challenging to have alerts, reminders or interactive pages in Github readme docs.

Programming Languages:

The major two types of programming language used to build the artefact:

- Configuration programming languages: Markdown and YAML.

Markdown is commonly used to write documentation, sometimes build web applications (in server-side rendered pages), and several other documentation-related use-cases.

YAML (YAML Ain't Markup Language) is used used to write the script for automation with-in the artifact. Workflows are defined in the `.Github/workflows/` directory and use YAML syntax, having either a `.yml` or `.yaml` file extension (Dabbish et al. 2012).

- Artifact building programming languages: This can be seen as in fig 5.1, where 96.3% of the artifact is JavaScript, 1.9% is CSS and 1.8% is HTML

The design implementation is done on Github and consumes features provided from Github to deploy the artifact. Moreover, as part of the web application development tasks to be built, the artifact has parts defined with HTML, React, and DHIS2-specific tools.

RESULTS AND ANALYSIS

This chapter covers the *results & analysis* of the artifact through several iterations, the significant challenges when building the artifact, and lastly, the takeaways from the project after the three iterations of the gamified application resource site.

6.1 ARTIFACT ITERATIONS

6.1.1 Iteration I:

The artifact (<https://app-resource-dev.vercel.app/>) was built based on inspiration from existing DHIS2 applications. Therefore, the initial considerations were:

- Artifact should be built with a step-by-step guidelines approach,
- Artifact should be motivating for participants to get engaged. The reason for using the docsify library is to have a familiar feel (with DHIS2 apps) to the artifact.

Challenges

The library used to build the artifact was docsify (<https://docsify.js.org/>), the same library used for some DHIS2 core applications. To build over the default page layout provided by the library (docsify) is not possible, especially to build interactive web applications that track individual participants' progress on the artifact, e.g., tracking progress on gamification.

Feedbacks

The feedback given on the artifact was from peer students and project supervisor, and the main concerns were security:

- The artifact did not have SSL certificate,
- The artifact did not have progress tracking for the participant; while working on the artifact

Requirements for iteration II

After the reviews on the artifact, the following points were taken to be done for the second iteration:

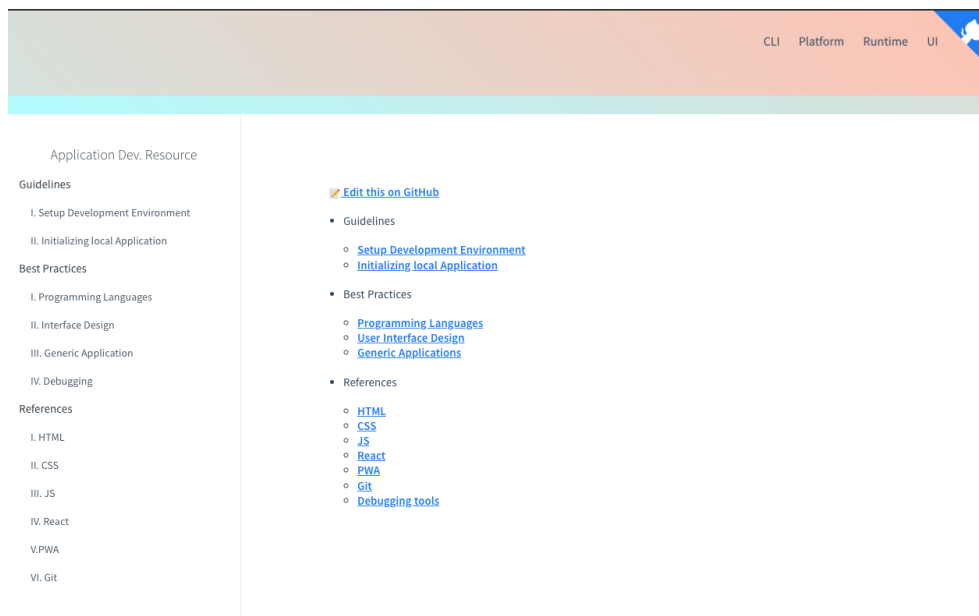


Figure 6.1: Artifact iteration 1

- The artifact should include gamification aspects
- The artifact should not save participants' credentials (username, password, or any information)
- The award of badges or rewards should be automated by the artefact

6.1.2 Iteration II

Challenges

The challenges of this iteration were probably the hardest, considering the requirements and the different parts to explore at the same artifact. Finally, Github was chosen to implement the requirements and to provide easier access for participants.

Feedbacks

For this last iteration of the artifact, there were new inputs from the DHIS2 core team regarding the readability of the tasks in the configurations of the artifact. After those changes were made, there were few willing participants to explore the artefact.

Requirements for Iteration III

- The artifact to have improved or readable content
- To explore the gamification aspect

6.1.3 Iteration III

Challenges

The challenges of this iteration were fixing the wording and improving the readability of the previous iteration.

Users explored the artefact in iteration III

After this iteration III, it can be seen in fig 6.2 the forks of the artifact, which means the accounts that attempted to demonstrate the gamified artifact, two people attempted it (jorgenpa and saifnoman). However, they did not manage to complete the tasks.

I waited for the accounts to work on some progress before sending them the survey and getting feedback from their end. However, until week 23/2022, there was no progress on their end; one of the users started configuring and working with the artifact (the initial steps), but the other user only forked it and did nothing. At last, however, I sent them the survey, but still no response.

6.2 TECHNICAL CHALLENGES

6.2.1 Artefact design:

In this thesis work, the artifact is built-in Github, which has dwarfed the artifact's gamification aspect by comparing Github to the game development frameworks. These shortages are:

Artifact Design:

The design of an artifact in IS is still elusive. It is a work in progress on approaching an artifact's design theory (A. Hevner and Chatterjee 2010). Therefore, building artifacts have design methods that open doors to many options or ways of building artifacts. This could make it harder to integrate the features with existing architectures within platforms.

Thus, design theories of both building artifacts and game design are necessary requirements for a gamified application-development resource site. Fig 6.2 shows the six traits of an artifact from the design theory of an artifact in DSR (A. Hevner and Chatterjee 2010), and having such components from the early stages of the artifact can structure the artifact itself, especially in the early stages of the artifact iterations.

Component	description
Core components	
1. Purpose and Scope	"What the system is for", the set of meta-requirements or goals that specifies the type of artifact to which theory applies
2. Constructs	Representation of the entities of interest in the theory.
3. Principle of form And function	The abstract "blueprint" or architecture that describes an IS artifact, either product or method/intervention.
4. Artifact mutability	The changes in state of the artifact anticipated in the theory, that is, what degree of artifact change is encompassed by the theory
5. Testable propositions	Truth statements about design theory
6. Justificatory knowledge	The underlying knowledge or theory from the natural or social or design sciences that gives a basis and explanation for the design (kernel theories)

Figure 6.2: Six components of an information system design theory (A. Hevner and Chatterjee 2010)

Software architectures of artefact:

In addition to the artifact design, the software architecture of the artifact needs careful consideration so that it does not have limitations for possible future requirements. The common use-cases of Github are hosting source codes of software, and the building process of those software follows recommended architectural patterns and practices.

Nevertheless, using the Github repository as an artifact becomes challenging in the case when newer requirements come on a given artifact. For instance: a challenge can happen in the artifact if there come new requirements for gamification integration into application development resource sites to include integration with other reward mechanisms like images or characters. Here, the software architecture challenges can be considered with tooling challenges,

Tooling challenges

- **Github:** Github is commonly used for hosting, sharing and management of software code, thus to use Github repository as an artifact might be challenging for users considering the importance of accessibility of web based applications.
- **Hosting environment:** To Hosting environment of the artifact from UiO, this could for instance be an access to Google Cloud web hosting included with the UiO student's Google package. Another option could be to leverage the services used by any of the Cloud computing services.

On the practical or technical preferences, the following design specifications come as helpful areas for having artifacts with PWA behaviours:

- Having requirements ready before starting designing,
- Considerations on architectures to have artifacts with rooms for newer features,
- Considerations to deploy artifacts, and which metrics of artifact performance to use,
- Consideration on testing of an artifact and access to testing environments,

Gamification challenges:

There are many challenges in gamifying a process, as seen at 2.4.3. Similarly, when looking at specially within OSS platforms, many incentive mechanisms can experiment:

- **Incentive mechanisms:**
Gamifying a process requires identifying the incentive mechanisms for the participants of a given process, by which the mechanisms can be industry or use-case-dependent. Moreover, the challenge increases for crowdsourced projects such as open-source projects are associated with incentive mechanisms for the public (participants). For instance, mention some examples of incentive mechanisms in OSS, and the incentive mechanism seen in Roberts, Hann and Slaughter 2006 shows that.

Social:

Even though the responses from DHIS2 core or those in closer proximity was faster, the social challenges with DHIS2 has been on asking questions in the DHIS2 community, for instance: slow responses on a post I put to request for possible participants to help with the thesis project. Slow responses are common in Open-source projects, therefore, optimal communication channels or access can be preferable approach instead of commenting in forums or community pages.

6.2.2 Challenges and shortages of the research process

Github for data collection

Software projects and development teams in software development platforms like Github and Gitlab usually understand *Stars* as a sign of the popularity of a project or as an approval rate of the project from the development community. There could be different reasons why *Github stars* and *forks* are essential. However, for new projects in Github (Gitlab or any similar source-code management platforms) *Stars* can be taken as incentive mechanisms for new members to join the popular project (for existing developers to work on the project).

During the second iteration of the artifact, *Stars* were considered as rewards (part of the incentive mechanism) for the individual users' accomplishments, but after discussing with the project supervisor, such an approach had two possible shortages:

- the users can be many, and it could be too much work for someone to check the completed milestones and awarded a stars on successful completed participants' repositories,
- after discussing the use of *stars*, the usage of badges became a better option as they convey the message better

The number of *stars* for the artifact (iteration three) or any completed participants' repositories were zero, which meets one of the hypothesis expectations of the artifact (iteration two). Nevertheless, it still begs questions about whether participants even considered the use of stars and whether the few numbers of participants had an effect on identifying *stars* as effective incentives.

- The Github repository (in iteration three) had 0 *stars* and 2 participant *forks*, 1 demonstration *fork* and another 1 demostration *fork*.

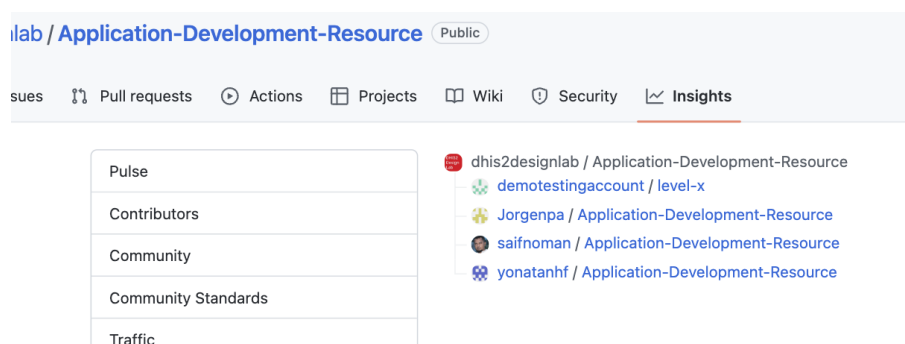


Figure 6.3: Artifact forks

As it can be seen in fig 5.1, there is zero amount of *stars* for the artifact could be an indication of a low rating of the artifact, different understanding of *stars* or low interest of *stars* as incentive mechanisms for participants. The natural and practical meaning of “starring a project” was never the subject of an in-depth and well-founded empirical investigation (Borges and Valente 2018).

Therefore, in this project, one of the challenges when using Github to collect data has been identifying effective rewards and whether *stars* or *badges* were motivating enough to be used as rewards.

Identifying (finding) specific focus group

A Focus group is one of the ways data can be collected and used to investigate new ideas that are widely used in many research fields. Using focus groups in design science research poses exciting opportunities and challenges (A. Hevner and Chatterjee 2010). In a focus group, participants are asked their opinions on a product, idea, or concept, and an interactive group setting allows for free discussion between the participants (Tremblay, A. R. Hevner and Berndt 2010).

One of the shortages of this study is, therefore, not having a focus group. Every interpretation of results is from communications with individuals and responses from either other peer students, the project supervisor, or the DHIS2 core team. They all have checked the artifact at a particular time and provided valuable inputs, and the inputs can be taken as unstructured interviews.

Burner Github accounts

After iteration III, the users' attempts and the challenge of follow-up due to Github's inability to verify users, it is problematic to take Github responses as legit data unless the user's reputation is also thighed with Github.

DISCUSSIONS

This chapter builds on the previous chapter on the research & analysis of the artifact. It discusses the project (thesis work) by addressing the research question, literature reviews of a digital platform, gamification, and web application developments. Moreover, the essential findings, how those findings were extracted, and the shortages of the study are discussed. Lastly, a few short recommendations were made by considering this study.

Research Question: How do aspects of gamification and documentation guidelines pertain to building web applications in a software development platform?

The research question at its core attempts to explore ways to motivate (indirectly re-enforce) Webapps builders via gamification. These ways are possible step-by-step guidelines, standard practices, or set-of essential parts, which are enough to enable builders to work with each other, debug problems or even build over previous builders' works. Having documenting every function is not feasible (Kirk, Roper and Wood 2007); moreover, documenting every component (functions) is impractical to adapt to the constantly changing patterns of programming languages, the increase in complexity of Webapps and the release of newer development frameworks.

The research question comprises three major parts: *applications & platforms (2.2)*, *documentation guidelines & web-application development (2.3)* and *gamification(2.4)*. Thus, to explore the three parts, initially, a hypothesis was setup to explore them, viz. taking DHIS2 as a digital platform (2.2), different types of web-applications (2.3), and then finally the research question takes the following hypotheses:

- Step-by-step guidelines (Procida 2020) are preferred ways of conveying the guidelines' content
- Gamification (2.4) can motivate web-application developers to follow guidelines

The approach taken was to explore the hypotheses by building an artifact, and the artifact that is built in this project has three iterations. The first iteration (6.1.1) explores the first hypothesis that step-by-step guidelines are practical for conveying message guidelines. The other two iterations consider the feedback from the first iteration and the hypothesis that gamification motivates developers. In parallel, in this process literature study on gamification and the possibility of designing gamified environment via Github was done.

Before building the web-page to explore an effective means to convey a message in guidelines, I had to look into the existing DHIS2 platform's web-pages that have documentation, for instance: the DHIS2 developer portal, DHIS2 UI and Design approach, and DHIS2 Application platform (3.1).

Those DHIS2 core applications use the same library (tool) called docsify (4.0.2) for displaying content for developers or users, and I decided to use the same library as it was fit enough to display the guidelines. Later after talking with the core team, step-by-step guidelines were taken as a preferred means to have the messages in the guidelines.

The hypothesis of gamification was initially brought into discussion with the project supervisor. The idea is that there could be many ways for builders to get motivated to follow step-by-step guidelines, e.g., many gamified online resources for programming or Webapps building exist. Therefore, among many other options, gamification possibly motivates developers to follow the first hypothesis of step-by-step guidelines.

7.1 FINDINGS

7.1.1 Step-by-step guidelines

Developer guidelines that are only with the important steps follow (Kirk, Roper and Wood 2007), and are motivating enough for builders (Graziotin et al. 2018, França, Da Silva and Sharp 2018) to follow comes as the optimal. These steps can be different according to the content; for instance, steps to configure specific settings need to be detailed more than the details needed for building a web-application because configuration steps are usually sensitive to jumps or mistakes in steps.

The project uses only a step-by-step procedure from the first iteration 6.1.1. They are effective because it is the commonly used approach in most developer guidelines of libraries or frameworks. Similarly, the feedback on the artifact did not have issues with those steps or the content. The negative feedback on the first iteration was mainly security concerns for missing SSL certificates and participants' privacy 5.0.3. In addition, the artifacts did not have any approach to gamification.

The feedback on the second iteration of the artifact 6.1.2 from the core-team was considering the readability of the artifact's configuration page of the step-by-step guidelines, still not about the step-by-step procedures of building the demo tasks. The takeaway is step-by-step guidelines approach is the optimal approach. However, when it comes to configuration or setup guides, the steps to put in the developer guidelines need additional supportive information.

Github repositories are reactive to several features, e.g.: forking, starring, git operations, automating process and many other features. In the iterations II & III, the readme part of the repository is used to display the step-by-step tasks, and the participant reads those steps and interacts with the Github features. However, Github has also limitations when it comes to accessibility. Therefore, interactive site generators like *readme (readme.com)* could be better options for gamified application-development-resource sites. With interactive resource-sties, developers can:

- Have their rewards (badges, stars or game ranks) in their profiles,
- Save the status of completed tasks, add their notes on the task or recommend possibly better options for doing the tasks,
- Can be handy to add new features, game characters and stories to the application-development site,
- Can have customisable UI to increase accessibility of the site

7.1.2 Gamification for motivation to follow guidelines

Gamified scenarios are many in our daily lives, plus many people these days have grown up with playing games, and a lot of people feel a sense of accomplishment winning games and it brings back fun memories for game players. Games are not limited to video games and they come in different formats which still adults can use them to feel the sense of accomplishment and get rewarded for excelling at doing tasks or reaching goals. 2.4

The gamification aspect of the artifact is visible in iterations 2 & 3 of the artifact, and practically there was mere reference on using Github features to demonstrate the gamification parts: aesthetics, dynamics and mechanics (Zichermann and Cunningham 2011). However, looking into those parts of gamification in contrast to how they are explored in the artefact:

- Aesthetics to handle to how a participant feels, when badges are awarded and would he/she continue for more badges,
- Dynamics to handle to the process of forking, and configuring the artifact (GitHub repository)
- Mechanics to handle the game features of levels, challenges in the form of step-by-step manner

Looking into the gamification, the design used (Zichermann and Cunningham 2011), and the use of the Github features could be debatable, and looks further exploration if Github doesn't fit into the game design explored in the artifact. Moreover, the following points can be incorporated to explore the design further with Github:

- **Adding stories to the Mechanics part**

Stories play vital roles in video games and generally how games are modelled, because stories convey messages and keep game participants engaged and motivated to pursue goals. Therefore, building a storyline by following gamification designs, the artifact is built. In the artifact the storyline can be equivalent to the step-by-step guidelines of building web-applications and the rewards (badges) of the tasks from the gamified environment.

At this moment the aesthetics part of the gamification (step-by-step guidelines) do not have stories, except step-by-step guidelines. Creative stories are vital for gamified environments to convey clear messages for participants so that the journey of learning the software development becomes more memorable. Therefore, exploring more into stories and aesthetics part of the game can increase the engagement of participants. Stories in gamification can include certain characters or fictitious story so that to emotionally engage the participant. Participants do not have characters to pick in the current form, but exploring stories further can add value into the aesthetics part of the game

- **Open-source incentive mechanism** The digital platform which is considered for this project is an open-source, therefore, the artifact considers open-source projects developers as participants. In general, majority of the game industry has finances as incentives, and it is commonly known opinion that open-source project developers are not interested in financial gains from the projects they build in public software management platform like Github. Therefore, one of the challenges is what can incentivise best for those various open-source project builders to get engaged and play around with the artifact.

Knowing the incentive mechanism could require extra research into the given demography of developers, then the gamification aspect of the artifact, especially the mechanics and aesthetics parts can be tuned to meet the selected incentives.

7.1.3 Limitations of the study

To combine the three parts: step-by-step guideline, gamification and building the artefact into a single artifact and have the research via DSR was challenging, specially the gamification design and artefact building aspect. The book on DSR by A. Hevner and Chatterjee 2010, provides ample resources and several historical scenarios into DSR, which at times was confusing me during the first times, starting to understand how the iterative behaviour gets feedback from participants.

In addition to that having to use the book on Game Design by Zichermann and Cunningham 2011 and meeting the requirement for the three parts of game design (aesthetics, mechanics and dynamics) into an artefact was hard. At last, I chose Github as an environment to enable those three design requirements of a game.

But Github has its own challenging aspects (5.0.4, 6.2.2):

- Burner accounts can participate and flood the data feedback
- Hard to add features in Github, for instance: if participants want different rewards outside than stars or badges, then it doesnt work
- Developers who have not experience with Github before can be excluded from using the artefact

CONCLUSION

This thesis work explored a gamified application development resource site, where applications are Webapps. With modifications, the gamified application development resource site can be a valuable part of DHIS2 to guide developers with clear and essential steps. With the addition of supportive information to configuration or setup guides, having a gamified application development resource site works when it comes to the step-by-step approaches. The additional information can be by adding videos to describe (or show) the step, hosting live workshops to explore the steps, and probably adding language preferences on the artifact can enhance the use of the step-by-step guidelines.

In addition, in most of the documentation libraries in DHIS2 is docsify, one of the top two trending libraries (per npm-trends list, <https://www.npmtrends.com/docsify-vs-docusaurus-vs-gitbook-vs-vuepress>). The advantage of static-site generators over interactive-site generator is speed of delivery and fewer features to implement, however, this makes them hard to have interactive environment for instance: to add gamification or user to save his/her status. Moreover, docsify have limitations to customisation and plug-in extensions, that is also the main reason why the decision was made to use Github for gamification environment. However, for a gamified application development resource site in DHIS2, interactive sites are feasible and more effective for gamification.

From the literature, it can be concluded that gamified guidelines pertain to web-application building do give a sense of accomplishment and a sense of achievement for the participants in a playful, challenging learning experience and at last getting rewarded for reaching certain milestones in the learning curve of the provided step-by-step guidelines.

In addition to that, adding video or animations to describe the complicated parts of the gamification can increase accessibility, for some developers have preferences to watch videos rather than follow only the documentations (different means of learning new ways). For instance, this can be attained by hosting videos tutorials or adding the videos parallel to the to-do task.

8.1 FUTURE WORKS

In the current gamified application development site, there is only one character for all the participants: the character of an application developer (builder), which can be assumed as the hero of the gamified environment. In the future, more character additions can be made by using the exact roles of the single character in building web applications (2.3) while maintaining or following step-by-step guidelines. Therefore having multiple characters and storylines for the different communities can be a research venue for more engaging outcomes. Furthermore, stories can be added to the gamification aspect, and each user is a character in the story.

Digital platforms, they have their own ecosystems and integrating two or more platforms requires common stories, standards, values and incentive mechanisms (in the context of gamification). Open-source platforms are challenging as participants of such platforms are willingly present in the platforms with a will to contribute to the common good of the community, therefore the merger between such platforms can be easier than otherwise.

8.1.1 Gamification frameworks (platforms)

Using Github as a gamification environment could have dwarfed the participation or use of the artifact, therefore it could be better to use gamification platforms like cocos2d-x (<https://github.com/cocos2d/cocos2d-x>), and others, because they are dedicated platforms and come with many gamification principles to implement, test and deploy faster. In addition to that, by using those platforms, artifacts can be built with several scenarios, to mention some, it could be: characters, game modes, maximum (minimum) number of participants to have, scalability and hardware requirements.

8.1.2 Exploring design patterns inside step-by-step guidelines

Step-by-step guidelines is one time process to achieve a given goal, but if the focus is on the design patterns users can return to do the tasks multiple times, thus increasing

8.1.3 Gamifying bottom-up

When gamifying an environment, the need for an organic network-effect mainly depends on the values and value creation processes relevant to the game environment participants. Based on existing models of digital platforms, the value created is similar and researches have shown many digital platforms are ditching the vanity metrics of value in search of new ways to look into value creation. Bottom-up approach to gamify is indeed perceived as valuable because people have higher autonomy compared to “top-down” gamification often used today’s games (Lessel et al. 2016). Therefore, an approach to gamify a web-application development resource site is to look into gamifying an environment by initially researching or analysing the users’ (or open-source communities) incentive mechanics, values and value creation processes, then build stories around their responses, characters, and include game mechanics.

8.1.4 Glimpse into the Metaverse

Looking into the technological trends of digital platforms in 2020-22, the concepts of Metaverse and migration of such traditional digital platforms like Meta (previously Facebook), Apple, Microsoft, Amazon, Snapchat, NVIDIA calls indicates the evolution of the commonly used client-server architectures to decentralised digital platforms. By definition, metaverse is a combination of the prefix “meta” (implying transcending) with the word “universe”, describes a hypothetical synthetic environment linked to the physical world, it serves as layer that provides singularity and integrations between several modern day terms: *virtual economy, avatars, NFTs, VR, XR, AR, MR, AI and Automation* (Mystakidis 2022).

Looking into gamification, players and users are constantly reminded that the current version of the metaverse is not theirs to grow and build upon, but owned by those creating the experiences for them

(Chainlink-docs 2022). Therefore, the metaverse is not still better environment than the traditional client-server digital platforms, but there could be take aways from the gamified environments in the metaverse. Moreover, there exist several initiatives for future works on integrating, inter-operating or migrating current traditional client-server based digital platforms into the metaverse and vice-versa seems an inevitable journey to maximize the value created from both architectures (Chainlink-docs 2022).



GLOSSARY

Github: A cloud-based service which is used by developers to share, store, automate work-flows and integrate software resource-codes. Equally important, for teams, it helps to track changes that are made by developers. Some common inter-changeable terms for Github are version control or Git.

Upstream: When talking about a branch or a fork, upstream is the primary branch on the original repository. (src: Github docs)

User/s: Users are people with personal GitHub accounts. Each user has a personal profile, and can own multiple repositories, public or private. (src: Github docs)

Artefact/s: Artefacts can be of various types (requirements, use cases, design documents, use case diagrams), which have customisable attributes and data types. (src: IBM docs)

PWA: Progressive Web Apps (PWAs) are web apps that use service workers, manifests, and other web-platform features in combination with progressive enhancement to give users an experience on par with native apps. (src: Mozilla mdn docs)



BIBLIOGRAPHY

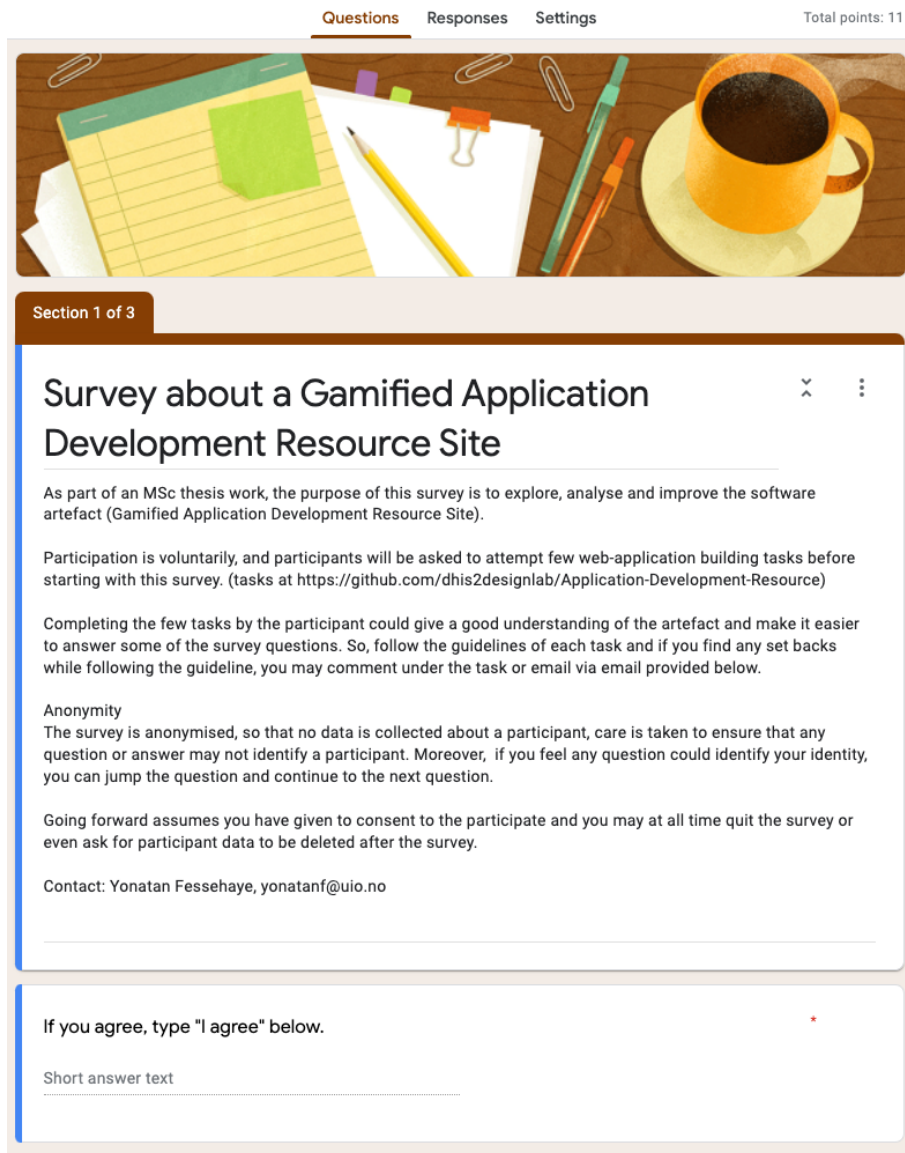
1. Farrukh Shahzad. 'Modern and responsive mobile-enabled web applications'. In: *Procedia Computer Science* 110 (2017), pp. 410–415.
2. Douglas Kirk, Marc Roper and Murray Wood. 'Identifying and Addressing Problems in Object-Oriented Framework Reuse'. In: *Empirical Softw. Engg.* 12.3 (June 2007), pp. 243–274. ISSN: 1382-3256. DOI: 10.1007/s10664-006-9027-z.
3. Daniele Procida. *Divio Documentation System: how-to-guides*. <https://documentation.divio.com/>. 2020.
4. Jinwoo Kim and F Javier Lerch. 'Why is programming (sometimes) so difficult? Programming as scientific discovery in multiple problem spaces'. In: *Information Systems Research* 8.1 (1997), pp. 25–50.
5. Jochen Rode. 'Nonprogrammer web application development'. In: *CHI'04 extended abstracts on Human Factors in computing systems*. 2004, pp. 1055–1056.
6. Eric Adu-Gyamfi, Petter Nielsen and Johan Ivar Sæbø. 'The dynamics of a global health information systems research and implementation project'. In: *SHI 2019. Proceedings of the 17th Scandinavian Conference on Health Informatics, November 12-13, 2019, Oslo, Norway*. 161. Linköping University Electronic Press. 2019, pp. 73–79.
7. Allen S Lee. 'Thinking about social theory and philosophy for information systems'. In: *Social theory and philosophy for information systems* 1 (2004), p. 26.
8. Amrit Tiwana. 'The Rise of Platform Ecosystems'. In: *Platform Ecosystems* (2014), pp. 3–21.
9. OSI-Docs. *Open Source Initiative*. <https://opensource.org/>. 2007.
10. Joel West. 'How open is open enough?: Melding proprietary and open source platform strategies'. In: *Research policy* 32.7 (2003), pp. 1259–1285.
11. Mark De Reuver, Carsten Sørensen and Rahul C Basole. 'The digital platform: a research agenda'. In: *Journal of Information Technology* 33.2 (2018), pp. 124–135.
12. Carla Bonina et al. 'Digital platforms for development: Foundations and research agenda'. In: *Information Systems Journal* 31.6 (2021), pp. 869–902.
13. Fabian Fagerholm and Jürgen Münch. 'Developer experience: Concept and definition'. In: *2012 international conference on software and system process (ICSSP)*. IEEE. 2012, pp. 73–77.

14. Lars Kristian Roland et al. 'P for Platform. Architectures of large-scale participatory design'. In: (2017).
15. Chainlink-docs. *Blockchain, Metaverse, DeFi*. <https://chain.link/education/>. 2022.
16. Yan Chen, Igor Pereira and Pankaj C Patel. 'Decentralized governance of digital platforms'. In: *Journal of Management* 47.5 (2021), pp. 1305–1337.
17. Vitalik Buterin et al. *Ethereum: A next-generation smart contract and decentralized application platform*. 2014.
18. Chris Cordle. *Why Angular 2/4 Is Too Little, Too Late*. 2017.
19. Daniel Graziotin et al. 'What happens when software developers are (un) happy'. In: *Journal of Systems and Software* 140 (2018), pp. 32–47.
20. César França, Fabio QB Da Silva and Helen Sharp. 'Motivation and satisfaction of software engineers'. In: *IEEE Transactions on Software Engineering* 46.2 (2018), pp. 118–140.
21. W3C. *Web Content Accessibility Guidelines*. <https://www.w3.org/TR/WCAG21/#background-on-wcag-2/>. 2018.
22. Gabe Zichermann and Christopher Cunningham. *Gamification by design: Implementing game mechanics in web and mobile apps*. " O'Reilly Media, Inc.", 2011.
23. Rania Elshiekh and Laurie Butgerit. 'Using gamification to teach students programming concepts'. In: *Open Access Library Journal* 4.8 (2017), pp. 1–7.
24. Sebastian Deterding et al. 'Gamification. using game-design elements in non-gaming contexts'. In: *CHI'11 extended abstracts on human factors in computing systems*. "Unknown", 2011, pp. 2425–2428.
25. Firas Layth Khaleel et al. 'The study of gamification application architecture for programming language course'. In: *Proceedings of the 9th International Conference on Ubiquitous Information Management and Communication*. 2015, pp. 1–5.
26. Richard Whiddington. 'Virtual Tours and Gamification, China's Museums Pivot Content for Coronavirus'. In: *Jing Travel, February* 14 (2020).
27. Iulian Furdu, Cosmin Tomozei and Utku Kose. 'Pros and cons gamification and gaming in classroom'. In: *arXiv preprint arXiv:1708.09337* (2017).
28. Gabriela Kiryakova, Nadezhda Angelova and Lina Yordanova. 'Gamification in education'. In: *Proceedings of 9th International Balkan Education and Science Conference*. 2014.
29. Valērija Platonova and Solvita Bērziša. 'Gamification in software development projects'. In: *Information technology and management science* 20.1 (2017), pp. 58–63.
30. Armando M Toda, Pedro HD Valle and Seiji Isotani. 'The dark side of gamification: An overview of negative effects of gamification in education'. In: *Researcher links workshop: higher education for all*. Springer. 2017, pp. 143–156.
31. Michael D Hanus and Jesse Fox. 'Assessing the effects of gamification in the classroom: A longitudinal study on intrinsic motivation, social comparison, satisfaction, effort, and academic performance'. In: *Computers & education* 80 (2015), pp. 152–161.
32. Anastasia Dimitriadou et al. 'Challenges in serious game design and development: Educators' experiences'. In: *Simulation & Gaming* 52.2 (2021), pp. 132–152.

33. UiO. *DHIS2 Design Lab*. <https://www.mn.uio.no/ifi/english/research/networks/hisp/dhis2-design-lab/>. 2020.
34. Alan Hevner and Samir Chatterjee. 'Design science research in information systems'. In: *Design research in information systems*. Springer, 2010, pp. 9–22.
35. Alan R Hevner et al. 'Design science in information systems research'. In: *MIS quarterly* (2004), pp. 75–105.
36. Jan vom Brocke and Alexander Maedche. 'The DSR grid: six core dimensions for effectively planning and communicating design science research projects'. In: *Electronic Markets* 29.3 (2019), pp. 379–385.
37. John Venable, Jan Pries-Heje and Richard Baskerville. 'FEDS: a framework for evaluation in design science research'. In: *European journal of information systems* 25.1 (2016), pp. 77–89.
38. Mike Cohn. *User stories applied: For agile software development*. Addison-Wesley Professional, 2004.
39. Timothy Kinsman et al. 'How do software developers use GitHub Actions to automate their workflows?' In: *2021 IEEE/ACM 18th International Conference on Mining Software Repositories (MSR)*. IEEE. 2021, pp. 420–431.
40. Andrew van der Stock. *OWASP Top Ten*. <https://owasp.org/www-project-top-ten/>. 2021.
41. Laura Dabbish et al. 'Social coding in GitHub: transparency and collaboration in an open software repository'. In: *Proceedings of the ACM 2012 conference on computer supported cooperative work*. 2012, pp. 1277–1286.
42. Jeffrey A Roberts, Il-Horn Hann and Sandra A Slaughter. 'Understanding the motivations, participation, and performance of open source software developers: A longitudinal study of the Apache projects'. In: *Management science* 52.7 (2006), pp. 984–999.
43. Hudson Borges and Marco Tulio Valente. 'What's in a GitHub star? understanding repository starring practices in a social coding platform'. In: *Journal of Systems and Software* 146 (2018), pp. 112–129.
44. Monica Chiarini Tremblay, Alan R Hevner and Donald J Berndt. 'The use of focus groups in design science research'. In: *Design research in information systems*. Springer, 2010, pp. 121–143.
45. Pascal Lessel et al. "' Don't Whip Me With Your Games" Investigating" Bottom-Up" Gamification'. In: *Proceedings of the 2016 chi conference on human factors in computing systems*. 2016, pp. 2026–2037.
46. Stylianos Mystakidis. 'Metaverse'. In: *Encyclopedia* 2.1 (2022), pp. 486–497.

APPENDIX-

This chapter hold the supplementary materials, viz. survey document.



The screenshot shows a web-based survey interface. At the top, there are navigation tabs for 'Questions', 'Responses', and 'Settings', with 'Questions' being the active tab. On the right side of the top bar, it says 'Total points: 11'. Below the navigation is a decorative header image featuring a desk with a yellow notepad, a pencil, a pen, and a cup of coffee. The main content area is titled 'Section 1 of 3' and contains the following text:

Survey about a Gamified Application Development Resource Site

As part of an MSc thesis work, the purpose of this survey is to explore, analyse and improve the software artefact (Gamified Application Development Resource Site).

Participation is voluntarily, and participants will be asked to attempt few web-application building tasks before starting with this survey. (tasks at <https://github.com/dhis2designlab/Application-Development-Resource>)

Completing the few tasks by the participant could give a good understanding of the artefact and make it easier to answer some of the survey questions. So, follow the guidelines of each task and if you find any set backs while following the guideline, you may comment under the task or email via email provided below.

Anonymity
The survey is anonymised, so that no data is collected about a participant, care is taken to ensure that any question or answer may not identify a participant. Moreover, if you feel any question could identify your identity, you can jump the question and continue to the next question.

Going forward assumes you have given to consent to the participate and you may at all time quit the survey or even ask for participant data to be deleted after the survey.

Contact: Yonatan Fessehaye, yonatanf@uio.no

Below the text is a form field with the instruction: 'If you agree, type "I agree" below.' and a red asterisk indicating a required field. The form field is labeled 'Short answer text' and has a dotted line indicating the input area.

Questions Responses Settings

Total points: 11

(web-application building)

Description (optional)

1. You may have heard/used one or more of the following options below which are used to build web-applications. Which of those languages (framework) do you feel are familiar?

	I have heard of	I have used with	Never heard of	Never used
HTML	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
JavaScript	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
CSS	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
ReactJS	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

How do you feel about playing games: tabletop, video, educational, puzzle or any type of a game at all?

- I really enjoy games
- I enjoy games moderately
- Sometimes, I enjoy games
- I dont enjoy games
- I hate games

After section 2 Continue to next section



Section 3 of 3

About the tasks from web-application building resource



Tasks at <https://github.com/dhis2designlab/Application-Development-Resource> . The task groups in the resource can be grouped into: basic HTML tasks, React tasks and DHIS2 tasks

How do you feel about the clarity of the tasks?

- Not clear steps
- Normal, I could understand the steps
- Yes, very clear

Does it motivate to complete the tasks and earn a badges?

- Yes
- No



Which tasks were

	Easy to complete	Medium challenge	Hard to complete
HTML/JavaScript/CSS t...	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
React tasks	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
DHIS2 tasks	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Setting up the environme...	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

What steps would you like to see in the web-application building tasks?

Short answer text

.....

Questions Responses Settings

Total points: 11

Does it feel a sense of step by step flow of information while doing the tasks?

- Yes
- No

Which tasks did you feel like to:

Complete and earn... Not Complete and ... To see more in the ... To see less in the f...

HTML/JavaScript/...	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
React tasks	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
DHIS2 tasks	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>






How do you feel about building more web-applications by such process and earning badges for them?

- Positive, and prefer it
- Neutral
- Not positive, and I prefer other ways

How do you feel about showing the badges to others?

- Great, I like it
- Neutral, it is normal
- Bad, I don't like it

What type of reward measurement do you prefer in the badges?

- Numbered levels, e.g.: level 1, level 2, level 3, ...
- General competency levels, e.g.: Beginner, Intermediate, Advanced, ...
- Points amount, e.g. 50, 70, 21, or ..
- Gain more stars (gems), e.g. , , , , , ...

Do you prefer other types of badges? Maybe ____

- a printed paper (certificate)
- or only in Github
- a different labels or wording
- to show it in LinkedIn, or other professional social networks

Was the language of this survey easy to understand?

- Not much
- I could understand the message
- Yes, very clear

According to an open-source survey by Github, some of the problems encountered in open-source projects (in Github) are listed below. Which one/s do you feel could be more gamified?

- Incomplete or confusing documentation
- Unresponsiveness
- Dismissive responses
- Conflict
- Unexplained rejection
- Unwelcoming language (content)