Enhancing Java learning through gamification: a web-based application

orcid.org/ 0000-0001-9885-0131

JT Sithole

Mini-Dissertation accepted in fulfilment of the requirements for the degree Master of Commerce in Informatics with Information Systems at the North-West University

Supervisor: Prof K Gorejena
Co-Supervisor: Mrs ME Nhlapo

Graduation: October 2026

DECLARATION

I hereby declare that this research mini-dissertation, titled "Java learning through gamification: A web-based application," is my original work and has not been previously presented in whole or in part for any degree or other qualification at North-West University or any other institution. All sources of information used or quoted have been duly acknowledged and referenced.

I have read and understood the university's policy on plagiarism and academic integrity and I confirm that this research paper conforms to all relevant ethical standards, including the protection of participant confidentiality as stipulated by the ethics committee.

DEDICATION

I dedicate this study to my parents and family who have been my anchor. Your unwavering belief and support in my potential even when the path seemed insurmountable has been the fuel for this journey. Thank you for the sacrifices you made during this journey.

To the students who struggle in silence with complex code: this work is dedicated to you. May this research contribute even in a small way to making the journey of learning to program a little less lonely and a lot more engaging.

Finally, to the almighty God, thank you for the strength, wisdom and perseverance to see this through to the end.

PREFACE

The journey of learning to program is often compared to learning a new language, but with an unforgiving syntax that tolerates no errors. For many Information Technology students, Java represents a significant hurdle—a module characterised by high failure rates, cognitive overload and often, a loss of motivation. The traditional "death by PowerPoint" approach has struggled to bridge the gap between theoretical concepts and the practical, hands-on resilience required to master code.

This study explores the intersection of gamification and computer science education. It stems from a desire to transform the learning experience from a passive struggle into an active, engaging quest. By applying design science research, this study does not merely theorise about a solution; it builds one.

JavaRealm was born out of the belief that learning should be rigorous but not rigid. Through this research, we navigate the complex balance between "fun" and "function," exploring how game mechanics like skill trees, experience points and leaderboards can be integrated with professional coding tools to create an environment where students feel safe to fail and motivated to succeed.

This dissertation documents the lifecycle of that artifact—from the initial frustrations voiced by students to the deployment of a functional web application designed to help them master the source code.

ACKNOWLEDGEMENTS

Completing a study of this magnitude is never a solitary endeavour. I would like to express my deepest gratitude to my supervisory team for your expert guidance, patience and constructive criticism throughout the process. Your mentorship pushed me to refine my ideas and ensured that JavaRealm became more than just an app but a scientifically grounded artifact, To the undergraduate and postgraduate IT students at North-West University who volunteered their time for questionnaires and focus groups. Your honest feedback both the praise and the criticism was the heartbeat of this study and my friends and colleagues for the late-night coding sessions, the debugging help and the moral support when the deadlines loomed large.

ABSTRACT

The challenge of student attrition and lack of engagement in introductory programming courses is a global concern in higher education. Java, specifically, presents a steep learning curve due to its strict syntax and abstract concepts, often leading to cognitive overload and demotivation among novice programmers. This study investigates how gamification can be leveraged to address these challenges by designing and developing a web-based learning application tailored to the needs of IT students.

Adopting a design science research (DSR) methodology, the study followed a rigorous cycle of problem identification, objective definition, design, development, and evaluation. Data was collected from information technology students at North-West University (NWU) through quantitative questionnaires (N=20) and qualitative focus groups (N=6). The findings revealed that students desire authentic practice over superficial "toy" games, with a strong demand for immediate feedback (100% agreement), professional-grade tools (simulating Visual Studio Code), and "safe" social competition where leaderboards are opt-in to reduce anxiety.

Based on these empirical insights, the study developed JavaRealm, a web application utilizing the MERN stack (MongoDB, Express, React, Node.js). The artifact features a "Skill Tree" for visual progression, an authentic Monaco code editor for practical tasks, and a narrative-driven "Boss Battle" system to test mastery.

The design verification confirms that JavaRealm is built to translate established motivational and engagement principles into a functional software artifact, grounded in empirical student needs data and established educational theory. The study contributes to the field of educational technology by providing a practical blueprint for implementing gamification in complex cognitive domains, proposing, based on empirical design evidence, that effective gamification must prioritise scaffolding, authenticity, structured progression, and user autonomy over simple points and badges — a set of design principles ready for empirical validation in future longitudinal research.

Key terms: Gamification, Java programming education, design science research (DSR), student motivation, web-based application, educational technology, software development

LIST OF ABBREVIATIONS

API	application programming interface
Baas	backend-as-a-service
Cors	cross-origin resource sharing
Dsr	design science research
Fr	functional requirement
Hei	higher education institution
Http	hypertext transfer protocol
Https	hypertext transfer protocol secure
Ict	information and communication technology
Ide	integrated development environment
It	information technology
Jdk	java development kit
Llm	large language model
Lms	learning management system
M	mean (statistics)
Mern	mongodb, express.js, react.js, and node.js (technology stack)
N	sample size (total)
Nfr	non-functional requirement
Npm	node package manager
Nosql	not only sql (database)
Nwu	north-west university
Oo	object-oriented
Oop	object-oriented programming
Pbl	points, badges, and leaderboards
Popia	protection of personal information act
Ro	research objective

Sd	standard deviation
Sdlc	software development life cycle
Spa	single page application
Stem	science, technology, engineering, and mathematics
Tam	technology acceptance model
Ui	user interface
Uml	unified modelling language
Ux	user experience
Vs	visual studio code

TABLE OF CONTENTS

DECLARATION ..... **I**

DEDICATION ..... **II**

PREFACE ..... **III**

ACKNOWLEDGEMENTS ..... **IV**

ABSTRACT ..... **V**

LIST OF ABBREVIATIONS ..... **VI**

TABLE OF CONTENTS ..... **VIII**

LIST OF TABLES ..... **XIII**

LIST OF FIGURES ..... **XIV**

INTRODUCTION AND BACKGROUND TO THE STUDY **1**

1.1 Introduction **1**

1.2 Background to the study **2**

1.3 Problem statement **3**

1.4 Justification of the problem **3**

1.5 Research aim and objectives **5**

1.5.1 Research aim 5

1.5.2 Research objectives 5

1.6 Research questions **6**

1.6.1 Main research question 6

1.6.2 Secondary research questions 6

1.7 Literature overview **6**

1.8 Research methodology **7**

1.9 Significance of the study **7**

1.10 Scope **7**

1.11 Chapter division **8**

1.12 Chapter summary **8**

LITERATURE REVIEW **9**

2.1 Introduction **9**

2.2 Definitions **9**

2.3 Previous studies in gamified programming education **10**

2.3.1 Foundational applications and empirical evidence 10

2.3.2 Emerging trends and personalised approaches 10

2.3.3 Meta-analytic synthesis 11

2.4 The impact of gamification in education **12**

2.4.1 Early conceptual studies and general benefits 12

2.4.2 Empirical evidence and meta-analyses 12

2.4.3 Limitations and future directions 13

2.5 Key challenges in java programming education **14**

2.5.1 Inherent complexity and cognitive load 14

2.5.2 Limitations of traditional pedagogies and attrition rates 14

2.5.3 Methodological critiques and ethical considerations 15

2.6 Impact of gamification in programming education 16

2.6.1 Empirical evidence from programming contexts 16

2.6.2 Transferability and methodological considerations.....16

2.7 Gamification elements in education 18

2.7.1 Common gamification elements and their motivational mechanisms 18

2.7.2 Design considerations and ethical implications.....20

RESEARCH METHODOLOGY.....22

3.1 Introduction.....22

3.2 The design science research (DSR) methodology.....22

3.3 Research paradigm.....23

3.4 Research strategy.....23

3.5 Population and sampling strategy 24

3.5.1 Sample size 24

3.5.2 Inclusion and exclusion criteria.....25

3.6 Data collection instruments and procedure 25

3.6.1 Data collection instruments 25

3.6.2 Data collection procedure.....26

3.7 Data analysis.....27

3.8 Technical development methodology.....27

3.8.1 Frontend architecture (user interface).....27

3.8.2 Backend architecture (code execution engine) 28

3.8.3 Cloud infrastructure and data persistence 29

3.8.4 Development environment 29

3.9 Pragmatic rigour 29

3.10 Ethical considerations 30

3.10.1 Permission and informed consent 30

3.10.2 Anonymity and confidentiality 31

3.11 Chapter summary 32

DATA ANALYSIS, DESIGN AND IMPLEMENTATION 33

4.1 Introduction 33

4.2 Presentation of empirical findings 33

4.2.1 Demographic profile of participants 33

4.2.2 Quantitative findings (questionnaire analysis) 34

4.2.3 Qualitative findings (focus group thematic analysis) 36

4.2.4 Discussion of findings and literature integration 38

4.3 System requirements specification 38

4.4 System design and architecture 39

4.4.1 Technical architecture 39

4.4.2 Gamification mechanics design 40

4.5 Implementation of the artifact: JavaRealm 42

4.6 Chapter summary 52

CONCLUSION AND RECOMMENDATIONS 53

5.1 Introduction **53**

5.2 Evaluation of the artifact **53**

5.2.1 Verification against requirements 53

5.2.3 Heuristic usability evaluation 55

5.3 Conclusions **56**

5.4 Research contributions **57**

5.4.1 Practical contribution 57

5.4.2 Methodological contribution 58

5.5 Limitations of the study **58**

5.6 Recommendations **59**

REFERENCES **60**

ANNEXURE A: DATA COLLECTION TOOLS **1**

Questionnaire **1**

Focus Group Discussion Guide **5**

ANNEXURE B: INFORMED CONSENT **1**

Informed Consent Form - Questionnaire **1**

Informed Consent Form - Focus Groups **3**

ANNEXURE C: RECRUITMENT MATERIAL **1**

ANNEXURE D: JAVAREALM INSTRUCTION MANUAL **1**

LIST OF TABLES

Table 4.1: Demographic profile of participants (N = 20) 33

Table 4.2: Key quantitative findings 34

Table 4.3: Thematic analysis of focus group data 37

Table 4.4: Requirements traceability matrix 39

Table 5.1: Artifact verification matrix 53

LIST OF FIGURES

Figure 4.1 The NoSQL Cloud Firestore schema design 41

Figure 4.2 The landing page (Home Page) 42

Figure 4.3 The landing page (Home Page) code 43

Figure 4.4 The ‘World Map’ 44

Figure 4.5 Main layout and world map code 45

Figure 4.6 The coding interface 46

Figure 4.7 The 'mission complete' modal 47

Figure 4.8 The 'mission complete' modal code 47

Figure 4.9 Leaderboard page 48

Figure 4.10 Leaderboard page code 49

Figure 4.11 The profile page 50

Figure 4.12 Profile page code 51

Figure 4.13 Example of boss battle 52

INTRODUCTION AND BACKGROUND TO THE STUDY

1.1 Introduction

Java programming, with its syntactical and conceptual complexities, can be a challenging subject for students (Robins *et al.,* 2003). Gamification practices can also be integrated, and this approach has the potential to address the identified challenges and improve the learning process (Deterding *et al.,* 2011). This idea is supported by recent studies, which indicate that gamification is an effective tool for facilitating learning outcomes in various educational environments (Sailer & Homner, 2020; Huang *et al.,* 2020).

Java is one of the modern, widely used programming languages and is suitable as a mandatory introductory subject for many IT professionals (Medeiros *et al.,* 2019). The capacity to think computationally, solve structured and unstructured problems, and manipulate information systems underpins modern society's progress, shaping industries from healthcare to finance, from the creative arts to state infrastructure. Amid a landscape crowded with programming languages, tools, and platforms, Java remains one of the most impactful and widely adopted languages, holding a central position in both academic curricula and professional practice owing to its object-oriented paradigm, reusability, and cross-platform compatibility (Deitel & Deitel, 2017). Java is based on the principle of "write once, run anywhere," which provides developers with the ability to develop applications that run smoothly on different platforms (Deitel & Deitel, 2017). In constructing and utilizing objects in programming, students develop computational thinking, problem-solving skills, and an appreciation of technology. Students who study Java programming are equipped with the aptitude for jobs in computer science and other sectors that heavily rely on technology and data analysis.

Nevertheless, mastering Java can be a complex process and can be particularly challenging for beginners, as noted by Watson and Li (2014). Students encounter core concepts that are conceptually difficult yet necessary for further progress, such as inheritance, polymorphism, and algorithmic logic (Robins *et al.,* 2003). Furthermore, traditional lecture-based approaches often fail to provide a practical context for theoretical concepts, which hinders students' ability to apply what they have learned (Watson & Li, 2014). As observed by Romero *et al.,* (2015), these techniques may be passive rather than interactive, thus resulting in low motivation and retention of knowledge. Inability to master them contributes to long-term anxiety, disinterest, and withdrawal from computing courses in many instances (Watson & Li, 2014; Bergin & Reilly, 2006).

The emergence of gamification, which is the use of game-design elements and mechanics in non-game contexts, offers a promising alternative. Gamification can help learners improve knowledge acquisition and problem-solving abilities in the context of programming education (Zhan *et al.,* 2022). Accordingly, the efficient use of game mechanics integrated into an easily accessible online environment can foster a more engaging and productive learning environment for beginner Java coders (Subhash & Cudney, 2018). By layering points, achievements, feedback, challenges, and competition onto learning activities, gamification has demonstrated potential to intrinsically motivate students, foster deeper engagement, and support meaningful learning outcomes across diverse educational domains (Deterding *et al.,* 2011; Sailer & Homner, 2020).

This study aims to address the challenges in Java programming education by developing a gamified web-based application and utilizing game elements such as scores, badges, and leaderboards to improve student participation and interest in learning while enabling more effective learning. It follows a design science approach, improving the application through continuous testing and adjustments until it effectively serves educational activities. It addresses an important knowledge gap by examining how gamification improves Java learning while making new contributions to the study of programming education.

1.2 Background to the study

Programming education serves as an important bridge to the digital era, providing students with necessary skills in computer science (Medeiros *et al.,* 2019). Nonetheless, the conventional lecture-based methods commonly fail to keep the students interested resulting in frustration and poor learning performance (Watson & Li, 2014). Such reduced motivation is especially worrying because learners are struggling with advanced software (Landers, 2014). Educational technology has realized that gamification is a potential solution to these issues (Dicheva *et al.,* 2015).

Programming education is an essential key to knowledge economies and is one of the most important facilitators of job preparation, innovation and creative problem-solving, on a fundamental level. Java is a fundamental language among millions of programmers and it finds application in education, software development and enterprise solutions. Its merits are a well-defined syntax, a full ecosystem and the capability to be executed on any virtually device with a feature that make it a staple in the introductory and intermediate computer science courses of universities (Deitel & Deitel, 2017). Nonetheless, the instruction and education of Java are marred with a number of professional and pedagogical obstacles:

- Cognitive load: For beginners, the simultaneous mastery of new syntax, abstraction and logic can be overwhelming. Threshold concepts are often "stuck places" requiring sustained, contextual practice to overcome (Robins *et al.,* 2003).

- Pedagogical limitations: Lecture-based formats fail to provide the granular, real-time feedback and adaptive learning which is essential in programming. Passive instruction limits engagement and many students fail to see the relevance of exercises outside real-world contexts (Romero *et al.,* 2015).

- Motivational gaps: Research has consistently found that motivation wanes when learners struggle with abstract concepts, perceive tasks as monotonous or irrelevant, or receive infrequent feedback (Sailer *et al.,* 2017).

Gamification has been a reaction to these problems. In the empirical research on STEM and computing, it has been demonstrated that points, levels, competition and narrative can be combined to boost motivation and persistence (Sailer *et al.,* 2017; Ratinho & Martins, 2023).

1.3 Problem statement

Despite Java's enduring importance, undergraduate students continue to experience high levels of disengagement and limited knowledge retention in programming modules. Traditional didactic instruction lacks personalisation and interactivity, while most existing gamified interventions lack strong theoretical grounding or contextual adaptation to local conditions. Consequently, the field lacks a validated, user-informed, motivationally grounded approach to supporting undergraduate mastery of Java programming — one that translates established gamification theory into concrete, empirically tested design decisions. Left unaddressed, this gap will continue to undermine engagement and retention in computer science and IT education.

1.4 Justification of the problem

The gap between pedagogical reality and curricular ambition in programming education is the long-running issue in Java education as well as in the overall teaching and learning of the language. South African universities had high dropout and failure rates of introductory programming modules, which were also a contributing factor to low overall representation of the IT labour force (DHET, 2020). The process is a self-perpetuating cycle: low programming self-efficacy and confidence reduce motivation, which results in the avoidance of programming courses and withdrawal from digital career opportunities (Bergin & Reilly, 2006).

Students who make a decision in favour of programming enhance their opportunities of working in the sphere of computer science or in many other areas that rely on digital technologies and the analysis of data. However, it is found that students frequently face difficulties in understanding various aspects of this subject, which causes lesser motivation and interest among them (Watson & Li, 2014). This disinterest prevents students from effectively learning programming and, in some cases, turns learners off from computer science completely (Bergin & Reilly, 2006). The process of turning everyday activities into games has become a way of increasing motivation and attentiveness (Kapp, 2012). Research has suggested that gamification can increase the learning attitudes in different educational environments (de-Marcos *et al.,* 2016). The paper confirms that the point system, badges and leader board and the challenges assist in keeping the students interested, motivating them and improve their performance as they proceed with the course (Hamari *et al.,* 2014).

Although the overall usefulness of gamification in education is investigated, there is a gaping need for more research to examine how individualized gamification models can improve the learning process even more. To illustrate, researchers like Subhash and Cudney (2018) have demonstrated that gamification enhances participation in STEM education but do not give attention to individual basis of learning in programming education. Equally, Zhan *et al.,* (2022) demonstrate that gamification elements such as badges and leaderboards positively impact motivation and achievement, but highlight that effects may diminish without investigating long-term motivation. Recent studies by Orji *et al.,* (2018) and Tondello *et al.,* (2019) show that personalized gamification has a positive effect on student motivation, engagement and learning outcomes. Nevertheless, such results can be quite general in the programming education environment.

The Orji *et al.,* (2018) review did not delve into the unique challenges and learning needs specific to programming disciplines. Similarly, the meta-analytic findings of Sailer and Homner (2020) require further adaptation and validation within the context of complex programming tasks and the diverse learning styles encountered in programming education (Field, 2017). These gaps highlight the need for research focused on tailoring gamified experiences to optimize engagement and educational effectiveness for diverse student populations. This study responds to this gap by incorporating theoretical models into the design of a user-informed gamified learning environment for Java programming. By doing so it will address the lack of depth in current gamification research and contributes new insights into how motivation and performance can be enhanced through theory-driven personalized design in programming education.

1.5 Research aim and objectives

1.5.1 Research aim

The primary aim of this study is to design and develop a fully functional gamified web-based application aimed at enhancing engagement and learning in Java programming among undergraduate students. This aim is pursued through a design-and-creation approach grounded in empirical student needs data and motivational theory, with the artifact's effectiveness in practice to be confirmed through future summative evaluation.

1.5.2 Research objectives

- To critically evaluate the theoretical and empirical evidence for gamification's effectiveness in supporting student engagement and motivation in programming education.
- To identify which gamification design factors they consider critical to sustained engagement in a Java learning context.
- To determine which specific gamification elements (e.g. points, badges, leaderboards, optional competition) undergraduate students perceive as most effective for engagement and motivation, relative to elements they perceive as counterproductive.
- To design, develop and verify a functional web-based gamified application for Java learning, and to evaluate the resulting artifact against the technical usability and functional requirements derived from the empirical findings.

Table 1.1: Mapping of research questions to research objectives

RQ	Research question	RO	Research objective
RQ1	What is the impact of gamified applications on student engagement and motivation in learning Java programming?	**RO1**	To critically evaluate, through a synthesis of existing literature, the theoretical and empirical evidence for gamification's effectiveness in supporting student engagement and motivation in programming education.
RQ2	What are the critical success factors of gamified applications in learning in Java?	**RO2**	To identify, through empirical engagement with target users, which gamification design factors they consider critical to sustained engagement in a Java learning context.

RQ3	Which gamification elements are most effective in enhancing student engagement and motivation in a Java programming context?	**RO3**	To determine, using empirical user data, which specific gamification elements (e.g. points, badges, leaderboards, optional competition) undergraduate students perceive as most effective for engagement and motivation, relative to elements they perceive as counterproductive.
RQ4	How can a web-based application that incorporates gamification elements be designed and developed to enhance Java learning?	**RO4**	To design, develop and verify a functional web-based gamified application for Java learning, and to evaluate the resulting artifact against the technical usability and functional requirements derived from the empirical findings.

Each RQ maps directly to its corresponding RO. Chapter 5 conclusions are structured to address RO1–RO4 in sequence, demonstrating how each objective was met through the DSR design-and-creation process.

1.6 Research questions

1.6.1 Main research question

How can a fully functional gamified web-based application enhance learning for Java programming undergraduate students?

1.6.2 Secondary research questions

- What is the impact of gamified applications on student engagement and motivation in learning Java programming?
- What are the critical success factors of gamified applications in learning in Java?
- Which gamification elements are most effective in enhancing student engagement and motivation in a Java programming context?
- How can a web-based application that incorporates gamification elements be designed and developed to enhance Java learning?

1.7 Literature overview

This study draws on a body of literature spanning four interconnected areas: the theoretical foundations and empirical evidence for gamification's effectiveness in education generally; the specific pedagogical challenges of Java programming; the documented impact of gamification

within programming education specifically; and the design of individual gamification elements (points, badges, leaderboards) and their motivational mechanisms. These four themes, synthesised in full in Chapter 2, directly inform the problem statement and research objectives above.

1.8 Research methodology

This research adopts an information systems perspective grounded in design science research (DSR) principles to evaluate how gamification can improve learning experiences in Java programming. DSR is a problem-solving paradigm that emphasizes the creation and evaluation of innovative artifacts to address real-world problems in a rigorous and methodical manner (Peffers *et al.,* 2007).

1.9 Significance of the study

The primary contribution is the artifact itself: a fully functional gamified Java learning application. This tool serves as a theoretically grounded, empirically informed prototype that can be further developed or adopted by the university to assist lecturers in managing large programming classes. It provides a platform where students can practice safely and receive instant feedback.

The study synthesizes and extends motivational theory within the context of programming education, offering a deeper understanding of how gamification supports engagement in Java learning. It also demonstrates the value and flexibility of design science research (DSR) as a methodology for driving innovation in educational technology. Furthermore, the study produces actionable design guidelines that can inform future empirical research, prototype testing and large-scale implementation studies.

The study provides a blueprint that course designers and policy makers can use to improve retention and engagement in programming modules. It also supports regional and national initiatives aimed at strengthening digital skills and enhancing graduate employability. On a broader societal level, the artifact contributes to promoting equity and representation in computer science by making programming learning more accessible, motivating and inclusive.

1.10 Scope

This study is positioned primarily as an information systems contribution within the design science research (DSR) paradigm; gamification theory, Java pedagogy and motivational psychology are drawn upon as informing disciplines that shape the artifact's design requirements, rather than as

domains in which the study seeks to make independent theoretical contributions. The scope of this study covers the complete software development lifecycle (SDLC) for a web-based educational application. It begins with requirements elicitation, where student surveys are used to identify motivational factors and learning needs. The system design phase follows, involving the creation of UML diagrams, database structures and UI wireframes to guide development. Implementation consists of building a functional application with both backend server logic and a responsive frontend interface. Finally, the system undergoes functional and usability testing to ensure that the application operates correctly and provides an effective user experience.

1.11 Chapter division

Chapter 1: Introduction and Background: Outlines the research problem, objectives, rationale and the scope of the software development.

Chapter 2: Literature Review: Provides an in-depth analysis of gamification, Java pedagogy that guide the system design.

Chapter 3: Research Methodology: Details the design science research process, data collection methods and the software development methodology used.

Chapter 4: System Design and Architecture: Presents the empirical findings from the data collection and details the technical architecture, database design and interface of the developed artifact.

Chapter 5: Conclusion and Recommendations: Summarizes the findings, discusses the limitations of the developed system and proposes future work for implementation and expansion.

1.12 Chapter summary

Chapter 1 established the need for a novel approach to pedagogical innovation in undergraduate Java education, as well as the associated opportunity. The proposed research aims to fill the longstanding gap between curricular ambition and educational effectiveness by incorporating gamification, motivational theory, and user feedback. This chapter presents a detailed argument, a clear problem statement, an extended theoretical and conceptual framework, and a roadmap for the remainder of the study. The following chapter will provide an in-depth analysis of the supporting literature, situate the study within the wider academic discourse, and offer a step-by-step design and impact guide.

LITERATURE REVIEW

2.1 Introduction

This chapter gives a thorough literature review of the recent literature on the use of gamification in education with a focus on how it may be used to improve the learning process in Java programming. The review starts with the definition of key terms: gamification, engagement and motivation and Java programming education. It proceeds to prior empirical research that directly examined gamification in the case of programming which preconditions a critical evaluation. From there it methodically investigates the overall influence of gamification in education and breaks down the most important issues faced by Java programming learners, analysing the impact of gamification on programming and explaining the elements of gamification that are used in the learning settings.

Through critical analysis of existing body of knowledge, this chapter points out gaps, contradictions and areas that need more research hence providing clear rationale and consistent theoretical coverage of the research. The chapter ends with a brief overview of key findings and points out the research questions that arise due to the existence of gaps. Given the principles of design science research (Hevner, 2007; Gregor & Hevner, 2013), this research discusses the theoretical consequences of gamification and develops a practical artifact to fill in the pedagogical gaps in the teaching of Java programming.

2.2 Definitions

Gamification: The use of game design elements (for example points and leaderboards) and game principles (for example narrative and immediate feedback) in non-game contexts to enhance user engagement and motivation (Deterding *et al.,* 2011; Kapp, 2012).

Engagement: A multidimensional construct comprising behavioural engagement (effort and participation), emotional engagement (interest and satisfaction) and cognitive engagement (investment in understanding complex ideas) (Fredricks *et al.,* 2004).

Java Programming: A high-level, class-based, object-oriented programming language designed for cross-platform compatibility. It is characterised by a verbose syntax and complex abstract concepts such as polymorphism and inheritance (Oracle, 2024).

design science research (DSR): A research paradigm in information systems that involves the creation of innovative artifacts to solve practical problems, contributing to both utility and theory (Hevner, 2007).

2.3 Previous studies in gamified programming education

Although the overall scope of gamification in teaching is vast (Hamari *et al.,* 2014; Dicheva *et al.,* 2015), the review is limited to studies that have directly resulted in implementing gamification in programming courses or have offered valuable theoretical or empirical information directly applicable to the field. The development of this subfield is characterised by a shift from general observations about the advantages of gamification to more specific studies in computer science.

Conceptual work in the early years by Deterding *et al.,* (2011) and Kapp (2012) laid the groundwork for applying game design elements to non-game environments such as education. These concepts have informed subsequent empirical studies on a range of topics. Student engagement and high attrition have long been identified as key problems in programming education (Guzdial, 2013; Watson & Li, 2014), and researchers have approached developing new pedagogical strategies including gamification in a natural manner.

2.3.1 Foundational applications and empirical evidence

There are several studies that directly support the use and effectiveness of gamification in programming education. Sailer et al. (2017) reported that gamification in programming education can improve adherence to instructions, encourage repeated practice, and foster a sense of achievement. They noted the need to identify the benefits of gamification for real-time feedback and skill acquisition in coding. Barna and Fodor (2018) argued that gamification can enhance knowledge acquisition and recall in programming lessons because of the interactive nature of the format, offering cognitive advantages in addition to increased engagement. Zhan *et al.,* (2022) found that gamification can counter learner problems and lack of motivation in programming languages, addressing core challenges in programming pedagogy. Zhan *et al.,* (2022) provided meta-analytic evidence from multiple programming education contexts by examining the impact of gamification across 21 empirical studies. Their analysis, which used game mechanics, points, badges, a leaderboard, and virtual rewards showed that students in the experimental group were much more engaged in online forums and in optional challenges.

2.3.2 Emerging trends and personalised approaches

More recent research illustrates that there is an increasing movement towards learner-focused and customised gamification approaches. Tondello *et al.,* (2019) examined the concept of

designing personality-based gamified learning technologies with reference to the context of computer science learning and addressing the needs and preferences of each student. Orji *et al.,* (2018) also made their contribution with a particular emphasis on the importance of the personalized gamification to engage students. Although they are not strictly speaking about programming, their results emphasise the motivating power of customised gamified experiences in motivating students in technical and educational environments, which, naturally, implies that less general solutions are more useful. Sailer & Homner (2020) estimates that student-centred design in the agile context will contribute to more engagement in the programming education, which highlights the continued applicability of modern agile practices applied to student-centred design, which implicitly implies the principles of gamification.

2.3.3 Meta-analytic synthesis

Sailer & Homner (2020) provide a meta-analytic review of gamification in educational settings encompassing computer science education in particular. This piece of work brings together empirical data on its impacts on engagement and learning outputs and provides a comprehensive, synthesized perspective on the state of the art in the discipline. It supports the overall positive influence but also indicates the areas, where additional and more specific research is needed. Notably, the meta-analytic evidence is not uniformly positive: effect sizes for competitive elements such as leaderboards vary considerably across studies, with some samples showing demotivating effects for lower-performing students (Hamari *et al.,* 2014), a methodological inconsistency that the present study's empirical design (Section 4.2.2) specifically probes by measuring variance, not just mean preference, for competitive features. Nonetheless, although these syntheses demonstrate the overall effectiveness of gamification, only a small number of them have studied its direct impact on the mastery of object-oriented programming, which justifies the relevance of more focused research.

These studies form a strong base that can be used to prove that gamification is an effective and frequently useful method of teaching, motivating and improving engagement in programming education. They are categorised as more general conceptual justifications all the way down to more specialized empirical interventions in programming language and computer science settings. Nevertheless, a common theme of this literature is the suggestion that further studies are required, especially the design and implementation of gamification features in particular, complex programming languages, such as Java. The shift of the general benefits to the specific applications is also a field that requires investigation and therefore the latter thematic analysis will critically examine the matter. To overcome these unrelenting difficulties, the following section will

discuss how the concept of gamification interventions has been implemented in programs to overcome parallel problems.

2.4 The impact of gamification in education

This segment is a critical analysis of the overall effects of gamification on the learning setting including its capacity to increase engagement, motivation and eventually improve the learning outcomes in various education settings. The key ideas that guide this field are the terms gamification, student engagement, learner motivation and academic achievement or learning outcomes. Some of the major keywords that can be used with this theme are game-based learning, educational technology, student performance and pedagogical innovation.

2.4.1 Early conceptual studies and general benefits

The studies of gamification in education have revealed an evident development of initial conceptual definitions to strong empirical research to show the beneficial impact of it. The conceptual framework of gamification was laid down by Kapp (2012) and Deterding *et al.,* (2011), who both focused on the use of the elements of game-design in non-games. Dicheva *et al.,* (2015) conducted the first systematic reviews of its use, which are efforts to expand the knowledge of the potential. Later empirical research has always indicated its effectiveness, such as de-Marcos *et al.,* (2016) and Domínguez et al. (2013), both of whom revealed the effectiveness of including game-based features into the learning process in terms of engagement and learning results across different fields. Although not dedicated solely to programming, Sailer et al. (2017) applied these findings specifically to the field of programming education, commenting on the fact that gamification can be used to provide instant feedback on the compliance with the instructions, allow repetitive learning and instil a sense of achievement.

2.4.2 Empirical evidence and meta-analyses

More up-to-date and extensive syntheses, including the systematic review by Hamari *et al.,* (2014), concur that the impact of gamification on the achievement of students, especially with regards to STEM subjects, is positive. This supports previous experimental results of Sailer *et al.,* (2017) on the involvement of certain game design features such as points, badges and leaderboard in enhancing student motivation and satisfaction of psychological needs, a finding corroborated by the broader meta-analytic evidence on gamification's positive effect on engagement and learning outcomes (Sailer & Homner, 2020).

The main advantage of the literature is that it has a growing empirical foundation that is more often based on experimental or quasi-experimental studies, as it is the case with Sailer *et al.,*

(2017). This gives a sound background of the overall impacts of gamification. Nevertheless, a comparative study indicates that Ratinho and Martins (2023) found gamified learning strategies improved motivation and Zhan *et al.,* (2022) found similar benefits with fewer competitive features, which poses a risk of unintended impacts on the use of gamified tools.

2.4.3 Limitations and future directions

Although the overall finding of beneficial impacts is presented, there is frequent discrepancy on the magnitude and various mechanisms of influence with some studies reporting higher effects on motivation than on deep conceptual learning or the reverse (Hamari *et al.,* 2014). One of the main pitfalls is that the majority of studies tend to have short interventions and their samples are frequently selected among the particular population of the university (Hamari *et al.,* 2014), which restricts the transfer of the results to the wider educational setting or to a different student group to consider in the scope of the present study. This is especially applicable to South African higher education, in which the demographic diversity of students and lack of technological access might affect the rate of engagement with ICT-related courses (Adeyemo, 2023). Different pedagogical approaches and different gamification designs adopted in the studies also complicate the idea of direct comparisons and do not allow developing universal best practices.

Ethical factors, including the possible high competition caused by the use of leader-boards, which may discourage less successful students, or the problem of data privacy in online classrooms, are either completely ignored or casually mentioned (Tondello *et al.,* 2019). The above will be central in designing the current study to create a positive and inclusive learning experience, based on the principles of beneficence and inclusivity in educational technology (Buchanan & Zimmer, 2021) and guarantee the strict compliance with the informed consent and data protection procedures. The foremost suggestion regarding the future research is always to conduct further studies on a long-term basis of engagement in multiple semesters, research about the particular design principles and studies in a variety of educational contexts beyond the common university courses.

There is still a significant gap in comprehending particular circumstances under which the gamification is most effective in the context of this or that learning goal and the demographics of students. Although the overall effect is good, a closer insight on how these findings can be applied to more intricate and syntax-heavy sciences such as Java programming and whether the effect is long-lasting learning or just a temporary activity, is generally missing. This broad theme is thus the cornerstone of the already proven, positive returns of gamification hence the rationale behind the research as a feasible and encouraging intervention within a conceptually demanding and

high-attrition setting like Java programming education. These broad findings are the ones that the proposed research is set to develop further, by digging deep into the particular application of gamification to address the particular pedagogical challenges of the Java programming.

2.5 Key challenges in java programming education

The purpose of the section will be to outline the most important challenges that students face in learning the Java programming language and criticize the constraints of traditional pedagogical methods in helping students overcome these challenges. Such concepts as object-oriented programming (OOP), algorithmic problem-solving, cognitive load, student attrition and the effectiveness of different pedagogical methods are its core concepts. Cognitive load theories (Sweller, 1988) give an idea on how to interpret the cognitive load of the Java complex syntax and abstract concepts. The keywords that can be used in this theme are programming education problems, OOP instruction, computer science instruction problems and learner engagement in programming.

2.5.1 Inherent complexity and cognitive load

Studies have continually stated the complex nature of the Java programming to be a key obstacle to students. Early investigations by Bergin and Reilly (2006) into the factors influencing the success of programming and by Guzdial (2013) into what is observed, have given emphasis on the steep learning curve presented by the complex syntax and the complexity of the principles of object-oriented (OO) design inherent in Java such as inheritance, polymorphism and encapsulation. This literature shows the combination of these abstract concepts with the command to use algorithms to solve problems is a key factor that leads to significant student complexity and growing cognitive load (Sweller, 1988). The abstractness of OOP and specially makes this steep learning curve so that students have to learn to think in new non-sequential ways that procedural thinking does not teach them to think.

2.5.2 Limitations of traditional pedagogies and attrition rates

More importantly, older pedagogical techniques, often involving passive lecturing and repetitive drills, do not always succeed in engaging students. According to Guzdial (2013), this absence of participation is one of the primary reasons why the rates of achievement and massive student attrition have been so high in conventional Java programming courses. Watson and Li (2014) once again have examined the long-standing problem of high failure rates in introductory programming, which solidifies these systemic problems and proves that the problem remains unchanged with time. According to more recent research, including those of Dicheva et al. (2015), customized and up-to-date methods are required in the education of programmers, which means

that the old, solid, academic methods will not be sufficient to prepare graduates to the demands of the dynamic modern work environment. This explains why more flexible and interactive teaching approaches are necessary, especially in the regional foundation such as South African higher education, where attention to ICT education is an ongoing priority (Adeyemo, 2023).

The theoretical grounding for why active, constructive approaches outperform passive lecturing predates the gamification literature itself. Papert (1980) argues that learners build genuine understanding by actively constructing and debugging their own artifacts, rather than by receiving instruction passively; in his account, the experience of finding and fixing one's own errors is not an unfortunate side effect of learning to program but a primary mechanism through which programming concepts are understood. This constructivist position offers a theoretical explanation, independent of any specific gamification framework, for why hands-on, mission-based coding exercises of the kind described in this study would be expected to outperform passive instruction: the value lies not in the game-like packaging itself but in the active construction and debugging the packaging is designed to encourage.

2.5.3 Methodological critiques and ethical considerations

Little contradiction exists in terms of whether these challenges are widespread in Java programming education but rather, there are more concerns about how best to address them. One of its strengths, though, is a good level of diagnostic ability: this kind of literature seems to be able to clearly state the position of issues with Java and the reasons behind it (Bergin & Reilly, 2006; Guzdial, 2013). Nevertheless, a significant drawback is the focus on most studies that do not adequately evaluate the solutions to the problem or offer empirically validated pedagogical models that are specifically aimed to address the issues that Java presents.

Qualitative studies (Kvale & Brinkmann, 2015) or surveys (Krosnick, 2018) are often used as research designs in this field, as these methods are effective in identifying a problem but occasionally do not meet the requirements of an experiment to determine the effectiveness of various instructional interventions. Sample sizes may be quite different (Etikan *et al.,* 2016) and data analysis is often aimed at identifying relationships and not making causal connections to create a useful instructional design (Field, 2017). There are ethical aspects, including the risks of stress and frustration of the students under the impact of too complicated concepts or insufficient assistance.

The implicit burden of success is usually on the student alone instead of pedagogical design and still, the current study would take a learner-centred design in order to alleviate this stress. Authors have continued to recommend more interactive and student-centred methods of learning and

additional research on how educational technology can help in learning complex programming skills (Dicheva *et al.,* 2015). Thus, while the literature robustly diagnoses Java learning challenges, it offers few empirically tested, technology-enhanced interventions to address them which is a gap this study seeks to fill through gamified design.

2.6 Impact of gamification in programming education

This section provides a critical analysis of the empirical evidence of the effect of gamification, specifically in the context of programming education, which discusses the actual impact of game mechanics and the burning problem of the transferability of findings to other programming languages. Imperative ones in this case are: gamification elements, learner engagement, programming proficiency and implicitly; transfer of learning and situated cognition. The keywords will be relevant and include such items as programming education gamification, python gamification research, student motivation in computer science and gamification game mechanics.

2.6.1 Empirical evidence from programming contexts

The early meta-analyses, including the study by Hamari *et al.,* (2014), in general, state that gamification can contribute to the creation of interactivity and enhancement of learning experiences in different fields. The use of personalized gamification in engaging students is a specific aspect that Orji *et al.,* (2018) consider and it is particularly applicable in the context of programming, where the adoption of more specific methods has become a central trend. Tondello *et al.,* (2019) also adds to the contributions made by the design of personality-based gamified learning applications to teach computer science, which could potentially be positive in case of bespoke solutions that would consider individual differences between learners. According to Sailer & Homner (2020), the learner-centric design in agile settings will positively influence the engagement in programming education, which can be achieved through the further relevance of combining modern pedagogical and developmental strategies with student-centred design, implicitly presenting the principles of gamification in it. All these investigations will prove the effectiveness of the implementation of gamification and its positive consequences in the context of programming and computer science, which proves that the increased involvement of students is a tangible phenomenon.

2.6.2 Transferability and methodological considerations

Although research such as Orji *et al.,* (2018) and Tondello *et al.,* (2019) offers strong evidence of increased engagement in particular programming education contexts, the direct applicability of research to the specifics of Java programming is a rather important area of consideration and possible argument. Various programming languages have different syntaxes and paradigms

(such as the multi-paradigm flexibility of Python and the strong object-oriented enforcement of Java) and common pedagogical introduction courses. Therefore, what can be found in one language cannot be translated to another language without meticulousness adaptation and re-assessment.

Their experimental or design-based nature (Zhan *et al.,* 2022; Gregor & Hevner, 2013) is also their strength, which enables making stronger inferences about the existence of a causal relationship between particular gamification designs and the reported engagement or learning outcomes (Hamari *et al.,* 2014; Tondello *et al.,* 2019). But a possible limitation, to the extent of their direct application to the context of the present study, is that they might not be as precise or as thorough as to reflect or take into account the conceptual complexities of Java programming, including the ability to learn and understand abstract principles of OOP or to retain complicated syntax in the long term. The emphasis of most of these studies is on gamification as a pedagogical approach but not as a complete game-based learning system but this fits into the present-day research.

Quantitative comparisons of engagement and activity levels among the groups or qualitative feedback on user experience (Guest *et al.,* 2012; Nowell *et al.,* 2017) are often the primary data analysis as it aids in the investigation and requires more powerful mixed-method approaches to understand it comprehensively. The issue of ethical implications based on competitive aspects (for example, leaderboards that may add unnecessary stress) or providing access to gamified aspects fairly is common in these particular areas of research may not be delved into in depth; nevertheless, the present-day research will address these concerns in its design proactively. The long-term outcomes of gamification and the influence it has on different groups of learners are commonly proposed as future studies by the authors of this article. This means that, to implement successful gamification models under Python or general computer science education, one needs to re-test them in Java.

The main literature gap here is that the successful gamification strategies of other programming languages or general computer science environments have been directly replicated and straight-forwardly adapted into Java and that the observed engagement can be directly reflected into an enhancement of the conceptual understanding and learning outcome on the unique complexities of Java. Although there is an overall positive effect, the available literature can be characterized by the lack of a detailed examination of the effect of the elements of gamification applied to the unique feature of the Java learning curve and its role in mastery. This is why this theme is so important in the proposed research because it shows the examples of successful gamifications

in other related areas and at the same time, provides the information about the need to adjust and reconsider such strategies to the specific pedagogical and cognitive needs in learning Java. This literature review therefore makes the proposed research a crucial milestone toward filling the gap of transferability of the success in gamification in other programming settings to the problems in Java education.

2.7 Gamification elements in education

In this section, the particular forms of gamification elements that are most frequently used in education, including points, badges, leaderboards and levels, are targeted and their underlying motivational mechanisms examined, with the vital necessity of context-specific implementation in Java described. Some of the important concepts in the discussion are the particular features of gamification, extrinsic motivation, intrinsic motivation, feedback mechanism and recognition of achievement. Operant conditioning theories (Skinner, 1938) are relevant to clarify the effectiveness of points and badges as rewards, whereas the social comparison could be used to explain the influence of leaderboards. The most important keywords related to this theme are; points, badges, leaderboards (PBL), digital badges in education, motivational design in learning and game mechanisms in computer science education.

2.7.1 Common gamification elements and their motivational mechanisms

A foundational taxonomy was presented by Deterding *et al.,* (2011), who defined points, badges and leaderboards (PBLs) as one of the most frequent and effective elements of gamified learning settings. Studies have always shown that points offered as results upon completing tasks and challenges present instant feedback on performance and develop a sense of progress (Sailer *et al.,* 2017). Under the circumstances of Java programming, it was possible to earn points on the completion of the coding tasks, the active involvement in online discussions or active support of other students. Hamari *et al.,* (2014) contend that points may be especially effective when using new members, who may get an initial feeling of success which results in competence.

Werbach and Hunter (2012) provide a structural account of why these elements function as they do, distinguishing dynamics (the overall experience, such as a sense of progression or status), mechanics (the processes that drive interaction, such as awarding points for completed challenges), and components (the concrete implementations of mechanics, such as points, badges and leaderboards themselves). Applied to JavaRealm, XP and badges are components instantiating a progression mechanic, which together serve the underlying dynamic of mastery: the sense that a student is visibly and verifiably becoming more competent at Java over time. This framing clarifies that points, badges and leaderboards are not interchangeable decorative

add-ons but specific implementations chosen to support a particular motivational dynamic, reinforcing the rationale for treating PBL design as a deliberate choice rather than a default feature set.

This emphasis on mastery as the central motivational dynamic is consistent with Koster's (2013) account of why games are engaging in the first place: he argues that games function fundamentally as pattern-learning systems, and that the experience of fun arises specifically from the act of recognising and mastering a pattern, not from surface aesthetics or rewards considered in isolation. On this view, a badge or level-up is not intrinsically motivating; it is motivating because it marks the visible mastery of a pattern the learner has worked to recognise, such as the syntax pattern of a for-loop or the structural pattern of a class definition. This supports the design rationale for structuring JavaRealm around discrete, master able 'missions' rather than open-ended practice, since each mission isolates a specific pattern for the learner to master and have that mastery explicitly recognised.

Badges, which are awarded upon mastering certain concepts in Java, help in recognising students who have mastered and have a physical feeling of accomplishment (Deterding *et al.,* 2011). They could be graphically illustrated to reflect a result of progress, e.g., it could be a title like Syntax Specialist or Object-Oriented Master, which visualizes student incentives to advance to further study and become a professional. This has subsequently been elaborated by more recent scholarship by Tondello *et al.,* (2019) who argues about the application of digital badges in personality-based gamified tools to teach computer science and notes how they can assist students become more autonomous and motivated learners. Park and Kim (2019) similarly frame well-designed badges as intrinsic motivators that support learner autonomy and self-directed engagement, underscoring badge design as a deliberate motivational tool rather than a superficial reward.

Leaderboards are often used to check performance and promote competition; they have a less obvious effect. Although they have the potential of motivating some students through social comparison, the research has shown that they may, in some situations, demotivate low-performing students (Sailer *et al.,* 2017); indicating that their design and application should be carefully considered to provide equal motivation to all students. The Seaborn and Fels (2015) study provide additional experiment findings on the relative effectiveness and possible traps of various gamification components.

2.7.2 Design considerations and ethical implications

Although the motivational potential of PBLs is generally accepted, there are also major debates about how best to design and integrate them, as well as potential negative aspects. For example, according to Sailer *et al.,* (2017), leaderboard performance can vary significantly based on students' demographics and cultural backgrounds, and poor design of competitive aspects can cause anxiety or create an unhealthy learning environment. A major strength of the literature in this field is that it identifies and classifies these factors and the overall motivational outcomes clearly (Deterding *et al.,* 2011; Sailer *et al.,* 2017).

One major weakness, however, is the lack of empirical studies that objectively compare the efficacy of various factors—or combinations thereof—in complex domains of learning, such as Java programming. Most of the available research considers a complex set of factors, making it difficult to determine the exact effect of each element. Self-report surveys used to measure motivation, or engagement measured through observational data, are common research designs and valuable, but they are vulnerable to inherent biases (Krosnick, 2018). The ethical considerations guiding the proposed research will include properly designing competitive aspects, emphasizing mastery-based feedback rather than comparative ranking, and providing equal opportunities for all students to progress and receive recognition. These guidelines guide the creation of the artifact within the design science framework of this study, ensuring that every aspect of gamification supports learners' well-being and is fairly represented. According to Tondello *et al.,* (2019) and other researchers, more research is needed on how these factors can be used to promote learner independence and mastery-oriented learning, rather than compliance.

One of the most important gaps, then, is the lack of knowledge about best practices for developing and applying these gamification elements in the Java programming learning environment. The literature, however, is silent on how point systems can be optimally configured to support complex coding tasks, whether certain Java concepts deserve mastery badges, and how leaderboards can be framed to encourage healthy competition without discouraging learners of different abilities. This theme is vital to the proposed study, as it will directly inform the selection and custom design of gamification elements for the Java programming learning environment and their strategic placement within the learning contexts associated with the identified challenges in Java learning.

This chapter has reviewed in detail the literature relevant to gamification in Java programming education. The review begins by defining key terms and outlining the landscape of previous studies on gamification across different programming and computer science settings. It provides

an overview of early research on the advantages of gamification and, subsequently, empirical studies on its application, with emphasis on studies analysing personalised gamification and its effects on engagement (Orji *et al.,* 2018; Tondello *et al.,* 2019).

Although the existing educational and programming-related literature indicates the potential of gamification to address enduring issues in Java programming learning, there is one critical gap. It does not contain research specifically designed, implemented, and evaluated with Java programming in mind, with an emphasis on effective methods for integrating particular gamification elements to improve conceptual knowledge and long-term interest. Thus, the literature review provides a solid basis and justification for the proposed study, which aims to fill this gap by designing and evaluating a gamified approach to teaching Java programming that specifically enhances understanding of abstract object-oriented programming (OOP) concepts and fosters learners' long-term engagement. The insights and gaps identified in this chapter have direct implications for the research methodology of the present study, which is described in Chapter 3, including the research design, sampling procedures, data collection tools, and analytical procedures to ascertain the effects of gamification on learning Java programming.

RESEARCH METHODOLOGY

3.1 Introduction

This chapter details the research methodology and design employed to guide the development of a gamified web-based application for Java programming education. The aim is to demonstrate methodological rigour and ensure the resulting artifact is both theoretically grounded and practically applicable. This study employs a design science research (DSR) approach.

As described by vom Brocke *et al.,* (2020) and expanded by Oates *et al.,* (2022) DSR is ideal for addressing real-world challenges through the creation and evaluation of innovative artifacts. The approach emphasizes iterative refinement and problem-solving enabling the integration of theoretical insights and stakeholder feedback during the design process (Gregor & Hevner, 2013). This structured process ensures the resulting gamified application is theoretically grounded, user-centred and ready for future implementation.

3.2 The design science research (DSR) methodology

Following the established DSR framework by Peffers *et al.,* (2007) the study's procedure is structured into the following sequential and iterative phases: The first phase which is the problem identification and motivation involved a thorough review of the literature (Chapter 2) and articulation of the problem statement (Chapter 1) focusing on the challenges students face in learning Java programming and the need for engaging technological solutions. This phase aimed to define the practical problem clearly and justify the value of creating a gamified application that addresses both motivational and educational gaps. The second phase, Objective Definition for a Solution, involved establishing the functional and non-functional requirements of the artifact. This included specifying the features, mechanics and educational alignment necessary for the application to be engaging, effective and pedagogically relevant thereby setting clear criteria for evaluating the success of the design.

During the design and development phase, a functional artifact and prototype of the web-based application was created. This initial design incorporated gamification elements informed by the literature and best practices in educational technology allowing the artifact to reflect both theoretical principles and practical usability considerations. The evaluation phase consisted of systematic data collection through questionnaires and focus group discussions ensuring that feedback was comprehensive, multi-faceted and aligned with academic validation standards. This phase assessed the utility, usability and pedagogical effectiveness of the functional artifact.

Finally, the communication phase involved detailed documentation of the artifact, the design and development process and the research findings in the dissertation (Chapters 4 and 5). This phase aimed to disseminate both the prescriptive knowledge (guidelines for future artifact development) and the descriptive knowledge (lessons learned and theoretical contributions) to the academic community ensuring that the study provides a rigorous well-documented and theoretically grounded contribution to the field of educational technology and gamified learning.

3.3 Research paradigm

The research paradigm adopted in the study is pragmatic research paradigm which is consistent with the objectives of design science. Pragmatism is focused on practical results and experimental improvement of interventions (Creswell & Plano Clark, 2021). Pragmatism in design science enables the production of solutions that can be applied and adapted to the real-life environment instantly (vom Brocke *et al.,* 2020). The paradigm is more appropriate to test how efficient gamified learning tools are in improving the results of the Java programming.

In this study, pragmatism supports the evaluation of gamified learning tools by integrating theoretical knowledge and user feedback to solve a specific educational challenge. As Creswell and Creswell (2023) note, pragmatism focuses on finding workable solutions to real-world problems. Consistently, Oates *et al.,* (2022) observe that design and creation research is inherently pragmatic, emphasizing artifact production and evaluation to solve practical challenges. This study aims to develop a functional and pedagogically effective learning tool that integrates iterative design and creation principles with the practical goals of Java programming education.

3.4 Research strategy

The study employs a design and creation strategy to develop the gamified application for Java programming education. Design and creation research focuses on the iterative development of artifacts to address specific real-world problems (Oates *et al.,* 2022). This methodology is particularly suitable for this study as the primary goal is to build a gamified application informed by stakeholder feedback gathered through questionnaires and focus groups.

The study involved users (undergraduate IT students) participating in iterative phases of evaluation, thus ensuring that the artifact meets the needs of the users and the educational goal (Gregor & Hevner, 2013). Focus groups and questionnaires allow the feedback loop to be built so that the functionality and educational value can be incorporated into the final design.

The design and creation strategy not only emphasizes the artifact's technical functionality but also its ability to achieve the desired educational outcomes such as enhanced motivation and improved performance. In contrast to purely theoretical methodologies design and creation research offers a pragmatic approach that bridges theory and practice. Baskerville *et al.,* (2018) argue that this strategy is ideal for developing solutions that address practical problems while contributing to academic knowledge. By focusing on design and creation, this study leverages the strengths of this strategy to provide a robust evaluation of the gamified application's effectiveness in increasing student engagement and motivation, as well as its potential scalability in diverse educational contexts.

3.5 Population and sampling strategy

The study makes use of a purposive sampling where the target population is undergraduate students studying IT either through computer science or information systems programs and taking or completed course in Java programming. This method allows for the selection of participants who are directly related to the research aims and have the basic knowledge of programming but have issues with learning Java (Etikan *et al.,* 2016). In research about design and creation the usefulness and effects of the artifact should be evaluated under the real-life scenarios to prove its efficiency (Oates *et al.,* 2022).

Invitations were sent to the eligible students in the Java programming course in the North-West University with the permission of the NWU gatekeeper (Annexure A). The participants filled an informed consent form (Annexure B) to make sure that they have been made aware of the terms of participation which outlines the purpose of the study, the right to withdraw at any time without academic penalty, the absence of monetary incentives and assurances of confidentiality and POPIA-compliant data protection.

3.5.1 Sample size

To initiate the study a purposive sample of 20 undergraduate students who were enrolled in Java programming modules at the North-West University (NWU) was selected to participate in the research. These students formed the participant pool for the questionnaire phase of the study. From this initial group a subset of between 6 participants was subsequently invited to take part in a focus group discussion. The focus group participants were drawn exclusively from the same purposive sample used during the questionnaire phase to ensure consistency and alignment across data collection methods. This group size was chosen to facilitate meaningful interaction in-depth discussion and effective group dynamics.

Following the completion of the questionnaire students who expressed interest in further participation were contacted and formally invited to the focus group. Prior to participation all invited students were provided with an informed consent form (Annexure B), which clearly outlined the purpose of the study, the voluntary nature of participation, participants' rights to withdraw at any stage without penalty, measures to ensure confidentiality and anonymity and compliance with the Protection of Personal Information Act (POPIA). POPIA was used to ensure data protection from data collection to data disposal all the 8 tenets of POPIA were observed.

Only students who signed and returned the informed consent form were permitted to participate in the focus group session, thereby ensuring ethical compliance and respect for participant autonomy.

3.5.2 Inclusion and exclusion criteria

3.5.2.1 Inclusion criteria

- Participants are university students aged 18 and above.
- Participants are capable of giving informed consent.
- Participants must be enrolled in or have completed a Java programming course.
- Participants must be in IT-related programs such as computer science or information systems.
- Have a basic understanding of English
- Language of data collection is English.

Exclusion criteria

- Participants who are complete beginners with no prior programming experience.
- Participants not enrolled in or who have not completed a Java programming course.
- Participants who are non-IT students without foundational programming knowledge.
- Vulnerable populations, such as minors under 18, who may not provide informed consent effectively.
- Participants who do not understand English sufficiently to engage with the study process.

3.6 Data collection instruments and procedure

3.6.1 Data collection instruments

The primary data collection instruments include questionnaires and focus groups. The questionnaires combine structured Likert-scale items with open-ended questions allowing the capture of measurable responses as well as detailed qualitative feedback. The constructs

assessed through the questionnaire include usability, motivational elements and feature relevance. The Usability items were used to evaluate participant perceptions of the interface, including ease of navigation, clarity of layout and overall intuitiveness. The motivational elements focus on preferences for gamification mechanics such as points, badges, leaderboards and other engagement-enhancing features, while feature relevance items determine the perceived importance and necessity of each proposed functionality for supporting Java learning. The structured items use a 5-point Likert scale ranging from 1 (Strongly Disagree) to 5 (Strongly Agree) while open-ended items encourage participants to provide detailed explanations, suggestions and insights that may not be captured through quantitative measures.

The focus group sessions provided an in-depth qualitative exploration of participant experiences and perceptions. These sessions were guided by a semi-structured discussion protocol in order to ensure that the key areas are addressed while allowing flexibility for emergent themes to arise (Creswell & Plano Clark, 2021). The discussions focused on participant feedback regarding the conceptual prototype layout, navigation and overall interface design, as well as the pedagogical alignment of gamified tasks and activities. Participants were asked to reflect on the emotional and motivational impact of the proposed gamification elements and provide insights into how these features may influence engagement, satisfaction and learning outcomes.

3.6.2 Data collection procedure

Recruitment of participants was conducted through invitations sent via the NWU LMS, following approval from the designated gatekeeper (Annexure A). All participants were required to complete informed consent forms (Annexure B) prior to participation the pool of 20 students received the questionnaire digitally through google forms allowing for anonymous, efficient and structured data collection. The questionnaire captured both quantitative and qualitative insights regarding usability, motivational elements and feature relevance of the conceptual gamified application.

A smaller group of 6 students participated in a focus group session with prior consent for transcription. The discussion followed a semi-structured guide starting with an introduction that explains the study, emphasizes voluntary participation and confidentiality and encourages open sharing. The warm-up questions asked participants to introduce themselves, discuss their experiences learning Java and reflect on any previous use of gamified applications and their impact on motivation. The discussion then covered key areas such as what makes a gamified learning tool effective, potential challenges, preferred gamification elements (e.g., badges, points, leaderboards, rewards, levels) and preferences for individual or team-based features. Participants also provided input on desired features, ways to display progress and achievements and the type of feedback that motivates them most. The session concluded with an opportunity for any

additional suggestions. Feedback from the focus group, together with questionnaire data, were used to refine the application's features, interface and gamification elements, ensuring the final design meets student needs and learning goals.

3.7 Data analysis

The analysis of participant feedback was integrated within an agile development framework which served as the guide for the iterative design of the gamified Java learning application. Agile methodology promotes continuous stakeholder involvement, adaptive planning and incremental development, making it particularly suitable for user-centred and participatory design processes (vom Brocke *et al.,* 2020). Feedback collected through questionnaires and focus groups were systematically analysed after each iteration to identify user needs, preferences and usability issues. These insights informed adjustments to the application's features, interface and gamification elements in subsequent design cycles ensuring alignment with principles of responsive design and evidence-based educational technology (Hamari *et al.,* 2014). The iterative feedback loop facilitates rapid prototyping, early identification of design flaws and continuous alignment with participant expectations and learning objectives (Tondello *et al.,* 2019), while grounding the design in real-world learning contexts that support both motivation and engagement—key components for effective gamified interventions (de-Marcos *et al.,* 2016).

As the study concludes at the development phase, no empirical usage data was collected; however, it presents a well-validated theoretically informed functional artifact of the gamified application prepared for future implementation and testing. This approach aligns with contemporary recommendations in design science research which emphasize the importance of establishing a robust conceptual foundation before full deployment (vom Brocke *et al.,* 2020; Peffers *et al.,* 2007).

3.8 Technical development methodology

In a response to the research aims, a workable software artifact was created using a contemporary and component-based web architecture. Development stack was chosen so that it can be scaled, responsive and with real programming experience to the user. The technical model is decoupled and comprises of a React.js Frontend, a Node.js Execution Server and Firebase Cloud Services.

3.8.1 Frontend architecture (user interface)

The client-side application was developed with the help of React.js which is the declarative JavaScript library of user interface construction. React was selected due to its component-based

nature and it is possible to develop reusable UI elements, like the navigation sidebar, level progression screens and interactive tutorials.

Language and logic: The application logic is implemented in JavaScript (ES6+), with the help of the modern syntax of the asynchronous operations and state management.

Styling: Tailwind CSS is a utility-first framework that has been used to provide a consistent, responsive design system. This provided a professional appearance to the landing pages and in-game interface and did not require custom CSS code to write per component.

Navigation: React router handles routing at the client-side, which allows a Single Page Application (SPA) experience. This enables one to move around game worlds, profiles and coding activities without the page loading up completely.

Code editing environment: The Monaco editor is an essential part of the artifact, which is known as Monaco-editor/react. Being the main driving force of Visual Studio Code, it is an authentic development experience that lets users do Java programming with all the features that this process requires: syntax highlighting, code completion and error detection in a convenient and easy-to-use web-based interface.

API communication: The Axios is utilized in handling asynchronous HTTP requests and forms the communication layer between the front-end interface and the backend execution server.

3.8.2 Backend architecture (code execution engine)

To enable the compilation and execution of Java code within a web environment, a dedicated local server was developed using Node.js and Express.js.

Runtime Environment: Node.js provides the non-blocking I/O model necessary for handling concurrent requests.

Java Compilation: The backend utilises the Node.js child process module. When a student submits code, the server utilizes the file system (fs) to write the code to a temporary .java file. It then spawns a shell command to execute the Java development kit (JDK) tools (javac for compilation and java for execution). The output or error streams are captured and returned to the frontend via the API.

Security: CORS (Cross-Origin Resource Sharing) middleware is implemented to securely manage requests between the frontend (running on the client’s browser) and the local execution server.

3.8.3 Cloud infrastructure and data persistence

The artifact leverages Google Firebase as a Backend-as-a-Service (BaaS) to handle identity management and data persistence, reducing the need for complex server-side infrastructure for non-execution tasks.

Authentication: Firebase Authentication manages user identity, supporting secure sign-in methods and session management. This ensures that student progress is securely linked to their unique profile.

Database: Cloud Firestore, a NoSQL document database, stores persistent data. The database schema is structured into two primary collections:

Users Collection: Stores player XP, current level, unlocked badges and progress history.

Worlds Collection: Stores the static content of the game, including level descriptions, tutorial text and quiz questions.

3.8.4 Development environment

The artifact was developed using standard industry tooling to ensure code quality and maintainability.

Version Control & Dependency Management: npm (Node Package Manager) was used to manage libraries and dependencies for both the frontend and backend environments.

IDE: Visual Studio Code (VS Code) served as the primary development environment.

Bootstrapping: The project was initialized using Create React App, which pre-configures build tools such as Webpack and Babel, streamlining the development process.

3.9 Pragmatic rigour

In design science research, rigor extends beyond methodological precision to include practical value and societal responsibility. the principles that guide pragmatic rigor are ethical reasoning, relevance, comprehensibility, and actionability (Robey *et al.,* 2018).

Ethical reasoning refers to the degree to which research considers the full range of stakeholders affected and equitably evaluates the consequences of design decisions (Robey *et al.,* 2018). In this study, ethical considerations were embedded in the artifact's design by prioritizing student well-being and reducing unnecessary performance pressure. The system emphasizes supportive feedback and structured progression to ensure that learning benefits all users.

Relevance concerns the connection between research and problems that stakeholders judge to be important (Robey *et al.,* 2018). The study directly addresses the challenge of low engagement and high attrition in introductory programming courses. A problem identified in Java programming education for IT students. By aligning the artifact's objectives and features with these expressed needs, the research maintains strong practical significance.

Comprehensibility refers to the extent to which research findings are communicated at a level appropriate to the intended audience (Robey *et al.,* 2018). The study presents the design rationale, system architecture, and feature implementation in clear and accessible language. By explicitly linking design decisions to identified student needs, the research ensures that educators, developers, and institutional stakeholders can readily understand and apply the findings.

Actionability refers to the extent to which research findings can be implemented in organizational settings through practical interventions (Robey *et al.,* 2018). This study demonstrates actionability by delivering a functional web-based application built using widely adopted technologies, supported by detailed documentation of the design and development process. This enables replication, institutional deployment, and future refinement.

By integrating these four principles, the study ensures that the gamified application is ethically responsible, practically relevant, clearly communicated, and ready for implementation within higher education contexts.

3.10 Ethical considerations

Participation in the study is voluntary and confidential. For the questionnaire phase, responses were fully anonymous with no personally identifying information collected. For the focus group phase, participants were assigned unique identification codes to protect their identities in all reports and publications. All data were handled in compliance with NWU ethical requirements and national privacy regulations (POPIA, 2013). Throughout the process, stakeholder rights including the right to withdraw, decline participation, or withhold information were respected and upheld.

3.10.1 Permission and informed consent

Ethical clearance for this study was obtained from the relevant North-West University (NWU) Ethics Committee, ensuring full compliance with institutional and legal standards (Kvale & Brinkmann, 2015). As a prerequisite for commencing research, such clearance verifies that all procedures adhere to established ethical norms and safeguards. Permission was secured from the relevant NWU departments, with the official letter of approval attached as Annexure B.

Participants received a comprehensive informed consent form (Annexure C) outlining the study's objectives, methodology, voluntary participation, rights to withdraw at any time without penalty and confidentiality measures, in line with best practices for informed consent (Creswell & Plano Clark, 2021; Krosnick, 2018). The consent process ensures participants fully understand what their involvement entails and the protections in place.

The study presents minimal risk and no direct financial incentives, maintaining ethical integrity and there is no conflict of interest, as the researcher holds no hierarchical or evaluative relationship with participants, ensuring unbiased data collection and interaction (Patton, 2015). Indirect benefits include contributing to research that aims to improve Java programming education through gamification, potentially enhancing learning outcomes for future students and informing curriculum design (Creswell & Creswell, 2023). Data confidentiality and participant anonymity were strictly maintained, with all information managed according to POPIA regulations, guaranteeing secure handling and protection of participant data throughout the research process.

3.10.2 Anonymity and confidentiality

Anonymity was upheld during the study at all times, so no participants' personally identifiable information associates with any collected data from them (Kvale & Brinkmann, 2015). Every participant received a singular identification number for data collection and analysis, confirming that their identity goes unidentified. Patton (2015) points out that this technique is a key part of ethical education research because it allows students to communicate their opinions freely, without anxiety about negative responses.

All data collected was treated with confidentiality. The information obtained from participants, including survey responses, was securely stored in encrypted digital files, accessible only to the research team (Kvale & Brinkmann, 2015). Data sharing with supervisors or other researchers were done in aggregate form to avoid revealing individual participant details (Guest *et al.,* 2012). Questionnaire data does not identify the participants in any way. Interview data were kept under lock and key. The data will be retained for a minimum of five years after qualification completion and thereafter destroyed, in accordance with NWU Research Data Management Policy (NWU, 2021). The data were used to develop an artifact therefore none of the collected information were seen. A summary of activities was created based on the data collected and this is aggregate data. Moreover, all findings presented in publications or presentations ensure that individual responses cannot be traced back to any specific participant. The privacy of the students' data was maintained in keeping up with the POPIA. The researcher handled data responsibly, support ethical and lawful research practices. The data were collected were analysed for the purposes of

this research mini-dissertation and the results may also be used in publications that arises out of this study.

3.11 Chapter summary

This chapter detailed the research methodology and design adopted, specifically using the DSR approach within a pragmatic research paradigm. The study's strategy involves the iterative design and creation of a conceptual gamified application, guided by the DSR process phases. The chapter defined the study context, the target population of undergraduate IT students and the purposive sampling strategy. Detailed plans were provided for the use of questionnaires and focus groups as data collection instruments, followed by a clear outlining of descriptive statistics and thematic analysis as the analytical techniques. Finally, the chapter ensured the research quality through validity measures and confirmed the ethical compliance with NWU and POPIA standards establishing a robust methodological foundation for the remainder of the study. The next chapter is the data analysis, design and implementation.

DATA ANALYSIS, DESIGN AND IMPLEMENTATION

4.1 Introduction

This chapter presents the empirical analysis, system design and implementation of the gamified web-based application for Java programming education, titled JavaRealm. Building on the design science research (DSR) methodology outlined in Chapter 3 (Hevner, 2007), this chapter serves two primary purposes to address the study's research objectives.

Firstly, it reports the empirical findings derived from the questionnaire survey (N = 20) and the focus group interview (N = 6) conducted with Information Technology (IT) students at North-West University (NWU). These findings provide the descriptive knowledge base required to understand the specific contextual constraints and motivational drivers of the target population. Secondly, it demonstrates the "design and development" phase of the DSR cycle (Peffers *et al.,* 2007), showing how these empirical insights were systematically translated into system requirements, architectural decisions and concrete gamification mechanics.

In accordance with DSR principles, this chapter closes the research loop between problem identification, user feedback and artifact construction. Section 4.2 presents the quantitative and qualitative findings. Section 4.3 synthesises these into a formal Requirements Traceability Matrix. Section 4.4 details the system architecture and database design. Section 4.5 presents the implemented functional artifact and Section 4.6 concludes the chapter.

4.2 Presentation of empirical findings

4.2.1 Demographic profile of participants

Following the exclusion of respondents with no prior exposure to Java programming to ensure data validity, a total of N = 20 participants were retained for the quantitative analysis. As presented in Table 4.1, the sample represents a diverse cross-section of academic maturity, with a strong representation of senior and postgraduate students.

Table 0.1: Demographic profile of participants (N = 20)

Variable	Category	Frequency (*n*)	Percentage (%)
Age	18–24	12	60
	25–29	7	35

	30+	1	5
Gender	Female	7	35
	Male	10	50
	Non-binary/Other	3	15
Academic Level	1st/2nd Year	5	25
	3rd Year	7	35
	4th Year/Postgrad	8	40

The inclusion of senior students (75% are 3rd year or higher) is significant for the validity of the artifact design. These participants have navigated the "bottleneck" modules of the Java curriculum and provided retrospective insights into where technical scaffolding is most needed.

4.2.2 Quantitative findings (questionnaire analysis)

The questionnaire employed five-point Likert scale items to measure student perceptions regarding usability, motivational affordances and social learning preferences. Descriptive statistics, specifically Mean (*M*) and Standard Deviation (*SD*), were calculated to identify areas of consensus and variance. Given the sample size (N = 20) and the study's design-and-creation focus, this analysis is intentionally descriptive rather than inferential: no significance testing or causal modelling was attempted, as the objective at this stage is to characterise user preferences to inform artifact design, not to test hypotheses about learning outcomes.

Table 0.2: Key quantitative findings

Construct / Statement	Agree/Strongly Agree (%)	Mean (*M*)	Standard Deviation (*SD*)	Interpretation
Feedback: "Receiving immediate feedback would help me stay engaged."	95	4.80	0.52	High Consensus: Critical Requirement

Usability: "A gamified app should be simple and easy to use."	95	4.75	0.55	Strong Agreement
Collaboration: "Team rewards would motivate me more than competition."	80	4.20	1.06	Moderate Agreement
Competition: "Competition with peers would increase my motivation."	80	4.10	1.02	Notable Variance: Minority Dissent

Two key insights emerged from this data:

1. **Immediate feedback is non-negotiable:** With the highest mean score (*M* = 4.80) and the lowest standard deviation (*SD* = 0.52), the demand for instant feedback is the strongest signal in the dataset. This mandates a technical architecture capable of real-time code compilation.
2. **The competition paradox:** While the concept of competition received a positive mean (*M* = 4.10), responses showed a comparable spread to collaboration (*SD* = 1.02 vs. *SD* = 1.06). Both constructs included a small minority of clearly negative responses alongside a majority of positive ones, suggesting that while most students respond positively to both competitive and collaborative mechanisms, a consistent minority find competition specifically less motivating or potentially demotivating. This finding aligns with the qualitative data discussed in Section 4.2.3, suggesting that competitive features must be flexible rather than mandatory.

Inferential analysis of questionnaire responses

While the descriptive statistics above identify areas of consensus and variance, they cannot on their own support claims about whether responses differ meaningfully between groups of respondents. Given the ordinal nature of five-point Likert-scale data and the modest, unequal group sizes in this sample, non-parametric tests were used in preference to parametric alternatives such as ANOVA or the independent-samples t-test, consistent with established guidance for ordinal data (Field, 2017).

Respondents' self-rated Java programming experience (Beginner, n=7; Intermediate, n=8; Advanced, n=4) was used as the grouping variable, as it was the only categorical variable in the dataset with three or more groups of workable size and a clear theoretical link to the constructs being measured. A series of Kruskal-Wallis H tests was run comparing all three experience groups across each of the eight constructs, with follow-up Mann-Whitney U tests run pairwise where an overall difference was detected.

The Kruskal-Wallis test found a statistically significant difference between experience groups for the Motivation_Consistency construct ("Gamified features… would motivate me to learn more consistently"), $H(2) = 8.18$, $p = .017$, with a large effect size (epsilon-squared = 0.454). Median responses were 5 for Beginners, 4 for Intermediate students, and 3 for Advanced students, indicating that self-rated motivational benefit from gamified features decreases as programming experience increases. A follow-up Mann-Whitney U test confirmed this was driven primarily by the difference between Beginner and Advanced respondents specifically, $U = 27.0$, $p = .011$, with a large effect size ($r = -0.929$); the Beginner–Intermediate comparison approached but did not reach significance ($U = 44.0$, $p = .053$), and the Intermediate–Advanced comparison was not significant ($U = 23.0$, $p = .228$).

No other construct showed a statistically significant difference across experience groups at alpha = .05: Engagement ($H(2) = 1.92$, $p = .383$), Competition ($H(2) = 3.22$, $p = .200$), Collaboration ($H(2) = 0.68$, $p = .713$), Feedback ($H(2) = 1.99$, $p = .371$), Rewards ($H(2) = 4.02$, $p = .134$), Usability ($H(2) = 0.40$, $p = .821$), and Intent to Use ($H(2) = 1.04$, $p = .595$).

This finding adds a more precise, statistically supported claim to what was previously a purely descriptive observation: it is specifically the expectation that gamified features will sustain consistent motivation rather than overall engagement, perceived usability, or intent to use that differs meaningfully by experience level, and the effect is concentrated at the extremes of the experience range (beginner versus advanced) rather than appearing as a smooth gradient. Given the small cell sizes involved, particularly for the Advanced group (n=4), this result should be treated as a precise description of this sample rather than a generalisable population claim, consistent with the sample-size limitations acknowledged in Section 3.11.

4.2.3 Qualitative findings (focus group thematic analysis)

An inductive thematic analysis was conducted on the focus group transcript following the six-phase approach proposed by Braun and Clarke (2022). Six participants (P1–P6) participated in the session. As illustrated in Table 4.3, four dominant themes emerged from the analysis. The

table outlines the definition of each theme, the sub-themes identified and representative quotes from the participants.

Table 0.3: Thematic analysis of focus group data

Theme	Definition	Sub-themes	Representative Quotes
1. Technical Performance & Reliability	The impact of system latency and network accessibility on user adoption and frustration.	• Latency intolerance • Snappiness	*"If I click 'Run Code' and it lags for ten seconds… I'm closing the tab. It needs to be snappy."* (P4)
2. Meaningful vs. Superficial Learning	The preference for authentic, industry-standard tools over simplified "toy" interfaces.	• Authentic practice • Cognitive scaffolding • "Small chunks" approach	*"I don't want to just click buttons; I want to understand the code."* (P5) *"If it breaks Java down into small chunks—like levels—it won't feel so overwhelming."* (P1)
3. The Competition Paradox	The tension between students who are motivated by ranking and those who experience anxiety from it.	• Leaderboard anxiety • Competitive drive • Collaborative preference	*"Put in a leaderboard. I want to see if I'm beating [peers]."* (P2) *"Leaderboards give me anxiety. I prefer collaboration."* (P6)
4. Visual & UX Preferences	The requirement for a professional aesthetic that signals legitimacy and reduces eye strain.	• Dark mode necessity • Professional identity • VS Code familiarity	*"It needs to feel like VS Code... Dark mode. Please."* (P4) *"Don't make it look childish."* (P5)

4.2.4 Discussion of findings and literature integration

The findings presented in Table 4.3 and Table 4.2 align with established educational technology theories, providing strong empirical justification for the design decisions embedded in the proposed artifact.

Performance and feedback

The overwhelming demand for "immediate feedback" (Quantitative M = 4.80; Theme 1) highlights the importance of rapid system responses in maintaining deep learner engagement. When feedback is delivered instantly, students can quickly identify errors, adjust their understanding, and remain focused on the task without losing momentum. Furthermore, the rejection of "laggy" interfaces supports the Technology Acceptance Model (TAM), where Perceived Ease of Use is heavily influenced by system responsiveness (Davis, 1989). Slow or delayed feedback can disrupt concentration and negatively affect users' perceptions of the platform's usability and effectiveness.

Authenticity and scaffolding

Theme 2's request for "small chunks" and authentic editors aligns with Cognitive Load Theory. By breaking complex Java syntax into "Quests," the artifact manages intrinsic cognitive load. The preference for real code editors over drag-and-drop blocks suggests that IT students view tool authenticity as a proxy for professional preparedness, consistent with the broader emphasis on practical relevance in IS research (Oates *et al.,* 2022).

Social dynamics

The "Competition Paradox" (Theme 3) While competition satisfies *Competence* for some (P2), it threatens the *Autonomy* and emotional safety of others (P6). This supports the broader argument that gamification affects users differently depending on context and individual differences (Hamari *et al.,* (2014), and aligns specifically with personality-based gamification design principles (Tondello *et al.,* 2019), necessitating a configurable design approach (e.g., Opt-in Leaderboards).

4.3 System requirements specification

By synthesising the quantitative variance regarding competition and the qualitative demand for authenticity and performance, the formal system requirements were defined. Table 4.4 presents the Requirements Traceability Matrix, which links each technical feature back to the empirical evidence that justified its inclusion.

Table 0.4: Requirements traceability matrix

ID	Requirement	Rationale (Evidence Source)
FR-01	Authentic Code Editor: System must support syntax highlighting and line numbers.	Theme 4 (P4's expectation of a VS Code-like environment).
FR-02	Immediate Feedback: System must return compilation results in < 2 seconds.	Table 4.2 (95% Agreement on feedback).
FR-03	Quest Structure: Topics must be presented as sequential "Quests" via a Skill Tree.	Theme 2 (P1's request for "small chunks").
FR-04	Plain-English Parsing: Compiler errors must be annotated with beginner-friendly hints.	Theme 2 (P5's frustration with cryptic errors).
FR-05	Streak Tracking: System must track daily activity to encourage habit formation.	Theme 2 (P2's comparison to Duolingo).
FR-06	Configurable Social: Leaderboards must be opt-in or anonymous.	Theme 3 (P6's anxiety vs. P2's drive).
NFR-01	Performance: Code execution must occur asynchronously.	Theme 1 (Intolerance for lag).
NFR-02	Accessibility: Must function on Eduroam (Port 80/443).	Theme 1 (Network constraints).

4.4 System design and architecture

4.4.1 Technical architecture

To satisfy the identified Non-Functional Requirements (NFRs), particularly regarding performance and accessibility, the system employs a decoupled architecture utilising the MERN stack (MongoDB/Firebase, Express, React, Node).

1. Presentation Layer (React.js): This framework was selected for its component-based structure, allowing for reusable UI elements such as the "Quest Card". It ensures a responsive Single Page Application (SPA) experience, minimising page reloads.
2. Logic Layer (Node.js/Express): Critical for satisfying NFR-01, the Node.js runtime utilizes a non-blocking I/O model. This enables the server to handle multiple concurrent code

compilation requests asynchronously, preventing the system queuing delays ("lag") that participants identified as a barrier to usage.

3. Data Layer (Firebase): Google Firestore (NoSQL) was selected to allow for flexible schema evolution as gamification features expand.

4.4.2 Gamification mechanics design

The mechanics were designed to balance the tension between competition and mastery identified in the empirical findings:

- XP Algorithm: A linear progression formula was implemented to ensure frequent early level-ups, aimed at building self-efficacy for beginners.
- Zeigarnik Badge System: Leveraging the Zeigarnik effect (the psychological tendency to remember and feel compelled to complete unfinished tasks), unearned badges are displayed as silhouettes to induce curiosity. Crucially, badges are awarded for pedagogical milestones (e.g., "First Loop Created") rather than arbitrary interaction, ensuring rewards signal actual competence.

4.4.3 Database design (Firestore schema)

To support the flexible nature of the gamified curriculum, the application utilises Google Cloud Firestore, a NoSQL document-oriented database. Unlike rigid relational schemas, this structure allows for the seamless addition of new gamification metadata (e.g., new badge types or quest parameters) without requiring database migration.

The data model is structured into two primary root collections: users (storing player progress) and worlds (storing static curriculum content).

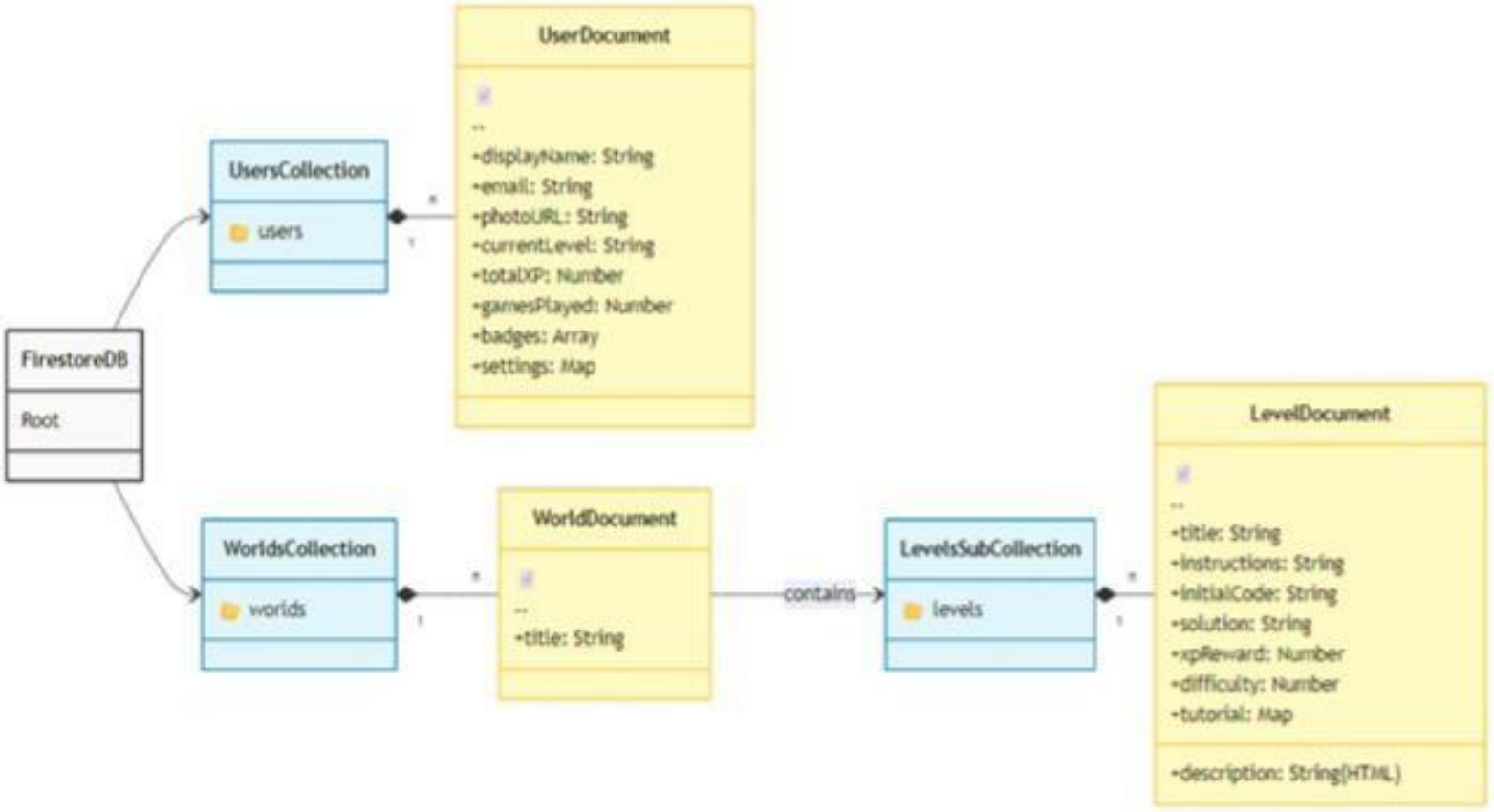


Figure 4.1 The NoSQL cloud Firestore schema design.

Figure 4.1 illustrates this schema design.

Users Collection: This collection stores the state of every student. Key fields include “currentLevel” (used to control access to the Skill Tree), “totalXP” (used for Leaderboard ranking) and the badges array (storing unlocked achievements).

Worlds and Levels Collection: The curriculum employs a hierarchical structure. The worlds collection represents broad topics (Sectors). Each world document contains a sub-collection named levels, which holds the specific programming challenges, solution validators and tutorial content. This separation ensures that the application loads only the necessary level data, optimizing bandwidth for students on mobile networks (NFR-02).

The structure utilises sub-collections for 'levels' to organize curriculum content efficiently, while user progress is denormalised in the 'users' collection to ensure fast read-access for the leaderboard.

4.5 Implementation of the artifact: JavaRealm

4.5.1 The landing interface and narrative

When a user accesses the application, they are presented with a "Professional/Cyber" interface implemented in Home (see Figure 4.2). The use of dark mode, glowing grid lines and floating 3D icons (<Terminal />, <CPU />) establishes an environment that feels like a professional developer tool rather than a toy. This directly satisfies Theme 4 (Visual and UX Preferences), where participants rejected "childish" designs in favour of an industry-aligned aesthetic.

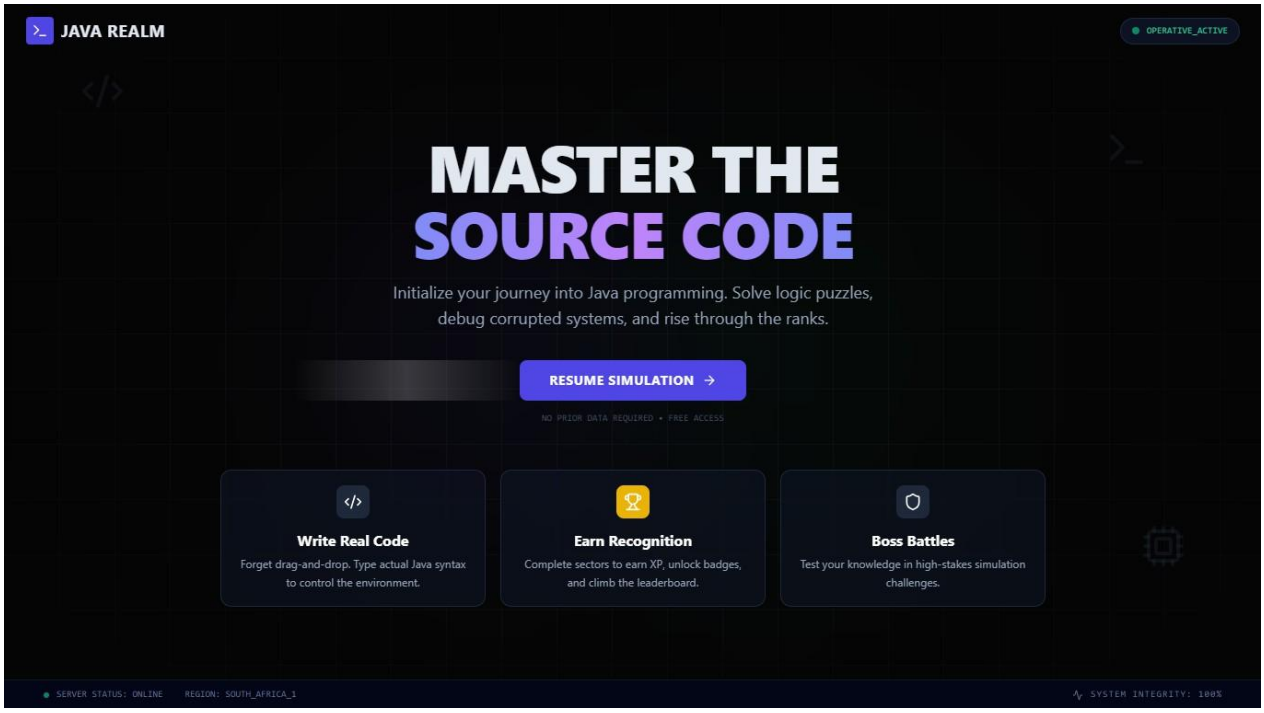


Figure 4.2 The landing page (Home Page)

Figure 4.2 features the dark-mode aesthetic and cyber-security motifs establish the professional tone requested by the focus group participants. Access on: [view the prototype]



Figure 4.3 The landing page (Home Page) code

4.5.2 The dashboard: sectors and progression

Upon logging in, the user encounters a Sector-based grid layout implemented in “WorldMap” (see Figure 4.4). The curriculum is broken down into manageable "Sectors" (e.g., Sector 01), each containing sequential level cards and a progress bar. This visual structure satisfies Theme 2, where participants like Participant 1 requested a "map of the module" to break Java down into "small chunks" and reduce cognitive overload (scaffolding).

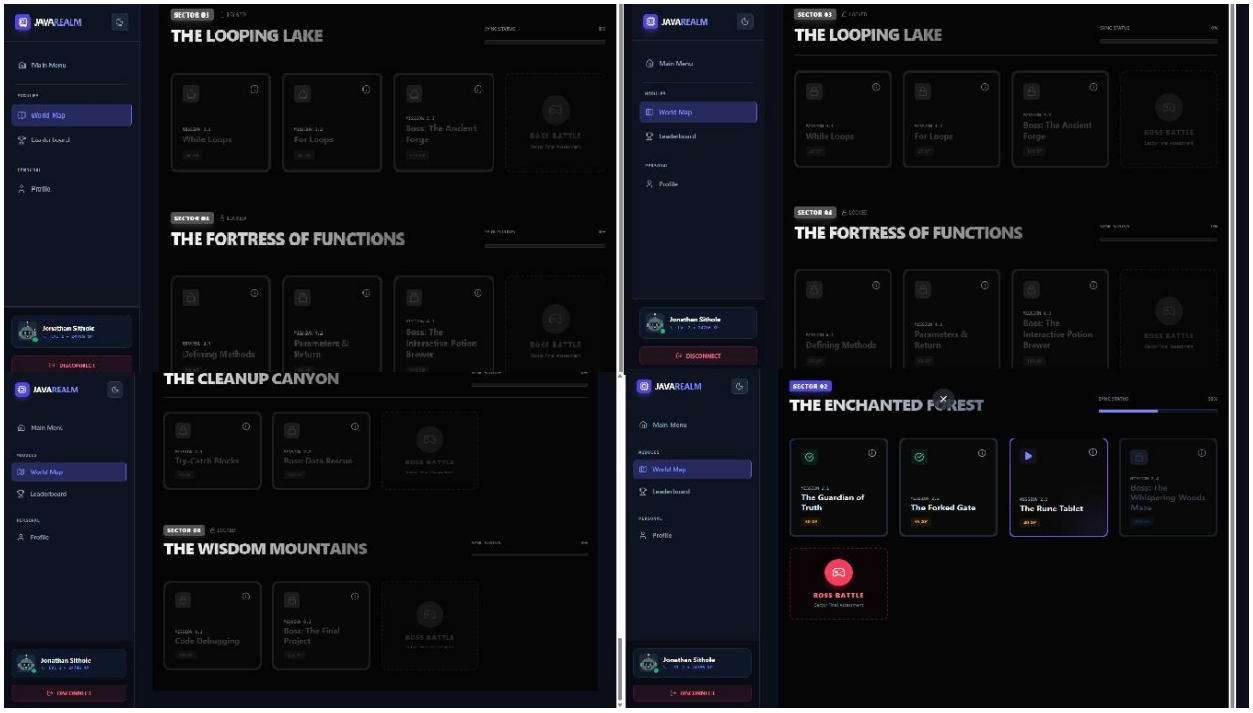


Figure 4.4 The ‘World Map’

Figure 4.4: The “World Map” showing Sector 01. Note the progression bar (top right), the grid of level cards and the distinct “Boss Battle” node at the end of the sequence. Access on: [view the prototype]

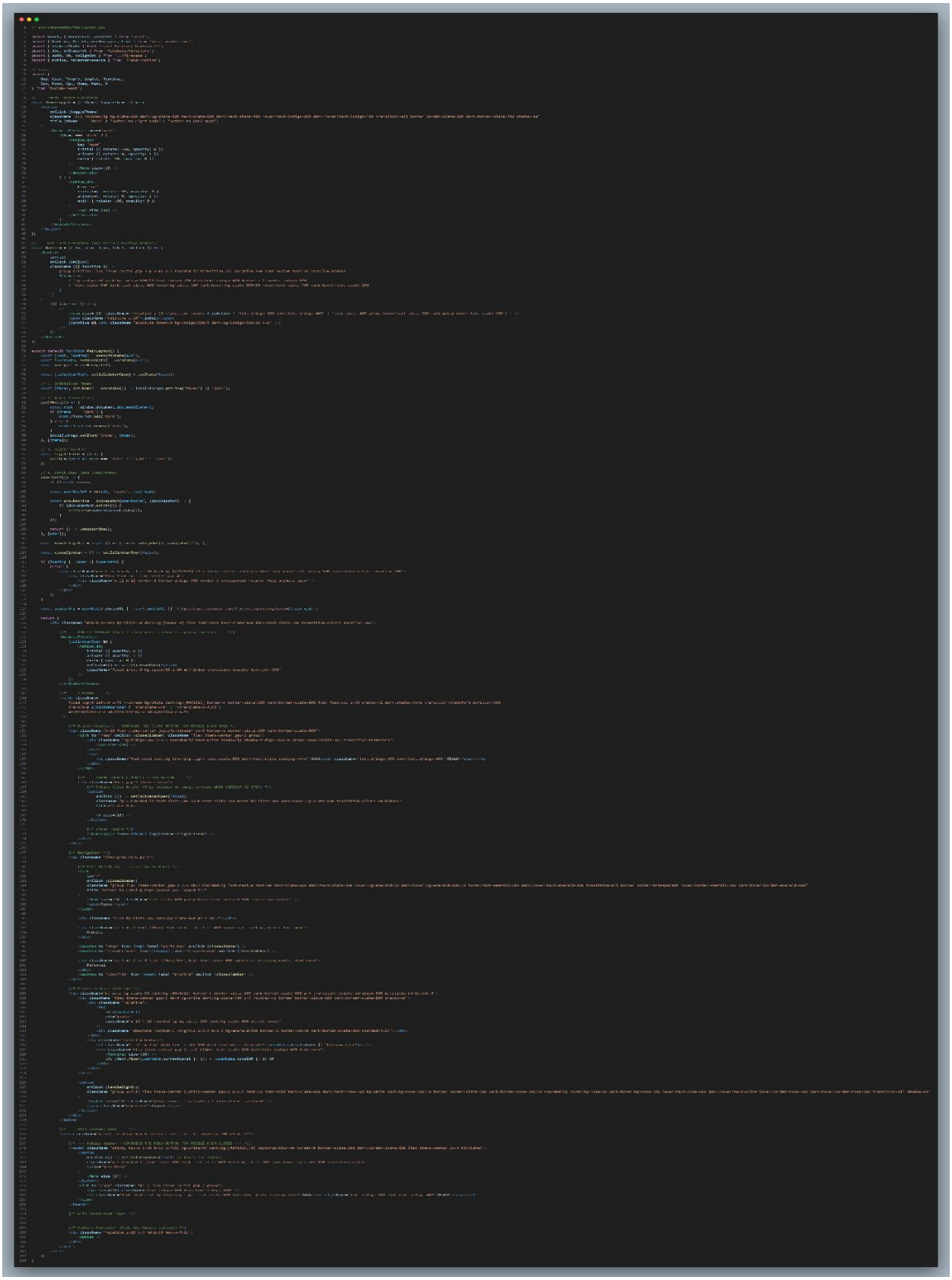


Figure 4.5 Main layout and world map code

4.5.3 The coding environment: authentic development

When a level is selected, the application loads the authentic coding interface implemented in Level (see Figure 4.4). The interface utilizes the *Monaco Editor* engine in the right pane, mimicking Visual Studio Code with syntax highlighting and line numbers. This design decision satisfies FR-01 and explicitly addresses Theme 2, where Participant 5 stated, "I don't want to just click buttons; I want to understand the code." Additionally, the inclusion of an "Intel" tab in the sidebar provides the necessary instructional support requested during the focus group.

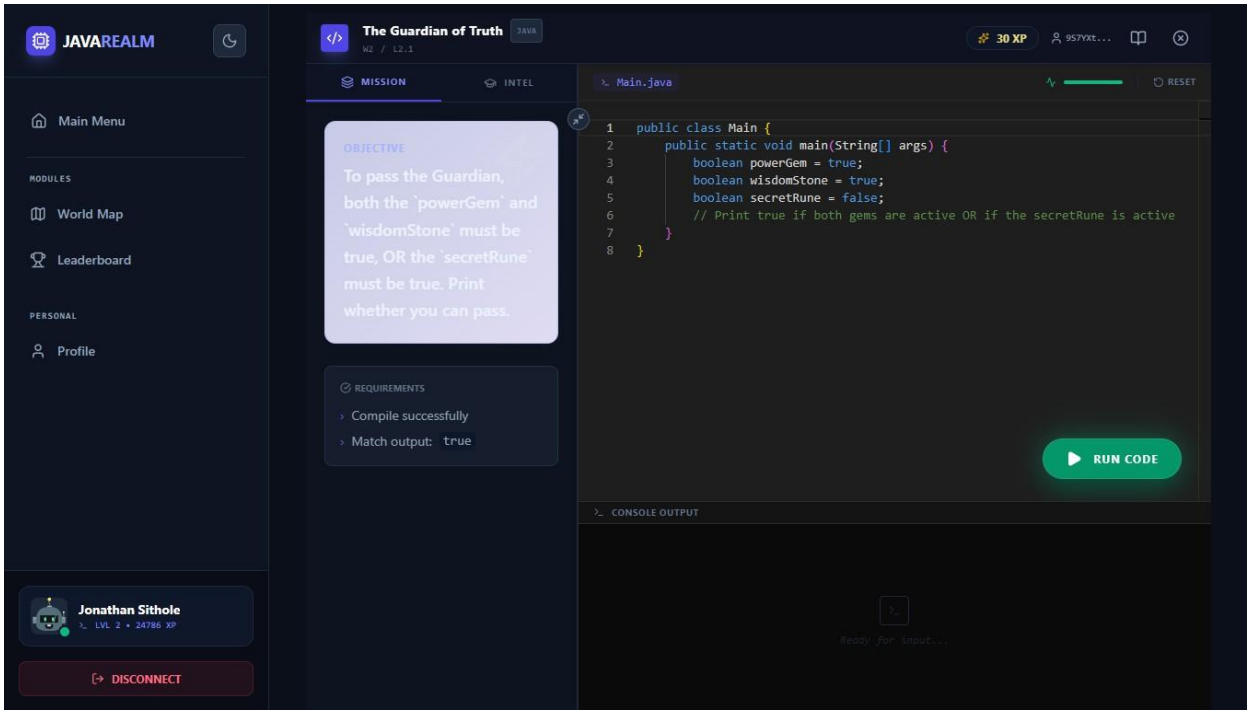


Figure 4.6 The coding interface

Figure 4.6 shows the primary coding interface. The left sidebar displays the "Mission Objective" and 'Intel' tabs. The centre pane features the Monaco Code Editor with syntax highlighting. The bottom pane shows the console output. Access on: [view the prototype]

4.5.4 Feedback and rewards system

When a user successfully compiles code, a "Mission Complete" modal appears accompanied by an animated XP bar and confetti effects, implemented in "Generic Success" (see Figure 4.5). This mechanism satisfies the 95% of users who demanded immediate feedback (see Table 4.2) and provides the positive reinforcement "dopamine hit" requested by participants during the focus group to sustain engagement.

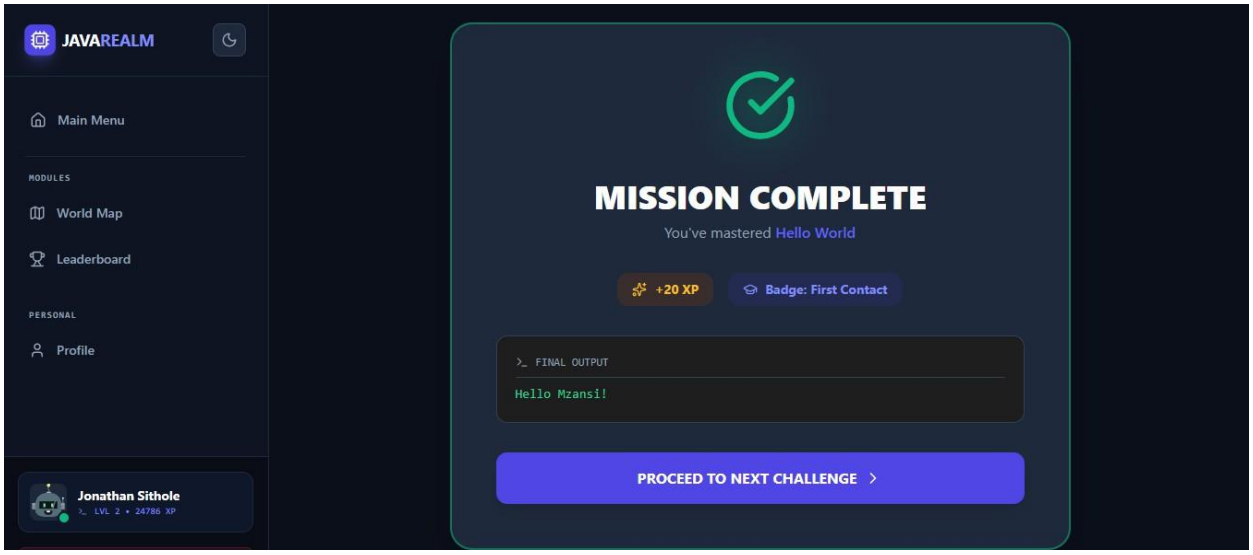


Figure 4.7 The 'mission complete' modal

Figure 4.7: The 'Mission Complete' modal. This screen appears immediately after successful code execution, utilizing confetti animations and clear XP summaries to provide positive reinforcement. Access on: [view the prototype]

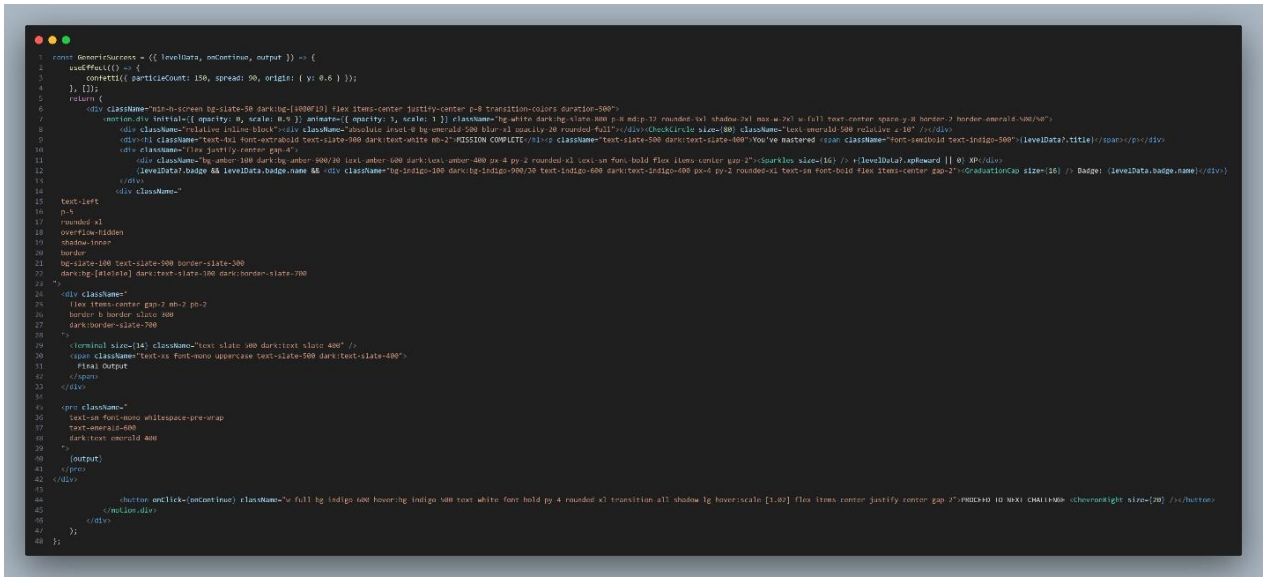


Figure 4.8 The 'mission complete' modal code

4.5.5 Social and competitive elements

When users navigate to the rankings via “LeaderboardPage”, they are presented with a tiered list that highlights their own standing via a "Your Rank" card (see Figure 4.6). By placing this feature on a separate page and emphasizing personal growth over global comparison, the system

satisfies Theme 3, addressing the "Competition Paradox" by allowing competitive students to track their status without inducing anxiety in those who prefer collaboration.

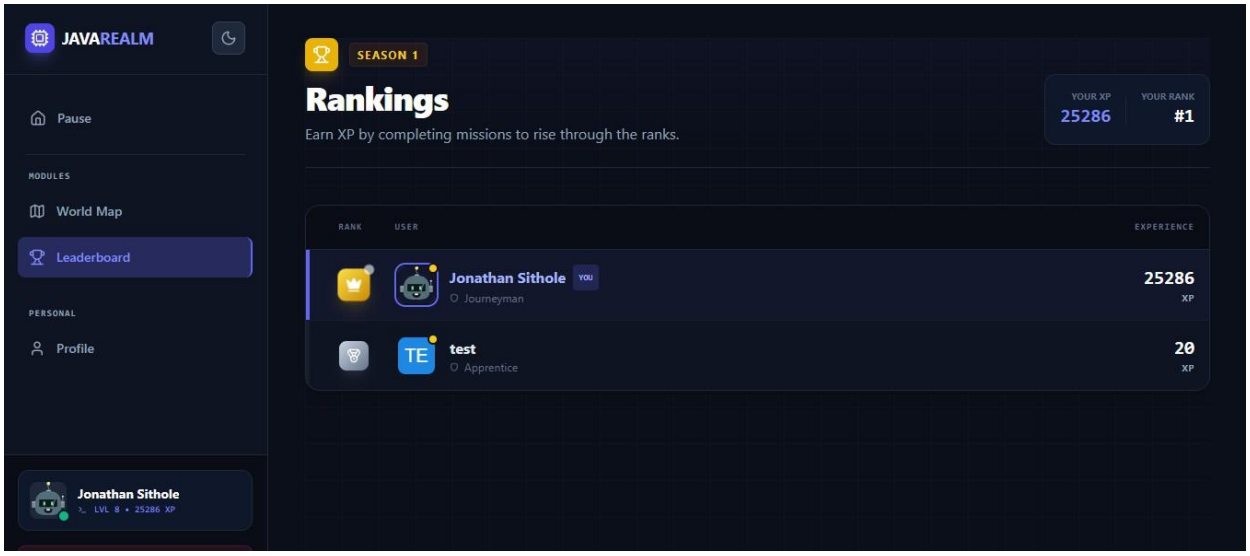


Figure 4.9 Leaderboard page

Figure 4.9: The Leaderboard interface. Note the 'Your Rank' card at the top, which allows students to track personal growth without necessarily scrolling through the entire competitive list. Access on: [view the prototype]



Figure 4.10 Leaderboard page code

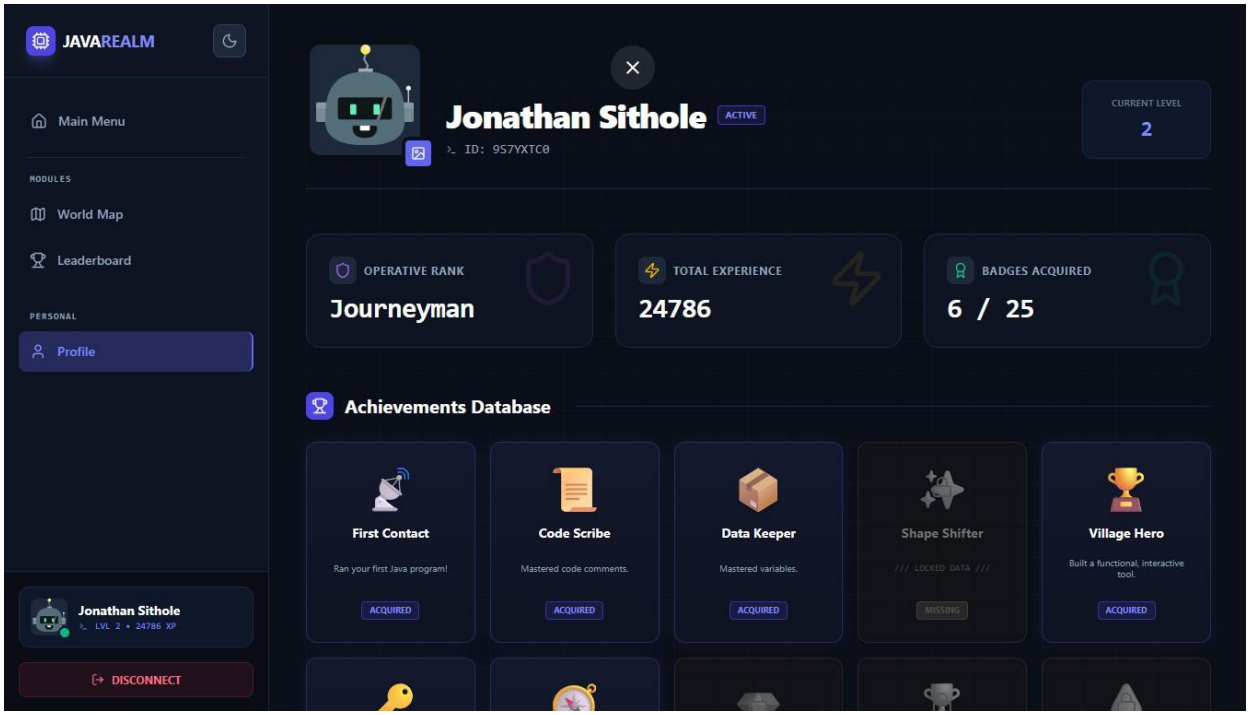


Figure 4.11 The profile page

Figure 4.11: The profile interface. The User Profile serves as a centralized dashboard for tracking long-term progress. To satisfy the motivational requirement for status and recognition (Theme 2), the top section displays the user's "Operative Rank" (e.g., Journeyman) and total Experience Points (XP). The "Achievements Database" below implements the Zeigarnik Badge System discussed in Section 4.4.2; earned badges (e.g., "First Contact") are fully illuminated, while unearned badges (e.g., "Shape Shifter") remain visible but "locked," inducing curiosity and motivating students to complete the necessary learning milestones to unlock them. Access on: [view the prototype]



Figure 4.12 Profile page code

4.5.6 Advanced gamification: boss battles

When a student completes all regular levels in a sector, they unlock a "Boss Battle" simulation. Unlike standard text-based challenges, these components require students to write Java logic to control a visual avatar in a high-stakes environment. This feature addresses the desire for "meaningful gamification" (Theme 2) by demonstrating the practical application of abstract code concepts in a visual, interactive manner.

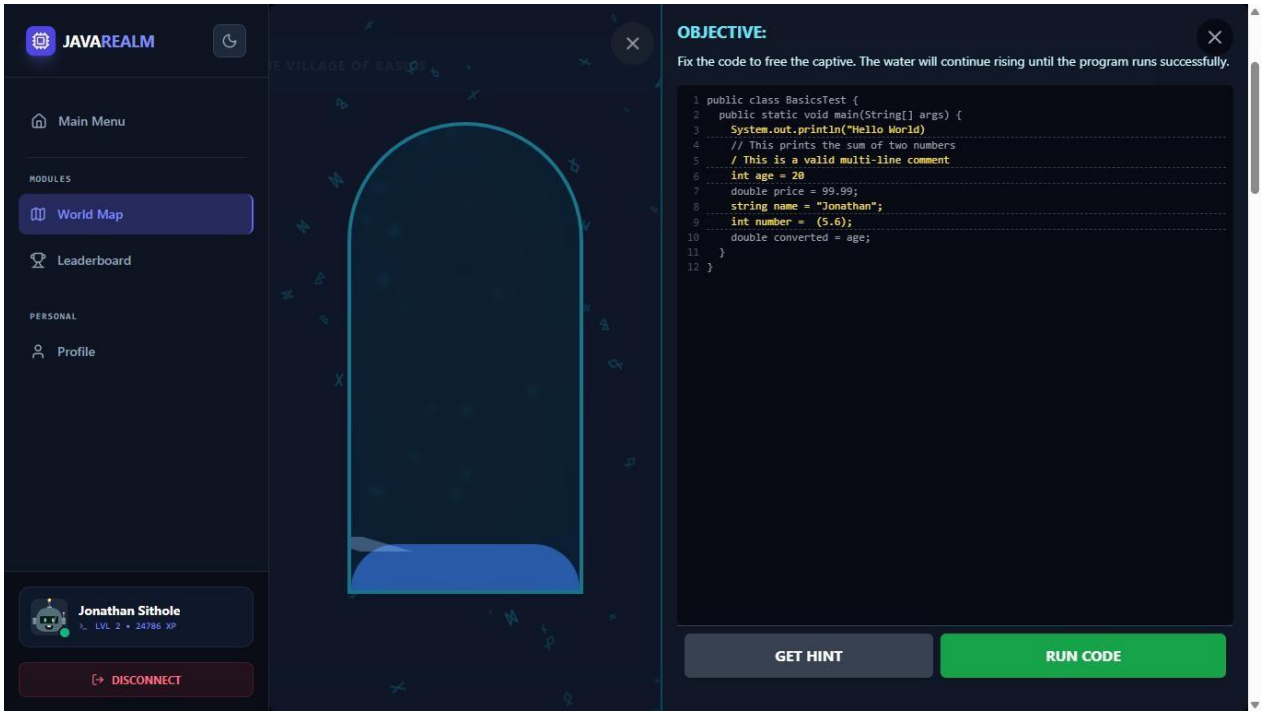


Figure 4.13 Example of boss battle

Figure 4.13: An example of a 'Boss Battle' simulation. Students must write correct Java logic to navigate the avatar through the challenge, visualizing the practical application of their code. Access on: [view the prototype]

4.6 Chapter summary

This chapter provided a detailed account of the empirical investigation and its systematic translation into the design and implementation of the "*JavaRealm*" artifact. The quantitative data and qualitative insights revealed clear student priorities regarding technical performance, meaningful practice and the need for "safe" social motivation. These findings were formalized into a Requirements Traceability Matrix, which directly informed the technical architecture and the selection of gamification mechanics. The resulting artifact is theoretically grounded and empirically aligned with the needs of NWU students, providing the foundation for the evaluation and discussion presented in Chapter 5.

CONCLUSION AND RECOMMENDATIONS

5.1 Introduction

This study set out to address the challenges of engagement and attrition in Java programming education at Higher Education Institutions (HEIs). Motivated by the high failure rates and student perceptions of Java as difficult and boring, the research aimed to design and develop a gamified web-based application, "*JavaRealm*," to enhance student motivation and understanding. Following the design science research (DSR) methodology, the study has progressed through problem identification (Chapter 1), theoretical grounding (Chapter 2), methodological planning (Chapter 3) and the design and implementation of the artifact based on empirical data (Chapter 4). This final chapter advances the DSR cycle by conducting a design verification of the artifact against the requirements established in Chapter 4, and by establishing the foundation for future summative evaluation. It is explicitly acknowledged that the DSR cycle is not fully closed at this stage — the summative evaluation measuring real-world learning outcomes is identified as the immediate next research priority. It discusses the findings in relation to the initial Research Objectives (ROs), outlines the theoretical and practical contributions of the study, acknowledges limitations and provides recommendations for future research.

5.2 Evaluation of the artifact

In DSR, evaluation is crucial to demonstrate the utility, quality and efficacy of the design artifact (Peffers *et al.,* 2007). Given the design and creation scope of this project a descriptive evaluation method was employed. This involves a structural assessment to verify that the final functional artifact "*JavaRealm*" successfully implemented the Functional (FR) and Non-Functional Requirements (NFR) derived from the student data in Chapter 4.

5.2.1 Verification against requirements

Table 5.1 presents the artifact verification matrix, mapping the requirements identified in Chapter 4 to the specific features implemented in the final build.

Table 0.1: Artifact verification matrix

ID	Requirement (From Empirical Data)	Implemented Feature (Artifact Evidence)	Status
FR-01	Authentic Code Editor: System must	Monaco Editor Integration: The artifact embeds the Visual Studio Code engine, providing real-	Met

	support syntax highlighting and line numbers.	time syntax highlighting and line numbering (Refer to Figure 4.3).	
FR-02	Immediate Feedback: System must return compilation results in < 2 seconds.	Asynchronous Node.js Compiler: The backend utilizes non-blocking I/O to return compilation results and error hints instantly via the console pane.	Met
FR-03	Quest Structure: Topics must be presented as sequential "Quests" via a Skill Tree.	Sector-Based Dashboard: The WorldMap.jsx component organizes content into "Sectors" and "Levels," unlocking sequentially (Refer to Figure 4.2).	Met
FR-04	Plain-English Parsing: Compiler errors must be annotated with beginner-friendly hints.	Intel Tab & Linter: A dedicated sidebar tab provides syntax guides and the custom linter parses critical errors into "Tactical Warnings" before execution.	Met
FR-05	Streak Tracking: System must track daily activity to encourage habit formation.	Not Implemented: Daily streak tracking was scoped during requirements elicitation but was deprioritised in favour of core feedback and quest-structure features within the development timeline.	Not Met
FR-06	Configurable Social: Leaderboards must be opt-in or anonymous.	Separate Leaderboard Page: Competition is isolated to a specific page LeaderboardPage.jsx, preventing passive anxiety on the main dashboard.	Met
NFR-01	Performance: Code execution must occur asynchronously.	Asynchronous Node.js Architecture: The execution engine uses non-blocking I/O (child_process) so the UI thread remains responsive during compilation, addressing the lag-intolerance identified in Theme 1.	Met
NFR-02	Accessibility: Must function on Eduroam (Port 80/443).	Not Verified: Standard HTTP/HTTPS ports were used in development; formal verification on the institutional Eduroam network was not conducted within the scope of this study and is recommended prior to deployment.	Not Verified

NFR-03	Meaningful Gamification: Avoid superficial clicks; use simulation.	Boss Battles: Implementation of visual simulations where code controls graphical avatars rather than just text output.	Met
NFR-04	Professional Aesthetic: "Dark mode" and cyber-security theme.	UI design: The application uses a slate-950 dark theme with professional iconography, adhering to the requested "Cyber/Pro" aesthetic.	Met

5.2.2 Assessment of authenticity vs. gamification

A core tension identified in the literature (Chapter 2) and the empirical findings (Chapter 4) was the balance between "fun" and "learning." The design verification confirms that *JavaRealm* prioritizes authentic practice. By enforcing strict syntax rules via the Monaco Editor while wrapping the experience in a narrative ("Master the Source Code"), the artifact is designed to avoid the "chocolate-covered broccoli" effect—a well-documented risk in gamification design where superficial game elements obscure or fail to integrate with the actual learning content.

5.2.3 Heuristic usability evaluation

In addition to the requirements verification reported in Section 5.2.1, a heuristic usability evaluation was conducted against Nielsen's (1994) ten usability heuristics. This is an expert walkthrough of the artifact's interface conducted by the researcher, comparing the five core screens of JavaRealm (the Home page, World Map, coding interface, Mission Complete modal, Leaderboard, and Profile/Achievements page) against established usability principles. It is reported here as a structural usability check and is explicitly not a substitute for empirical user testing: a heuristic evaluation identifies design issues by inspection against known principles, but it does not measure how real users actually behave when using the system, and cannot detect every usability problem a representative user sample would encounter. It is presented as an appropriate interim evaluation step given the time available for this revision, addressing the absence of any usability-focused evaluation in the original submission, while a formal usability study with real participants remains the priority recommendation for future research (Section 5.6).

Across the ten heuristics, JavaRealm shows clear strengths in visibility of system status (e.g., persistent XP counters, sync indicators, and progress bars across the World Map, Leaderboard, and Profile screens), consistency and standards (a uniform dark-mode visual language and navigation structure across all screens), recognition rather than recall (objectives and requirements displayed in context during missions, and named, described achievements on the

Profile page), and aesthetic and minimalist design (an uncluttered, consistently styled interface). These observations are consistent with the External Examiner's independent assessment that the artifact's figures "show JavaRealm's interface and functionality well" and that it is "demonstrably working".

The most significant gap identified relates to error prevention and error recovery (Nielsen's Heuristics 5 and 9): the screens reviewed do not show how the system responds when submitted code fails to compile or produces incorrect output, so it is not possible to confirm from this evaluation alone whether error feedback is presented in a way that supports learning rather than causing frustration. This is recorded as a priority item for direct verification, and connects to the External Examiner's separate concern regarding backend input handling and security (Section 5.5). A secondary, lower-priority observation concerns flexibility and efficiency of use: no accelerators for more experienced users (e.g., the ability to skip introductory content or navigate directly to a specific level) were identified, which may be a contributing factor in the lower self-rated motivational benefit that more experienced students report from gamified features (Section 4.2.2).

This evaluation does not replace the usability study recommended throughout this chapter. It is reported as a legitimate, citable interim step (Nielsen, 1994) that narrows down which aspects of the interface most need empirical attention — namely error handling and feedback during the compile/run cycle — ahead of a future formal usability study.

5.3 Conclusions

This section discusses the study's outcomes in relation to the specific Research Objectives (ROs) defined in Chapter 1.

5.3.1 RO1: To critically evaluate the theoretical and empirical evidence for gamification's effectiveness in programming education.

Building on the theoretical grounding established in Chapter 2, this section assesses how that theory was translated into concrete design decisions, rather than restating the underlying argument. JavaRealm is designed to support key aspects of student motivation. It promotes autonomy by allowing students to choose their own learning path instead of following a rigid sequence. It strengthens competence through a visual Skill Tree and XP system that clearly tracks progress and makes growth visible. It also encourages a sense of community by giving users a shared professional identity as Operatives and offering optional leaderboards for social interaction and connection. The platform is further structured with the intent of keeping students engaged and minimising frustration, with these effects subject to empirical validation in future

deployment. This is achieved by carefully balancing the difficulty of challenges with the learner's developing skills. The curriculum is scaffolded, beginning with simple foundational tasks and gradually increasing in complexity toward advanced challenges such as Boss Battles, preventing learners from feeling overwhelmed. In addition, the system provides immediate feedback on code execution. This instant response is designed to create a tight feedback loop, supporting students in staying focused, correcting mistakes quickly, and remaining immersed in the learning process benefits that are theoretically grounded and require longitudinal testing to confirm empirically.

5.3.2 RO2: To assess, through empirical engagement with target users, which gamification design factors are critical to engagement.

The empirical analysis in Chapter 4 revealed that students are sophisticated users who reject childish interfaces. The most significant finding was the competition paradox, where a consistent minority of students reported finding leaderboards demotivating despite an overall positive mean response (M = 4.10, SD = 1.02). This finding challenges the default assumption in many gamified systems that points and leaderboards are universally motivating. The study proposes, on the basis of empirical student feedback, that for technical education, competitive elements should be designed as opt-in features to minimise the risk of inducing anxiety — a design recommendation to be tested in future summative evaluation.

5.3.3 RO3 & RO4: To design, develop and verify a functional artifact evaluated against empirically derived requirements.

The development of *JavaRealm* demonstrates that modern web technologies (React.js, Firebase, Monaco Editor) can bridge the gap between educational theory and software practice. The artifact is designed to create a scaffolded environment where complex Java concepts are broken down into manageable sectors, directly addressing the cognitive overload reported by participants — an alignment that awaits empirical confirmation through longitudinal deployment.

5.4 Research contributions

5.4.1 Practical contribution

The primary practical contribution is the *JavaRealm* artifact itself. Unlike commercial tools (e.g., Duolingo) which are often too generic, or standard IDEs (e.g., NetBeans) which provide no scaffolding, *JavaRealm* offers a hybrid solution tailored to the NWU curriculum. For students, it is designed to provide a safe, low-stakes environment to practice syntax before entering high-stakes assessments while for educators The "Boss Battle" concept offers a novel way to assess logic without relying solely on multiple-choice questions or static code submissions.

It is important to note that the contributions listed above are grounded in design verification and theoretical alignment. The empirical confirmation of learning outcome improvements remains the subject of future longitudinal research, as detailed in Section 5.6.

5.4.2 Methodological contribution

The study demonstrates the effectiveness of the design science research (DSR) methodology for student-led software projects. By rigorously mapping qualitative focus group data directly to technical architecture (e.g., the decision to use Node.js to solve "lag"), the study provides a blueprint for how future researchers can translate user needs into software specifications.

5.5 Limitations of the study

While the study met its stated objectives, four limitations meaningfully constrain the strength of the claims that can be drawn from it.

1. Sample Size**:** The empirical data was drawn from a small sample (N = 20) at a single university (NWU). While sufficient for a DSR artifact design, the findings regarding student preferences may not be statistically generalizable to all South African institutions.
2. Longitudinal evaluation: The current study concludes at the design verification stage of the DSR cycle, consistent with design-and-creation research that establishes a robust artifact before full deployment (vom Brocke *et al.,* 2020). This is a deliberate scope boundary rather than an oversight — the artifact has been rigorously verified against all specified requirements derived from empirical student data. The long-term impact of *JavaRealm* on student pass rates, time-to-competency, and engagement across a full semester remains the most critical direction for future research and is prioritised in Section 5.6.
3. Mobile Constraints**:** Although the application is responsive, the *Monaco Editor* (a core feature for authenticity) is inherently difficult to use on small mobile screens. This limits the "anywhere, anytime" learning potential for students without laptops.
4. Depth of Analysis: The questionnaire data were initially analysed using descriptive statistics only; non-parametric inferential tests (Kruskal-Wallis H, Mann-Whitney U) were subsequently added in Section 4.2.2 to test for group differences by programming experience level, addressing this limitation in part. The focus group thematic analysis, however, remains descriptive and qualitative by design, as is standard for thematic analysis; no statistical testing of qualitative themes was undertaken or is methodologically appropriate. The study therefore still cannot speak to whether observed preferences translate into measurable learning gains; that question requires the summative evaluation outlined in Section 5.6.

5.6 Recommendations

5.6.1 Recommendations for future research

Summative evaluation (Priority): The most critical direction for future research is a summative evaluation deploying JavaRealm across a full semester. This should include: (1) pre- and post-tests of Java programming competency; (2) System Usability Scale (SUS) scores from participants; (3) retention and pass rate tracking across the module. Such a study would directly answer whether JavaRealm demonstrably enhances Java learning outcomes — completing the DSR evaluation cycle that the current study initiates at the design and development stage (Peffers *et al.*, 2007).

AI Integration: With the rise of Large Language Models (LLMs), future iterations could integrate an AI tutor to provide dynamic, context-aware hints within the Intel tab, rather than static tutorial text.

Collaborative Coding: Addressing the desire for collaboration (Theme 3), future versions could implement "Pair Programming" quests using WebSockets for real-time multiplayer coding.

5.6.2 Recommendations for implementation

Curriculum Integration: It is recommended that JavaRealm be utilised as a supplementary tool for Tutorial sessions at NWU, specifically for first-year students struggling with the transition from block-based coding to text-based Java.

REFERENCES

Adeyemo, K.S. 2023. The status of digital innovation and data security in South African higher education. *South African Journal of Higher Education*, 37(2):26-39. https://doi.org/10.20853/37-2-5001

Barna, B. & Fodor, S. 2018. An empirical study on the use of gamification on IT courses at higher education. In: Auer, M., Guralnick, D. & Simonics, I., eds. *Teaching and learning in a digital world*. Cham: Springer. pp. 684-692. https://doi.org/10.1007/978-3-319-73210-7_80

Baskerville, R.L., Baiyere, A., Gregor, S., Hevner, A. & Rossi, M. 2018. Design science research contributions: Finding a balance between artifact and theory. *Journal of the Association for information systems*, 19(5):358-376. https://doi.org/10.17705/1jais.00495

Bergin, S. & Reilly, R. 2006. Predicting introductory programming performance: a multi-institutional multivariate study. *Computer Science Education*, 16(4):303-323. https://doi.org/10.1080/08993400600997096

Braun, V. & Clarke, V. 2022. *Thematic analysis: a practical guide*. London: Sage.

Buchanan, E. & Zimmer, M. 2021. Internet research ethics. In: Zalta, E.N., ed. The Stanford encyclopedia of philosophy. Fall 2021 ed. Stanford: *Metaphysics Research Lab*, Stanford University.

Creswell, J.W. & Creswell, J.D. 2023. *Research design: qualitative, quantitative and mixed methods approaches*. 6th ed. Thousand Oaks, CA: Sage.

Creswell, J.W. & Plano Clark, V.L. 2021. *Designing and conducting mixed methods research*. 4th ed. Thousand Oaks, CA: Sage.

Davis, F.D. 1989. Perceived usefulness, perceived ease of use and user acceptance of information technology. *MIS Quarterly*, 13(3): 319-340. https://doi.org/10.2307/249008.

Deitel, P.J. & Deitel, H.M. 2017. *Java how to program, early objects*. 11th ed. Boston: Pearson.

de-Marcos, L., Garcia-Lopez, E. & Garcia-Cabot, A. 2016. On the effectiveness of game-like and social approaches in learning: comparing educational gaming, gamification & social networking. *Computers & Education*, 95:99-113. https://doi.org/10.1016/j.compedu.2015.12.008

DHET (Department of Higher Education and Training). 2020. *Statistics on post-school education and training in South Africa: 2018*. Pretoria: DHET.

Deterding, S., Dixon, D., Khaled, R. & Nacke, L. 2011. From game design elements to gamefulness: defining "gamification". In: *Proceedings of the 15th International Academic MindTrek Conference*. Tampere, Finland: ACM. pp. 9-15. https://doi.org/10.1145/2181037.2181040

Dicheva, D., Dichev, C., Agre, G. & Angelova, G. 2015. Gamification in education: a systematic mapping study. *Educational Technology & Society*, 18(3):75-88. https://www.jstor.org/stable/jeductechsoci.18.3.75

Domínguez, A., Saenz-de-Navarrete, J., de-Marcos, L., Fernández-Sanz, L., Pagés, C. & Martínez-Herráiz, J.J. 2013. Gamifying learning experiences: practical implications and outcomes. *Computers & Education*, 63:380-392. https://doi.org/10.1016/j.compedu.2012.12.020

Etikan, I., Musa, S.A. & Alkassim, R.S. 2016. Comparison of convenience sampling and purposive sampling. American Journal of Theoretical and Applied Statistics, 5(1):1-4. https://doi.org/10.11648/j.ajtas.20160501.11

Field, A. 2017. *Discovering statistics using IBM SPSS statistics*. 5th ed. London: Sage.

Fredricks, J.A., Blumenfeld, P.C. & Paris, A.H. 2004. School engagement: potential of the concept, state of the evidence. *Review of Educational Research*, 74(1):59-109. https://doi.org/10.3102/00346543074001059

Gregor, S. & Hevner, A.R. 2013. Positioning and presenting design science research for maximum impact. *MIS Quarterly*, 37(2):337-355. https://doi.org/10.25300/MISQ/2013/37.2.01

Guest, G., MacQueen, K.M. & Namey, E.E. 2012. *Applied thematic analysis*. Thousand Oaks, CA: Sage.

Guzdial, M. 2013. Exploring hypotheses about media computation. In: *Proceedings of the ninth annual international ACM conference on International computing education research*. San Diego, CA: ACM. pp. 19-26. https://doi.org/10.1145/2493394.2493397

Hamari, J., Koivisto, J. & Sarsa, H. 2014. Does gamification work? a literature review of empirical studies on gamification. In: *Proceedings of the 47th Hawaii International Conference*

on System Sciences. Waikoloa, HI: IEEE. pp. 3025 – 3034. https://doi.org/10.1109/HICSS.2014.377

Hevner, A.R. 2007. A three cycle view of design science research. *Scandinavian Journal of information systems*, 19(2):87-92.

Huang, R., Ritzhaupt, A.D., Sommer, M. & Zhu, J. 2020. The impact of gamification in educational settings on student learning outcomes: a meta-analysis. *Educational technology research and development*, 68(4):1875-1901. https://doi.org/10.1007/s11423-020-09807-z

Kapp, K.M. 2012. *The gamification of learning and instruction: game-based methods and strategies for training and education.* San Francisco: Pfeiffer.

Koster, R. 2013. *A theory of fun for game design*. 2nd ed. Sebastopol, CA: O'Reilly Media.

Krosnick, J.A. 2018. Questionnaire design. In: Vannette, D.L. & Krosnick, J.A., eds. *The Palgrave Handbook of Survey Research*. Cham: Palgrave Macmillan. pp. 439-455. https://doi.org/10.1007/978-3-319-54395-6_53

Kvale, S. & Brinkmann, S. 2015. *Interviews: learning the craft of qualitative research interviewing*. 3rd ed. Thousand Oaks, CA: Sage.

Landers, R.N. 2014. Developing a theory of gamified learning: linking serious games and gamification of learning. *Simulation & Gaming*, 45(6):752-768. https://doi.org/10.1177/1046878114563660

Medeiros, R.P., Ramalho, G.L. & Falcão, T.P. 2019. A systematic literature review on teaching and learning introductory programming in higher education. *IEEE Transactions on Education*, 62(2):77-90. https://doi.org/10.1109/TE.2018.2864133

Nielsen, J. 1994. *Usability Engineering*. San Francisco: Morgan Kaufmann.

Nowell, L.S., Norris, J.M., White, D.E. & Moules, N.J. 2017. Thematic analysis: striving to meet the trustworthiness criteria. *International Journal of Qualitative Methods*, 16(1):1-13. https://doi.org/10.1177/1609406917733847

Oates, B.J., Griffiths, M. & McLean, R. 2022. *Researching information systems and computing*. 2nd ed. London: Sage.

Oracle. 2024. The Java™ Tutorials. https://docs.oracle.com/javase/tutorial/ Date of access: 15 Oct. 2025.

Orji, R., Tondello, G.F. & Nacke, L.E. 2018. Personalizing persuasive strategies in gameful systems to gamification user types. In: *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems.* Montreal: ACM. pp. 1-14. https://doi.org/10.1145/3173574.3174009

Papert, S. 1980. *Mindstorms: Children, Computers, and Powerful Ideas*. New York: Basic Books.

Park, S. & Kim, S. 2019. A badge design framework for a gamified learning environment: cases analysis and literature review for badge design. *JMIR Serious Games*, 7(2):e14342. https://doi.org/10.2196/14342

Patton, M.Q. 2015. *Qualitative research and evaluation methods*. 4th ed. Thousand Oaks, CA: Sage.

Peffers, K., Tuunanen, T., Rothenberger, M.A. & Chatterjee, S. 2007. A design science research methodology for information systems research. *journal of management information systems*, 24(3):45-77. https://doi.org/10.2753/MIS0742-1222240302

Ratinho, E. & Martins, C. 2023. The role of gamified learning strategies in students' motivation in high school and higher education: A systematic review. *Heliyon*, 9(8):e19033. https://doi.org/10.1016/j.heliyon.2023.e19033

Robins, A., Rountree, J. & Rountree, N. 2003. Learning and teaching programming: a review and discussion. *computer science education*, 13(2):137-172. https://doi.org/10.1076/csed.13.2.137.14200

Robey, D., Taylor, W.T.F. & Grabowski, L.J. 2018. Pragmatic rigor: principles and criteria for conducting and evaluating practitioner scholarship. *Engaged Management ReView*, 2(3):1-14. https://doi.org/10.28953/2375-8643.1053 https://commons.case.edu/emr/vol2/iss3/2/

Romero, M., Usart, M. & Ott, M. 2015. Can serious games contribute to developing and sustaining 21st century skills? *Games and Culture*, 10(2):148-177. https://doi.org/10.1177/1555412014548919

Sailer, M., Hense, J.U., Mayr, S.K. & Mandl, H. 2017. How gamification motivates: an experimental study of the effects of specific game design elements on psychological need satisfaction. *Computers in Human Behavior*, 69:371-380. https://doi.org/10.1016/j.chb.2016.12.033

Sailer, M. & Homner, L. 2020. The gamification of learning: a meta-analysis. *Educational Psychology Review*, 32(1):77-112. https://doi.org/10.1007/s10648-019-09498-w

Seaborn, K. & Fels, D.I. 2015. Gamification in theory and action: a survey. *International Journal of Human-Computer Studies*, 74:14-31. https://doi.org/10.1016/j.ijhcs.2014.09.006

Skinner, B.F. 1938. *The behavior of organisms: an experimental analysis*. New York: Appleton-Century.

Subhash, S. & Cudney, E.A. 2018. Gamified learning in higher education: a systematic review of the literature. *Computers in Human Behavior*, 87:192-206. https://doi.org/10.1016/j.chb.2018.05.028

Sweller, J. 1988. Cognitive load during problem solving: effects on learning. *Cognitive Science*, 12(2):257-285. https://doi.org/10.1207/s15516709cog1202_4

Tondello, G.F., Mora, A., Marczewski, A. & Nacke, L.E. 2019. Empirical validation of the Gamification User Types Hexad scale in English and Spanish. *International Journal of Human-Computer Studies*, 127:95-111. https://doi.org/10.1016/j.ijhcs.2018.10.002

vom Brocke, J., Winter, R., Hevner, A.R. & Maedche, A. 2020. Special issue editorial: accumulation and evolution of design knowledge in design science research: a journey through time and space. *Journal of the Association for information systems*, 21(3):520-544. https://doi.org/10.17705/1jais.00611

Watson, C. & Li, F.W. 2014. Failure rates in introductory programming revisited. In: *Proceedings of the 2014 Conference on Innovation & Technology in Computer Science Education*. Uppsala, Sweden: ACM. pp. 39-44. https://doi.org/10.1145/2591708.2591749

Werbach, K. & Hunter, D. 2012. *For the Win: How Game Thinking Can Revolutionize Your Business*. Philadelphia: Wharton Digital Press.

Zhan, Z., He, L., Tong, Y., Liang, X., Guo, S. & Lan, X. 2022. The effectiveness of gamification in programming education: evidence from a meta-analysis. *Computers and Education: Artificial Intelligence*, 3:100096. https://doi.org/10.1016/j.caeai.2022.100096

ANNEXURE A: DATA COLLECTION TOOLS

Questionnaire

Title: Impact of Gamification on Java Programming Learning

Section 1: Demographics and Background

1. Age:
    - Under 20
    - 20–24
    - 25–29
    - 30+
2. Gender:
    - Male
    - Female
    - Prefer not to say
    - Non-binary
3. Academic level:
    - 1st-year student
    - 2nd-year student
    - 3rd-year student
    - Postgraduate student
4. Have you completed or are you currently enrolled in a Java programming course?
    - Yes

- No

5. Rate your Java programming experience:
    - None
    - Beginner
    - Intermediate
    - Advanced

Section B: Perceptions of Gamified Learning (5-point Likert scale: 1 = Strongly Disagree, 5 = Strongly Agree)

6. I believe gamification can make Java programming more engaging.

    1 = Strongly Disagree

    2 = Disagree

    3 = Neutral

    4 = Agree

    5 = Strongly Agree

7. Gamified features (e.g., points, badges, leaderboards) would motivate me to learn more consistently.

    1 = Strongly Disagree

    2 = Disagree

    3 = Neutral

    4 = Agree

    5 = Strongly Agree

8. Competition with peers would increase my motivation to learn Java.

1 = Strongly Disagree

2 = Disagree

3 = Neutral

4 = Agree

5 = Strongly Agree

9. Collaboration (team-based rewards or challenges) would motivate me more than competition.

    1 = Strongly Disagree

    2 = Disagree

    3 = Neutral

    4 = Agree

    5 = Strongly Agree

10. Receiving immediate feedback (e.g., progress bars, hints) would help me stay engaged.

    1 = Strongly Disagree

    2 = Disagree

    3 = Neutral

    4 = Agree

    5 = Strongly Agree

11. I would find rewards (badges, levels, certificates) meaningful in tracking my learning progress.

    1 = Strongly Disagree

    2 = Disagree

3 = Neutral

4 = Agree

5 = Strongly Agree

12. A gamified app should be simple and easy to use.

1 = Strongly Disagree

2 = Disagree

3 = Neutral

4 = Agree

5 = Strongly Agree

13. I would likely use a gamified Java learning app regularly if it was available.

1 = Strongly Disagree

2 = Disagree

3 = Neutral

4 = Agree

5 = Strongly Agree

Section C: Open-ended

14. What gamification feature do you think would help you most in learning Java?

15. What possible challenges or drawbacks do you see in using a gamified learning app?

Focus Group Discussion Guide

Title: Exploring Student Perspectives on a Gamified Java Learning Application

Purpose:

To gather in-depth qualitative insights from IT students about how a gamified learning tool can support engagement, motivation, and understanding in Java programming education. The discussion will guide the design and refinement of the proposed application.

1. Introduction (5 mins)

Welcome and thank you for participating. This focus group is part of a study to design a gamified web application that supports students learning Java programming. Your feedback will help shape how the application looks and functions. Please feel free to speak openly—there are no right or wrong answers. Your responses will remain confidential, and participation is voluntary. The session will take about 45 to 60 minutes.

2. Warm-Up Questions (5 mins)

Please introduce yourself (name, year of study). How do you usually feel about learning Java programming? easy, challenging, enjoyable, frustrating?

Impact of Gamification

Have you used gamified apps (for learning or entertainment) before? How did they affect your motivation?

How do you think gamification could change your experience of learning Java?

Critical Success Factors

3. What do you think would make a gamified learning tool successful for Java programming?

4. What challenges or barriers might stop you from using such a tool?

Effective Gamification Elements

5. Which gamification features (badges, points, leaderboards, challenges, rewards, levels) would keep you most engaged? Why?

6. Would you prefer individual rewards or team-based features? How do you feel about competition vs. collaboration?

Design considerations

7. If you could design this gamified Java app, what features would you definitely include?

8. How would you want your progress and achievements to be displayed?

9. What type of feedback (immediate, delayed, personalized) would motivate you most?

7. Closing (5 mins)

If you could tell the developers one thing to include (or avoid) in this tool, what would it be?

Is there anything we didn’t ask that you think is important?

Thank you for your time and input. Your ideas will help create a better learning tool for students like you

ANNEXURE B: INFORMED CONSENT

Informed Consent Form - Questionnaire

Title of the Study: Enhancing Java learning through gamification: a web-based application

Researcher: Jonathan Sithole

Institution: North-West University

Supervisor: Prof. K Gorejena

Co-supervisor: Mrs. M.E Nhlapo

Dear Participant,

You are kindly invited to take part in an academic research study conducted by **Jonathan Sithole**, a postgraduate student at North-West University (NWU).

The purpose of this study is to explore how gamification can enhance student motivation and engagement in learning Java programming. Your participation will help shape the design of a more student-centred learning solution.

This study will use a quantitative research approach, involving a structured online questionnaire. You will be asked to respond to statements based on your own experience and preferences. This will take approximately 15–20 minutes to complete.

Basic demographic information such as age, gender, year of study, and prior Java experience will be requested. Please note that the questionnaire is anonymous and no personal identifying information (such as names or student numbers) will be collected. All data will be kept strictly confidential and will only be used for academic research purposes. Only the researcher will have access to the responses.

The link to the online questionnaire will be distributed through NWU's official communication platforms by a designated gatekeeper. Participation is entirely voluntary and you are free to decline or withdraw at any time without any negative consequences.

Please note the following:

- This study is anonymous—no names or identifying details will be collected.

- Your responses will be treated with strict confidentiality.
- Participation is voluntary and you may withdraw at any time.
- The survey takes 15–20 minutes to complete.
- The results will be available upon request after the study is concluded.
- No harm or manipulation is involved in this study.
- The information gathered will only be used for academic purposes.

By completing the questionnaire, you are giving your informed consent to participate in this study.

Thank you for your time and valuable contribution.

Sincerely,
Jonathan Sithole

Informed Consent Form - Focus Groups

What Participation Involves

If you agree to participate, you will join a focus group discussion lasting approximately 45–60 minutes. The session will involve open-ended questions about your learning experiences, preferences for gamified elements, and ideas for improving Java programming education. The discussion will be audio-recorded (with your consent) to ensure accuracy in capturing your responses.

Confidentiality and Data Protection

Your identity will remain confidential. No names or personal identifiers will be linked to your responses in any reports or publications. All collected information will be securely stored and handled in compliance with the Protection of Personal Information Act, 4 of 2013 (POPIA), which governs the lawful collection, processing, and storage of personal data in South Africa. Only the researcher will have access to the recordings and transcripts, which will be used solely for academic purposes and destroyed after the study is completed.

Voluntary Participation

Your participation is entirely voluntary. You may choose not to answer any question you are uncomfortable with and may withdraw from the discussion at any time without penalty or negative consequences. There is no financial compensation for participating in this study.

Please note the following:

- This session will be audio-recorded for research accuracy (with your consent).
- Your responses will remain confidential and anonymous in all reports.
- You may withdraw at any point without consequences.
- The discussion will last approximately 45–60 minutes.
- No harm, deception, or manipulation is involved.
- All data will be processed and stored in accordance with POPIA.

Consent Statement

By signing below, you confirm that:

1. You have read and understood the information provided above.

2. You voluntarily agree to participate in this focus group discussion.

3. You consent to the audio recording of the session.

Name of Participant: _____

Signature: _____ **Date:** _____

Researcher's Signature: _____ **Date:** _____

ANNEXURE C: RECRUITMENT MATERIAL

PARTICIPANT RECRUITMENT EMAIL

Subject: Invitation to Participate in Research on Java Learning

Dear Participant,

My name is Jonathan Sithole, and I am conducting research as part of my MCom in Informatics at North-West University. I am developing a web-based application designed to enhance student learning in Java programming by using gamification techniques such as points, badges, and leader boards.

I am inviting undergraduate students who are currently enrolled in Java programming courses to participate in this study. Your participation will involve completing a short survey and, if selected, participating in a focus group discussion. The purpose of this study is to evaluate the effectiveness of gamification in improving engagement and learning outcomes in programming education.

By participating in this study, you will contribute to research that aims to improve Java programming education and experience an innovative learning tool that could potentially benefit your studies.

Participation is entirely voluntary, and all responses will be kept confidential.

Thank you for considering this opportunity. Your insights would be incredibly valuable to this research.

Sincerely,

Jonathan Sithole

32332424@nwu.ac.za

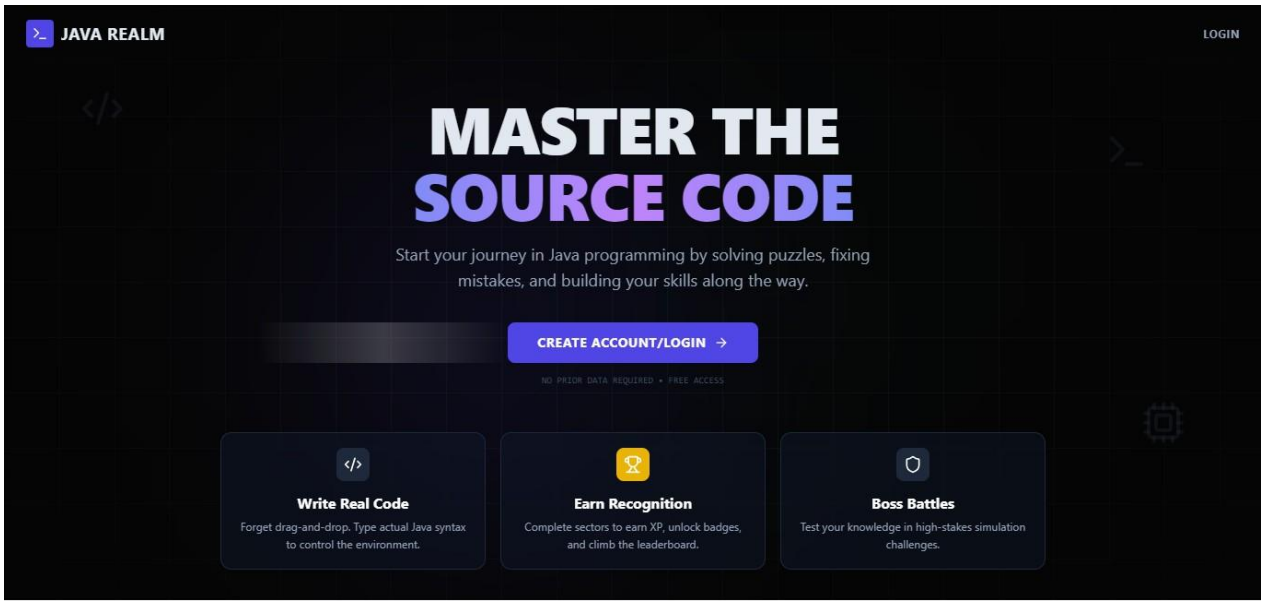
0633675620

ANNEXURE D: JAVAREALM INSTRUCTION MANUAL

link to prototype: https://gamificationproject-fea30.firebaseapp.com/

INSTRUCTION MANUAL

1. Home Page: Your Entry Point



The Home Page is the front door of Java Realm.

1.1. Purpose

- Introduce the system
- Provide a clear starting action
- Establish the game-based theme

1.2. How to Navigate

- Click the large central button:

o “Create Account / Login” if you are new or logged out

o “Resume Simulation” if you have logged in before

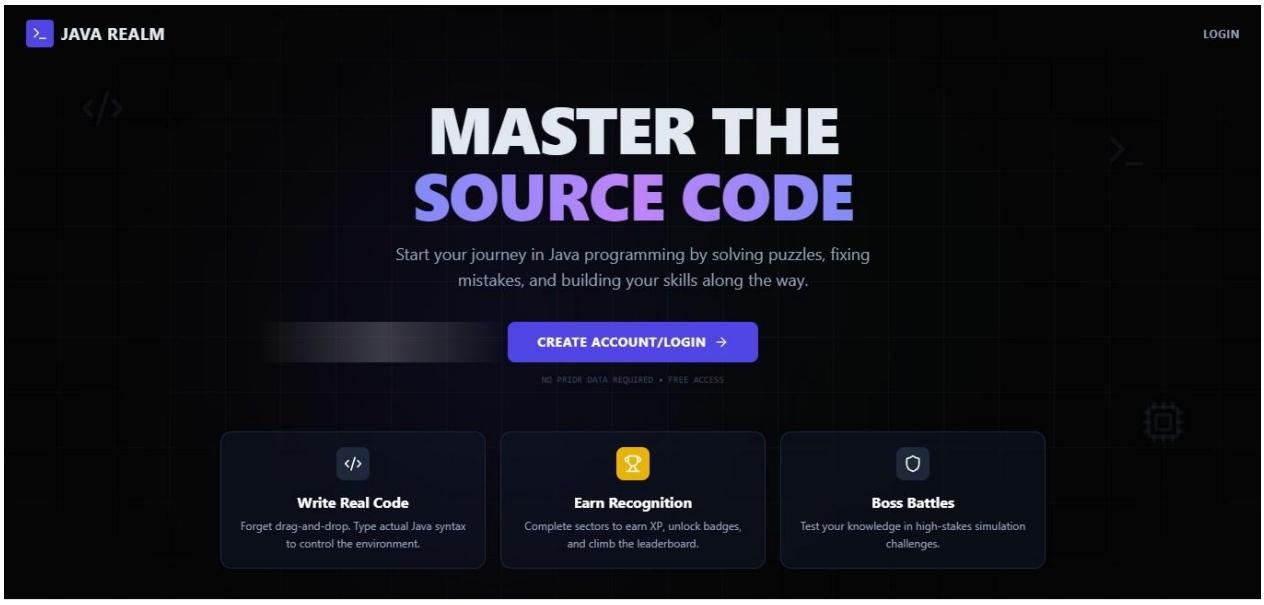
1.3. What You’ll See

- Java Realm Logo (Top-Left): Confirms you are in the main system
- Feature Overview: Scroll down to view the three pillars:
  - Writing Real Code
  - Earning Recognition
  - Boss Battles

1.4. System Response

Clicking the main button transitions you to the Identity Portal. This transition is immediate and intentional.

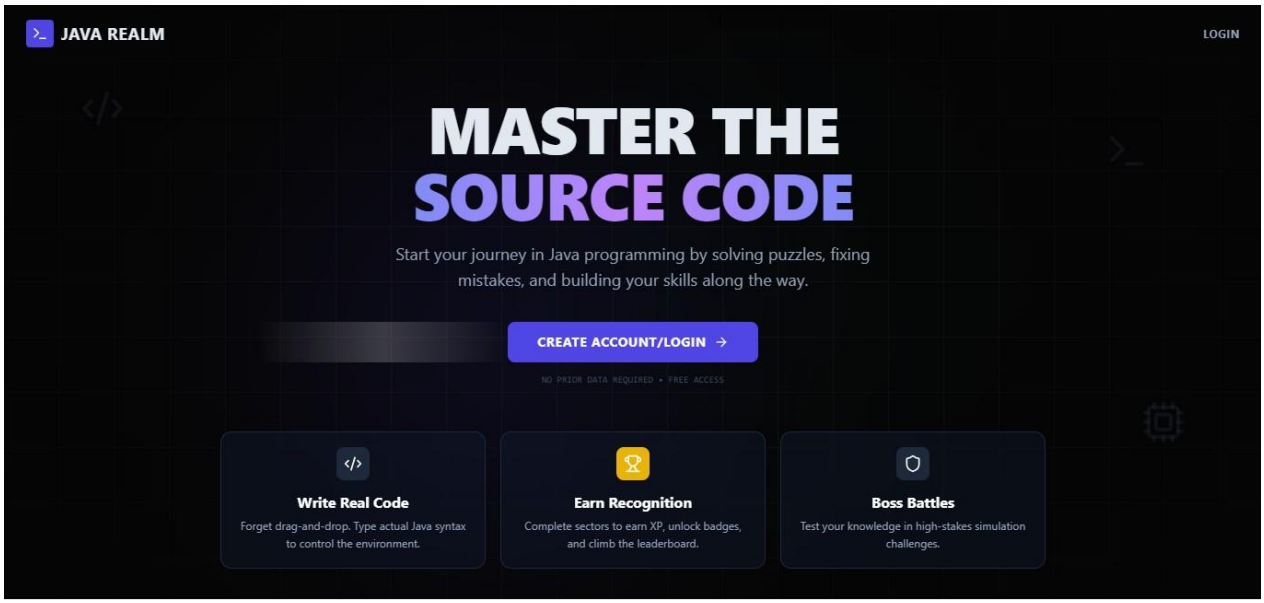
2. Login & Registration



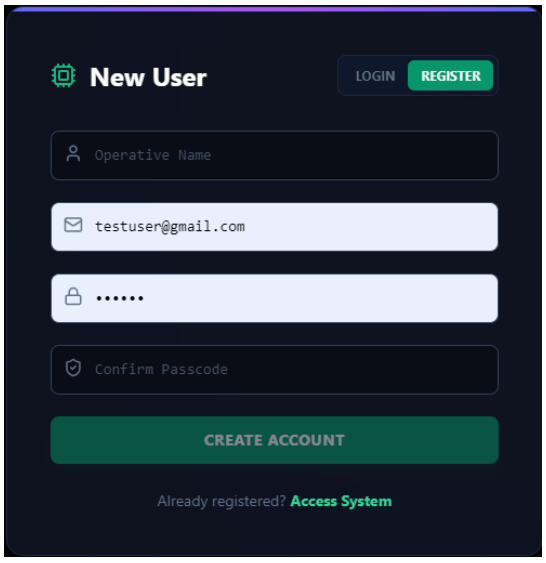
Before progress can be saved, you must establish an identity.

2.1. Switching Between Forms

- Use the toggle at the top:
  - LOGIN – for existing users



o REGISTER – to create a new account



2.2. Creating an Account

- Enter a valid email address
- Create a secure password

This allows the system to track: - Experience Points (XP) - Badges and achievements - Level completion

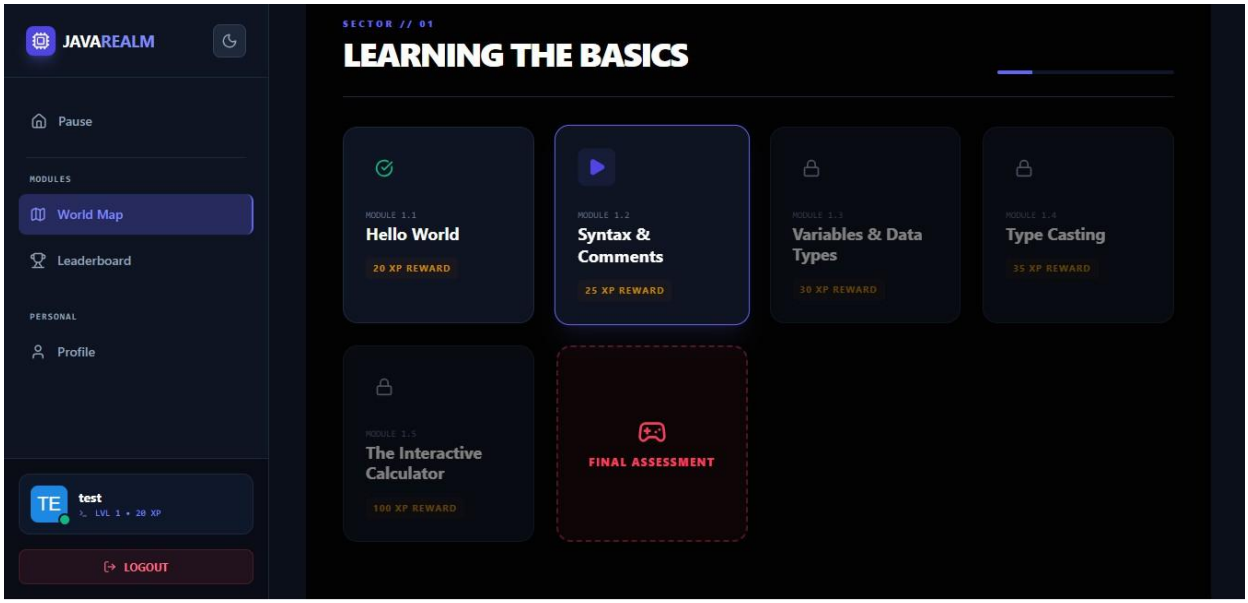
2.3. Forgot Password

- Click “Reset Access Codes” to open the recovery page

2.4. System Response

- Successful login sends you directly to the World Map
- Errors appear immediately with clear feedback

3 World Map



The World Map is your central navigation hub.

3.1. Purpose

- Show all learning content
- Enforce structured progression
- Provide visual feedback on progress

3.2. Understanding the Map

- Sectors: Each sector represents a Java topic
- Module Cards: Individual lessons within a sector

3.3. Card States

- Lock Icon: Module is locked and cannot be clicked yet (this is intentional)
- Play Icon: Your current active mission
- Checkmark: Module completed successfully

Locked modules are not broken — they ensure you learn concepts in the correct order.

3.4. Boss Battles

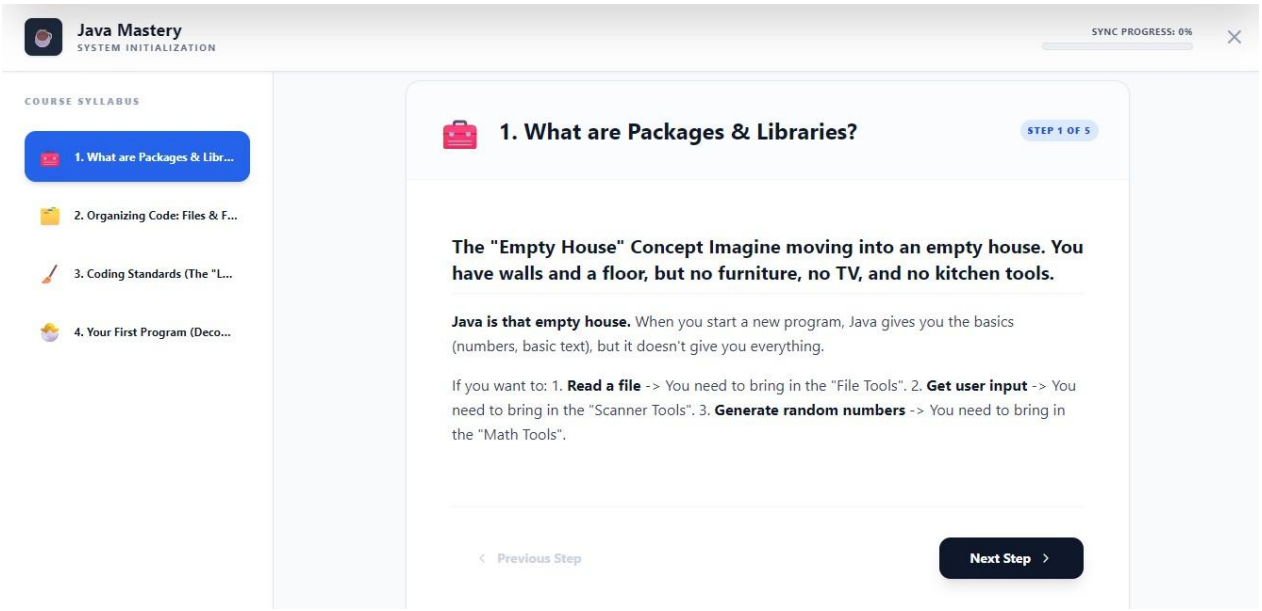
- Red cards at the end of a sector
- Unlock only after all other modules turn green

3.5. System Response

Clicking an unlocked module immediately launches the Level Interface, hiding the World Map to reduce distractions.

4. Sector 0





Before writing code, all users must complete System Initialization.

4.1. Purpose

- Teach basic concepts
- Familiarize users with system navigation

4.2. Navigation

- Left Sidebar: Course syllabus topics
- Center Panel: Lesson content
- Next Step Button: Advances the lesson

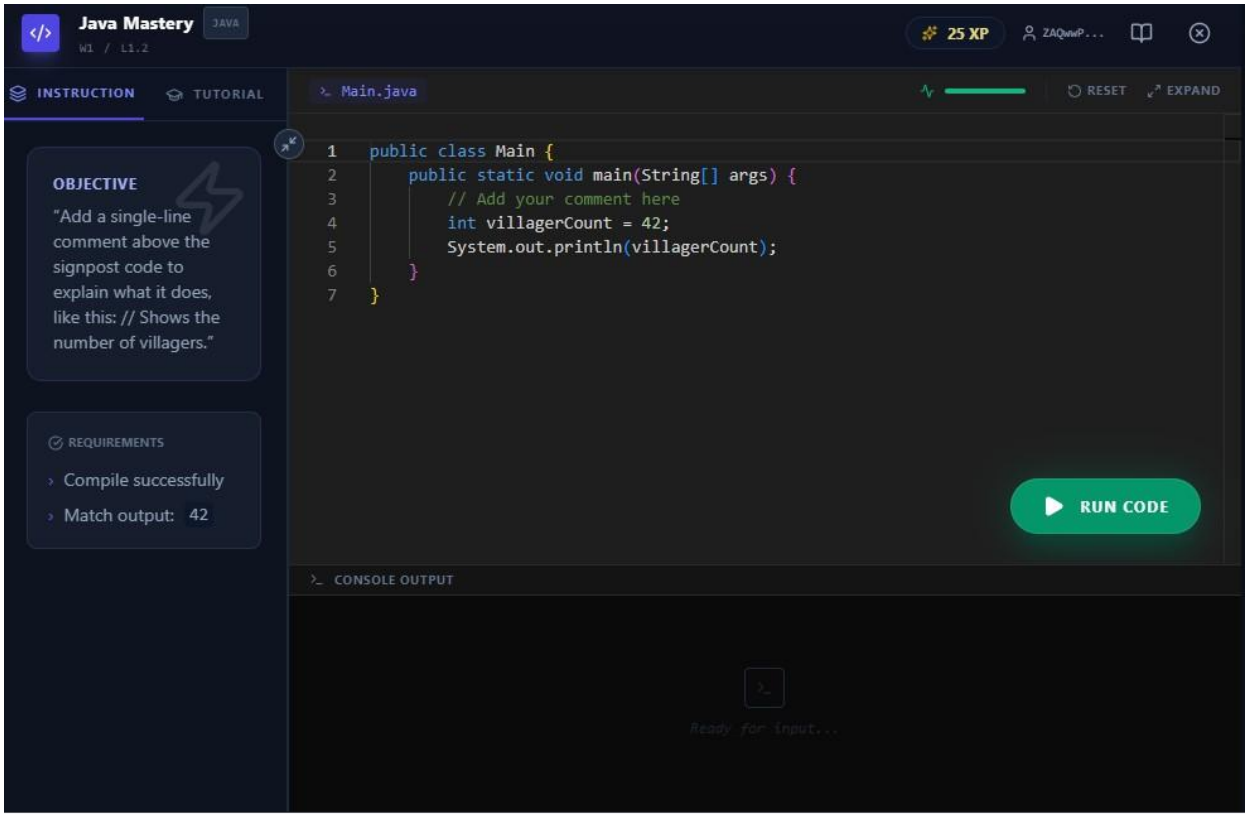
4.3. Quizzes

- Appear at the end of lessons
- Multiple-choice format
- You must answer 3 out of 5 correctly to pass

4.4. Exiting

- Click the “X” in the top-right corner
- Progress up to the last completed lesson is saved

5. Level Interface



This is where you write real Java code.

5.1. Layout Overview

5.1.1. Left Sidebar

- Instruction Tab: Explains the objective
- Tutorial Tab: Syntax references and examples

On mobile devices, open this sidebar using the Hamburger Menu (three lines).

5.1.2. Code Editor (Top-Right)

- Typing Area: Where you write Java code
- Reset Button: Clears all code and restarts the level
- Maximize Button: Hides other panels for focus

5.1.3. Controls & Console (Bottom-Right)

- RUN CODE: Compiles and executes your code
- Console Output: Displays system feedback

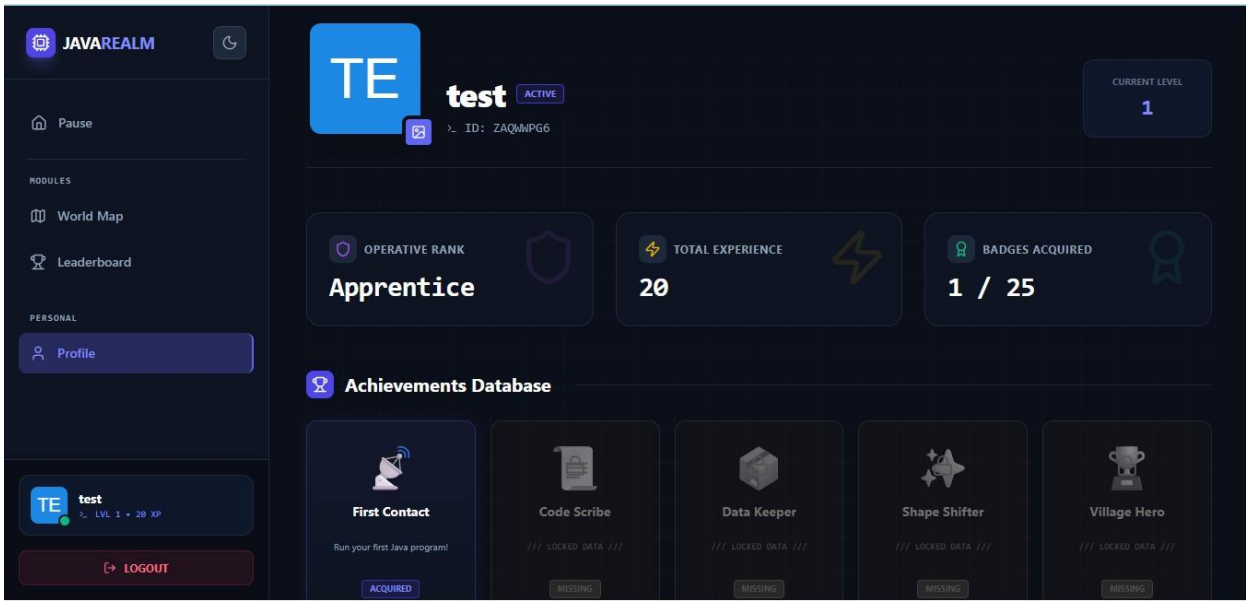
o green text = Success

o red text = Error (includes line number)

5.2. Hover-Header Navigation

- Move your mouse to the very top of the screen to reveal the header
- Click “Abort Mission” to safely exit and return to the World Map

6. Profile Page



This page tracks your growth and achievements.

6.1. Avatar

- Click your profile picture to:

o Upload a custom image

o Select a system-preset avatar

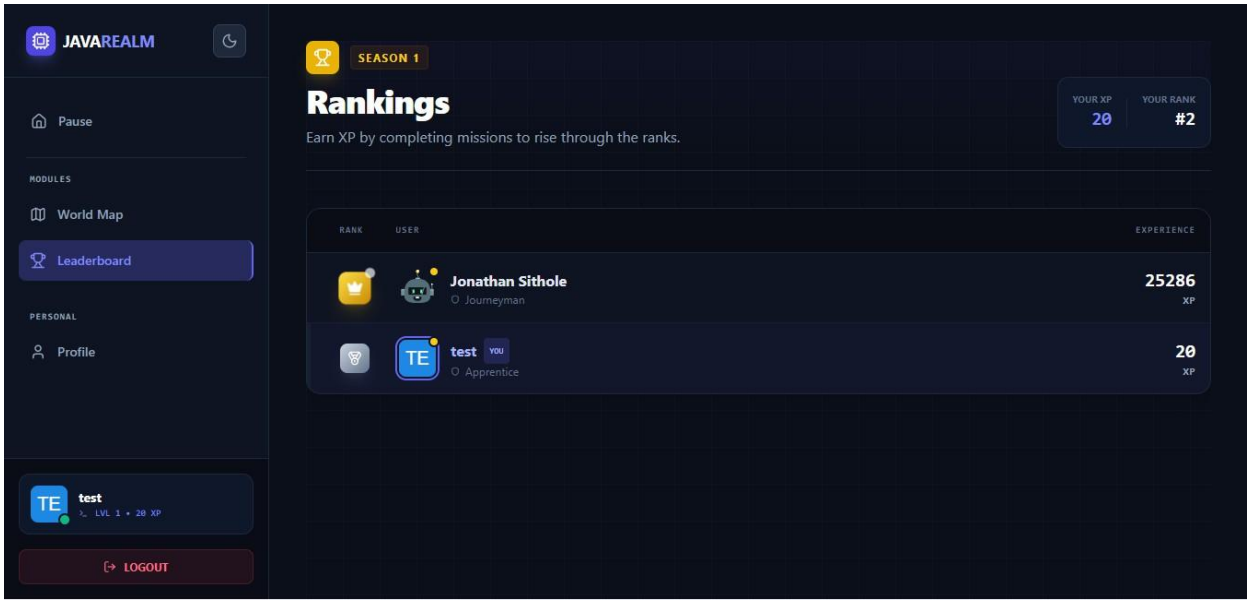
6.2. Stats Cards

- Operative Rank: Based on badges earned
- Total Experience: Cumulative XP

6.3. Achievements Database

- Color badges = earned
- Grayscale badges = locked

7. Leaderboard



The Leaderboard displays competitive progress.

7.1. Features

- Your position highlighted in indigo
- Top three players marked with:

o Gold Crown

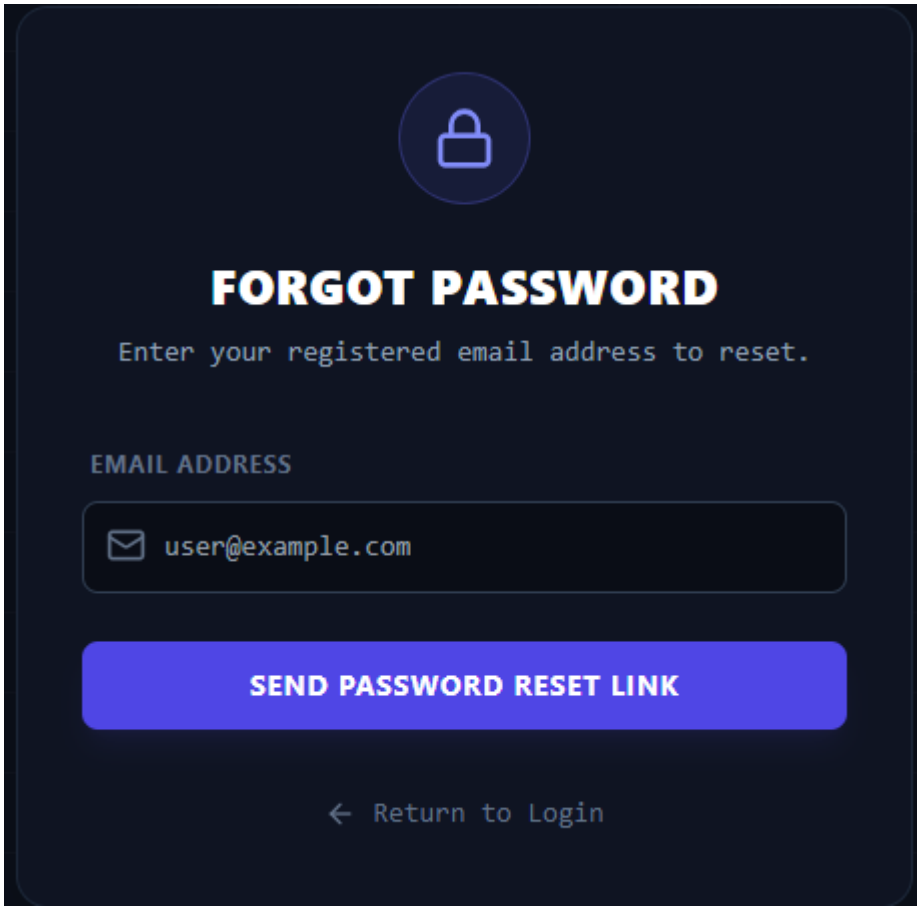
o Silver Medal

o Bronze Medal

7.2. System Response

Rankings update in real time when XP is earned.

8. Forgot Password Page



8.1. Purpose

Restore account access securely.

8.2. How It Works

- Enter your registered email
- Click Send Reset Link

For security reasons, the system does not confirm whether an account exists.

9. Quick-Start Navigation Tips

- Lost? Click the Terminal icon to return to the Home Page or World Map

- Stuck? Open the Tutorial Tab inside a level
- Can’t see RUN CODE? Ensure browser zoom is set to 100%
- Want to exit safely? Use Abort Mission — progress is preserved