**Feasibility Study**

**For Recode – A Smart Coding Challenge Generator for Students**

**Document Control**

|  |  |
| --- | --- |
| Information | Details |
| Document ID | RECODE-FS-001 |
| Document Owner | Group |
| Issue Date | 07/08/2025 |
| Last Saved Date | 08/08/2025 |
| File Name | Recode\_FS.docx |

**Document History**

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Issue | Date | Changes |
| 1.0 | Added information | 08/08/2025 | Initial Document |
| 2.0 | Fixed Minor Issues | 10/08/2025 | 2nd Document |
|  |  |  |  |

**Document Approvals**

|  |  |  |  |
| --- | --- | --- | --- |
| Role | Name | Signature | Date |
| Project Manager | Mohau Liphoko | ML | 10/08/2025 |
| Team Leader | Brandon v. Vuuren | BV | 10/08/2025 |

**Table of Contents**

[1. Executive Summary 5](#_Toc205830969)

[2. Problem Statement 6](#_Toc205830970)

[2.1 Business Environment 6](#_Toc205830971)

[2.2 Business Problem 6](#_Toc205830972)

[3. Requirements Statement 7](#_Toc205830973)

[3.1 Business Drivers 7](#_Toc205830974)

[3.2 Business Requirements 8](#_Toc205830975)

[4. Feasibility Assessment 8](#_Toc205830976)

[4.1 Potential Solutions 8](#_Toc205830977)

[4.2 Solution 1: Recode 8](#_Toc205830978)

[4.2.1 Description 8](#_Toc205830979)

[4.2.2 Assessment 9](#_Toc205830980)

[4.2.3 Results 10](#_Toc205830981)

[4.2.4 Risks 10](#_Toc205830982)

[4.2.5 Issues 10](#_Toc205830983)

[4.2.6 Assumptions 11](#_Toc205830984)

[4.3 Solution 2: Code Visualisation Sandbox 11](#_Toc205830985)

[4.3.1 Description 11](#_Toc205830986)

[4.3.2 Assessment 12](#_Toc205830987)

[4.3.3 Results 12](#_Toc205830988)

[4.3.4 Risks 13](#_Toc205830989)

[4.3.5 Issues 13](#_Toc205830990)

[4.3.6 Assumptions 14](#_Toc205830991)

[4.4 Solution 3: Use Open-source code to build the system 14](#_Toc205830992)

[4.4.1 Description 14](#_Toc205830993)

[4.4.2 Assessment 15](#_Toc205830994)

[4.4.3 Results 15](#_Toc205830995)

[4.4.4 Risks 15](#_Toc205830996)

[4.4.5 Issues 16](#_Toc205830997)

[4.4.6 Assumptions 16](#_Toc205830998)

[5. Feasibility Ranking 17](#_Toc205830999)

[5.1 Ranking Criteria 17](#_Toc205831000)

[5.2 Ranking Scores 18](#_Toc205831001)

[6. Feasibility Result 18](#_Toc205831002)

[7. Implementation Plan 19](#_Toc205831003)

[Implementation Strategy 19](#_Toc205831004)

[Phase 1 - MVP 19](#_Toc205831005)

[Phase 2 -Expansion 19](#_Toc205831006)

[Phase 3 -Scaling 20](#_Toc205831007)

[Project Team 20](#_Toc205831008)

[Project Phases 20](#_Toc205831009)

[1. Planning 20](#_Toc205831010)

[2. Development - Conducted in phased 2 - 3 week sprints 20](#_Toc205831011)

[Phase 1 - MVP Development (Months 1-6) 20](#_Toc205831012)

[Phase 2 - Feature Expansion (Months 7-12) 21](#_Toc205831013)

[Phase 3 - Scaling and Partnerships (Months 13-24) 21](#_Toc205831014)

[3. Testing 22](#_Toc205831015)

[4. Deployment 22](#_Toc205831016)

[Execution Practices 22](#_Toc205831017)

[Communication & Collaboration Plan 22](#_Toc205831018)

[Appendix A: Recode 23](#_Toc205831019)

[Appendix B: Code Visualisation Sandbox Cost Analysis 24](#_Toc205831020)

[Appendix C: Moodle software cost analysis 25](#_Toc205831021)

[References 28](#_Toc205831022)

# 1. Executive Summary

Recode Labs is a team of twelve developers from North-West University with one clear goal which is to change the way students learn, practice, and think about coding. These twelve developers have a variety of skills and knowledge in the technological landscape, have a string growth mindset, and show excellent analytical and collaborative abilities. Using these skills, a business case and feasibility study need to be done to check if a possible solution can be found.

Many first-year university students are introduced to coding concepts in lectures but lack regular, structured opportunities to apply this knowledge in a hands-on environment. According to Masegosa, Cabañas, Maldonado & Morales (2024: Introduction), “many students struggle to connect programming theory learned in lectures with practical problem-solving due to a lack of accessible, course-specific exercises”. Current methods often rely on generic problem sets or require significant manual preparation from lecturers, limiting reach and effectiveness. This leads to reduced engagement, slower skill development, and weaker problem-solving ability.

ReCode proposes three potential solutions to address the disconnect between theoretical programming instruction and practical application among first-year IT students. The first solution is the development of the *ReCode*, a basic but functional version of the platform that delivers core coding challenges aligned with lecture content. The second solution is the implementation of a *Code Visualisation Sandbox*, an interactive environment that helps students understand how code executes by visualising processes like logic flow, variable changes, and error handling. The third solution involves *leveraging open-source code* to accelerate development by adapting and integrating existing learning tools into the ReCode system. These solutions will be explored in more detail in the Feasibility Assessment section.

ReCode will be developed using Scrum, a form of agile methodology well-suited for iterative and collaborative development. Scrum enables the project team to break the platform’s development into smaller, manageable sprints, ensuring early delivery of functional components and allowing for continuous testing and feedback. This is particularly beneficial in an educational context where lecturer and student input can be used to improve the system’s functionality and alignment with learning outcomes. The agile approach also reduces the risk of large-scale system failure by catching bugs and design issues early in the development cycle.

In conclusion, ReCode represents a practical and scalable solution to a well-documented problem in first-year programming education. With a development plan grounded in proven methodologies and a focus on educational effectiveness, ReCode has strong potential to improve learner engagement, performance, and long-term coding competence across South African higher education institutions.

# 2. Problem Statement

## 2.1 Business Environment

Many South African universities now include programming modules in both general education and IT-specific qualifications. These courses aim to equip students with critical problem-solving and computational thinking skills. However, there is a widening gap between the theoretical content delivered during lectures and the practical skills required to apply that knowledge effectively.

Traditional classroom environments often rely on passive learning methods, and students are expected to bridge the gap independently outside of lecture hours. Due to large class sizes and limited lecturer availability, students receive minimal individualised support, and practical activities are not always tailored to the actual content covered in lectures. Existing online coding platforms are often either too advanced, misaligned with the local university syllabus, or lack the contextual grounding needed for beginners, especially those without prior exposure to IT in high school.

In this context, scalable and adaptable tools that automatically align practice with curriculum content are essential for improving learning outcomes, reducing student dropout rates, and enhancing programming competency in first-year university students. The ReCode project aims to meet this educational demand through a platform that can generate content in line with lecture material, provide immediate feedback, and scale with institutional needs.

## 2.2 Business Problem

Current programming education models struggle to provide students with consistent and relevant coding practice. Many students, especially those new to programming, find it difficult to apply the concepts they’ve learned in theory (such as loops, conditionals, data structures, and debugging) to practical exercises. Without sufficient reinforcement, this gap often leads to frustration, low engagement, and underperformance in coding-related modules.

While some universities provide access to generic online platforms (like Codecademy, HackerRank, or W3Schools), these tools do not integrate well with the local curriculum, and lecturers lack the time and resources to design and grade weekly practice activities tailored to each lesson. Additionally, students receive limited real-time feedback, which slows down the learning process and reduces opportunities for self-correction.

The proposed solution, ReCode, will allow lecturers to upload their lecture material (such as slides or notes) and automatically generate customised coding challenges for their students. This will ease the workload on academic staff while giving students immediate access to relevant exercises that reinforce the material they’ve just learned. The platform can further support student learning with feedback loops, code testing environments, and usage analytics.

The solution must be developed and tested before the start of the next academic year to ensure it can support the incoming first-year cohort. This timeframe allows for pilot testing, refinement based on feedback, and early integration into selected programming courses.

# 3. Requirements Statement

## 3.1 Business Drivers

The success of the ReCode project depends on several key business drivers that justify its development and guide its implementation. These drivers reflect both the educational context of South African universities and the institutional goals of improving teaching efficiency and student success.

**Educational Demand:** The growing national and global emphasis on coding literacy has made programming a core component in many degree programmes, especially in IT and Computer Science. South African universities are under increasing pressure to equip students with industry-relevant coding skills from their first year of study. This demand fuels the need for innovative teaching tools that offer both theory and practice in an integrated format.

**Academic Staff Capacity:** Lecturers and tutors are expected to deliver both theoretical knowledge and practical engagement in increasingly large classrooms. However, time and resource constraints limit their ability to design, distribute, and assess weekly coding exercises. ReCode directly addresses this pressure by allowing lecturers to automate the generation of practical tasks, saving time and ensuring consistency in learning opportunities.

**Digital Learning Alignment:** The strategic goals of most higher education institutions include the integration of digital tools into the learning experience. ReCode aligns with these goals by offering an online, scalable platform that complements lecture content and promotes independent learning. It supports blended learning models and enhances curriculum delivery through interactive practice and automated feedback.

**Student Success and Retention:** Early exposure to practical coding exercises improves student comprehension, confidence, and performance in programming modules. Tools like ReCode contribute to improved academic outcomes, which in turn enhance student retention rates (particularly important in high-dropout modules like first-year programming). This platform also encourages self-directed learning, allowing students to engage at their own pace and build foundational coding competence over time.

## 3.2 Business Requirements

|  |  |
| --- | --- |
| Business Problem/Opportunity | Project Requirement |
| Students lack personalised practice tasks. | Generate exercises based on lecture content (automated) |
| Lecturers are overloaded. | The system requires minimal input from academic staff |
| Manual marking is not scalable. | Auto-grading of student code submissions |
| Students want gamification and feedback. | Progress tracking and basic gamification (badges, ELO) |

# 4. Feasibility Assessment

## 4.1 Potential Solutions

* **Solution 1**: Recode
* **Solution 2**: Code Visualisation Sandbox
* **Solution 3**: Use Open-source code to build the system

## 4.2 Solution 1: Recode

### 4.2.1 Description

The proposed solution focuses on making ReCode scalable and resilient by addressing both macro and micro environmental factors without altering the platform’s core purpose. Instead of changing the original concept, this solution enhances it by implementing technological and structural design choices that make the platform adaptive, flexible, and easy to maintain.

At the micro level, institutional pressures such as modifications in module structures, varying levels of staff involvement, and changes in student behaviour are considered (Mugisha and Ssenkusu, 2025). On a macro scale, developments like AI integration, new policies including POPIA compliance, and economic shifts affecting cloud service affordability are also considered (Salimi, 2025; Adepoju, 2025).

To address these challenges, the solution adopts a scalable microservice architecture where key functionalities like badge creation, slide parsing, and notifications are separated into deployable FastAPI services. This division reduces risk and simplifies updates. The platform supports multiple deployment environments (staging, development, and production) using containerisation tools such as Docker alongside cloud services like Railway or Fly.io, which ensures durability against infrastructure changes or failures (Dragoni et al., 2017).

Disaster recovery plans include scheduled database backups via Supabase (sesto-dev,2025), retry mechanisms for Celery tasks through Redis jobs, and fallback options if the Judge0 API becomes unavailable. Proactive monitoring and alerting are achieved with tools like Redis monitoring, Sentry, and log aggregation using Supabase logs or the ELK stack. Security and compliance are maintained through JWT authentication, HTTPS, and encrypted data storage, with modular updates planned to accommodate evolving regulations such as POPIA.

Finally, resource management and auto-scaling features provided by platforms like Render and Railway ensure that backend services adjust dynamically to user traffic, preventing performance degradation during peak periods (Mugisha and Ssenkusu, 2025).

This approach guarantees that ReCode’s primary goal (providing immediate, practical reinforcement of lecture content) is preserved and improved, maximising the platform’s institutional value and longevity by keeping it compliant, relevant, and effective under both anticipated and unforeseen changes.

### 4.2.2 Assessment

Impact alignment was conducted using SWOT and PESTEL analyses, where platform components were mapped against real-world Political, Economic, Social, Technological, Environmental, and Legal scenarios to assess their potential effects. To ensure robustness, stress testing and load simulations were planned using tools like Locust to evaluate whether the system’s performance would deteriorate under spikes in usage, such as multiple concurrent code uploads, slide parsing, and badge generation ( Bhagya, 2023). Failure injection scenarios involved employing mock services and retry mechanisms to test the platform’s recovery capabilities under simulated infrastructure failures, including code execution API timeouts and Redis crashes. Additionally, change forecast mapping was carried out to evaluate the system’s adaptability to critical risks such as a decline in faculty involvement, shifts in NWU’s technology policies, and the potential loss of access to free-tier hosting services (Salimi, 2025).

### 4.2.3 Results

|  |  |  |
| --- | --- | --- |
| Solution | Feasibility Score | Assessment Method |
| Scalability and Resilience | 8/10 | Change Forecast Mapping, Load Testing, Failure Simulation, PESTEL Alignment. |

### 4.2.4 Risks

|  |  |  |  |
| --- | --- | --- | --- |
| Risk Description | Risk Likelihood | Risk Impact | Actions Required to Mitigate Risk |
| Free-tier limits are modified by (Render/Railway) hosting platforms | Medium | High | App containerisation, multi-cloud migration support and cost forecasting. |
| Microservices' overhead delays initial rollout of the system | Medium | Medium | Give the most important microservices priority; use a monolith temporarily if necessary. |
| External APIs (Judge0, n8n) become unavailable or are deprecated | High | High | Utilise local fallback containers as well as Celery retries to decouple execution. |
| Increased inter-service traffic, which results in data breaches | Low | High | Use scoped tokens and HTTPS to enforce the API gateway and monitor logs. |

### 4.2.5 Issues

|  |  |  |
| --- | --- | --- |
| Issue Description | Issue Priority | Actions Required to Resolve Issue |
| Modular architecture results in a slightly longer dev time | Medium | Limit unnecessary features in ReCode and create modules in parallel |
| Insufficient knowledge of Redis and Celery among the NWU dev team | Medium | Conduct onboarding meetings and provide clear service documentation |
| Concurrent slide parsing causes a performance drop | High | Batch parse jobs and add cache for content that is frequently retrieved |

### 4.2.6 Assumptions

* Supabase and its services (Auth, Storage, DB) are accessible through NWU.
* Free-tier services (Render, Railway) remain accessible throughout the ReCode launch.
* At least once a week, lecturers will contribute to uploading content.
* The code engine Judge0 or similar will continue to be available or containerised.
* The skill level of the present development team can support FastAPI, Celery, Redis and microservice architecture.
* Students at NWU have regular access to the internet and devices that can run the web version of ReCode.

## 4.3 Solution 2: Code Visualisation Sandbox

### 4.3.1 Description

This solution is an interactive programming environment that visually represents the execution of code one step at a time for the students' benefit. This allows students to see exactly how the variables that are used change, how the loops iterate, and how functions execute in real time. The platform aims to simplify abstract coding concepts by providing a clear, animated representation of program flow, logic, and state changes. According to Zhang A.G. et al, the use of visual representations makes it simple to navigate program code flows.

Key features of the platform include step-by-step execution, allowing learners to pause, rewind, and fast-forward through their code to better understand each operation. It provides real-time tracking of variable states and memory allocation changes, helping students visualise how data evolves during program execution. To accommodate diverse curricula, the platform supports multiple programming languages such as Python, JavaScript, and Java. Additionally, it offers debugging assistance by highlighting errors and suggesting possible corrections, making it easier for learners to identify and fix mistakes. The platform also includes code-sharing functionality, enabling students to share their code and visualisations with peers or instructors for collaborative feedback and support.

### 4.3.2 Assessment

The Code Visualisation Sandbox directly addresses the learning barrier many beginners may face when learning the art of programming: the difficulty of understanding what the computer is doing when running their code. This visual approach caters especially to visual learners and those struggling with abstract logical reasoning.

The proposed solution offers several advantages in enhancing programming education. It makes abstract programming concepts more tangible by allowing students to visualise how code behaves in real time. “Regular use of visualisation tools in class is rather rare: approximately 20% of programming courses use software visualisations regularly” (Isohanni, E. et al.). This, coming from the research, suggests that there is a gap in this area. This approach encourages experimentation without the fear of breaking code, fostering a safe environment for learning through trial and error. Additionally, it improves debugging skills by clearly showing cause-and-effect relationships, which helps learners understand why specific issues occur. The tool is versatile and can be used in both self-paced learning and classroom environments, providing flexible learning opportunities. “In the visualisation graphs, it will be possible to observe the student’s performance in the activities carried out, where, from a bar graph, their assertiveness can be measured against the expected context.” (Santos, R.A. et al, 2023).

However, there are challenges to consider. Complex visualisations may require significant computing resources, which could limit accessibility on low-end devices. Moreover, there is a risk that learners might become overly reliant on visual tools, potentially slowing their ability to develop internal logic reasoning, an essential skill for traditional assessments such as written exams. “Behaviour-related features, such as total time spent on theoretical and practical contents and forums, can be used to find clusters of procrastination and thus to focus students” (Silva, D. et al., 2023). A study on entropy in human behaviour shows us that different students have different ways of learning. A challenge here that is observed is that some students may struggle, so there may have to be systems set in place to help them navigate and use the software efficiently.

### 4.3.3 Results

|  |  |  |
| --- | --- | --- |
| Solution | Feasibility Score | Assessment Method |
| Code Visualisation Sandbox | 6/10 | Student Engagement, Concept Retention, Error Reduction, Adaptability |

### 4.3.4 Risks

|  |  |  |  |
| --- | --- | --- | --- |
| Risk | Risk Likelihood | Risk Impact | Mitigation Strategy |
| Over-reliance on visualisation, slowing independent reasoning. | Medium | Medium | Integrate gradual reduction of visualisation support as students’ progress. |
| High resource usage for complex programs | Medium | High | Optimise visualisation rendering and limit step execution for large code. |
| Limited offline usability | Low | Medium | Develop a lightweight offline version or local execution mode. |
| Steeper learning curve for educators unfamiliar with the tool | Medium | Low | Provide comprehensive instructor onboarding and tutorials. |

### 4.3.5 Issues

|  |  |  |
| --- | --- | --- |
| Issue Description | Issue Priority | Actions Required to Resolve the Issue |
| Some advanced features may require paid subscriptions to sustain development. | Medium | Identify which features require subscriptions and explore free or open-source alternatives. If unavoidable, include subscription costs in the budget proposal. |
| Potential compatibility issues with older browsers or low-end devices. | High | Conduct testing across various browsers and hardware specs. Optimise code for low-resource devices and provide fallback options where needed. |
| Limited adoption if educators are resistant to changing their teaching style. | High | Provide onboarding support, training sessions, and case studies that show benefits of teaching workflow. Involve educators early in the platform’s design process. |

### 4.3.6 Assumptions

* Students have access to devices with reliable internet connections.
* Instructors are willing to integrate the sandbox into their teaching workflow.
* The platform will be regularly updated to support evolving programming languages and best practices.

## 4.4 Solution 3: Use Open-source code to build the system

### 4.4.1 Description

The third solution is to construct the system using one of the many easily accessible open-source codes. Using an open-source E-learning platform as a starting point, the code will be modified to satisfy the requirements needed to address the students' issues.

The software that will be used to build the system is called Moodle; a comprehensive, secure and integrated system for developing learning environments. Moodle is a learning management system that benefits both lecturers and students. It is used by universities, schools, corporations and non-profit organisations to assist them in teaching (Moodle, 2025; García-Peñalvo and Conde, 2014).

Lecturers use this platform to upload the course they are teaching or wish to teach. Under the course they teach, they can post interactive tests, PDF resources, and video lessons. Thereafter, they can monitor progress and provide due dates and grading criteria for the lessons. To receive certificates at the end of the course, students need to register and take part in numerous lessons (Moodle, 2025).

The system would have to be modified for our system by integrating it with the student’s timetable so that the lessons are made available as they appear on the timetable. Then, after the lesson has been taught, the quizzes and practice questions would then be available right after class (Sarkar, 2024).

### 4.4.2 Assessment

To assess the system's effectiveness, a small-scale version will be developed. Additionally, the system's compatibility with the student's schedule will be evaluated. To assess how effectively the module teachings fit into the students' schedules and how quickly the practice questions and quizzes are accessible following each module, one module will be uploaded to the system (Sarkar, 2024; Gamage, Ayres and Behrend, 2022).

### 4.4.3 Results

|  |  |  |
| --- | --- | --- |
| Solution | Feasibility Score | Assessment method |
| Moodle | Medium | The feasibility of the system was assessed by how well it meets the requirements needed for the system; how easy it will be for students to use it, and the time required for the system to become fully operational. Factors that were considered are cost, scalability, ease of integration and the availability of technical support. The Moodle system solution received a medium feasibility score, which indicates how feasible the system is (Moodle,2025) |

### 4.4.4 Risks

|  |  |  |  |
| --- | --- | --- | --- |
| Risk Description | Risk Likelihood | Risk Impact | Actions required to mitigate risk |
| Compliance issues are caused by using licenses that are incompatible with the software, which Moodle is licensed under when modifying and using the system. | High | High | Ensure that all open-source components’ licenses are tracked and documented. The system developers need to be aware of the license the system uses |
| The system may encounter bugs and errors that may lead to system downtime. This will lead to students not being able to use the system and practice their skills. | Medium | High | Do thorough bug testing and have a maintenance plan. Have good comments on the code and provide technical support for troubleshooting. |

### 4.4.5 Issues

|  |  |  |
| --- | --- | --- |
| Issue Description | Issue Priority | Actions required to resolve the issue |
| Teachers and students may have difficulty using Moodle. Therefore, teachers may struggle with setting up quizzes, conditional activities, and grading requires training or documentation. | High | Provide teachers with the online training course that Moodle has on how to use the system. Have additional tutorial videos and documentation on how to use the system since it has been modified. |
| There might be a version mismatch with the web server and the database, which can cause installation failures or site crashes. | High | Ensure that the web server and the database are updated and constantly check if they are compatible. |

### 4.4.6 Assumptions

* System developers are familiar with using PHP and can set up PHP applications.
* The system will be able to integrate with the existing systems in NWU.
* Students and teachers have reliable internet access to use the platform effectively since it is web-based
* Students are willing to use the platform to improve their coding skills, and teachers are willing to put in extra effort to ensure that students understand the content of their modules.
* There will be ongoing technical support for Moodle updates, backups, and troubleshooting.
* Learning materials, including course content and assessments, will be available or created promptly for upload into the Moodle system.
* The Moodle installation will be configured to comply with relevant data protection laws and institutional privacy policies (e.g., POPIA).
* User data will be securely stored and protected using appropriate access controls, encryption, and authentication mechanisms.
* Any custom themes, plugins, or integrations will be compatible with the installed version of Moodle and will be tested before deployment to prevent system instability.

# 5. Feasibility Ranking

## 5.1 Ranking Criteria

The three possible solutions were ranked using defined criteria. Each solution is scored out of 5 for each criterion, and the product of the score and weight (representing the relevancy of the criterion) is calculated. The criteria are as follows:

**Maintainability**: The ease with which the system can be maintained and upgraded after delivery. A higher score indicates better long-term support and easier improvements.

**Scalability**: The system’s ability to grow across users, sites, and features. A scalable system supports long-term sustainability and can accommodate institutional growth.

**Affordability**: How cost-effective the system is to develop, deploy, and maintain. A higher score means reduced financial burden.

**User-Friendliness**: How intuitive the system is for both staff and students, and how effectively it supports learning outcomes.

**Development Pace**: How quickly the solution can be delivered without sacrificing quality, allowing for rapid implementation and feedback.

## 5.2 Ranking Scores

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Solution** | **ReCode** | | | **Visual Sandbox** | | | **Open Source Base** | | |
| **Criteria** | **Score** | **Weight** | **Total** | **Score** | **Weight** | **Total** | **Score** | **Weight** | **Total** |
| Maintainability | **5** | **25%** | **1.25** | **4** | **25%** | **1** | **3** | **25%** | **0.75** |
| Scalability | **4** | **25%** | **1** | **4** | **25%** | **1** | **3** | **25%** | **0.75** |
| Affordability | **4** | **20%** | **0,8** | **3** | **20%** | **0,6** | **4** | **20%** | **0.8** |
| User- friendliness | **3** | **10%** | **0,3** | **4** | **10%** | **0,4** | **3** | **10%** | **0,3** |
| Development Pace | **4** | **20%** | **0,8** | **3** | **20%** | **0,6** | **2** | **20%** | **0,4** |
| **Total Score** | **20** | **100%** | **4.15** | **18** | **100%** | **3,6** | **15** | **100%** | **3** |

# 6. Feasibility Result

The most feasible solution based on the results of the feasibility ranking is **ReCode**. It outperforms the other options across all major criteria, especially in maintainability, scalability, and development pace. Designed with a microservice architecture, robust deployment strategy, and modular security, the MVP ensures that ReCode remains resilient and adaptive under both macro (e.g., AI integration, policy shifts) and micro (e.g., staff changes, module restructuring) institutional pressures.

This MVP implementation is tightly aligned with NWU's core goals: to reinforce lecture content in real time, remain affordable and accessible to students, and support long-term maintenance. Its adaptability makes it a forward-thinking solution that will continue to serve both educators and students even as needs evolve.

The second-ranked solution is the **Visual Sandbox**, a highly interactive tool designed to deepen learners’ understanding of code execution. While it provides excellent user-friendliness and visualisation tools, it requires more development time and resources, slightly reducing its overall feasibility at this stage. It remains a promising option for future integration into ReCode as a visual aid module.

The third-ranked solution is the **Open-Source Base**, which involves adapting an existing open-source LMS like Moodle. This approach is attractive for its affordability and foundation of ready-made features, but it poses limitations in terms of customisation speed, integration with NWU’s unique structure, and user responsiveness. While useful for broader curriculum management, it may not meet ReCode’s primary goal of immediate, practical reinforcement of in-class content.

# 7. Implementation Plan

## Implementation Strategy

### Phase 1 - MVP

* Python support only.
* Core functionality: slide parsing, NLP topic extraction, challenge generation, instant grading.
* Core frontend functionality: UI for slide upload, real-time challenge display, instant grading feedback.
* Technologies: Vue 3, Tailwind CSS, Shadcn, Vite for fast dev.
* Basic user authentication and session management integrated with the backend API.

### Phase 2 -Expansion

* Add JavaScript and Java support.
* Extend frontend to support multiple languages (UI components for JavaScript and Java challenges).
* Introduce gamification (leaderboards, badges, progression).
* Improve NLP accuracy and adaptability.
* Lecturer dashboard: interactive UI for manual curation of topics, analytics charts, and challenge adjustments.  
  Enhanced frontend error handling and notifications.

### Phase 3 -Scaling

* Extend to other faculties and institutions.
* Add advanced analytics, collaborative coding features and export options.

To ensure that ReCode is developed efficiently and meets its intended goals, the implementation of the project will follow an Agile-Scrum methodology with a feature-based development approach. This allows for iterative releases, continuous feedback, and adaptability to the evolving needs of lecturers and students while maintaining momentum.

## Project Team

* **Project Manager:** Mohau Liphoko
* **Team Lead:** Brandon van Vuuren
* **Backend Developers**: Brandon van Vuuren (Backend Team Lead), Kay Matlakaneg, Vonani Nkombyane, Caitlin Calder and Themba Mabuza
* **Frontend Developers**: Lené Kriel (Frontend Team Lead), Annemé Janse van Rensburg, Ethan Afrika, Altus Aucamp
* **Business Analysts**: Kayla Supra (Business Analyst), Sinovuyo Sondara, Lwazi Lata, Tumi Phororo

## Project Phases

### 1. Planning

Under the approval of the project manager, planning sessions will define the scope, objectives, timelines, and resource allocation. A project roadmap will be finalised before development begins. Figma wireframes will be developed during this phase to guide the frontend team’s UI/UX design and to provide the backend team with clear insights into the data and APIs the frontend will require.

### 2. Development - Conducted in phased 2 - 3 week sprints

### Phase 1 - MVP Development (Months 1-6)

* **Core Technologies:**
  + Python backend stack (python-pptx, SpaCy, Judge0, Supabase, Redis/Celery).
  + Frontend stack: Vue 3, Tailwind CSS, Vite, basic authentication modules.
* **Core Functionality:**
  + Slide content extraction, NLP topic mapping, coding challenge generation, and automated grading with instant feedback.
  + Frontend features: slide upload UI, challenge display, instant grading feedback, basic user login.
  + Backend API endpoints supporting frontend requirements.
  + Initial integration between frontend and backend.
* **Pilot Deployment:**
  + Limited rollout within the first-year programming module at NWU.
  + Collect user feedback (students and lecturers) to validate core functionalities and usability.
* **Success Metrics:**
  + 50% reduction in lecturer preparation time
  + 30% increase in student challenge participation
  + Stable uptime above 99%
  + Positive usability feedback on frontend interfaces.

### Phase 2 - Feature Expansion (Months 7-12)

* Additional language support: JavaScript and Java (for 2nd-year students).
* Gamification features: leaderboards, badges, and ELO-style ranking.
* Enhanced NLP accuracy with additional training data.
* Lecturer dashboard for manual curation, analytics, and topic weighting.
* Improve UI/UX based on Phase 1 feedback.
* Error handling and user notifications.

### Phase 3 - Scaling and Partnerships (Months 13-24)

* Deployment across multiple NWU modules and faculties.
* Partnerships with other universities and coding academies.
* Integration with learning management systems (e.g., Moodle, Canvas).

### 3. Testing

* Unit and integration tests for backend components and frontend UI.
* End-to-end tests covering user flows (upload → challenge → grading → feedback).
* Automated UI testing integrated into CI/CD pipeline.
* Usability testing sessions with real users at each phase.

### 4. Deployment

* Deploy thoroughly tested versions to production environments with rollback capabilities.
* Provide end-user documentation, training videos, and onboarding materials.
* Set up monitoring and alerting for uptime and performance metrics.
* Plan phased rollouts with clear feedback loops and support channels.

### Execution Practices

* **2-3 Weekly Sprints:** Progress meetings on Fridays to assess progress, address issues, and gather feedback from the project manager.
* **GitHub Project Management:** Centralised tracking of tasks and deadlines.
* **CI/CD Pipeline:** Automated build, testing, and deployment via GitHub Actions.
* **Progress Reporting:** Reports prepared for major deliverables to ensure transparency and gather feedback.

### Communication & Collaboration Plan

* **Primary Communication (Official Updates & Documentation)** - Microsoft TeamsUsed for official project announcements, meeting scheduling, progress reports, and sharing project documents.
* **Secondary Communication (Day-to-Day Coordination)** - WhatsApp GroupsSeparate community groups for backend, frontend, and business analysis teams to coordinate daily tasks and share short updates.
* **Real-Time Collaboration (Problem-Solving & Code Reviews)** -DiscordUsed for quick voice/video calls, live coding sessions, and screen sharing when resolving technical issues or conducting pair programming.
* **Version Control & Task Tracking** -GitHub All code is committed via GitHub with pull request reviews. GitHub Projects or Issues are used for tracking tasks and progress.

**Dependencies & Prerequisites**

* Access to NWU course materials (slides for NLP model training).
* Judge0 server instance or API access.
* Supabase environment and credentials.
* Development machines with required software packages installed.
* Figma design files and wireframes for consistent UI/UX implementation.
* Node.js and npm installed for frontend development.
* Access to API documentation for integration between the frontend and backend.
* Access to test devices and browsers for cross-platform UI testing (desktop, tablet, mobile).

## Appendix A: Recode

|  |  |  |  |
| --- | --- | --- | --- |
| **Category** | **Description** | **cost** | **Pricing Sources** |
| Development Costs | Salary for developers skilled in FastAPI, Celery, and Redis | R285,000 - R570,000 | [PayScale](https://www.payscale.com/research/ZA/Job=Software_Developer/Salary) |
| Cloud Infrastructure | Hosting services, e.g. Railway | R1,900 - R5,700 per month | Railway: [check source here](https://railway.app/pricing)  [Fly.io:](https://fly.io/pricing)  [Render:](https://render.com/pricing) |
| Database Services | Supabase for Database, Storage backups | R950 - R3,800 per month | [Supabase pricing:](https://supabase.com/pricing) |
| Monitoring Tools | Sentry, Redis monitoring, and Supabase logs | R0 - R2,850 per month | [Judge0](https://ce.judge0.com/):  [n8n cloud:](https://n8n.io/pricing) |
| Security and compliance | HTTPs certificates (Let's Encrypt is free) | R0 - R1,900 per year | [Let’s Encrypt](https://letsencrypt.org/): |
| Training | Developer training on Redis, Celery, etc | R38,000 - R95,000 | Local trainers, [Udemy courses](https://www.udemy.com/topic/docker/) |
| Maintenance | Bug fixes, backups, and disaster recovery | R19,000 - R57,000 per month | [Estimated support contracts](https://itadon.com/blog/how-to-implement-a-backup-and-disaster-recovery-process-a-step-by-step-guide/) |
| Load Testing Tools | Locust | R0 - R950 per month | [Locust is open-source](https://locust.io/) |
| Miscellaneous Costs | Contingency, licences, etc | R9,500 - R28,500 | Varies depending on software and plugins used e.g. [plugin license](https://fastercapital.com/) |

***Total Initial Cost of 3-4 months development:*** *R335,000 - R670,000*

***Monthly Recurring Costs:*** *R24,000 - R73,000*

## Appendix B: Code Visualisation Sandbox Cost Analysis

|  |  |  |  |
| --- | --- | --- | --- |
| **Type** | **Cost Item** | **Description** | **Price** |
| **Personnel** | Developer Salaries | Average R40 654 per month × 3 devs × 6 months | R731 772 |
|  | UX/UI Design | One designer, R47 635 per month × 3 months | R142 905 |
|  | Project Management | PM at R47 635 per month × 6 months | R285 810 |
|  | Training & Dev | Course licenses | R0 (free resource) |
|  | Miscellaneous | Supplies, internet, etc. | R12 000 |
| **Infrastructure** | Hardware | Laptops + monitors for developers (3 units) | R69 000 |
|  | Cloud Hosting & Storage | AWS mini stack & storage | R1 000 – R2 000/month |
|  | Backup Systems | GitHub free tier, minimal costs | R0 |
| **Software Licenses** | Visualisation Library | Optional premium visualisation library license | R10 000 |
| **TOTAL** |  | **One-Time Costs** +  **6-Month Ops** | **R1 241 487 +**  **R6,000/month** |

## Appendix C: Moodle software cost analysis

|  |  |  |
| --- | --- | --- |
| **Expense Category** | **Expense Description** | **Expense value** |
| Development and Customisation | The most important cost will be the salary for the developers who will help modify the system. According to Glassdoor (2025), an average salary for front-end developers and system developers is R34,000/month and R31,000/month, respectively. Business analysts make an average of R32,000/month, while project managers make R43,000/month on average. For this project, there is one project manager, four business analysts and four front-end developers, and five back-end developers. | R2,772,000 |
| Training and development | Students and teachers will need to get training on how to use the system. The most cost-effective way is to produce a tutorial video that will be hosted on YouTube. Users of the system can then refer to the video whenever they want to. Video production for a mid-range video that will last for 5-15 mins costs R25,000-R75,000. This will probably involve several days of filming, expert editing, and obtaining rights for music use (Astral Studios,2022). | R50 000 |
| Software Licence | There are no licensing costs since Moodle is an open-source system. The upgrades received are also free. | R0 |
| Security and Backup Procedures | We’ll use GitHub to keep backups of the software so we can track changes, keep multiple copies, and store important project files securely. Features like repositories, branches, and commit history mean we’re protected if the system fails or if something gets deleted by mistake. The repositories will be private so that access to the software is restricted to authorised members only. By using GitHub, there will be no extra costs for backup and security. | R0 |
| **Total** |  | R2,822,000 |

# References

Masegosa, A. R., Cabañas, R., Maldonado, A. D. & Morales, M. (2024). Learning Styles Impact Students’ Perceptions on Active Learning Methodologies: A Case Study on the Use of Live Coding and Short Programming Exercises. Education Sciences, 14(3), art. 250. Available at: <https://doi.org/10.3390/educsci14030250>

**App B**  
Jobted, n.d. Jobted – salary and job information. Available at: https://za.jobted.com [Accessed 9 August 2025].  
School of IT, n.d. School of IT – courses and information. Available at: https://www.schoolofit.co.za [Accessed 9 August 2025].  
N2W Software, n.d. N2W Software – cloud backup solutions. Available at: https://n2ws.com [Accessed 9 August 2025].  
CloudZero, n.d. CloudZero – cloud cost intelligence. Available at: https://www.cloudzero.com [Accessed 9 August 2025].

**App C**  
Glassdoor, 2025. Front-end developer salaries. Available at: <https://www.glassdoor.com/Salaries/front-end-developer-salary-SRCH_IM1023_KO0,19.htm> [Accessed 9 August 2025].  
Astral Studios, 2022. How much does a video production cost? Available at: <https://www.astralstudios.co.za/how-much-does-a-video-production-cost/> [Accessed 9 August 2025].

**Sol 1**  
Unknown author, n.d. Toward sustainable education: A contextualized model for educational technology adoption for developing countries. [Accessed 7 August 2025].  
Unknown author, n.d. SWOT analysis of blended learning in public universities of Uganda: A case study of Muni University. [Accessed 7 August 2025].  
Salimi, F., 2025. Aligning policy and practice: The World Bank’s approach to EdTech in Sub-Saharan Africa. [Accessed 7 August 2025].  
sesto-dev, 2025. supabase-database-backup. GitHub. Available at: <https://github.com/sesto-dev/supabase-database-backup> [Accessed 10 August 2025].

**Sol 2**  
Santos, R.A., Barbi, D., Ramos, V.F.C. & Gauthier, F.A.O., 2023. Data visualization for learning analytics indicators in programming teaching (short paper). In: *4th International Computer Programming Education Conference (ICPEC 2023)*, OASIcs, Volume 112, pp. 10:1–10:7.  
Hayatpur, D., Xia, H. & Wigdor, D., 2023. CrossCode: Multi-level visualization of program execution. *arXiv preprint*. Available at: <https://doi.org/10.48550/arXiv.2304.03445> [Accessed 10 August 2025].  
Zhang, A.G., Tang, X., Oney, S. & Chen, Y., 2024. CFlow: Supporting semantic flow analysis of students' code in programming problems at scale. *arXiv preprint*. Available at: <https://arxiv.org/abs/2404.10089> [Accessed 10 August 2025].  
Silva, D. et al., 2023. Students’ learning behaviour in programming education analysis: Insights from entropy and community detection. *Entropy*, 25(8), Article 1225.

**Sol 3**  
Moodle, 2025. About Moodle. Available at: <https://docs.moodle.org/500/en/About_Moodle> [Accessed 8 August 2025].  
García-Peñalvo, F.J. & Conde, M.Á., 2014. E-learning platforms in higher education: Case study. *Procedia – Social and Behavioral Sciences*, 141, pp.1106–1110. Available at: <https://www.sciencedirect.com/science/article/pii/S187705091400550X> [Accessed 10 August 2025].  
Sarkar, B., 2024. How Bayes Business School maintained quality education and smooth transition to Moodle LMS 4. Moodle. Available at: <https://moodle.com/news/moodle-lms-bayes-business-school/> [Accessed 10 August 2025].  
Gamage, S.H.P.W., Ayres, J.R. & Behrend, M.B., 2022. A systematic review on trends in using Moodle for teaching and learning. *International Journal of STEM Education*, 9(1), p.9. Available at: <https://stemeducationjournal.springeropen.com/articles/10.1186/s40594-021-00323-x> [Accessed 10 August 2025].