Project Plan

Superior Planning System

IT3C

Ian Donker - Team Leader

Timothy Adewale - Scrum Master

David Hlavacek – Code Reviewer

Stefan Tasca - Secretary

Batuhan Gezgin - Developer

A blue and black logo

Description automatically generated

## Table of contents

[Table of contents 2](#_Toc257702073)

[Version Control 2](#_Toc171274409)

[Foreword: 2](#_Toc1871645293)

[Chapter 1: Introduction & Background 2](#_Toc403984427)

[Chapter 2: Project Results 3](#_Toc568101479)

[Chapter 3: Project Activities 5](#_Toc2074456882)

[Sprint Schedule Overview 6](#_Toc814567749)

[Sprint 1: Initial Setup and Basic Framework Development 6](#_Toc1115102482)

[Sprint 2: Core Feature Development 7](#_Toc1857571139)

[Sprint 3: Enhanced Functionality and Validation 8](#_Toc646099386)

[Sprint 4: Final Testing, Deployment, and Handover 8](#_Toc150772266)

[Chapter 4: Intermediate Results 9](#_Toc1652043481)

[Chapter 5: Project Boundaries 11](#_Toc244579670)

[Chapter 6: Quality Control 14](#_Toc2092067508)

[Chapter 7: Project Organization 14](#_Toc522554252)

[Chapter 8: Planning 18](#_Toc1165749799)

[Chapter 9: Cost and Benefits 19](#_Toc1501813884)

[Costs 20](#_Toc1166557654)

[Benefits 20](#_Toc1460816211)

[Chapter 10: Risk Analysis 21](#_Toc923093004)

[Chapter 10: Appendix 24](#_Toc835314796)

[10.1 Value Chain Analysis 24](#_Toc1341354832)

[10.2 GAP Analysis 25](#_Toc45535018)

## Version Control

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Date | Content | Editor(s) |
| 0.1 |  |  |  |
| 0.2 | 10/05/25 | Added Chapter 1 | David Hlavacek |

## Chapter 1: Introduction

NHL Stenden University of Applied Sciences is a Dutch institution offering practice-oriented higher education. As part of the Secure Programming minor (Academic Year 2024-2025), student teams are tasked with building real-world applications that integrate security best practices throughout the software development life cycle.

Today’s competitive gamers—and the community managers who support them—must track performance across multiple titles and genres using spreadsheets, disparate websites, or ad-hoc tools. This fragmented approach is time-consuming, prone to data inconsistency, and raises privacy concerns when usernames or third-party credentials are exposed.

Over the coming eight-week period, the team will work under the guidance of module coordinator Rob Loves and lecturer Remco Hassing. Rob and Remco act as both internal stakeholders and product owners; external stakeholders are the players, e-sports teams, and community organizers who will rely on our app to view individual game metrics, genre-level summaries (FPS, MOBA, RPG, etc.), and a consolidated “global” performance dashboard (Profile).

#### Project Details:

* **Project Name:**
* **Project Manager:**
* **Project Supervisors:** Rob Loves and Remco Hassing

The ultimate goal of this project is to replace manual stat-tracking workflows with a secure, maintainable Android application—implemented in Kotlin with Jetpack Compose—that fetches data from public game-API providers, encrypts sensitive information at rest and in transit, and offers an intuitive interface for gamers to monitor and improve their performance. The detailed requirements and design decisions follow in the chapters below.

## Chapter 2: Project Results

The problem

The team has noticed a growing trend among players who actively track their in-game performance across multiple competitive titles. These players often juggle stats from different games, switching between third-party stat websites, forums, and spreadsheets to monitor progress, compare metrics, and decide what to focus on next. The experience is fragmented, inconsistent, and lacks meaningful cross-game insights.

What makes the problem even worse is that there’s no single tool that pulls all this data together and makes sense of it, like by game genre, play style, or personal goals. Players often don’t notice when their performance starts to drop or when they've been playing for way too long. Plus, there's no personalized, mobile-friendly experience that fits how they play or what games they focus on.

### Solution

To meet this growing need, the team will design and develop a cross-game stat tracking mobile application, built specifically for Android using Kotlin, by June 20th. The goal is to give players a centralized and responsive interface to monitor performance, manage their game profiles, and receive alerts based on custom stat thresholds.

**The app allows users to:**

* Browse and search a curated list of supported games, filterable by genre.
* Add and manage usernames per game, with real-time validation.
* View detailed stats per game, genre-level performance, and global aggregates.
* Receive push notifications when performance dips below or rises above personalized thresholds.
* Customize their profile with display names, avatars, and genre preferences.

**Behind the scenes, the backend:**

* Is built using scalable technologies such as Spring Boot, FastAPI, or Express.
* Integrates securely with game-specific APIs to fetch user stats.
* Uses PostgreSQL with encryption and Redis for fast stat caching.
* Serves a clean RESTful API structure for user management, game data, and notifications.

**Boundaries & Ethical Scope**

* The application is strictly read-only with respect to game data; it cannot alter any in-game progression or real-time match data.
* It collects only essential user data (usernames and preferences), ensuring high privacy standards with AES-256 encryption and TLS 1.2+ transport security.
* It does not handle financial transactions or health-related data.

Value Proposition  
By offering this centralized and customizable experience, the app not only empowers players with data-driven insights into their performance but also supports healthier play habits through proactive alerts. Whether a casual player trying to improve or a competitive gamer optimizing strategy, the app serves as a useful tool.

## Chapter 3: Project Activities

## **Chapter 3 Project Activities**

This chapter provides a comprehensive overview of the project activities, organized into weekly sprints that guide the team's efforts throughout the duration of the project.

### **Sprint-Schedule Overview**

|  |  |  |
| --- | --- | --- |
| **Sprint** | **Dates** | **Demo / review date** |
| **Sprint 1** | 12 May – 25 May 2025 | Fri 23 May |
| **Sprint 2** | 26 May – 8 Jun 2025 | Fri 6 Jun |
| **Sprint 3** | 9 Jun – 22 Jun 2025 | Fri 20 Jun |
| **Sprint 4** | 23 Jun – 6 Jul 2025 | Thu 3 Jul (final) |

### **Sprint 1: Planning & Foundations**

### **Objectives:**

* Define the app concept and architecture.
* Prepare planning, documentation, and tools.
* Begin security integration via Threat Modeling.

**Activities:**

* Brainstorm and finalize app concept with team.
* Define main and sub-features of the application.
* Create product backlog and user stories in Scrum tool (GitLab).
* Setup GitLab repository and branching strategy.
* Create initial README, code of conduct, license file.
* Define security goals.
* Create **Threat Model** (STRIDE or OWASP).
* Create Planning documents:
  + Application concept document
  + Plan of Action (PoA)
  + Secure SDLC strategy (what secure practices will be used in each phase)
  + List of external APIs and OpenData sources.

**Deliverables:**

* App concept document
* Threat Modeling
* Planning document
* Initial Git repo with README

### **Sprint 2: Secure Development – Backend & Frontend (Week 3–4)**

**Objectives:**

* Develop core backend and frontend features.
* Integrate secure coding principles from OWASP.
* Apply authentication method.

**Activities:**

* Design app screens (wireframes/mockups).
* Implement Kotlin Android app architecture (MVVM preferred).
* Build UI screens (login, dashboard, stats view).
* Set up secure authentication (Firebase Auth, OAuth2, etc.).
* Use **OpenData APIs** and **2 external APIs** (e.g., game stats).
* Apply secure coding practices:
  + Input validation
  + Secure storage (SharedPreferences with encryption)
  + Avoiding hardcoded secrets
* Track all commits in Git.

**Deliverables:**

* Functional app skeleton
* Authentication setup
* API Integration
* Secure programming implementation evidence

### **Sprint 3: Cryptography, Secure Testing & Code Review (Week 5–6)**

**Objectives:**

* Apply **Cryptography** and secure testing practices.
* Conduct **Code Review** and improve code quality.

**Activities:**

* Integrate cryptographic methods.
* Implement secure local storage using encryption.
* Perform unit tests, UI tests.
* Peer code review sessions (mandatory).
* Document security measures taken.
* Apply **static code analysis** and linters (e.g. Android Lint).
* Draft report chapters:
  + Secure Programming
  + Cryptography
  + Secure SDLC
  + Code Review
  + Testing

**Deliverables:**

* Secure crypto implementation
* Testing & code review documentation (mandatory appendix)
* Updated Git history with tags/releases
* Draft report with technical chapters

### **Sprint 4: Final Integration, CI/CD, Report & Presentation (Week 7–8)**

**Objectives:**

* Finalize application, set up CI/CD.
* Complete documentation and report.
* Prepare and deliver the presentation.

**Activities:**

* Set up CI/CD pipeline in GitLab:
  + Build trigger
  + Linter
  + Static code analysis
  + Test runner
* Finalize all features and UI.
* Conduct usability testing if time allows.
* Compile final report (12,000–16,000 words) with:
  + Summary (in Dutch or English depending on program)
  + Introduction
  + Planning/Methodology
  + Secure programming application
  + Cryptography
  + Secure SDLC
  + Threat modeling
  + Code review
  + Conclusion & recommendations
  + APA-referenced sources
* Deliver group presentation (5–10 minutes per member).
* Peer-assessment form for group contribution.

**Deliverables:**

* Working, secure app in GitLab with proper CI/CD
* Final report (PDF with appendices)
* Final presentation (PPT/video)
* Peer assessment (point distribution)

## Chapter 4: Intermediate Results

* **Sprint 1**
  + Focus: MVP scope definition, system and architecture planning.
  + Security Activities:
    - Initial threat modeling
    - GitLab repository and CI setup
    - Planning cryptographic use
  + Deliverables & Report Chapters:
    - Project plan
    - Product backlog
    - Draft threat model
    - Report Chapters: Introduction, Project Planning
* **Sprint 2**
  + Focus: Implementation of core user persona management.
  + Security Activities:
    - Cryptographic implementation
    - Static code analysis
  + Deliverables & Report Chapters:
    - Working persona module
    - Encrypted local storage
    - CI/CD pipeline foundation
    - Report Chapters: Concise Summary, Research, Secure Programming, Cryptography
* **Sprint 3**
  + Focus: Development of social features and secure login/auth.
  + Security Activities:
    - Secure communication implementation
    - Authentication logic
    - Expanded test coverage
  + Deliverables & Report Chapters:
    - Friend comparison functionality
    - Authentication system
    - Test reports
    - Report Chapters: Threat Modeling, Code Review, Secure SDLC
* **Sprint 4**
  + Focus: Final UI polish, integrations, and documentation.
  + Security Activities:
    - Secure UI review
    - Final threat model update
  + Deliverables & Report Chapters:
    - Final production-ready app version
    - Complete documentation
    - Code review summary
    - Report Chapters: Conclusions, References, Appendices

## Chapter 5: Project Boundaries

## Project Scope

The project delivers a Kotlin‑based Android application that serves as a lifelong stats vault for gamers. Its main goals are to

• collect public statistics—such as matches played, wins, kill/death ratio, total hours, and achievements—from at least two documented game‑platform APIs (e.g., Steam Web API and Riot Games API);

• merge those per‑game numbers into a single, durable “persona” profile so users can monitor their entire gaming history in one place;

• protect data integrity and privacy using a hardware‑backed Ed25519 key pair created on first launch, with every record stored in an encrypted SQLCipher database and individually signed;

• enable direct peer‑to‑peer comparison so friends running the app can exchange signed summaries and see comparative lifetime metrics without a central account system;

• provide an optional wellness panel that analyses total play hours and late‑night sessions, issuing in‑app alerts when unhealthy patterns appear;

• demonstrate the full secure‑programming workflow required by the minor, including threat modelling, static analysis, unit tests, code review, and a CI pipeline;

• produce a running APK, source code in GitHub, a secure‑programming report, and a final presentation in week 9.

## Inclusions (MVP)

• Connection to at least two official game APIs.

• Encrypted on‑device database with signed records.

• Automatic key‑pair generation, record signing, and verification.

• Friend comparison via signed JSON exchanges between devices.

• Dashboards: Lifetime Persona Overview and Play‑Time Wellness.

• Unit tests for cryptographic and parsing components.

• Threat‑model diagram, code‑review log, and CI workflow committed to the repository.

## Exclusions

The first release will deliberately leave out certain elements to keep the workload realistic:

• Backend or cloud synchronisation; all data remains on the user’s device.

• Versions for iOS, desktop, or web.

• Monetisation features such as ads, payments, or in‑app purchases.

• Streaming overlays, widgets, or OBS integration.

• Machine‑learning analytics, skill predictions, or other advanced data science.

• Scraping of undocumented or private endpoints—only officially documented APIs will be used.

• Full social‑network capabilities beyond direct friend comparison.

## Constraints

• Time: nine weeks, culminating in the final assessment and presentation in week 9.

• Module mandates: use two external APIs, apply demonstrable cryptography, include an open‑data element, perform threat modelling, code review, CI/CD, and implement an authentication mechanism.

• Resources: restricted to Android Studio, Kotlin, Jetpack Compose, and open‑source libraries approved by lecturers Rob Loves and Remco Hassing.

• Data availability: functionality depends on data exposed by the chosen APIs; if an API becomes unavailable or heavily rate‑limited, the linked feature will be postponed.

• Budget: tooling and hosting must be free or supplied by the university.

• Compliance: the app stores only anonymous gameplay statistics and follows GDPR principles, avoiding personal identifiers.

## Conclusion

These boundaries keep the project achievable within the nine‑week minor while still covering every secure‑programming requirement. By focusing on core aggregation, peer comparison, and wellness alerts on a single Android platform, the team can deliver a polished proof of concept and establish a solid foundation for future expansion to cloud sync, additional platforms, or advanced analytics after the assessment phase.

## Chapter 6: Quality Control

To deliver a high-quality, secure, and maintainable Android application, the team will implement rigorous quality checks throughout design, planning, development, and testing phases.

**Sprint Planning & Progress Tracking**

* Bi-weekly sprint planning and retrospectives to review completed work against acceptance criteria, identify blockers, and adjust priorities.
* Weekly (minimum) stand-ups for quick alignment; all tasks and bugs tracked in GitHub Issues under the Scrum Master’s oversight.

**Design Reviews**

* Initial screen sketches are made in Figma, then shared with the team and client for feedback.
* Before code starts, we confirm layouts are clean, easy to navigate, and match our style guide.

**Code Quality & Security**

* Enforce a shared Kotlin style guide via ktlint in the CI pipeline—any style error fails the build.
* Every merge request needs at least one approving code review before it’s merged.
* Run an automated dependency scan each build to spot known vulnerabilities.

**Testing**

* Unit tests cover all core logic and data-handling code.
* UI tests verify key flows (login, game list, stats view).
* CI builds are blocked if any test fails, ensuring new changes don’t break existing features.

**Performance & Reliability**

* Track app startup time and screen-load speeds.
* Use Android Studio profilers to catch memory leaks or excessive CPU/network use.

**Documentation & Release Notes**

* Maintain all setup guides, API specs, and architecture notes as Markdown in the repo.
* After each sprint, publish release notes summarizing new features, bug fixes, and security updates.
* Include screenshots or sample API calls for any major UI or endpoint changes.

By embedding these straightforward checks into our two-week sprints and making quality a shared responsibility, we’ll keep the app stable, secure, and on track.

## Chapter 7: Project Organization

**The job distribution of the team goes as follows:**

* Ian Donker - Team Leader
* Timothy Adewale - Scrum Master
* David Hlaváček- Code Reviewer
* Stefan Tasca - Secretary
* IBatuhan Gezgin - Developer

 Name: Batuhan Gezgin

Telephone number: +31 6 81979297

Email: batuhan.gezgin@student.nhlstenden.com

 Name: Timothy Adewale

Telephone number: +234 813 348 0312

Email: [adewale.adewale@student.nhlstenden.com](mailto:adewale.adewale@student.nhlstenden.com)

Name: Stefan Ionut Tasca

Telephone number: +40 787210982

Email: [stefan.tasca@student.nhlstenden.com](mailto:stefan.tasca@student.nhlstenden.com)

Name: David Hlaváček

Telephone number: +421 907 888 923

Email: david.hlavacek@student.nhlstenden.com

Name: Ian Donker

Telephone number: +31 6 46159508

Email: ian.donker@student.nhlstenden.com

## Stakeholder Groups

We categorize the stakeholders into the following groups:

• Manage Closely

• Keep Satisfied

• Keep Informed

• Monitor

1. Manage Closely  
Primary stakeholders are lecturers **Rob Loves** and **Remco Hassing**, together with the student project team.

Communication:  
• Face‑to‑face project meeting every **Wednesday** on campus during Atelier hours.  
• Team‑only Microsoft Teams meeting every **Monday at 18:00** for sprint planning; extra meetings arranged whenever necessary.  
• WhatsApp group (team only) for quick coordination and emergency alerts.  
• Minutes of each scheduled meeting are posted in the Teams channel within 24 hours.  
• Any blocker, scope change, or security incident is flagged immediately in WhatsApp and, if needed, followed by an urgent Teams call.

2. Keep Satisfied  
Stakeholders in this category include the NHL Stenden minor committee and any external reviewers who will assess the project.

Communication:  
• They receive formal updates through the lecturers and via the required project deliverables (interim demo, final presentation, and written report).  
• We provide access to milestone builds and documentation ahead of assessment dates so they can review progress without needing ongoing correspondence.

3. Keep Informed  
This group covers fellow students and other lecturers who may act as beta testers.

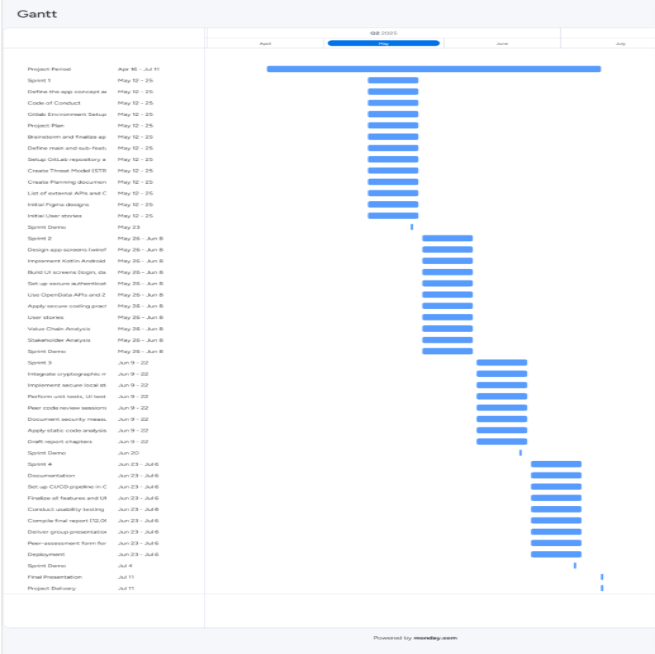
Communication:  
• Brief progress announcements are given during open Atelier moments.  
• Feature snapshots and calls for feedback are posted in the project’s public Teams space at the end of each sprint.  
• Usability‑test sessions are run near the close of each sprint, and findings are shared informally in class.

4. Monitor  
Stakeholders we simply observe include game‑platform API providers (Steam, Riot, Xbox Live, etc.), competing stat‑tracking apps, and relevant regulatory bodies such as the Dutch DPA (GDPR).

Communication:  
• The team tracks API changelogs, competitor releases, and privacylaw updates.  
• Significant developments are discussed at the next Wednesday meeting or, if urgent, flagged immediately in the WhatsApp group and followed up in Teams.

## Chapter 8: Planning

Additional elaboration concerning the planning will be done via the Gantt chart, which will be attached in addition to this document. The Gantt chart is consistent with the project activities stated in chapter 3.  For a more detailed and clearer view check the file name VGC Gantt in the documents folder.



## Chapter 9: Risk Analysis

The following risk analysis is based on the model from Grit (2021) and adapted to fit the GamerCV project. Since the project has no external clients or budget, the focus is on internal project risks, technical dependencies, and external events.

## Internal risks

|  |  |  |
| --- | --- | --- |
| Risk | Description | Mitigation Strategy |
| Inadequate planning | Poor mapping of features and tasks can cause delay. | Create a clear feature list and technical design document early in development. |
| Poor team coordination | Miscommunication or unclear roles can cause duplicated or missing work. | Use GitLab actively; Assign roles clearly; Conduct regular syncs, stand-ups and meetings; |
| Feature creep | Adding extra features (e.g. more games, UI flair) may delay core development and the whole premise of the project. | Lock MVP scope early. Track extras as ‘future features’ |

## Technical risks

|  |  |  |
| --- | --- | --- |
| Risk | Description | Mitigation Strategy |
| OpenAPI limitations | APIs might have rate limits or missing data. | Test endpoints early. Mock responses. Use retry logic. |
| Inconsistent data models | Games expose data in different structures. | Normalize API data using mappers. |
| Mobile UI complexity | Stat-heavy screens might not fit well on mobile devices. | Use wireframes. Prioritize clarity and minimal design. |

## External risks

|  |  |  |
| --- | --- | --- |
| Risk | Description | Mitigation Strategy |
| API deprecation/downtime | Public APIs may be disabled or change format without notice. | Document all endpoints. Add fallback logic. Cache key stats locally. |
| Team availability conflicts | Personals conflicts like exams or illness can reduce team capacity. | Set up flexible workload distribution. Conduct weekly stand-ups and have clear communication. |
| Tooling failure | Dependency errors or broken dev environments might block progress. | Use Git and commit regularly. Keep setup guides updated. |

## Risk assessment criteria

A risk matrix has been created to help classify and prioritize the risks in a visual manner.

A graph with text on it

AI-generated content may be incorrect.

*Figure x. Risk assessment matrix*