# ExamRush – Project Booklet

Authors: Georgi Dimitri, Alessio Perrone

Academic Year: 2024/2025  
Course: Advanced Programming

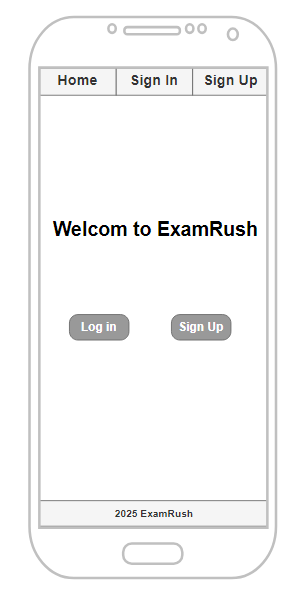
## 1. Project Overview

ExamRush is a web-based quiz application designed to help students practice and enhance their knowledge through interactive quizzes. It supports both teachers and students, offering functionalities like quiz creation, deck management, account registration/login, score tracking, and gameplay with timing.

The system supports two main roles:  
- Students: Can register, play quizzes, view performance stats.  
- Teachers: Can create decks, upload questions, and manage their teaching materials.  
  
The project includes features like:  
- JWT-based login system  
- Interactive quiz gameplay (with shaking gesture support)  
- Responsive frontend for mobile usage  
- Teacher/student profile management  
- RESTful backend API

## 2. Application Mockup

Home

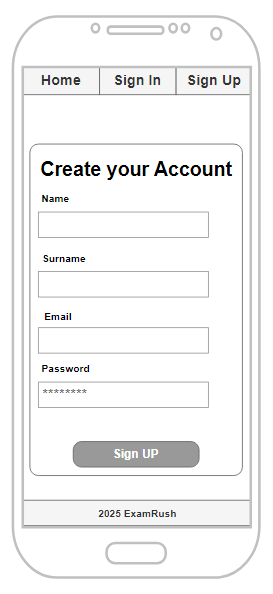


Login

A screen shot of a login screen

AI-generated content may be incorrect.

Registrati



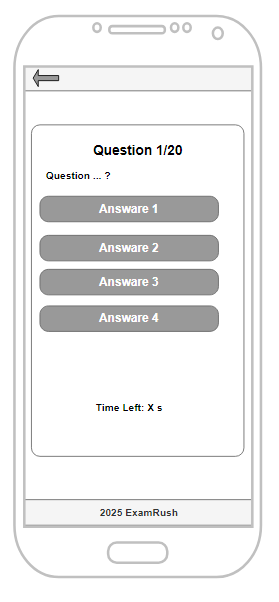
Deck

A white rectangular object with black text

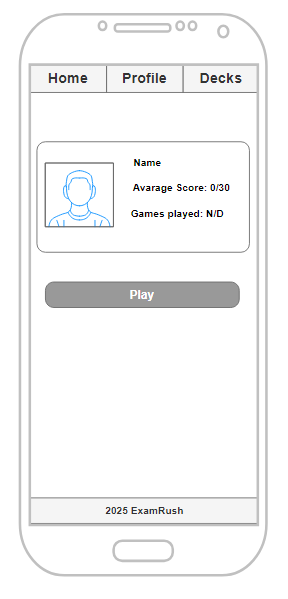
AI-generated content may be incorrect.A white rectangular object with a plus and a cross

AI-generated content may be incorrect.

QuestionsDeck



Login

A screenshot of a card

AI-generated content may be incorrect.

StudentPage

A screenshot of a phone

AI-generated content may be incorrect.A screenshot of a phone

AI-generated content may be incorrect.

## 3. Scrum Process

The ExamRush development was carried out using the Scrum framework. The team conducted several sprints, each lasting one week. Each sprint focused on specific deliverables with clear user stories, acceptance criteria, and regular retrospectives.

### Team Members and Roles

- Georgi Dimitri: Frontend developer, responsible for UI/UX and gameplay implementation.  
- Alessio Perrone: Backend developer and DevOps, responsible for authentication, APIs, and deployment.

**Sprint 1: Project Setup & Authentication**

Sprint Duration: 2 week

Goals: Initialize the project, configure authentication system, and create user model.

|  |  |
| --- | --- |
| **User Story** | **Task** |
| US1 | As a new user, I want to register an account using my email and password so that I can access the application. |
| US2 | As a user, I want to log in using my credentials so that I can resume using the application. |
| US3 | As a user, I want to upload and edit my profile picture so that I can personalize my account. |

**Sprint 2: Profile Management & Deck System**

Sprint Duration: 1 week

Goals: Add cartoon filter and animations for enhanced user interaction.

|  |  |
| --- | --- |
| **User Story** | **Task** |
| US4 | As a user, I want to apply a cartoon filter to my profile picture so that it looks fun and unique. |
| US5 | As a user, I want an interactive and animated user interface so that the application is engaging and easy to use. |

### Sprint 3: Quiz Gameplay & Interaction

Sprint Duration: 2 week

Goals: Allow students to browse, search, and interact with quizzes.

|  |  |
| --- | --- |
| **User Story** | **Task** |
| US6 | As a student, I want to browse and select from multiple question decks so that I can choose a topic to study. |
| US7 | As a student, I want to answer multiple-choice questions interactively so that I can test my knowledge. |
| US8 | As a student, I want to search for decks using a search bar so that I can quickly find decks by topic, subject, or keyword. |

### Sprint 4: UI Enhancements & Finalization

Sprint Duration: 2 week

Goals: Allow teachers to create, upload, and test question decks, and students to track statistics and performance data.

|  |  |
| --- | --- |
| **User Story** | **Task** |
| US9 | As a student, I want to see statistics about my performance (e.g., average score, strengths, weaknesses, progress over time) so that I can track my knowledge and identify areas for improvement. |
| US10 | As a teacher, I want to create and upload multiple-choice question decks so that my students can use them for studying. |
| US11 | As a teacher, I want to test the decks I upload by answering the questions myself so that I can ensure they are accurate and functional. |

## 4. Project Analysis

The following ER diagram visually represents the relationships between the key entities in the system, such as users, profiles, decks, questions, and performance statistics. This diagram serves as a blueprint for the database schema, ensuring that all user stories and functionalities are supported efficiently.

A diagram of a computer

AI-generated content may be incorrect.

## 5. Function Point Analysis

The Function Point Analysis was conducted by identifying all the relevant components within the ExamRush system, including internal and external files, inputs, outputs, and inquiries.

## Internal Logical Files (ILF)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Type | Functionality | RET | DET | Fields | Complexity | FPs |
| Internal Logical File (ILF) | Users | 1 | 6 | user\_id, email, name, surname, score, average\_score, played\_games, image | Average | 10 |
| Internal Logical File (ILF) | Decks | 1 | 4 | title, description, teacher\_id, cards | Average | 10 |
| Internal Logical File (ILF) | Cards | 1 | 6 | card\_id, question, answer, options, correct\_option, deck\_id | Complex | 15 |

Explanation:  
- RET: Record Element Types - Number of logical subgroups of data.  
- DET: Data Element Types - Number of individual fields.  
- Complexity: Derived using the ILF Complexity Matrix.  
- FPs: Function Points based on complexity.

## ILF Complexity Matrix Reference

|  |  |  |  |
| --- | --- | --- | --- |
| Number of RETs | 1-19 DETs | 20-50 DETs | 51+ DETs |
| 1 RET | Low | Low | Average |
| 2-5 RETs | Low | Average | High |
| 6+ RETs | Average | High | High |

## External Interface Files (EIF)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Type | Functionality | RET | DET | Fields | Complexity | FPs |
| External Interface File (EIF) | (None) | - | - | - | - | - |

Note: No external data sources are referenced, hence no EIFs.

## External Inputs (EI)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Type | Functionality | RET | DET | Fields | Complexity | FPs |
| External Input (EI) | Register User | 1 | 6 | email, name, surname, score, average\_score, played\_games | Average | 4 |
| External Input (EI) | Update User Stats | 1 | 3 | played\_games, score, average\_score | Low | 3 |
| External Input (EI) | Create Deck | 1 | 4 | title, description, teacher\_id, cards | Average | 4 |
| External Input (EI) | Upload Profile Image | 1 | 1 | image | Low | 3 |

## External Outputs (EO)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Type | Functionality | RET | DET | Fields | Complexity | FPs |
| External Output (EO) | Get User Info | 1 | 6 | user\_id, email, name, surname, score, average\_score, played\_games, image | Average | 5 |
| External Output (EO) | Get Deck Info | 1 | 4 | title, description, teacher\_id, cards | Average | 5 |
| External Output (EO) | Get Profile Image | 1 | 1 | image | Low | 4 |

## External Inquiries (EQ)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Type | Functionality | RET | DET | Fields | Complexity | FPs |
| External Inquiry (EQ) | Get User by Email | 1 | 6 | user\_id, email, name, surname, score, average\_score, played\_games, image | Average | 4 |
| External Inquiry (EQ) | Get Decks | 1 | 3 | title, description, teacher\_id | Low | 3 |

## Summary of Function Points (FPs)

|  |  |
| --- | --- |
| Component | Total FPs |
| ILF | 35 |
| EIF | 0 |
| EI | 14 |
| EO | 14 |
| EQ | 7 |
| Total | 70 |

## 5. Effort Estimation – COCOMO II

To estimate the development effort for ExamRush, we applied the COCOMO II Post-Architecture model. It relies on Source Lines of Code (SLOC) and factors in scale and cost drivers.

### SLOC Estimation

Based on the Function Point total (70 FP) and using an average value of 53 SLOC per FP for JavaScript/Node/React systems, the estimated SLOC is calculated as: 70 × 53 = 3710.

Estimated SLOC: 3710

### COCOMO II Parameters

|  |  |
| --- | --- |
| Estimated SLOC | 3710 |
| Effort (PM) | 10.56 Person-Months |
| Development Time (TDEV) | 7.28 Months |
| Effort Adjustment Factor (EAF) | 1.0 (Nominal) |

According to the COCOMO II model, the estimated effort required to develop ExamRush is 10.56 Person-Months, with a total development time of approximately 7.28 months.

In fact, more in details:

### 5.1 COCOMO II Formula

### Scale Factors

The scale factors (SF\_i) influence the exponent E in the COCOMO II model. Each factor is rated as Very Low, Low, Nominal, High, Very High, or Extra High. For this estimation, all scale factors are assumed to be at nominal level, resulting in the following:

|  |  |
| --- | --- |
| Scale Factor | Rating (Nominal) |
| Precedentedness (PREC) | Nominal |
| Development Flexibility (FLEX) | Nominal |
| Architecture / Risk Resolution (RESL) | Nominal |
| Team Cohesion (TEAM) | Nominal |
| Process Maturity (PMAT) | Nominal |

Sum of scale factors:



### Cost Drivers

Cost drivers affect the Effort Adjustment Factor (EAF). In this estimation, all cost drivers are assumed to be nominal:

|  |  |
| --- | --- |
| Cost Driver | Rating (Nominal) |
| Required Software Reliability (RELY) | Nominal |
| Database Size (DATA) | Nominal |
| Product Complexity (CPLX) | Nominal |
| Developer Capability (DCAP) | Nominal |
| Personnel Continuity (PCON) | Nominal |
| Platform Experience (PLEX) | Nominal |
| Use of Software Tools (TOOL) | Nominal |
| Multisite Development (SITE) | Nominal |

Effort Adjustment Factor (EAF) = 1.00

### Final Estimation

* **Effort (PM)**

**Immagine che contiene testo, Carattere, bianco, tipografia

Il contenuto generato dall'IA potrebbe non essere corretto.**

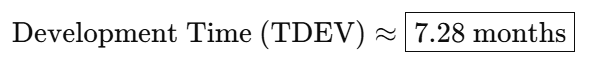
****

****

* **TDEV**

**Immagine che contiene testo, Carattere, schermata, bianco

Il contenuto generato dall'IA potrebbe non essere corretto.**

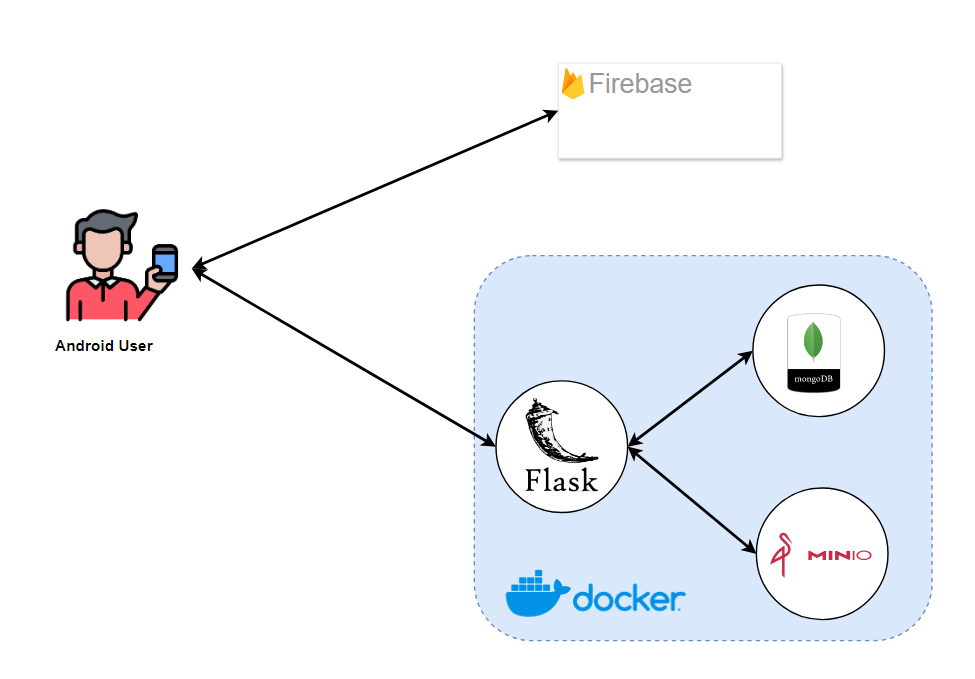
****

## 6. System Architecture

The ExamRush system follows a layered client-server architecture, consisting of three main components:

- Frontend (React Native): Provides the mobile interface for both students and teachers.  
- Backend (Node.js/Express): Handles authentication, API logic, game sessions, and statistics.  
- Database (MongoDB): Stores user profiles, decks, questions, and scores.

The following diagram illustrates the high-level architecture of the system:

  
Teachers and students interact with the system through a shared mobile interface. Authentication tokens (JWT) are used to manage access and role-based functionalities.

## 7. Conclusion

This document presented the ExamRush application, outlining its development through Agile Scrum methodology, Function Point and COCOMO II effort estimation, and a detailed architectural overview.  
  
The application offers an engaging and educational platform where users can challenge themselves with quizzes while teachers manage and create content easily.  
  
The use of modern technologies and agile practices ensured iterative progress, fast feedback, and a clean, scalable codebase.